

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 25, 2002

Introduction. SAS is a powerful computer package that is used in many different fields for analyzing and organizing statistical data. It is easy to use for small, medium, large, or very large data sets. SAS can print out summaries of data, draw graphs, carry out statistical tests, estimate statistical parameters, compute P-values, and many other things. The disadvantages are that you have to communicate with SAS in an arcane language that takes some learning. 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, PowerMac, and UNIX computers. Nowadays most people use windowed versions of SAS, but there are also command-line or DOS-Prompt versions that are accessible over a modem or terminal interface. 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 and about how to run SAS in batch mode. However, most of what we say below applies to Windows SAS or SAS in general, for example for the PCs in the Olin Business School computer lab.

Running SAS: In all cases, communications with SAS are composed of three 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 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) along with other information about what it was thinking, and
- (iii) *SAS Output File*: Written by SAS, the results of the analyses that you requested, or, more precisely, the results for your requests that SAS could figure out.

This 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 or copied into other programs.

In windows mode, the three parts are in three different windows. You write your program in one window and click on a button to “Submit” it. SAS then fills in the Log and Output windows with the appropriate information. You can also write the input or program file with a text editor that is not part of SAS and import it into a program window.

In batch mode, the three parts are three different text files. You write the input file using a text editor or word processor (such as `Notepad` or `Microsoft Word` on a Microsoft Windows computer) and submit it to SAS by entering a command at the keyboard. SAS then writes two text files in response, a log file and an output file, although the output file will be missing if SAS feels that there is no output to write.

If you have an ArtSci computer account at Washington University, then you can use SAS batch mode from either a terminal or over a modem. You write SAS programs using either a UNIX text editor or else a Microsoft or MacIntosh machine and then transfer the program file using `ftp`, `WS_Ftp` (for PCs), or `fetch` (for MacIntosh computers). We first discuss using SAS in windows mode (mostly for PCs) and then SAS batch mode.

SAS in Windows Mode: To use windows SAS on the PCs in the ArtSci computer lab, start from the PC opening screen and enter

`Start | Programs | The SAS System`

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 `The SAS System`. Windows SAS is only on the PCs in the ArtSci lab and not on the MacIntoshes.

You will then see a drop-down menu with the two choices (in some order):

`Online SAS Help`
`The SAS System for Windows V8`

The first choice is the online version of the SAS program documentation. This is a complete HTML version of the 10 or 20 large SAS printed manuals that, together, take up around 5 feet of shelf space and perhaps tens of thousands of pages. We will give information about using this resource below.

The second choice is windows SAS itself. Incidentally, the `V8` stands for Version 8, not the popular vegetable drink.

Opening Windows SAS: Click on **The SAS System for Windows V8** to open SAS. Most windows versions of SAS look very similar whether they are on MacIntosh, Microsoft Windows, or UNIX machines. All versions have a Log window, an Output window (which may be missing unless you have output), and one or more program windows.

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. The top stacked window is named **Log (Untitled)**. The lower stacked window is named **Editor (Untitled 1)** and can be used to enter a SAS program or input file. An **Output** window will appear later when you have output. 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.

The **Editor (Untitled 1)** program window is initially empty. As you load SAS, you may notice that the program is writing various copyright and initialization information to the “Log” window. In particular, you will be told how much time it took for SAS to load itself, perhaps around 0.29 seconds. In the old days when shared computer time was more expensive, this might mean that you had already spent a great deal of money.

Each SAS window can be expanded to full size by clicking on a symbol that looks like an open box on the upper-right corner of each window. This will completely hide the other windows. However, any window can be brought to the foreground by clicking the name of that window on the SAS TaskBar underneath the displayed SAS windows.

Creating a SAS Program. The first step in using SAS in windows mode is to write a list of commands or instructions (called a “SAS program”) in a program window such as the window with title **Editor (Untitled 1)**. After you write the program, you run it by entering

Run | Submit

on the SAS main menu. (See below for an example program. Again, the notation | means “then”. That is, you use the mouse to click on **Run** and then, on the drop-down window that appears, on **Submit**.)

If your program has any correct output, an **Output** window will jump to the foreground with the program output. If for some reason it does not, you can bring it to the front by clicking on the appropriate button on the SAS TaskBar at the bottom of the screen. Curiously, the **LAST PART** of the output will always be shown instead of the beginning, but you can scroll back to the beginning. The program window will be cleared as soon as you click **Submit**, but you can click on **Run | Recall Last Submit** to get it back.

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. 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.

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 (see the next section for how to do this).

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 file into that window
File Save As...	Save that window as a text file
File Print	Print that window
File Exit...	EXIT the SAS Windows environment
Edit Clear all	Clear that window
Run Submit	Tell SAS to process that program
Run Recall Last Submit	Bring back a previously processed program
Help SAS System Help	Documentation for Windows features
Help Books and Training SAS Online Doc	If implemented, the SAS online program documentation
Tools Text Editor	Open a new program edit window

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 save the entire text associated with that window, but only for that window. If you want to save the contents of all three windows, you must **Save** each window separately.

