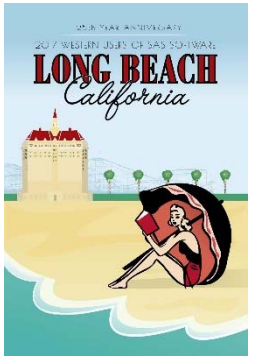
Introduction to DATA Step Programming
SAS Basics II

Susan J. Slaughter, Avocet Solutions



SAS Essentials

- Section for people new to SAS
- Core presentations
  1. How SAS Thinks
  2. Introduction to DATA Step Programming
  3. Introduction to SAS Procedures
- We'll go fast
  - Slides are on my website
- There will be a test
  - Do you have the handout?

DATA versus PROC steps

- Two basic parts of SAS programs

DATA step	PROC step
Begin with DATA statement	Begin with PROC statement
Input and modify data	Perform analysis or task
Create SAS data set	Produce report
Flexibility of programming	Like filling out a form

Susan says: This is a simplification

DATA versus PROC steps

- A simple example

```
DATA temps;
   Fahrenheit = 68;
   Celsius = (Fahrenheit - 32) * 0.5556;
PROC PRINT  DATA = temps;
   TITLE 'Temperature Conversions';
RUN;
```

DATA step

PROC step

Temperature Conversions

Obs	Fahrenheit	Celsius
1	68	20.0016

SAS log

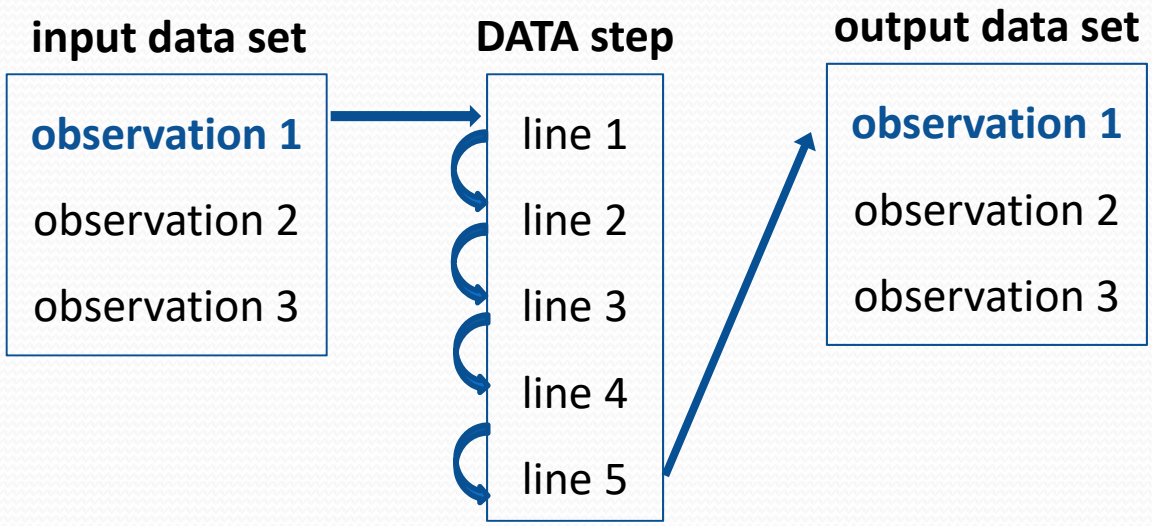
Be sure to check your SAS log!

```
1   DATA temps;
2      Fahrenheit = 68;
3      Celsius = (Farenheit - 32) * 0.5556;
NOTE: The data set WORK.TEMPS has 1 observations and 2 variables.
NOTE: DATA statement used (Total process time):
      real time           0.01 seconds
      cpu time            0.00 seconds

4   PROC PRINT  DATA = temps;
5      TITLE 'Temperature Conversions';
6   RUN;
NOTE: There were 1 observations read from the data set WORK.TEMPS.
NOTE: PROCEDURE PRINT used (Total process time):
      real time           0.09 seconds
      cpu time            0.03 seconds
```

