PSTAT 130

SAS BASE PROGRAMMING

- Lecture 6 -

Dr. Julie Swenson

Objectives

- Variable Assignment/Creation
- SAS Functions
- Date Constant
- Conditional Processing
- The Data Step
  - Compile and Execute

Create New Variables

- Use variable assignment statements in the DATA step to create new variables

- An assignment statement
  - Evaluates an expression
  - Assigns the resulting value to a variable

- General form of an assignment statement

```
DATA output-SAS-data-set;
     SET input-SAS-data-set;
     variable = expression;
RUN;
```

SAS Expressions

- An expression contains operands and operators that form a set of instructions that produce a value

- Operands are
  - Variable names
  - Constants

- Operators are
  - Arithmetic symbols (+,-,/,*, etc.)
  - SAS functions

Arithmetic Operators

- Selected operators for basic arithmetic calculations in an assignment statement

Operator	Action	Example	Priority
+	Addition	Sum=x+y;	III
-	Subtraction	Diff=x-y;	III
*	Multiplication	Product=x*y;	II
/	Division	Divide=x/y;	II
**	Exponent	Raise=x**y;	I
-	Negative	Negative = -x;	I

Variable Assignment

- Examples of creating new variables using arithmetic operators
  - `TotalComp = Salary + Bonus`
  - `NetPay = GrossPay - Tax`
  - `NewPay = Salary * (1 + Raise)`
  - `Percent = Score/Maximum`

SAS Functions

- A SAS function is a routine that returns a value that is determined from specified arguments

- General form of a SAS function:

```
Function-name(argument1, argument2, …);
```

- Example:

```
Total=sum(Salary, Bonus);
```

SAS Functions

- SAS functions
  - perform arithmetic operations
  - compute sample statistics (for example: mean, median, and standard deviation)
  - manipulate SAS dates and process character values
  - perform many other tasks

- Sample statistics functions ignore missing values.

Numeric Functions

- **MIN(x,y,z)**- Returns the smallest value
- **MAX(x,y,z)**- Returns the largest value
- **MEDIAN(x,y,z)**- Calculates the median
- **MEAN(x,y,z)** – Calculates the average value
- **STD**(x,y,z) – Returns the standard deviation of the values
- **ABS(x)** – Returns the absolute value of x
- **FACT(x)** – Calculates the factorial, x!
- **COS(x)** – Returns the cosine of x
- **TAN(x)** – Returns the tangent of x

Character Functions

- **LOWCASE(x)** – Converts all letters in **x** to lower case
- **UPCASE(x)** – Converts all letters in **x** to upper case
- **TRIM(x)** – Removes all trailing blanks
- **SUBSTR(string,position,length)** – Returns a substring of given length within the characters in the string, starting at the given position:
  - `SUBSTR('Statistics',1,4) = 'Stat'`
- **REVERSE(string)** – reverse the order of letters
  - `REVERSE('Statistics') = 'scitsitatS'`

Date Functions

- **TODAY()** obtains the SAS date value from the system clock.
- **MDY(month,day,year)** uses numeric month, day, and year values to return the corresponding SAS date value.
- **YEAR(SAS-date)** extracts the year from a SAS date and returns a four-digit value for year.
- **QTR(SAS-date)** extracts the quarter from a SAS date and returns a number from 1 to 4.
- **MONTH(SAS-date)** extracts the month from a SAS date and returns a number from 1 to 12.
- **WEEKDAY(SAS-date)** extracts the day of the week from a SAS date and returns a number from 1 to 7, where 1 represents Sunday, and so on.

Class Exercise 1

- Using the **admit** data set found in the data1 folder
  - Copy the data set into the file **work.admit**
    - Create a new variable **Height_ft**
      - Assign the correct calculation using the **height** variable.
  - Create a report using the **work.admit** data set
    - Only display **Name Age Height Height_ft**
    - Has an appropriate title
    - Suppress observation numbers

SAS Date Constants

- Use a Date Constant to return a SAS date value for a specific date.
  - Example:

```
evaldate = '1JAN2020'd;
```

  - Sets the value of **evaldate** to 21915, which is the SAS date value (number of days since 1/1/1960) corresponding to January 1, 2020. The text portion can be in the form of 'ddmmmyyyy' or 'ddmmmyy'

Class Exercise 2

- Write a program that displays the SAS date value for June 30th, 2020.
  - What is the date?

