STAT 541

Chapter 13: Formatting Data

Creating a Format with Overlapping Values

VALUE *format-name* (MULTILABEL);

- allows the assignment of multiple labels or external values to internal values.
- Example of VALUE statement assigning multiple labels to a single internal value:

```
value one (multilabel)
  1='ONE in English'
  1='UNO in Spanish';
```

(Multiple labels can also be assigned to a single range of internal values.)

Creating a Format with Overlapping Values (continued)

- Example of assigning multiple labels to overlapping ranges of internal values:

```
value age (multilabel)
15-29='below 30'
15-19='15 to 19'
20-29='20 to 29';
```

Creating a Format with Overlapping Values (continued)

- Multilabel formatting allows an observation to be included in multiple rows or categories.
- To use multilabel formats, specify the MLF option in class variables in procedures that support it (e.g., PROC TABULATE, PROC MEANS, PROC SUMMARY).

Creating a Format with Overlapping Values (continued)

```
proc format;
value age (multilabel)
     15-29='below 30'
     15-19='15 to 19'
     20-29='20 to 29';
data age;
input age books @@;
cards;
15 13 20 13 25 22
;
proc means sum maxdec=0;
class age/mlf;
format age age.;
var books;
run;
```

```
The MEANS Procedure

   Analysis Variable : counter

                    N
age              Obs              Sum
-------------------------------------
'15 to 19'         1               13

'20 to 29'         2               35

'below 30'         3               48
-------------------------------------
```

Creating Custom Formats Using the Picture Statement

- PICTURE statements can be used to create a template for printing numbers.

PICTURE *format-name*

value-range='picture';

- *Value-range* is the individual value or range of values to be labeled
- Picture specifies a template for formatting values of numeric variables. The template is a sequence of at most 40 characters enclosed in quotation marks.

Creating Custom Formats Using the Picture Statement (continued)

- There are three types of characters in pictures:

1. Digit selectors
2. Message characters
3. Directives

Digit Selectors in the Picture Statement (continued)

- Digit selectors:
  - are numeric characters--0 through 9.
  - define positions for numeric values.
- Nonzero digit selectors add zeros to the formatted value as needed.
- Zero digit selectors do not add any zeros to the formatted value.

Digit Selectors in the Picture Statement (continued)

- Example for Digit Selectors

Picture Definition	Data Values	Formatted Values
picture month 1-12='99';	01	01
	1	01
	12	12
picture month 1-12='00';	01	1
	1	1
	12	12

Message Characters in the Picture Statement (continued)

- Message characters:
  - are nonnumeric characters that print as specified in the picture.
  - are inserted into the picture after the numeric digits are formatted.
  - must come after digit selectors in picture definitions.

Message Characters in the Picture Statement (continued)

- Example for Message Characters

Picture Definition	Data Values	Formatted Values
Picture millA low-high = '009.9M' (mult=.00001);	1450000	1.4M
Picture millB low-high = '009.9M' (prefix='$' mult=.00001);	1450000	$1.4M
Picture millC (round) low-high = '009.9M' (prefix='$' mult=.00001);	1450000	$1.5M

- M is the message character in the examples above.
- The multiplier (MULT) is a number that the value is to be multiplied by before formatting.
- The PREFIX option can be used to append text in front of digits.
- The ROUND option rounds the value to the nearest integer before formatting. Without the ROUND option, the format multiplies the value by the multiplier, truncates the decimal portion (if any), and prints the result according to the picture definition. With the ROUND option, the format multiplies the value by the multiplier, rounds that result to the nearest integer, and then formats the value according to the picture definition. A value of .5 rounds to the next highest integer.

Directives in the Picture Statement (continued)

- Directives:
  - are special characters that can be used in the picture to format date, time, or datetime values.
  - *must* specify the DATATYPE= option in the PICTURE statement. The option specifies that the picture applies to a SAS date, SAS time, or SAS datetime value. The option value is either DATE, TIME, or DATETIME.

