FORMAT, you can design informats for reading and interpreting non-standard data, and you can design formats for displaying data in non-standard ways.

...Example 7.2...

```
data c99;
infile "f:\sasclass\data\cancer99.dat";
input (county gender age cause place) ($2. $1. 3. $4. $1.);
run;

proc format;                                                                      1
value $gen
'1' = 'MALE'
'2' = 'FEMALE'
;

value $cau
'C340'-'C349' = 'BRONCHUS & LUNG'
'C180'-'C189' = 'COLON'
'C500'-'C509' = 'BREAST'
'C250'-'C259' = 'PANCREAS'
'C61 '        = 'PROSTATE'
;
run;

proc freq data=c99;
where cause in : ('C34' 'C18' 'C50' 'C25' 'C61');                                 2
table cause*gender/norow nocol nopercent;
format cause $cau. gender $gen.;                                                  3
run;
```

```
Table of cause by gender
cause            gender
Frequency       |MALE    |FEMALE  |  Total
----------------+--------+--------+
COLON           |   1705 |   1940 |   3645
----------------+--------+--------+
PANCREAS        |   1065 |   1155 |   2220
----------------+--------+--------+
BRONCHUS & LUNG |   5317 |   4319 |   9636
----------------+--------+--------+
BREAST          |     41 |   3113 |   3154
----------------+--------+--------+
PROSTATE        |   2050 |      0 |   2050
----------------+--------+--------+
Total              10178    10527    20705
```

1 PROC FORMAT is used to create formats used to label values of the GENDER and to group and label values of the variable CAUSE.
2 Observations used in PROC FREQ are limited with a WHERE statement.
3 The formats are associated with the variables with a FORMAT statement within the procedure.

The format $GND labels values of GENDER. The format $CAU not only labels values of the variable CAUSE, but also groups observations according to the rules expressed on the left side of the ='s in PROC FORMAT. The rules require a knowledge of the data. Four of the causes use a range of values, a starting value and an ending value separated by a dash. One cause needs only a single value, prostate cancer, since there is only one value allowed for that cause, i.e. C61 followed by one blank. Each format is used with a character variable so the format name begins with a $. Question: in what order do the data values appear on the two axes of the table? The values are in order of the underlying data, not the formatted values. That can be changed using an option in PROC FREQ.

...Example 7.3...
```
proc freq data=c99 order=formatted;                                                       1
where cause in : ('C34' 'C18' 'C50' 'C25' 'C61');
table cause*gender/norow nocol nopercent;
format cause $cau. gender $gen.;
run;
```

```
cause            gender
Frequency       |FEMALE  |MALE    |  Total
----------------+--------+--------+
BREAST          |   3113 |     41 |   3154
----------------+--------+--------+
BRONCHUS & LUNG |   4319 |   5317 |   9636
----------------+--------+--------+
COLON           |   1940 |   1705 |   3645
----------------+--------+--------+
PANCREAS        |   1155 |   1065 |   2220
----------------+--------+--------+
PROSTATE        |      0 |   2050 |   2050
----------------+--------+--------+
Total              10527    10178    20705
```

1 The ORDER= option requests a table that is in order according to the FORMATTED values of the variables.

Formats can also be created to be used with numeric data. There is one numeric variable in the data set C99, AGE, and the next example creates a format that is used to create a table of cancer deaths according to age group.

...Example 7.4...
```
proc format;
value age                                                                                 1
low - 44   = '<45'                                                                        2
45  - 54   = '45-54'                                                                      3
55  - 64   = '55-64'
65  - 74   = '65-74'
75  - high = '75+'                                                                        4
;
value $pla
'A'-'C'    ='HOSP/OTHER'
'D'        ='HOSP/INPATIENT'
'E'        ='OTHER INST'
'F'        ='RESIDENCE'
'G','H','N'='OTHER'                                                                       5
other      ='UNKNOWN'
;
run;

proc freq data=c99;
tables place*age/norow nopercent missprint;                                               7
format place $pla. age age.;
run;
```

```
Table of place by age
place          age

Frequency      |
Col Pct        |   .    |<45     |45-54   |55-64   |65-74   |75+     |  Total
---------------+--------+--------+--------+--------+--------+--------+
UNKNOWN        |      0 |     12 |     21 |     34 |     60 |    108 |      .
               |      . |      . |      . |      . |      . |      . |
---------------+--------+--------+--------+--------+--------+--------+
HOSP/OTHER     |      1 |     69 |    132 |    225 |    340 |    473 |   1239
               |      . |   4.32 |   4.03 |   3.76 |   3.33 |   2.91 |
---------------+--------+--------+--------+--------+--------+--------+
HOSP/INPATIENT |      2 |   1090 |   2028 |   3519 |   5623 |   7980 |  20240
               |      . |  68.25 |  61.94 |  58.75 |  55.00 |  49.08 |
---------------+--------+--------+--------+--------+--------+--------+
OTHER INST     |      0 |     61 |    178 |    381 |    909 |   2838 |   4367
               |      . |   3.82 |   5.44 |   6.36 |   8.89 |  17.45 |
---------------+--------+--------+--------+--------+--------+--------+
RESIDENCE      |      2 |    278 |    680 |   1401 |   2498 |   3398 |   8255
               |      . |  17.41 |  20.77 |  23.39 |  24.44 |  20.90 |
---------------+--------+--------+--------+--------+--------+--------+
OTHER          |      0 |     99 |    256 |    464 |    853 |   1570 |   3242
               |      . |   6.20 |   7.82 |   7.75 |   8.34 |   9.66 |
---------------+--------+--------+--------+--------+--------+--------+
Total                 .     1597     3274     5990    10223    16259    37343

Frequency Missing = 240
```

1 A format is created to be used with a numeric variable, the name does not have a $ prefix.
2 The word LOW is an allowable entry in PROC FORMAT, meaning the lowest value encountered in the data.
3 Numeric ranges use a dash to separate values.
4 The word HIGH is also an allowable entry in PROC FORMAT, the highest value encountered in the data.
5 Individual values in a format are separated by commas.
6 OTHER is a 'catch all' entry meaning all values not captured by other entries in the format.
7 The MISSPRINT option places missing data in the table, but not in computed percentages.

