Guide to Using SAS

The only way to learn a new programming language is by writing in it... This is the basic hurdle; to leap over it you have to be able to create the program text somewhere, compile it successfully, load it, run it, and find out where your output went. With these mechanical details mastered, everything else is comparatively easy.

B. Kernighan and D. Ritchie, *The C Programming Language*

S. Sawyer — Washington University — September 29, 2010

Introduction. SAS is a widely used and powerful computer package for analyzing statistical data. It is currently the most commonly used statistical package when large databases have to be managed, but is also easy to use for small or medium-sized data sets.

In general, SAS can print out summaries of data, draw graphs, carry out statistical tests, estimate statistical parameters, compute P-values, and many other things. A disadvantage is that you have to communicate with SAS by writing a "SAS program" in a language that takes some learning, and SAS's output can also be difficult to decipher. However, the form of much of SAS's output ("ANOVA tables") are standard ways of presenting statistical analyses in books, journals, and technical reports and is worth learning for its own sake.

SAS is available on Microsoft Windows, dual-boot Mac, and UNIX computers. Nowadays most people use windowed versions of SAS, but SAS can also be used from a command-line or DOS-Prompt interface for those who prefer. For historical reasons, SAS from a command-line interface is called "batch mode". The basic steps of using SAS and the form of the output are the same in all cases.

The following applies to using SAS on any computer. Later on we will talk about running SAS on the PCs in the ArtSci computer lab at Washington University. (Other computer labs at WashU often also have SAS. The local WashU Software Library will also allow you to install a version of SAS on your own desktop or laptop for a modest license fee.) However, most of what we say below applies to Windows SAS or SAS in general.

Running SAS: Getting output or analyzing a problem using SAS can always be thought of as being composed of three logical parts:

(i) *SAS Input File (or Program File)*: Written by you, this is a list of instructions to SAS that includes your data (or else where SAS can find it) and what you want SAS to do with it.

(ii) *SAS Output File*: Written by SAS, the results of the analyses that you requested, or, at least, the results for your requests that SAS could figure out. This is written in a form suitable to be shown to other people. In particular, error or warning messages do not applear in this output file.

(iii) *SAS Log File*: Written by SAS, an error or log file which tells you why SAS couldn't figure out what you wanted to do (if that was the case). The log file also gives detailed information about what SAS was thinking at each step, which can be very useful at times.

This setup may seem clumsy or overly formal, but a SAS analysis of even medium complexity can be composed of two or three data sets and seven or eight types of analyses. A formal input program preserves for later reference exactly what data you were referring to and what you wanted done. Both the input and output files can be kept and copied into other programs.

In windows mode, the three parts are in three different windows. You write your program in one window (or load it from a pre-existing file) and click on a button to "Submit" it. (After you open SAS, click on either `Run` then `Submit` on the Main Menu, or else click on the `icon of a running man` on the Main Iconbar.) SAS then fills in the Output window with the appropriate information. Error and Log-file information are written to a second window, but this window is not immediately visible unless your program has no displayable output at all. (This can happen on the first attempt for a new program.)

In batch mode (or command-line mode), the three parts are three different text files. You write the input file using a text editor or ASCII editor (such as `Notepad` or `Wordpad` on a Microsoft Windows computer) and submit it to SAS by entering a command at the keyboard. (See below.) SAS then writes two text files in response, a log file and an output file, although the output file may be missing if SAS feels that there is no output to report. By convention, if your program file is `myfile.sas`, the output file is `myfile.lst`, and the log file is `myfile.log`.

If you have an ArtSci computer account at Washington University, then you can use SAS batch mode from a terminal or a using a secure `telnet` program from a desktop or laptop. In a UNIX account, you can write a SAS program using a UNIX text editor such as `vi` or `pico` or `emacs` and then run it by entering `sas myfile` at the UNIX command line. (See discussion below.) We first discuss using SAS in windows mode.

Opening SAS in Windows Mode: To use windows SAS on one of the PCs in the ArtSci computer lab, first click on the SAS icon on the desktop. Alternatively, you can click on

```
Start | All Programs | SAS | SAS 9.2
```

The notation `|` above means "then": that is, you use the mouse to click on `Start` and then, on the drop-down or drop-up window that appears, on `Programs`, then on `SAS`, etc. Windows SAS in the ArtSci computer lab is only on the PCs and not on the MacIntoshes.

