

ICER Biostatistics Unit



Introduction to the SAS Data Step

Presented by:

Tara Dudley, Mstat

Jennifer Hoff, MS



What is SAS?

- * Statistical Analysis Software*
developed by SAS Institute in Cary, NC
- * Main uses of SAS
 - Data Management
 - Analysis
 - Reporting techniques

Introduction to SAS

Data Step

- * General overview of Basic components
- * Temporary vs. Permanent SAS data sets
- * Set and Merge
- * Creating new variables
- * Working with Dates
- * If then else
- * Reducing a data set
- * Formats and Labels

Basics about SAS

- * SAS is composed of three windows
 - Program Editor
 - where you write and submit programs
 - Log
 - where SAS displays messages which indicate any errors that may be in the program
 - Output
 - where results appear after submitting program

Basic Components of SAS

- * Every SAS program is constructed using the Data step and/or Procedures
- * Any combination of Data steps and/or Procedures may be used
- * Run statements should be used throughout program

Basic Components of SAS, Continued

* Data step

- SAS statements that read data, create new data sets or variables, modify data sets, perform calculations

* Procedures

- SAS statements that can perform statistical analyses, create graphs

Basic Components of SAS, Continued

* Run statement

- Tells SAS that the Data step or Procedure has ended
- Good practice to end each Data step or Procedure with a run statement
- Must still SUBMIT the SAS program for it to be processed

Most Common Programming Mistake

- * Failing to end a SAS statement with a semi-colon

▪
;

- * Must end every SAS statement with a semi-colon!!!!

Getting Data into a SAS Data Set

- * Text or ASCII file
 - Infile command (or Data line if small data set) and the input statement
- * Data from another software program
 - Use import wizard
 - May need to save data in another format as an intermediate step
- * Existing SAS dataset \Leftarrow Today's focus

Types of SAS Data Sets

* Temporary

- Only exist during the SAS session
- One-part data set name (for example, lab)

* Permanent

- Can be saved to disk or to computer hard drive
- Available for use anytime
- Two-part data set name (for example, mydata.lab)

Permanent Data Sets - Libraries

- * Permanent SAS data sets are saved in a pre-designated subdirectory or folder
- * SAS refers to as a Library
- * Libraries can be specified through a graphic interface or by using the libname statement

Permanent Data Sets - Libname Statement

* Libname statement

- Tells SAS where the permanent data set is to be stored or is already stored

* Syntax

- Libname mydata 'c:/sasdata';
 ↓ ↓ ↓
keyword libref directory

Permanent Data Sets - Libname Statement

* Syntax

```
Data mydata.lab;
```

Two level name

- Library reference.data set name
- Creates a permanent data set, lab, located in directory c:\sasdata

Data Statement

- * Indicates the beginning of a SAS Data step
- * Begins with the word DATA followed by the name of the SAS data set (should begin with a letter and be 8 or fewer characters long)
- * Example
 - Data lab;

Using Existing SAS Data Set

* Set statement

- Copy: creates a data set with all the variables and observations of the data set named in the set statement
- Append: stacks 2 or more data sets together

Set Statement

Example 1- Copy

* SAS code

```
Libname in 'c:\';
```

```
data white;  
  set in.sourceA;  
run;
```

```
data yellow;  
  set sourceB;  
run;
```

Set Statement

Example 1 - Copy

* Output Data sets

White Data set

| ID | GROUP | DOB | GENDER |
|----|-------|-----|--------|
|----|-------|-----|--------|

| | | | |
|---|---|-----------|---|
| 1 | A | 01JAN1910 | F |
| 2 | A | 02FEB1920 | M |
| 3 | A | 03MAR1930 | M |
| 4 | A | 04APR1940 | F |
| 5 | A | 05MAY1950 | |

Yellow Data set

| ID | GROUP | GENDER |
|----|-------|--------|
|----|-------|--------|

| | | |
|---|---|------|
| 3 | B | Fem |
| 5 | B | Male |
| 6 | B | Fem |

Set Statement

Example 2 - Append

* SAS code

```
data twosets;  
  set white yellow;  
run;
```

* SAS log

- NOTE: The data set WORK.TWOSETS has 8 observations and 4 variables.
- NOTE: The DATA statement used 0.02 seconds.