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 (see step (vii) below for more details). 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** .

Your program will disappear after step (iii). That is, the program edit window will be cleared. (Aren't you glad that you saved it first as a text file?)

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 these by clicking the **Log** file button on the SAS TaskBar. It is generally 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.

If you need to retrieve your program later (for example, to correct the errors listed in the **Log** file), enter **Run | Recall Last Submit** from the SAS Main Menu. This only works if a program edit window is highlighted. If you have deleted all program edit windows in the meantime and you see only **Log** and **Output** windows, then click on **Tools | Text Editor** to create a new program edit window and then enter **Run | Recall Last Submit** .

WARNINGS: (1) SAS adds material to the end of existing windows and sometimes only displays the top part of that window (with the old material). In that case, use the scroll bar on the right of the window to move to the end of the window. 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.

(2) 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.

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


SAVING YOUR PROGRAM AND OUTPUT:

- (iv) Make the program edit window active by clicking it.
- (v) If the window is blank, enter **Run | Recall Last Submit** .
- (vi) Click **File | Save As...**
- (vii) 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`.
- (viii) Enter `myfile.sas` as the program file name in the dialog box and press `OK`.
 - (ix) 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 is the case (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 Online Documentation:** On the PCs in the ArtSci computer lab, clicking on

Start | Programs | The SAS System | Online SAS Help

leads to the first page of the SAS online program documentation. The Math439 and Math475 Web sites have a link to the same documentation on the main SAS Web site. (At the SAS Web site, use the user name `onlinedoc` with password `sas`.)

Since this documentation is the contents of between 10 and 20 large printed manuals, finding what you want is not always easy. Click on

Master Index

near the top of this page to get to an index for all ten thousand or so pages. This is comprehensive, but often has 50 to 100 entries for any topic of interest. For an index for topics that are more specific to statistics, scroll down to the paragraph beginning with `SAS/STAT` and click on `SAS/STAT User’s Guide`.

You will now see the chapter headings for the two large SAS/STAT manuals. The first ten chapters or so are devoted to statistical theory. This covers many interesting topics, but can be quite technical. Following are chapters devoted to each of SAS’s main statistical procedures. A few SAS statistical procedures are considered add-ons to basic counting and tabulating procedures and are documented

elsewhere. (Look for the paragraph that begins SAS/PROC instead of SAS/STAT. PROC stands for Basic Procedures.) The syntax for handling data within SAS programs is explained in the SAS/LGREF (Language Reference) manual, generally in much more detail that you would want to know. However, most of what you will be interested in is in the SAS/STAT manual with some additional topics in SAS/PROC.

Scroll to the end of the **SAS/STAT User's Guide** page and click on **Index**. This leads to a smaller but more usable index for SAS's statistical procedures only. If you have windows SAS installed on your own computer or laptop, you might find it useful to have desktop shortcut icons for both the **Master Index** and this more specific **SAS/STAT Index**.

SAS's routines for manipulating matrices are explained in the

SAS/IML User's Guide

linked from the main SAS documentation page. Here IML stands for "Interactive Matrix Language". These routines are not very interactive (you still have to write SAS programs), but this is one of the most powerful matrix packages available.

What Text Editors to Use. If you want to write a SAS program in an outside editor and then load it into SAS, or else if you want to use SAS batch mode, then you will need to use a text editor. You can use any word processor that writes "plain text" or "ASCII" or "non-document" files. If the program file contains invisible format characters for font changes or centering — or sometimes even tab characters — then SAS will find it different or impossible to read the file without help.

Two widely-used text editors on UNIX systems are **vi** and **emacs**. You can use **Notepad** or **Wordpad** on Microsoft Windows computers. General-purpose word processors like **Microsoft Word** or **WordPerfect** or **WordStar** are fine as long as you make sure that they write their output files in "ASCII" or "text" or "non-document" mode. In **Word**, this means that you should save files as "Text Only with Line Breaks".

Specifically, you must make sure that (i) no invisible formatting characters other than end-of-line characters are written to the output file and (ii) each line on the screen ends with an end-of-line character. In **Microsoft Word**, "Text Only" can write some lines on the screen without end-of-line control characters. This will have no effect on lines with SAS commands, but can lead to confusing errors in program lines with data.

SAS in Batch Mode: To use SAS in batch or command-line mode, you first need a command-line or console or terminal or "DOS-Prompt" computer window. If you called up a computer using a terminal program like **telnet** or a

modem program, then you will probably be in command-line mode already. On a Windows PC, starting from the opening screen, you can enter something like

```
Start | Programs | Command Prompt
```

(You may have to replace `Command Prompt` by `DOS-Prompt` or `Accessories | ...`.)

In batch mode, you use a text editor to write a “SAS input file” or “SAS program” that describes your data and tells what analyses you want performed. (See the previous section and also the example below.)

