Introduction

How to Use This Document

The examples given in this document are for using SAS under a Unix environment. We assume you are familiar with basic Unix commands and at least one of the editors available in Unix. We also assume you have basic statistical knowledge. This document is not intended to substitute for the vendor-supplied SAS documents. The term SAS refers to the software command language, and the basic command structure is the same across all platforms for SAS products.

This document is intended to introduce researchers to using SAS software from the Unix environment. At present there are several variations of the Unix system. University Information Technology Services (UITS) at Indiana University, Bloomington, offers such Unix-based operating systems as Solaris, and AIX. This document assumes you know the basics of UNIX computing. To learn more about Unix, see Getting Started with UNIX. You may also enroll in an UITS STEPS or PROSTEPS class by contacting the UITS IT Training & Education, which offers a Unix for Beginners class.

UITS supports SAS software on several timesharing Unix-like environments: AIX by IBM (the Research SP system computers; node aries05), and SunOS (Steel and Nations cluster). Graduate students and staff need a faculty sponsor for accounts on the research-only computers (SP system). Undergraduates are only eligible for accounts on Steel and the Nations cluster. If you want to set up an account on any of the timesharing comp uters, use the appropriate account generation system:

IUB
: Network ID Services

IUPUI
: IUPUI Account Forms

For more information related to the SAS System at IU, please visit our SAS Page.

What is SAS?

SAS is a software system for data analysis and management. In addition to data management facilities and general purpose statistical procedures (Base SAS), and SAS/STAT for statistical analyses. SAS includes the SAS/ETS procedures for econometric and time series analysis, the SAS/GRAPH procedures for color graphics, SAS/IML facilities for matrix manipulation.

The data management capabilities of SAS include:

- Reading data in almost any format.
- Reading, writing, combining multiple files.
- Convenient transformation of data and creation of new variables; elaborate looping and conditional transformation capabilities.
- Storing and using output from statistical procedures in the same run.
- Producing specially-formatted output for reports; printing mailing labels.
- Sorting data; subsetting data; analyzing multiple subsets of cases.
- Storing data and data documentation in SAS libraries.

The statistical capabilities of SAS include the following:

- Univariate descriptive statistics; univariate and multivariate frequency distributions; bar charts, star charts, pie charts, scatter plots, time plots.
- Standardization and ranking of observations; construction of scales.
- Linear probability models, loglinear contingency table models, logistic regression, repeated measurement analysis, probit models.
- Correlations, other measures of association for quantitative variables.
- Multiple regression, regression with linear constraints, stepwise regression; quadratic response-surface

regression models; nonlinear regression; extensive regression diagnostics.
- T-tests, analysis of variance and covariance, analysis of nested designs, multivariate analysis of variance and covariance (including repeated measures); variance components models; ANOVA with ranks.
- Factor analysis with principle components analysis, canonical correlation analysis; cluster analysis. Life tables; fully parametric regression models for survival data.

Example Data

Overview of Sample Data

Suppose a researcher collected the following data during a study to investigate computer anxiety in middle school children. The data were collected from 40 ninth graders in three different school systems. The information collected on each student is: identification number, gender, school system, previous computer experience, scores on a 10-item Likert type computer anxiety scale, scores on a 10-item Likert type mathematics anxiety scale, math scores for a given testing period, and computer test scores for the same testing period. With this information in hand the researcher wanted to write a SAS program to analyze data, both descriptive and inferential.

Let's look into various aspects of creating a SAS program for this data analysis. The first task is to present these data in an orderly form so the SAS software can read and analyze them. There are several variables involved in this research. In SAS Version 8, variables are named with 32 or fewer characters, but must begin with a letter. Let us name these variables according to SAS conventions:

- **ID** student identification number
- **SEX** gender of the student
- **EXP** previous computer experience in months/yrs
- **SCHOOL** name of school system
- **C1 thru C10** 10 scores on the computer anxiety scale
- **M1 thru M10** 10 scores on the math anxiety scale
- **COMPSCOR** computer test score for a given testing period
- **MATHSCOR** math score for the same testing period

Once the variables are named according to SAS conventions, the next task is to prepare a code book with details of the data layout. Following is a code book for the research in discussion.

```
VARIABLE NAME         WIDTH          COLUMNS            VALUE LABELS


  ID                    2              1-2          none
  SEX                   1              1            M=male, F=female
  EXP                   1              4            1=1 yr or less,2=2 yrs, 3=3 yrs
  SCHOOL                1              5            1=rural,2=city, 3=suburban
  C1                    1              6            1=strongly agree, 2=agree,
                                                    3=undecided, 4=disagree, 5=agree
  C2                    1              7              "
  C3                    1              8              "
  C4                    1              9              "
  C5                    1              10             "
  C6                    1              11             "
```