Existing SAS Data Sets

White Data set

| ID | GROUP | DOB | GENDER |
|----|-------|-----|--------|
|----|-------|-----|--------|

| | | | |
|---|---|-----------|---|
| 1 | A | 01JAN1910 | F |
| 2 | A | 02FEB1920 | M |
| 3 | A | 03MAR1930 | M |
| 4 | A | 04APR1940 | F |
| 5 | A | 05MAY1950 | |

Yellow Data set

| ID | GROUP | GENDER |
|----|-------|--------|
|----|-------|--------|

| | | |
|---|---|------|
| 3 | B | Fem |
| 5 | B | Male |
| 6 | B | Fem |

Set Statement

Example 2 - Append

* SAS Output

| ID | GROUP | DOB | GENDER |
|----|-------|-----------|--------|
| 1 | A | 01JAN1910 | F |
| 2 | A | 02FEB1920 | M |
| 3 | A | 03MAR1930 | M |
| 4 | A | 04APR1940 | F |
| 5 | A | 05MAY1950 | |
| 3 | B | . | Fem |
| 5 | B | . | Male |
| 6 | B | . | Fem |

Using Existing SAS Data Set

* Merge statement

- Combines two or more data sets
- Often used in conjunction with the By statement
- Leaving off the By statement can lead to unexpected and possibly disastrous results
- Warning: if a variable exists in both data sets, the output data set will contain the value of the variable from the rightmost data set in the merge statement

Merge Statement Example 3

* **WARNING** - RARELY USED !!!

* SAS code

```
data mergeset;  
  merge white yellow;  
run;
```



* SAS log

- NOTE: The data set WORK.MERGESET has 5 observations and 4 variables.
- NOTE: The DATA statement used 0.02 seconds.

Existing SAS Data Sets

White Data set

| ID | GROUP | DOB | GENDER |
|----|-------|-----|--------|
|----|-------|-----|--------|

| | | | |
|---|---|-----------|---|
| 1 | A | 01JAN1910 | F |
| 2 | A | 02FEB1920 | M |
| 3 | A | 03MAR1930 | M |
| 4 | A | 04APR1940 | F |
| 5 | A | 05MAY1950 | |

Yellow Data set

| ID | GROUP | GENDER |
|----|-------|--------|
|----|-------|--------|

| | | |
|---|---|------|
| 3 | B | Fem |
| 5 | B | Male |
| 6 | B | Fem |

Merge Statement Example 3

* SAS output



| ID | GROUP | DOB | GENDER |
|----|-------|-----------|--------|
| 3 | B | 01JAN1910 | Fem |
| 5 | B | 02FEB1920 | Male |
| 6 | B | 03MAR1930 | Fem |
| 4 | A | 04APR1940 | F |
| 5 | A | 05MAY1950 | |

Matched Merge - By Statement

- * Almost always used in conjunction with the By statement
- * When using a matched merge, the variable you want to match on is identified
- * Each data set must be sorted by the match variable before the merge

Merge Statement with By - Example 4

* SAS code - sorting data sets

```
Proc sort data=yellow;
```

```
  by id;
```

```
run;
```

```
Proc sort data=white;
```

```
  by id;
```

```
run;
```

Merge Statement with By - Example 4

* SAS log - sorting data sets

- NOTE: The data set WORK.YELLOW has 3 observations and 3 variables.
- NOTE: The PROCEDURE SORT used 0.02 seconds.

- NOTE: The data set WORK.WHITE has 5 observations and 4 variables.
- NOTE: The PROCEDURE SORT used 0.02 seconds.

Merge Statement with By - Example 4

- * SAS code - merging with By statement

```
data mergeby;  
  merge white yellow;  
  by id;  
run;
```
- * SAS log - merging with By statement
 - NOTE: The data set WORK.MERGEBY has 6 observations and 4 variables.
 - NOTE: The DATA statement used 0.02 seconds.

