Introduction to SAS Statistical Package

Lecture 3

- Topics
  - DATA STEP statements
  - PUT and OUTPUT statements
  - Subsetting files
  - Data set options
  - Concatenating and merging files

- References
  The Little SAS Book - Chapter 5

DATA STEP
- begins with DATA statement
- consists of statements that create a SAS data set
- names the data set
- defines variables and their location or reads in an existing SAS data set
- step ends when SAS encounters a new step (DATA or PROC statement) or a RUN statement

DATA STEP
- creates SAS data set
- always begins with a DATA statement
  - DATA name of data set;
- reads and modifies data
- includes numeric and character functions, IF THEN/ELSE logic, DO loops
- 2-level name -- permanent data set

DATA STEP STATEMENTS
- DATA beginning of step names output SAS data set
- SET reads in existing SAS data set
- DROP drops variables from data set
- KEEP keeps variables in data set
- IF /THEN IF/THEN ELSE conditional logic used to create or redefine variables, subset files

DATA STEP STATEMENTS
- LENGTH Specifies the number of bytes for storing variables
- LABEL assigns descriptive labels <=256 characters
MORE DATA STEP STATEMENTS

- WHERE selects observations for processing
- PUT writes lines of text to file or SAS log
- OUTPUT writes data to SAS data set

GLOBAL STATEMENTS

- LIBNAME associates between a libref and a SAS data library
- TITLE prints at the top of each page in OUTPUT window

GLOBAL STATEMENTS

- provide information to SAS
- use anywhere in a SAS program
- are not executable; they take effect as soon as SAS compiles program statements
- in effect for the SAS session or until replaced with new statement

LIBNAME STATEMENT

- association between a libref and a SAS data library
- lasts only for the duration of the SAS session or until you change it or discontinue it with another LIBNAME statement.

TITLE STATEMENT

- enclose in quotes
- prints at the top of each page in OUTPUT window
- limit of 10 (Title, Title2, ..Title10)
- limit of 132 characters each
- in effect for session until new title statement

TITLE STATEMENT

Syntax:

\texttt{TITLEn ‘text string’;}

Example:

\texttt{Title1 ‘Example of title’; Title2 ‘Second title’;}

\texttt{proc print data=\texttt{mylib.prec1}; Title ‘Listing of Prec1’; run;}
The PUT Statement

PUT statement writes information to the log. This is useful to determine
- which piece of code is executing
- which piece of code is not executing
- the current value of a particular variable
- the current values of all variables.

writes lines of text to SAS LOG window (text strings and variable values)
- used for debugging
- values can be written in list, column or formatted style

writes the literal text string.
Example:

```
PUT 'text';
```

writes the name of the variable followed by an equal sign and the value.
Example:

```
PUT variable-name=;
```

writes the name of each variable in the PDV followed by an equal sign and the value of the variable.

```
PUT ALL;
```

```
data temp; set mylib.prec1 ;
Hgtm=hgt*.0254;
Wgtkg=wgt*.454;
bmi=wgtkg/(hgtm*hgtm);
if 0<bmi<24 then bmigrp=0;
else if bmi>=24 then bmigrp=1;
put id= bmi= bmigrp= name=;
run;
```
NOTE: Variable name is uninitialized.
id=24904 bmi=22.614486808 bmigrp=0 name=.
id=24906 bmi=19.783992855 bmigrp=0 name=.
id=24907 bmi=23.424404057 bmigrp=0 name=.
id=24908 bmi=25.112800997 bmigrp=1 name=.

OUTPUT STATEMENT
- every DATA step has an implied OUTPUT when the run statement is executed
- tells SAS to write out the current observation before returning to the beginning of the DATA step
- can use the OUTPUT statement anywhere in a Data step; the run statement no longer outputs

Syntax
```
OUTPUT <filename(s)>
```

SUBSETTING DATA

OUTPUT STATEMENT
- can write observations to multiple SAS data sets
- tells SAS to write the current observation to a SAS data set immediately, not at the end of the DATA step.
- any data set named on the OUTPUT statement must appear in DATA statement
- observation is written to all data files listed in DATA statement if none specified on the OUTPUT statement.

