CHAPTER 1

Getting Started Using SAS® Software

1.1 The SAS Language

Many software applications are either menu driven, or command driven (enter a command—see the result). SAS is neither. With SAS, you use statements to write a series of instructions called a SAS program. The program communicates what you want to do and is written using the SAS language. There are some menu-driven front ends to SAS, SAS/ASSIST software for example, which make SAS appear like a point-and-click program. However, these front ends still use the SAS language to write programs for you. You will have much more flexibility using SAS if you learn to write your own programs using the SAS language. Maybe learning a new language is the last thing you want to do, but be assured that although there are parallels between SAS and languages you know (be they English or FORTRAN), SAS is much easier to learn.

SAS programs A SAS program is a sequence of statements executed in order. A statement gives information or instructions to SAS and must be appropriately placed in the program. An everyday analogy to a SAS program is a trip to the bank. You enter your bank, stand in line, and when you finally reach the teller's window, you say what you want to do. The statements you give can be written down in the form of a program:

```
I would like to make a withdrawal.
   My account number is 0937.
   I would like $200.
   Give me five 20s and two 50s.
```

Note that you first say what you want to do, then give all the information the teller needs to carry out your request. The order of the subsequent statements may not be important, but you must start with the general statement of what you want to do. You would not, for example, go up to a bank teller and say, "Give me five 20s and two 50s." This is not only bad form, but would probably make the teller's heart skip a beat or two. You must also make sure that all the subsequent statements belong with the first. You would not say, "I want the largest box you have" when making a withdrawal from your checking account. This statement belongs with "I would like to open a safe deposit box." A SAS program is an ordered set of SAS statements like the ordered set of instructions you use when you go to the bank.

SAS statements As with any language, there are a few rules to follow when writing SAS programs. Fortunately for us, the rules for writing SAS programs are much fewer and simpler than those for English.

The most important rule is

Every SAS statement ends with a semicolon.

This sounds simple enough. But while children generally outgrow the habit of forgetting the period at the end of a sentence, SAS programmers never seem to outgrow forgetting the semi-colon at the end of a SAS statement. Even the most experienced SAS programmer will at least occasionally forget the semicolon. You will be two steps ahead if you remember this simple rule.

Layout of SAS programs There really aren't any rules about how to format your SAS program. While it is helpful to have a neat looking program with each statement on a line by itself and indentions to show the various parts of the program, it isn't necessary.

- SAS statements can be in upper- or lowercase.
- Statements can continue on the next line (as long as you don't split words in two).
- Statements can be on the same line as other statements.
- Statements can start in any column.

So you see, SAS is so flexible that it is possible to write programs so disorganized that no one can read them, not even you. (Of course, we don't recommend this.)

Comments To make your programs more understandable, you can insert comments into your programs. It doesn't matter what you put in your comments—SAS doesn't look at it. You could put your favorite cookie recipe in there if you want. However, comments are usually used to annotate the program, making it easier for someone to read your program and understand what you have done and why.

There are two styles of comments you can use: one starts with an asterisk (*) and ends with a semicolon (;). The other style starts with a slash asterisk (/*) and ends with an asterisk slash (*/). The following SAS program shows the use of both of these style comments:

```
* Read animals' weights from file;
DATA animals;
   INFILE 'c:\MyRawData\Zoo.dat';
   INPUT Lions Tigers;
PROC PRINT DATA = animals;  /* Print the results */
RUN;
```

Since some operating environments interpret a slash asterisk (/*) in the first column as the end of a job, be careful when using this style of comment not to place it in the first column. For this reason, we chose the asterisk-semicolon style of comment for this book.

Errors People who are just learning a programming language often get frustrated because their programs do not work correctly the first time they write them. To make matters worse, SAS errors often come up in bright red letters, and for the poor person whose results turn out more red than black, this can be a very humbling experience. You should expect errors. Most programs simply don't work the first time, if for no other reason than that you are human. You forget a semicolon, misspell a word, have your fingers in the wrong place on the keyboard. It happens. Often one small mistake can generate a whole list of errors. Don't panic if you see red.

1.2 SAS Data Sets

Before you run an analysis, before you write a report, before you do anything with your data, SAS must be able to read your data. Before SAS can read your data, the data must be in a special form called a SAS data set.[1] Getting your data into a SAS data set is usually quite simple as SAS is very flexible and can read almost any data. Once your data have been read into a SAS data set, SAS keeps track of what is where and in what form. All you have to do is specify the name and location of the data set you want, and SAS figures out what is in it.

Variables and observations Data, of course, are the primary constituent of any data set. In traditional SAS terminology the data consist of variables and observations. Adopting the terminology of relational databases, SAS data sets are also called tables, observations are also called rows, and variables are also called columns. Below you see a rectangular table containing a small data set. Each line represents one observation, while Id, Name, Height, and Weight are variables. The data point Charlie is one of the values of the variable Name and is also part of the second observation.

Variables (Also Called Columns)

Observations (Also Called Rows)		Id	Name	Height	Weight
	1	53	Susie	42	41
	2	54	Charlie	46	55
	3	55	Calvin	40	35
	4	56	Lucy	46	52
	5	57	Dennis	44	.
	6	58		43	50

Data types Raw data come in many different forms, but SAS simplifies this. In SAS there are just two data types: numeric and character. Numeric fields are, well, numbers. They can be added and subtracted, can have any number of decimal places, and can be positive or negative. In addition to numerals, numeric fields can contain plus signs (+), minus signs (-), decimal points (.), or E for scientific notation. Character data are everything else. They may contain numerals, letters, or special characters (such as $ or !) and can be up to 32,767 characters long.

