Chapter 2
Creating Variables Conditionally

Arthur Li

The IF-THEN/ELSE statement
Steps for Creating a Variable

❖ Most of the time, you create a variable that is based on an existing variable

❖ Creating an variable generally consists of the following steps:

1. Evaluate the existing variable
2. Creating the new variable
3. Checking the accuracy of the newly-created variable

Steps for Creating a Variable

- You can use some SAS procedures to evaluate the existing variable

  - PROC PRINT
  - PROC CONTENTS (esp. variable type)
  - PROC MEANS for numerical variable (NMISS option)
  - PROC FREQ for categorical variables (MISSING or MISSPRINT)

Steps for Creating a Variable

- Suppose that you would like to create an variable AGE_HI

  - AGE_HI = 1 for AGE > median (AGE)
  - AGE_HI = 0 for AGE ≤ median (AGE)

Steps for Creating a Variable

❖ Evaluating the AGE variable:

Program 2.1:

```
title 'Evaluate the AGE variable';
proc means data=hearing n nmiss median maxdec=2;
    var age;
run;
```

```
                 Evaluate the AGE variable

                   The MEANS Procedure

                 Analysis Variable : Age

                     N
                 N      Miss          Median
                ----------------------------
                33         1           26.00
                ----------------------------
```

Steps for Creating a Variable

- IF-THEN/ELSE statement:

IF expression THEN statement;
<ELSE statement;>

Program 2.2:

```
data hearing2_1;
    set hearing;
    if age > 26 then age_hi = 1;
    else age_hi = 0;
run;
```

Steps for Creating a Variable

❖ Checking AGE_HI is created correctly:

Program 2.3:

```
title 'Checking AGE_HI is created correctly';
proc means data=hearing2_1 n nmiss min max maxdec=2;
    class age_hi;
    var age;
run;
```

```
                Checking AGE_HI is created correctly

                         The MEANS Procedure

                     Analysis Variable : Age
                   N             N
     age_hi      Obs      N   Miss          Minimum          Maximum
-----------------------------------------------------------------------
          0       18     17      1            15.00            26.00

          1       16     16      0            28.00            36.00
-----------------------------------------------------------------------
```

Handing Missing Values When Creating Variables

- Numerical missing value (.):

```
if age eq . then ...;
```

- A character missing value: a blank space enclosed in either single or double quotation marks

Handing Missing Values When Creating Variables

- The MISSING function:

MISSING(numeric-expression |
character-expression)

- If the argument contains a missing value, the MISSING function will return a value of 1; otherwise, it will return 0.

```
if missing(age) eq 0 then
    if age > 26 then age_hi = 1;
    else age_hi = 0;
```

Handing Missing Values When Creating Variables

Program 2.4:

```
data hearing2_2;
    set hearing;
    if missing(age) eq 0 then
        if age > 26 then age_hi = 1; else age_hi = 0;
run;
title 'Creating AGE_HI considering the missing value';
proc means data=hearing2_2 n nmiss min max maxdec=2;
    class age_hi;
    var age;
run;
```

```
           Creating AGE_HI considering the missing value

                      The MEANS Procedure
                  Analysis Variable : Age
                 N              N
      age_hi    Obs     N    Miss        Minimum        Maximum
---------------------------------------------------------------
           0     17    17       0          15.00          26.00

           1     16    16       0          28.00          36.00
---------------------------------------------------------------
```

Handing Missing Values When Creating Variables

Program 2.5:

```
title 'Use the MISSING option to show missing values';
proc means data=hearing2_2 n nmiss min max maxdec=2 missing;
    class age_hi;
    var age;
run;
```

```
          Use the MISSING option to show missing values

                       The MEANS Procedure
                   Analysis Variable : Age
                    N            N
     age_hi       Obs     N    Miss          Minimum          Maximum
--------------------------------------------------------------------
          .         1     0       1                .                .

          0        17    17       0            15.00            26.00

          1        16    16       0            28.00            36.00
--------------------------------------------------------------------
```

TRUE and FALSE: Logical Expressions

```
if missing(age) eq 0 then
     if age > 26 then age_hi = 1;
     else age_hi = 0;
```

```
if not (missing(age) eq 1) then
     if age > 26 then age_hi = 1;
     else age_hi = 0;
```

- TRUE: numerical values other than 0 or the missing value
- FALSE: the missing value and 0

```
if not missing(age) then
     if age > 26 then age_hi = 1;
     else age_hi = 0;
```

TRUE and FALSE: Logical Expressions

- Assign pregnant women to group “A” and non-pregnant women to group “B”.

Program 2.6:

```
title 'Check PREG variable';
proc freq data=hearing;
    tables preg/missing nocum nopercent;
run;
```

```
              Check PREG variable


              The FREQ Procedure
               Preg    Frequency
               -----------------
                  .           4
                  0          19
                  1          11
```

TRUE and FALSE: Logical Expressions

- Assign pregnant women to group “A” and non-pregnant women to group “B”.

