Applying SAS Formats and Informats

Temporarily Assigning Formats to Variables

In your SAS reports, formats control how the data values are displayed. To make data values more understandable when they are displayed in your procedure output, you can use the FORMAT statement, which associates formats with variables.

Formats affect only how the data values appear in output, not the actual data values as they are stored in the SAS data set.

Syntax, FORMAT statement:

FORMAT *variable(s) format-name***;**

- *variable(s)* is the name of one or more variables whose values are to be written according to a particular pattern
- *format-name* specifies a SAS format or a user-defined format that is used to write out the values.

Tip: The FORMAT statement applies only to the PROC step in which it appears.

You can use a separate FORMAT statement for each variable, or you can format several variables (using either the same format or different formats) in a single FORMAT statement.

Table 12.1 *Formats That Are Used to Format Data*

FORMAT Statement	Description	Example
`format date mmddyy8.;`	associates the format MMDDYY8. with the variable Date	01/06/17
`format net comma5.0` `gross comma8.2;`	associates the format COMMA5.0 with the variable Net and the format COMMA8.2 with the variable Gross	1,234 5,678.90
`format net gross dollar9.2;`	associates the format DOLLAR9.2 with both variables, Net, and Gross	$1,234.00 $5,678.90

For example, the FORMAT statement below writes values of the variable Fee using dollar signs, commas, and no decimal places.

```
proc print data=cert.admit;
  var actlevel fee;
  where actlevel='HIGH';
  format fee dollar4.;
run;
```

Figure 12.1 *FORMAT Statement Output*

Obs	ActLevel	Fee
1	HIGH	$85
2	HIGH	$125
6	HIGH	$125
11	HIGH	$150
14	HIGH	$125
18	HIGH	$85
20	HIGH	$150

Specifying SAS Formats

The table below describes some SAS formats that are commonly used in reports.

Table 12.2 *Commonly Used SAS Formats*

Format	Description	Example
COMMA*w.d*	specifies values that contain commas and decimal places	comma8.2
DOLLAR*w.d*	specifies values that contain dollar signs, commas, and decimal places	dollar6.2
MMDDYY*w*.	specifies values as date values of the form 09/12/17 (MMDDYY8.) or 09/12/2017 (MMDDYY10.)	mmddyy10.
w.	specifies values that are rounded to the nearest integer in *w* spaces	7.
w.d	specifies values that are rounded to *d* decimal places in *w* spaces	8.2
$w.	specifies values as character values in *w* spaces	$12.
DATE*w*.	specifies values as date values of the form 16OCT17 (DATE7.) or 16OCT2017 (DATE9.)	date9.

Field Widths

All SAS formats specify the total field width (*w*) that is used for displaying the values in the output. For example, suppose the longest value for the variable Net is a four-digit number, such as 5400. To specify the COMMA*w.d* format for Net, you specify a field width of 5 or more. You must count the comma, because it occupies a position in the output.

Note: When you use a SAS format, specify a field width (*w*) that is wide enough for the largest possible value. Otherwise, values might not be displayed properly.

Figure 12.2 *Specifying a Field Width (w) with the FORMAT Statement*

```
format net comma5.0;
 5   ,   4   0   0
 1   2   3   4   5
```

Decimal Places

For numeric variables, you can also specify the number of decimal places (*d*), if any, to be displayed in the output. Numbers are rounded to the specified number of decimal places. In the example above, no decimal places are displayed.

Writing the whole number 2030 as 2,030.00 requires eight print positions, including two decimal places and the decimal point.

Figure 12.3 *Whole Number Decimal Places*

```
format qtr3tax comma8.2;
 2   ,   0   3   0   .   0   0
 1   2   3   4   5   6   7   8
```

Formatting 15374 with a dollar sign, commas, and two decimal places requires 10 print positions.

Figure 12.4 *Specifying 10 Decimal Places*

```
format totsales dollar10.2;
 $   1   5   ,   3   7   4   .   0   0
 1   2   3   4   5   6   7   8   9   10
```

Examples: Data Values and Formats

This table shows you how data values are displayed when different format, field width, and decimal place specifications are used.

