Financial Econometrics

Module Introduction and Overview

Contents

1 Introduction to the Module

Welcome to the module on Financial Econometrics. The first objective of this module is to introduce the main econometric methods and techniques used in the analysis of issues related to finance. A module with the title *Financial Econometrics* assumes that such a field exists. However, as this quote reveals, this is far from true:

> What is … financial econometrics? This simple question does not have a simple answer. The boundary of such an interdisciplinary area is always moot and any attempt to give a formal definition is unlikely to be successful. Broadly speaking, financial econometrics [aims] to study quantitative problems arising from finance. It uses statistical techniques and economic theory to address a variety of problems from finance. These include building financial models, estimation and inferences of financial models, volatility estimation, risk management, testing financial economics theory, capital asset pricing, derivative pricing, portfolio allocation, risk-adjusted returns, simulating financial systems, hedging strategies, among others (Fan, 2004: 1).

In this module we define financial econometrics as 'the application of statistical techniques to problems in finance'. Although econometrics is often associated with analysing economics problems such as economic growth, consumption and investment, the applications in the areas of finance have grown rapidly in the last few decades. Your key text by Chris Brooks, *Introductory Econometrics for Finance*, lists the following examples:

1. Testing whether financial markets are weak-form informationally efficient.
2. Testing whether the capital asset pricing model or arbitrage pricing theory represent superior models for the determination of returns on risky assets.
3. Measuring and forecasting the volatility of bond returns.
4. Explaining the determinants of bond credit ratings used by the ratings agencies.
5. Modelling long-term relationships between prices and exchange rates.
6. Determining the optimal hedge ratio for a spot position in oil.
7. Testing technical trading rules to determine which makes the most money.
8. Testing the hypothesis that earnings or dividend announcements have no effect on stock prices.
9. Testing whether spot or futures markets react more rapidly to news.
10. Forecasting the correlation between the returns to the stock indices of two countries.

The above list does not include all the possibilities, and you might think of many other topics that could be added to the list.

If financial econometrics is simply the application of econometrics to finance issues, does this mean that econometric tools you have studied in previous courses are the same as those used in this module? A simple answer to this question is *yes*. Many of the concepts that you have encountered in your other studies such as regression and hypothesis testing are highly relevant for this module. In fact, all the topics introduced in this module will require that you have a deep understanding of these concepts. However, the emphasis and the set of problems dealt with in finance issues are different from the economic problems you have encountered in previous courses. To start with, the nature of the data in finance issues is very different. Financial data are observed at a much higher frequency (in some instances minute-by-minute frequency). For macroeconomic data, we consider ourselves lucky if we are able to observe data on a monthly basis. Furthermore, recorded financial data such as stock market prices are those at which the transaction took place. There is no possibility for measurement error. This is in contrast to macroeconomic data, which are revised regularly.

Also the properties of financial series differ. For instance, the module *Econometric Analysis and Applications* analyses whether a series has a unit root, and devises methods to estimate models when the variables are integrated of order one. In financial econometrics these issues are not a major concern. Although we observe prices most of the time, financial econometrics mainly deals with asset or portfolio returns. Since returns are stationary, most of the methods used in this module also apply to stationary series.

This may imply that models of financial returns are much easier to deal with. However, this is not the case. The analysis of financial data brings its own challenges. As you will see in Unit 1, financial returns possess some common properties that need to be incorporated in econometric models. For instance, returns of assets such as stocks and bonds exhibit time-varying volatility. This requires introducing new models and estimation techniques to model time varying volatility. Not only that, financial returns can exhibit asymmetry in volatility, which requires further modification of existing models. Furthermore, financial data are not normally distributed. As you have seen in your previous studies, the assumption of normality has been central for estimation and hypothesis testing. Unfortunately, even in finance applications, existing econometric techniques still find it difficult to deal with models that assume non-normal distribution.