If a variable contains letters or special characters, it must be character data. However, if it contains only numbers, then it may be numeric or character. You should base your decision on how you will use the variable.[2] Sometimes data that consist solely of numerals make more sense as character data than as numeric. ZIP codes, for example, are made up of numerals, but it just doesn't make sense to add, subtract, multiply, or divide ZIP codes. Such numbers make more sense as character data. In the previous data set, Name is obviously a character variable, and Height and Weight are numeric. Id, however, could be either numeric or character. It's your choice.

[1] There are exceptions. If your data are in a format written by another software product, you may be able to read your data directly without creating a SAS data set. For database management systems and spreadsheets, you may be able to use SAS/ACCESS. See section 2.18 for more information. For SPSS you can use the SPSS data engine. See appendix D.

[2] If disk space is a problem, you may also choose to base your decision on storage size. You can use the LENGTH statement, discussed in section 8.15, to control the storage size of variables.

Missing data Sometimes despite your best efforts, your data may be incomplete. The value of a particular variable may be missing for some observations. In those cases, missing character data are represented by blanks, and missing numeric data are represented by a single period (.). In the data set above, the value of Weight for observation 5 is missing, and its place is marked by a period. The value of Name for observation 6 is missing and is just left blank. The use of a period for missing numeric data turns out to be very useful, as you will see.

Size of SAS data sets SAS can handle up to 32,767 variables in a single data set. The number of observations, on the other hand, is limited only by your computer's capacity to handle and store them.

Rules for SAS names The rules for naming SAS data sets and variables changed considerably with Version 7 of SAS software. Prior to Version 7, names could be only 8 characters long; now many names can be longer. These are the rules for variable names and data set member names:

- Names must be 32 characters or fewer in length.[3]
- Names must start with a letter or an underscore (_).
- Names can contain only letters, numerals, or underscores (_). No %$!*&#@, please.[4]
- Names can contain upper- and lowercase letters.

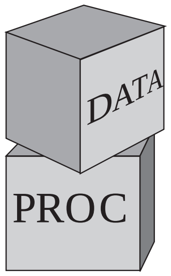
This last point is an important one. SAS is insensitive to case so you can use uppercase, lowercase or mixed case—whichever looks best to you. SAS doesn't care. The data set name heightweight is the same as HEIGHTWEIGHT or HeightWeight. Likewise, the variable name BirthDate is the same as BIRTHDATE and birThDaTe. However, there is one difference for variable names. SAS remembers the case of the first occurrence of each variable name and uses that case when printing results. That is why, in this book, we use mixed case for variable names but lowercase for other SAS names.

Documentation stored in SAS data sets In addition to your actual data, SAS data sets contain information about the data set such as its name, the date that you created it, and the version of SAS you used to create it. SAS also stores information about each variable, including its name, type (numeric or character), length (or storage size), and position within the data set. This information is sometimes called the descriptor portion of the data set, and it makes SAS data sets self-documenting.

[3] Librefs, filerefs, and format names must be 8 characters or fewer in length; informat names must be 7 characters or fewer, and member names for versioned data sets must be 28 characters or fewer.

[4] It is possible to use special characters, including spaces, in variable names if you use the system option VALIDVARNAMES=ALL and a name literal of the form '*variable-name*'N. See the online documentation for details.

1.3 The Two Parts of a SAS Program



SAS programs are constructed from two basic building blocks: DATA steps and PROC steps. A typical program starts with a DATA step to create a SAS data set and then passes the data to a PROC step for processing. Here is a simple program that converts miles to kilometers in a DATA step and prints the results with a PROC step:

DATA step

```
DATA distance;
   Miles = 26.22;
   Kilometer = 1.61 * Miles;
```

PROC step

```
PROC PRINT DATA = distance;
RUN;
```

DATA and PROC steps are made up of statements. A step may have as few as one or as many as hundreds of statements. Most statements work in only one type of step—in DATA steps but not PROC steps, or vice versa. A common mistake made by beginners is to try to use a statement in the wrong kind of step. You’re not likely to make this mistake if you remember that DATA steps read and modify data while PROC steps analyze data, perform utility functions, or print reports.

DATA steps start with the DATA statement, which starts, not surprisingly, with the word DATA. This keyword is followed by a name that you make up for a SAS data set. The DATA step above produces a SAS data set named DISTANCE. In addition to reading data from external, raw data files, DATA steps can include DO loops, IF-THEN / ELSE logic, and a large assortment of numeric and character functions. DATA steps can also combine data sets in just about any way you want, including concatenation and match-merge.

Procedures, on the other hand, start with a PROC statement in which the keyword PROC is followed by the name of the procedure (PRINT, SORT, or PLOT, for example). Most SAS procedures have only a handful of possible statements. Like following a recipe, you use basically the same statements or ingredients each time. SAS procedures do everything from simple sorting and printing to analysis of variance and 3D graphics. Other SAS procedures perform utility functions such as importing data files and data entry.

A step ends when SAS encounters a new step (marked by a DATA or PROC statement), a RUN statement, or, if you are running in batch mode, the end of the program.[1] RUN statements tell SAS to run all the preceding lines of the step and are among those rare, global statements that are not part of a DATA or PROC step. In the program above, SAS knows that the DATA step has ended when it reaches the PROC statement. The PROC step ends with a RUN statement, which coincides with the end of the program.

[1] If you use SAS long enough, you may run into an exception. Steps can also terminate with a QUIT, STOP, or ABORT statement.

While a typical program starts with a DATA step to input or modify data and then passes the data to a PROC step, that is certainly not the only pattern for mixing DATA and PROC steps. Just as you can stack building blocks in any order, you can arrange DATA and PROC steps in any order. A program could even contain only DATA steps or only PROC steps.

To review, the table below outlines the basic differences between DATA and PROC steps:

DATA steps	**PROC steps**
▶ begin with DATA statements	▶ begin with PROC statements
▶ read and modify data	▶ perform specific analysis or function
▶ create a SAS data set	▶ produce results or report

As you read this table, keep in mind that it is a simplification. Because SAS is so flexible, the differences between DATA and PROC steps are, in reality, more blurry. The table above is not meant to imply that PROC steps never create SAS data sets (many do), or that DATA steps never produce reports (they can). Nonetheless, you will find it much easier to write SAS programs if you understand the basic functions of DATA and PROC steps.

