FUNDAMENTALS OF

SAS PROGRAMS

CREDIT:

HTTP://WWW.PSYCH.YORKU.CA/LAB/SAS/SASPROG.HTM

SAS programs

This document provides an overview of SAS procedures and SAS programming statements.

Contents

SAS program steps

All SAS programs consist of a sequence of "steps". There are only two kinds of steps:

DATA step

A DATA step creates a SAS dataset (a collection of data together with a "data dictionary", which defines the variables and their properties). Data must be in the form of a SAS dataset before it can be analyzed by SAS procedures.

In the example SAS program, these lines create the dataset CLASS from raw data input:

```
DATA CLASS;
   INPUT NAME $ SEX $ AGE HEIGHT WEIGHT;
   CARDS;
JOHN       M 12 59.0   99.5
JAMES      M 12 57.3   83.0
 ... (more data lines)
```

or "datalines;"

PROC step

A PROCedure step calls a SAS procedure to analyse or process a SAS dataset.

In the example SAS program, these lines call two SAS procedures to analyze the CLASS dataset:

```
PROC PRINT;
PROC MEANS;
   VARIABLES HEIGHT WEIGHT;
```

A SAS program can contain any number of DATA and PROC steps. The SAS statements in each step are executed all together. Once a dataset has been created, it can be processed by any subsequent DATA or PROC step.

SAS statements

- All SAS statements start with a ***keyword*** (DATA, INPUT, PROC, etc.)
- ***All SAS statements end with a semicolon (;)*** . (The most common problem students encounter is omitting a semicolon -- SAS thinks that two statements are just one.)
- SAS statements can be entered in ***free-format*** : You can begin in any column, type several statements on one line or split a single statement over several lines (as long as no word is split.).
- ***Uppercase and lowercase are equivalent, except inside quote marks*** (`sex = 'm';` is not the same as `sex = 'M';`).

SAS Procedures

SAS Procedures exist to carry out all the forms of statistical analysis. As the above examples indicate, a procedure is invoked in a "PROC step" which starts with the keyword PROC, such as:

```
PROC MEANS DATA=CLASS;
   VAR HEIGHT WEIGHT;
```

The VAR or VARIABLES statement can be used with all procedures to indicate which variables are to be analyzed. If this statement is omitted, the default is to include ***all variables*** of the appropriate type (character or numeric) for the given analysis.

Some other statements that can be used with most SAS procedure steps are:

BY variable(s);
: Causes the procedure to be repeated automatically for ***each*** different value of the named variable (s). The data set must first be sorted by those variables.

ID variable(s);
: Give the name of a variable to be used as an observation **ID**entifier.

LABEL var='label';
: Assign a descriptive label to a variable.

WHERE (expression);
: Select only those observations for which the expression is true.

For example, the following lines produce separate means for males and females, with the variable SEX labelled 'Gender'. (An ID statement is not appropriate, because PROC MEANS produces only summary output.)

```
PROC SORT DATA=CLASS;
   BY SEX;
PROC MEANS DATA=CLASS;
   VAR HEIGHT WEIGHT;
   BY SEX;
   LABEL SEX='Gender';
```

If the DATA= option is not used, SAS procedures process the most recently created dataset. In the brief summaries below, the required portions of a PROC step are shown in **bold**. Only a few representative options are shown.

Descriptive statistics

PROC CORR

Correlations among a set of variables.

```
PROC CORR DATA=SASdataset  options;
          options:NOMISS ALPHA
   VAR variable(s);
   WITH variable(s);
```

PROC FREQ

Frequency tables, chi^2 tests

```
PROC FREQ DATA=SASdataset;
   TABLES variable(s) / options;
          options:NOCOL NOROW NOPERCENT
   OUTPUT OUT=SASdataset;
```

PROC MEANS

Means, standard deviations, and a host of other univariate statistics for a set of variables.

```
PROC MEANS DATA=SASdataset  options;
          options:N  MEAN  STD  MIN  MAX  SUM  VAR  CSS USS
   VAR variable(s);
   BY variable(s);
   OUTPUT OUT=SASdataset  keyword=variablename ... ;
```

Statistical options on the PROC MEANS statement determine which statistics are printed. The (optional) OUTPUT statement is used to create a SAS dataset containing the values of these statistics.