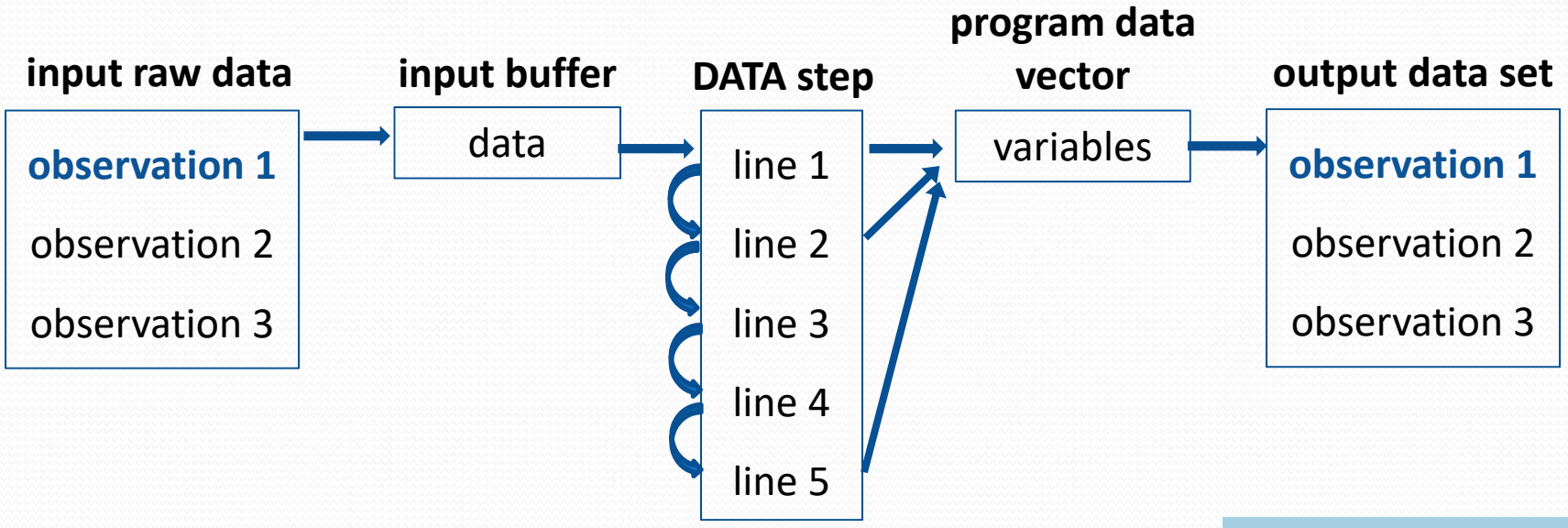
DATA step's built-in loop

- DATA steps execute line-by-line and observation-by-observation



DATA step's built-in loop

- Behind the scenes SAS builds
  - Input Buffer (if reading raw data)
  - Program Data Vector



Reading data in a DATA step

- Raw data
  - INPUT statement
  - With DATALINES or INFILE statement
- SAS data sets
  - SET statement
  - MERGE statement

Reading raw data

- Data can be internal or external to program
  - Internal if inside DATA step
    - Also called instream data
  - External if in separate file
    - Also called text, ASCII, sequential, flat file, CSV files
- INPUT statements read raw data
  - Three basic styles of INPUT statements
  - List style, column style, and formatted style

Reading raw data

- List style input general form:

  INPUT *varname* $ *varname*;

- Data must have
  - Values separated by spaces
  - No embedded spaces in data values
  - No character values longer than 8
  - No special formatting (like dates or dollar signs)
  - Missing values marked with a period (.)

Reading internal raw data

```
* Input student enrollment data;
DATA students;
   INPUT ID $ Name $ AmtPaid Course $ New;
   DATALINES;
78374 Adam    350.00 597 1
75638 Michele 525.00 221 1
78634 Jacob   625.00 221 0
28746 .          .   597 2
58743 Zina    250.00 435 0
45378 Amy     250.00 435 0
87463 Angela  525.00 221 1
46732 Trevor  450.00 597 0
23867 Michael 450.00 597 0
   ;
RUN;
```

List style input

Internal data

Reading internal raw data

WORK.students data set

	ID	Name	AmtPaid	Course	New
1	78374	Adam	350	597	1
2	75638	Michele	525	221	1
3	78634	Jacob	625	221	0
4	28746		.	597	2
5	58743	Zina	250	435	0
6	45378	Amy	250	435	0
7	87463	Angela	525	221	1
8	46732	Trevor	450	597	0
9	23867	Michael	450	597	0