Table 12.3 *Displaying Data Values with Formats*

Stored Value	Format	Displayed Value
38245.3975	COMMA9.2	38,245.40
38245.3975	8.2	38245.40
38245.3975	DOLLAR10.2	$38,245.40
38245.3975	DOLLAR9.2	$38245.40
38245.3975	DOLLAR8.2	38245.40
0	MMDDYY8.	01/01/60
0	MMDDYY10.	01/01/1960
0	DATE7.	01JAN60
0	DATE9.	01JAN1960

TIP If a format is too small, the following message is written to the SAS log:

```
NOTE: At least one W.D format was too small for the number to be
printed. The decimal might be shifted by the 'BEST' format.
```

The FORMAT Procedure

Definitions

SAS format
: determines how variable values are printed according to the data type: numeric, character, date, time, or timestamp.

SAS informat
: determines how data values are read and stored according to the data type: numeric, character, date, time, or timestamp.

A Word about PROC FORMAT

SAS provides you with formats and informats that you can use to read and write your data. However, if the SAS formats or informats do not meet your needs, you can use the FORMAT procedure to define your own formats and informats. PROC FORMAT stores user-defined formats and informats as entries in a SAS catalog.

The following output of Work.Carsurvey has a value of `1` or `2` for Sex, and values of `B`, `G`, `W`, and `Y` for Color. SAS does not provide formats to make the values for Sex and Color easier to read. You can create your own formats to format the values. You can also apply a format to the values of Income.

Figure 12.5 *Work.Carsurvey Data Set*

Obs	Age	Sex	Income	Color
1	19	1	14000	Y
2	45	1	65000	G
3	62	2	35000	W
4	31	1	44000	Y
5	58	2	83000	W
6	68	1	44000	B
7	17	2	15000	G
8	70	2	33000	B

The PROC FORMAT Statement

To begin a PROC FORMAT step, you use a PROC FORMAT statement.

Syntax, PROC FORMAT statement:

PROC FORMAT <*options*>**;**

options includes the following:

- LIBRARY=*libref* specifies the libref for a SAS library to store a permanent catalog of user-defined formats
- FMTLIB displays a list of all of the formats in your catalog, along with descriptions of their values.

Anytime you use PROC FORMAT to create a format, the format is stored in a format catalog. If the SAS library does not already contain a format catalog, SAS automatically creates one. If you do not specify the LIBRARY= option, the formats are stored in a default format catalog named Work.Formats.

The libref Work signifies that any format that is stored in Work.Formats is a temporary format; it exists only for the current SAS session.

Permanently Storing Your Formats

To store formats in a permanent format catalog named Formtlib.Formats:

- Specify a LIBNAME statement that associates the libref with the permanent SAS library in which the format catalog is to be stored.

  ```
  libname formtlib 'c:\sas\formats\lib';
  ```

- Specify the LIBRARY= option in the PROC FORMAT statement and specify the libref formtlib.

  ```
  PROC FORMAT LIBRARY=formtlib;
  ```

  The LIBRARY= option accepts a libref and a catalog in the format *library.format*. When the LIBRARY= option specifies a libref and not a catalog, PROC FORMAT uses the catalog Formats.

When you associate a user-defined format with a variable in a subsequent DATA or PROC step, use the Library libref to reference the location of the format catalog.

Any format that you create in this PROC FORMAT step is now stored in a permanent format catalog called Formtlib.Formats.

```
libname formtlib 'C:\Users\Student1\formats\lib';
proc format library=formtlib;
   ...more SAS statements...
run;
```

In the program above, the catalog Formtlib.Formats is located in the SAS library **`C:\Users\Student1\formats\lib`**, which is referenced by the libref Formtlib.

Notice that LIB= is an acceptable abbreviation for the LIBRARY= option.

```
proc format lib=formtlib;
```

Defining a Unique Format

The VALUE Statement

Use the VALUE statement to define a format for displaying one or more values.

Syntax, VALUE statement:

VALUE *format-name*
range1='label1'
range2='label2'
...more format-names...;

The following are true about *format-name*:

- A format name must begin with a dollar sign ($) if the format applies to character data.
- A format name must be a valid SAS name.
- A format name cannot be the name of an existing SAS format.
- A format name cannot end in a number.
- A format name does not end in a period when specified in a VALUE statement.
- A numeric format name can be up to 32 characters long.
- A character format name can be up to 31 characters long.

Tip: If you are running a version of SAS prior to SAS®9, the format name must be a SAS name up to eight characters long and cannot end in a number.

