Using SAS on UNIX

Last revised: 10/09/00

SAS is a comprehensive Information Delivery System that provides a wide variety of applications including:

- data access
- data management
- data analysis
- data presentation

SAS provides several user interfaces ranging from a Windows Environment for novices to batch programs for advanced users.

SSCC runs version 8 of SAS.

Invoking SAS

SAS can be invoked simply by typing the word "sas" at the UNIX prompt. To take advantage of the many options available with the command, use the general form of the SAS command as follows:

```
sas filename -option1...-optionn
```

where `filename` gives the name of the file containing the SAS program to be executed. Specifying a filename on the SAS command invokes a noninteractive session.

`options` specifies a SAS system option to configure your session. Some options include:

`-linesize n` Specifies the line size of the SAS output. The range of linesize is 64 to 256. The default is 132 for noninteractive and batch modes.

`-nodms` Specifies that an interactive line mode SAS session be invoked.

`-obs n` Specifies the last observation from a data set that SAS is to read.

`-pagesize n` Specifies the number of lines that can be placed in a page of SAS output. Values can range from 15 to 32,767. The default is 60 for noninteractive and batch modes.

`-memsize n` Specifies the maximum amount of memory a procedure call may use. The default is 32m. Use caution when specifying this option.

`-log file` Specifies that SAS write the log of the SAS session to "file". By default, the SAS log is written to the file name `filename.log` where `filename` is the name of the file containing the SAS commands.

`-print file` Specifies that SAS write the SAS output to "file". By default, the SAS output is written to the file name `filename.lst` where `filename` is the

name of the file containing the SAS commands.

Example:

```
sas test1 -obs 0 -noreplace
```

This command executes the program `test1.sas`, the file containing the SAS commands, with the system options OBS and NOREPLACE. These two options used together are a good way of checking to make sure your syntax is correct in your file containing the SAS statements. It causes SAS to execute each DATA and PROC step in the program without actually reading any of the data.

Modes of Execution

You have four ways of running SAS: interactive SAS Windowing Environment, interactive line mode, noninteractive mode, and batch mode. The following examples show how to invoke SAS in each of these modes.

Interactive SAS Windowing Environment Mode

Typing `sas` at the UNIX prompt brings up SAS in the Windowing Environment mode. The SAS Windowing Environment is an interactive windowing system that enables you to write and modify your programs, run them, and monitor the output. You can use menus or type commands within the SAS Windowing Environment. Choose File: Exit from the menu to terminate the SAS Windowing Environment and returns control to the operating system.

Note: If after typing `sas` on the command line and you see a `1?` prompt on your screen, you are **NOT** in the SAS Windowing Environment but in interactive line mode. Typing `endsas;` at the `1?` prompt will take you back to the UNIX prompt. The SAS Windowing Environment is only available from X-displays.

Interactive Line Mode

Typing this command:

```
> sas -nodms
```

at the UNIX prompt brings up SAS in interactive line mode. A `1?` prompt appears on your screen and you can begin entering SAS statements. After each statement a line number prompt appears. The `ENDSAS;` statement terminates SAS interactive mode and returns control to the operating system.

Noninteractive Mode

To invoke SAS in noninteractive mode, enter the SAS command followed by the name of the file containing the SAS program to be executed. For example, suppose you have stored your SAS statements in a file named `foo.sas`. To invoke SAS in noninteractive mode and execute the program `foo.sas` you would type the following:

```
> sas foo
```

Note that you do not have to include the file extension in the filename when the file extension is `.sas`. SAS uses `.sas` by default.

You do not get another UNIX prompt until SAS finishes executing the program. When SAS finishes and you get the UNIX prompt, two new files are in your working directory which contain the SAS output. `foo.log` contains the log of the SAS session and `foo.lst` contains the output from the SAS commands in `foo.sas`.

Note: If you have stored your SAS statements in a file which has some file extension other than `.sas`, the log and lst files that are created will have filenames that include the extension. For example, if your command file was named `foo.ext`, the log file created would be `foo.ext.log` and the output file would be `foo.ext.lst`.

Routing Output

In the example just shown, SAS created two files: one to hold the SAS output and the other one for a log of session messages. If you want to direct your output and log to other files, use the PRINT and LOG system options. For example,

```
> sas foo -print report -log report.log
```

The output goes to file `report` and the log goes to file `report.log`.

Batch Mode

To execute a program in batch mode, you simply run it in the background by typing an `&` at the end of the command. For the `foo.sas` example above, type the following:

```
> sas foo &
```

The only difference between running batch and running noninteractive is that in batch mode, your job is executed in the background, meaning you do not have to wait until the SAS program finishes execution before you get the UNIX prompt. In other words, your shell is available for other work. Submit only one SAS background job at a time.

