1

INTRODUCTION TO SAS

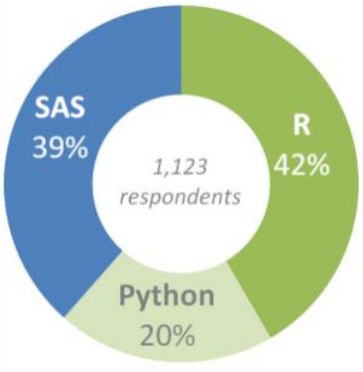
Notice: For readers who have no access to expensive SAS subscriptions, please start from chapter 3 (Free SAS University Edition). For this book, readers can use the free version to finish 80% of the tasks. In other words, students at schools without a SAS subscription could still learn SAS.

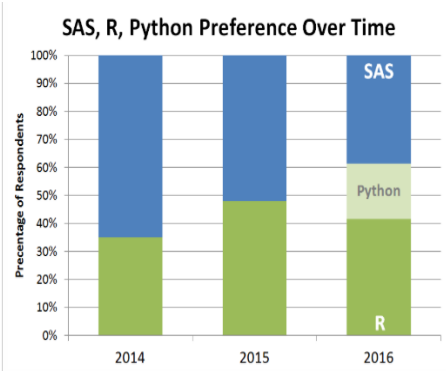
Our society has entered a so-called "Big Data" era, e.g.; for example, see Shi, Zhang and Khan (2017), and Fang and Zhang (2016). Because of this, it is important to master one computer language for business school students since we need a means to process data. This is especially true for students in various Business Analytics, Data Analytics or Data Science programs. In addition, students should understand the process and issues when dealing with data. To satisfy this purpose, there exist many good languages, such as R, Python, Matlab, C/C++, Octave, Julia, and SAS, just to mention a few. For this book, we adopt SAS. There are several reasons why we chose this wonderful language. After reading this chapter, readers would know why and have a much clearer picture about the super ability of SAS. In particular, the following topics will be covered:

- Comparisons between R, Python, and SAS
- SAS' superior capacity to process data and data analysis
- Strong support from the SAS company and community
- PC SAS vs. UNIX SAS
- Choosing a meaningful variable name
- How to debug SAS programs
- SAS Online documentation and online sample code
- Code alignment, and saving small programs for future usage
- R installation and one-line R code for this book (course)
- SAS Studio

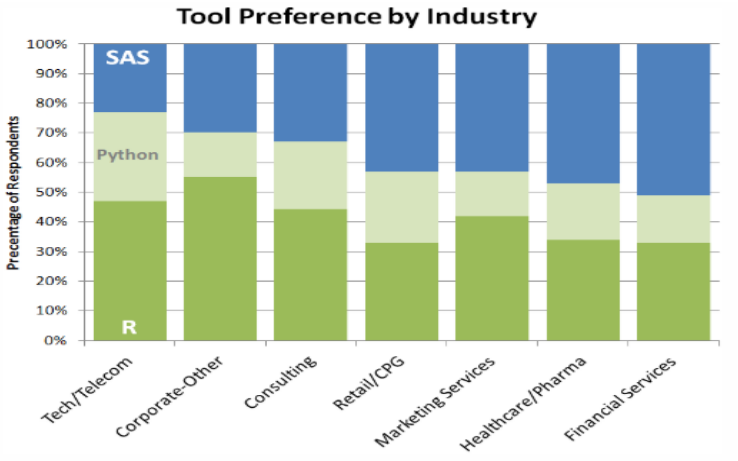
1.1 R, PYTHON, MATLAB VS SAS

It is said that R, Python and SAS represent the top three computer languages for business analytics and data science. Here is the question: Which do you prefer to use: SAS, R, or Python?" The following left bar chart reports the answer, from Burtch (2016).





From the above pie chart on the left, we could see that R seems to have a bigger share when compared with SAS and Python. Another way to look at the whole picture is to study the trend on their market shares. The above right image shows such a trend over the past several years (2014 to 2016). The market share for SAS is shrinking while Python's market share is increasing. The consensus is that Python would gain more share in the near future. Burtch (2016) shows further that different industries would definitively have their own preferences (see the image below).



The information about the shares of various industries is more relevant since this book is about financial data analytics. Comparing and contrasting different potential languages is an easy way to motivate a potential learner. Since the language chosen is SAS, it should be compared with other languages. Nowadays, SAS, R and Python are the top three languages in terms of their ability to process data. The table below shows scores based on the cost, ease of learning, data handling capability, graphical capability, level of advancements in tools, job market preference, and customer services support, and community.

Table 1.1: Comparison between R, SAS and Python (5 being the best)
Source: https://certs-school.com/sasvsr.php

Parameter	SAS	R	Python
Availability	2	5	5
Ease of learning	4.5	2.5	3.5
Data handling capabilities	4	4	4
Graphical capabilities	3	4.5	4
Advancements in tools	4	4.5	4
Job scenario	4.5	3.5	2.5
Customer services support, and community	4	3.5	3

Many scores are quite reasonable such as cost (availability). However, a few are problematic. For example, SAS is superior to R and Python in terms of data handling capacity. The most critical disadvantage for both R and Python is their lack of support. For SAS support, see the related section later in the chapter.