OUTPUT STATEMENT
- can create more than one SAS data set from one input file

Syntax: OUTPUT <filename(s)>

Example Class3_2.SAS:
```
DATA males females;
set mylib.prec1;
IF SEX=0 then output females;
ELSE output males;
RUN;
```

Subsetting Rows
In a DATA step, you can subset the rows (observations) in a SAS data set with a:

WHERE statement
DELETE statement
subsetting IF statement.

The WHERE statement in a DATA step can also be used in a PROC step.
Deleting Rows

• You can use a DELETE statement to control which rows are written to the SAS data set.

• General form of the DELETE statement:

```sas
IF expression THEN DELETE;
```

• The expression can be any SAS expression.

Deleting Rows

```sas
data hgt; set mylib.class2;
  if height=. then delete;
run;
```

Deleting Rows

```
proc print data=hgt;
  var birthdate weight height;
  format birthdate mmdyy10.;
run;
```

<table>
<thead>
<tr>
<th>Iname</th>
<th>birthdate</th>
<th>weight</th>
<th>height</th>
</tr>
</thead>
<tbody>
<tr>
<td>Richardson</td>
<td>02/03/1966</td>
<td>189</td>
<td>72</td>
</tr>
<tr>
<td>Lowrey</td>
<td>03/23/1970</td>
<td>235</td>
<td>71</td>
</tr>
<tr>
<td>Tierney</td>
<td>.</td>
<td>229</td>
<td>70</td>
</tr>
<tr>
<td>Sommers</td>
<td>06/10/1959</td>
<td>156</td>
<td>66</td>
</tr>
</tbody>
</table>

Kegan has been deleted

Deleting Rows

• You can use a subsetting IF statement to control which rows are written to the SAS data set.

• General form:

```sas
IF expression;
```

• The expression can be any SAS expression.

• The subsetting IF statement is valid only in a DATA step.

Selecting Rows

Process Flow of a Subsetting IF

```
DATA Statement
  Read Observation or Record
    IF expression
      False
        Continue Processing Observation
      True
        Output Observation to SAS Data Set
```

Process Flow of a Subsetting IF

```
DATA Statement
  Read Observation or Record
    IF expression
      False
        Continue Processing Observation
      True
        Output Observation to SAS Data Set
```
Selecting Rows
Select rows that have known height

```sas
data hgt; set mylib.class2;
if height^=.;
run;
```

Deleting Rows

```sas
proc print data=hgt;
var birthdate weight height;
format birthdate mmddyy10.;
run;
```

EXPRESSIONS
- character constants enclosed in quotes
- compares one thing to another
- usual logical, arithmetic, and relational operators

Logical
- AND
- OR
- NOT

Arithmetic
- +
- -
- /
- *
- **

Relational
- =
- <
- >
- <=
- >=

- use parentheses to group operations

Subsetting Data: WHERE Statement

The WHERE statement
- enables you to select observations that meet a certain condition
- can be used in DATA step and with most SAS procedures.

Subsetting Data: WHERE Statement

General form of the WHERE statement:

```
WHERE where-expression;
```

- where-expression is a sequence of operands and operators.
- Operands include
  - variables
  - constants.

Subsetting Data: WHERE Statement

Operators include
- comparison operators
- logical operators
- special operators
- functions.
Comparison Operators

<table>
<thead>
<tr>
<th>Mnemonic</th>
<th>Symbol</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>EQ</td>
<td>=</td>
<td>equal to</td>
</tr>
<tr>
<td>NE</td>
<td>^=</td>
<td>not equal to</td>
</tr>
<tr>
<td>GT</td>
<td>&gt;</td>
<td>greater than</td>
</tr>
<tr>
<td>LT</td>
<td>&lt;</td>
<td>less than</td>
</tr>
<tr>
<td>GE</td>
<td>=&gt;</td>
<td>greater than or equal to</td>
</tr>
<tr>
<td>LE</td>
<td>&lt;=</td>
<td>less than or equal to</td>
</tr>
<tr>
<td>IN</td>
<td></td>
<td>equal to one of a list</td>
</tr>
</tbody>
</table>

Character comparisons are case-sensitive. The IN operator allows commas or blanks to separate values.