2 The Module Authors

Bassam Fattouh graduated in Economics from the American University of Beirut in 1995. Following this, he obtained his Masters degree and PhD from the School of Oriental and African Studies, University of London, in 1999. He is a Professor in Finance and Management and Academic Director for the MSc in International Management for the Middle East and North Africa at the Department for Financial and Management Studies, SOAS. He is also currently Senior Research Fellow and Director of the Oil and Middle East

Programme at the Oxford Institute for Energy Studies at the University of Oxford. He has published in leading economic journals, including the *Journal of Development Economics, Economics Letters, Economic Inquiry, Macroeconomic Dynamics* and *Empirical Economics*. His research interests are mainly in the areas of finance and growth, capital structure and applied non-linear econometric modelling, as well as oil pricing systems.

Jonathan Simms is a tutor for CeFiMS, and has taught at University of Manchester, University of Durham and University of London. Dr Simms has contributed to development of various CeFiMS modules including *Econometric Principles & Data Analysis*; *Econometric Analysis & Applications*; *Risk Management: Principles & Applications*; *Financial Engineering*; *Introduction to Valuation*; *Introduction to Law and to Finance*; *Public Financial Management: Reporting and Audit*; *Corporate and Investment Banking*; and *Banking Strategy*.

3 Study Resources

This module mainly uses one key text:

> Chris Brooks (2019) *Introductory Econometrics for Finance*. 4th Edition. Cambridge UK: Cambridge University Press.

This text has been chosen because it is extremely clear, it contains a large number of examples and covers a lot of different topics. It is a useful text to refresh your memory of some basic concepts studied in previous courses (especially Chapters 2, 3 and 4). The text has a very useful companion website with rich resources for students including practice questions, solutions to end of chapter questions, data in Excel files, and R codes. The link for the companion resources can be found at:

https://www.cambridge.org/gb/academic/subjects/economics/finance/introductory-econometrics-finance-4th-edition?format=PB

Although the text covers a lot of subject areas, in some units you may need to rely more heavily on the study guide and suggested readings.

The units in the module will closely follow the presentation in the key text. However, for some of the units, this is not feasible either because the chapter does not cover the topic at all, or covers it in a superficial way. In such cases you may find that the study guide is more demanding than the material presented in the key text, because the study guide analyses the issues using mathematics (though at a relatively basic level). This is necessary to gain a deeper understanding of the issues being considered.

Throughout this module it is essential that you do all the readings and solve all the exercises. In this module each idea builds on the previous ones in a logical fashion, and it is important that each idea is clear to you before you move on. You should therefore take special care not to fall behind with your schedule of studies.

Software

R

This module will use R. This is a widely used programming environment for data analysis and graphics. You will use this software to do the exercises in the units. The results presented in the units are also from R.

R is free software, released under the GNU General Public License (R Core Team, 2019). Instructions for downloading R, and a brief introduction on how to use it, are provided below.

The best advice on using R is to stay focused on the subject that is being studied in each unit, and to do the exercises for the unit; this will reinforce your understanding and also develop your confidence in using data and R.

The units include all of the R commands that are required to complete the examples and exercises. However, as you become more confident in using R, or if you are already familiar with R, you may find that you develop your own R commands.

4 Module Overview

The objective of the module is to extend your knowledge and equip you with methods and techniques that allow you to analyse finance-related issues.

This module starts by illustrating how to measure financial returns, the main variable that we try to model in financial applications. There are various definitions of returns, and Unit 1 illustrates how to compute the various types of returns. After defining financial returns, Unit 1 presents some stylised facts about the properties of financial returns. These include volatility clustering, asymmetric volatility and non-normality. Unit 1 then introduces various measures of moments of the distribution of financial returns, and how these can be computed for samples of financial returns. The material covered in this unit sets the scene for the rest of the module and thus it is important that you make yourself familiar with these concepts.

Unit 2 provides a brief introduction to the main principles of matrix algebra. The modules *Econometric Principles and Data Analysis* and *Econometric Analysis and Applications* develop the basic regression concepts and statistical tools without referring to matrix algebra. This is essential to develop understanding of the basic concepts involved in regression analysis. However, in most theoretical and practical applications the researcher often deals with multivariate relations. As you will discover in this unit, the simplest way to tackle these multivariate relations is to switch to matrix notation. Matrix algebra eliminates the need to use summation signs and subscripts and helps present the results in a simple way. In some of the units of this module, it will be very difficult to present the proofs and results without using matrix notation. Although matrix notation simplifies the presentation of the results, the fact remains that you may be learning a new language. Learning a new language can be exciting but it is also challenging. To help you to understand and apply matrix algebra, we use matrix algebra in some financial applications, namely the multi-factor models and portfolio theory.