Notice that even though LOW and HIGH were used in the format for AGE, there are still missing values for the variable AGE in the table. Also, there are observations with a value of 'Z' for the variable PLACE in the data set. However, they are treated as missing in the table (they are in the row labeled UNKNOWN together with real missing values). More about both of these 'features' later in the notes.

...CREATING AND USING FORMATS

There are a number of rules for creating and using formats. One of the rules forces you to think ahead in that when you create a format, the format type determines the variable type with which the format can be used....

FORMAT TYPES
if you plan to use a format with a character variable, create a character format
if you plan to use a format with a numeric variable, create a numeric format

FORMAT NAMES
the name of the format determines the format type
if a format name begins with a $, it is character format
if there is no $ at the start of the name, it is a numeric format
a format name can be up to thirty-two characters in length (a $ counts as part of the thirty-two characters)
a format name must start with a letter or underscore (the first character or the first character after the $)
a format name can only contain letters, numbers, and underscores
a format name cannot end in a number

FORMAT RULES-LEFT SIDE OF ='S
values on the left side of the ='s are variable values
individual values, multiple values separated by commas, ranges of values separated by a dash are allowed
combinations of individual values, multiple values, and ranges are allowed
numeric values are not enclosed in quotes
character values are enclosed in quotes
the words LOW, HIGH, and OTHER may appear in addition to variable values

FORMAT RULES-RIGHT SIDE OF ='S
a format label may be up to 32,000 characters in length (the content to the right of the ='s sign)
use quotation marks around format labels (the content to the right of the ='s sign)

The use of quotation marks around all entries on the right side of the ='s for all formats is a convention rather than a rule, as is quotation marks around entries on the left side of the ='s for character formats. You can create formats without the quotation marks as shown in the next example.

...Example 7.5...
```
proc format;
value $gna
1,M = MALES
2,F = FEMALES
;
value $gnb
'1','M' = 'Males'
'2','F' = 'Females'
;
value gnc
1 = males
2 = females
;
run;

data test;
input (ga gb gc) (2*$1. 1.);
datalines;
1M1
2F2
M11
F22
;
run;

proc print data=test noobs;
var ga gb gc;
format ga $gna. gb $gnb. gc gnc.;
run;
```

```
  ga         gb         gc
MALES      Males      males
FEMALES    Females    females
MALES      Males      males
FEMALES    Females    females
```

The output from PROC PRINT shows that all the formats worked as intended. Also, if you look at the LOG from this job, you will see that there are no errors in PROC FORMAT and that all formats ($gna, $gnb, gnc) are created successfully...

```
193  proc format;
194  value $gna
195  1,M = MALES
196  2,F = FEMALES
197  ;
```

```
NOTE: Format $GNA has been output.

198  value $gnb
199  '1','M' = 'Males'
200  '2','F' = 'Females'
201  ;
NOTE: Format $GNB has been output.
202  value gnc
203  1 = males
204  2 = females
205  ;
NOTE: Format GNC has been output.
206  run;
```

Example 7.5 shows another rule to remember...

multiple values on the left side of the equals sign must be separated by commas

If you do not separate values by commas, there will not be any error messages in the LOG but the formats you create will not be what you intended. The next example shows you one way to display the formats that you have created, and what happens if you leave out the commas when using multiple values on the left side of the ='s.

...Example 7.6...

```
proc format;
value $gnw
    1,M = MALES
    2,F = FEMALES;
value $gnx
    '1','M' = 'Males'
    '2','F' = 'Females';
value $gny
    1 M = MALES
    2 F = FEMALES;
value $gnz
    '1' 'M' = 'Males'
    '2' 'F' = 'Females';
run;

proc format;
select $gnw $gnx $gny $gnz;
run;
```

There are no errors in the LOG...

```
225  proc format;
226  value $gnw  1,M = MALES
227      2,F = FEMALES;
NOTE: Format $GNW has been output.
228  value $gnx  '1','M' = 'Males'
229      '2','F' = 'Females';
NOTE: Format $GNX has been output.
230  value $gny  1 M = MALES
231      2 F = FEMALES;
NOTE: Format $GNY has been output.
232  value $gnz  '1' 'M' = 'Males'
233      '2' 'F' = 'Females';
NOTE: Format $GNZ has been output.
```

