PSTAT 130

SAS BASE PROGRAMMING

- Lecture 5 -

Dr. Julie Swenson

Objectives

- Keep/Drop vs. Keep=/Drop=
- Read Raw Data
  - Input Statements
- The IMPORT Procedure
- The DATASETS Procedure

Manage a Data Set

- When we read in data, are we controlling which variables are output?

- Think about reading in data from an existing SAS data set. What does the following statement produce?

```
data work.temp_allgoals;
    set data1.allgoals;
run;
```

Keep and Drop Examples

- Use DROP if you want to keep most of the variables

```
data work.empdata1;
  set data1.empdata;
  drop EmpID Hire;
run;
```

- Use KEEP if you only want to keep a few variables

```
data work.empdata2;
  set data1.empdata;
  keep Firstname Lastname;
run;
```

Keep= or Drop=

- Use the KEEP= or DROP= option in a SAS statement to eliminate variables from the input statement. The eliminated variables cannot be used in expressions.

```
DATA libref.new-data-set;
   SET SAS-data-set(KEEP=variables)
or
   SET SAS-data-set(DROP=variables)
run;
```

Read Data Fields

- The INPUT statement tells SAS the name and type of each variable in the data set, and how to read the data

- Simple form of the INPUT statement

```
INPUT variable <$> <options>;
```

- General form of the INPUT statement

```
INPUT variable <$> start-column <-- end-column> <.decimals> <@|@@>;

INPUT <pointer-control> variable <$> <:|&|~>  <informat.> <@|@@>;
```

Input Statement Example

```
data work.students;
input firstname $ gender $ age;
datalines;
John Male 19
Wendy Female 22
;
run;
```

Types of Raw Data Input

- List input – each data value is separated by a space (the "delimiter")

```
John Male 22
Wendy Female 19
```

- Column input – each data value is in a fixed location

```
John          Male          22
Wendy         Female        19
```

- Formatted input – uses SAS formats (called informats)

```
John          Male      4/12/91
Elizabeth     Female    8/24/90
```

Format vs. Informat

- A Format controls the way SAS outputs (displays) data.
- An Informat controls the way SAS reads in data.

(Simple) List Input

- If your data values are separated by a single space, use list input

- Example

```
data work.students;
input Name $ Team $ Age;
datalines;
David Male 19
Amelia Female 23
Ravi Male 17
Ashley Female 20
Jim Male 26
;
run;
```

Default Attributes of List Input

- All data values must be separated by a single space
- All variables must be in standard format
  - Character and numeric values cannot contain spaces
  - Character values cannot be longer than 8 characters
  - Numeric values cannot contain commas or dollar signs
  - Dates will be read as characters rather than date values

Column Input

- If your data values are in the fixed columns, and consist of “standard” character and numeric values, use column input

- Example

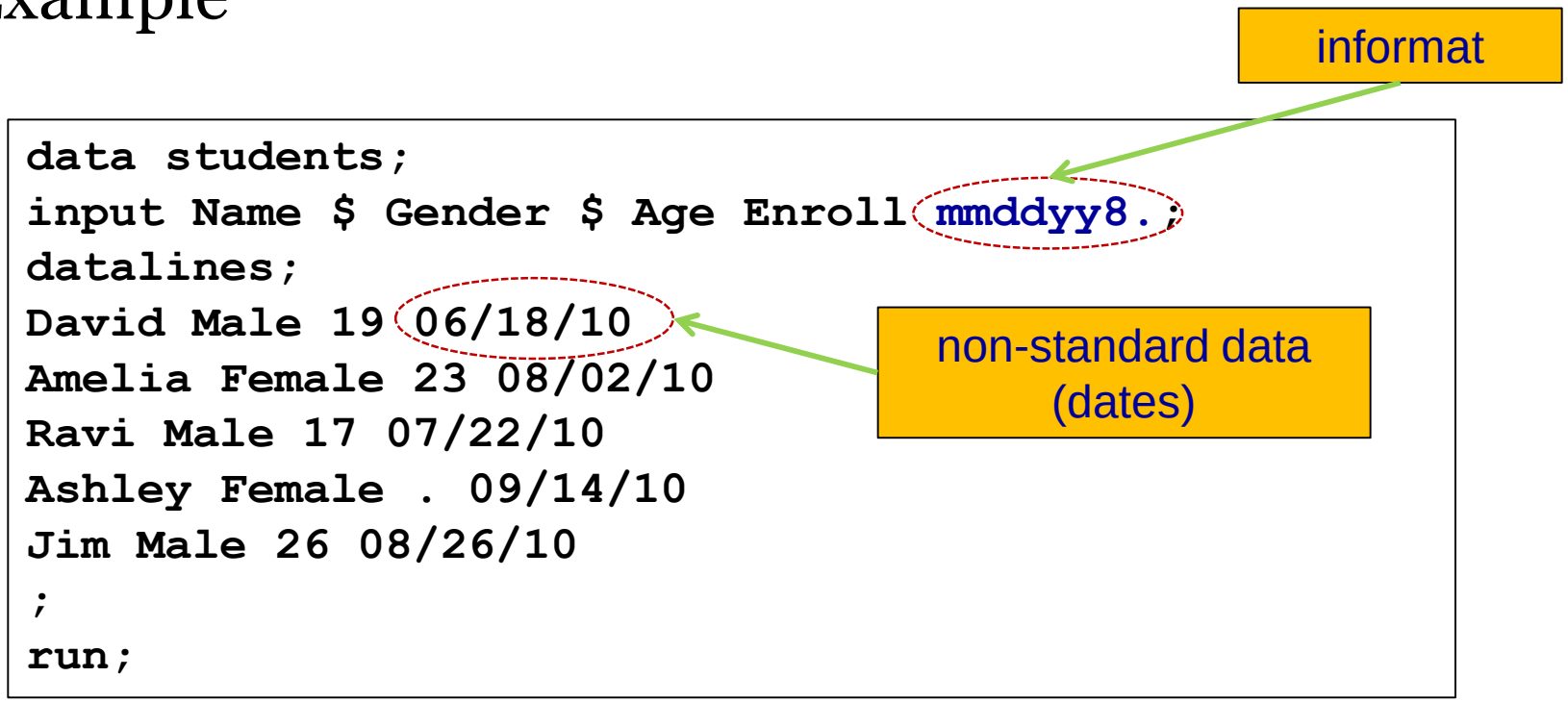
```
data work.students;
input Name $ 1-6 Gender $ 9-14 Age 18-20;
datalines;
David   Male       19
Amelia  Female     23
Ravi    Male       17
Ashley  Female     20
Jim     Male       26
;
run;
```

