



Part 1

Getting Started

Chapter 1 **What Is SAS?** 3

Chapter 2 **Writing Your First SAS Program** 11

2 *Learning SAS by Example: A Programmer's Guide*



Chapter 1

What Is SAS?

- 1.1 Introduction 3
- 1.2 Getting Data into SAS 4
- 1.3 A Sample SAS Program 4
- 1.4 SAS Names 7
- 1.5 SAS Data Sets and SAS Data Types 8
- 1.6 The SAS Display Manager and SAS Enterprise Guide 9
- 1.7 Problems 9

1.1 Introduction

SAS is a collection of modules that are used to process and analyze data. It began in the late '60s and early '70s as a statistical package (the name *SAS* originally stood for Statistical Analysis System). However, unlike many competing statistical packages, SAS is also an extremely powerful, general-purpose programming language. We see SAS as the predominant software in the pharmaceutical industry and most Fortune 500

4 *Learning SAS by Example: A Programmer's Guide*

companies. In recent years, it has been enhanced to provide state-of-the-art data mining tools and programs for Web development and analysis.

This book covers most of the basic data management and programming tools provided in Base SAS. Statistical procedures are not covered here.¹

The only way to really learn a programming language is to write lots of programs, make some errors, correct the errors, and then make some more. You can download all the programs and data files used in this book from this book's companion Web site at <http://support.sas.com/cody> and from the CD that accompanies this book. If you already have access to SAS at work or school, you are ready to go. If you are learning SAS on your own and do not have a copy of SAS to play with, we highly recommend that you obtain the SAS Learning Edition 4.1. This is a relatively inexpensive, fully functional version of SAS that was developed primarily for students for learning purposes only. Anyone can buy it, either through SAS Publishing, Amazon.com, or other retailers. With a pre-set die date of 12/31/08, you can use the SAS Enterprise Guide 4.1 point-and-click interface, or write and modify SAS code using the SAS Program Editor. You will be able to run any program in this book using the SAS Learning Edition...it is an ideal way to learn SAS.

1.2 Getting Data into SAS

SAS can read data from almost any source. Common sources of data are raw text files, Microsoft Office Excel spreadsheets, Access databases, and most of the common database systems such as DB2 and Oracle. Most of this book uses either text files or Excel spreadsheets as data sources.

1.3 A Sample SAS Program

Let's start out with a simple SAS program that reads data from a text file and produces some basic reports to give you an overview of the structure of SAS programs.

¹ See Ron Cody and Jeffrey K. Smith, *Applied Statistics and the Programming Language, 5th ed.* (Englewood Cliffs, NJ: Prentice Hall, 2005), which is available from SAS Press, for details on using SAS for statistical analysis.

For this example, we have a text file with data on vegetable seeds. Each line of the file contains the following pieces of information (separated by spaces):

- Vegetable name
- Product code
- Days to germination
- Number of seeds
- Price

In SAS terminology, each piece of information is called a *variable*. (Other database systems, and sometimes SAS, use the term *column*.) A few sample lines from the file are shown here:

File `c:\books\learning\veggies.txt`

Cucumber	50104-A	55	30	195
Cucumber	51789-A	56	30	225
Carrot	50179-A	68	1500	395
Carrot	50872-A	65	1500	225
Corn	57224-A	75	200	295
Corn	62471-A	80	200	395
Corn	57828-A	66	200	295
Eggplant	52233-A	70	30	225

In this example, each line of data produces what SAS calls an *observation* (also referred to as a *row* in other systems). A complete SAS program to read this data file and produce a list of the data, a frequency count showing the number of entries for each vegetable, the average price per seed, and the average number of days until germination is shown here:

Program 1-1 A sample SAS program

```
*SAS Program to read veggie data file and to produce
several reports;

options nocenter nonumber;

data veg;
  infile "c:\books\learning\veggies.txt";
  input Name $ Code $ Days Number Price;
  CostPerSeed = Price / Number;
run;
```

6 Learning SAS by Example: A Programmer's Guide

```
{ title "List of the Raw Data";  
  proc print data=veg;  
  run;  
  
{ title "Frequency Distribution of Vegetable Names";  
  proc freq data=veg;  
    tables Name;  
  run;  
  
{ title "Average Cost of Seeds";  
  proc means data=veg;  
    var Price Days;  
  run;
```

At this point in the book, we won't explain every line of the program—we'll just give an overview.

SAS programs often contain DATA steps and PROC steps. *DATA steps* are parts of the program where you can read or write the data, manipulate the data, and perform calculations. *PROC* (short for procedure) *steps* are parts of your program where you ask SAS to run one or more of its procedures to produce reports, summarize the data, generate graphs, and much more. DATA steps begin with the word DATA and PROC steps begin with the word PROC. Most DATA and PROC steps end with a RUN statement (more on this later). SAS processes each DATA or PROC step completely and then goes on to the next step.

SAS also contains *global* statements that affect the entire SAS environment and remain in effect from one DATA or PROC step to another. In the program above, the OPTIONS and TITLE statements are examples of global statements. It is important to keep in mind that the actions of global statements remain in effect until they are changed by another global statement or until you end your SAS session.

All SAS programs, whether part of DATA or PROC steps, are made up of statements. Here is the rule: all SAS statements end with semicolons. This is an important rule because if you leave out a semicolon where one is needed, the program may not run correctly, resulting in hard-to-interpret error messages.

Let's discuss some of the basic rules of SAS statements. First, they can begin in any column and can span several lines, if necessary. Because a semicolon determines the end of a SAS statement, you can place more than one statement on a single line (although this is not recommended as a matter of style).