The FMTLIB option in PROC FORMAT produces the following output (it is restricted to the four formats listed in the SELECT statement - otherwise all formats created during the current SAS session are listed)....

```
----------------------------------------------------------------------------
|       FORMAT NAME: $GNW     LENGTH:    7   NUMBER OF VALUES:    4        |
|   MIN LENGTH:   1  MAX LENGTH:  40  DEFAULT LENGTH   7  FUZZ:        0   |
|--------------------------------------------------------------------------|
|START           |END             |LABEL  (VER. 8.2     10OCT2001:11:21:10)|
|----------------+----------------+----------------------------------------|
|1               |1               |MALES                                   |
|2               |2               |FEMALES                                 |
|F               |F               |FEMALES                                 |
|M               |M               |MALES                                   |
----------------------------------------------------------------------------

----------------------------------------------------------------------------
|       FORMAT NAME: $GNX     LENGTH:    7   NUMBER OF VALUES:    4        |
|   MIN LENGTH:   1  MAX LENGTH:  40  DEFAULT LENGTH   7  FUZZ:        0   |
|--------------------------------------------------------------------------|
|START           |END             |LABEL  (VER. 8.2     10OCT2001:11:21:10)|
|----------------+----------------+----------------------------------------|
|1               |1               |Males                                   |
|2               |2               |Females                                 |
|F               |F               |Females                                 |
|M               |M               |Males                                   |
----------------------------------------------------------------------------

----------------------------------------------------------------------------
|       FORMAT NAME: $GNY     LENGTH:   11   NUMBER OF VALUES:    2        |
|   MIN LENGTH:   1  MAX LENGTH:  40  DEFAULT LENGTH  11  FUZZ:        0   |
|--------------------------------------------------------------------------|
|START           |END             |LABEL  (VER. 8.2     10OCT2001:11:21:10)|
|----------------+----------------+----------------------------------------|
|1 M             |1 M             |MALES     2                             |
|F               |F               |FEMALES                                 |
----------------------------------------------------------------------------

----------------------------------------------------------------------------
|       FORMAT NAME: $GNZ     LENGTH:    7   NUMBER OF VALUES:    2        |
|   MIN LENGTH:   1  MAX LENGTH:  40  DEFAULT LENGTH   7  FUZZ:        0   |
|--------------------------------------------------------------------------|
|START           |END             |LABEL  (VER. 8.2     10OCT2001:11:21:10)|
|----------------+----------------+----------------------------------------|
|1M              |1M              |Males2                                  |
|F               |F               |Females                                 |
----------------------------------------------------------------------------
```

You should notice that the formats that were created with commas separating values are correct. Those without the commas have incorrect starts, ends, and labels.

...FORMATS AND MISSING DATA

In addition to listing the expected value or ranges of values of a variable, the words LOW and HIGH can be used on the left of the ='s in PROC FORMAT. These words mean the lowest value and highest value of the variable when the format is associated with a variable. Though the use of these words might seem obvious, the special case of missing data poses a problem (as already shown in example 7.4)

...Example 7.7...

```
proc format;
value age2grp                                                          1
low - 19   = "TEEN"
20  - 54   = "MIDDLE-AGE"
55  - high = "SENIOR"
;
run;
```

```
data ages;
input age @@;                                                                                  2
datalines;
10 20 56 32 78 34 65 . 86 . 45 185 32 100 56 12 10 0 199 87 54 13
;
run;

proc freq data=ages;
table age;
format age age2grp.;                                                                           3
run;

                                         Cumulative    Cumulative
       age    Frequency     Percent     Frequency      Percent
---------------------------------------------------------------
TEEN                 5       25.00             5        25.00
MIDDLE-AGE           6       30.00            11        55.00
SENIOR               9       45.00            20       100.00

Frequency Missing = 2                                                                          4
```

1 A NUMERIC format is created using the words LOW and HIGH.
2 The @@ option on the INPUT statement is used to read multiple values of the variable age that are on a single line of data (the @@ option tells SAS to hold onto the line of data being read for subsequent input of more data).
3 The format is used to group values of the variable AGE in PROC FREQ.
4 PROC FREQ reports two values of the variable AGE as MISSING and does not include them in either the LOW or HIGH range.

There are two things to remember about the results of this example. The most important is another format rule...

NEITHER the LOW or HIGH range values in a format include MISSING NUMERIC values

Next is that PROC FREQ does not include missing values in the frequency counts unless you tell it otherwise via the MISSING option.

...Example 7.8...

```
proc freq data=ages;
table age/missing;                                                                             1
table age/missprnt;                                                                            2
format age age2grp.;
run;

                                         Cumulative    Cumulative
       age    Frequency     Percent     Frequency      Percent
---------------------------------------------------------------
         .           2        9.09             2         9.09                                  3
TEEN                 5       22.73             7        31.82
MIDDLE-AGE           6       27.27            13        59.09
SENIOR               9       40.91            22       100.00

                                         Cumulative    Cumulative
       age    Frequency     Percent     Frequency      Percent
---------------------------------------------------------------
         .           2         .               .            .                                  4
TEEN                 5       25.00             5        25.00
MIDDLE-AGE           6       30.00            11        55.00
SENIOR               9       45.00            20       100.00

Frequency Missing = 2
```

1 The MISSING option is used in PROC FREQ to include missing values in the table and in the computing of percentages.
2 The MISSPRINT option is used in PROC FREQ to include missing values in the table, but NOT in the computing of percentages.
3 MISSING values are present in the table and influence the computed percentages.
4 MISSING values are present in the table, but DO NOT influence the computed. percentages.

You might have noticed that PROC FREQ still reports the FREQUENCY MISSING when the MISSPRINT option used. It is redundant when a one-dimensional table is produced, but does provide useful information in multi-dimensional tables (see example 7.4).

In addition to the missing values of age in the previous examples, you might have noticed two values of age that you might not want included in the SENIOR age group, i.e. 185 and 199. Assume you are willing to only allow an upper value of age at 110. You might also want to put a lower limit on the TEEN range, assume it is 10. You could have changed the way you wrote the format that grouped observations by age.

...Example 7.9..
```
proc format;
value age2grp                                                                        1
10  - 19   = "TEEN"
20  - 54   = "MIDDLE AGE"
55  - 110  = "SENIOR"
other      = "TOO LOW/TOO HIGH"
;
run;

data ages;
input age @@;
datalines;
10 20 56 32 78 34 65 . 86 . 45 185 32 100 56 12 10 0 199 87 54 13
;
run;

proc freq data=ages;
table age;
format age age2grp.;
run;

                                             Cumulative    Cumulative
             age    Frequency     Percent     Frequency      Percent
-------------------------------------------------------------------
TEEN                       4       23.53             4        23.53
MIDDLE AGE                 6       35.29            10        58.82
SENIOR                     7       41.18            17       100.00

Frequency Missing = 5                                                                2
```

1 The word OTHER is used on the left side of the ='s in PROC FORMAT to specify that all values not included in the ranges of age are to be considered as bad data.
2 All values of age not in the specified ranges were grouped with missing data (age 0, 185, and 199, plus the two missing values of age).