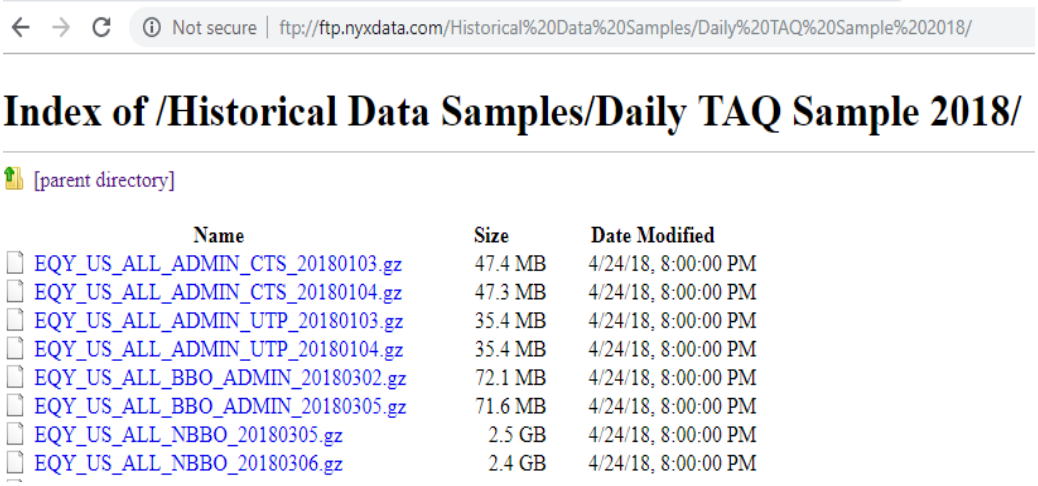
1.2 DATA ORIENTED

Many researchers argue that they usually have to spend about 80% of time to process data. This is especially true when we have very big and complex data structures, so SAS could be viewed as a data oriented language. For this very reason, we adopt SAS for this book. When processing data, we might need a very complex or multi-step procedure such as deleting missing values, replacing certain values with other specific values, converting different frequencies like estimating monthly returns from daily returns and summarizing data in a certain ways like volume-weighted returns and the like. For other languages, it is quite difficult to achieve such a complex goal. On the contrary, SAS was born for this task.

Here is another illustration related to the size of input data sets. We have tried to input a 1G data set using R and Python. Unfortunately, we failed miserably. We could not even retrieve such a scale data set into our software, let alone process it. Later in the book, we look at high-frequency data which has several gigabytes for just one day's data set. Think about this question: how does one process a 2G zip file? In the next section, we show one such example.

1.3 THREE EXAMPLES (2.5G)

Example #1: How to process one day's high frequency data? From the NYSE daily sample data we could download a data set called EQY_US_All_BBO_20180303.gz with a size of 2.5G, shown below.



The unzipped file is 14 G, shown below.

```
Directory of F:\data\TAQ\NYSE_daily_TAQ\20180305
 03/05/2018  09:24 PM    323,406,077 EQY_US_ALL_BBO_ADMIN_20180305
 03/05/2018  09:24 PM 14,427,710,445 EQY_US_ALL_NBBO_20180305
 02/11/2019  09:37 AM      1,030,291 QY_US_ALL_REF_MASTER_20180305
 02/11/2019  12:09 PM  3,224,311,897 EQY_US_ALL_TRADE_20180305
 02/11/2019  09:48 AM    306,642,383 EQY_US_ALL_TRADE_ADMIN_20180305
```

It is quite difficult to use R or Python to process the data set. Below is the R code to read the trade data.

```
setwd("f://data/TAQ/NYSE_daily_TAQ/20180305/")
x<-readLines("EQY_US_ALL_NBBO_20180305")
```

On the other hand, it is easy to use SAS to generate related SAS data sets, shown below.

```
 02/11/2019  04:52 PM    25,965,297,664 quote20180305.sas7bdat
 02/11/2019  04:52 PM     2,192,258,048 quote20180305.sas7bndx
 02/12/2019  04:52 PM     4,428,988,416 trade20180305.sas7bdat
 02/12/2019  04:52 PM       609,526,784 trade20180305.sas7bndx
```

In Chapter 12 : Dealing with Big Data, we will show the related SAS program.

Example #2: For CRSP, Yan (2019) tries to introduce CRSP to our classrooms. He generated one R data set for monthly data. However, he could not generate one R data set

for daily data. When teaching at UB to Quantitative Finance students, he has to generate annual daily data sets, i.e., 90 instead just one.

Example #3: Last year, one of our students did a term project by testing the Benford Law using SEC 10K filings. She processed 60G data. In their paper related to PIN (Probability of Informed Trading), Yan and Zhang (2014) process about 500G TAQ data. Honestly speaking, we have no clue how to use R or Python to process those data.