PROC UNIVARIATE

Univariate statistics and displays for a set of variables.

```
PROC UNIVARIATE DATA=SASdataset  options;
          options:PLOT
   VAR variable(s);
   BY variable(s);
   OUTPUT OUT=SASdataset  keyword=variablename ... ;
```

Linear models

SAS statements and options for regression (PROC REG) are described in more detail in the document PROC REG Summary. SAS statements and options for analysis of variance (PROC ANOVA and PROC GLM) described in the document PROC ANOVA and PROC GLM.

PROC ANOVA

Analysis of variance (balanced designs)

```
PROC ANOVA DATA=SASdataset  options;
   CLASS variable(s);
   MODEL dependent(s)= effect(s);
```

PROC GLM

General linear models, including ANOVA, regression and analysis of covariance models.

```
PROC GLM DATA=SASdataset  options;
   CLASS variable(s);
   MODEL dependent(s)= effect(s);
   OUTPUT OUT=SASdataset  keyword=variablename ... ;
```

PROC REG

Regression analysis

```
PROC REG DATA=SASdataset  options;
   MODEL dependent(s) = regressors
 / options;
   PLOT variable | keyword. *
        variable | keyword. = symbol ;
   OUTPUT OUT=SASdataset  P=name R=name ... ;
```

Plots and charts

PROC CHART

Histograms and bar charts

```
PROC CHART DATA=SASdataset  options;
   VBAR variable / options;
   HBAR variable / options;
           options: MIDPOINTS=  GROUP=  SUMVAR=
```

PROC PLOT

Scatter plots

```
PROC PLOT DATA=SASdataset  options;
           options: HPERCENT=  VPERCENT=
   PLOT yvariable *
        xvariable =  symbol / options;
   PLOT (yvariables) *
        (xvariables) =  symbol / options ;
           PLOT options: BOX  OVERLAY  VREF=  HREF=
   BY  variable(s) ;
```

Note that the parenthesized form in the PLOT statement plots **each** y-variable listed against each x-variable.

Utility procedures

PROC PRINT

Print a SAS data set

```
PROC PRINT DATA= SASdataset  options;
           options: UNIFORM  LABEL  SPLIT='char'
   VAR variable(s);
```

```
BY variable(s);
SUM variable(s);
```

PROC SORT
Sort a SAS data set according to one or more variables.

```
PROC SORT DATA=SASdataset  options;
          options: OUT=
  BY variable(s);
```

SAS datasets and the DATA step

SAS names

Variable names and SAS dataset names are:

- 1 - 8 characters in length
- begin with A-Z or _ (underscore)
- cannot contain blanks or special symbols (e.g., &, %, $, #, etc.)

SAS variables:

- Character variables (e.g., `NAME='Michael';`)
- Numeric variables
- Missing data: represented by `'.'` for numeric variables; by `' '` (a space) for character variables.

The DATA step

The SAS DATA step is used to create or process SAS datasets. A DATA step can read raw data (INPUT statement), or data from an existing SAS dataset (SET statement). The key feature of the DATA step is this: SAS carries out all statements in the DATA step in order ***for each input observation*** .

Some of the (many) statements that can be used in the DATA step are:

DATA
The DATA statememt signals the start of a DATA step and names the dataset(s) to be created.

```
DATA SASdataset(s);
```

INPUT
The INPUT statement specifies how raw data is to be read. ***List input*** reads data in free format. Simply list the names of your variables, in the order they appear on the data lines. A $ sign following the name of any variable indicates that variable is to be read as characters.

```
INPUT NAME $ SEX $ AGE HEIGHT WEIGHT;
```

Column input reads data in specified columns. Use column input when your data is not separated by blanks, to read character fields longer than 8 characters, or when you do not want to read all

the information on each data line.

```
INPUT NAME $1-8 SEX $11 AGE 13-14 HEIGHT 16-19 WEIGHT 22-25;
```

SET

The SET statement reads observations from an existing SAS dataset. These statements simply make a copy of the CLASS dataset.

```
data newclass;
   set class;
```

Assignment

The assignment statement creates new variables or changes existing variables. All the usual arithmetic operations, and many SAS functions can be used.