Unit 3 provides a brief review of the maximum likelihood estimation method. In your previous study of econometrics, it is probable that the least squares (LS) method was used to derive the estimates of the model's parameters and for hypothesis testing. Least squares is just one of many estimation techniques available for econometricians. In Units 4, 5, 6 and 8 of this module, you will encounter models such as GARCH, ARMA and binary choice models that can't be estimated by least squares. Instead, econometricians rely on maximum likelihood estimation, which is a flexible technique, more general than OLS and, under fairly general conditions, yields consistent and efficient estimates of the parameters. However, like any estimation technique, maximum likelihood is based on a particular underlying philosophy and principles. In Unit 3 you will be introduced to these principles and how these can be applied to derive estimates of the parameters and test hypotheses about the estimated parameters. This is one of the most challenging units, but hopefully, by using various examples, you will be able to gain a deep understanding of how the estimation method works, and you will be able to identify its strengths and weaknesses.

Unit 4 presents univariate time series models. In these types of models, a series is modelled in terms of its own past values and some disturbance terms (also known as shocks). Univariate time series models were introduced in the module *Econometric Analysis and Applications*. These models are different from the structural models you have studied in other courses, in the sense that these models are *atheoretical* – that is, they are not based on any underlying theoretical frameworks but are data driven. These models are the first building blocks for estimating financial returns and help illustrate some of the key properties of financial returns. The aim of this unit is to introduce these models, such as the autoregressive model (AR), the moving average model (MA), and a combination of these two (ARMA models).

Unit 5 presents some of the econometric methods used for modelling and forecasting volatility of asset returns. Volatility models have attracted the attention of academics and practitioners, and are widely used in many areas of finance, including models of value-at-risk, option pricing, and portfolio allocation. One of the stylised facts about asset returns is that the variance of the error terms is not equal at every point in time, and hence the error terms are said to suffer from heteroscedasticity. Thus, in modelling financial returns, one should consider approaches that relax the assumption of homoscedasticity. ARCH (autoregressive conditional heteroscedastic) and GARCH (generalised autoregressive conditional heteroscedastic) models do exactly that. They relax the assumption of constant variance and exploit the heteroscedasticity feature to model the variance of returns over time. As you will study in this unit, GARCH models are also flexible enough to allow us to incorporate asymmetry in the volatility of financial asset returns.

Unit 6 extends the GARCH model from the univariate to the multivariate setting. This proves to be an important extension because it allows researchers and financial analysts to model time-varying conditional covariance and correlation between the returns of various financial assets. This technique opens the way for many financial applications such as dynamic asset pricing models, portfolio selection, dynamic hedging, value-at-risk, and volatility transmission between assets and markets. Multivariate GARCH models also help researchers to model some of the features of asset returns, such as correlation clustering.

Unit 7 presents vector autoregressive (VAR) models, which can be thought of as generalisations to the univariate time series models. VAR models represent an improvement over univariate time series models because they allow variables to be modelled not only in terms of their own lags and their own shocks, but also in terms of the lags of other variables. This provides greater flexibility and allows us to examine the dynamic interactions between a set of variables. VAR models have become very popular in the econometrics literature and are widely used in the areas of macroeconomics and finance. The tools which have developed around VAR, such as impulse response analysis, Granger causality and variance decompositions (all discussed in this unit) have become central to understanding the interaction among variables. VAR models have also been used extensively for forecasting purposes, where these models have exhibited a better performance than structural models, especially in out-of-sample forecasting.