Default Attributes of Column Input

- The data values must occupy the same columns within each observation
  - This is called “fixed” or “aligned”
- Character variables can
  - Be longer than 8 characters
  - Contain spaces
- You can skip some data fields, if desired
- The data must be in “standard” format
  - Numbers may not contain commas or dollar signs
  - Dates will be read as character, instead of numeric, variables

Formatted Input

- If your data contains “non-standard” values use formatted input (with “informats”)
- Example



Default Attributes of Formatted Input

- Data can be in “non-standard” format
  - Numbers can contain commas and dollar signs
  - Dates can be read into numeric variables
- Data can be listed or in fixed columns

Pointer Control

- With formatted input, you can "point" at the first column of each variable, instead of using start and end columns

- Note: An informat specifies
  - the width of the input field
  - how to read the data values that are stored in the field

Pointer Control

- Absolute pointer control
  - You can move the pointer to a specific column, using the @ symbol
  - @n moves pointer to column n

```
input <@n1> var1 <$>fmt1. <@n2> var2 <$>fmt2. …;
```

Pointer Control

- Relative pointer control
  - You can also move the pointer forward a specific number of columns forward, using the + symbol
  - +n moves the pointer forward n columns

```
input <+n1> var1 <$>fmt1. <+n2> var2 <$>fmt2. …;
```

Named Input

- If your data contains values that are assigned to variable names, use named input

- Example

```
data alumni;
input Fname=$ Lname=$ Age=;
datalines;
fname=Jack lname=Johnson age=45
age=44 fname=Jason lname=Lezak
lname=Douglas fname=Michael age=75
;
run;
```

Class Exercise 1

- The data set below contains the 2010 population estimates (in millions) of several US states. What type of input statement should be used to read in this data?

```
Calif 36.9
Texas 24.8
NewYork 19.5
Florida 18.5
Illinois 12.9
```

- Write the input statement.

Class Exercise 1 - continued

- The data set below contains the 2010 population estimates (in millions) of several US states. What type of input statement should be used to read in this data?

```
California   36.9
Texas        24.8
New York     19.5
Florida      18.5
Illinois     12.9
```

- Write the input statement.

Class Exercise 1 - continued

- The data set below contains the 2010 population estimates of several US states. What type of input statement should be used to read in this data?

```
California   36,961,664
Texas        24,782,302
New York     19,541,453
Florida      18,537,969
Illinois     12,910,409
```

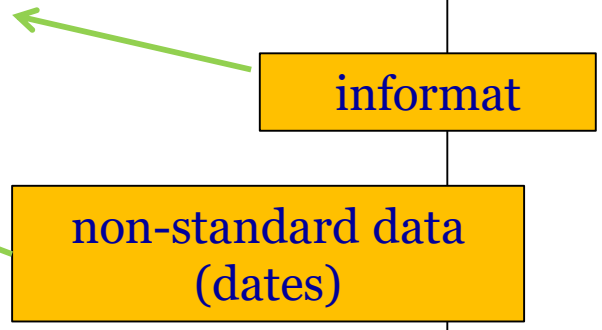
- Write the input statement.