1.4 The DATA Step's Built-in Loop

DATA steps read and modify data, and they do it in a way that is flexible, giving you lots of control over what happens to your data. However, DATA steps also have an underlying structure, an implicit, built-in loop. You don't tell SAS to execute this loop: SAS does it automatically. Memorize this:

DATA steps execute line by line and observation by observation.

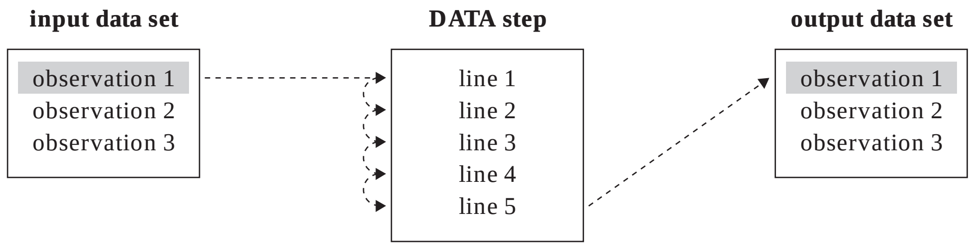
This basic concept is rarely stated explicitly. Consequently, new users often grow into old users before they figure this out on their own.

The idea that DATA steps execute line by line is fairly straightforward and easy to understand. It means that, by default, SAS executes line one of your DATA step before it executes line two, and line two before line three, and so on. That seems common sense, and yet new users frequently run into problems because they try to use a variable before they create it. If a variable named Z is the product of X and Y, then you better make sure that the statements creating X and Y come before the statements creating Z.

What is not so obvious is that while DATA steps execute line by line, they also execute observation by observation. That means SAS takes the first observation and runs it all the way through the DATA step (line by line, of course) before looping back to pick up the second observation. In this way, SAS sees only one observation at a time.

Imagine a SAS program running in slow motion: SAS reads observation number one from your input data set. Then SAS executes your DATA step using that observation. If SAS reaches the end of the DATA step without encountering any serious errors, then SAS writes the current observation to a new, output data set and returns to the beginning of the DATA step to process the next observation. After the last observation has been written to the output data set, SAS terminates the DATA step and moves on to the next step, if there is one. End of slow motion; please return to normal megahertz.

This diagram illustrates how an observation flows through a DATA step:



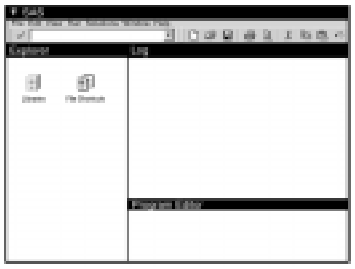
SAS reads observation number one and processes it using line one of the DATA step, then line two, and so on until SAS reaches the end of the DATA step. Then SAS writes the observation in the output data set. This diagram shows the first execution of the line-by-line loop. Once SAS finishes with the first observation, it loops back to the top of the DATA step and picks up observation two. When SAS reaches the last observation, it automatically stops.[1]

Here is an analogy. DATA step processing is a bit like voting. When you arrive at your polling place, you stand in line behind other people who have come to vote. When you reach the front of the line you are asked standard questions: "What is your name? Where do you live?" Then you sign your name, and you cast your vote. In this analogy, the people are observations, and the voting process is the DATA step. People vote one at a time (or observation by observation). Each voter's choices are secret, and peeking at your neighbor's ballot is definitely frowned upon. In addition, each person completes each step of the process in the same order (line by line). You cannot cast your vote before you give your name and address. Everything must be done in the proper order.

[1] If this seems a bit too structured, don't worry. You can override the line-by-line and observation-by-observation structure in a number of ways. For example, you can use the RETAIN statement, discussed in section 3.9, to make data from the previous observation available to the current observation. You can also use the OUTPUT statement, discussed in sections 5.11 and 5.12, to control when observations are written to the output data set.

1.5 Choosing a Mode for Submitting SAS Programs

So far we have talked about writing SAS programs, but simply writing a program does not give you any results. Just like writing a letter to your representative in Congress does no good unless you mail it, a SAS program does nothing until you submit or execute it. You can execute a SAS program several ways, but not all methods are available for all operating environments. Check in the SAS Companion for your operating environment or with your SAS site representative to find out which methods are available to you. The method you choose for executing a SAS program will depend on your preferences and on what is most appropriate for your application and your environment. If you are using SAS at a large site with many users, then ask around and find out which is the most accepted method of executing SAS. If you are using SAS on your own personal computer, then choose the method that suits you.



SAS windowing environment If you type SAS at your system prompt, or click on the SAS icon, you will most likely get into the SAS windowing environment (also called interactive SAS). In this environment, formerly called display manager, you can write and edit SAS programs, submit programs for processing, and view and print your results. In addition, there are many SAS windows for performing different tasks such as managing SAS files, customizing the interface, accessing SAS Help, and importing or exporting data. Exactly what your windowing environment looks like depends on the type of computer or terminal you are using, the operating environment on the computer, and what options are in effect when you start up SAS. If you are using a personal computer, then the SAS windowing environment will look similar to other programs on your computer, and many of the features will be familiar to you.

Non-interactive mode Non-interactive mode is where your SAS program statements are in a file on your system, and you start up SAS specifying that you want to execute that file. SAS immediately starts to process your file and ties up your computer, or window, until it is finished. The results are usually placed in a file or files, and you are returned to your system prompt.