Using SAS in Windows Mode: Windows versions of SAS look very similar whether they are on MacIntosh, Microsoft Windows, or UNIX machines. All versions have a Log window, an Edit (program) window, and an Output window, which may be missing unless you have output.

PC Windows SAS currently opens with three tiled windows, a lefthand narrow window with indexes of various sorts next to two larger windows stacked one above the other. When you first open the program, the top stacked window is named `Log (Untitled)`. The lower stacked window is named `Editor (Untitled)` and can be used to enter a SAS program or input file. An `Output` window will appear later after you run a program. You can have an unlimited number of program edit (or `Editor`) windows, but at most one `Log` and at most one `Output` windows. Output from subsequent SAS runs is added to the end of the `Log` and `Output` windows.

To run a program in SAS, you first write a SAS program in a `Program Editor` window, or else a pre-existing program file and then edit it. The built-in editor is colorized, so that comments, SAS key words, and variable names are in different colors, which makes easier to avoid spelling and simple syntax errors. You then either click on `Run` then `Submit` oe else click on the `icon of a running man`, as mentioned above.

After you run a program, the `Output` window will then generally fill the right half of the screen and the `Log` and `Program Editor` window will disappear. To view a window that has disappeared, either click on `View` on the Main Menu then `Program Editor` or `Log` or `Output`, or, more simply, click on the window's icon on the Taskbar at the bottom of the two main SAS windows. If you select the `Program Editor` window at this stage, your current program will have disappeared. (SAS is waiting for you to enter new program commands, whose output will be added to your `Output` window.) To retrieve your program, click on `Run` (when a Program Editor window is active) then `Recall Last Submit`.

You can also write a SAS program in a text or ASCII editor such as `Notepad` or `Wordpad` on a Windows system or `vi` or `pico` or `emacs` on UNIX systems, and then load the program into SAS by highlighting a `Program Editor` window on clicking on `File` then `Open`. Alternatively, you can load a SAS program file using `Copy and Paste` or `Drag-and-Drop` from `My Computer` or `Copy and Paste` from an internet browser. If you `Copy and Paste` or `Drag-and-Drop` a program file into the `Output` window, the file will be loaded and run automatically and the Output (if there is any) will appear in the `Output` window.

Depending on the version of SAS, you many also be able to click on `myfile.sas` in `My Computer` and have it load or run.

In SAS, the Last Output is First: When you run a program in SAS, the LAST PART of the output will always be shown instead of the beginning. However, you can scroll back to the beginning. If you write more SAS commands in the `Edit` window or load a new program, and then click on `Run` then `Submit` as above, the result will be as if the new commands or new program were added to the end of the previous program. The new output will be added to the end of the `Output` window and the new log-file information will be added to the end of the `Log` window.

If your program has serious errors and has no output, then the `Log` window will jump to the foreground with information about your errors. Again, the LAST PART of the log file is shown instead of the beginning. The log window has a listing of your input file along with error messages and indications of precisely how SAS interpreted your program. This might be very different from how you interpreted it. SAS often has a summary of any serious errors at the end of the log file.

If SAS understands enough of your program to produce any output, then it also writes analyses and printouts to the `Output` window. Statements in your program that SAS could not decipher because of spelling or format errors will generate error messages in the `Log` window but are ignored in the `Output` window.

To "reset" the program so that you can run a new version of your old program and get fresh Output and Log-file information, you must highlight each of the `Output` and `Log` windows and, for each, click on `Edit` on the Main Menu then `Clear All`. However, this does not "reset" the page numbers in the `Output`. To guarantee that `Output` page numbers begin again at Page 1 at this stage, you must either include `pageno=1` in an `options` statement in your program (see the example SAS program below) or else close and re-open the SAS program.

After you write a SAS program, before submitting it to SAS for processing, you should SAVE IT AS A TEXT FILE in case something happens and you need the program later. To do this, highlight the `Program Editor` window, click on `File` on the Main Menu, then on either `Save` or `Save As...`

Commands in Windows SAS: A large number of commands are available through the menu bar at the top of each SAS window. Some of the most useful command sequences are listed below. Again , "`File | AAA | BBB...` " means "click `File` on the main menu, then click `AAA` on the menu that drops down, . . . "

The following commands apply only to the current window, which is generally the last window in which you clicked the mouse.