What happened when the OTHER range value was used? All of the 'bad' ages, i.e. those below 10 and those above 110 were captured as bad data, but they were grouped via the format with MISSING data. There is a VERY SHORT sentence in the SAS documentation that explains this behavior (another rule to remember)...

any values grouped with missing data by a format are considered as missing data

If you wanted to distinguish between missing data and reported values of age that were out of the acceptable range, you would have to modify the VALUE statement in PROC FORMAT.

...Example 7.10...
```
proc format;
value age2grp                                                                                  1
10  - 19   = "TEEN"
20  - 54   = "MIDDLE AGE"
55  - 110  = "SENIOR"
other      = 'TOO LOW/TOO HIGH'
.          = 'MISSING'
;
run;

proc freq data=ages;
table age;
format age age2grp.;
run;
                                                     Cumulative    Cumulative
             age    Frequency     Percent     Frequency      Percent
--------------------------------------------------------------------
TOO LOW/TOO HIGH           3       15.00             3        15.00                            2
TEEN                       4       20.00             7        35.00
MIDDLE AGE                 6       30.00            13        65.00
SENIOR                     7       35.00            20       100.00

Frequency Missing = 2
```

1 In addition to the range OTHER, a separate category is established for MISSING data. The order of the lines in the value statement does not matter since SAS rewrites the order in sort sequence order when it creates the format. Missing numeric values would not be picked up in the OTHER category even though OTHER occurred before the missing range.
2 There are only two missing values in the table.

The behavior of a character format differs from that of a numeric format when missing values are encountered. We can create a format to be used with a character variable and use PROC FREQ to show the result of using the format when missing data are encountered. The Apgar Score is a qualitative measure used to evaluate an infant's health at birth. The score ranges from 0 to 10. In the following example, the character X is used to represent a score of 10.

...Example 7.11...
```
proc format;
value $apgar                                                                                   1
low -'6'   = 'NOT HEALTHY'
'7' - high = 'HEALTHY'
;
run;

data births;
input apg_score : $1. @@;                                                                      2
datalines;
1 X 3 7 9 . 6
;
run;

proc freq data=births;                                                                         3
table apg_score;
format apg_score $apgar.;
run;
                                              Cumulative    Cumulative
apg_score      Frequency     Percent     Frequency      Percent
-------------------------------------------------------------
HEALTHY              3      100.00             3       100.00                                  4
```

```
Frequency Missing = 4
```

1 A format is created to be used with character data. The format name (the value) begins with a $.
2 The variable APG_SCORE is a character variable and there is one observation with missing data. Notice the 'X' used to represent a score of 10.
3 The format is used in PROC FREQ to groups and label the observations.
3 There are 4 observations with missing data and 3 observations that are labeled HEALTHY.

Why did this happen? When using a character format, LOW will include MISSING data since a BLANK (SPACE) is an allowable value for a character variable. In this example, there was one missing value and it ended up grouped with the three observations that had real data in the range LOW-'6' because of the rule mentioned earlier.

the LOW range value in a format includes MISSING CHARACTER values

and...

any values grouped with missing data by a format are considered as missing data

There are three HEALTHY observations because the 'X' was grouped with the values '7' and '9'. When working with character data, it is quite helpful to know the sorting sequence used by SAS (shown in Appendix C). Letters come after numbers in the sort sequence. Thus, the word HIGH used in the range for HEALTHY infants includes the 'X' that appears in the data.

We can create another category for missing data as was done with the numeric format...

...Example 7.12...

```
proc format;
value $apgar
low -'6'   = 'NOT HEALTHY'
'7' - high = 'HEALTHY'
' '        = 'UNKNOWN'
;
run;
```

However, the LOG contains an error message.

```
ERROR: These two ranges overlap: LOW-6 and - (fuzz=0).
NOTE: The previous statement has been deleted.
```

and no format is created. The error is the result of the missing values being included in the LOW range. A format cannot have values repeated on the left side of the ='s and the value missing is assigned to two groups, NOT HEALTHY and MISSING. The following format will work correctly (try it on your own)...

```
proc format;
value $apgar
'0' - '6'   = 'NOT HEALTHY'
'7' -  high = 'HEALTHY'
' '         = 'UNKNOWN'
;
run;
```

...USING SAS-SUPPLIED FORMATS TO GROUP OBSERVATIONS

There are several SAS-supplied formats that allow you to group observations. The next example shows how to group observations based on the content of a character variable.

...Example 7.13...

```
data procedures;                                                                   1
input pr : $4. @@;
datalines;
3734 3791 3751 740 7499 412 4101
;
run;

proc freq data=procedures;                                                         2
table pr;
format pr $2.;                                                                     3
run;
```

```
                                Cumulative    Cumulative
pr    Frequency     Percent     Frequency      Percent
-------------------------------------------------------
37           3       42.86             3        42.86
41           2       28.57             5        71.43
74           2       28.57             7       100.00
```

1 A data set is created that contains ICD-9-CM procedure codes. The length of the character variable PR is 4.
2 PROC FREQ is used to count the number of individual procedures.
3 The SAS-supplied format $2. is used to group the values of the variable PR based on the first two characters in the variable value.

ICD-9-CM procedure codes can contain from 3 to 4 characters. However, the first 2 characters define the major subgroups of procedures. For example, in the above data, the first three procedures are...

```
3734  EXCISION OR DESTRUCTION OF OTHER LESION OR TISSUE OF HEART
3791  OPEN CHEST CARDIAC MASSAGE
3751  HEART TRANSPLANTATION
```

The major subgroup they all belong to is...

```
37  OTHER OPERATIONS ON HEART AND PERICARDIUM
```

There is no need to create a new variable to count procedures using only the first two characters in the procedure code. The format $2. will group observations based on the first two characters of any character variable. If you were using ICD-9-CM diagnosis codes (normally a character variable 5 length 5), they may contain from 3 to 5 characters. The format $3. can be used to count diagnoses in major subgroups.