Directives in the Picture Statement (continued)

- Example for Directives

```
proc format lib=form541;
  picture dt
   low-high = 'TIME STAMP: %A %B %d, %Y.'
              (datatype=date)
  ;
  picture tm
    low-high = '%I:%M.%S%p'
               (datatype=time);

data _null_;
  file print;
  now = today();
  tm = time();
  put  now dt40.  tm tm.;
run;
```

TIME STAMP: Wednesday January 18, 2012. 11:7.55PM

%A = full weekday name
%B = full month name
%d = day of the month with no leading zero
%Y = year with century

%I = 12-hr clock time with no leading zero
%M = minute as a decimal number 0-59 with no leading zero
%S = second as a number 0-59 with no leading zero
%p = AM or PM

dt40. displays the value of variable *now* up to 40 characters

Managing Custom Formats: Using FMTLIB with PROC FORMAT to Document Formats

- Adding the keyword FMTLIB to the PROC FORMAT statement displays a list of all the formats in the specified catalog, along with descriptions of values.
- The SELECT and EXCLUDE statements allow you to process specific formats instead of processing an entire catalog.

Managing Custom Formats: Using FMTLIB with PROC FORMAT to Document Formats (continued)

- Example:

```
libname form541 'f:\STAT 541\sas formats';
proc format lib=form541 fmtlib;
 select dt tm;
 *exclude dt;
```

Managing Custom Formats: Using FMTLIB with PROC FORMAT to Document Formats (continued)

Example of format listings from a specified catalog

FORMAT NAME: DT LENGTH: 25 NUMBER OF VALUES: 1 MIN LENGTH: 1 MAX LENGTH: 40 DEFAULT LENGTH 25 FUZZ: STD		
START	END	LABEL (VER. V7|V8 18JAN2012:23:31:24)
LOW	HIGH	TIME STAMP: %A % N P F M0

FORMAT NAME: TM LENGTH: 10 NUMBER OF VALUES: 1 MIN LENGTH: 1 MAX LENGTH: 40 DEFAULT LENGTH 10 FUZZ: STD		
START	END	LABEL (VER. V7|V8 18JAN2012:23:31:24)
LOW	HIGH	%I:%M.%S%p N P F M0

Managing Custom Formats: Using PROC CATALOG to Manage Formats

- Formats are saved as catalog entries. Therefore, PROC CATALOG can be used to manage the formats.
- PROC CATALOG can:
  1. Create a listing of catalog contents
  2. Copy a catalog or selected entries within a catalog
  3. Delete or rename entries within a catalog

Managing Custom Formats: Using PROC CATALOG to Manage Formats (continued)

- Example:

```
proc catalog catalog=form541.formats;
  copy out=work.formats;
  select dt.format;
run;
proc catalog cat=work.formats;
  contents;
run;
```

- Use the full catalog entry name of DT.FORMAT for DT in the SELECT statement.
- The format DT is copied from the form541.formats catalog to the work.formats catalog.
- The CONTENTS statement displays the contents of the work.formats catalog.

Using Custom Formats

- SAS statements in a DATA Step can permanently assign a format to a variable.
- A format can be temporarily specified for a variable in a PROC step.
- PROC DATASETS can be used to assign, change, or remove the format associated with a variable in a SAS data set.

Using Custom Formats (continued)

- Example:

```
proc datasets lib=Mylib;
   modify flights;
   format dest $dest.;
   format baggage;
quit;
```

- Mylib is the name of the SAS library that contains the data that needs to be modified.
- Flights is the name of the SAS data set to be modified.
- The format $dest is associated with variable dest.
- Since no format is associated with variable baggage, the format associated with the variable is removed.

Using a Permanent Storage Location for Formats

- When a format is permanently associated with a variable, it is important to know where the format is located and to reference it whenever the variable is being used.
- The location of the format is determined when the format is created in PROC FORMAT.

Using a Permanent Storage Location for Formats (continued)

- Formats can be stored anywhere. However, SAS must be told which format catalogs to search before the formats can be accessed.
- When a format is referenced, SAS automatically looks through the following libraries in this order:
  - Work.formats
  - Libref.formats(The library libref is recommended for formats because it is automatically searched when a format is referenced. Use LIB=Libref in the PROC FORMAT step that creates the format. Use the same libname statement with the library name Libref in the program that needs to reference the format.)

Using a Permanent Storage Location for Formats (continued)

- When other libraries or catalogs need to be searched, use the FMTSEARCH= system option to indicate where to search for formats.

OPTIONS FMTSEARCH = (*catalog-1 catalog-2… catalog-n);*

Substituting Formats to Avoid Errors

- If SAS fails to locate the format you need, it issues an error message and stops processing the step. The system behavior defaults to FMTERR.
- To prevent this, use the NOFMTERR option where SAS substitutes a format (w. or $w.) for the missing format and continues processing.