`File | Open...`	Read a program file into that window
`File | Save As...`	Save that window as a text file
`File | Print...`	Print that window
`Edit | Clear all`	Clear that window
`Run | Submit`	Tell SAS to run that program
`Run | Recall Last Submit`	Bring back a previously processed program
`Help | SAS Help and Documentation`	SAS Documentation

The command `File | Open` and the `Run` menu are only available for program windows.

When you “save a window” by entering “`File | Save As...`”, you do save the entire text associated with that window, but only for that window. In order to save the SAS program, the `Output` window, and the `Log` window, you must `Save` each window separately.

Running a SAS Program. After writing a SAS program in a program edit window and before running it, you should always

(i) RECHECK your program, to make sure that you have not made any obvious simple errors,
(ii) If the program has more than two or three lines, SAVE A COPY OF YOUR PROGRAM to somewhere safe, for example to a floppy disk, the `Desktop`, or to a directory on the computer that you will be able to find later. Then if something happens later, it will be much easier to read in this file (using `File | Open`) than to rewrite the file from scratch. Save the program file using the name `myfile.sas`, for example. Then
(iii) SUBMIT IT to SAS by entering `Run | Submit` .

As mentioned earlier, your program will disappear after step (iii). That is, the program edit window will be cleared, but can be retrieved by entering `Run` (after `Viewing` the Program Editor) then `Recall Last Submit`.

If your program had no serious errors, the `Output` window should jump to the front, as mentioned earlier. In general, you should always check the `Log` file to make sure that there were no errors. You can do this by clicking the `Log` file button on the SAS TaskBar, which will display the last few lines of the `Log`. SAS usually display a summary of errors near the end of the Log file, so that errors should be summarized there.

NOTE: It is usually sufficient to look at the end of the `Log` file, where SAS summarizes the serious errors in the program. If SAS says nothing about errors at the end of the `Log` file, then your program is probably OK.

Warnings: (1) If you use an outside editor to write a SAS program and then load the edited file into SAS, or else if you want to use SAS batch mode, make sure that the file is not saved with invisible formatting characters for changing fonts, centering, and so forth. Sometimes even tab characters can cause problems. SAS will be confused by these invisible characters and you will get a variety of strange error messages.

To avoid these problems, make sure that the file is saved as “Plain Text (*.txt)” or “Text Document (*.txt)” or something similar. DO NOT save it in “Rich Text Format (*.rtf)” or “Word Document” (*.doc or *.docx) format.

(2) Sometimes the `Output` window will jump quite a bit with a slight move of the control bar during scrolling. This is due to embedded end-of-page characters within the `Output` file.

(3) Sometimes it may be useful to clear the `Log` and `Output` windows to make sure that any text in these windows is from your current project. To do this, enter `Edit | Clear all` within both of these windows. That is, highlight the `Log` window and enter `Edit | Clear all`, then highlight the `Output` window and enter `Edit | Clear all` again.

Saving your Program and Output: After you have finished with a program and are satisfied with the program itself and the output, you should save both as text files:

(i) Make the program edit window active by clicking it.
(ii) If the window is blank, enter `Run | Recall Last Submit`.
(iii) Click `File | Save As...`
(iv) In the dialog box that appears, select a directory or else keep the current directory. On your own computer, you may want to select a directory called something like `sas2002` or `MathXXX`. On a computer in a Computer Lab, you could select a floppy disk (drive `A:` on a PC), the `Desktop`, or a directory or folder on the computer such as `C:\TEMP`. You can later copy these files to your ArtSci (or other) computer account by using a file-copy program like `WS_Ftp` or `ftp` or `fetch`. See below for information about using `WS_Ftp`.
(v) Enter `myfile.sas` as the program file name in the dialog box and press `OK`.
(vi) Go through the same steps for the `Output` window and call the output file `myfile.lst`. If you choose, you can also save `myfile.log`.

The extensions `.sas`, `.lst`, and `.log` are the default file extensions in batch mode and are good choices in general. (You may want to use a different name than "`myfile`".)

WARNING: If your program has no valid output, then SAS does not clear the pre-existing `Output` window nor add new material.