Class Exercise 2 - continued

- What are two different ways of obtaining a SAS date value?

- Use both methods to obtain the date value for July 17th, 2020.

DateTime Values and DatePart

- A SAS datetime value is the number of seconds between midnight, January 1st, 1960, and a specific date and time.
  - Example: 12/01/2009 9:15am is stored as 1,291,281,300 seconds since 01/01/1960 12:00am.

- The DATEPART function will return just the date portion, 14945, which is the number of days since 01/01/1960
  - Example:

```
Birthdate = datepart(Birthdatetime);
```

Conditional Processing

- Execute statements conditionally using IF-THEN logic
- Select rows to include in a SAS data set

Conditional Execution

- General form of IF-THEN and ELSE statements

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

  - Expression contains operands and operators that form a set of instructions that produce a value
  - Examples:

```
if Hours < 40 then Status = 'Part Time';
```

```
if JobCode = 'PILOT' then Bonus = Salary * 0.1;
else Bonus = 0;
```

  - Only one executable statement is allowed per IF-THEN or ELSE statement

Conditional Execution

- To allow more than one statement, use DO and END statement.

```
IF expression THEN DO;
      executable statements
END;
ELSE DO;
      executable statements
END;
```

Operators and Special Operators

- Comparison operators work in IF statements
  - GT (>)
  - LT (<)
  - EQ (=)
  - Etc.
- Logical operators work in IF statements
  - AND (&)
  - OR (|)
- The special operators we've discussed do not work in IF statements

Index Function

- For example, the **`contains`** operator does not work in an IF statement
- The **`index()`** function is a viable alternative: it finds the starting location of an excerpt in a source
- **`index()`** reads in two arguments
  - Source
  - Excerpt

Index Function

- General form:
  - index(source, excerpt)
- Example:
  - a = ‘University of California’;
  - b = index(a, ‘Cali’);
    - Or equivalently: b = index(‘University of California, ‘Cali’);
  - Then b would be equal to 15

Conditional Execution

- In a DATA step, you can subset the rows (observations) in a SAS data set with a
  - WHERE statement
  - subsetting IF statement
  - IF-THEN DELETE statement
- The WHERE statement in a DATA step is the same as the WHERE statement you saw in PROC step

Select and Delete Rows

- Use an IF statement to include only those rows that meet the criteria

```
IF expression;
```

- Use an IF-THEN DELETE statement to exclude rows that meet the criteria

```
IF expression THEN DELETE;
```

Where vs. If

- If the variable already exists in the input data set, you can use a WHERE or IF statement

```
data work.empdata;
   set data1.empdata;
   where jobcode = 'PILOT';
run;
```

- If you are evaluating a calculated variable, use IF:

```
data work.empdata;
   set data1.empdata;
   Bonus = Salary * 0.1;
   if Bonus > 5000;
run;
```

Class Exercise 3

- Do the following once using WHERE statements, then again using IF statements
  - Create a data set called **`work.admit1`** from **`data1.admit`**
    - Select patients whose age is less than 50. Does this step execute properly?
  - Create a report using **`work.admit1`**
    - Display patients whose age is at least 30. Does this step execute properly?

Where or Subsetting If?

