应用SAS实现金融大数据研究

韩　燕◎著

USING SAS IN FINANCIAL BIG DATA RESEARCH

北京理工大学出版社
BEIJING INSTITUTE OF TECHNOLOGY PRESS

Using SAS in Financial Big Data Research

韩 燕 著

北京理工大学出版社
BEIJING INSTITUTE OF TECHNOLOGY PRESS

版权专有　侵权必究

图书在版编目（CIP）数据

应用 SAS 实现金融大数据研究：英文/韩燕著. —北京：北京理工大学出版社，2021.4

ISBN 978-7-5682-9737-0

Ⅰ.①应… Ⅱ.①韩… Ⅲ.①统计分析-应用软件-应用-金融-数据处理-英文 Ⅳ.①F830.41

中国版本图书馆 CIP 数据核字（2021）第 067390 号

出版发行／北京理工大学出版社有限责任公司
社　　址／北京市海淀区中关村南大街 5 号
邮　　编／100081
电　　话／（010）68914775（总编室）
（010）82562903（教材售后服务热线）
（010）68948351（其他图书服务热线）
网　　址／http：//www.bitpress.com.cn
经　　销／全国各地新华书店
印　　刷／北京九州迅驰传媒文化有限公司
开　　本／710 毫米×1000 毫米　1/16
印　　张／10　　责任编辑／武丽娟
字　　数／190 千字　　文案编辑／武丽娟
版　　次／2021 年 4 月第 1 版　2021 年 4 月第 1 次印刷　　责任校对／刘亚男
定　　价／62.00 元　　责任印制／李志强

图书出现印装质量问题，请拨打售后服务热线，本社负责调换

Preface

Financial research relies extensively on data. Mastering a statistical tool capable of handling the huge quantity of financial data is a necessary technique for every financial empiricist. There are many such tools, for example, Matlab, Stata, R, SQL, Python, etc. As a veteran in the field of empirical financial research, I have been using SAS for quite a long time. During the long years of experience, I gradually start appreciating SAS's powerful yet smart ability to help me navigate through the ocean of financial data. I should admit that I have very limited, but not totally zero, knowledge of using other statistical software. However, I still feel obliged to compare SAS with other software in the context of financial research. The most distinguished advantage of SAS over other software is its ability to handle big data. This advantage is all the more meaningful when it comes to financial data. Let me explain this advantage as follows.

Big data analysis has become trendy during the past five to ten years, largely

due to the rapid development of IT technology. To analyze data, we need to find the data in the first place. In this regard, financial data has long been collected, compiled, and distributed in a systematic, thorough, and scientific way. Most of the financial data originate from exchanges and the companies' legally published periodic reports. In many countries, the financial information is universally formatted. These features make the financial data most easily to be collected and converted into commercial database. Many companies provide such database, such as WRDS, Thomson Reuters, CSMAR, and WIND. The Nobel laureate, Professor Eugene Fama, once mentioned that he had started using WRDS database to conduct researches since 1970s. In this sense, big data has been in place in financial research for about half a century, long ahead of the recent boom of big data analysis. To some degree, the financial data define the research topics, methodologies, and even sub-disciplines of today's financial research.

According to my personal observation, the researchers' choice of the statistical software in the business and economics schools in universities varies from school to school. Interestingly, those in the finance departments are more likely to choose SAS, while those in economics and econometrics are more likely to choose Stata. This pattern of choice does have a reason. SAS treats data as a table, which stores and processes data line by line. Therefore, SAS theoretically has unlimited ability to dealing with any number of lines, although it necessarily takes a long time once the data are prohibitively large. In contrast, Stata treats data as a matrix, which requires, at least in theory, to read all data into the computer's memory before starts the processing. This way, Stata's ability to handle data is only as powerful as the computer's memory size. However, Stata's treating data as a matrix has a deeper rationale. That is, most modern econometric models are expressed in matrices, which means processing data in the form of a matrix is a more natural way in econometric research context. Partly

due to this reason, when my students ask me why I choose SAS over Stata, I often half seriously and half jokingly answer them: "That is because I am not an econometrician."