C7	1	12	"
C8	1	13	"
C9	1	14	"
C10	1	15	"
M1	1	16	"
M2	1	17	"
M3	1	18	"
M4	1	19	"
M5	1	20	"
M6	1	21	"
M7	1	22	"
M8	1	23	"
M9	1	24	"
M10	1	25	"
MATHSCOR	2	26-27	
COMPSCOR	2	28-29	

In the above code book VARIABLE NAME stands for the name of the variable in the data, and WIDTH stands for the number of fields taken by each variable. For example, the variable ID takes a maximum of two fields/columns since the highest ID number is 40; EXP takes a maximum of 1 column/field. COLUMNS stands for the column number/s on a given line where a value for each variable can be found by SAS. VALUE LABELS means the value represented within a variable. For example, within the variable SEX, M represents male and F represents female students. Within the variable SCHOOL, 1, 2, 3 represent rural, city, and suburban schools, respectively.

Now let us examine how the data layout will look on a coding sheet or on a computer terminal. These information/variable values are being copied from questionnaires filled in by students. The variables are placed into appropriate columns based on the code book prepared earlier.

```
01M121231122452221135412139 44

02F223254452112334454222125 26

03F112115511411211221551148 45
```

Note that on every line a given variable appears in the same column(s). For example, the variable SEX appears in column 3 of every line. In the above data no blank space is left between variables. You may choose to leave a blank space after each variable as:

```
01 M 1 2 1 2 3 1 1 2 2 4 5 2 2 2 1 1 3 5 4 1 2 1 39 44

02 F 2 2 3 2 5 4 4 5 2 1 1 2 3 3 4 4 5 4 2 2 2 1 25 26

03 F 1 1 2 1 1 5 5 1 1 4 1 1 2 1 1 2 2 1 5 5 1 1 48 45
```

Whichever style (format) you choose, as long as you convey the format correctly to SAS, it should not have any impact on the analysis. In the above layout there are only three lines of data where each line stands for an observation (information about each person). Note that each subject has only one line (record) of data. In another situation you may have more than one record per subject/observation.

Suppose these data are stored in a file in your directory under the name clas.dat. The data can be entered directly to a Unix environment using an editor (e.g., vi, emacs, pico) or can be typed onto a floppy diskette from a microcomputer and then uploaded to the Unix environment using FTP (File Transfer Protocol) or any other appropriate communications package.

Downloading Sample Data

If you are interested in obtaining a copy of this data file you may copy it from the Stat/Math website (http://www.indiana.edu/~statmath).

To obtain a copy of the sample files:

1. Click Sample program file (http://www.indiana.edu/~statmath/stat/sas/CLAS.SAS) and follow the instruction into the pop-up window.
2. Then click Sample data file (http://www.indiana.edu/~statmath/stat/sas/CLAS.DAT).
3. Transfer these files to your Unix account.

Contact a UITS consultant if you need assistance.

Writing a SAS Program: The DATA Step

A SAS program consists of two steps: DATA steps and PROC steps.

In the DATA step you may include commands to create data sets and programming statements to perform data manipulations. The DATA step begins with a DATA statement.

In the PROC (Procedure) step you invoke SAS procedures from the library to run statistical analysis on a given data set. The PROC step begins with a PROC statement.

These steps contain SAS statements. An important feature of the SAS language is that every SAS statement ends with a semicolon (;). Without a semicolon a SAS statement is incomplete.

DATA Statement

DATA *dataname*

The first word, DATA, tells SAS that you want to read a data file and store it in a SAS data set with a name you specify. Replace dataname with an appropriate SAS name (32 or fewer characters), e.g. trial, company, drug, behavior. In the example given below, "dataname" is replaced by the name anxiety. Note the semicolon at the end of the statement.

DATA anxiety;

INPUT Statement

INPUT *var1 column# var2 column# var3 column# ..... varn column#*;

The INPUT statement tells SAS the names of the variables and the column numbers read on a specified line. Variable names in SAS can contain from one to eight characters. They may contain numbers but must begin with a letter. If your data contain more than one line per case (observation), indicate the line number before specifying the variables on that line.

INPUT id 1-3 company 8-10 #2 insal 6-10 finalsal 18-23 #3 retire 15-19;

The above INPUT statement informs SAS that there are three lines of data for each subject/observation. The lines are indicated by a # sign.

INPUT statements need not contain column numbers provided there is a space between each variable value on the data line. This is referred to as free format as opposed to the fixed format where you specify the column numbers. If a variable contains a character value, indicate it by a $ sign after the variable name. If you are choosing the free format, a character variable should not exceed eight characters and should not include embedded blanks. Free format may not be a good idea if you have a large number of variables.