Reading internal raw data

- Always check the SAS log!

```
13   * Input student enrollment data;
14   DATA students;
15      INPUT ID $ Name $ AmtPaid Course $ New;
16      DATALINES;

NOTE: The data set WORK.STUDENTS has 9 observations and
5 variables.
NOTE: DATA statement used (Total process time):
      real time           0.06 seconds
      cpu time            0.01 seconds

26      ;
27   RUN;
```

Reading external raw data

- INFILE statement
  - Tells SAS where to find a raw data file
- General form (Windows):

```
INFILE 'c:\path\filename' options;
```

Reading external raw data

```
* Input student enrollment data;
DATA students;
   INFILE 'c:\MyRawData\stu.dat';
   INPUT ID $ Name $ AmtPaid Course $ New;
RUN;
```

INFILE statement

List style input

Reading external raw data

- Always check the SAS log!

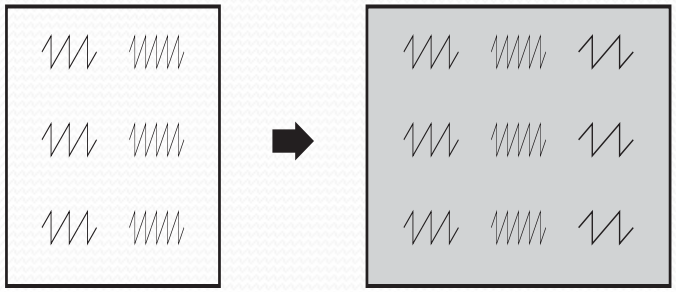
```
30    * Input student enrollment data;
31    DATA students;
32       INFILE 'c:\MyRawData\stu.dat';
33       INPUT ID $ Name $ AmtPaid Course $ New;
34    RUN;

NOTE: 9 records were read from the infile 'c:\MyRawData\stu.dat'.
      The minimum record length was 26.
      The maximum record length was 26.
NOTE: The data set WORK.STUDENTS has 9 observations and 5 variables.
NOTE: DATA statement used (Total process time):
      real time           0.01 seconds
      cpu time            0.01 seconds
```

Reading a single SAS data set

- SET statement
- General form:

```
DATA new-data-set;
    SET old-data-set;
```



- Example:

```
DATA class;
    SET class;
```

- If data set names are same, then new replaces old

Assignment statements

- Set one thing equal to another
- General form:

```
variable-name = expression;
```

Example	Type of expression
`x = 5;`	numeric constant
`x = 'five';`	character constant
`x = y;`	a variable
`x = y + 1;`	addition
`x = y * z;`	multiplication

SAS functions

- Functions perform calculation or transformation
- SAS has hundreds of functions
- Types of functions
  - Character
  - Date and time
  - Descriptive statistics
  - Mathematical
  - Probability
  - Random number
  - State and ZIP code

SAS functions

- Functions are often used in assignment statements
- General form:

```
var = function-name(argument, argument,…);
```

- Example:

```
PhoneNum = '(714) 637-9398';
AreaCode = SUBSTR(PhoneNum, 2, 3);
```

Value of AreaCode is ‘714’

SAS functions

- Examples:

```
Test1 = 80;
Test2 = .;
Test3 = 100;
AverageScore1 = Test1 + Test2 + Test3 / 3;
```

Value of AverageScore1 is missing

```
AverageScore2 = MEAN(Test1, Test2, Test3);
```

Value of AverageScore2 is 90

Assignment statements

- Example:

```
* Student enrollment data;
DATA students;
    SET students;
    Quarter = 'Fall';
    FirstInitial = SUBSTR(Name, 1,1);
RUN;
```

	ID	Name	AmtPaid	Course	New	Quarter	FirstInitial
1	78374	Adam	350	597	1	Fall	A
2	75638	Michele	525	221	1	Fall	M
3	78634	Jacob	625	221	0	Fall	J
4	28746		.	597	2	Fall	
5	58743	Zina	250	435	0	Fall	Z
6	45378	Amy	250	435	0	Fall	A
7	87463	Angela	525	221	1	Fall	A
8	46732	Trevor	450	597	0	Fall	T
9	23867	Michael	450	597	0	Fall	M

Subsetting IF

- Special form of IF statement
- General form:

```
IF condition;
```

- Example:

```
IF Age >= 21;
```

SAS will keep only observations with ages 21 or over.