Unit 8 deals with models in which the dependent variable *ie* the variable that needs to be explained by a set of determinants, is in fact a dummy variable. There are many cases where these models can be useful. For instance, financial analysts may be interested as to why some firms list on the stock market while others don't; why some firms issue dividends while others don't; and why some firms decide to raise external finance while others don't. In all these examples, what we observe is whether a firm lists or not, issues dividends or not, or raises external finance or not. Thus, the relevant dependent variable is a dummy variable that takes the value of 1 if the event occurs, and zero if the event does not occur. Such models, known as limited dependent variable models, raise a set of estimation issues that are different from the ones you have encountered so far. The purpose of this unit is to introduce you to limited dependent models and discuss how these models can be applied to finance issues.

5 Learning Outcomes

After you complete your study of this module, you will be able to:

- define and compute measures of financial returns
- discuss the stylised statistical properties of asset returns
- formulate models using matrix notation
- derive the OLS estimators using matrix algebra
- explain the principles of maximum likelihood estimation
- derive the maximum likelihood estimators and discuss their properties
- use maximum likelihood estimation, and apply the hypothesis tests available under maximum likelihood estimation
- analyse, estimate and forecast using autoregressive (AR), moving average (MA), and autoregressive-moving average (ARMA) models
- apply the Box–Jenkins approach to time series models
- model and forecast volatility using autoregressive conditional heteroscedastic (ARCH) models
- estimate, interpret and forecast with generalised autoregressive conditional heteroscedastic (GARCH) models
- construct, estimate and interpret multivariate GARCH models
- test for spill-over of volatility between assets
- use vector autoregressive (VAR) models to analyse and interpret interaction between financial variables, including impulse response analysis
- undertake tests of hypotheses and Granger causality in a VAR framework
- formulate, estimate and interpret limited dependent variable models, including logit and probit models
- discuss models with multinomial linear dependent variables.

6 R

R is an implementation of the object-oriented mathematical programming language S. It is developed by statisticians around the world and is free software, released under the GNU General Public License. Syntactically and functionally it is very similar (if not identical) to S+, the popular statistics package.

R is much more flexible than most software used by econometricians because it is a modern mathematical programming language, not just a program that does regressions and tests. The S language is the de facto standard for statistical science. Since most users have a statistical background, the jargon used by R experts sometimes differs from what an econometrician (especially a beginning econometrician) may expect. Code written for R can be run on many computational platforms with or without a graphical user interface, and R comes standard with some of the most flexible and powerful graphics routines available anywhere. And of course, R is completely free for any use.

(extracted from Farnsworth, 2008)

Because the R software is a programming language and not just an econometrics program, some of the functions we will be interested in are available through libraries (sometimes called packages) obtained from the R website http://www.r-project.org/.

The data files required are available on the Virtual Learning Environment (VLE).

Starting up

The R software must be installed on your system. If it is not, follow the installation instructions appropriate to the operating system (OS). Installation is especially straightforward for Windows users.

To install R

Go to the website of R http://www.r-project.org/.

There, choose your preferred CRAN mirror (*eg* http://cran.ma.imperial.ac.uk/) and click on the link referring to your OS in the box "Download and Install R". Note that these units are written with the version "Windows", but Mac OS and Linux versions are also downloadable. (CRAN mirrors are provided in locations around the world to reduce download time and internet traffic.)

On the next page, click on "base" and you will be redirected to a page where there will be a link to download the latest version of R. At the top will be written something like "Download R 3.6.0 for Windows" (or whatever is the most up-to-date version). Click on this link. Updated versions of R are made available over time, but the commands explained in the module files should be similar.

Save the file "R-3.6.0-win.exe" (or whatever is the latest version of the file) in any location, for example on your Desktop. Double click on this file to start the installation, and follow the instructions. You will be asked to choose the place where you want the file to be, and you can also choose to have a shortcut to R in the Start Menu and a Desktop icon. Choose the standard installation.

To get started with R

To open R, double click on the desktop icon or on the shortcut. The Command Window opens.

To use R, you can either type a command in the Command Window, or run a pre-written program.

1. To type a command in the command window

   You just type it on the line starting with **>** and press the key Enter on your keyboard. Note that R is case sensitive. In the unit file, if it says "type in" or "can be performed with", this means "Type in the following command in the command window and then press Enter".