SAS Programs

SAS programs consist of one or more steps made up of instructions called statements. The steps are always one of two types: DATA steps or PROC steps. Below is an example of a simple SAS program that reads in data (DATA step) and then prints it out (PROC step):

```
data study;
   infile "~mcdermot/sasstuff/weight.dat";
   input patient $ pre_wgt post_wgt;
run;
proc print;
   var patient pre_wgt post_wgt;
```

```
run;
```

It is good programming practice to separate steps with a RUN statement.

SAS Statement Syntax

A SAS statement is an instruction that tells SAS to perform some task or gives it information. Most SAS statements begin with a keyword. Keywords tell SAS to expect certain instructions or information to follow. In the example above, DATA and INPUT are keywords.

You must signal the end of each SAS statement with a semicolon. One of the most common mistakes new SAS users make is forgetting the semicolon.

You separate elements within statements with one or more blanks. The number does not matter. Some elements are separated by special characters, depending on the requirements of the statement.

You can write statements:

- in uppercase, lowercase, or a mixture of the two,
- over more than one line,
- with more than one statement on a line.

All of the following are acceptable ways to write the same SAS statement:

```
class STYLE BEDROOMS;

class
      style
      bedrooms;

class style bedrooms; var price;
```

Data Access

SAS can access data from a variety of sources including:

- raw data ready for data entry,
- data stored in files formatted by other software products, such as database management software,
- data stored in other file formats including hierarchical files, variable-length fields and records, and multiple records.

Once SAS accesses and reads the data, it organizes them into a SAS data set. The data values are arranged in a rectangular table-like structure. The columns of the table are called variables and the rows are called observations.

SAS data sets also contain descriptor information. During processing, SAS reads the descriptor information to determine the attributes of the data set and the number and characteristics of each variable.

SAS provides several ways to get your data into SAS data sets. One way is to produce a program consisting of SAS statements, which instruct SAS to perform specific tasks. The following SAS statements are used to create a SAS data set:

- the DATA statement,
- the INPUT statement,
- the CARDS or INFILE statement.

A DATA statement instructs SAS to create and name a SAS data set. DATA statements begin with the keyword DATA and specify the name you select for the data set.

`data study;` instructs SAS to create a SAS data set named STUDY. SAS data set names must begin with a letter and contain eight or fewer characters consisting of letters, numbers, or underscores. Note: Version 8 of SAS allows up to 32 characters for data set names.

The INPUT statement provides the information SAS requires to organize data into a SAS data set, such as the variable's name, type, and, if necessary, column location. INPUT statements follow the DATA statement as in the following example:


```
data baseball;
   input player $ 1-30 atbats 35-40 errors 42-46 salary 50-60;
```

This is an example of using column input format; i.e. the data were entered by placing variables in specified columns. The statement begins with the keyword INPUT and is followed by a variable name, PLAYER. The `$` indicates that the variable is character (contains letters or other non-numeric characters) and is placed after the variable name. 1-30 specifies the column location for the variable PLAYER.

Another method of entering raw data is by listing variable names without column locations. This method is referred to as list input format. Entering data in list input format is simpler than using column input format but places specific restrictions on your data. These restrictions include:

- Variable values must contain 32 or fewer characters (8 or fewer characters with version 6).
- Each data value on each data line must be separated from the next value by at least one blank column.
- Missing values, both character and numeric, must be represented by single periods.
- Numeric values must include any necessary decimal points.

Below is an example of an input statement using list input format.

```
input patient $ trt day response;
```

The CARDS statement informs SAS that data lines immediately follow. The CARDS statement follows the INPUT statement, as follows:

```
data repeated;
   input patient $ trt day response;
   cards;
   Ege  1 1 .25
   Ege  1 2 .01
   Kulkarni 2 1 .75
   Kulkarni 2 2 .34
   ;
```

The single semicolon marks the end of the data lines.

When the raw data are stored on disk or tape, you must tell SAS where to find the data by using an INFILE statement. In its simplest form, the INFILE statement begins with the keyword INFILE followed by the name of the file. As the following example illustrates, INFILE statements precede INPUT statements.

```
data consult;
   infile "~/data/mydat.dat";
   input client $ answer time;
run;
```

In the above example, the INFILE statement instructs SAS that the data reside on disk in `~/data/mydat.dat`.