The conclusion is that if we are offered a big data set, such as 10G,, there is a good chance that the only language that would work is SAS.

1.4 PC SAS VS. UNIX SAS

Generally speaking, there are two types of SAS: PC-SAS and UNIX SAS. For many users, they don't have a choice since this depends on the SAS subscriptions of their schools/companies/organizations. Occasionally, the lucky ones could access both. PC SAS resides on an individual machine while UNIX SAS is installed on a server. Both have their own advantages and disadvantages.

For PC SAS, an individual user has full control since the software is on his/her own machine. Thus, storage space is usually not an issue since users could easily add more external drives. For example, a 10TB external drive costs less than $200 at Best Buy. The second advantage for using PC SAS is that users don't have to learn any UNIX commands. Third, PC SAS users could easily use SAS Editor or Notepad to write and modify their programs. For UNIX SAS users, they have to learn one UNIX editor such as `vi`. Fourth, for PC SAS, colour is used to distinguish different categories, such as commands and errors. In addition, PC SAS is able to import data from Excel files. Finally, for running PC SAS, we could run part of it, i.e., just highlighting the part of our program and running it. This feature makes our debugging task a little bit easier.

For UNIX SAS, the major advantage is associated with the server. For example, users could access many commonly shared SAS data sets. This is especially true for accessing some financial databases such as CRSP and Compustat, see Chapter 5: Open data and several financial databases for more detailed discussion. Second, users could log on to a server from anywhere. This makes doing research extremely convenient by using SAS. All a researcher needs is a computer with internet access. Third, we could apply a UNIX

no hang-up feature. Assume a job needs one week to finish. We could run a UNIX SAS program with a `nohup` feature and log out. After we logout, the SAS program is still running. One week later, we could re-log in the server to download the results. Fifth, users could run multiple UNIX SAS jobs simultaneously. This feature is not available for PC SAS users. For a good programming practice when working across PC SAS and UNIX SAS, see Zhao (2017). We denote a whole chapter, Chapter 2: UNIX SAS, to this topic.

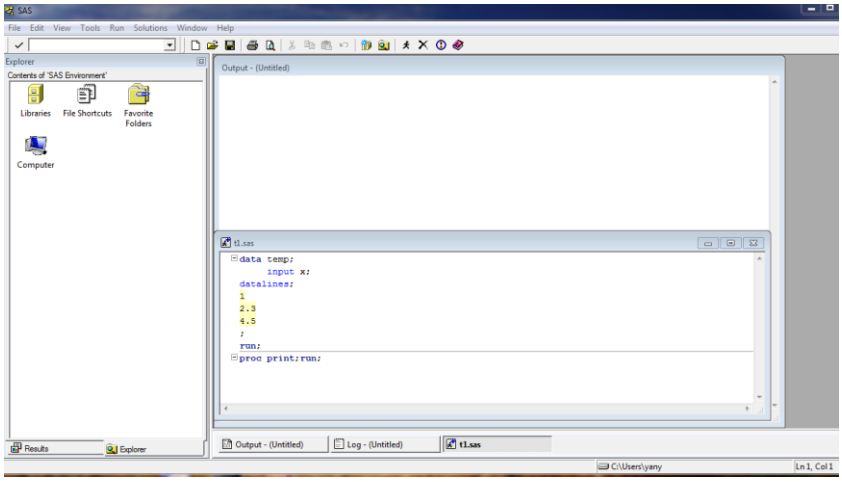
1.5 RUNNING A PC SAS PROGRAM

There are two types of PC SAS. One is called SAS, while the other is called SAS Studio, shown below.

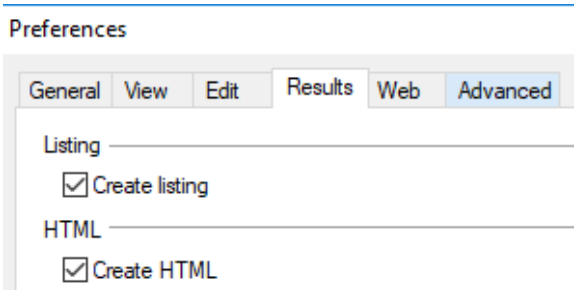
We will discuss the left one first. At the end of the chapter, we will discuss SAS Studio. When PC-SAS is launched, we see three panels (windows): 1) for the program, 2) for the log file, and 3) for the output file. A log file will help us debug our programs. First, let's enter the following code.

```
data temp;
     input x;
     datalines;
2
3
10
-3
;
run;
proc print;run;
```

Later, we will explain a similar program in more detail.



The "Output" window is for the Listing destination. For Pre-SAS 9.3, this is the default method for viewing output. Starting with 9.3, SAS switched to HTML being the default method for viewing output. While the new method is graphically nicer and easier to share output with others, if users still prefer the output window as our listing destination, we can go to Tools -> Option -> Preferences -> Results tab, and uncheck "Create HTML", and check "Create Listing" (or leave both checked, and get both results).