Non-interactive mode is useful in many situations. This mode is good if you want your program to execute immediately, but you do not want to or cannot use a windowing environment. Non-interactive mode is usually started by typing SAS at your system prompt (shown here as $), followed by the filename containing your program statements:

```
$ SAS MyFile.sas
```

Batch or background mode With batch or background mode, your SAS program is in a file. You submit the file for processing with SAS. Your SAS program may start executing immediately, or it could be put in a queue behind other jobs. Batch processing is used a lot on mainframe computers, which are capable of executing many processes at one time. You can continue to work on your computer while your job is being processed, or better yet, you can go to the baseball game and let the computer work in your absence. Batch processing is usually less expensive than other methods and is especially good for large jobs which can be set up to execute at off hours when the rates are at their lowest. When your job is complete, the results will be placed in a file or files, which you can display or print at any time.

Batch processing may not available for your operating environment. Check in the SAS Companion for your operating environment to see if it is available, then check with your SAS site representative to find out how to submit SAS programs for batch processing. Even sites with the same operating environment may have different ways of submitting jobs in batch mode.

Interactive line mode This mode is mentioned only because you might see it in the SAS documentation, and you might get into it by accident. In interactive line mode, you are prompted for SAS statements one line at a time. There is no easy way to correct mistakes once you have entered them, so unless you are an excellent typist, and an excellent programmer, interactive line mode is exceedingly frustrating.

If you do find yourself in this mode (you will know when you get a 1? as a prompt), you can get out by typing ENDSAS; and pressing ENTER. For example

```
    1? ENDSAS;
```

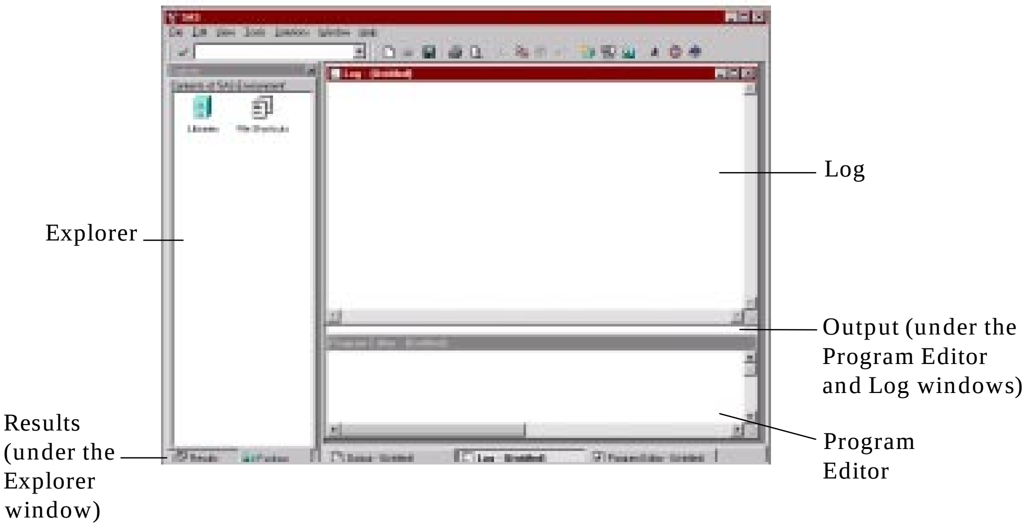
Seek assistance from your SAS site representative to find out why you got into line mode and how to avoid it in the future.

1.6 Windows and Commands in the SAS Windowing Environment

It used to be that SAS looked pretty much the same on all platforms, and you couldn't change its appearance. But now SAS adopts the look and feel of your operating environment, and there are many ways in which you can customize your SAS environment. This is good for you because many aspects of the SAS windowing environment will be familiar, and if you don't like the default view, you can change it. It makes writing about it more difficult, because we can't tell you exactly what your SAS session will look like and how it will behave. However, there are many common elements between the various operating environments, and you will probably already be familiar with those elements which are different.

The SAS Windows

There are five basic SAS windows: the Results and Explorer windows, and three programming windows: Program Editor, Log, and Output. It is possible to bring up SAS without all these windows, and sometimes the windows are not immediately visible (for example, in the Windows operating environment, the Output window comes up behind the Program Editor and Log windows), but all these windows do exist in your SAS session. There are also many other SAS windows that you may use for tasks such as getting help, changing SAS system options, and customizing your SAS session. The following figure shows the default view for a Microsoft Windows 95 SAS session, with pointers to the five main SAS windows.


Log
Explorer
Output (under the Program Editor and Log windows)
Results (under the Explorer window)
Program Editor

Program Editor This window is a text editor. You can use it to type in, edit, and submit SAS programs as well as edit other text files such as raw data files.

Log The Log window contains notes about your SAS session, and after you submit a SAS program, any notes, errors, or warnings associated with your program as well as the program statements themselves will appear in the Log window.

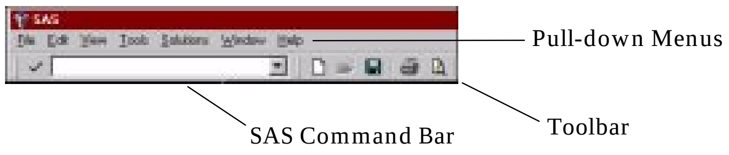
Output If your program generates any printable results, then they will appear in the Output Window.

Results The Results window is like a table of contents for your Output window; the results tree lists each part of your results in an outline form.

Explorer The Explorer window gives you easy access to your SAS files and libraries.

The SAS Commands

There are SAS commands for performing a variety of tasks. Some tasks are probably familiar, such as opening and saving files, cutting and pasting text, and accessing Help. Other commands are specific to the SAS System, such as submitting a SAS program, or starting up a SAS application. You may have up to three ways to issue commands: menus, the tool bar, or the SAS command bar (or command line). The following figure shows the location of these three methods of issuing SAS commands in the Windows operating environment default view.