   The unit files include all of the R commands required to work on the examples and exercises. To save time, you can copy and paste the required coding from the unit files into R. Note that if multiple lines of code are pasted into R in one go, they will be executed in sequence immediately, without the need to press Enter.
2. Every time you see the sign **>**, it means that R is ready for a new command. When you do not see it or when you see the sign **+**, it means that R is still working on a previous command or is stuck because your command has not been properly written (for example, a bracket may be missing at the end of the previous command). In this case, press the red button STOP in the toolbar to stop the process.
3. To run a pre-written program

   In the Menu bar go to File, then choose "New script". A new Editor Script Window opens.

   You can type all commands you wish in this file, one command per line. To run them on R, select with the mouse the commands you wish to run, and then simultaneously press the keys CTRL and the key R (CTRL + R) (or alternatively go to Edit in the Menu bar and choose "Run line or selection").

   Save your program by choosing "Save as" in File in the Menu bar (the Menu bar specific to the Editor, that is to say, the one that is available when you are in the Editor Window, and not in the Command Window). All script files have the extension ".R". Then you can close it and open it from the File menu in the Menu bar.

   In your program file you can write things that are not commands (for example, notes or explanations); but to indicate to R that these are not commands, put the symbol # in front of each non-command line.

To get help at any time, go to Help in the Menu bar. Alternatively, there are several methods of obtaining help in R. You may type in alternatively the following commands:

```
?qt
help(qt)
help.start()
help.search("covariance")
```

The last command is for a search on the term covariance, for example. Please note the use of the quotation marks " ". If you choose to develop R commands in Word, for example, and then copy the commands to R, please make sure you use the " " symbols, and not the Word symbols “ ”.

Preceding a command with a question mark, or writing a command as an argument in help(), gives a description of its usage and functionality. The help.start() function brings up a menu of help options, and help.search() searches the help files for the word or phrase given as an argument. Many times, though, the best help available can be found by a search online. The help tools above only search through the R functions that belong to packages already installed on your computer. However, often users have the following type of question: “Does R have a function to do …”. Users do not know if functionality exists because the corresponding package is not installed on their computer, and therefore resort to the R website. To search the R website for functions and references, use:

```
RSiteSearch("scatter plot")
```

This command example searches the R website for ‘scatter plot’ functions.

The results from the search should appear in your web browser.

By default, R has a couple of excellent free manuals in PDF format. If you are not already familiar with using R you are advised to read *An Introduction to R*. To access the manual, click Help | Manuals and the list of available documents will be shown.

To quit R, in the Menu bar go to File, then choose “Exit”. You are asked “Save workspace image”? Click on No.

In case R is installed in another language, to change the language of the Menu bar into English, in the Menu bar choose Edit (second from the left in the Menu bar) and then choose the last option “Preferences”. A new window opens. In the top right of it, there is an empty box for “Language for menus and messages”. Type in this box “English” and click OK. You will have to exit R and start it again.

You may want to specify a working directory for your R session, so that R chooses data files (and scripts) directly from this directory; then there is no need to specify the full path to access the files. The command getwd() returns an absolute filename representing the current working directory of the R process. The command setwd() helps you specify the working directory. For example, to set the working directory to “C:\Documents and Settings\my files\R SOAS”, you would type in:

```
setwd("C:/Documents and Settings/my files/R SOAS")
```

This avoids the need to put the command scripts and the data files within the R directory on your computer. **Note that, in this command, the slash / is used**.

Alternatively, you can set the working directory using File (on the Menu bar), then Change dir… and then browse through the folders on your device to select the folder you want to use as the working directory for that session.

To save your work, go to File | Save Workspace… Provide a filename for the workspace, and it will be saved with the extension .RData. The file will be saved in your current working directory, but you can browse to another location if you prefer.

A saved workspace can be loaded with File | Load Workspace… R will display files with the .Rdata extension in the current working directory, but again, you can browse to other locations

Every time you start R, before running any command or program, it is best to clean up any object that may remain in memory.