1.6 CHOOSING A MEANINGFUL VARIABLE NAME

To make our SAS program more readable, it is a good idea to choose meaningful variable names. For example, to estimate annual returns, `return` is better than `x`. For annual

income estimation, a variable called `annualIncome` is better than `y`. For a variable name,

1) it must be between 1 to 32 characters in length,

2) must start with a letter (a-z or A-Z) or an underscore (_), and

3) can continue with any combination of numbers, letters, and underscores.

For the following two lines, we define two macro variables. In Chapter 9: SAS Simple Marco, we will have detailed discussion relate to SAS Macro.

```
%let n=10;
%let n_stocks=10;
```

In the above two lines, the second variable is more meaningful than the first one. In the next few chapters, we will explain the meaning of those two lines.

1.7 SAS IS CASE INSENSITIVE

Unlike R which is case sensitive, SAS is case insensitive. Hence, the following three lines of SAS code are equivalent.

```
Data temp;
DATA TEMP;
DaTa TeMp;
```

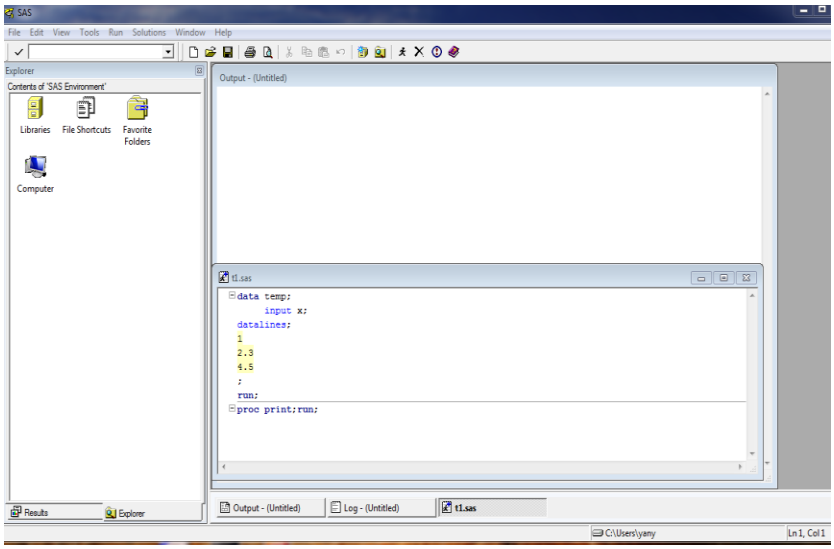
1.8 A FEW EXAMPLES

Each SAS command is followed by a semicolon (;). Key words are separated by a blank or blanks. Extra blanks are ignored. This means 3 blanks are treated as one blank and an extra blank line (row) in the program is ignored as well. Below is a simple program.

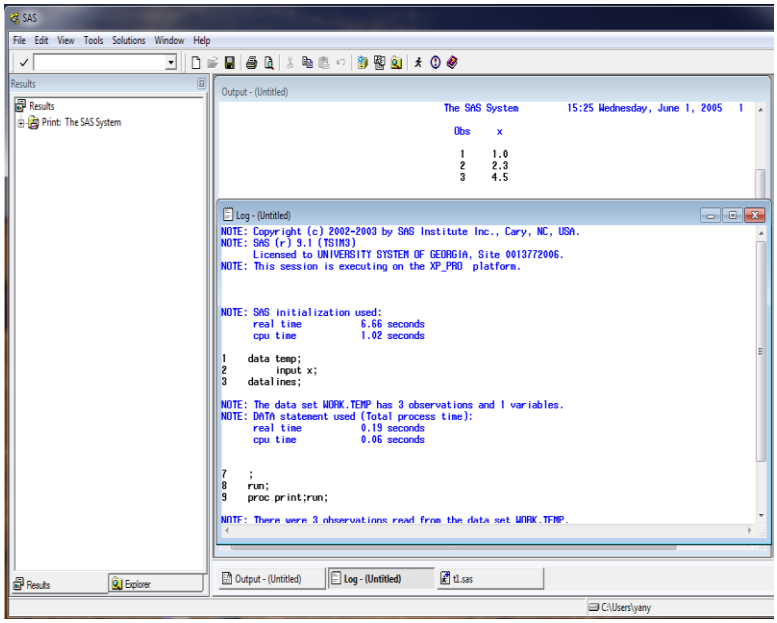
```
data temp;
     Input x;
     datalines;
1
2.3
4.5
;
run;
```

```
proc print;run;
```

For `DATA TEMP;`, `DATA` is the keyword and `TEMP` is the name of the data set. Note that we end the first line with a semicolon (;). For the second line, `INPUT` is another keyword, and `x` is our variable name. The third line tells the program that we are going to receive input data after this line. Obviously, lines 4 to 6 are our data. After we type our input values, we end the data input with a semicolon. The line 8 has `RUN;` which indicates the end of our "data block". For many cases, this `run;` is optional. The last line would then print the above data block. For using PC SAS, we could launch the SAS software, and then type the above code.



After typing our program in the editing window, click n the small "running man" icon on the menu bar (see the following image).