If that happens (that is, if your SAS program is "totally wrong"), then the contents of the current `Output` window remain and may be SAS output for a different SAS program, even if that program analyzed different data for a different purpose on a different day. Most SAS users have been misled by this at least once. In particular, it is good practice to always check the contents of the `Log` window before looking at the `Output` window.

SAS Documentation: Clicking on

```
Help | SAS Help and Documentation
```

leads to the SAS online documentation. This has detailed information about the syntax and available options for SAS commands and, in many cases, information about the statistical theory behind the various tests.

The online documentation replaces information that used to be contained in between 10 and 20 large printed manuals, each about 1000 pages or so. However, since the documentation is a distillation of 10,000 to 20,000 pages of text documentation, it can sometimes be difficult to find specific information. Key word searches can leads to hundreds of entries for any topic of interest.

One useful step is to use the `Index` search window to look for entries of the form `XXX Procedure`, such as `FREQ procedure` or `GLM Procedure`. After the search window has stabilized (with perhaps hundreds of entries), click on the first entry for `syntax`. This should take you to an explanation of the syntax and available options for that SAS procedure. There will also be links to explanations of that statistical theory behind the various options. These may be condensed in some cases, but at least will be correct and relatively complete.

SAS in Batch Mode: To use SAS in batch or command-line mode, you first need a command-line or console or terminal or “Command-Prompt” or “DOS-Prompt” computer window. Normally, on a Windows PC, click on

```
Start | All Programs | Accessories | Command Prompt
```

on the Desktop. This will open a console (or “Command Prompt”) window. The default directory will be the home directory for the current account. (The default directory is the directory whose contents you will see if you enter “`dir`” at the command line.)

Normally, command-line windows on Microsoft PCs have white or greyish-white text on a black background, which can be only marginally legible on some computers. To get black text on a white background (as in `Notepad` or most Microsoft programs), enter

```
color f0
```

(this is “color eff-zero”, NOT “color eff-oh”). Then press Enter. The console window should now be noticeably more legible.

PCs in the WashU ArtSci Computer Lab: `Command Prompt` has been removed from the `Accessories` menu on PCs in the WashU ArtSci computer lab, but you can enter

```
Start | All Programs | Math408-Prompt
```

This loads a Command Prompt window that already has black text on a white background and starts in a more accessible default directory `c:\temp`.

In batch mode, you normally use a text editor to write a SAS program (for example, `myfile.sas`) that describes your data and tells what analyses you want performed. (See an example below.)

The file that you create should have the extension "`sas`", as in "`myfile.sas`", and should be saved in the default directory of the Command Prompt window. You can also use any program-file extension that you like, such as "`myfile.cat`". On UNIX terminals or consoles, or telnet windows to most UNIX servers, you enter

```
sas myfile
```

followed by pressing the "Enter" key. Enter `sas myfile.cat` if your input file is named `myfile.cat`. Entering `sas myfile.sas` and `sas myfile` have the same effect. If SAS is installed and the computer can find it, files named "`myfile.log`" and "`myfile.lst`" will appear in the default directory, perhaps after a second or two. (See below if you get an error message instead.) The two files have the same content as the Log and Output windows in Windows SAS. If `myfile.sas` has graphics procedures other than text graphics, a SAS window with the graphic will open as well.

The syntax is more complicated for a Command Prompt or console window on a Microsoft PC. Assume, for example, that the PC has SAS installed in the directory `c:\sasv9` and that the full path for the main SAS executable program `sas.exe` is `c:\sasv9\sas.exe`. Then enter

```
c:\sasv9\sas -sysin myfile
```

to run SAS on `myfile.sas` in the default directory. If the path has any embedded spaces, enclose the path with quotation marks.

On the computers in the WashU ArtSci computer lab as currently configured, the command line should be

```
"c:\Program Files\SAS9.2\SASFoundation\9.2\sas.exe" -sysin myfile
```

This can be automated by writing a batch file with name (for example) `bsas.bat` consisting of the two lines

```
Rem Windows batch file for command-line SAS
"c:\Program Files\SAS9.2\SASFoundation\9.2\sas.exe"  -sysin %1
```

(The first line is optional.) Then entering

```
bsas myfile
```

runs SAS on `myfile.sas`.