OPTIONS FMTERR | NOFMTERR;

Creating Formats from SAS Data Sets

- PROC FORMAT'S CNTLIN = option is used to read the input control data set and create the format.
- The input control data set must be of a certain form with all the information needed to create the format.

INPUT CONTROL DATA SETS (CNTLIN=)

```
data one;
input ssn $9. name $;
cards;
123456789 Mickey
123456789 Mickey
;
```

```
data formatssn;
set one;
fmtname='$person';
type='C';
rename ssn=start name=label;
```

```
proc sort data=formatssn
  out=formatssn nodupkey;
  by start;
```

```
proc format cntlin=formatssn;
```

TYPE:
C for Character FORMAT
N for Numeric FORMAT
I for Numeric INFORMAT
J for Character INFORMAT

```
proc format;
  value $name
  '123456789'='Mickey';
```

```
proc print data=one;
  var ssn;
  format ssn $name.;
```

```
proc print data=one label;
  var ssn;
  format ssn $person.;
  label ssn='Celebrity';
```

The SAS System

Obs	Celebrity
1	Mickey
2	Mickey

Creating SAS Data Sets from Custom Formats

- Use PROC FORMAT'S CNTLOUT = option to create a SAS data set (a.k.a. output control data set).
- The output data set will contain variables that completely describe all aspects of each format, including optional settings.

Creating SAS Data Sets from Custom Formats (continued)

- Control output data sets are useful when you need to modify a format but no longer have the specifications for the format in a SAS program or in the form of an input control data set.
  1. Use the CNTLOUT= option to obtain the output control data set associated with a format.
  2. Edit the data set so that it is suitable for use with the CNTLIN= option.
  3. Create the format using the updated data set using the CNTLIN= option.

Creating SAS Data Sets from Custom Formats (continued)

```
proc format cntlout=outform;
 value $gender 'M'='male' 'F'='female';
```

Creating SAS Data Sets from Custom Formats (continued)

This is data set outform.

		F					D										
		M					E	L		P			N				
		T	S		L		F	E		R			O		S	E	
		N	T		A		A	N	F	E	M	F	E	T	E	E	
O		A	A	E	B	M	M	U	G	U	F	U	I	D	Y	X	X	H
B		M	R	N	E	I	A	L	T	Z	I	L	L	I	P	C	C	L
S		E	T	D	L	N	X	T	H	Z	X	T	L	T	E	L	L	O
1	GENDER	F	F	female	1	40	6	6	0		0		0	C	N	N	
2	GENDER	M	M	male	1	40	6	6	0		0		0	C	N		

Creating SAS Data Sets from Custom Formats (continued)

This is the PROC CONTENTS output for data set OUTFORM.

```
Data Set Name: WORK.OUTFORM              Observations:          2
Member Type:   DATA                      Variables:             17
Engine:        V612                      Indexes:               0
Created:       22:16 Tue, Jul 13, 2011   Observation Length:    63
Last Modified: 22:16 Tue, Jul 13, 2011   Deleted Observations:  0
Protection:                              Compressed:            NO
Data Set Type:                           Sorted:                NO
Label:

          -----Engine/Host Dependent Information-----

               Data Set Page Size:        8192
               Number of Data Set Pages: 1
               File Format:               607
               First Data Page:           1
               Max Obs per Page:          129
Obs in First Data Page:    2
```

Creating SAS Data Sets from Custom Formats (continued)

This is the rest of the PROC CONTENTS output for data set OUTFORM.

-----Alphabetic List of Variables and Attributes-----

#	Variable	Type	Len	Pos	Label
7	DEFAULT	Num	3	22	Default length
16	EEXCL	Char	1	52	End exclusion
3	END	Char	1	9	Ending value for format
12	FILL	Char	1	46	Fill character
1	FMTNAME	Char	8	0	Format name
9	FUZZ	Num	8	28	Fuzz value
17	HLO	Char	10	53	Additional information
4	LABEL	Char	6	10	Format value label
8	LENGTH	Num	3	25	Format length
6	MAX	Num	3	19	Maximum length
5	MIN	Num	3	16	Minimum length
11	MULT	Num	8	38	Multiplier
13	NOEDIT	Num	3	47	Is picture string noedit?
10	PREFIX	Char	2	36	Prefix characters
15	SEXCL	Char	1	51	Start exclusion
2	START	Char	1	8	Starting value for format
14	TYPE	Char	1	50	Type of format