In the above example, the last line "`proc print;run;`" could have other equivalent versions (see the code below).

```
data temp;
     Input x;
     datalines;
1
2.3
4.5
;
run;
title ' proc print;run;';
proc print;run;
title ' proc print data=temp;run;';
proc print data=temp;run;
title ' proc print data=work.temp;run;';
proc print data=work.temp;run;
```

For "`proc print;run;`" the default data set value will be the last one. For "`proc print data=temp;run;`", the software just prints the data set called `temp`. Another way

is to use "`proc print data=work.temp;run;`", `work` in front of our `temp` data set is the name of current default directory, followed by our data set. In the next chapter, we will explain the concept of directories.

Below is our second example by using data downloaded from Yahoo!Finance. First, download IBM's historical daily data from Yahoo!Finance and input it into SAS. You can use http://datayyy.com/data_csv/ibmDaily.csv as our input data set. The first several lines are shown below.

```
Date,Open,High,Low,Close,Adj Close,Volume
1962-01-02,7.713333,7.713333,7.626667,7.626667,0.689273,387200
1962-01-03,7.626667,7.693333,7.626667,7.693333,0.695299,288000
1962-01-04,7.693333,7.693333,7.613333,7.616667,0.688370,256000
1962-01-05,7.606667,7.606667,7.453333,7.466667,0.674813,363200
1962-01-08,7.460000,7.460000,7.266667,7.326667,0.662160,544000
```

Assume that the data set is used under `c:/temp`. Below is our program.

```
data temp;
     infile 'c:/temp/ibmDaily.csv' firstobs=2 dlm=',';
     input Date Open High Low Close AdjClose Volume;
run;
proc print data=temp(obs=3);run;
```

The output is shown below.

The SAS System 14:28 Tuesday, June 26, 2018

Obs	Date	Open	High	Low	Close	Adj Close	Volume
1	.	7.71333	7.71333	7.62667	7.62667	0.68927	387200
2	.	7.62667	7.69333	7.62667	7.69333	0.69530	288000
3	.	7.69333	7.69333	7.61333	7.61667	0.68837	256000

Obviously, the date variable is not shown properly since a dot is the missing code for SAS. The reason is that 1962-01-02 could not be treated as a numeric value, the default setting for SAS input. We could treat it as a string. For SAS, a dollar sign indicates a string variable.

```
data temp;
     infile 'c:/temp/ibmDaily.csv' firstobs=2 dlm=',';
     informat date $10.;
     input Date Open High Low Close AdjClose Volume;
```

```
run;
proc print data=temp(obs=10);run;
```

The output is shown below.

The SAS System 14:34 Tuesday, June 26, 2018

Obs	date	Open	High	Low	Close	Adj Close	Volume
1	1962-01-02	7.71333	7.71333	7.62667	7.62667	0.68927	387200
2	1962-01-03	7.62667	7.69333	7.62667	7.69333	0.69530	288000
3	1962-01-04	7.69333	7.69333	7.61333	7.61667	0.68837	256000

If we simply add a dollar sign after date (shown below), the date output would not complete.

```
data temp;
     infile 'c:/temp/ibmDaily.csv' firstobs=2 dlm=',';
     input Date $ Open High Low Close AdjClose Volume;
run;
proc print data=temp(obs=3);run;
```

The output is shown below.

The SAS System 14:38 Tue

Obs	Date	Open	High	Low	Close	Adj Close	Volume
1	1962-01-	7.71333	7.71333	7.62667	7.62667	0.68927	387200
2	1962-01-	7.62667	7.69333	7.62667	7.69333	0.69530	288000
3	1962-01-	7.69333	7.69333	7.61333	7.61667	0.68837	256000

The reason is that the default length for a string variable is 8. For our input date variable, it needs 10.

1.9 TWO TYPES OF COMMENTS

The purpose of adding various types of comments is to explain the purpose of the program or different parts of a program. With appropriate comments, our programs will be readable. This would help researchers/students/programmers who wrote the program in the first place. When running a program, the underlying software would ignore those comment lines. In other words, the software would treat them as a blank or nothing. As

we mentioned already that meaningful variables names, such as *return* instead of *x* for a return variable, would make our programs more readable.

The first type of comment starts with a * and ends with a semicolon (see an example below).

```
data temp;                * you can use another name;
     Input year return;   * two input variables;
     datalines;
2015 0.1
2016 -0.05
2017 0.15
2018 0.12

;
run;
```

For a short comment, this method is quite convenient. However, a for multiple-line comment, this method is quite cumbersome. For those scenarios, we use the second method: it starts with `/*` and ends with `*/` (see an example below).

```
/* For this program, we plan to do following:
     1) retrieve data for one stock
     2) retrieve S&P500 index
     3) estimate both returns
     4) merge them by date
     5) estimate the market risk of the stock
*/
```

1.10 HOW TO DEBUG?

The best learning process is though debugging your own and other programs where finding the error is the most important step. For majority of issues, after we find the location of the bug, the problem is obvious. Because of this, the first rule is "divide and conquer". This means that if we have a dozen potential bugs, we could separate them one by one. For instance, we could run several lines to limit ourselves with the first one or two bugs. After this part of the code is clean, we move a step further. Actually, we could use the above-mentioned two types of comments to debug others' programs.