During an R session, objects are created and stored by name. The command objects(), or the command ls(), can be used to display the names of the objects which are currently stored within R. The command rm() removes a data object whose name should be given as an argument.

To remove all objects, type in:

```
        rm(list=ls())
```

Alternatively, use Misc | Remove all objects.

Removing all objects is useful if, for example, you work on one exercise and dataset, then you want to work on another data set and exercise in the same R session. If you do not remove the first set of objects before starting on the second question, your workspace will contain objects you created for both questions, which could be confusing. Remember to save a workspace before removing all the objects!

Installing and loading packages

The basic setup for R provides considerable statistical and mathematical functionality.

However, you will also need to use some of the more specialised econometric functions that have been written for R, and made available in packages. The module units will indicate when you need to use a particular function, and the unit will also indicate which package is required.

To use a function from a package, the relevant package must be installed on your machine. To install a package go to Packages | Install package(s)… You will be asked to select a CRAN mirror for this session. Once the mirror has been selected, you can choose from the list of available packages to install. Some of the packages (for example dynlm) take a few minutes to install, but installation of a package is a one-off procedure, and once the package is installed, you will not need to install it again.

To use a package in an R session, *the package must be loaded in that session*. Loading a package takes considerably less time than installing it. To load a package, go to Packages | Load package… You will see a list of packages that you have previously installed, and you can select the package you want to load.

To summarise, to access a package it must first be installed on your device. And to use the package in an R session, the package must be loaded in that session.

If you try to use a function but you have not loaded the relevant package in your R session, you will get an error message saying R could not find the function.

Reading data from text files and creating zoo objects

The data for the examples and exercises in the module units is provided in tab-delimited text files. R can read data from files in other formats, but tab-delimited text files are very simple, they can be created and opened in many applications, and they are robust to updates.

To work on the data in R, it must first be read from the text file into what is called a data frame, using a read.table command. A data frame is a matrix or table that contains the data in R. Once the data is in a data frame, it is then possible to use R to perform various operations, such as plotting the data, and estimating models.

However, many models in finance specifically focus on the time element of the observations. Furthermore, financial data can have irregular dates. For example, if you are using daily data, there may be no observations for weekends and bank holidays. For these reasons we have mainly used 'zoo' objects to handle the data in R, rather than data tables or regularly dated time series objects. The zoo package provides an infrastructure for regular and irregular time series (Z's Ordered Observations – hence 'zoo'), and is maintained by Achim Zeileis (Zeileis and Grothendieck, 2005). The zoo objects are indexed by the dates of the observations, which allows us to perform operations that specifically relate to the timing of the data.

The commands for reading the data from the text files and creating the zoo objects are provided in the units. These commands assume that the data text files are located in the current working directory.

R outputs

The outputs that you obtain can be viewed in R, and can also be copied to word processing applications. The tables of outputs in the module units were obtained in R and formatted for production. Some of the formatted tables of output were produced with the stargazer package (Hlavac, 2018). Advice on how to use stargazer is also provided in the units, where relevant. However, please note that it is not possible to use stargazer on the outputs of all of the packages used in this module.

References

Brooks C (2019) *Introductory Econometrics for Finance.* 4th Edition. Cambridge: Cambridge University Press.

Fan J (2004) *An Introduction to Financial Econometrics*. Department of Operation Research and Financial Engineering, Princeton, NJ: Princeton University, November.

Farnsworth GV (2008) *Econometrics in R*. [Online]. October 26. Available from: http://cran.r-project.org [Accessed 30 April 2020]

Hlavac M (2018) *stargazer: Well-Formatted Regression and Summary Statistics Tables*. R package version 5.2.2. Available from: https://CRAN.R-project.org/package=stargazer [Accessed 30 April 2020]

R Core Team (2019) *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. Available from: https://www.R-project.org/ [Accessed 30 April 2020]

Zeileis A and G Grothendieck (2005) 'zoo: S3 Infrastructure for Regular and Irregular Time Series', *Journal of Statistical Software*, 14 (6), 1–27. Available from: http://doi.org/10.18637/jss.v014.i06 [Accessed 30 April 2020]