Examples:

where Salary>25000;
where EmpID='0082';
where Salary=.;
where LastName=' '; 
where JobCode in('PILOT', 'FLTAT');
where JobCode in('PILOT', 'FLTAT');

Logical Operators

AND
if both expressions are true, then the compound expression is true

where JobCode='FLTAT' and Salary>50000;

OR
if either expression is true, then the compound expression is true

where JobCode='PILOT' or JobCode='FLTAT';

NOT
 can be combined with other operators to reverse the logic of a comparison.

where JobCode not in('PILOT', 'FLTAT');

Selecting Rows

Select rows that have a known height

```sas
data hgt; set mylib.class2;
if height^=.;
run;
```

Example: Class3_3.SAS

Select only males

```sas
data male; set mylib.prec1;
if sex=1; *subsetting if;
proc print data=male;

data male; set mylib.prec1;
where sex=1; *data step where;
proc print data=male;
proc print data=mylib.prec1;
where sex=1; *proc step where;
run;
```
WHERE or Subsetting IF?

<table>
<thead>
<tr>
<th>Step and Usage</th>
<th>WHERE</th>
<th>IF</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROC step</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>DATA step (source of variable)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Assignment statement</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>SET statement (single data set)</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>SET/MERGE (multiple data sets)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Variable in ALL data sets</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Variable not in ALL data sets</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>

DATA SET OPTIONS

specify actions that apply only to the SAS data set with which they appear

- select only the first or last n observations for processing
- dropping variables from processing or from the output data set
- renaming variables

DATA SET OPTIONS

• DATA statement - applies to output file
• SET or MERGE - applies to input file
• PROC statement with DATA= option applies to procedure results

DATA SET OPTIONS

Specify a data set option(s) in parentheses after a SAS data set name
(separate multiple data set options with spaces)

Syntax:

```plaintext
DATA data-set-name (options);
SET data-set-name (options);
```

Controlling Which Observations Are Read

• By default, SAS begins processing a SAS data set with the first observation and continues processing until the last observation.
• The FIRSTOBS= and OBS= data set options can be used to control which observations are processed.
• You can use FIRSTOBS= and OBS= with input data sets only. You cannot use either data set option in the DATA statement.
The OBS= Data Set Option

• The OBS= data set option specifies an ending point for processing an input data set.

\[
\text{SAS-data-set(OBS=\text{n})}
\]

• This option specifies the number of the last observation to process, not how many observations should be processed.

The FIRSTOBS= Data Set Option

• The FIRSTOBS= data set option specifies a starting point for processing an input data set.

\[
\text{SAS-data-set(FIRSTOBS=\text{n})}
\]

• FIRSTOBS= and OBS= are often used together to define a range of observations to be processed.

DATA SET OPTIONS

```
DATA temp; SET old(FIRSTOBS=20 OBS=50);
```

reads until it reaches observation 50 starting at number 20

```
PROC PRINT DATA=temp (OBS=50);
```

print out first 50 observations

DATA SET OPTIONS

- **KEEP = variable list** lists variables to keep
- **DROP = variable list** lists variables to drop
- Syntax help -- type HELP DSOPTIONS in command box

Controlling Variable Input

- In the DATA step, the DROP and KEEP statements apply only to output SAS data sets.
- However, the DROP= and KEEP= data set options can apply to both input and output SAS data sets.

KEEP DATA SET OPTION

Subsetting variables when reading data file

Syntax:

```
DATA filename; SET oldfile(KEEP=varnames);
```

**KEEP** variables are the only variables which may be used in subsequent program statements and written to output data set

Example:

```
DATA temp2 ;
SET templ (keep=id var1 var2);
```
**KEEP DATA SET OPTION**

Subsetting variables when creating a file

Syntax:

```sas
DATA filename (KEEP=varnames); SET oldfile;
```

**KEEP** variables are the only variables which are written to output data set. All variables in the SET file can be used in the data step.

Example:

```sas
DATA temp2 (keep=id var1 var2); SET temp1;
```

---

**Example: CLASS3_4.sas**

```sas
data temp( keep= id msbp mdbp sex);
set mylib.prc1 (obs=20);
proc contents data=temp;
title 'Example - data set options'; run;
```

---

**EXERCISE I**

---

**Concatenating SAS Data Sets**

- Use the SET statement in a DATA step to concatenate SAS data sets.