If there are decimal points in your data, you may enter the decimal points as they are or omit them when entering the data and later indicate in the INPUT statement that a given variable has a specified number of decimal points. Suppose you have a variable gpa in your study and the value is to be indicated with three digits of which the last two are decimal places, e.g., 3.89 If you decide to enter the decimal points in your data file, indicate this in your INPUT statement as: INPUT gpa 1-4;. Another choice is to leave out the decimal (389) and later indicate in the INPUT statement that the variable gpa has two decimal points: INPUT GPA 1-3 (.2);. This means that the variable gpa is given in col. 1-3, and the last 2 places are decimal places.

INFILE Statement

INFILE '*path/filename*';

If the data is stored in a separate file (clas.dat in the above example) an INFILE command is used to read the data set into the SAS program, e.g., INFILE '/pathname/clas.dat';. Replace the pathname with the name of the directory in which the data are stored. Data files stored in another directory can also be read through the INFILE command. Data files in another user's directory can also be accessed in the same way, provided proper file protection is set for the source file. SAS can also read several data files from within the same program file.

The INFILE command is usually entered immediately after the DATA line.

DATA anxiety;
INFILE '/usr1/jdoe/clas.dat';

Replace /usr1/jdoe with an appropriate pathname.

Note: Unix is case sensitive. When you are referring to an external file, you must use the correct case. For example, "clas.dat" and "Clas.dat" are two different filenames in Unix and you must match the case correctly whenever referring to another file with single quotes.

CARDS Statement

The CARDS statement tells SAS that data lines are included next. The ends of the data lines are indicated by a semicolon at the beginning of a new line, e.g.,

```
CARDS;
25 32 82 32 1
22 42 .  36 2
;
```

The CARDS statements are usually entered toward the end of the DATA step.

```
DATA anxiety;
        INPUT id 1-3 sex 3 test1 4-5 test2 6-7 test3 8-9;
        IF test1=99 THEN test1=.;
        avscore=(test1+test2+test3)/3;
```

```
CARDS;
0011993240
0022424548
;
```

Missing values in a data set can be represented either by a blank or by a period. If you choose a free format (leaving a space after each variable in the data set and not specifying the column numbers in the INPUT statement) make sure you represent missing values with a period. When SAS encounters a blank or a period in a data set the system regards it as a missing value.

One can assign a missing value to a variable (e.g. 9, 99, 999, 000) and let SAS know which value for a given variable is assigned as missing. Suppose, for a variable mathscor, 99 is assigned as the missing value. Immediately after the INPUT statement you may specify:

IF mathscor=99 THEN mathscor=.;

This statement will assign a missing value whenever it encounters a value of 99 within the variable mathscor.

SAS Functions

In the DATA step you can use a number of SAS functions, e.g., MEAN (computes arithmetic mean), SUM (calculates sum of arguments), VAR (calculates the variance), ABS (returns absolute value), SIN (calculates sine), LOG (produces the natural logarithm), SQRT (calculates the square root).

For instance, to create a new variable final which will be the arithmetic mean (average) of the 3 scores (variables: test1, test2, and test3), you would use the following command:

final=MEAN(test1,test2,test3);

There are a number of SAS operators that could be used in a DATA step, e.g.: ** (raise to a power), * (multiplication), / (division), + (addition), - (subtraction), = or EQ (equal to), >= or GE (greater than or equal to), AND, OR, NOT.

IF ID <= 20 THEN group=1; ELSE group=2;

In the above example, a new variable group is created with 2 categories. Observations with id numbers 20 or lower form group 1 and observations with id numbers greater than 20 form group 2. For details see SAS Language: Reference Version 8, and SAS Language and Procedures: Usage Version 8.

LABEL Statement

You can use LABEL statements either in DATA steps or in PROC steps to give labels to variables.

LABEL variable='variable label';

Replace variable with the name of the variable, and variable label with the label you want to assign to the variable. A SAS variable is limited to eight characters, whereas a label assigned to a SAS variable can have up to 40 characters, including blanks. Labels should be enclosed in quotes, and the LABEL step is terminated by a semicolon, e.g.,

LABEL exp='years of computer experience';
LABEL mathscor='score in mathematics';

FORMAT Statement

The FORMAT statement associates formats with variables in a DATA step. In the above example the variable sex has two values (1,2) and school has 3 values (1,2,3). To associate these values with appropriate value labels, use the format statement. The FORMAT statement may be used in a DATA step or in a PROC step. However, when you define format with PROC FORMAT it appears as the very first line in a SAS program.

```
PROC FORMAT;
  value $sex 'M'='male' 'F'='female';
  value school 1='rural' 2='city' 3='suburban';
```

Once you define the format as above, you should associate the format with the variable/s through a format statement after the INPUT line.

```
PROC FORMAT;
  value $sex 'M'='male' 'F'='female';
  value school 1='rural' 2='city' 3='suburban';
DATA anxiety;
  INPUT id 1-3 sex $ 4 school 5 test 6-7;
  FORMAT sex $sex. school school.;
```