Menus Most operating environments will have pull-down menus located either at the top of each window, or at the top of your screen. If your menus are at the top of your screen, then the menus will change when you activate the different windows (usually by clicking on them). You may also have, for each window, context-sensitive pop-up menus that appear when you press the right or center button of your mouse.

Tool bar You will probably see the tool bar only if you are working in a highly graphical system, such as a personal computer or UNIX system running X Windows. The tool bar gives you quick access to commands that are already accessible through the pull-down menus.

SAS command bar The command bar is a place that you can type in SAS commands. In some operating environments the command bar is located with the tool bar (as shown here); in other operating environments you may have a command line with each of the SAS windows (usually indicated by `Command=>`). Most of the commands that you can type in the command bar are also accessible through the pull-down menus or the tool bar.

Controlling your windows The Window pull-down menu gives you choices on how the windows are placed on your screen. You can also activate any of the programming windows by selecting it from the Window pull-down menu, typing the name of the window in the command line area of your SAS session, or simply clicking on the window.

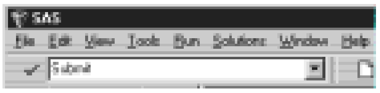
1.7 Submitting a Program in the SAS Windowing Environment

Naturally after going to the trouble of writing SAS programs, you want to see some results. As we have already discussed, there are several ways of submitting your SAS programs. If you are using SAS in a windowing environment, such as Windows or UNIX, then you will probably want to submit your programs from within the SAS windowing environment.

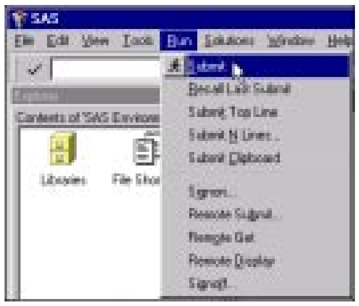
Getting your program into the Program Editor The first thing you need to do is get your program into the Program Editor window. You can either type your program into the Program Editor, or you can bring the program into the Program Editor window from a file. The commands for editing in the Program Editor and for opening files should be familiar. SAS tries to follow conventions that are common for your operating environment. For example, to open a file in the Program Editor, you can select `Open` from the File pull-down menu. For some operating environments you may have an `Open` icon on your tool bar, and you may also have the option of pasting your file into the Program Editor from the clipboard.

Submitting your program Once your program appears in the Program Editor, you execute it using the SUBMIT command. Depending on your operating environment, you have a few choices on how to execute the SUBMIT command.

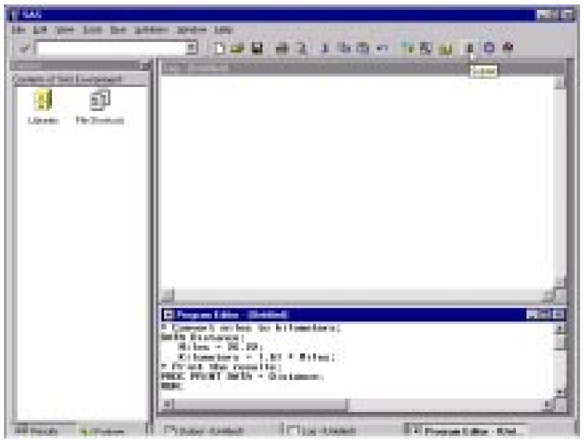
Use the `Submit` icon on the tool bar.



Make the Program Editor window active and enter SUBMIT in the command line area of your SAS session.



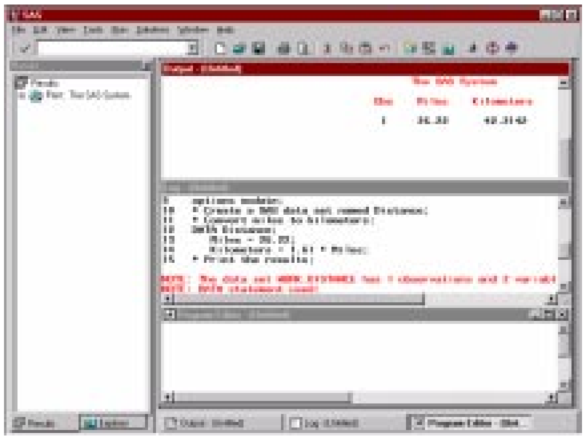
Make the Program Editor window active and select `Submit` from the Run pull-down menu.



```
* Convert miles to kilometers;
DATA Distance;
   Miles = 26.22;
   Kilometers = 1.61 * Miles;
* Print the results;
PROC PRINT DATA = Distance;
RUN;
```

The figure shows a program in the Program Editor ready to be submitted using the `Submit` icon on the tool bar (Windows 95).

Viewing the SAS Log and Output After you submit your program, it is cleared from the Program Editor window, and the results of your program go into the Log and Output windows. At first it may be a shock for you to see your program disappear in front of your eyes. Don't worry; the program you spent so long writing is not gone forever. If your program produced any output, then you will also get new entries in your Results window (your Results window may not be visible if you do not have the SAS Explorer window active). The Results window is like a table of contents for your SAS output and is discussed in more detail in section 1.9. The figure is an example of what your screen might look like after you submit a program.



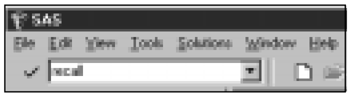
```
                                   The SAS System

                              Obs    Miles    Kilometers

                               1     26.22     42.2142
```

```
9      options nodate;
10     * Create a SAS data set named Distance;
11     * Convert miles to kilometers;
12     DATA Distance;
13        Miles = 26.22;
14        Kilometers = 1.61 * Miles;
15     * Print the results;

NOTE: The data set WORK.DISTANCE has 1 observations and 2 variabl
NOTE: DATA statement used:
```

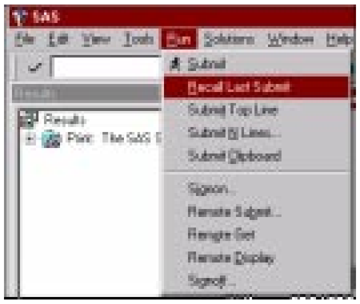
You may not see all three of the programming windows (Program Editor, Log, and Output) at the same time. In some operating environments, the windows are placed one on top of the other. You can bring a window to the top by clicking on it, typing its name in the command line area, or selecting it from the Window menu.