First, we convert the second type of comments into the first type. Then we comment out all the programs and run it. Since it is an empty shell, we will not get any error messages. Then we "release" the first part of the program, i.e., a few lines by changing the location of "/*". Then we run and debug the program. After cleaning the all bugs, we move the "/*" further down the program until we reach the end of the program.

1.11 SAS WARNING MESSAGES

Sometimes, we receive a warning message after running a SAS program (see one example below).

```
title 'temp2';
10         data temp2;
11              infile 'year_ret.csv' missover;
12              inut year ret;
                ____
                1
WARNING 1-322: Assuming the symbol INPUT was misspelled as inut.
```

From the above warning message, we know that `inut` should be `input`. Such a warning message or messages would have no material impact on our final result. However, it is a good practice or habit to identify and remove all the causes leading to those warning message(s).

1.12 CODE ALIGNMENT

From the previous sections, we know that for SAS programming, blanks would not play a role. This means that alignment will not have a material impact on our final results. However, a good alignment would increase the readability of our programs. The following two programs are the same except for the alignment.

```
data temp;
Input x;
datalines;
1
2.3
4.5
;
```

```
run;proc print;run;
```

The same code but with an appropriate alignment.

```
data temp;
     Input x;
     datalines;
1
2.3
4.5
;
run;
proc print;run;
```

Obviously the second one is more readable. For many more complex programs, this is crucial (see another more complex example below).

```
libname myh 'data/';
%macro all_period;
    %do j=2006 %to 2014;
        %let year=%sysfunc(substr(&j,3,2));
        %do i=1 %to 12;
                  * your code here;
         %end;
     %end;
%mend all_period;
%all_period;
```

For the above program, we plan to process monthly data over many years. Assume that all the data sets are organized by each month. Because of the size, we could generate two loops: one loop for year, and another for month. With the appropriate alignment, we could see those two loops clearly. At this moment, learners don't have to understand the meaning of those two loops, but just remember that appropriate alignment would make our programs more readable.

1.13 SAS ONLINE DOCUMENTATION

Users can find many online materials related to learning SAS. Below are a few sources.

SAS Product Documentation
https://support.sas.com/documentation/
SAS Product index from A-Z
https://support.sas.com/documentation/productaz/index.html
SAS procedure by name (version 9.4)
http://documentation.sas.com/?docsetId=allprodsproc&docsetTarget=procedures.htm&docsetVersion=9.4&locale=en
SAS procedure by product (version 9.4)

http://documentation.sas.com/?docsetId=allprodsproc&docsetTarget=p1vzipzy6l8so0n1gbbh3ae63czb.htm&docsetVersion=9.4&locale=en

To save space, only a few links are given. There many ways to find good and useful links. Another way is to type `.c1(20)` (see the related information later in the chapter).

1.14 SAS ONLINE SAMPLE CODE

We could combine SAS online sample code with the above section. However, since this link is so important. It is worthwhile to list it separately. The link is given below.

http://ftp.sas.com/samples/index

After we go to the above web link, we see the following image. To save space, only the top part is shown.

The files below are ordered by the date they were pushed to the Web. Please enter the complete link into your browser window to access the file.

In some cases, the files are in zip format that will need to be downloaded to your computer.

Note: Ascii files are named Axxxxx;
Compressed ascii files are named Axxxxx.Z;
Binary files are named Bxxxxx

where xxxxx is the four or five digit pubcode.

File name	Date added	Description	File Location
A56196	APR94	SAS Software Solutions: Basic Data Processing (Thomas Miron)	http://ftp.sas.com/samples/A56196
A56137	JUN94	SAS/OR User's Guide: Project Management, Version 6, 1st Ed	http://ftp.sas.com/samples/A56137
A56141	JUL94	SAS System for Regression, 2nd Edition	http://ftp.sas.com/samples/A56141
A56140	AUG94	SAS System for Linear Models, 3rd Edition	http://ftp.sas.com/samples/A56140

We could use keyword searching to get the SAS code we need. For example, we could try the keyword "financial" and find the following link.

http://ftp.sas.com/samples/A57601

1.15 SEVERAL SAS BOOKS FOR FINANCE

Currently, there are a few books applying SAS to financial empirical research. One book is by Boehmer, Broussard and Kallunki (2002). Its related web links are given below.

a) Amazon book, https://www.amazon.com/Using-Financial-Research-Ekkehart-Boehmer/dp/1590470397

b) Data and code download, http://www.sas.com/store/prodBK_57601_en.html

c) http://support.sas.com/publishing/authors/boehmer.html

Another book is Applied Econometrics Using the SAS System by Ajmani, Vivek (2009). Its Amazon link is given below.

https://www.amazon.com/Applied-Econometrics-Using-SAS-System/dp/0470129492/ref=sr_1_16?ie=UTF8&qid=1530279516&sr=8-16&keywords=sas+books+finance