If SAS is installed on your computer but you don't know where, enter `Start | Search` from the Desktop and then enter `sas.exe` under "All files and folders". In that case, replace the path `c:\sasv9` above with the path to the directory in which `sas.exe` is located. Again, if the path has any embedded spaces, enclose the path within quotation marks.

SAS often writes an error summary about 5 lines from the end of the `Log` file. A quick way to detect whether SAS has detected errors in a batch-mode program is to enter the command "`tail myfile.log`" in UNIX. The UNIX command `tail` displays the last 10 to 15 lines of any text file. Versions of `tail` and most other UNIX utilities are available for Microsoft Windows machines.

Batch Mode Warnings: (1) If, after you enter `sas myfile` on a UNIX system, your computer issues a number of confusing error messages and then says something like

```
sas: Command not found
```

then the computer could not find SAS. This means either that SAS is not installed on the computer, or else that the computer directory in which SAS is installed is not on the "command path" for your account.

When you try to run a program, the computer does not generally search its entire permanent memory for the program, but just those directories that are listed in the "command path".

On the ArtSci UNIX server, the solution is to enter

```
pkgaddperm sas
```

at the command line, followed by Enter. This adds SAS to your "permanent" command path. To put your permanent command path into effect, you must first log off of the computer and then log in again.

If this happens on computer systems other than ArtSci, then you should talk to a system administrator.

(2) In batch mode, SAS does not delete a pre-existing output file `myfile.lst` if you have no correct output. If that is the case (that is, if `myfile.sas` is "totally wrong"), then the current file `myfile.lst` may be the output of `sas myfile` for a previous version of `myfile.sas`, even if that version analyzed different data for a different purpose several weeks ago. Most SAS users have been misled by this at least once. In particular, it is good practice to always check the end of `myfile.log` before looking at `myfile.lst` .

Transferring Text Files: On a PC, you can transfer text files between the PC and a SAS host computer by using the command-line program `ftp` or else the windowed program `WS_Ftp`. MacIntosh users can use the program `fetch`. The main problem is is that the files MUST BE TRANSFERRED AS TEXT OR ASCII FILES.

Microsoft Windows, UNIX, and MacIntosh computers have different conventions for end-of-line invisible control characters in text files. Transfer programs can generally transfer files in either ASCII (or text-file) or BINARY mode. ASCII tranfers adjust the end-of-line characters appropriately while BINARY transfers copy the file as is. Technically speaking, a text file on a UNIX system (for example) with Microsoft Windows or MacIntosh end-of-line characters is not a UNIX text file, and UNIX programs are not required to treat them nicely.

Many modern computer programs on UNIX, Microsoft Windows, and MacIntosh platforms are intelligent enough to read text files in any of the three formats without any problems. Eventually, all computer programs will behave in that way. However, many current SAS programs are extremely touchy about out-of-place invisible characters of any kind. If you are lucky, you will get a large number of annoying errors. If you are not lucky, you will have incorrect output without any warning messages unless you look carefully in the log file. (PC Windows SAS seems to have fewer problems in this regards than mainframe UNIX SAS.)

Problems usually arise when SAS sees what it views as an unusual control character next to numerical data. This causes SAS to treat that number as part of a text string (because of the invisible character) and then treat that number as missing data because it is a text string and not a number. As you might expect, this can lead to an interesting debugging experience.

The default for file transfers for the UNIX or DOS command-line programs `ftp` is ASCII transfers, which is what you want. This adjusts end-of-line characters in text files to the target computer. Unfortunately, the default for the windowed `WS_Ftp` on Microsoft Windows systems is BINARY transfers. This would be the correct mode for graphics images and for `Microsoft Word` files, but can be disastrous for SAS text files.

You can run `WS_Ftp` on the PCs in the ArtSci computer lab by clicking on the `WS_Ftp` icon on the desktop or else by entering

```
Start | Programs | Ws_ftp | WS_ftp
```

When `WS_Ftp` loads, it presents a screen for you to log on to remote computer account, for example an ArtSci account. You will then see two directories, one listing files in the remote account in the right-hand panel and one listing files on the PC, usually the `WS_ftp` home directory. Change the PC directory to a directory with fewer files, for example `C:\Temp`, so that you will be able to find files after they are transferred.