I can add another interesting observation to corroborate my argument that econometricians and those with strong econometric backgrounds tend to choose Stata. The similar choice pattern also shows up in the students who learn their econometric courses taught by the professors with different backgrounds. Empirical research methods have been compulsory courses in many business, economics, and finance programs at undergraduate, graduate, and doctoral levels in many universities. However, the professors teaching econometric have different backgrounds. In many schools, it is the econometric professor who teaches the students the concepts and methodologies of empirical research, no matter the students major in econometric or not. Needless to say, econometricians' teaching econometrics can provide the most advanced, thorough, and rigorous knowledge of the field to the students. But when it comes to the application of empirical research methods on a specific economics or finance question, the researchers in that particular field typically have a specific preference of certain econometric methods. Often the case, top tier economics and finance journals reject the papers whose main contribution is merely to apply a "better" econometric method to an old research question. In other words, high quality economics and finance research puts more weight on ideas over econometric techniques. Therefore, there has gradually emerged a new norm in which the econometric course is taught by an economic or finance professor who does not major in econometrics, statistics, or math, but specializes in specific research areas. Over the past years, I have been teaching and working with many students. It seems that those who learn econometrics from professors majoring in econometrics tend to choose Stata, while those who learn econometrics from professors majoring in finance tend to

choose SAS.

Given that SAS does not treat data as matrices, it seems to lose to Stata in terms of timely incorporating the newest statistical mythologies into the software. But on the flip side, the nearly unlimited ability to process any number of lines of data does make SAS more suitable, and in certain studies the only choice, for financial research. Those who have no experiences in handling financial data may not fully appreciate the massive quantity of financial data. Let me put it in perspective. Compared to the U. S. financial market, Chinese financial market has a very short history. We have just celebrated the thirtieth anniversary of the Chinese stock market as I finish writing this book. However, there have been more than 47 million trading-day observations for all Chinese bonds, and more than 11 million trading day observations for all Chinese stocks. For the microstructure (tick-by-tick) data, the number of observations is thousands of times larger than the daily data. The exceedingly large size of financial posts a series of challenges to us. For example, sorting data is a routine process. However, most of personal computers will have difficulty in sorting a data set with hundreds of millions of data by several variables. Matching data is another simple yet challenging task for Big data. The typical way of matching data is to first use a Cartesian product and then eliminate those that are not matched. A Cartesian product of two tables with x and y lines generate a x times y lines temporary table. Imagine how daunting the task could be if you are trying to match two tables which both have 10 million lines. In these scenarios, we need to find smarter ways to conduct the data processing. Fortunately, SAS can provide many handy tools for us.

This book summarizes my experience in using SAS to conduct financial research on big data. One of my research areas is market microstructure, which studies the tick-by-tick data of trades and quotes. As mentioned above, market

microstructure data are especially large, hence a more demanding task for researchers. I often wait for a whole day to get one result. Although often frustrated and disappointed by the tedious calculation process, I have learned a lot from these studies. Most of all, I gradually master the skills that enable me to decipher the regularities hidden in the tremendous amount of data. There are many books discussing how to use SAS. I do not intend to add to this long list. This book is rather focused on how to use SAS to conduct sophisticated financial researches, especially in the context of big data. I assume that the readers have already understood the basics of SAS. The topics of this book mainly cover the advanced research and coding issues that are seldomly discussed in the general-purpose and introductory-level SAS books. I hope the experience shared in this book can be of some help for you to conduct high-quality financial researches.

Han Yan was supported by the National Natural Science Foundation of China under grant number 71772013.