1.16 ONE-LINE R CODE

For this book or this course, we have one-line R code (shown below).

```
> source("http://datayyy.com/sas/week1.R")
```

where > is the R prompt. After hitting the enter key, we see the following output.

```
*-------------------------------------------------------*
* Financial Data Analytics using SAS          2019     *
*-------------------------------------------------------*
* .c1: Introduction to SAS                               *
*-------------------------------------------------------*
* >.c1       # go to chapter 1  (a dot in front c1)     *
* >.uu       # go to various utility functions          *
* >.sas      # back to the main menu                    *
*-------------------------------------------------------*
```

Readers can find detailed instructions on how to download R software in Appendix A. The explanation above is clear. After typing `.c1`, we could see the extra materials associated with the first chapter (shown below).

```
> .c1
function(i){
" i  Chapter 1: Introduction to SAS
  -  ------------------------------------
  1  R installation and one-line R code
  2  Why this course and how to teach
  3  Comparisons between R, Python and SAS
  4  What is SAS? and SAS with big data
  5  Command ends with;case-insensitive,mutiple spaces
  6  Comparisons between PC SAS and UNIX SAS
  7  1st SAS program
  8  Run a PC SAS program
  9  SAS three files
 10  SAS is not case sensitive
 11  SAS command line end with a semicolon
 12  Proc print;
 13  header.sas and add a title for printing a data set
 14  1st type of comment
 15  2nd type of comment
 16  How to debug SAS programs
 17  SAS studio
 18  SAS word.temp (working directory)
 19  YouTubes
 20  Links

 Example #1:>.c1    # see the above list
 Example #2:>.c1(1) # see the 1st explanation
```

Similarly, after typing `.uu` or `.uu()`, we could see the following popup.

```
> .uu
function(){
"
 *-------------------------------------------*
 * Utilities          -- short-cut --        *
 *-------------------------------------------*
 * .inClassEx         # .ice                 *
 * .allChapters       # .all                 *
 * .calendar          # .cal                 *
 *-------------------------------------------*
 * >.ice              # see a list of ice    *
 * >.uu               # back to utilites     *
 * >.ss               # back to main menu    *
 *-------------------------------------------*
```

To get a list of all the chapters, type `.all` (see command and related image below). Note that there is a dot in front of `all`.

```
> .all
function(){
"
 *-------------------------------------------------------------------------------------------*
 * Financial Data Analytics using SAS                  by Yuxing Yan            2019    *
 *-------------------------------------------------------------------------------------------*
 * .c1  Introduction to SAS               .c16 Parsing SEC filings                           *
 * .c2  UNIX SAS                          .c17 Introduction to CRSP, Compusta,TAQ            *
 * .c3  Free SAS (SAS University Edition) .c18 How to replicate a published paper?          *
 * .c4  Open Data                         .c19 Replication (a): Multi-factor models          *
 * .c5  Data Input                        .c20 Replication (b): Liquidity (illiquidity )    *
 * .c6  Data Mainipulation                .c21 Replication (c): Momentum strategy           *
 * .c7  SAS PROC (Procedures)             .c22 Replication (d): Industry momentum           *
 * .c8  Data Output                       .c23 Replication (e): Transaction costs/spread    *
 * .c9  Simple Macro Language             .c24 Replication (f): 52-week high strategy       *
 * .c10 Simple Graphs/Data Visualization  .c25 Replication (g): Maximum trading strategy    *
 * .c11 Simple String Manipulation        .c26 Replication (h): Delisting returns           *
 * .c12 Dealig with Big Data              .c27 Replication (i): Earnings surprises          *
 * .c13 IML/Matrix Manipulation/Macros (2) .c28 Replication (j): Trading Direction          *
 * .c14 Text Analysis                     .c29 Replication (k): CEO: narcissism index       *
 * .c15 SAS,R, Python, and MySQL          .c30 Replication (m): R2 across countries         *
 *                                        .c31 Replication (n): Benford Law                 *
 *                                        .c32 Replication (o): Dynamic Conditional Corr    *
 *                                        .c33 Replication (p): Volatility spillover        *
 *                                        .c34 Replication (q): Investor sentiment index    *
 *                                        .c35 Replication (r): PIN(Infomred trading)       *
 *                                        .c36 Term projects                                *
 *-------------------------------------------------------------------------------------------*
```

REFERENCES

Analytics Training, 2011, R vs SAS (Comparison and Opinion), 2011, http://www.learnanalytics.in/blog/?p=9

Anaconda for SAS, https://github.com/sassoftware/sas_kernel

Boehmer, Ekkehart, John Paul Broussard and Juha-Pekka Kallunki (2002), SAS, https://www.amazon.com/Using-Financial-Research-Ekkehart-Boehmer/dp/1590470397

Burtch, Linda and Burtch Works, 2016, SAS vs R vs Python: Which Tool Do Analytics Pros Prefer?, https://www.kdnuggets.com/2016/07/burtchworks-sas-r-python-analytics-pros-prefer.html