Notice that the statement begins with the keyword VALUE and ends with a semicolon after all the labels have been defined. The following VALUE statements create the GENDER, AGEGROUP, and $COL formats to specify descriptive labels that are later assigned to the variables Sex, Age, and Color respectively:

```
proc format;
   value gender
               1 = 'Male'
               2 = 'Female';
   value agegroup
             13 -< 20  = 'Teen'
             20 -< 65  = 'Adult'
             65 - HIGH = 'Senior';
   value $col
             'W' = 'Moon White'
             'B' = 'Sky Blue'
             'Y' = 'Sunburst Yellow'
             'G' = 'Rain Cloud Gray';
run;
```

The VALUE range specifies the following types of values:

- a single value, such as 24 or '**S**'
- a range of numeric values, such as 0-1500
- a range of character values enclosed in quotation marks, such as '**A**'-'**M**'

- a list of unique values separated by commas, such as 90,180,270 or '**B**', '**D**', '**F**'. These values can be character values or numeric values, but not a combination of character and numeric values (because formats themselves are either character or numeric).

When the specified values are character values, they must be enclosed in quotation marks and must match the case of the variable's values. The format's name must also start with a dollar sign ($). For example, the VALUE statement below defines the $COL format, which displays the character values as text labels.

```
proc format lib=formtlib;
   value $col
         'W' = 'Moon White'
         'B' = 'Sky Blue'
         'Y' = 'Sunburst Yellow'
         'G' = 'Rain Cloud Gray';
run;
```

When the specified values are numeric values, they are not enclosed in quotation marks, and the format's name should not begin with a dollar sign ($).

Specifying Value Ranges

You can specify a non-inclusive range of numeric values by using the less than symbol (<) to avoid any overlapping. In this example, the range of values from 0 to less than 13 is labeled as Child. The next range begins at 13, so the label Teenager would be assigned to the values 13 to 19.

```
proc format lib=formtlib;
   value agefmt
         0-<13='child'
         13-<20='teenager'
         20-<65='adult'
         65-100='senior citizen';
run;
```

You can also use the keywords LOW and HIGH to specify the lower and upper limits of a variable's value range. The keyword LOW does not include missing numeric values. The keyword OTHER can be used to label missing values as well as any values that are not specifically addressed in a range.

```
proc format lib=formtlib;
   value agefmt
         low-<13='child'
         13-<20='teenager'
         20-<65='adult'
         65-high='senior citizen'
         other='unknown';
run;
```

TIP If applied to a character format, the keyword LOW includes missing character values.

When specifying a label for displaying each range, remember to do the following:

- Enclose the label in quotation marks.
- Limit the label to 32,767 characters.
- Use two single quotation marks if you want an apostrophe to appear in the label:

```
000='employee''s jobtitle unknown';
```

To define several formats, you can use multiple VALUE statements in a single PROC FORMAT step. In this example, each VALUE statement defines a different format.

```
proc format;
   value gender
               1 = 'Male'
               2 = 'Female';
   value agegroup
             13 -< 20  = 'Teen'
             20 -< 65  = 'Adult'
             65 - HIGH = 'Senior';
   value $col
             'W' = 'Moon White'
             'B' = 'Sky Blue'
             'Y' = 'Sunburst Yellow'
             'G' = 'Rain Cloud Gray';
run;
```

The SAS log prints notes informing you that the formats have been created.

Log 12.1 *SAS Log*

```
146  proc format lib=formtlib;
147      value gender    1 = 'Male'
148                      2 = 'Female';
NOTE: Format GENDER is already on the library FORMTLIB.FORMATS.
NOTE: Format GENDER has been output.
149      value agegroup  13 -< 20  = 'Teen'
150                      20 -< 65  = 'Adult'
151                      65 - HIGH = 'Senior';
NOTE: Format AGEGROUP is already on the library FORMTLIB.FORMATS.
NOTE: Format AGEGROUP has been output.
152      value $col      'W' = 'Moon White'
153                      'B' = 'Sky Blue'
154                      'Y' = 'Sunburst Yellow'
155                      'G' = 'Rain Cloud Gray';
NOTE: Format $COL is already on the library FORMTLIB.FORMATS.
NOTE: Format $COL has been output.
```

Associating User-Defined Formats with Variables

How SAS Finds Format Catalogs

To use the GENDER, AGEGROUP, and $COL formats in a subsequent SAS session, you must assign the libref Formtlib again.

```
libname formtlib 'C:\Users\Student1\formats\lib';
```

SAS searches for the formats GENDER, AGEGROUP, and $COL in two libraries, in this order:

- the temporary library referenced by the libref Work
- a permanent library referenced by the libref Formtlib

SAS uses the first instance of a specified format that it finds.

TIP You can delete formats using PROC CATALOG.