Step and Usage	WHERE	IF
PROC Step	Yes	No
DATA Step - Source of Variable		
INPUT statement (i.e., existing variable)	Yes	Yes
assignment statement (i.e., created variable)	No	Yes
SET/MERGE (multiple data sets)		
Variable in ALL data sets	Yes	Yes
Variable not in ALL data sets	No	Yes

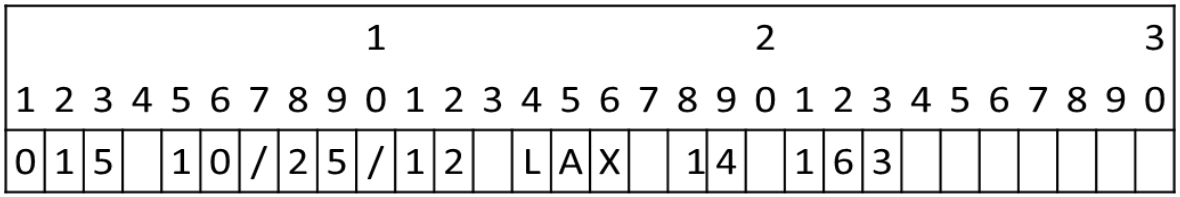
Looking Behind the Scenes

- The DATA step is processed in two phases
  - Compile
  - Execution

```
data work.lax;
      infile 'raw-data-file';
      input  Flight $ 1-3
             Date $ 4-11
             Dest $ 12-14
             FirstClass 15-17
             Economy 18-20;
run;
```

Looking Behind the Scenes

- At compile time, SAS creates
  - An input buffer to hold the current raw data file record that is being processed



  - A program data vector (PDV) to hold the current SAS observation

Flight $3	Date $8	Dest $3	FirstClass N8	Economy N8
015	10/25/12	LAX	14	163

  - The descriptor portion of the output data set

Compile the Data Step

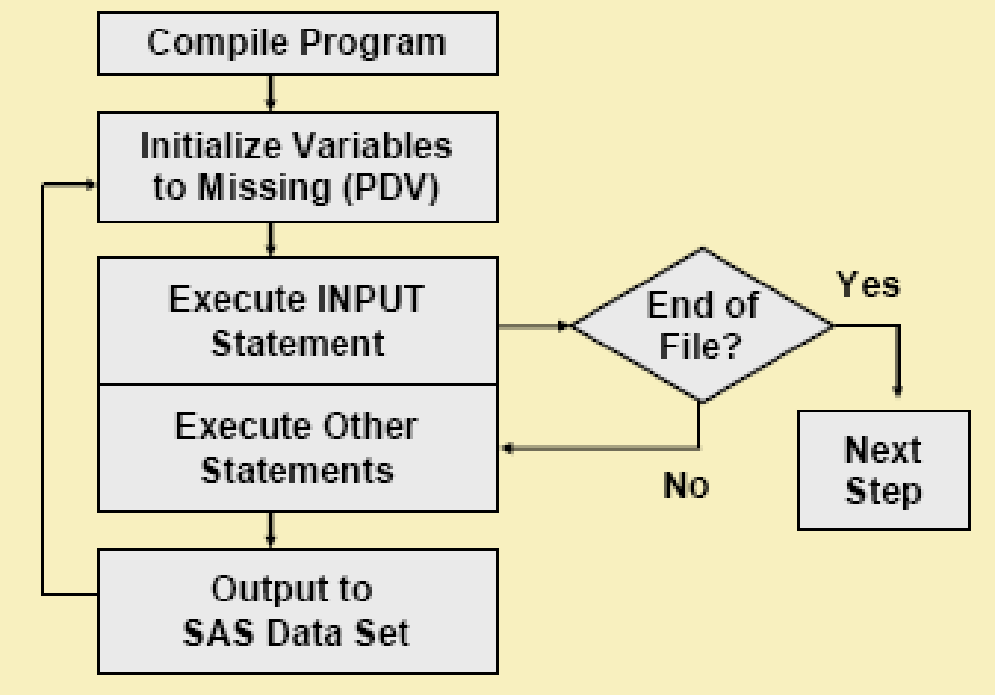
SAS creates the data set placeholder

SAS opens the data source

```
data work.lax;
     infile 'raw-data-file';
     input Flight $ 1-3
           Date $ 4-11
           Dest $ 12-14
           FirstClass 15-17
           Economy 18-20;
run;
```