Writing a SAS program: the PROC Step

The next step is to create PROC (procedure) steps. SAS procedures read the SAS data and perform various computations and print the results of these computations. The FREQ procedure computes the frequencies on specified variables; the TTEST procedure performs a t-test analysis on the specified variables. In short, the statements that ask SAS to process or analyze a specified data set are known as PROC steps. The DATA steps and PROC steps can be used in any order within a SAS program. As the DATA step starts with a DATA statement the PROC step starts with a PROC statement.

```
PROC PRINT DATA=dataname;
  VAR var1 var2;
```

This procedure requests SAS to print data values for variables 1 and 2. If the VAR statement is omitted data values for each variable in the data set will be printed. DATA=dataname is an optional statement. If this is omitted SAS selects the most recently created data set within the program. It is a good practice to specify the dataname along with the PROC statement. Replace the dataname with the name of the data created in the DATA step. Printing out a few variables before doing any analysis is a good way to check whether the data are being read by SAS as you want them to be read.

FREQ Statement

```
PROC FREQ;
  TABLES var1 var2 var1*var2;
```

This statement produces tables showing distribution of variable values. In the above example SAS will display variable values for var1 and var2, and the combined frequency distributions for var1 and var2. For example, if you wanted to get the gender breakdown for the test scores discussed previously, you would use the following command:

```
PROC FREQ;
  TABLES sex;
```

The output generated by this command would look like this:

```
                                Cumulative  Cumulative
            SEX   Frequency   Percent   Frequency    Percent
            ------------------------------------------------
```

```
f            5      50.0           5       50.0
m            5      50.0          10      100.0
```

MEANS Statement

PROC MEANS;
  VAR *var1 var2*;

This statement computes descriptive statistics for specified variables. If the VAR statement is omitted, descriptive statistics for each variable in the data set will be calculated.

Again, referring back to the grades data, if you wanted to generate some basic descriptive statistics for the students' test scores, the commands and output would look like this:

PROC MEANS;
  VAR test1 test2 test3;

```
              The SAS System     10:00 Thursday, November 2, 1995


     Variable   N          Mean       Std Dev       Minimum       Maximum
     ---------------------------------------------------------------------
     TEST1     10    80.1000000    10.4504120    60.0000000    90.0000000
     TEST2     10    82.6000000     8.4616783    72.0000000    96.0000000
     TEST3     10    81.2000000    10.6854002    64.0000000    93.0000000
     ---------------------------------------------------------------------
```

CORR Statement

PROC CORR;
  VAR *var1 var2*;

A correlation analysis is performed to quantify the strength of association between two numeric variables. For example, if you wanted to see if there were a correlation between student test scores, the commands and output would look like this:

PROC CORR;
  VAR test1 test2 test3;

```
                   Correlation Analysis


  Pearson Correlation Coefficients / Prob > |R| under Ho: Rho=0  / N = 10

                    TEST1             TEST2             TEST3

   TEST1           1.00000           0.75692           0.91522
                    0.0               0.0113            0.0002

   TEST2           0.75692           1.00000           0.86857
```

```
                0.0113          0.0             0.0011

TEST3           0.91522         0.86857         1.00000
                0.0002          0.0011          0.0
```

ENDSAS statement

By this statement you indicate that there are no more data steps or procedure steps to be read or processed. ENDSAS should be followed by a semicolon (e.g., ENDSAS;). ENDSAS exist the SAS system and returns you to the UNIX environment.

Writing and Executing A SAS Program

Writing a SAS Program

Now that we have looked at various steps that go into creating a SAS program, the next step is to write one. The example used here is for the data set discussed earlier. A SAS statement may begin in any column on a line. However, for the sake of clarity, lines starting with DATA or PROC statements begin in column 1; all other statements are indented. In the following example SAS is asked to create a data set named anxiety after reading the data steps. The data are embedded within the SAS program.

A SAS program in its simple form can be as follows:

```
DATA anxiety;
        INPUT id 1-2 sex $ 3 exp 4 school 5
                c1 6 c2 7 c3 8 c4 9 c5 10 c6 11 c7 12 c8 13 c9 14 c10 15
                m1 16 m2 17 m3 18 m4 19 m5 20 m6 21 m7 22 m8 23 m9 24 m10 25
                mathscor 26-27 compscor 28-29;
CARDS;


[data lines entered here]


;


PROC PRINT;
        VAR sex school exp mathscor compscor;
PROC FREQ DATA=anxiety;
        TABLES sex exp school;
ENDSAS;
```

The input format specified in the above example can also be written in a shorter form with a mixed style column and formatted input.