Existing SAS Data Sets

White Data set

ID GROUP DOB GENDER

1 A 01JAN1910 F

2 A 02FEB1920 M

3 A 03MAR1930 M

4 A 04APR1940 F

5 A 05MAY1950

Yellow Data set

ID GROUP GENDER

3 B Fem

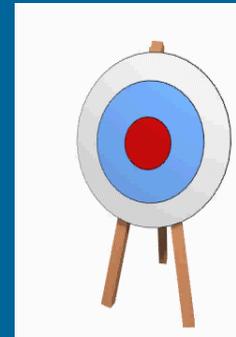
5 B Male

6 B Fem

Merge Statement with By - Example 4

* SAS Output

| ID | GROUP | DOB | GENDER |
|----|-------|-----------|--------|
| 1 | A | 01JAN1910 | F |
| 2 | A | 02FEB1920 | M |
| 3 | B | 03MAR1930 | Fem * |
| 4 | A | 04APR1940 | F |
| 5 | B | 05MAY1950 | Male * |
| 6 | B | . | Fem * |



Matched Merge - In Option

- * When merging data sets, it is sometimes necessary to know what data came from which data set
- * In option with merge statement can be used for this purpose
- * Syntax - after the data set name (in=temporary variable)

Merge Statement with In Option - Example 5

* SAS code

```
data mergecom;  
    merge white (in=inwt) yellow (in=inyel);  
by id;  
if inwt=inyel;  
    * Alternate if inwt=1 and inyel=1;  
run;
```

Merge Statement with In Option - Example 5

* SAS log

- NOTE: There were 5 observations read from the dataset WORK.WHITE.
- NOTE: There were 3 observations read from the dataset WORK.YELLOW.
- NOTE: The data set WORK.MERGECOM has 2 observations and 4 variables.
- NOTE: DATA statement used:
real time 0.02 seconds
cpu time 0.02 seconds

Merge Statement with In Option - Example 5

* SAS output

| ID | GROUP | DOB | GENDER |
|----|-------|-----------|--------|
| 3 | B | 03MAR1930 | Fem |
| 5 | B | 05MAY1950 | Male |



Overwriting in Merges



- * If the two data sets have variables with the same name, the value of the variable in the data set named last will overwrite the value of the variable in the data set named first.
- * Option - rename those variables in the merging
- * Syntax - after the data set name (rename = (oldvariable = newvariable))

Merge Statement with Rename - Example 6

* SAS code

```
data mergeren;  
    merge white (rename=(gender=wgender))  
          yellow (rename=(gender=ygender));  
    by id;  
run;
```

Merge Statement with Rename - Example 6

* SAS log

- NOTE: There were 5 observations read from the dataset WORK.WHITE.
- NOTE: There were 3 observations read from the dataset WORK.YELLOW.
- NOTE: The data set WORK.MERGEREN has 6 observations and 5 variables.
- NOTE: DATA statement used:

| | |
|-----------|--------------|
| real time | 0.01 seconds |
| cpu time | 0.01 seconds |

Merge Statement with Rename - Example 6

* SAS output

| ID | GROUP | DOB | WGENDER | YGENDER |
|----|-------|-----------|---------|---------|
| 1 | A | 01JAN1910 | F | |
| 2 | A | 02FEB1920 | M | |
| 3 | B | 03MAR1930 | M | Fem |
| 4 | A | 04APR1940 | F | |
| 5 | B | 05MAY1950 | | Male |
| 6 | B | | | Fem |

Types of Merges

- * One to One - Today's focus
- * Many to One
- * One to Many
- * Many to Many - Need SQL code

Modifying the Data

- * Good practice not to overwrite the original data set or variables
- * Can use arithmetic operations and functions to create new variables
- * Can use IF Then (IF Then Else) statements to create new variables



Dates in SAS

- * Dates are stored internally as integers
 - Referenced from January 1, 1960
- * Allows for easy sorting and calculations



Dates in SAS

| <u>Date</u> | <u>Internally</u> |
|---------------|-------------------|
| June 30, 1959 | -185 |
| Jan 1, 1960 | 0 |
| Jan 20, 1960 | 19 |
| Feb 20, 1960 | 50 |
| Dec 31, 1960 | 365 |
| Nov 30, 2000 | 14944 |



Calculating Age Attempt 1

* SAS code

```
data wtage;  
  set white ;  
  refdate='30Nov00'd;  
  agetemp=refdate-dob;  
  
run;
```