Conditional logic: IF-THEN

- IF-THEN statements
- General form:

```
IF condition THEN action;
IF condition AND condition THEN action;
IF condition OR condition THEN action;
```

- Examples:

```
IF State = 'AZ' THEN Region = 'West';
IF State = 'CA' AND County = 'Yolo'
   THEN Area = 'Central';
```

Conditional logic: IF-THEN/ELSE

- IF-THEN/ELSE statements
- General form:

```
IF condition THEN action;
  ELSE IF condition THEN action;
  ELSE action;
```

Conditional logic: IF-THEN/ELSE

- IF-THEN/ELSE statements
- Bad example:

```
IF State = 'CA' THEN Region = 'West';
IF State = 'MA' THEN Region = 'East';
IF State = 'TX' THEN Region = 'Gulf';
```

- Good example:

```
IF state = 'CA' THEN Region = 'West';
  ELSE IF State = 'MA' THEN Region = 'East';
  ELSE IF State = 'TX' THEN Region = 'Gulf';
  ELSE Region = 'Unknown';
```

Conditional logic

- Example:

```
* Student enrollment data;
DATA students;
   SET students;
   IF Name NE '';
   IF New = 1 THEN NewStudent = 'yes';
      ELSE IF New = 0 THEN NewStudent = 'no';
      ELSE NewStudent = '?';
RUN;
```

	ID	Name	AmtPaid	Course	New	Quarter	FirstInitial	NewStudent
1	78374	Adam	350	597	1	Fall	A	yes
2	75638	Michele	525	221	1	Fall	M	yes
3	78634	Jacob	625	221	0	Fall	J	no
4	58743	Zina	250	435	0	Fall	Z	no
5	45378	Amy	250	435	0	Fall	A	no
6	87463	Angela	525	221	1	Fall	A	yes
7	46732	Trevor	450	597	0	Fall	T	no
8	23867	Michael	450	597	0	Fall	M	no

DO groups

- A single IF-THEN statement can have only one action
- For multiple actions, use DO/END statements
- General form:

```
IF condition THEN DO;
  action;
  action;
END;
```

- Example:

```
IF State = 'HI' THEN DO;
  Region = 'West';
  Capital = 'Honolulu';
END;
```

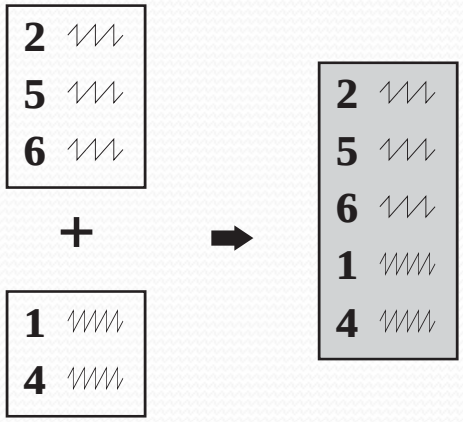
DO groups

- Example:

```
* Student enrollment data;
DATA students;
    SET students;
    IF Course = '221' THEN DO;
        CourseName = 'Introduction to SAS';
        Instructor = 'Susan Slaughter';
    END;
RUN;
```

	ID	Name	AmtPaid	Course	New	Quarter	FirstInitial	NewStudent	CourseName	Instructor
1	78374	Adam	350	597	1	Fall	A	yes		
2	75638	Michele	525	221	1	Fall	M	yes	Introduction to SAS	Susan Slaughter
3	78634	Jacob	625	221	0	Fall	J	no	Introduction to SAS	Susan Slaughter
4	58743	Zina	250	435	0	Fall	Z	no		
5	45378	Amy	250	435	0	Fall	A	no		
6	87463	Angela	525	221	1	Fall	A	yes	Introduction to SAS	Susan Slaughter
7	46732	Trevor	450	597	0	Fall	T	no		
8	23867	Michael	450	597	0	Fall	M	no		