- General form of a DATA step concatenation:

```sas
DATA SAS-data-set:
 SET SAS-data-set1 SAS-data-set2 . . .;
<other SAS statements>
RUN;
```

---

**SET STATEMENT**

Combined output file contains:

- total number of observations in all files
- duplicates if IDs exist in more than once in any file
- total number of all variables from all input files
- variable will be set to missing for all observations from file listed on the SET statement if the file does not contain a variable present in another file
Concatenating SAS Data Sets
You can read any number of SAS data sets with a single SET statement.

```
data work.qtr1;
set work.jan work.feb work.mar;
run;
```

Example: Class3_5.SAS

### SET1
```
 lname  fname  age  sex  race  wgt  hgt
Richardson  Thomas  34  1  2  189  72
Lowrey  Jane  29  2  1  235  71
Tierney  John  32  1  2  229  70
Sommers  Joan  33  2  1  156  66
Kegan  Charlotte  19  2  1  140  65
Jones  Robert  37  1  2  156  67
```

Example: Class3_5.SAS

### SET2
```
lname  fname  age  sex  race  wgt  hgt  grp
Richardson  Thomas  24  1  2  189  72  2
Peterson  Jane  32  2  1  235  64  1
Holmes  Sherlock  32  1  2  229  70  1
Smith  Joan  33  2  1  156  66  1
```

Example: Class3_5.SAS

```
data mylib.setall; set mylib.set1 mylib.set2;
proc print data=mylib.setall;
title 'setall data set';
proc contents;
run;
```

Merging SAS Data Sets

```
data work.qtr1;
set work.jan work.feb work.mar;
run;
```

```
data mylib.setall; set mylib.set1 mylib.set2;
proc print data=mylib.setall;
title 'setall data set';
proc contents;
run;
```
Merging SAS Data Sets

– Use the MERGE statement in a DATA step to join corresponding observations from two or more SAS data sets.
– General form of a DATA step match-merge:

```
DATA SAS-data-set;
MERGE SAS-data-sets;
BY BY-variable(s);
<other SAS statements>
RUN;
```

Example:
```
DATA all;
MERGE mylib.ex1 mylib.ex2;
by id;
```

Example : Class3_6.sas
```
proc sort data=mylib.ex1; by id;
proc sort data=mylib.ex2; by id;
data all;
merge mylib.ex1 mylib.ex2; by id;
proc print data=all;
run;
```

MERGE STATEMENT

Syntax:
```
DATA SAS-data-set;
MERGE SASfile1 SASfile2 ... ;
[BY key-variables;] Optional
```

Example: First, sort by the key variables
```
DATA all;
MERGE mylib.ex1 mylib.ex2;
by id;
```

SORT PROCEDURE

Syntax of PROC SORT
```
Proc sort data=SASfile1;
by var1 .. varn;
```

Example:
```
Proc sort data=mylib.ex1; by id;
Proc sort data=mylib.ex2; by id;
```

MERGING

Combined MERGE output file contains:
- total number of observations in all files for key variables
- total of all variables in all files
- variables are set to missing for any observation that does not match

Example : Class3_6.sas
```
proc sort data=mylib.ex1; by id;
proc sort data=mylib.ex2; by id;
data all;
merge mylib.ex1 mylib.ex2; by id;
proc print data=all;
run;
```
EXAMPLE OF MATCH MERGING

<table>
<thead>
<tr>
<th>ID</th>
<th>AGE</th>
<th>SEX</th>
<th>ID</th>
<th>AGE</th>
<th>SEX</th>
<th>TRT</th>
</tr>
</thead>
<tbody>
<tr>
<td>121</td>
<td>21</td>
<td>M</td>
<td>121</td>
<td>21</td>
<td>M</td>
<td>1</td>
</tr>
<tr>
<td>223</td>
<td>34</td>
<td>F</td>
<td>312</td>
<td>23</td>
<td>M</td>
<td>0</td>
</tr>
<tr>
<td>312</td>
<td>23</td>
<td>M</td>
<td>334</td>
<td>37</td>
<td>F</td>
<td>0</td>
</tr>
<tr>
<td>334</td>
<td>37</td>
<td>F</td>
<td>401</td>
<td>1</td>
<td>F</td>
<td>1</td>
</tr>
<tr>
<td>452</td>
<td>29</td>
<td>F</td>
<td>558</td>
<td>36</td>
<td>M</td>
<td>0</td>
</tr>
</tbody>
</table>

EX1 + EX2 → ALL

MERGE PROBLEMS

- Variables common between files contain the data value from the rightmost mentioned file in the MERGE statement.
- If you are merging two data sets that have variables with the same names, the values from the last data set that was read write over the values read from other data sets. SAS includes only one variable of a given name in the new data set.