Getting your program back

Unfortunately for most of us, our programs do not run perfectly every time. If you have an error in your program, you will most likely want to edit the program and run it again. To get your program back in the Program Editor window, use the RECALL command. You have two choices for executing the RECALL command.



Make the Program Editor the active window, then enter RECALL in the command line area of your SAS session.



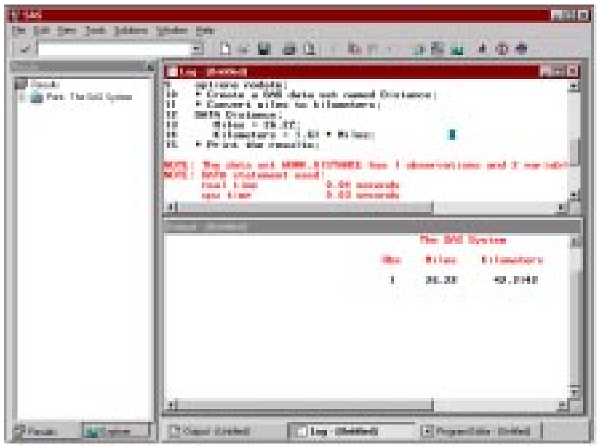
Make the Program Editor the active window, then select `Recall Last Submit` from the Run pull-down menu.

The RECALL command will bring back the last block of statements you submitted. If you use the RECALL command again, it will insert the block of statements submitted before the last one, and so on and so on, until it retrieves all the statements you submitted.

1.8 Reading the SAS Log

Every time you run a SAS job, SAS writes messages in your log. Many SAS programmers ignore the SAS log and go straight to the output. That's understandable, but dangerous. It is possible—and sooner or later it happens to all of us—to get bogus results that look fine in the output. The only way to know they are bad is to check the SAS log. Just because it runs doesn't mean it's right.

Where to find the SAS log The location of the SAS log varies depending on the operating environment you use, the mode you use (SAS windowing environment, non-interactive, or batch), and local settings. If you submit a program in the windowing environment, you will, by default, see the SAS log in your Log window as in the following figure.



```
    options nodate;
    * Create a SAS data set named Distance;
    * Convert miles to kilometers;
    DATA Distance;
       Miles = 26.22;
       Kilometers = 1.61 * Miles;
    * Print the results;

NOTE: The data set WORK.DISTANCE has 1 observations and 2 variab
NOTE: DATA statement used:
      real time           [illegible] seconds
      cpu time            [illegible] seconds
```

```
                    The SAS System

              Obs    Miles    Kilometers

               1     26.22     42.2142
```

If you submit your program in batch or non-interactive mode, the log will be written to a file that you can view or print using your operating environment's commands for viewing and printing. The name given to the log file is generally some permutation of the name you gave the original program. For example, if you named your SAS program Marathon.sas, then it is a good bet that your log file will be Marathon.log. At some installations the log and output files are written to a single file, so don't be surprised if you find them together.

What the log contains People tend to think of the SAS log as either a rehash of their program or as just a lot of gibberish. OK, we admit, there is some technical trivia in the SAS log, but there is also plenty of important information. Here is a simple program that converts miles to kilometers and prints the result:

```
* Create a SAS data set named distance;
* Convert miles to kilometers;
DATA distance;
   Miles = 26.22;
   Kilometers = 1.61 * Miles;
* Print the results;
PROC PRINT DATA = distance;
RUN;
```

If you run this program, SAS will produce a log similar to this:

```
❶ NOTE: Copyright (c) 1998 by SAS Institute Inc., Cary, NC, USA.
  NOTE: SAS (r) Proprietary Software Version 7 BETA  (TS B1)
        Licensed to XYZ Inc., Site 0098541001.
  NOTE: This session is executing on the WIN_95  platform.

  NOTE: SAS initialization used:
        real time           24.00 seconds

❷ 1    * Create a SAS data set named distance;
  2    * Convert miles to kilometers;
  3    DATA distance;
  4       Miles = 26.22;
  5       Kilometers = 1.61 * Miles;
  6    * Print the results;
❸ NOTE: The data set WORK.DISTANCE has 1 observations and 2 variables.
  NOTE: DATA statement used:
        real time           2.80 seconds

❷ 7    PROC PRINT DATA = distance;
  8    RUN;

❹ NOTE: PROCEDURE PRINT used:
        real time           1.30 seconds
```

The SAS log above is a blow-by-blow account of how SAS executes the program.

❶ It starts with notes about the version of SAS and the SAS site number.

❷ It contains the original program statements with line numbers added on the left.

❸ The DATA step is followed by a note containing the name of the SAS data set created (WORK.DISTANCE), and the number of observations (1) and variables (2). A quick glance is enough to assure you that you did not lose any observations or accidentally create a lot of unwanted variables.

❹ Both DATA and PROC steps produce a note about the computer resources used. At first you probably won't care in the least. But if you run on a multi-user system or have long jobs with large data sets, these statistics may start to pique your interest. If you ever find yourself wondering why your job takes so long to run, a glance at the SAS log will tell you which steps are the culprits.