Symbol	Operation	Example
**	Exponentiation	Y = X **2;
*	Multiplication	AREA = LEN * WIDTH ;
/	Division	DENSITY = MASS / VOLUME;
+	Addition	PRICE = COST + MARKUP;
-	Subtraction	COST = PRICE - MARKUP;

Use parentheses to indicate grouping in complex expressions:

```
AVG = (TEST1 + 2*TEST2 + 5*FINAL) / 8 + BONUS;
```

IF

The IF statement is used for conditional processing.

```
IF expression
   THEN statement;
   ELSE statement;
```

The ELSE statement is optional. The IF ... THEN parts comprise a single statment. For example,

```
If age < 13 then group = 'preteen';
            else group = 'teen';
If sex = 'F' then SX = 1;      /* Dummy variable for sex */
             else SX = 0;
SX = (sex='F');                /* same as above (if no missing) */
```

SAS comparison operators are shown below. You can use either the symbol or the two-letter abbreviation.

Symbol	Abbrev	
<, <=	LT, LE	less than, less than or equal
>, >=	GT, GE	greater than, greater than or equal
=, ^=	EQ, NE	equal, not equal

A special form of the IF statement is used for **subsetting a dataset**. To extract the males from the CLASS dataset:

```
DATA MALES;
   SET CLASS;
   IF SEX='M';
```

The statement `IF SEX='M';` is equivalent to each of the statements:

```
   IF SEX='M' THEN OUTPUT;
   IF SEX^='M' THEN DELETE;
```

Comments

Two types of comments: the comment statement (`* ... ;`) and comment stuff (`/* ... */`) The comment statement (like all SAS statements) must end with a semi-colon. Comment stuff can appear anywhere a single blank can appear. The comments are shown **bold** in the example below. Note that an entire statement is treated as a comment.

```
data class;
  * Read in the variables;
  input name $ sex $ age height weight;
  /*  ignore next statement
  age = age + 3;
  */
```

SAS functions

SAS contains several hundred functions which can be used in the DATA step. Here are some of the more commonly used ones.

ABS(x)
: Absolute value, $|x|$.

EXP(x)
: Exponential, e^x ; EXP(1) = 2.71828183....

INT(x)
: Truncate x to an integer; INT(3.145) = 3.

LOG(x)
: Natural logarithm, $log_e(x)$; LOG(10) = 2.30258509....

LOG10(x)
: Common logarithm, $log_{10}(x)$; LOG10(10) = 1.

MOD(x,d)
: Remainder when x is divided by d; MOD(10,3) = 1.

ROUND(num)
ROUND(num, unit)
: Round a number to the nearest integer (or nearest specified unit); ROUND(3.678) = 4; ROUND (3.678,.1) = 3.7.

SQRT(x)
: Calculate the square root of x.

NORMAL(seed)
: Return a normally distributed random number

UNIFORM(seed)
: Return a uniform [0,1] random number.

Another collection of SAS functions calculate various statistics for a single observation across a set of

variables (rather than across observations, as in PROC MEANS). For each of these, the argument can be a list of variable names, separated by commas, or the keyword OF followed by a SAS variable list. These functions all ignore missing data: the result is computed from the non-missing values.

MEAN(v1,v2,...)
MEAN(OF age ht)
MEAN(OF V1-V6)
Mean of the non-missing values of the variables
MAX Maximum
MIN Minimum
STD Standard deviation
VAR Variance
USS Uncorrected sum of squares, Σv_i^2.
CSS Corrected sum of squares, $\Sigma (v_i - v\ bar)^2$.

Example

The example below reads the CLASS data set variables, and creates several additional variables with DATA step programming statements.

```
DATA CLASS;
   INPUT NAME $ SEX $ AGE HEIGHT WEIGHT;
   If age < 13 then group = 'preteen';
               else group = 'teen';
   logwt = log10(weight);          /* transform variables */
   rootht= sqrt(height);
   CARDS;
JOHN       M 12 59.0   99.5
JAMES      M 12 57.3   83.0
 ... (more data lines)
```

data lines;

SOME EXAMPLES

OF

READING IN DATA SETS

IN SAS

Previous Page | Next Page

Additional Data Sets

Data Sets for the Storing and Managing Data in SAS Files Section

DATA Step to Create the Data Set USCLIM.HIGHTEMP

```
libname usclim 'SAS-data-library';

data usclim.hightemp;
   input State $char14. City $char14. Temp_f Date $ Elevation;
   datalines;
Arizona        Parker          127 07jul05 345
Kansas         Alton           121 25jul36 1651
Nevada         Overton         122 23jun54 1240
North Dakota   Steele          121 06jul36 1857
Oklahoma       Tishomingo      120 26jul43 6709
Texas          Seymour         120 12aug36 1291
;
```