INPUT id 1-2 sex $ 3 (exp school) (1.) (c1-c10) (1.) (m1-m10) (1.) (mathscor compscor) (2.);

In this case the program will read variable id from columns 1-2 and sex from column 3. The next two variables exp and school have a width of 1 column each and start in column 4. The variables c1 through c10 (10 variables in sequential order) have a width of 1 column each and start immediately after the variable school.

If you wanted to read only the variables id and the last 2 variables (mathscor, compscor) you could write the INPUT statement as:

INPUT ID 1-2 @26 (mathscor compscor) (2.);

The @ moves the pointer to column 26 and reads two variables with a width of 2 columns each.

The example below illustrates the above data set being read by SAS from an external file named clas.dat stored in the same directory. Comments are provided for documentation purposes. Statements enclosed in /* .... */ or *.....; are ignored by SAS and will not be used while executing the program. The TITLE statement is used to provide title lines to be printed on the output file.

```
* Computer Anxiety in Middle School Children;
/* The following procedure specifies value labels for variables */

PROC FORMAT;
        VALUE $sex 'M'='male' 'F'='female';
        VALUE exp 1='upto 1 year' 2='2-3 yrs' 3='3+ yrs';
        VALUE school 1='rural' 2='city' 3='suburban';
DATA anxiety;
        INFILE '/pathname/clas.dat';
        /* remember that Unix is case sensitive, so anything within single quotes must have t
        INPUT ID 1-2 SEX $ 3 (exp school) (1.) (c1-c10) (1.) (m1-m10) (1.)
                (mathscor compscor) (2.);

        FORMAT sex $sex. exp exp. school school.;

/* conditional transformation. Missing values are declared .*/

        IF mathscor=99 THEN  mathscor=.;
        IF compscor=99 THEN  compscor=.;

/* recoding variables. Several items are to be reversed in scoring. */
/* the Likert type questionnaire had a choice range of 1-5 */

        C3=6-c3; c5=6-c5; c6=6-c6; c10=6-c10;
        m3=6-m3; m7=6-m7; m8=6-m8; m9=6-m9;
        compopi = sum (of c1-c10) /*Find sum of 10 items using SUM function */;
        mathatti = m1+m2+m3+m4+m5+m6+m7+m8+m9+m10 /* Add item by item */;
```

```
        LABEL id='STUDENT IDENTIFICATION' sex='STUDENT GENDER'
                exp='YRS OF COMP EXPERIENCE' school='SCHOOL REPRESENTING'
                mathscor='SCORE IN MATHEMATICS'
                compscor='SCORE IN COMPUTER SCIENCE'
                compopi='TOTAL FOR COMP SURVEY'
                mathatti='TOTAL FOR MATH ATTI SCALE';

PROC PRINT;
   VAR exp school mathscor compscor compopi mathatti;
TITLE 'LISTING OF THE VARIABLES';

PROC FREQ DATA=anxiety;
   TABLES sex exp school;
   TABLES (exp school)*sex;
TITLE 'FREQUENCY COUNT';
PROC MEANS DATA=anxiety;
   VAR compopi mathatti mathscor compscor;
TITLE 'Descriptive Statistics';

* The following procedure creates a data subset with males alone;

DATA males;
        SET anxiety;
   IF sex EQ 'M';

* sorting the data by school to use the CLASS statement in printing;
* out the means for each school system in a following step;

PROC SORT DATA=males;
        BY school;

PROC MEANS DATA=males;
        CLASS school;
        VAR compopi mathatti mathscor compscor;
ENDSAS;
```

Modes of Execution

Suppose that you saved the above program in to a file, clas.sas, in your root directory. Selecting a mode of

execution to run your SAS job depends on the type of job you run, and the system you work with. Because Unix operating systems are multitasking systems, you can run various processes at the same time. For example, you can have one process running in the foreground and several in the background. A foreground process executes while you wait. The terminal cannot be used while the job is running. A background process executes independently of the shell, and the terminal is free for other computing activities. SAS interactive line mode, display manager mode, and non-interactive mode all run in the foreground. A batch mode runs in the background. Because Unix is case sensitive, use lowercase when you invoke SAS. Within SAS you may use upper or lower case.

A SAS job creates temporary scratch files during every mode of execution. These scratch files will be deleted once the execution is complete. In some instances your disk may not have enough space to accommodate the scratch file. In such a situation you can redirect your scratch file to a temporary system disk (/tmp and /scr). This is done while invoking SAS:

sas -work /tmp clas1 &

The -work parameter reroutes the scratch file to a temporary system disk (/tmp). You may replace /tmp with /scr. You may also copy your large files into these directories for temporary use.