To help make this clear, let's look at some of the statements in Program 1-1.

You could write the DATA step as shown in Program 1-2. Although this program is identical to the original, notice that it doesn't look organized, making it hard to read. Notice, too, that spacing is not critical either, though it is useful for legibility. It is a common practice to start each SAS statement on a new line and to indent each statement within a DATA or PROC step by several spaces (this author likes three spaces).

Program 1-2 An alternative version of Program 1-1

```
data veg; infile "c:\books\learning\veggies.txt"; input
Name $ Code $ Days Number
Price; CostPerSeed =
Price /
Number;
run;
```

Another thing to notice about this program is that SAS is not case sensitive. Well, this is almost true. Of course references to external files must match the rules of your particular operating system. So, if you are running SAS under UNIX or Linux, file names will be case-sensitive. As you will see later, you get to name the variables in a SAS data set. The variable names in Program 1-1 are Name, Code, Days, Number, Price, and CostPerSeed. Although SAS doesn't care whether you write these names in uppercase, lowercase, or mixed case, it does "remember" the case of each variable the *first* time it encounters that variable and uses that form of the variable name when producing printed reports.

1.4 SAS Names

SAS names follow a simple naming rule: All SAS variable names and data set names can be no longer than 32 characters and must begin with a letter or the underscore (_) character. The remaining characters in the name may be letters, digits, or the underscore character. Characters such as dashes and spaces are not allowed. Here are some valid and invalid SAS names.

Valid SAS Names

Parts
LastName
First_Name
Ques5
Cost_per_Pound
DATE
time
X12Y34Z56

Invalid SAS Names

8_is_enough	Begins with a number
Price per Pound	Contains blanks
Month-total	Contains an invalid character (-)
Num%	Contains an invalid character (%)

1.5 SAS Data Sets and SAS Data Types

We will talk a lot about SAS data sets throughout this book. For now, you need to know that when SAS reads data from anywhere (for example, raw data, spreadsheets), it stores the data in its own special form called a SAS data set. Only SAS can read and write SAS data sets. If you opened a SAS data set with another program (Microsoft Word, for example), it would not be a pretty sight—it would consist of some recognizable characters and many funny-looking graphics characters. In other words, it would look like nonsense. Even if SAS is reading data from Oracle tables or DB2, it is actually converting the data into SAS data set format in the background.

The good news is that you don't ever have to worry about how SAS is storing its data or the structure of a SAS data set. However, it is important to understand that SAS data sets contain two parts: a descriptor portion and a data portion. Not only does SAS store the actual data values for you, it stores information about these values (things like storage lengths, labels, and formats). We'll discuss that more later.

SAS has only two types of variables: character and numeric. This makes it much simpler to use and understand than some other programs that have many more data types (for example, integer, long integer, and logical). SAS determines a fixed storage length for every variable. Most SAS users never need to think about storage lengths for numerical

values—they are stored in 8 bytes (about 14 or 15 significant digits, depending on your operating system) if you don't specify otherwise. The majority of SAS users will never have to change this default value (it can lead to complications and should only be considered by experienced SAS programmers). Each character value (data stored as letters, special characters, and numerals) is assigned a fixed storage length explicitly by program statements or by various rules that SAS has about the length of character values.

1.6 The SAS Display Manager and SAS Enterprise Guide

Because SAS runs on many different platforms (mainframes, microcomputers running various Microsoft operating systems, UNIX, and Linux), the way you write and run programs will vary. You might use a general-purpose text editor on a mainframe to write a SAS program, submit it, and send the output back to a terminal or to a file. On PCs, you might use the SAS Display Manager, where you write your program in the Enhanced Editor (Editor window), see any error messages and comments about your program and the data in the Log window, and view your output in the Output window. In addition to the Enhanced Editor, an older program, simply called the Program Editor, is available for Windows and UNIX users. As an alternative to the Display Manager, you may enter the SAS environment using SAS Enterprise Guide, which is a front-end to SAS that allows you to use a menu-driven system to write SAS programs and produce reports.

There are many excellent books published by SAS that offer detailed instructions on how to run SAS programs on each specific platform and the appropriate access method into SAS. This book concentrates on how to write SAS programs. You will find that SAS programs, regardless of what computer or operating system you are using, look basically the same. Typically, the only changes you need to make to migrate a SAS program from one platform to another is the way you describe external data sources and where you store SAS programs and output.

1.7 Problems

Solutions to odd-numbered problems are located at the back of this book and on the CD that accompanies this book. Solutions to all problems are available to professors. If you are a professor, visit the book's companion Web site at <http://support.sas.com/cody> for information about how to obtain the solutions to all problems.

10 *Learning SAS by Example: A Programmer's Guide*

1. Identify which of the following variable names are valid SAS names:

Height
HeightInCentimeters
Height_in_centimeters
Wt-Kg
x123y456
76Trombones
MiXeDCasE

2. In the following list, classify each data set name as valid or invalid:

Clinic
clinic
work
hyphens-in-the-name
123GO
Demographics_2006

3. You have a data set consisting of Student ID, English, History, Math, and Science test scores on 10 students.
- The number of variables is _____
 - The number of observations is _____
4. True or false:
- You can place more than one SAS statement on a single line.
 - You can use several lines for a single SAS statement.
 - SAS has three data types: character, numeric, and integer.
 - OPTIONS and TITLE statements are considered global statements.
5. What is the default storage length for SAS numeric variables (in bytes)?