CHAPTER 1

INTRODUCTION

Introduction

This book is designed to provide a programming introduction for graduate students and other reasearchers of financial economics. We provide a strictly application-oriented introduction to using SAS to analyze a set of typical finance problems. For each of those problems, we provide a programming solution that uses a variety of features available in the SAS System. We discuss each program in detail and point out potential pitfalls. It is not our purpose to make a scientific contribution in finance, econometrics, or computer science. Rather, by providing a sound solution to each of our sample problems, we prepare readers to implement their own original ideas by extending our methodological groundwork. The book is designed such that by working through the chapters, readers assemble a toolkit of programming skills that enable them to find solutions to other and more complex empirical problems. We go beyond any other such book by focusing strictly on a selection of finance applications. This allows us to address specific issues that arise—for example, when merging return data with financial statement data, combining transaction-based trade prices with intraday bid-ask quotes, and then executing various hypothesis tests.

We also address several basic empirical methods that are frequently applicable in empirical financial research. We discuss various regression models, including OLS, logit, probit, and vector autoregressive models. We also show how to test for and correct for heteroscedasticity and autocorrelation. In addition, we program an event study and perform a variety of statistical tests. We also discuss discriminant analysis, variance ratio tests, and pooled cross-sectional and time-series models. In short, our programming examples cover a variety of approaches that should give the reader a firm starting point for several additional issues and for econometric extensions that go beyond the scope of this introductory book.

Although we briefly introduce almost every programming statement used in this text, we assume that the reader is familiar with the basic structure of the SAS language. Each SAS installation includes an in-depth online tutorial that is helpful in obtaining this basic working knowledge. We also assume that readers know how to access SAS from their platform of choice. SAS is largely platform independent; it is used in almost the same way whether on a PC, a mainframe, UNIX, or other system, and it provides a graphical user interface (GUI) or at least command menus on all platforms. We encourage every reader to use our sample programs and experiment with alternative programming statements, different options, and other data sets. Only continual practice will make statistical programming the highly efficient tool it was designed to be.

Generally, this book should be used in conjunction with SAS documentation. The online version (first available with Version 8) makes it easy to search for help and contains extensive documentation on the SAS language. SAS OnlineDoc contains a very detailed description of the econometrics and statistics associated with the SAS procedures. For this reason, we provide only brief references to methodological issues in this book. Similarly, the SAS Web site (www.sas.com) contains additional up-to-date technical reports and problem discussions by its staff and other users. Moreover, this Web site is probably the best place to search for specific methods or procedures that are not yet documented elsewhere. This book discusses only sample applications of certain language elements; for additional features, you should always consult SAS OnlineDoc.

Working with SAS

SAS software has a versatile display manager mode (driven by a GUI or by menus, depending on the operating system) that is helpful to access and explore data both interactively and in batch mode. The latter is generally more appropriate for more complex programming for data preparation and arrangement (and works most easily within the GUI). Because this is one of the major issues discussed in this book, we do not discuss interactive applications. Rather, we focus on the programming statements that are available in the DATA steps to manipulate, arrange, and analyze data. We also discuss PROC steps that are used to run preprogrammed routines (and may also have programming capabilities). To investigate a research question, most solutions involve combinations of several DATA and PROC steps. The DATA step and most procedures can be used to generate new data sets. These can then be used as input for the next set of SAS statements.

Each SAS program generates two types of reports: a *log* and an *output*. The SAS *log* contains the report of the execution of your program. The log is very important for verifying that programs have run error free, because this cannot always be seen from the output. The log contains several types of messages. For example, note messages report details of the execution, including the number of observations read from the source file and used in statistical procedures. Error messages indicate that the execution of the program was not completed because of errors in the program. Finally, warning messages indicate potentially ambiguous commands, which may or may not lead to incorrect or unexpected computations.

The *output* file contains the results of the program, such as estimation results or the printed contents of data sets.

Ground Rules

In SAS, each command must end with a semicolon, and each program must end with a RUN statement (a RUN is not necessary when more DATA or PROC steps follow—both imply a RUN statement). SAS is not sensitive to additional blanks, additional semicolons, or additional RUN statements. This relatively free format makes it easy to structure program files in ways that are straightforward to read. Comments may appear anywhere in a program and are enclosed between "*" and ";" or within "/*" and "*/".

After a SAS program is submitted for execution, the individual steps are executed sequentially: each step executes as soon as SAS either finds the beginning of the next DATA or PROC step or finds a RUN statement. In case of an error, the SAS log will most often display the exact location and reason for aborting the step (most syntax errors are already highlighted in red when you type the program using the SAS Program Editor). The most important skill to learn is to find programming errors based on the system's error messages. Because an error in one step will not necessarily prevent execution of subsequent steps, it may cause a flurry of errors in later steps even when their syntax is correct. This implies that to debug a program, it is most efficient to begin the search for errors at the beginning of the program, which is listed sequentially in the SAS log file. Correcting one error and running the program again may indeed resolve several error messages simultaneously.

One of the most important ground rules in empirical research is to always examine all intermediate results. For example, when a new data set is generated, it should be inspected before it is used for further computations. The inspection should be done visually and by computing descriptive statistics and, especially, by listing extreme values. Checking intermediate (and, of course, final) results is extremely important, because several programming errors or omissions follow the correct syntax and will not cause an error message to be generated. For example, several commonly used financial return databases code missing or dubious values as –99. This is reasonable because a financial return is, by definition, always larger than –1.0 and we can easily separate the error indicators from correct returns. Several colleagues tell tales of top-level publications in which someone forgot to check for these –99s and (puzzlingly) highly significant negative results were found. The only way to avoid this is to investigate any new data set very carefully.

SAS Data Sets

SAS has its own data storage system, the SAS data set, which all SAS procedures require as input. Data sets are organized in tabular form, where each record contains one observation and each column contains a field (variable). Large data sets can be associated with indexes for faster access. Data sets are ODBC (open database connectivity) compatible, so they can be accessed from and write to several popular database and spreadsheet programs. In a PC environment, for example, Microsoft Access tables can be saved directly as SAS data sets through ODBC. From inside SAS, each data set can be exported to formats such as Dbase or

Microsoft Excel (similar procedures are available for other platforms). Furthermore, the procedure CPORT is available if data sets have to be transferred between systems running SAS on different platforms. To keep the programming examples in this book platform independent, we will assume throughout that data files are in ASCII (text) format, although most users will store their data in database or spreadsheet programs. In general, one would use either menu commands or PROC IMPORT and PROC EXPORT to read and write non-SAS file formats.

Conventions Used in This Book

Each chapter is a self-contained discussion of a specific empirical issue and the program used to address it. Most contain a brief discussion of the underlying finance issues to put the sample analysis into perspective. At the beginning of each chapter, we list the major finance concepts that will be discussed and the type of data employed. Within the text, SAS commands and keywords appear in ALL CAPS. Variable names appear in SMALL CAPS.

The programs we discuss in each chapter are found at the end of that chapter. For easier reading, we divide each program into sections that we discuss individually within the chapter. Both the complete program listing and the individual sections can be used to practice and experiment with variations of the programs, but the sections must be submitted sequentially in their original order. In addition, we omit RUN statements within the chapters for brevity. To submit individual sections, a RUN statement must be included at the end of the section. Otherwise it will execute only after the next set of statements is submitted.