Interactive mode

If you do not have a full-screen terminal (e.g., vt100) and want to run SAS in interactive mode, you may use interactive line mode. If you do have a full-screen terminal, you have the option of running your SAS programs in either interactive line mode or display manager mode. To start an interactive line mode, at the system prompt, type:

sas -nodms

The option "nodms" stands for No Display Manager.

Now the system invokes the SAS system and the SAS prompt with a line number and question mark, 1?, appears on your terminal. Type in the SAS statements and press ENTER. A statement is not complete until it is succeeded by a semicolon. SAS responds to statements as you enter them.

Following is an example of a SAS line-mode session saved from the UITS IBM RS/6000 computer called SP05. You may want to try it to have a feel for the SAS line mode. Enter the statements as given and press RETURN at the end of each line. If you notice a mistake in typing before pressing RETURN, just move the cursor backward and retype it. If SAS responds to you with an error, retype the whole line correctly.

$ sas -nodms

```
NOTE: Copyright (c) 1999-2001 by SAS Institute Inc., Cary, NC, USA.
NOTE: SAS (r) Proprietary Software Release 8.2 (TS2M0)
      Licensed to INDIANA UNIVERSITY, Site 0009093004.
NOTE: This session is executing on the AIX 5.1 platform.


      The help with SAS computing, contact the UITS Stat/Math
      Center (855-4724; email statmath@indiana.edu; URL
      http://www.indiana.edu/~statmath).

NOTE: SAS initialization used:
      real time           0.21 seconds
      cpu time            0.05 seconds

  1? data gender;
  2? input id sex $ group test1 test2;
  3? totscore=test1+test2;
  4? cards;
  5> 01 m 1 42 41
  6> 02 m 1 41 37
  7> 03 f 1 43 36
  8> 04 f 1 34 32
  9> 05 f 2 32 34
```

```
 10> 06 f 2 30 35
 11> 07 m 2 30 28
 12> 08 m 2 26 30
 13> ;

NOTE: The data set WORK.GENDER has 8 observations and 6 variables.
NOTE: DATA statement used:
      real time           2:30.17
      cpu time            0.03 seconds

 14? proc print;
 15? var sex group totscore;
 16? title 'Printing Variables';
 17? run;


                            Printing Variables                              1
                                          14:12 Tuesday, August 15, 2000

                        Obs    sex    group    totscore

                         1      m       1         83
                         2      m       1         78
                         3      f       1         79
                         4      f       1         66
                         5      f       2         66
                         6      f       2         67
                         7      m       2         58
                         8      m       2         56
NOTE: There were 8 observations read from the dataset WORK.GENDER.
NOTE: PROCEDURE PRINT used:
      real time           48.46 seconds
      cpu time            0.10 seconds


  18? endsas;

NOTE: SAS Institute Inc., SAS Campus Drive, Cary, NC USA 27513-2414
NOTE: The SAS System used:
      real time           10:21.89
      cpu time            1.06 seconds
```

You can enter any number of statements during line mode session. If you want to save the command lines you entered during an interactive session, enter the following lines before you end the session.

```
18? options spool;
19? filename mylog 'myprog.prg';
20? proc printto log=mylog;
21? run;
22? %list;
23? endsas;
```

All SAS statements entered during the session will be stored in the default directory under the myprog.prg or a name you specify. This file will contain only the SAS statements. This file could be edited and later read into a SAS interactive session us ing the %INCLUDE command. You may also use the %INCLUDE command to read a SAS program file into a line-mode session. For example, to read a file, clas.sas, into the line-mode session, at the line mode prompt (e.g. 1?, 2?, 5?), type:

%INCLUDE clas;

Now the entire file will be read into the session. If the filename has extension other than ".sas," you must enclose the full filename with extension in single quotes (e.g., clas.prg) in the INCLUDE statement. If there is an ENDSAS line at the end of your included file, SAS will, after execution, exit the job and take you to the system prompt.

If you want to save the .log and .lst files from the session to separate files in your directory, use the altlog and altprint options while invoking SAS in interactive mode.

sas -nodms -altlog myfile.log -altprint myfile.lst

Replace the log and the listing file with appropriate filenames. Your log and listing files will be saved in the default directory under the filenames specified. Refer to the SAS Companion for the UNIX Environment for details.

Non-interactive mode

Once a SAS program file is created and saved you can run it non-interactively. Suppose you want to execute the command file, clas.sas, non-interactively. At the system prompt, type:

sas clas

(Make sure that the data file, clas.dat, is accessible during the job.)