Fang, Bing and Peng Zhang, Big Data in Finance, in Big Data Concepts, Theories, and Applications, S. Yu and S. Guo, Eds., ed Cham: Springer, 2016, pp. 391-412

Jain, Kunal, 2017, Python vs. R (vs. SAS) – which tool should I learn? https://www.analyticsvidhya.com/blog/2017/09/sas-vs-vs-python-tool-learn/

Kromme, Jeroen, 2017, Python & R vs. SPSS & SAS, http://www.theanalyticslab.nl/2017/03/18/python-r-vs-spss-sas/

Parvez, Irshad, 2017, SAS vs R vs Python - Which is the Best Analytics Tool to Learn?, https://www.linkedin.com/pulse/sas-vs-r-python-which-best-analytics-tool-learn-irshad-parvez/

SAS home, https://www.sas.com/en_us/home.html

SAS Product Documentation, https://support.sas.com/documentation/

Product index from A-Z, https://support.sas.com/documentation/productaz/index.html

SAS procedure by name (version 9.4), http://documentation.sas.com/?docsetId=allprodsproc&docsetTarget=procedures.htm&docsetVersion=9.4&locale=en

SAS procedure by product (version 9.4) http://documentation.sas.com/?docsetId=allprodsproc&docsetTarget=p1vzipzy6l8so0n1gbbh3ae63czb.htm&docsetVersion=9.4&locale=en

Shi, Xiang, Peng Zhang and Samee U. Khan, 2017, Quantitative Data Analysis in Finance,in Handbook of Big Data Technologies, A. Y. Zomaya and S. Sakr, Eds., Springer, 2017

Verma, Eshna, SAS versus R - Which is Better?, https://www.simplilearn.com/sas-vs-r-which-is-better-rar393-article

Yan, Yuxing (2019), CRSP for Teaching, https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3303504

Zhao, James, 2000, Good Programming Practice when Working Across PC SAS and UNIX SAS, http://support.sas.com/resources/papers/proceedings17/1363-2017.pdf

SUMMARY

In this chapter, we have discussed comparisons between several widely used computer languages such as R, Python, Matlab and SAS. We discussed the advantages and disadvantages of SAS and why SAS is a better choice for Financial Data Analytics. In addition, we have discussed how to launch PC SAS, run the simplest SAS program by entering data from our keyboard, and debug our programs.

In the next chapter, we will discuss issues related to UNIX SAS which is another type of SAS. Topics include advantages of UNIX SAS, a list of widely used UNIX commands, an UNIX editor of `vi`, how to connect with a server via SSH, ftp, putty, FileZella and how to run UNIX SAS on a server.

Appendix A: How to download R

To download and install R (free computational software), we have the following 5 steps.

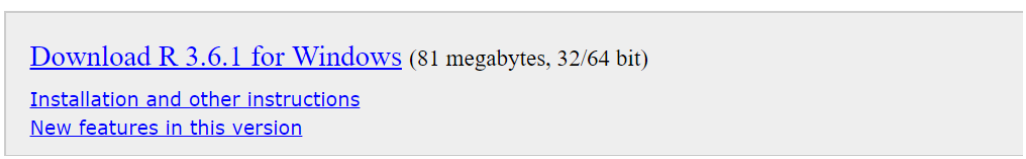
Step 1: Go to http://www.r-project.org

Step 2: Click "CRAN" under "Download" (left-hand side)

Step 3: Choose a mirror address close to your location.

Step 4: Choose the appropriate software (PC, Mac)

Step 5: Click "base". For example, for Windows, we have the following result)



EXERCISE

1.1 Is SAS free? How about R and Python?

1.2 What are the advantages and disadvantages of SAS when compared with R and Python?

1.3 In the financial industry, which language has the highest market share in terms of total usage?

1.4 For R, Python, and SAS, which language enjoys the highest score in terms of support?
1.5 Is SAS case sensitive?
1.6 Is code alignment important? If it is (not), why we should pay attention to it?
1.7 What are the main differences between UNIX and PC SAS?
1.8 How many panels do we have when running PC SAS?
1.9 How many extra files would we generate when running a UNIX SAS program?
1.10 Download WMT's historical monthly data from Yahoo!Finance and input it into SAS.
1.11 When inputting data, what is the usage of a dollar sign?
1.12 For the following program, what is the name of the default directory for the `temp` data set?

```
data temp;
     Input year ret;
     datalines;
2000 0.1
2001 0.2
2002 -0.05
;
run;
```

1.13 What is the default length for an input string variable?
1.14 Search online to find out what the maximum data set size we could retrieve into our software in terms of R, Python and SAS?
1.15 Why do we add comments to our programs? How do we add a comment?
1.16 How do we debug our programs? How do we debug others' programs?
1.17 What is the usage of `header.sas`?
1.18 From where could we find SAS online help?
1.19 There is a SAS book called "The Little SAS Book: A Primer". Download its related data sets and code.
1.20 Launch SAS University Edition and run a few simple programs.