DATA Step to Create the Data Set USCLIM.HURRICANE

```
libname usclim 'SAS-data-library';

data usclim.hurricane;
   input @1 State $char11. @13 Date date7. Deaths Millions Name $;
   format Date worddate18. Millions dollar6.;
   informat State $char11. Date date9.;
   label Millions='Damage';
   datalines;
Mississippi 14aug69 256 1420 Camille
Florida     14jun72 117 2100 Agnes
Alabama     29aug79 5   2300 Frederick
Texas       15aug83 21  2000 Alicia
Texas       03aug80 28  300  Allen
;
```

DATA Step to Create the Data Set USCLIM.LOWTEMP

```
libname usclim 'SAS-data-library';

data usclim.lowtemp;
   input State $char14. City $char14. Temp_f Date $ Elevation;
   datalines;
Alaska         Prospect Creek -80 23jan71 1100
Colorado       Maybell        -60 01jan79 5920
Idaho          Island Prk Dam -60 18jan43 6285
Minnesota      Pokegama Dam   -59 16feb03 1280
North Dakota   Parshall       -60 15feb36 1929
South Dakota   McIntosh       -58 17feb36 2277
Wyoming        Moran          -63 09feb33 6770
;
```

DATA Step to Create the Data Set USCLIM.TEMPCHNG

```
libname usclim 'SAS-data-library';

data usclim.tempchng;
   input @1 State $char13. @15 Date date7. Start_f End_f Minutes;
   Diff=End_f-Start_f;
   informat State $char13. Date date7.;
   format Date date9.;
   datalines;
North Dakota  21feb18 -33 50  720
South Dakota  22jan43 -4  45  2
South Dakota  12jan11 49  -13 120
South Dakota  22jan43 54  -4  27
South Dakota  10jan11 55  8   15
;
```

Note on Catalogs USCLIM.BASETEMP and USCLIM.REPORT

The catalogs USCLIM.BASETEMP and USCLIM.REPORT are used to show how the DATASET S procedure processes both SAS data sets and catalogs. The contents of these catalogs are not important in the context of this book. In most cases, you would use SAS/AF, SAS/FSP, or other SAS products to create catalog entries. You can test the examples in this section without having these catalogs.

DATA Step to Create the Data Set CLIMATE.HIGHTEMP

```
libname climate 'SAS-data-library';

data climate.hightemp;
   input Place $ 1-13 Date $ Degree_f Degree_c;
   datalines;
Libya         13sep22 136 58
California    10jul13 134 57
Israel        21jun42 129 54
Argentina     11dec05 120 49
Saskatchewan  05jul37 113 45
;
```

DATA Step to Create the Data Set CLIMATE.LOWTEMP

```
libname climate 'SAS-data-library';

data climate.lowtemp;
   input Place $ 1-13 Date $ Degree_f Degree_c;
   datalines;
Antarctica    21jul83 -129 -89
Siberia       06feb33 -90  -68
Greenland     09jan54 -87  -66
Yukon         03feb47 -81  -63
Alaska        23jan71 -80  -67
;
```

DATA Step to Create the Data Set PRECIP.RAIN

```
libname precip 'SAS-data-library';

data precip.rain;
   input Place $ 1-12 @13 Date date7. Inches Cms;
   format Date date9.;
   datalines;
La Reunion  15mar52 74 188
Taiwan      10sep63 49 125
Australia   04jan79 44 114
Texas       25jul79 43 109
Canada      06oct64 19 49
;
```

DATA Step to Create the Data Set PRECIP.SNOW

```
libname precip 'SAS-data-library';

data precip.snow;
   input Place $ 1-12 @13 Date date7. Inches Cms;
   format Date date9.;
   datalines;
Colorado    14apr21 76 193
Alaska      29dec55 62 158
France      05apr69 68 173
;
```

DATA Step to Create the Data Set STORM.TORNADO

```
libname storm 'SAS-data-library';

data storm.tornado;
   input State $ 1-12 @13 Date date7. Deaths Millions;
   format Date date9. Millions dollar6.;
   label Millions='Damage in Millions';
   datalines;
Iowa        11apr65 257 200
Texas       11may70 26  135
Nebraska    06may75 3   400
Connecticut 03oct79 3   200
Georgia     31mar73 9   115
;
```

Previous Page | Next Page | Top of Page