In chapter 8, there are examples that show how to use SAS-supplied formats to group observations based on the values of variables that are dates. For example, observations can be grouped by year using the format YEAR. or month using the format MONTH.

...ADDITIONAL NOTES ON FORMAT RANGES

In addition to expressing ranges with values separated by a "-", there are instances when other symbols are required in a format. For example, what if we have data where one of the variables is a number of days. We want to create counts, but with observations grouped by weeks.

...Example 7.14...

```
proc format;
value weeks                                                                     1
low - 19   = "< 20 WEEKS"
20  - 49   = "20-49 WEEKS"
50  - high = "50+ WEEKS"
;
run;

data day_week;
input days @@;
weeks = days / 7;                                                               2
datalines;
140 136 250 300 350 400 348 120
;
run;

proc freq data=day_week;
tables weeks;
format weeks weeks.;
run;

                                        Cumulative    Cumulative
      weeks    Frequency     Percent     Frequency      Percent
-----------------------------------------------------------------
< 20 WEEKS            1       12.50            1        12.50
19.42857143           1       12.50            2        25.00                  3
20-49 WEEKS           3       37.50            5        62.50
49.71428571           1       12.50            6        75.00
50+ WEEKS             2       25.00            8       100.00
```

1 A format (WEEKS) is created to group observations by weeks.
2 A new variable (WEEKS) is created by dividing days by 7.
3 The results of PROC FREQ are not as expected.

These results occurred since two values of the variable weeks fall between the ranges, i.e. they are not included in any of the ranges in PROC FORMAT. When a value of a variable cannot be found in the values on the left side of the ='s in a format, the actual value of the variable is used. The value 19.42 falls between the upper end of the first range (19) and the lower end of the next range (20), while 49.71 falls between the upper end of the second range (49) and the lower end of the final range (50). The format can be modified in either of two ways to eliminate this problem...

```
proc format;
value weeks
low -< 20  = "< 20 WEEKS"
20  -< 50  = "20-49 WEEKS"
50  - high = "50+ WEEKS"
;
run;
```

This format indicates that you want all values that are less than 20 weeks included in the first range, and all values less than 50 weeks included in the second range. If you wanted fractional values treated differently, i.e. any value of weeks greater than 19 to be in the second range (20-49) and any value greater than 49 in the last range (50+), the format would be expressed as...

```
proc format;
value weeks
low  - 19   = "< 20 WEEKS"
19  <- 49   = "20-49 WEEKS"
49  <- high = "50+ WEEKS"
;
run;
```

The rules for the behavior of "-<" versus "<-" in a format are...

-<   include all values greater than or equal to the lower range value, and less than the upper range value
<-   include all values greater than the lower range value, and less than or equal to the upper range value

...ADDING AND REMOVING FORMATS
PROC FORMAT creates rules for the display of the values of variables. It does not apply these rules to any data sets. Application of a format to a variable in a data set is done with a FORMAT statement.

...Example 7.15...

```
proc format;                                                                   1
value $gen
'1' = 'MALE'
'2' = 'FEMALE'
;
value age
low - 44   = '<45'
45  - 54   = '45-54'
55  - 64   = '55-64'
65  - 74   = '65-74'
75  - high = '75+'
;
value $pla
'A'-'C'    ='HOSP/OTHER'
'D'        ='HOSP/INPATIENT'
'E'        ='OTHER INST'
'F'        ='RESIDENCE'
'G','H','N'='OTHER'
other      ='UNKNOWN'
;
value $cau
'C340'-'C349' = 'BRONCHUS & LUNG'
'C180'-'C189' = 'COLON'
'C500'-'C509' = 'BREAST'
'C250'-'C259' = 'PANCREAS'
'C61 '        = 'PROSTATE'
;
run;

data c99;
infile "f:\sasclass\data\cancer99.dat";
input (county gender age cause place) ($2. $1. 3. $4. $1.);
format gender $gen. age age. place $pla. cause $cau.;                          2
run;

proc print data=c99 (obs=5) obs='observation';                                 3
run;

observation    gender     age     place             cause              county
       1       MALE      65-74    HOSP/INPATIENT    BRONCHUS & LUNG      01
       2       FEMALE    55-64    OTHER INST        COLON                01
       3       FEMALE    <45      RESIDENCE         C56                  01
       4       MALE      55-64    HOSP/INPATIENT    C911                 01
       5       FEMALE    45-54    HOSP/INPATIENT    C80                  01
```

1   PROC FORMAT is used to create rules (formats) for the display of variable values.
2   The formats are permanently associated with variables using a FORMAT statement in a data step.

3 PROC PRINT display the first five observations (with a renamed OBS column). No format statement is needed in the procedure since the formats are permanently associated with the variables.

You can no longer see the original values of four of the variables. Notice that causes that are not listed in the creation of the format $cau. are listed with their original values. You can temporarily remove a format from a variable remove a format from a variable in a procedure.

...Example 7.16...

```
proc print data=c99 (obs=5) obs='observation';
format age cause;                                                                                    1
run;

observation    gender    age    place             cause    county
      1        MALE       70    HOSP/INPATIENT    C349       01
      2        FEMALE     62    OTHER INST        C189       01
      3        FEMALE     36    RESIDENCE         C56        01
      4        MALE       61    HOSP/INPATIENT    C911       01
      5        FEMALE     45    HOSP/INPATIENT    C80        01
```

1 A FORMAT statement is used that specifies variable names but no formats.

If no format is used is a format statement (i.e. only variables are specified), the formats are removed from the specified variables. You can now see the original vales of the variables age and cause.