Program 2.6:

```
data hearing2_3;
    set hearing;
    if not missing(preg) then
        if preg then group = "A";
        else group = "B";
run;
```

TRUE and FALSE: Logical Expressions

- Assign pregnant women to group “A” and non-pregnant women to group “B”.

Program 2.6:

```
title 'Check Group is created correctly';
proc freq data=hearing2_3;
    tables preg*group/missing norow nocol nopercent;
run;
```

```
                 Table of Preg by group
        Preg       group

        Frequency,        |A        |B        |  Total
        ---------+---------+---------+---------+
               . |       4 |       0 |       0 |       4
        ---------+---------+---------+---------+
               0 |       0 |       0 |      19 |      19
        ---------+---------+---------+---------+
               1 |       0 |      11 |       0 |      11
        ---------+---------+---------+---------+
        Total           4        11        19        34
```

TRUE and FALSE: Logical Expressions

```
if foo=10 or 20;
```

```
if foo=10 or foo=20;
```

The LENGTH Attribute

- Create a character variable (AGE_CAT) based on the AGE variable:

Program 2.7:

```
data hearing2_4;
    set hearing;
    if not missing(age) then
        if age > 26 then age_cat="old";
        else age_cat="young";
run;

title 'The first 5 observations of HEARING2_4 data set';
proc print data=hearing2_4(obs=5);
    var age age_cat;
run;
```

The LENGTH Attribute

- Create a character variable (AGE_CAT) based on the AGE variable:

```
    The first 5 observations of HEARING2_4 data set

                 Obs     Age     age_cat

                   1      26       you
                   2      26       you
                   3      32       old
                   4      32       old
                   5      34       old
```

The LENGTH Attribute

- The LENGTH statement:

```
LENGTH variable(s) <$> length;
```

- The LENGTH statement must be placed before any other reference to the variable in the DATA step.

The LENGTH Attribute

Program 2.8:

```
data hearing2_5;
    length age_cat $ 5;
    set hearing;
    if not missing(age) then
        if age > 26 then age_cat="old";
        else age_cat="young";
run;

title 'The first 5 observations of HEARING2_5 data set';
proc print data=hearing2_5(obs=5);
    var age age_cat;
run;
```

```
Obs     Age     age_cat


  1      26      young
  2      26      young
  3      32      old
  4      32      old
  5      34      old
```

DO Group

- To execute a group of statements as one unit, use the DO statement:

```
DO;
        SAS statement1
        ...
        SAS statementn
END;
```

- You can nest DO groups within DO groups.
- A DO group is often used within IF-THEN/ELSE statements.

DO Group

- Create two variables:
  - PREG_INFO :
    - ✓ 1 if the PREG variable is not missing
    - ✓ 0 if PREG is missing
  - PREG_SMOKER:
    - ✓ 1 if PREG is not missing & SMOKE = “current”
    - ✓ 0 if PREG is not missing & SMOKE ^= “current”

DO Group

Program 2.9:

```
title 'Frequency Tables: Preg by Smoke';
proc freq data=hearing;
    tables preg*smoke/missing norow nocol nopercent;
run;
```

```
                 Frequency Tables: Preg by Smoke

                          The FREQ Procedure
                        Table of Preg by smoke

      Preg      smoke

      Frequency|        |current |never   |past    |  Total
      ---------+--------+--------+--------+--------+
             . |      0 |      1 |      1 |      2 |      4
      ---------+--------+--------+--------+--------+
             0 |      0 |      6 |      9 |      4 |     19
      ---------+--------+--------+--------+--------+
             1 |      1 |      1 |      8 |      1 |     11
      ---------+--------+--------+--------+--------+
      Total           1        8       18        7       34
```

DO Group

Program 2.9:

```
data hearing2_6;
    set hearing;
    if not missing(preg) then
    do;
        preg_info = 1;
        if smoke ="current" and preg = 1 then preg_smoker = 1;
        else preg_smoker = 0;
    end;
    else preg_info = 0;
run;

title 'Check if PREG_SMOKER and PREG_INFO are created
correctly';
proc freq data=hearing2_6;
    tables preg_smoker preg_info /missprint;
run;
```

DO Group

Program 2.9:

```
   Check if PREG_SMOKER and PREG_INFO are created correctly

                     The FREQ Procedure

                                          Cumulative    Cumulative
preg_smoker    Frequency     Percent       Frequency       Percent
-----------------------------------------------------------------
          .           4           .               .             .
          0          29       96.67              29         96.67
          1           1        3.33              30        100.00

                    Frequency Missing = 4


                                          Cumulative    Cumulative
 preg_info     Frequency     Percent       Frequency       Percent
 --------------------------------------------------------------
          0           4       11.76               4         11.76
          1          30       88.24              34        100.00
```