Note that the extension .sas is omitted since SAS automatically picks the file clas.sas for execution. Use the full name of the file if it does not have a .sas extension (e.g., sas clas.exp, sas clas.txt.). The cursor will move to the first column of the next line and rest there until the job is completed. You cannot use the terminal for any other job until the program is executed. When the job is completed the system prompt reappears. At this point, if your job was successfully executed, two new files (log and listing) will appear in your directory. The log file will have an extension of .log (e.g. clas.log). This file contains your program and any error message accompanying it. A second file with an extension of .lst (e.g. clas.lst) will be stored in your directory. This file contains the output of the procedures executed. If there are errors in the program file a listing file may or may not be created depending on the nature of the errors. It is always a good idea to check your .log file by displaying it on the terminal with the more or cat command or by printing it. If there are errors, edit the SAS program file and execute the program again.

One disadvantage of a non-interactive job is that it will tie up your terminal until the job is completed. However, if you want the terminal to be freed for other work while the job is running in the background, type:

sas clas &

You will see a process identification number (PID) for the background task and your terminal will be free for other computing while the job is running. Check the status of the job with the ps command from the system prompt. Once the job is executed, the log and listing file will be stored in the default directory. To kill a background process, use the kill command.

Batch Execution on the Research SP

Batch queues are available on all of the Research SP nodes (running IBM AIX). CPU-intensive jobs that require more CPU time to process must be submitted to the batch queue for execution.

You are required to use IBM's LoadLeveler batch queuing system for CPU-intensive jobs that require more than 20 minutes of CPU time. A type of queue is available for statistical jobs on the SP machines:

- stat - Jobs requiring up to 8-day of CPU time

To submit a SAS job to LoadLeveler, create a script file, e.g., sasjob1, with the following lines:

```
#@ requirements = (Feature == "sas")
#@ initialdir = directory
#@ error = filename
#@ class = stat
#@ queue
sas inputfile
```

Replace "directory" with the directory where the command file is stored, although if you submit the job from

the same directory as the command file, you don't need to specify the directory. Replace "filename" with the name of a file to which any script errors will be written. Finally, replace "inputfile" with the name of the SAS command file (e.g., clas.sas). The error/log file will be stored in the same directory in a file named sasjob1.err.

To submit the job, at the system prompt, type:

llsubmit sasjob1

The output files will be stored in the directory specified in the script file. You may log out after submitting the job.

For more information on batch jobs (e.g., checking status queue, canceling jobs) on the SP system, refer to Getting Started on AIX: A User's Guide to Research SP, available at the UITS Support Center or online (http://sp-www.iu.edu/getting.started.shtml). Another useful document, Submitting Batch Jobs to LoadLeveler, is available at (http://sp-www.iu.edu/LLguide.shtml).

SAS Data Sets

Creating a SAS Data Set

If you are handling a large data set, it is advisable to create a SAS data set and work with that. Under Unix systems, a SAS file is a specially structured Unix file. A SAS file is readable only by SAS from the operating system under which it was creat ed. A stored SAS file cannot be edited using an editor like emacs, vi, etc. SAS data sets are referenced with a one- or two-level name. The two-level name is of the form libref.member-name, where "libref" refers to the directory in which the data set is t o be stored or read, and "member-name" refers to the name of the SAS data file to be created or read. The one-level name is of the form member-name (the libref is omitted). In this case, the SAS system stores the files in the temporary WORK data library.

Suppose we want to create a SAS data set with the program file (clas1.sas) we created earlier. We want to create a SAS data set with the entire DATA steps without any PROC steps. Note the first two lines in the following program where the LIBNAME comma nd is issued to reference the directory in which the SAS file is to be created, and the two-level name links the member-name (e.g., anxiety) to the libref (e.g., try1).

```
LIBNAME try1'pathname';
DATA try1.anxiety;
        INFILE clas.dat;
        INPUT id 1-2 sex $ 3  exp 4 school 5
   (c1-c10) (1.) (m1-m10) (1.) (mathscor compscor) (2.);

        IF mathscor=99 THEN  mathscor=.;
        IF compscor=99 THEN  compscor=.;

        c3=6-c3; c5=6-c5; c6=6-c6; c10=6-c10;
        m3=6-m3; m7=6-m7; m8=6-m8; m9=6-m9;

        compopi = SUM (OF c1-c10);
        mathatti = m1+m2+m3+m4+m5+m6+m7+m8+m9+m10;

        LABEL id='STUDENT IDENTIFICATION' sex='STUDENT GENDER'
                exp='YRS OF COMP EXPERIENCE' school='SCHOOL REPRESENTING'
                mathscor='SCORE IN MATHEMATICS'
                compscor='SCORE IN COMPUTER SCIENCE'
                compopi='TOTAL FOR COMP SURVEY'
                mathatti='TOTAL FOR MATH ATTI SCALE';
RUN;
ENDSAS;
```

Replace "pathname" with the appropriate directory. When the job is executed (any execution mode), a SAS data set named anxiety.sas7bdat will be stored in your directory. Under certain Unix platforms the filename extension (e.g., sas7bdat) may differ.