Permanent SAS Data Sets

SAS creates two types of SAS data sets: temporary and permanent. A temporary SAS data set exists only for the duration of the current SAS program or interactive session. Therefore, data stored in temporary SAS data sets cannot be retrieved for use in later sessions; the data set must be recreated each time you begin a new session.

A permanent SAS data set exists after the end of the current program or interactive session. Data stored in permanent SAS data sets can be retrieved for use in future programs or sessions.

Data set names of the form STUDY, REPEATED, and BASEBALL as in preceding examples, are temporary data sets. When assigning permanent data sets, you specify a *two-level* name such as SAVE.STUDY or RESEARCH.REPEATED. The first part of the name is called the libref. A libref is the name by which you reference the directory containing the SAS data sets.

Actually, temporary data sets also have two-level names. SAS automatically assigns a libref of WORK to temporary data sets. For example, when you submit DATA STUDY; SAS creates a temporary data set named WORK.STUDY. Subsequently, you need use only the one-level data set name when referring to temporary SAS data sets, because SAS already assumes the default libref WORK.

When you create a permanent SAS data set, you must specify both the libref and the data set name. You must specify a libref other than WORK because SAS reserves that libref for temporary SAS data sets. Use a LIBNAME statement to assign a libref that tells SAS where to find the directory that contains or will contain the permanent SAS data sets.

For the example SAS data set, RESEARCH.REPEATED, an example LIBNAME statement might be:

```
libname research '~/sasdata';
```

The keyword LIBNAME is followed by the libref (the first-level of the two-level SAS data set name). The quoted string is the directory containing the SAS data sets you want to access.

Below is an example that creates a permanent sas data set and writes it to `~/sasdata`:

```
libname research '~/sasdata';
data research.repeated;
infile "~/dissert/study1.dat";
input patient $ trt day response;
run;
```

When referring to a permanent SAS data set, use the full two-level name, that is, both the libref and the data set name. For example, assume you have submitted the SAS code in the example above to create a permanent SAS data set. In order to obtain a plot of RESPONSE*DAY in a later SAS program, you could submit the following code:

```
libname research '~/sasdata';
proc plot data=research.repeated;
   plot response*day;
run;
```

Note that a DATA step is not needed here because the data were read into a permanent SAS data step in the previous SAS program. It is only necessary to specify a LIBNAME statement and then refer to the data set in the PROC step.

There are several advantages to working with permanent SAS data sets:

- SAS reads permanent SAS data sets faster than raw data files.
- Permanent SAS data sets are self-documenting.
- Permanent SAS data sets reflect transformations applied to the data.

There are two disadvantages to using permanent SAS data sets:

- Permanent SAS data sets cannot be read directly by software other than SAS or other software for converting data to other proprietary types like STAT/TRANSFER or DBMS/COPY.
- Permanent SAS data sets can only be read by SAS on the operating system they were created on. **Note that with version 8 of SAS this is no longer true.**

Version 8 permanent SAS data sets are portable across operating systems.

The second disadvantage (and the 1st if you want SPSS to read your SAS data set) can be overcome by converting your permanent SAS data set into a transport file.

SAS Transport Files

At some point it may become necessary to read a permanent SAS data set on some operating system other than the one that created the data set. The process of moving a SAS file from one operating system to another is referred to as *transporting* the file. In order to transport a SAS data set, you must first convert it to a format that can be recognized by the operating system that will read the SAS data set. In fact, even if you are not planning to transport a SAS file, it is still an excellent idea to convert the SAS data set to a transport file before moving the data to some other location like a 4mm DAT tape or CD.

Note: New to version 8 of SAS, a permanent SAS data set can be read on any operating system without converting the data set to a transport file.

Converting a SAS data set to a transport file requires the following three steps:

1) Issue a LIBNAME statement with the XPORT engine to identify the transport file that is to be created. The following example illustrates a LIBNAME statement for a transport file to be saved on disk with a filename of archive.dat. The libref used is TRANFILE:

```
libname tranfile xport "archive.xpt";
```

Note that the LIBNAME associated with transport files must contain a fully qualified path name, not just a directory.

2) Issue a LIBNAME statement to identify the directory containing the SAS data set(s) you want to convert into a transport file. This LIBNAME statement will be similar to the one described in the previous section:

```
libname old '~/sasstuff';
```

3) Use the COPY procedure with a SELECT statement to create the transport file from the SAS data set. The SELECT statement is where you specify the SAS data set you want to convert to a transport file. If you omit the SELECT statement, SAS will create a transport file containing all of the SAS data sets in the specified directory. This is a convenient way of converting a lot of SAS data sets at once.