There are two ways to permanently disassociate formats from variables. One uses a procedure, the other uses a data step.

...Example 7.17...

```
proc datasets lib=work nolist;                                                                       1
modify c99;                                                                                          2
format age cause;                                                                                    3
quit;                                                                                                4

proc print data=c99 (obs=5)  noobs;                                                                  5
run;

gender    age    place             cause    county
MALE       70    HOSP/INPATIENT    C349       01
FEMALE     62    OTHER INST        C189       01
FEMALE     36    RESIDENCE         C56        01
MALE       61    HOSP/INPATIENT    C911       01
FEMALE     45    HOSP/INPATIENT    C80        01
```

1 PROC DATASETS is used. The library is specified (LIB=WORK) that contains the data set to be altered. The NOLIST options prevents the listing of the names of all data sets in the specified library.
2 A MODIFY statement specifies the data set to be altered.
3 The FORMAT statement with only variable names permanently removes formats from two variables.
4 A QUIT statement ends the procedue (not a RUN statement).
5 No FORMAT statement is needed in PROC PRINT to see the original values of age and cause.

...Example 7.18...

```
data c99;
set c99;                                                                                             1
format gender place;                                                                                 2
run;

proc print data=c99 (obs=5) noobs;                                                                   3
run;
```

```
gender    place    age    cause    county
  1         D       70    C349       01
  2         E       62    C189       01
  2         F       36    C56        01
  1         D       61    C911       01
  2         D       45    C80        01
```

1 The original data set C99 is read with a SET statement.
2 A FORMAT statement is used to remove the formats from variables gender and place.
3 PROC PRINT shows the original values of all the variables.

PROC DATASETS is the better way to remove formats. Using a data step (example 7.16) requires reading and writing all the observations in the data set. However, using the procedure only makes changes in the header (descriptor portion) of the data set and does not read or write any observations. Another thing to keep in mind is that if you are removing formats in using a data step, the format statement MUST appear after the SET statement. Do you have any idea why?

...CREATING NEW VARIABLES USING FORMATS
Though formats can avoid the necessity of creating new variables for performing grouped analyses, formats also allow you to create new variables within a data step. The PUT function can create a new variable based on the value of an already existing variable according to rules specified in a format.

...Example 7.19...

```
proc format;                                                                      1
value agegr
low - 12   = "<13"
13  - 19   = "13-19"
20  - 44   = "20-44"
45  - 64   = "45-64"
65  - 84   = "65-84"
85  - high = "85+"
;
run;

data c99;
infile "f:\sasclass\data\cancer99.dat";
input (county gender age cause place) ($2. $1. 3. $4. $1.);
agroup = put(age,agegr.);                                                         2
run;

proc print data=c99 (obs=5) noobs;
var gender cause age agroup;
run;
```

```
gender    cause    age    agroup
  1       C349      70    65-84
  2       C189      62    45-64
  2       C56       36    20-44
  1       C911      61    45-64
  2       C80       45    45-64
```

1 PROC FORMAT creates a rule named AGE.
2 The PUT function creates a new variable based on the value if the variable age and the rule agegr.

The PUT function requires two arguments (arguments are the information within the parentheses that follow the name of the function): a variable name; a format name. The new variable that results from the use of the PUT function is ALWAYS a CHARACTER variable regardless of the type of variable being used to create the new variable.

When the format AGEGR was created, we planned to use it with the numeric variable age in either a subsequent data step or procedure. Therefore, the name (or VALUE) of the format did NOT begin with a $. Also, all the values on the left side of the ='s in PROC FORMAT were expressed without quote marks.

As with ALL formats, the values on the right side of the ='s do have quote marks (remember that is the safe method of creating a format, but not always required). A PUT function was used within the data step to create the new variable AGROUP. Though AGE is a NUMERIC variable and AGEGR is a NUMERIC format, the variable AGROUP is a character variable since the PUT function ALWAYS results in the creation of a character variable. Remember that a format (or VALUE) is merely a set of rules that you set up someplace in your SAS job. The rule is invoked within the data step with a PUT function. The rule AGEGR tells SAS to look a the value of the variable AGE and compare it to values on the left side of the ='s. When an appropriate value or range of values is found, SAS uses the values on the right side of the ='s to create the new value.

...CREATING NEW VARIABLES WITH IF-THEN-ELSE

Up to now, we have used formats to assign new values to variables or to create new variables (sometimes referred to as recoding). Example 7.1 showed that recoding could also be doe using an if-then-else statement. Though formats are an efficient way to recode variable values, sometimes it may be more convenient to use an if-then-else statement in lieu of a format.

In example 7.17, a PUT function and a format were used to create a new variable named AGROUP based on values of the variable AGE. You could also create the variable AGROUP using an if-then-else statement.

...Example 7.20...

```
data c99;
length agroup $5 gender $7;
infile 'f:\sasclass\data\cancer99.dat';
input (county gender age cause place) ($2. $1. 3. $3. $1.);

     if age lt 13 then agroup = '<13';                                                      1
else if age lt 20 then agroup = '13-19';
else if age lt 45 then agroup = '20-44';
else if age lt 65 then agroup = '45-64';
else if age lt 85 then agroup = '65-84';
else                   agroup = '85+';

if gender eq '1' then gender = 'MALE';
else
if gender eq '2' then gender = 'FEMALE';
else                  gender = 'UNKNOWN';

run;

proc freq data=c99;
table gender*agroup/norow nocol nopercent;                                                  2
run;
```

```
gender     agroup

Frequency|13-19   |20-44   |45-64   |65-84   |85+     |<13     |  Total
---------+--------+--------+--------+--------+--------+--------+
FEMALE   |     26 |    809 |   4578 |  10429 |   3152 |     47 |  19041
---------+--------+--------+--------+--------+--------+--------+
MALE     |     43 |    645 |   4741 |  10850 |   2219 |     44 |  18542
---------+--------+--------+--------+--------+--------+--------+
Total          69     1454     9319    21279     5371       91    37583
```

1 An IF-THEN-ELSE statement is used to create a new variable named AGROUP based on the values of the variable AGE. Notice, because of the way the IF-THEN-ELSE statement works (it stops as soon as a true portion is found), only one side of the age range was specified in each portion of the statement.
2 The new variables are used to count the number of observations grouped by gender and age group..