Finally, BE SURE TO SELECT ASCII AND NOT BINARY TRANSFERS by clicking on the appropriate radiobutton on the `WS_Ftp` screen. If you now click on files in the remote account on the right-hand panel, then they will be copied to the

PC, and vice versa. You can now print them by (for example) loading them into `Notepad` and entering `File | Print` within `Notepad`.

Problems with Tab Characters: Unless SAS is told otherwise, some versions of SAS treat tab characters as unknown control characters. If the tab character is next to a number, SAS will treat the number as part of a text string containing the invisible tab character. If a numerical value is expected, SAS will read it as a missing value.

If you are used to separating numbers on a line of data with tab characters as opposed to spaces, or if you export data from a spreadsheet as a text file, which will then normally have tab-separated columns, and if this version of SAS is tab-unaware, then YOU MUST WARN SAS that the file contains tab characters. You can do this by adding the strange-looking command

```
infile datalines expandtabs;
```

to all "data steps" in your SAS program. (See the example below.) This tells SAS to treat tab characters as spaces, which is what you want. If your SAS program has tab characters and you do not have this command or an equivalent, then you may have errors in your log file, mysteriously missing data, or worse.

Printing Your Output: You should be able to print any SAS window by entering `File | Print` in that window. In SAS batch mode, you can print text files in the same way as you would print any text file, for example by using `print` or `lpr` in UNIX or entering `type myfile > lpt1:` in Microsoft Windows. You can also run `Notepad myfile.lst` or `Notepad myfile.sas` on a PC and then `File | Print` within `Notepad`.

If `File | Print` does not work in windowed SAS, it may be because SAS is saving your output until you exit SAS, at which time all of your output will be printed at one time. This is how printing was traditionally handled in SAS. Printing a program file or output meant adding the text to the end of a local "print file" rather than printing it. The print file was only actually printed when you exited SAS. Some versions of SAS may still behave in the same way.

If you are connected to a UNIX server through an X-windows interface on a Mac or PC, then `File | Print` in a SAS window may direct printed output to a printer near you. However, it probably will not. In that case, you will have to (i) save each window as a text file in your account, which you should do anyway, (ii) transfer it to your local PC or Mac (see "Transferring Text Files" above), and then (iii) print it on your Mac or PC as you would any other text file. You may want to save the files in a subdirectory of your account, for example `sas434` or `sas475` or `mathxxx`, to keep the main directory of your account from becoming too cluttered.

You can then transfer the text files to your local computer (Mac or PC) and print them as you would any text file. (See the preceding sections.) For example, on a PC, you can load them into `Notepad` and enter `File | Print` within `Notepad`.

An Example of Using SAS. The example SAS program on the top of the next page or this page mimics a typical business or database application of SAS. The exception is that we have only 8 records and 3 pieces of information (or variables) for each record. SAS programs in business or medicine may have millions of records and thousands of variables.

If you are using windows SAS, move the mouse pointer to the beginning of a blank program window and enter the `Example SAS Program` in the program window.

Line numbers `0001-0027` may appear in a separate list at the left of the program edit window. BE SURE that you have entered the program exactly as above, with no typographical errors.

We now discuss the structure of SAS program files in general, and the commands that appear in this example in particular.

A SAS program consists of a number of SAS statements or commands along with either some included data or else statements about where SAS can find data. We begin with a short discussion of the syntax of SAS commands.

SAS Commands. Technically, a SAS command or SAS statement consists of a keyword (or verb) possibly followed by one or more modifiers. All SAS commands must end with a semicolon (;). Thus "`options ls=75 ps=66 pageno=1 nocenter;`", "`data list1;`, and "`datalines;`" are examples of SAS commands. You can have more than one SAS statement on one line, or else have one SAS command spread out over several lines. Line structure does not matter for SAS commands.

However, BE CAREFUL TO INCLUDE THE FINAL SEMICOLON IN ALL SAS STATEMENTS! If you leave out a semicolon, SAS will assume that everything up to the next semicolon is part of that command, assuming that SAS can figure out what you are doing at all. Leaving out a semicolon is the most common mistake in writing SAS programs, and will likely result in a series of rude and unpleasant remarks being written to the `Log` file.