Calculating Age Attempt 1

* SAS output



| ID | DOB | AGETEMP |
|----|-----------|---------|
| 1 | 01JAN1910 | 33206 |
| 2 | 02FEB1920 | 29522 |
| 3 | 03MAR1930 | 25840 |
| 4 | 04APR1940 | 22155 |
| 5 | 05MAY1950 | 18472 |



Calculating Age Attempt 2

* SAS code

```
data wtage;  
  set white;  
  refdate='30Nov00'd;  
  agetemp=(refdate-dob)/365.25;  
run;
```



Calculating Age

Attempt 2

* SAS Output

| ID | DOB | AGETEMP |
|----|-----------|---------|
| 1 | 01JAN1910 | 90.9131 |
| 2 | 02FEB1920 | 80.8268 |
| 3 | 03MAR1930 | 70.7461 |
| 4 | 04APR1940 | 60.6571 |
| 5 | 05MAY1950 | 50.5736 |



Calculating Age Attempt 3

* SAS code

```
data wtage;  
  set white ;  
  refdate='30Nov00'd;  
  age=int((refdate-dob)/365.25);  
run;
```



Calculating Age

Attempt 3

* SAS Output



| ID | DOB | AGE |
|----|-----------|-----|
| 1 | 01JAN1910 | 90 |
| 2 | 02FEB1920 | 80 |
| 3 | 03MAR1930 | 70 |
| 4 | 04APR1940 | 60 |
| 5 | 05MAY1950 | 50 |

Multitude of Functions

- * Arithmetic - abs,max,mix,mod,sqrt,sign
- * Array-dim,hbound,lbound
- * Character - compress,index,scan,substr
- * Date & Time - date,intck,mdy,timepart
- * Mathematical - log ordinal
- * Probability - poisson,probchi,probnorm
- * Random Number - normal ranuni
- * Sample statistic - mean,n,std,sum

If - Then - Else

- * Another way to create new variables from existing variables
- * Syntax
 - IF <Condition> THEN <Action>
ELSE <Different Action>

If-Then-Else Example Attempt 1

* SAS code

```
data wtage2;  
  set wtage;  
  if age<65 then agecat=1;  
  else agecat=2;  
  
run;
```

If-Then-Else Example Attempt 1

* SAS output

| Obs | AGE | AGECAT |
|-----|-----|--------|
| 1 | 90 | 2 |
| 2 | 80 | 2 |
| 3 | 70 | 2 |
| 4 | 60 | 1 |
| 5 | 50 | 1 |

“Realistic” Data Set

* Data set - AgeEx

| ID | AGE |
|----|-----|
| 10 | 27 |
| 20 | 65 |
| 30 | . |
| 40 | 777 |

If-Then-Else Example

Attempt 1

* Data set - AgeEx



| ID | AGE | AGECAT |
|----|-----|--------|
| 10 | 27 | 1 |
| 20 | 65 | 2 |
| 30 | . | 1 |
| 40 | 777 | 2 |

If-Then-Else Example Attempt 2 (Better)

* SAS code

```
data cat2;  
  set ageex;  
  agecat=0;    *Set an initial value;  
  if 0<age<65 then agecat=1;  
    else if 65<=age<100 then agecat=2;  
    *Designate ranges;  
  if age=. then agecat=.;  
run;
```

If-Then-Else Example Attempt 2 (Better)

* SAS output

| Obs | AGE | AGECAT |
|-----|-----|--------|
| 1 | 27 | 1 |
| 2 | 65 | 2 |
| 3 | . | . |
| 4 | 777 | 0 |

Reducing the Size of a Data Set

- * Occasionally, may want to reduce the size of a data set by:
 - reducing the number of observations (rows) using the where statement
 - reducing the number of variables (columns) using drop and keep statements

Where Statement

- * Where statement can be used to create a subset data set



- * Selects observations which meet the specified criteria - reducing the number of observations (rows)

Where Statement Example

* SAS code

```
data xy;  
  set white yellow;  
  where gender='M';  
run;
```

Existing SAS Data Sets

White Data set

| ID | GROUP | DOB | GENDER |
|----|-------|-----|--------|
|----|-------|-----|--------|

| | | | |
|---|---|-----------|-----|
| 1 | A | 01JAN1910 | F |
| 2 | A | 02FEB1920 | M * |
| 3 | A | 03MAR1930 | M * |
| 4 | A | 04APR1940 | F |
| 5 | A | 05MAY1950 | |