Contents

1 Basic Rules when Using SAS ····· 1

1.1 Build Our Own Research Database ····· 1

1.2 Importing and Exporting Data ····· 2

1.3 Managing Libraries and Data Sets ····· 5

1.4 Enhancing SAS Efficiency ····· 8

1.5 Best Practice to Make Our Codes and Data More Robust ····· 10

2 Advanced Usage of SAS ····· 13

2.1 Loops and Arrays ····· 13

2.2 BY statement ····· 16

2.3 Lag and Dif ····· 17

2.4 Date and Times Formats and Informats ····· 20

2.5 PROC MEANS ····· 21

3 Manipulating Tables ····· 25
3.1 Concatenating Tables ····· 26
3.2 Merging Tables in SQL procedure ····· 29
3.3 Merging Tables in DATA Step Using MERGE Statement ····· 33
3.4 Modifying Tables ····· 34
3.5 Transposing Tables ····· 36
3.6 Subsetting Tables ····· 37
4 SAS Macros ····· 39
4.1 Understanding the Basics of SAS Macros ····· 39
4.2 How SAS Executes a Macro ····· 41
4.3 The Coding Rules in Macros That are Different From Other SAS Codes ····· 43
4.4 Return The Total Number of Observations ····· 44
4.5 Existence of A Macro Variable and The Zero Observation of A Dataset ····· 46
5 Using SAS to Execute Financial Research Methodologies ····· 49
5.1 Storing Results Generate by SAS Procedures ····· 49
5.2 Summarizing Data and Statistical Tests ····· 50
5.3 Regressions ····· 53
5.4 Simulation Methods ····· 57
5.5 Event Studies ····· 61
6 Research on Mutual Funds ····· 71
6.1 The Major Research Questions in Mutual Fund Studies ····· 71
6.2 Calculating Fund Returns ····· 77
6.3 Calculating Fund Alpha's Using a Macro ····· 81
6.4 Calculating Fund Flows ····· 84

7 Market Microstructure Research …… 93
7.1 Research on Decomposing Bid-ask Spread and Estimating PIN … 93
7.2 Estimating the Microstructure Measures …… 98
参考文献 …… 141

1

Basic Rules when Using SAS

This book is not intended to be a manual or introductory book of SAS. Readers are assumed to have reasonable knowledge and/or experiences of SAS. When more than one method is present to accomplish the same goal, the author offers only one method, partly because of the lack of knowledge, partly because being thorough is by no means the aim of this book.

1.1 Build Our Own Research Database

A research project may last several months, or several years at the most. But as academia, our researches last as long as our career. Often the time, we will find the coding work highly repetitive. Take the following example. In reality,

fund management firms usually launch different mutual fund products in order to cater to the needs of different customers. For example, different classes. But for most studies in mutual funds, these different classes of funds should be treated as one fund. Therefore, an important job is to identify the unique fund. Every time we launch a mutual fund research project, we need to do this. Apparently, it saves us a lot of time if we can gradually build our database. We don't want to redo everything.

It's common that multiple researchers work together on one project. So collaborating becomes vital in terms of research efficiency. It is extremely rare that everyone in the research team is using the same software and the same computer. A very common scenario is that someone is using SAS on a Windows computer, someone is using SAS on Mac computers, and someone is using Stata etc. To make the things even more complicated, once the data contain non-English characters, the coding becomes vital. Therefore, it is a good practice to always require the codes and data are compiled in UTF-8 format. This can be done through a special clause in the Libname command.

1.2 Importing and Exporting Data

Finance research usually uses readily compiled data. These data are typically from some data providers, such as WRDS, CSMAR, and RESSET. So the first job of finance research is to "get these data into SAS." A typical way is to download the data from the data provider's website, and then to import the data into our libraries. An atypical way is to directly access the data on the data provider's database as long as such service is provided by the provider. The following summarizes the best practice of importing data.

First, check the properties of the variables. If a variable contains both numeric values and character values, some of the values, especially the numeric values are likely to be ignored. For example, if SAS has assigned the column to be numeric, then all the characters will not be imported. To avoid this problem, first in the excel file, add a character, say $ to every cell of the column, then import the file, then in SAS, modify the imported file using the following code.

```
Variablename = substr(variablename,2);
```

Then we can import the original data into SAS. Depending on the formats of the original data, we may choose one of the following codes.