Stacking SAS data sets



- SET statement
- Concatenates = appends
- General form:

```
DATA new-data-set;
    SET old-data-set-1 … old-data-set-n;
```

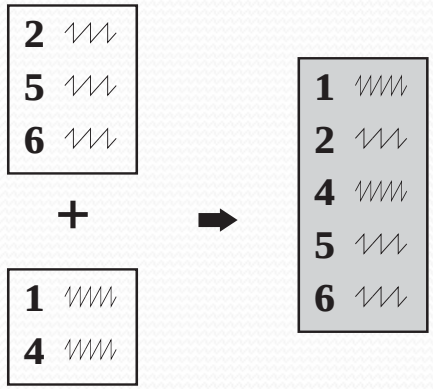
- Example:

```
DATA new_stacked;
    SET class1 class2 class3;
```

- Order of observations in new data set depends on order in SET statement

Interleaving SAS data sets

- SET statement
- Interleave observations
- General form:



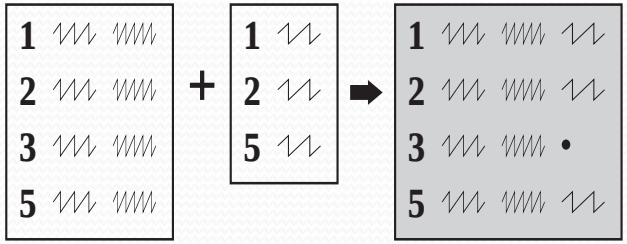

```
DATA new-data-set;
    SET old-data-set-1 … old-data-set-n;
    BY variable-list;
```

- Example:

```
DATA new_stacked;
    SET class1 class2 class3;
    BY ID;
```

- Data sets must be sorted by BY variables
  - Use PROC SORT

Matching SAS data sets 1-to-1



- MERGE statement
- General form:

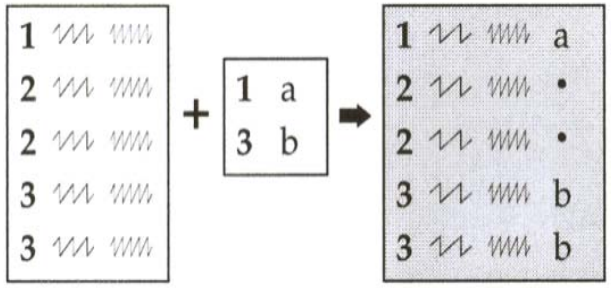
```
DATA new-data-set;
    MERGE old-data-set-1 old-data-set-2;
    BY variable-list;
```

- Example:

```
DATA new_merged;
    MERGE class demog;
    BY ID;
```

- Data sets must be sorted by BY variables
  - If needed, use PROC SORT

Matching SAS data sets many-to-1



- MERGE statement
- General form:

```
DATA new-data-set;
    MERGE old-data-set-1 old-data-set-2;
    BY variable-list;
```

- Example:

```
DATA new_merged;
    MERGE class instructors;
    BY ID;
```

- Data sets must be sorted by BY variables
  - If needed, use PROC SORT

Pop quiz

1) Only one statement is required in DATA steps. What is it?

   DATA statement

2) Complete this sentence: "Data steps execute _____and _____."

   DATA steps execute line-by-line and observation-by-observation.

3) What statement tells SAS where to find an external raw data file?

   INFILE statement

Pop quiz

4) What statement do you use to read a single SAS data set?

   SET statement

5) What do assignment statements do?

   Set a variable equal to something.

6) Write a statement that sets a variable named Score equal to the number 10.

```
Score = 10;
```

Pop quiz

7) Write a statement that sets a variable named State equal to Alaska.

```
State = 'Alaska';
```

8) What will this statement do? `IF Name = 'Joe';`

   Keep only observations where the value of the variable Name equals Joe.

9) What statement do you use to end a DO group?

   END statement

Pop quiz

10) What statement do you use to match observations from one data set with observations from another data set?

MERGE statement

Other presentations

- Next up in this room
  - Introduction to SAS Procedures: SAS Basics III
- Hands-on Workshop Friday 9:00-10:30
  - SAS Studio: A New Way to Program in SAS, Lora Delwiche

Thank you!

I hope you can stay for the next presentation.

Susan Slaughter
Avocet Solutions

Can download slides from
www.avocetsolutions.com