The first two lines in the example program are comments. Any text between an initial `*` and the next following semicolon (`;`) is considered to be a comment and is ignored by SAS. In particular, comments have the syntax of a SAS statement whose verb is "`*`" and which is guaranteed to do absolutely nothing (other than tell readers what the program is doing).

The third line (`options....`) is a command that sets some system parameters for printing and displaying output in windows. In many older versions of SAS, the defaults for printing are 120 columns and around 30 lines per page. This produces

```
* Example SAS Program;
* Summary information about employees;
options ls=75 ps=60 pageno=1 nocenter;
data list1;
     infile datalines expandtabs;
     input name $ region $ salary;
     if salary<10000 then income=´Low ´;
     else income=´High´;
datalines;
  Michael    NW     9635
  George     NW    12393
  Helen      SE     9465
  Linda      S     11549
  Rebecca    SE     7398
  Jill       NW    11762
  Margaret   SE    11550
  Bettie     S     10983
run;
* Display the data;
proc print;
     title ´Salaries´;
     run;
* Display a 2x3 table of income levels by region;
proc freq;
     table income*region;
     run;
* Display summary statistics of salary by region;
proc means sum mean min max;
     class region;
     var salary;
     run;
```

awkward output in computer windows and when printed on $8\frac{1}{2}$ by $11''$ paper. The modifier "`pageno=1`" makes sure that page numbers in the output begin with page 1 for each time that you click `Run | Submit`. Many versions of Windows SAS increment page numbers throughout a SAS session, so that consecutive runs begin with larger and larger page numbers. This option has no effect in batch mode. The modifier "`nocenter`" tells SAS to left-justify output. This generally looks better unless you want to post your output on your bulletin board or refrigerator door.

The next part of the program tells SAS what data to use.

SAS Data Sets. The structure of most SAS programs is to (i) define one or more data sets, (ii) act on them by one or more "SAS procedures". (iii) perhaps define more data sets, etc. A "data step" is a part of a SAS program that defines a data set. The 15 lines of the example program between `data list1;` and `run;` constitute a data step that creates a SAS data set. A larger SAS program might read data from a computer file instead of listing it in the program.

In general, a SAS data set consists of one or more records, each of which has data about one or more "SAS variables" such as name, income, region, etc. The best way to think about a SAS data set is as a rectangular array whose rows are "records" (one record per individual) and whose columns are "SAS variables" (like age, income, height, and weight, or name, rank, and serial number). A SAS variable is a particular type of information about each record. Note that this is exactly how the data appears in the sample program. The first column is names, the second is region, and the third is a salary.

In this sense, the structure of a SAS data set is exactly the same as that of a matrix in linear algebra, except that SAS data set columns can contain text instead of numbers. SAS's matrix routines make use of this by, in effect, allowing you to multiply together purely-numerical SAS datasets as if they were matrices. These routines are generally used only if you want to introduce new statistical tests or procedures.

A data step in SAS generally consists of reading lines from a source (such as text in the program or an outside computer file) and reformatting them to form records in the data set. In the example program, the command `data list1;` opens a data set and names it "`list1`". The name is only necessary if you have more than one data set and need to refer to this data set by name later on. In this example, you could replace "`data list1;`" by just "`data;`".

The next command "`infile datalines expandtabs;`" is necessary only if the data in the program has tab characters. If the program has no tab characters, or if your version of SAS is "tab-aware", then this command could be left out.

The third command in the data step "`input name $ region $ salary;`" tells SAS to read three SAS variables from the data and name them `name`, `region`, and `salary`, respectively. The `$` following `name` and `region` tell SAS that these variables hold text. Otherwise, SAS assumes that a variable contain numbers. The data itself is in the lines between the SAS command `datalines;` and the following `run;` command. Line structure is important in actual program data. In this case, the first three (space-separated) words on each data line are assigned to the SAS variables `name`, `region`, and `salary` for each record. If a data line had more than three words, then the extra words would be ignored.

The two lines in the data step that say "`if salary<10000 then income='Low '; else income='High';` define a new SAS variable `income` whose value depends on `salary`. The variable `income` can be thought of as the fourth column in the data set. SAS determines from context that `income` is a text variable and not a numerical variable. Note the extra space in 'Low '. Most versions of SAS assign a buffer on the first instance of a text variable and will not expand it. If this happens, 'High' for income would be recorded as 'Hig'.