Input Statements

- Can have mixed input styles
  - Caveat: when the named input style appears in an input statement, all following variables must be in the same form

- The other three input styles can be mixed freely:

```
data students;
input Name $ Gender $ Age Enroll mmddyy8.;
datalines;
David Male 19 06/18/20
Amelia Female 23 08/02/19
Jim Male 26 08/26/19
;
run;
```



Column Pointer Controls

- Column pointer control
  - Absolute: @n
  - Relative: +n
    - Generally moves the pointer forward
    - Can move pointer back, but still needs + symbol
      - i.e. + (-1)

Column Pointer Controls

- Example:

```
data people;
input name $12. +(-1) age;
datalines;
john smith 25
jane doe   29
;
run;

proc print data=people;
run;
```

Line Pointer Controls

- Line pointer control
  - Absolute: #n
  - Relative: /

Line Pointer Controls

- Example:

```
data ucsb_alumni;
input name $ 1-15 #2 age 13-14 #3;
datalines;
Leroy Chiao
male          59
astronaut
Benjamin Bratt
male          56
actor
;
run;

proc print data=ucsb_alumni;
run;
```

Read Excel Spreadsheets

- Create a SAS data set from an Excel spreadsheet using the Import Wizard (SAS 9.4)

- Create a SAS data set from an Excel spreadsheet using PROC IMPORT

The IMPORT Procedure

- General form of the IMPORT procedure

```
PROC IMPORT DATAFILE='external-file-name'
            OUT=SAS-data-set
            DBMS=file-type;
   GETNAMES=YES;
RUN;
```

Excel Import Example

```
PROC IMPORT DATAFILE='/home/user/DallasLA.xls'
            OUT=WORK.tdfwlax
            DBMS=XLS REPLACE;
     SHEET='DFWLAX';
     GETNAMES=YES;
RUN;
```

- Imports the file
  - `'/home/user/DallasLA.xls'`
- Outputs the data set to `work.tdfwlax`
- Specifies the type of file to import as XLS (`dbms`)
- Overwrites an existing SAS data set (`replace`)
- Specifies which sheet SAS should import (default is 1st sheet)
- Specifies to SAS to use the first row of data as variable names

Assign Variable Attributes

- SAS allows you to
  - Assign permanent attributes to SAS variables
  - Change or override permanent variable attributes

Default Variable Attributes

- When a variable is created in a DATA step, the
  - Name, type, and length of the variable are automatically assigned
  - Remaining attributes such as label and format are not automatically assigned
- When the variable is later used in a PROC step, the output uses
  - the variable name
  - a system determined format

Specify Variable Attributes

- Use LABEL and FORMAT statements in the
  - DATA step to permanently assign the attributes (stored in the descriptor portion of the data set)
  - PROC step to temporarily assign the attributes (for the duration of the step only)

Assignments in DATA vs. PROC Steps

```
DATA work.bonus;
    set data1.empdata;
    Bonus = Salary * .1;
    label Bonus = 'Annual Bonus';
    format Bonus Dollar12.2;
RUN;

PROC PRINT data=work.bonus label;
    label Bonus = 'Incentive Bonus';
    format Bonus Dollar12.;
run;
```

Permanent Attributes

Temporary Attributes

The DATASETS Procedure

- You can use the DATASETS procedure to modify a variable's
  - name
  - label
  - format
  - informat

The DATASETS Procedure

- General form of PROC DATASETS, used for changing variable attributes:

```
PROC datasets LIBRARY=libref;
      MODIFY SAS-data-set;
      RENAME old-name-1=new-name-1
             <. . . old-name-n=new-name-n>;
      LABEL variable-1='label-1'
            <. . . variable-n='label-n'>;
      FORMAT variable-list-1 format-1
             <. . . variable-list-n format-n>;
      INFORMAT variable-list-1 informat-1
               <. . . variable-list-n informat-n>;
RUN;
```

Class Exercise 2

- Use the data set `insure` in the folder data1
  - Create a data set, `work.insure1`, that only reads in the variables `Name Policy Company PctInsured` and `BalanceDue`
  - Create a data set, `work.insure2`, that only outputs the variables `ID Name Company PctInsured Total`
  - Create a data set, `work.insure3`, that only outputs the variables `Name` and `BalanceDue`
  - Create a data set, `work.insure4`, that only reads in the variables `ID` and `BalanceDue`

Class Exercise 3

- Create a new data set called `work.insure` from the `insure` data set the data1 folder
  - Assign the following permanent labels
    - Full Name
    - Policy Number
    - Percent Insured
    - Total Amount
    - Balance Due
  - Assign the following permanent formats
    - Dollar9.2 to Total
  - Check the descriptor, and output the report

Class Exercise 3 - continued

- Create a new report that *temporarily* changes the labels to
  - Given Name
  - Insured Percentage
- Now permanently change the labels to
  - Given Name
  - Insured Percentage
- Create a report using the updated data set