   The following example illustrates a PROC COPY step that creates a transport file from the SAS data set DISSERT, located in the directory referenced by OLD in a LIBNAME statement defined in step 2, above. TRANFILE refers to the libref defined in step 1, above:

```
 proc copy in=old out=tranfile;
select dissert;
run;
```

Once you have created a transport file, you can then transfer the file to any other operating system and read the file into SAS (called *importing* the file). *Make sure to use binary mode when transferring the file.* The SAS statements for importing a SAS transport file are almost identical to the ones above except the IN= and OUT= information on the PROC COPY is reversed. Below is an example:

```
 libname tranfile xport "archive.xpt";
 libname sasfile '~/sasstuff';
 proc copy in=tranfile out=sasfile;
select dissert;
run;
```

Writing External Files from a SAS Data Set

Suppose you need to create an external file from a SAS data set so you can process the records with another software package. The FILE and PUT statements may be used for this purpose. The FILE statement tells SAS where to put the data and the PUT statement tells SAS how to organize the data in the external file. FILE and PUT statements follow the same rules as the INFILE and INPUT statements discussed earlier in this document.

The following example retrieves the SAS data set, research.repeated, created in the example above, and from it, creates an external data file in the `~/rawdata` directory named `repeated.dat`:

```
libname research '~/sasdata';
data temp;
   set research.repeated;
   file '~/rawdata/repeated.dat';
   put patient $ 1-10 trt 12-14 day 16-18 response 20-26;
run;
```

The SET statement is used to retrieve data from existing SAS data sets.

Reading and Writing Compressed Data

To read or write external files containing compressed data, you cannot reference the data directly with the INFILE or FILE statement like was illustrated above for uncompressed data. An extra SAS statement, called FILENAME, is required. The FILENAME statement takes the following general form when used to read or write compressed data:

```
filename fileref pipe 'UNIX-command';
```

where `fileref` is the name by which you reference the file and `UNIX-command` is the name of a UNIX command, executable program, or shell script to which you want to

route output or from which you want to read input.

`zcat` is the UNIX command used to write out a compressed file in uncompressed format. Use `zcat` in conjunction with the FILENAME statement to read compressed data as in the following example:

```
filename raw pipe 'zcat compresd.dat';
data one;
   infile raw;
   input var1-var3;
run;
```

Note that the INFILE statement refers to the data file only indirectly using the fileref (RAW) assigned in the FILENAME statement.

`compress` is the UNIX command used to compress raw data. Use `compress` in conjunction with the FILENAME statement to write out compressed data as in the following example:

```
filename out pipe 'compress > outdata.Z';
data two;      set one;
   file out;
   put var1-var3;
run;
```

Note that the FILE statement refers to the data file only indirectly using the fileref (OUT) assigned in the FILENAME statement.

For a complete discussion of compressed files, refer to SSCC Publication #7-5, "Using Compressed Files".

You can also read compressed SAS transport files directly in your SAS program. This is documented in *How to Read Compressed SAS Transport Files Directly in your SAS Program* (SSCC How-to #28). You can also read and write compressed SAS data sets. This is documented in *How to Write/Read UNIX Compressed SAS Data Sets* (SSCC How-to #1).

SAS Documentation

SAS Institute provides documentation for over 40 of its manuals online with version 8. You can access them at http://www.ssc.wisc.edu/saspdf/. There are two versions of the documents: an HTML version for navigating and browsing and a PDF version for printing.
The only printed manual SAS Institute sent with version 8 is:

The Little SAS Book, Second Edition (for Version 8 of SAS)

Below is a short list of printed SAS version 6 manuals that you may find useful:

SAS Language and Procedures: Introduction, Version 6

SAS Language and Procedures: Usage (Volumes I and II)

SAS Companion for UNIX Environments: Language, Version 6

SAS Companion for UNIX Environments: User Interfaces, Version 6

All of the above manuals (plus many more) are available for short term loan in the CDE Print Library, Social Science 4457.

SSCC staff have prepared a self-instructional workbook for learning SAS called *SAS Workbook for Writing SAS Programs to Process Data on UNIX*. This document is available in Soc. Sci. 2470 and 7413 and is free of charge. The on-line version may be found at http://www.ssc.wisc.edu/Sscc/Pubs/PDF/sas.pdf

For features new to SAS Version 8, visit
http://www.sas.com/service/library/onlinedoc/v8/whatsnew/.

To subscribe to the SAS listserver, visit http://www.stattransfer.com/lists.html. This web site provides a subscription service to all the major statistical software listservers including SAS. The SAS listserver provide a depth of information and support that is essentially impossible for staff at any one institution (like ours) to duplicate.