A NOTE ABOUT SYNTAX: SAS reads data after a datalines; command line-by-line and stops when it reads any line that contains a semicolon. Thus the line run; after datastep; could have been left out. When SAS reads the line proc print;, it would assume that the data ended on the previous line and would then process proc print; as a command. Similarly, the run; after proc print; is optional since SAS will read the proc in proc means... as beginning another procedure.

If your data actually contains semicolons, for example as part of a name, then you must make special provisions.

In some versions of Windows SAS, a final run; statement in the program is required. Otherwise, SAS will wait for you to write and "submit" further SAS statements. If SAS does not compile in that circumstance, add a final run; statement to the program.

SAS procedures: A SAS *procedure* is a block of SAS statements that begins with proc. The SAS procedure ends with any of (i) a run; statement, (ii) a SAS command that begins with either proc or data, which starts either the next procedure or another data step, or (iii) the end of the program. The first word after proc is the name of the procedure. A SAS procedure acts on one or more SAS data sets to produce output. In more complicated SAS programs, SAS procedures often refer to a particular SAS data set by name. If no data set is specified, they act on the last data set opened, which is list1 here.

SAS procedures can be modified either by options within the actual proc statement or else by separate statements after proc.... within the body of the SAS procedure. The code for proc means illustrates both types of modifiers.

In this example, proc print lists the data set and proc freq will write a 2×3 table of counts for income versus region. The procedure name "freq" is short for "frequency". This comes from the Latin word "frequentia", which means "count" or "crowd" (as of people). Thus proc freq counts things, of which this is a typical example. The proc means procedure in the example program computes summary statistics within each region.

Running a SAS Program. If you entered the program in a program window in windowed SAS, you should first SAVE THE PROGRAM so that you do not have to enter it again. Use the File | Save as... button sequence, as discussed earlier. Save the program as list1.sas (or whatever you want to call it).

Next, submit the program to SAS by entering Local | Submit. (This corresponds to entering sas list1 in batch mode.) If your program has no errors (for example, if you did not make any typographical errors in entering the program above), then the `Output` window will come to the front. You will now be viewing the LAST PAGE of the output. This has the same information as the LAST FEW LINES of the file `list1.lst` in batch mode. Use the scroll bar on the right to move to the BEGINNING of the output. Then scroll through the output from top to bottom.

The output in this case will be composed of three logical pages corresponding to the three SAS procedures or "SAS procs" in the program. (SAS procedures often write more than one output page.) Each logical page in the `Output` ends with a "new page" control character, which may cause the output to jump if you scroll through it continuously.

The first page is the output of `proc print`, which just lists the data set. The second page is the output of `proc freq`, and the third page comes from `proc means`. See the output file `list1.lst` on the Math434 and Math475 Web Sites for the output itself or, better yet, enter the program into SAS and run it yourself. Both `proc freq` and `proc means` can also do much more complicated things with the proper arguments. (The program `list1.sas` on the Web site also writes a scatterplot.)

Correcting Mistakes: Even the most carefully written program can have errors. As an example of an error, assume that you left out the line with with the statement `datalines;` in the sample program.

After the program is submitted, the program window will likely remain in front with the program area blank. If the `Log` window is not visible, make it appear by clicking on the `Log` button on the SAS TaskBar. (In batch mode, read the `Log` file `myfile.log`.) Just after the line "`Michael...`" in the Log window or Log file, there will be a flurry of error messages as SAS tries to interpret the line "`Michael...`" as a SAS command.

Eventually, an error message like "ERROR: No CARDS or INFILE statement" will appear. This indicates that SAS could find no data to read. ("CARDS" is the same as "`datalines`" and is a carry-over from previous times when data was entered on punched cards. An "INFILE" statement can tell SAS to fetch data from an outside text file.) There will be no output file, since SAS could find no data to analyze.

To fix this error, retrieve the program by entering `Run | Recall Last Submit`. Move the mouse pointer to the end of the line beginning with `else income=...` Click it to tell SAS that you want to enter something there. Enter `datalines;` at this point. SAS commands like `data...` and `input...` and `datalines;` are "free-form" and can be entered in any order in any number of lines. Save the file (so that you have a correct copy), and enter `Run | Submit` to resubmit it. Hopefully, there will be no more errors, and the `Output` window will jump to the foreground.