To import delimited/CSV files, we can use:

```
proc  import
  datafile  ='a:\..\filename.txt'
  out  =tnfdata
  dbms  =dlm(or csv)
  replace;
  Delimiter ='#';
  Getnames =yes;
run;
```

To import Excel files, we can use:

```
proc import
   datafile = '/folders/myshortcuts/Downloads/abc_
export.xlsx'
   out =work.x1 dbms =xlsx replace;
   datarow =3;
   sheet ='Sheet1';
   Getnames =no;
run;
```

To import aligned txt files, we can use: (The input file shouldn't contain the headers.)

```
Data  test.ffm (drop = dt);
 infile    'c: \ffm.txt';
 input     dt   $
      Mkt_RF  SMB  HML  RF;
 Date = intnx('month', Input (cats(dt,'01'), anydtdte11.),
0, 'end');
 format  Date  date9.;
 Mkt_RF = Mkt_RF /100;
 SMB = SMB /100;
 HML = HML /100;
 RF = RF /100;
run;
```

The following codes can import a large variety of files, for example csv, txt, and bcp. Just specify the correct file name and extension, there is no need to change other parts of the codes. The extension is enough to tell SAS what type of file we are importing. Notice how the following codes import time. If the date and time are in "good shape," then the following is the best way to import date and time variables. If they are not in good shape, then we have to import them in character form and then use other functions to convert them into date and time. "Good shape" entails us to find the correct information.

```
data work.abc_import;
 % let _EFIERR_ = 0; /* set the ERROR detection macro
variable */
  infile '/…/abc.csv'
  MISSOVER DSD lrecl =32767 firstobs =2 delimiter ='|';
```

```
    length STKCD $ 12.;
    length CONAME $180.;
    length FYENDDT $12.;
    length FYREPORT_IND $ 16.;
    input RANO STKCD $ CONAME $ FYENDDT $ FYREPORT_IND $;
    if_ERROR_then call symputx('_EFIERR_',1);
run;
```

In the above codes, lrecl = 32,767 specifies the length and the location, so there is no need to change them. Pay attention to delimiter and firstobs options. The delimiter option specifies the delimiter. The firstobs option specifies from which line to import the data. So if the data contain a header, then firstobs = 2, otherwise, firstobs = 1. When the original data contain a header, and we want to change the variable name, there is no need to "change" the variable name. Just specifying a different name in the input option will do, because firstobs = 2 instructs SAS to ignore the header, and the input option specifies the variable name in the SAS data, hence the header in the original data has no role in importing. The only role of the header is to "remind" us how to name the variables.

1.3 Managing Libraries and Data Sets

We can protect a data set by assigning a password to the data set. SAS has three types of password: read, write, and alter. For protecting a data set against others to alter a data set, assign write and alter password to it. For example:

```
data test.datasetpw (write = password alter = password);
set csmar.cf_fdatvaln; run;
```

```
proc sort data=test.datasetpw (write=password alter=
password); by netu; run;
proc datasets library=mylib; modify students(write=
password alter=password); run;
```

The first line of codes define a new data set with write and alter password protection. The second line of codes sorts an existing data set, which either has already been assigned passwords or has not been assigned passwords and is assigned by these codes now.

To delete a password, use the following codes:

```
proc datasets library=cnmarket nolist;
modify dailymarketreturn (read=abc/);
```

Note that the above codes remove the read protection of data set "cnmarket. dailymarketreturn," which has been formerly assigned to a password of abc. And do not forget the slash followed by the password. Notice that SAS password protection does not prevent the data set from being deleted in Windows Explorer.

Use the following codes to re-order variables in the current data set.

```
data dataset-name;
 retain var1 var2;
 set dataset-name;
run;
```

Notice that the variables following RETAIN statement will appear in the first place. Variables not appeared after RETAIN statement will appear in the same order as they were.

Then let us look at how to copy a data set to another library. Say there are two libraries: liba, libb. There is a data set name dataone in liba. The following program copies this data set into libb.