Yellow Data set

| ID | GROUP | GENDER |
|----|-------|--------|
|----|-------|--------|

| | | |
|---|---|--------|
| 3 | B | Fem |
| 5 | B | Male * |
| 6 | B | Fem |

Where Statement Example

* SAS log

- NOTE: There were 2 observations read from the dataset WORK.WHITE. WHERE gender='M';
- NOTE: There were 1 observations read from the dataset WORK.YELLOW. WHERE gender='M';
- NOTE: The data set WORK.XY has 3 observations and 4 variables.

Where Statement Example

* SAS output

| ID | GROUP | DOB | GENDER |
|----|-------|-----------|--------|
| 2 | A | 02FEB1920 | M |
| 3 | A | 03MAR1930 | M |
| 5 | B | . | Male |

Reducing the Data Set Size

- * Reduce the number of observations (rows) - Where statement
- * Reduce the number of variables (columns) - Keep or Drop statements

Keep and Drop Statements

- * Bringing data into the data step
 - Syntax
 - (keep = <list of variables>) or
 - (drop=<list of variables>)
 - after the data set name in the set or merge statement
- * Deciding variables to retain in data set
 - Syntax
 - keep <variable list>

Keep and Drop Statements

* SAS code

```
data wtage;
  set white (keep= id dob);
  refdate='30Nov00'd;
  age=int((refdate-dob)/365.25);
  drop refdate;
  * Alternate keep id dob age;
run;
```

Formatting the Data

- * Creating labels
 - Describe what variable names represent
- * Creating formats
 - Control how the data are displayed
 - Do not affect the actual values of the data

Formatting the Data, Continued

* SAS code

```
proc format;  
  value catfmt  
    1='1: younger than 65'  
    2='2: 65 and Over';  
run;
```

Formatting the Data, Continued

* SAS code

```
data wtage2;  
  set wtage;  
  agecat=0;  
  if 0<age<65 then agecat=1;  
    else if 65<=age<100 then agecat=2;  
  format agecat catfmt.;  
  label agecat="Age Category";  
run;
```

Formatting the Data, Continued

* SAS output

| Agecat | Age Category | | | |
|-------------------|--------------|---------|-------------------------|-----------------------|
| | Frequency | Percent | Cumulative Frequency | Cumulative Percent |
| 1:younger than 65 | 2 | 40.00 | 2 | 40.00 |
| 2:65 and Over | 3 | 60.00 | 5 | 100.00 |

Formatting the Data with Ranges

* SAS code

```
proc format;  
  value age2_  
    20-<30='Age 20-29'  
    30-<40='Age 30-39'  
    40-<50='Age 40-49'  
    50-<60='Age 50-59'  
    60-<70='Age 60-69'  
    70-<79='Age 70-79'  
    80-<100='Age 80 or Over';  
run;
```

Formatting the Data with Ranges, Continued

* SAS code

```
data ageinfo;  
  set agedata;  
  age2=age;  
  format age2 age2_.;  
run;
```

Formatting the Data with Ranges, Continued

* SAS output

| AGE | AGE2 | AGE | AGE2 |
|-----|-----------|-----|-------------|
| 23 | Age 20-29 | 82 | Age Over 80 |
| 34 | Age 30-39 | 200 | 200 |
| 36 | Age 30-39 | | |
| 48 | Age 40-49 | | |
| 52 | Age 50-59 | | |
| 58 | Age 50-59 | | |
| 62 | Age 60-69 | | |
| 71 | Age 70-79 | | |
| 73 | Age 70-79 | | |



Summary

- * Temporary vs. Permanent SAS data sets
- * Set and Merge
- * Creating new variables
- * Dates
- * If then else
- * Reducing a data set
- * Formats and Labels



Analyzing the Data

- * Proc univariate and Proc means
 - Provide descriptive statistics such as the sample size, mean, standard deviation, median, mode, percentile

- * Proc freq
 - Produces counts and percentages for one variable and two variable tables

- * Proc plot
 - Produces a simple plot of two variables