The order of age groups in the output from PROC FREQ might not be what you wanted. The order is based on the value of the variable and SAS considers a "<" as having a higher value than any number (in this case, 8 - look at the default sorting order of characters in appendix C). One way to get the correct result is to modify the value of the variable AGP.

```
     if age lt 13 then agroup = ' <13';
else if age lt 20 then agroup = '13-19';
else if age lt 45 then agroup = '20-44';
else if age lt 65 then agroup = '45-64';
else if age lt 85 then agroup = '65-84';
else                   agroup = '85+  ';
```

A space is now the first character in the value of AGROUP for those observations with an age less than 13. Spaces have a lower sorting value than numbers and those with ages less than 13 will now appear on the left of the table, not the right.

Since SAS stops the execution of an IF-THEN-ELSE statement when a true portion is found, it is more efficient to write such statements in the order that the various parts of the statement are expected to occur. If you look at the output in example 7.18, the most efficient way to write the statement would be as follows.

```
     if age ge 65 and age le 84 then agp = "65-84";
else if age ge 85               then agp = "85+";
else if age ge 45 and age le 64 then agp = "45-64";
else if age ge 20 and age le 44 then agp = "20-44";
else if age lt 13               then agp = " <13";
else                                 agp = "13-19";
```

or...

```
     if 65 le age le 84 then agp = "65-84";
else if       age ge 85 then agp = "85+";
else if 45 le age le 64 then agp = "45-64";
else if 20 le age le 44 then agp = "20-44";
else if       age lt 13 then agp = " <13";
else                         agp = "13-19";
```

Either of these IF-THEN-ELSE statements has the same result. Using either would give the same output as shown in example 7.18. Since the sections of the statement are not in age group order, there are sections where both of the ends of the age range must be checked to put an observation into the appropriate group.

...INFORMATS

Up to this point, PROC FORMAT has been used to create FORMATS, i.e. rules for labeling the values of variables or for grouping observations based on the values of variables. These rules only control how data values are displayed. The actual values of variables are not changed.

PROC FORMAT can also be used to create INFORMATS. You have already used some SAS-supplied informats to read raw data and create numeric (e.g. 6.) and character (e.g. $6.) variables. You have also used some SAS-supplied informats that modified the values of raw data that were read and converted them to a different value, i.e. DATE informats. If you read the value 10111979 with the informat MMDDYY10., SAS converts the number and stores the value as the number of days since 1/1/1960. The date informat changed the variable value from that which was read from the raw data.

You can write your own informats, or rules for reading raw data, using PROC FORMAT. Instead of a VALUE statement, you use an INVALUE statement. Many of the same rules that applied to formats are true for informats. An important distinction to remember is that FORMATS only control display of data values while INFORMATS actually change the values of data. Also, names of INFORMATS are limited to seven characters and that includes the '$' used for character INFORMATS.

...Example 7.21...

```
proc format;
invalue ges                                                          1
140 - 349 = _same_
other     = .
;
invalue $gender                                                      2
'M','F'= _same_
other  = ' '
;
run;

data births;
input
gender $gender1.                                                     3
ges        ges3.
;
datalines;
M999
F210
m245
f349
X350
;
run;

proc print data=births;
run;
```

```
Obs    gender    ges
 1       M         .
 2       F       210
 3               245
 4               349
 5                 .
```

1 A numeric informat is created to read numeric data. Any values in the range 140 through 349 will maintain that value, while values outside that range are converted to missing.
2 A character informat is created to read character data. Any values that are either M or F will maintain that value, while other values are converted to missing. Notice that a period was not used to indicate missing character data. A SPACE was used.
3 The informats are used to read the data. Just as when you use formatted input with SAS-supplied informats, the number at the end of the informat tells SAS how many columns to read in the raw data. What is the length of the variable GENDER? of variable GES?

You have already seen special words that you could use when creating formats (LOW, HIGH, OTHER). The last example introduces another special word that is only used when creating informats, _SAME_. The other thing that differentiates this special word from the others is that it is on the RIGHT side of the ='s, not the left.

Notice that the values of the variables shown in PROC PRINT output. Only the uppercase M and F retained their values. All other values are now missing. You can modify the character informat so it not only reads values of the variable gender, but also converts them to uppercase...

...Example 7.22...
```
proc format;
invalue ges
140 - 349 = _same_
other     = .
;

invalue $gender (upcase)                                                                     1
'M'    = 'M'
'F'    = 'F'
other  = ' '
;
run;

data births;
input
gender $gender1.
ges        ges3.
;
datalines;
M999
F210
m245
f349
X350
;
run;

proc print data=births;
run;

Obs    gender    ges                                                                         2
 1       M         .
 2       F       210
 3       M       245
 4       F       349
 5                 .
```

1 The UPCASE option was added to PROC FORMAT just after the name of the informat. This option tells SAS to convert the character data to uppercase before comparing it to the values on the left of the ='s in the informat.
2 Both the character and numeric data have been converted, but all the values of gender are present.

Another useful feature of informats is the capability of converting character values within raw data into numeric values in a data set. The following example contains grades on several tests. In a few instances, a letter grade is entered instead of a numeric grade. You can write rules to convert the letter grades to numbers as you read the data.