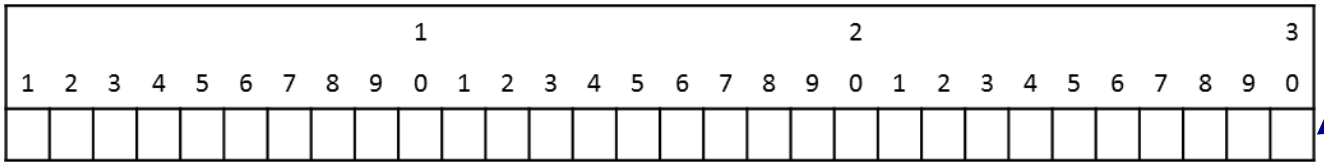
SAS prepares the input buffer

Data Step Execution: Summary



Data Step Execution: Details

- At compile time, SAS created an empty Input Buffer, and an empty Program Data Vector, to store the incoming data

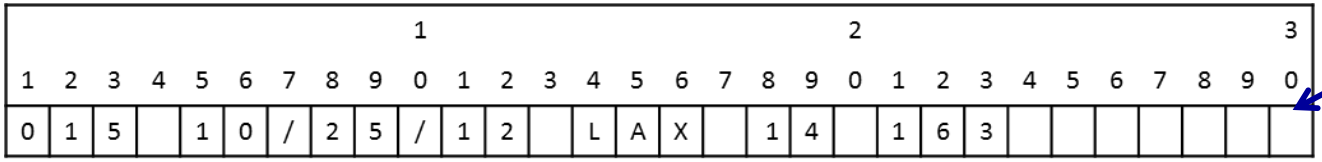


Empty Input Buffer

Flight $3	Date $8	Dest $3	FirstClass N 8	Economy N8
			.	.

Empty Program Data Vector

- At execution, it loads each line of data into the input buffer, parses it into variables, and outputs those values to the SAS data set



Input Buffer with a line of data

Flight $3	Date $8	Dest $3	FirstClass N 8	Economy N8
15	10/25/12	LAX	14	163

Record output to SAS data set

What Are Data Errors?

- SAS detects data errors when
  - the INPUT statement encounters invalid data in a field
  - illegal arguments are used in functions
  - impossible mathematical operations are requested

Examining Data Errors

- When SAS encounters a data error
  - a note that describes the error is printed in the SAS log
  - the input record being read is displayed in the SAS log (contents of the input buffer)
  - the values in the SAS observation being created are displayed in the SAS log (contents of the PDV)
  - a missing value is assigned to the appropriate SAS variable
  - execution continues.

Programming Errors: Tips

- In SAS 9.4: use the Enhanced Editor as it color-codes keywords and color-codes errors in red.
- In SAS 9.4 and SAS Studio: check that keywords are color-coded correctly.
- Write your program in small parts and test each part.
- In SAS 9.4: clear the Log and Output windows before running your program.
- Review the Log, looking for red text.
- Confirm the number of records and variables in each data set using the Log.
- Keep all variables in your interim data sets.
- Inspect the data sets you create in the Output Data window, or by using PROC Print.

Class Exercise 4

- Create a new data set called **`work.talent`** from the **`data1.talent`** data set
  - Create two new variables `CurrentRate` and `Stage`
  - For individuals who do stage acting
    - Set their `CurrentRate` as 500 more than their Rate
    - Set the `Stage` variable to 1 (true)
  - For all other individuals
    - Set their `CurrentRate` to be the same as their Rate
    - Set the `Stage` variable to 0 (false)

Class Exercise 4 - continued

- Create a report using the **`work.talent`** data set
  - Suppress observation numbers
  - Display the variables: last name, first name, rate, current rate, and stage
  - Use the following column headings
    - Last Name
    - First Name
    - Previous Rate
    - Current Rate
    - Stage Work
  - Format CurrentRate with the same format as Previous Rate
  - Format the Stage variable as Yes (for 1) and No (for 0)