Assigning Formats to Variables

Just as with SAS formats, you associate a user-defined format with a variable in a FORMAT statement.

```
data work.carsurvey;
  set cert.cars;
  format Sex gender. Age agegroup. Color $col. Income Dollar8.;
run;
```

Remember, you can place the FORMAT statement in either a DATA step or a PROC step. By placing the FORMAT statement in a DATA step, you permanently associate a format with a variable. Note that you do not have to specify a width value when using a user-defined format.

When you submit the PRINT procedure, the output for Work.CarSurvey now shows descriptive labels instead of the values for Age, Sex, Income, and Color.

```
proc print data=work.carsurvey;
run;
```

Output 12.1 *Work.CarSuvery Data Set with Formatted Values*

Obs	Age	Sex	Income	Color
1	Teen	Male	$14,000	Sunburst Yellow
2	Adult	Male	$65,000	Rain Cloud Gray
3	Adult	Female	$35,000	Moon White
4	Adult	Male	$44,000	Sunburst Yellow
5	Adult	Female	$83,000	Moon White
6	Senior	Male	$44,000	Sky Blue
7	Teen	Female	$15,000	Rain Cloud Gray
8	Senior	Female	$33,000	Sky Blue

When associating a format with a variable, remember to do the following:

- Use the same format name in the FORMAT statement that you specified in the VALUE statement.
- Place a period at the end of the format name when it is used in the FORMAT statement.

If you do not format all of a variable's values, then those that are not listed in the VALUE statement are printed as they appear in the SAS data set. In the example below, the value of 2 was not defined in the VALUE statement for GENDER as shown in observation 3, 5, 7, and 8.

```
libname formtlib 'C:\Users\Student1\formats\lib';
proc format lib=formtlib;
   value gender
               1 = 'Male';
   value agegroup
             13 -< 20  = 'Teen'
             20 -< 65  = 'Adult'
             65 - HIGH = 'Senior';
   value $col
```

```
                'W' = 'Moon White'
                'B' = 'Sky Blue'
                'Y' = 'Sunburst Yellow'
                'G' = 'Rain Cloud Gray';
run;
data work.carsurvey;
  set cert.cars;
  format Sex gender. Age agegroup.Color $col. Income Dollar8.;
run;
proc print data=work.carsurvey;
run;
```

Output 12.2 *Work.Carsurvey Data Set with Missing Formatted Values*

Obs	Age	Sex	Income	Color
1	Teen	Male	$14,000	Sunburst Yellow
2	Adult	Male	$65,000	Rain Cloud Gray
3	Adult	2	$35,000	Moon White
4	Adult	Male	$44,000	Sunburst Yellow
5	Adult	2	$83,000	Moon White
6	Senior	Male	$44,000	Sky Blue
7	Teen	2	$15,000	Rain Cloud Gray
8	Senior	2	$33,000	Sky Blue

Displaying User-Defined Formats

When you build a large catalog of permanent formats, it can be easy to forget the exact spelling of a specific format name or its range of values. Adding the keyword FMTLIB to the PROC FORMAT statement displays a list of all the formats in your catalog, along with descriptions of their values.