Accessing a SAS Data Set

To read an existing SAS data set, use a two-level name of the form libref.member.name.

The following example illustrates how to access the SAS data set (e.g., anxiety.sas7bdat) created and run some SAS procedures with it. (Note that try2 is given as libref and anxiety is given as member-name.)

```
LIBNAME try2 '/usr1/jdoe';
DATA anxiety2;
  SET try2.anxiety;

PROC TTEST;
  CLASS sex;
  VAR compopi;
TITLE 'T-TEST';

PROC CORR DATA=anxiety2;
  VAR compscor mathscor compopi mathatti;

PROC REG DATA=anxiety2;
  MODEL compopi=mathscor mathopi compscor;

ENDSAS;
```

The above job invokes the SAS data set, anxiety.sas7bdat, from the directory specified and runs requested SAS procedures. Refer to SAS Language, and SAS Companion for the UNIX Environment for further information on SAS data files.

Reading Compressed Files

It is possible with SAS to compress your ASCII-test data files so that they take up less space and then have SAS automatically uncompress the files when you execute a command file. Note that SAS cannot read compressed SAS data sets or compressed SAS transport files. If your data are normal ASCII-text files, you can compress your files so that they take up considerably less space (a 50% reduction or more is possible).

To compress a data file, use the Unix compress command which employs then Lempel-Ziv compression method. At the Unix prompt, type:

compress filename

Replace "filename" with the name of the data file you wish to compress. This creates a new file with the extension ".Z". For example, if you compress a file called "test.dat," a compressed file called "test.dat.Z" would be created, replacing the original file (test.dat).

To read this file into SAS without having to uncompress it beforehand, you should add the following SAS command to the command file:

FILENAME alias PIPE 'zcat filename';

Replace "alias" with the file handle (a nickname for the data file) you wish to use which can be up to eight

characters long. Replace "filename" with the name of the compressed file including the ".Z" extension. You must also include the path if the file is somewhere other than the default directory (i.e. the directory from which you launch SAS).

For example, if you had a data file called "test.dat" with 10 variables you wished to compress and then use in SAS, first compress the file at the Unix prompt:

compress test.dat

Next, create a SAS command file with the following lines:

```
DATA test1;

FILENAME test PIPE 'zcat test.dat.Z';

INFILE test;

INPUT v1-v10;

RUN;
```

Finally, execute your SAS command file normally. SAS generates a data set called test1.

SAS Transport Libraries

SAS also handles transport format data sets. A transport format file is created when you want to move your SAS data set to another operating system. Also, if you are bringing SAS data sets from another operating system into the Unix platform, you have to use a transport file. Note that the various Unix platforms are distinct in the way they create SAS data sets. That means a SAS data set created under one Unix system may not be readable under another. For example, a SAS data set created under sUNos (e.g. Steel) is not readable by SAS under IBM AIX (Research SP node aries05). You want to create a transport file for this purpose.

Suppose you want to create a SAS transport format file from the SAS data file, anxiety.sas7bdat. Define a LIBNAME to read the SAS data set, and another libname to write a SAS transport file. The XPORT (for transport engine) parameter is used to indicate that we want to create a transport format file. A transport format file always has fixed block size with a record length of 80, and a block size of 8000. The select statement may be omitted if there is only one SAS file stored in the directory or if you want to convert all the members of a single SAS data library in to SAS transport format library.

```
LIBNAME test1 '/usr1/jdoe';

LIBNAME test2 XPORT '/usr1/jdoe/trans.xpt';

PROC COPY IN=test1 OUT=test2;

        SELECT anxiety;

RUN;
```

Once the job is executed, a file called trans.xpt will be created and stored in the directory specified.

To read a transport format file (e.g., trans.xpt) stored on the disk and create a SAS data file, as in the example given above, define two libnames (one for reading, and one for writing) as in:

```
LIBNAME test1 '/usr1/jdoe';

LIBNAME test2 XPORT '/usr1/jdoe/trans.xpt';

PROC COPY IN=test2 OUT=test1;

RUN;
```

When there is more than one file stored in a transport library a select statement may be used to access the file of your choice. If you want to read/write SAS transport files involving format library, use the CIMPORT/CPORT procedures instead of the COPY procedure. See SAS Language Reference V. 8, and SAS Procedures Guide V. 8 for information.

Further Reading

Vendor-supplied documents are available for reference at Swain Hall and Business/SPEA Library Reserve Collections, IUPUI University Library and at the UITS Stat/Math Center (410 N. Park Ave.). Faculty, staff, and students at Indiana University can access the complete SAS 8 documentation online from: http://www.indiana.edu/~statmath/stat/sas/sashtml/onldoc.htm

Documents may be ordered through the IU Bookstore or bought directly from the SAS Publishing (phone: 1-800-727-3228) (http://www.sas.com/pubs).