...Example 7.23...
```
proc format;
invalue fixgrade
A = 95
B = 85
C = 75
D = 65
;                                                                                            1
run;

data grades;
input grade : fixgrade. @@;                                                                  2
datalines;
100 C 91 A 82 D C 74                                                                         3
;
run;
```

```
proc print data=grades;
run;

Obs    grade
 1      100                                                                                        4
 2       75
 3       91
 4       95
 5       82
 6       65
 7       75
 8       74
```

1 A numeric informat is created, but notice the values on the left side of the ='s. They are character values. This rule will convert the values on the left to those on the right.
2 List input is used to read the raw data. A numeric variable, GRADE, is created.
3 The raw data contains both numbers and letters.
4 The data set contains only numbers.

Notice that you did not have to list all the numeric grades in the informat, only the letter grades. When SAS uses the informat, it looks at the raw data that is read and then looks at the rules on the left side of the ='s. If there is no rule for a given value, the variable is assigned the value without conversion to some other value. What would have happened in the last example to the observation with the raw data grade value of C if there was no rule to convert C in the informat? What would the value of that observation be in the data set? Do you think there would be any extra messages in the SAS LOG? What normally happens when you try to read non-numeric data with a numeric informats? There are four questions, do you have any answers (that's another question)? Here's another, what's the length of the variable GRADE in the last example? of GENDER in the previous examples where a user-written informat was used to read the data?

PROC FORMAT in example 7.22 produces a message in the LOG file...

```
NOTE: The informat name '@FIXGRADE' exceeds 8 characters. Only the first 8 characters will be used.
```

When INFORMATS are created, an '@' is added to the name when they are stored for future use. Remember that the names of INFORMATS are limited to seven characters, not eight, even though the NOTE in the SAS LOG reminds you that the name exceeds eight characters. The '@' is considered as one of the characters in the name, even though you did not use it.

Informats are very useful if you know the allowable values or allowable ranges of variable values. If your intent is to maintain all allowable values and convert all others to missing, you need not write if-then-else statements. You can convert the data as they are being read by using informats. This task was shown in examples 7.20 and 7.21 where both an allowable range and allowable values were shown in PROC FORMAT. You will not be able to see the values of the bad data in your data set. However, there are many instances where all you want to do is change bad data to missing values.

...FORMAT LIBRARIES (PERMANENT FORMATS)
You have already learned that when you create a temporary SAS data set, it is stored in the WORK library. The location of the work library is dependent on where you are running SAS. On a PC, it is normally a directory on the hard disk. When you create formats via PROC FORMAT, unless you specify otherwise, they are also stored in the WORK library and are temporary. As soon as you end the SAS session, the formats created during the session are erased as are the temporary SAS data sets.

Formats are all stored within one file, not a data set, but a catalog. A catalog is a special type of file that SAS uses to store various types of information (that is not much of an explanation of a SAS catalog, but it is all you really have to know). If you want the formats you create to be saved in a catalog that will be available in subsequent SAS sessions, you must use an option in PROC FORMAT that indicates where to store the permanent format catalog.

...Example 7.24...

```
libname library "i:\";                                                1

proc format library=library;                                          2
value $gender
"1" = "MALE"
"2" = "FEMALE"
;
run;
```

1 The libname LIBRARY is assigned to the directory I:\.
2 The option LIBRARY=LIBRARY is used to tell PROC FORMAT to store the format $gender in the directory I:\.

Without the option LIBRARY=LIBRARY in PROC FORMAT, the SAS LOG would show...

```
NOTE: Format $GENDER has been output.
```

With that option, the SAS LOG shows...

```
NOTE: Format $GENDER has been written to LIBRARY.FORMATS.
```

The libname LIBRARY is special. Whenever you try to use a format is a SAS session, SAS first looks for the format in the WORK library. If it can't find the format in the WORK library, it next looks in the library with the libname LIBRARY (yes, it is a bad choice of names). If SAS can't find the format in either the WORK or LIBRARY libraries, you will get an error message. Since SAS looks in the library LIBRARY by default, most SAS users create their own permanent format libraries in a directory that they later equate to the libref LIBRARY by using a libname statement. You can use PROC FORMAT again either to replace an old format or to put new formats in the permanent format library.

You can also direct storage of formats to other directories. If you do, you must then add that directory to the list that is searched when SAS looks for formats.

...Example 7.25...

```
libname extra "i:\";

proc format;                                                          1
value $gender
'1' = 'MALE'
'2' = 'FEMALE'
;
run;

proc format library=extra;                                            2
value $gender
'1' = 'FEMALE'
'2' = 'MALE'
;
run;

data test;
input gender : $1. @@;
datalines;
1 2 1 2
;
run;
```

```
title 'USING FORMAT FROM LIBRARY=WORK';                                                                    3
proc print data=test;
format gender $gender.;
run;

options fmtsearch=(extra work);                                                                            4

title 'USING FORMAT FROM LIBRARY=EXTRA';                                                                   5
proc print data=test;
format gender $gender.;
run;

USING FORMAT FROM LIBRARY=WORK
Obs    gender
 1     MALE
 2     FEMALE
 3     MALE
 4     FEMALE

USING FORMAT FROM LIBRARY=EXTRA
Obs    gender
 1     FEMALE
 2     MALE
 3     FEMALE
 4     MALE
```

1 A format named $gender is added to the WORK library.
2 A format with the same name is added to the library EXTRA/
3 The format in the WORK library is used to display values of the variable gender.
4 The FMTSEARCH option is used to change the default order that SAS uses to look for formats;
5 The format in the library EXTRA is now used to display values of the variable gender.

If you want to change the default order that SAS uses to search for a formats, you use FMTSEARCH option, for example....

```
options fmtsearch=(extrafmts library work);
```

forces SAS to start looking for a format in the library EXTRA, then the library LIBRARY, and then WORK. That is the reverse of the order you would get if you just specified EXTRA in the FMTSEARCH list.