The file that you create should have the extension “sas”, as in “myfile.sas”. However, you can use any extension that you like, such as “myfile.cat”. One command-line oriented computer systems (such as the ArtSci UNIX server), enter

```
sas myfile
```

on the command line 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, then files named “myfile.log” and “myfile.lst” will appear, perhaps after a second or two. The file `myfile.log` has a listing of `myfile.sas` along with error messages, and will be the same as the contents of the Log window in Windows mode. If SAS understood enough of `myfile.sas` to produce any output, then it will also write a file “myfile.lst” with analyses and printouts. Statements in your input file that SAS could not decipher because of spelling or format errors will generate error messages in `myfile.log` but will generate no output in `myfile.lst`.

On a Microsoft Windows computer with PC SAS installed, assume that the main SAS executable program `sas.exe` is in the directory `d:\sasv8` (for example). Assume that you are in a command-line window and “myfile.sas” is in the default directory. (This means that you will see an entry for `myfile.sas` if you enter “dir”. If SAS is installed on your computer but you don’t know where, you can enter `Start | Find` to find out.) Then entering

```
d:\sasv8\sas -sysin myfile
```

(followed by Enter) runs SAS in batch mode on `myfile.sas`. Two SAS windows will appear briefly and then files `myfile.log` and `myfile.lst` will appear in the default directory.

A frequent user of batch mode in PC SAS might define a batch file to automate this process. If you create a text file named (for example) `bsas.bat` with the lines

```
rem This is a batch file to run SAS in batch mode
d:\sasv8\sas -sysin %1
```


then entering `bsas myfile` at the command line will call SAS in batch mode. The `rem` line in `bsas.bat` is optional and can be left out.

SAS usually 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`.

Which is Better: Batch or Windows Mode? A primary advantage of SAS batch mode is that you can use your own text editor. Batch mode can be much faster to use if you are used to manipulating text files. The editor that is built into SAS Windows mode is inconvenient for complex programs, although it does allow you to import text files that you have written elsewhere. If you are

experienced in entering, using, and exiting a text editor quickly, then SAS batch mode may be faster and more convenient than Windows mode.

The SAS Windows environment is more convenient if your SAS input program is short or if you do not have to do extensive editing. The online documentation under **Help | SAS System Help** is useful for the syntax of some SAS commands and procedures and what they do, but are not as complete or easy-to-read as SAS manuals. Windows environments usually have built-in text editors, but they are not as powerful as **vi** or **emacs** or most Mac or DOS or Microsoft Windows text editors. Many experienced SAS users prefer SAS batch mode because (i) they can use a more powerful editor and (ii) exactly the same environment is available over a dial-up modem and on a wide variety of different computers. If you have a text editor that you are comfortable with, then SAS batch mode may be easier for you.

X-windows Interfaces: You can also run windows SAS on a UNIX machine that is connected to a PC or MacIntosh. Specifically, in addition to PC Windows SAS on the PCs in the ArtSci computer lab, you may be able to run Windows SAS directly on the ArtSci UNIX server. This may be preferable for extremely large SAS programs, or if you want exactly the same output as you would get using SAS in batch mode in your ArtSci UNIX account. To use Windows SAS on the ArtSci UNIX server, enter

```
Start | Programs | Exceed | SAS (artsci.wustl.edu)
```

from the PC starting screen, where | means “then” as before.

You will then be presented with a window that asks for a User name and Password for an ArtSci computer account. You cannot proceed beyond this point unless you have an ArtSci account. Arts & Sciences students at Washington University automatically have ArtSci accounts. If you do not know your User name and Password for your ArtSci account, a local consultant will be able to tell you what they are.

After you log on, the SAS Windows environment will load into your ArtSci account and display on your computer. SAS may look as if it is running on your computer, but it is actually running on the ArtSci UNIX server through an “X-windows” connection.

The main difficulties that arise now is if you want to transfer data from the PC to SAS on the UNIX server, or else print SAS output. There is currently no way to print directly from the UNIX server, which is in a different building. In either case, you will have to transfer text files either from the PC to your ArtSci account or from your ArtSci account to the PC. (See the next section.)

EXITING FROM X-WINDOWS SAS: If you have connected to SAS on UNIX server through an X-windows, do not just exit SAS and leave. That would leave the connection to your UNIX account still open so that the next person using

the computer could use it. The best way to exit is to close the **Exceed** program that provides the X-windows interface.

To exit **Exceed**, look for a window with an array of bright, strange-looking icons that says **Exceed** at the top of the window. Click on the icon that looks like a stop sign. When you confirm that you want to quit the X-windows connection, all SAS windows will disappear.

T**ransferring 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 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 transfers 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, many version 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 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, then YOU MUST WARN SAS that the file contains tab characters. In most cases, you 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, the you will 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.

```

* 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;

```

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. SAS’s defaults assume 120 columns and around 30 lines per page. This produces awkward output in computer windows and when printed on 8½ 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`. Some versions of windows SAS increment page numbers throughout a SAS session, so that each run begins with a different page number. There is 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 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, 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`. 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``. Some 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 earlier versions of windows SAS, a final `run;` statement in the program was required. Otherwise, SAS would wait for you to write and “submit” further SAS statements. I have not seen SAS Windows environments behave this way recently.

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 `proc`, which starts the next procedure, (iii) the `data` verb of the next data step, or (iv) 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 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 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.