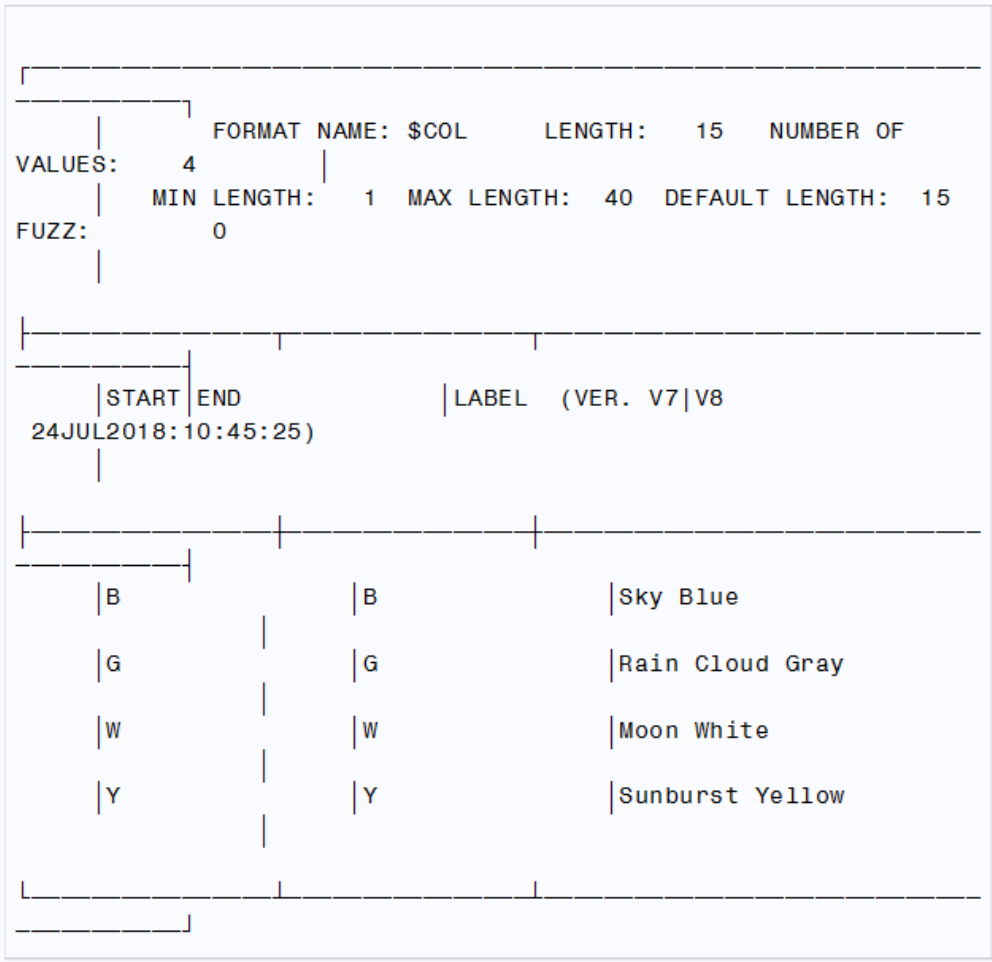
```
libname formtlib 'c:\sas\formats\lib';
proc format library=formtlib fmtlib;
run;
```

When you submit this PROC step, a description of each format in your permanent catalog is displayed as output.

Output 12.3 *Output of the Formtlib Catalog*

```
┌──────────────────────────────────────────────────────────────────────
──────────┐
     │       FORMAT NAME: AGEGROUP LENGTH:    6   NUMBER OF
VALUES:    3        │
     │   MIN LENGTH:   1  MAX LENGTH:  40  DEFAULT LENGTH:   6
FUZZ: STD
     │
├──────────────┬──────────────┬──────────────────────────────────────
──────────┤
     │START│END             │LABEL  (VER. V7|V8
 24JUL2018:10:45:25)
     │
├──────────────┼──────────────┼──────────────────────────────────────
──────────┤
     │              13│             20<Teen
               │
     │              20│             65<Adult
               │
     │              65│HIGH           │Senior
               │
└──────────────┴──────────────┴──────────────────────────────────────
──────────┘
```

```
┌──────────────────────────────────────────────────────────────────────
──────────┐
     │       FORMAT NAME: GENDER   LENGTH:    6   NUMBER OF
VALUES:    2        │
     │   MIN LENGTH:   1  MAX LENGTH:  40  DEFAULT LENGTH:   6
FUZZ: STD
     │
├──────────────┬──────────────┬──────────────────────────────────────
──────────┤
     │START│END             │LABEL  (VER. V7|V8
 24JUL2018:10:45:25)
     │
├──────────────┼──────────────┼──────────────────────────────────────
──────────┤
     │              1│              1│Male
               │
     │              2│              2│Female
               │
└──────────────┴──────────────┴──────────────────────────────────────
──────────┘
```



```
┌──────────────────────────────────────────────────────────────────────
──────────┐
     │       FORMAT NAME: $COL     LENGTH:   15   NUMBER OF
VALUES:    4         │
     │   MIN LENGTH:   1  MAX LENGTH:  40  DEFAULT LENGTH:  15
FUZZ:        0
     │
├─────────────┬──────────────┬──────────────────────────────────────
──────────┤
     │START│END             │LABEL  (VER. V7|V8
 24JUL2018:10:45:25)
     │
├─────────────┼──────────────┼──────────────────────────────────────
──────────┤
     │B               │B               │Sky Blue
               │
     │G               │G               │Rain Cloud Gray
               │
     │W               │W               │Moon White
               │
     │Y               │Y               │Sunburst Yellow
               │
└─────────────┴──────────────┴──────────────────────────────────────
──────────┘
```

In addition to the name, range, and label, the format description includes the following details:

- length of the longest label
- number of values defined by this format
- version of SAS that was used to create the format
- date and time of creation