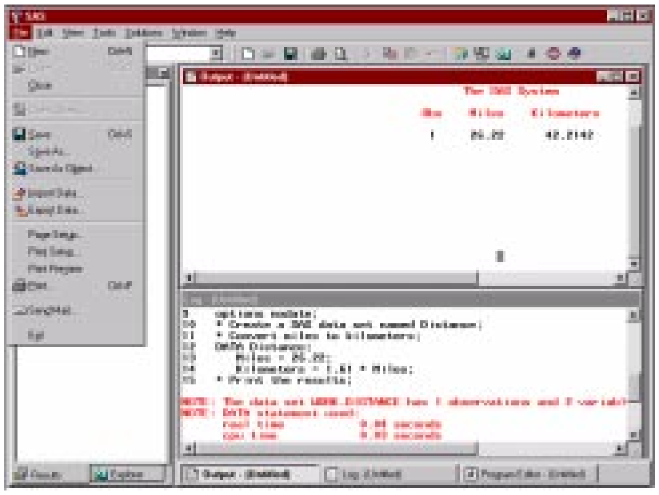
If there were error messages, they would appear in the log, indicating where SAS got confused and what action it took. You may also find warnings and other types of notes which sometimes indicate errors and other times just provide useful information.

1.9 Viewing and Printing the SAS Output

How you view or print your output depends on how you submit your programs. If you submit your program in the SAS windowing environment, then your output will, by default, go to the Output window. If you choose another way to submit your programs, either batch or non-interactive, then your output will probably be in a file on your computer. Use your operating environment's commands to view and print the output (also called listing) files. For example, if you execute your SAS program in non-interactive mode on a UNIX system, then your output will be in a file with an extension .lst. To view the file, you can use either the cat or more commands, and to print the file you would use your system's command for printing files (usually you would type either `lp` or `lpr`).

The Output window After submitting your program in the SAS windowing environment, your results will go to your Output window. If you have the SAS Explorer option turned on (some operating environments have this turned on by default, while others do not), then you will also see a listing of the different parts of your output in your Results window. The following figure shows what your screen might look like after submitting a simple program under Windows 95.

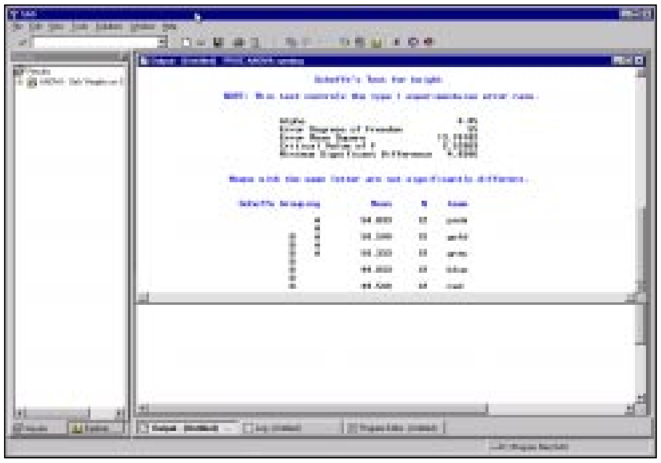
Printing or saving the contents of the Output window If you want to print or save the entire contents of the Output window, first activate the Output window by clicking in it, then select either `Print` or `Save As` from the `File` pull-down menu. If you are not using a personal computer, then your environment may not be set up for printing from within SAS. If you cannot print from within SAS, then save the output to a file and use your system's command for printing files.



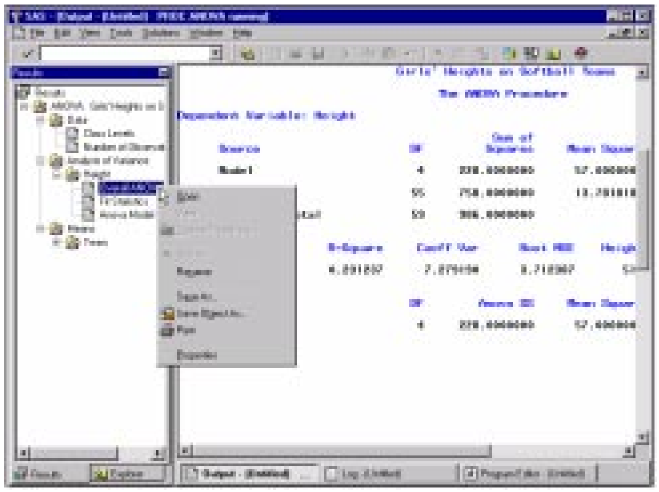
```
                                   The SAS System

                          Obs    Miles    Kilometers

                           1     26.22      42.2142
```

```
9     options nodate;
10    * Create a SAS data set named Distance;
11    * Convert miles to kilometers;
12    DATA Distance;
13       Miles = 26.22;
14       Kilometers = 1.61 * Miles;
15    * Print the results;

NOTE: The data set WORK.DISTANCE has 1 observations and 2 variabl
NOTE: DATA statement used:
      real time           0.05 seconds
      cpu time            0.05 seconds
```

The Results window When you have a lot of output, the Results window can be very helpful. The Results window is like a table of contents for your output. It lists each procedure that produces output, and if you open, or expand, the procedure in the Results tree, you can see each part of the procedure output. The following figure shows what your screen might look like if you ran the Analysis of Variance (ANOVA) procedure.



There is one entry in the Results window for the ANOVA procedure. Notice that in the Output window, you see the end of the procedure's output. If you expand the ANOVA procedure in the results tree, by clicking on the plus (+) signs, then you will see all the different parts of the ANOVA output. Double click on the output you want to see, and it will appear at the top of the Output window. The following figure shows what your screen would look like after you double click on the `Overall ANOVA` item in the Results window.



```
                                    Girls' Heights on Softball Teams

                                          The ANOVA Procedure

Dependent Variable: Height

                                                          Sum of
        Source                          DF               Squares       Mean Square

        Model                            4           228.0000000         57.000000

                                        55           758.0000000         13.781818

                        Total           59           986.0000000

                        R-Square     Coeff Var      Root MSE      Heigh

                        0.231237      7.275158      3.712367

                                        DF              Anova SS       Mean Square

                                         4           228.0000000         57.000000
```

Printing or saving parts of the output Using the Results window, it is possible to print or save just the parts of the output you want. First highlight the item you want in the Results window, then bring up the context-sensitive menu. In the Windows operating environment you do this with the right mouse button; in other operating environments, it may be the middle or right mouse button. Then select either `Print` or `Save As` from the pop-up menu. You may also be able to print or save from the File pull-down menu once you highlight the output part you want. If your SAS environment is not set up for printing from within SAS, then save your results to a file and use your operating environment's command for printing files.