The RENAME= Data Set Option

You can use a RENAME= data set option to change the name of a variable.

General form of the RENAME= data set option:

```
SAS-data-set(RENAME=(old-name-1=new-name-1
  old-name-2=new-name-2
  ...
  old-name-n=new-name-n))
```

Example: Class3_7.SAS

```sas
DATA temp; SET old
  (rename=(age=agenew v23=yob));
```

The RENAME= Data Set Option corrects problem of merging 2 files that contain variables with the same name.

Example: Class3_7.sas

```sas
data without_rename ; merge mylib.ex3
  mylib.ex4; by id;
proc print data=without_rename;
title 'without renaming age';
run;
```
Example : Without_rename

<table>
<thead>
<tr>
<th>id</th>
<th>age</th>
<th>clinic</th>
<th>region</th>
</tr>
</thead>
<tbody>
<tr>
<td>1877</td>
<td>47</td>
<td>3</td>
<td>.</td>
</tr>
<tr>
<td>3422</td>
<td>.</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>4511</td>
<td>34</td>
<td>.</td>
<td>1</td>
</tr>
<tr>
<td>8922</td>
<td>35</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>8922</td>
<td>22</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

Example : Class3_7.sas

```sas
data rename ;
merge mylib.ex3 mylib.ex4(rename=age=age2);
by id;
proc print data=rename;
title 'rename age';
run;
```

Example : rename

<table>
<thead>
<tr>
<th>id</th>
<th>age</th>
<th>clinic</th>
<th>region</th>
<th>age2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1877</td>
<td>47</td>
<td>3</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>3422</td>
<td>23</td>
<td>2</td>
<td>3</td>
<td>.</td>
</tr>
<tr>
<td>4511</td>
<td>33</td>
<td>.</td>
<td>1</td>
<td>34</td>
</tr>
<tr>
<td>8922</td>
<td>46</td>
<td>2</td>
<td>1</td>
<td>35</td>
</tr>
<tr>
<td>8922</td>
<td>46</td>
<td>2</td>
<td>1</td>
<td>22</td>
</tr>
</tbody>
</table>

The IN= Data Set Option

Use the IN= data set option to determine which data set(s) contributed to the current observation.

```
variable
```

General form of the IN= data set option:

variable is a temporary numeric variable that has two possible values:

0 indicates that the data set did not contribute to the current observation.
1 indicates that the data set did contribute to the current observation.

DATA SET OPTION - IN

- IN variables may be addressed to subset for the output file - available for use during the DATA step
- available only within the current DATA step
- must be set to equal a permanent variable in an assignment statement to be retained in the dataset

DATA SET OPTION - IN

```
DATA filename;
MERGE filen1 (IN=inname1) filen2 (IN=inname2) ... filenn(IN=inname);
```

IN is a data set option on the MERGE statement
Example: Class3_8.sas

- Use a dataset option to assign an IN variable "count1" for the ex1 file
- Use a subsetting IF to restrict output file to only those records that are on ex1 (Exclude records that only exist on ex2)

Note: can not use WHERE statement to subset because count1 variable does not exist on ex1. It was created during the DATA step.

```sas
data all;
merge mylib.ex1(in=count1)
   mylib.ex2; by id;
   if count1=1;
title 'includes only records on ex1'; run;
```

Note: Record 401 will not be included in output file.

---

Example: Class3_9.sas

- Use a dataset option to assign an IN variable "count1" for the ex1 file and "count2" for the ex2 file
- Use a subsetting IF to restrict output file to only those records that exist on both files

Note: can not use WHERE statement to subset because count1 variable does not exist on ex1. It was created during the DATA step.

```sas
data all2;
merge mylib.ex1(in=count1)
   mylib.ex2 (in=count2); by id;
   if count1=1 and count2=1;
proc print data=all2;
title 'includes only records that match';
run;
```

Note: Record 223, 401, 452 will not be included in output file.