Viewing your output using a browser PC users,[1] you can change your output destination from the standard listing in the Output window to a internet browser such as Internet Explorer or Netscape. In Windows operating environments, make this change in the Results tab of the Preferences window. Access the Preferences window from the `Options` item in the Tools pull-down menu.

[1] At the time this book was written, this feature was available only for PC users, but it may also be availabe for UNIX users in future releases.

1.10 Using SAS System Options

System options are parameters you can change that affect the SAS System—how it works, what the output looks like, how much memory is used, error handling, and a host of other things. The SAS System makes many assumptions about how you want it to work. This is good. You do not want to specify every little detail each time you use SAS. However, you may not always like the assumptions SAS makes. System options give you a way to change some of these assumptions.

A long list of system options can be found in your online documentation for the SAS language. The options are grouped by the following general areas:

Communications	Log and procedure output control
Environment control	Macro
Files	Sort
Input control	System administration
Graphics	

Not all options are available for all operating environments. A list of options specific to your operating environment appears in the SAS Companion documentation for your operating environment. You can see a list of system options and their current values by opening the SAS System Options window or by using the OPTIONS procedure. To use the OPTIONS procedure, submit the following SAS program and view the results in the SAS log:

```
PROC OPTIONS;
RUN;
```

There are four ways to specify system options. Some options can be specified only by using some of these methods. The SAS Companion documentation tells you which methods are valid for each system option:

1. Your system administrator (this could be you if you are using a PC) can create a SAS configuration file which contains settings for the system options. This file is accessed by the SAS System every time SAS is started.

2. Specify system options at the time you start up SAS from your system's prompt (called the invocation).

3. Change selected options in the SAS System Options window if you are using the SAS windowing environment.

4. Use the OPTIONS statement as a part of your SAS program.

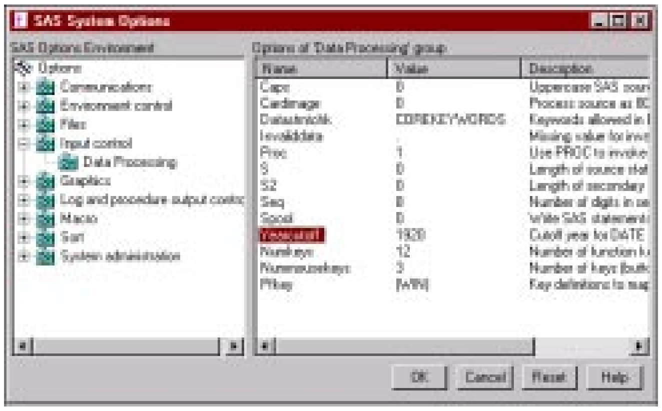
The methods are listed here in order of increasing precedence; method 2 will override method 1, method 3 will override method 2, and so forth. If you are using the SAS windowing environment, methods 3 and 4, the SAS System Options window and OPTIONS statement will override each other—so whichever was used last will be in effect. Only the last two methods are covered here. The first two are very system dependent; to find out more about these methods see the SAS Companion documentation for your operating environment.

OPTIONS statement The OPTIONS statement is part of a SAS program and affects all steps that follow it. It starts with the keyword OPTIONS and follows with a list of options and their values. For example

```
OPTIONS LINESIZE = 80 NODATE;
```

The OPTIONS statement is one of the special SAS statements which do not belong to either a PROC or a DATA step. This global statement can appear anywhere in your SAS program, but it usually makes the most sense to let it be the first line in your program. This way you can easily see which options are in effect. If the OPTIONS statement is in a DATA or PROC step, then it affects that step and the following steps. Any subsequent OPTIONS statements in a program override previous ones.

The SAS System Options window You can view and change SAS system options through the SAS System Options window. Open it by either typing OPTIONS in the command line area on your screen, or by selecting it from the Tools pull-down menu. To change the value of an option, first locate the option by clicking on the appropriate category on the left side of the screen. A list of options and their current values will appear on the right side of the screen. Right click on the option itself to modify the value or set it to the default.



Options of 'Data Processing' group

Name	Value	Description
Caps	0	Uppercase SAS sour
Cardimage	0	Process source as 8C
Datastmtchk	COREKEYWORDS	Keywords allowed in I
Invaliddata	.	Missing value for inva
Proc	1	Use PROC to invoke
S	0	Length of source stat
S2	0	Length of secondary
Seq	8	Number of digits in se
Spool	0	Write SAS statement
Yearcutoff	1920	Cutoff year for DATE
Numkeys	12	Number of function k
Nummousekeys	3	Number of keys (butt
Pfkey	[WIN]	Key definitions to map

Common options The following are some common system options you might want to use:

CENTER | NOCENTER	This option works as a switch. CENTER centers your output on the page. NOCENTER left justifies your output.
DATE | NODATE	This is also a switch. With DATE, today's date will appear at the top of each page of output; with NODATE it will not.
NUMBER | NONUMBER	This switch controls whether or not page numbers appear on each page of SAS output.
LINESIZE = *n*	With LINESIZE you can control the maximum length of output lines. Possible values for *n* are 64 to 256.
PAGESIZE = *n*	PAGESIZE controls the maximum number of lines per page of output. Possible values for *n* are 15 to 32767.
PAGENO = *n*	Starts numbering output pages with *n*.
YEARCUTOFF = *yyyy*	Specifies the first year in a hundred-year span for interpreting two-digit dates.