Executing One of Several Statements
Multiple IF-THEN/ELSE statements

❖ Multiple IF-THEN/ELSE statements

IF expression **THEN** statement;
ELSE IF expression **THEN** statement;
<...
ELSE IF expression **THEN** statement;
<ELSE statement;>>

Multiple IF-THEN/ELSE statements

- Create a variable AGEGROUP:
    - AGEGROUP = 1 for AGE ≤ 20
    - AGEGROUP = 2 for 20 < AGE ≤ 30
    - AGEGROUP = 3 for AGE > 30

Multiple IF-THEN/ELSE statements

Program 2.10:

```
data hearing2_7;
    set hearing;

    *method1;
    if age > 30 then agegroup1 = 3;
    else if age > 20 then agegroup1 =2;
    else if age > . then agegroup1=1;

    *method2;
    if not missing(age) then
        if age <=20 then agegroup2 = 1;
        else if age <= 30 then agegroup2 = 2;
        else agegroup2= 3;
run;
```

Threshold values in descending order

Threshold values in ascending order

Multiple IF-THEN/ELSE statements

Program 2.10:

```
title 'Check AGEGROUP1 is created correctly';
proc means data=hearing2_7 missing n nmiss min max maxdec=2;
    class agegroup1;
    var age;
run;

title 'Check AGEGROUP2 is created correctly';
proc means data=hearing2_7 missing n nmiss min max maxdec=2;
    class agegroup2;
    var age;
run;
```

Multiple IF-THEN/ELSE statements

Program 2.10:

```
                 Check AGEGROUP1 is created correctly

                        The MEANS Procedure

                      Analysis Variable : Age

                     N              N
    agegroup1      Obs      N    Miss          Minimum          Maximum
  -----------------------------------------------------------------------
            .        1      0       1                .                .

            1       10     10       0            15.00            20.00

            2       12     12       0            23.00            30.00

            3       11     11       0            31.00            36.00
  -----------------------------------------------------------------------
```

Multiple IF-THEN/ELSE statements

Program 2.11:

```
data hearing2_8;
    set hearing;
    length trial $4;
    if preg = 1 then do;
        trial = "A";
        requireInfo = 0;
    end;
    else if preg = 0 then do;
        trial = "B";
        requireInfo = 0;
    end;
    else do;
        trial = "Wait";
        requireInfo = 1;
    end;
run;

title 'Checking if TRIAL and REQUIREINFO are created correctly';
proc freq data=hearing2_8;
    tables (trial requireInfo)*preg/missing nocol norow nopercent;
run;
```

Creates two variables: TRIAL and REQUIREINFO, based on the variable PREG:.

Executing Statements Using the SELECT Group

❖ The SELECT group:

Begin SELECT group

```
SELECT <(select-expression)>;
      WHEN-1 (when-expression-1
         <..., when-expression-n>) statement;
<... WHEN-n (when-expression-1
         <..., when-expression-n>) statement;>
    <OTHERWISE statement;>
END;
```

End SELECT group

Executing Statements Using the SELECT Group

- The SELECT group:

```
SELECT <(select-expression)>;
      WHEN-1 (when-expression-1
         <..., when-expression-n>) statement;
<... WHEN-n (when-expression-1
         <..., when-expression-n>) statement;>
    <OTHERWISE statement;>
END;
```

Any SAS expression that can be evaluated into a single value

Executing Statements Using the SELECT Group

❖ The SELECT group:

When a select-expression is specified, …

```
SELECT <(select-expression)>;
      WHEN-1 (when-expression-1
         <..., when-expression-n>) statement;
<... WHEN-n (when-expression-1
         <..., when-expression-n>) statement;>
    <OTHERWISE statement;>
END;
```

- ❑ SAS compares the results from select-expression and when-expression and returns a value of TRUE or FALSE.
- ❑ If it is TRUE for a WHEN statement, the corresponding statement is executed;
- ❑ If it is FALSE, a comparison is performed for either the next when-expression within the current WHEN statement or the one in the next WHEN statement.

Executing Statements Using the SELECT Group

❖ The SELECT group:

When a select-expression is specified, …

```
SELECT <(select-expression)>;
      WHEN-1 (when-expression-1
         <..., when-expression-n>) statement;
<... WHEN-n (when-expression-1
         <..., when-expression-n>) statement;>
    <OTHERWISE statement;>
END;
```

- ❑ If there is no WHEN-condition that is TRUE, the OTHERWISE statement is executed if one exists.
- ❑ If there is no OTHERWISE statement, SAS will issue an error message and terminate DATA step execution.
- ❑ If the comparison is TRUE for more than one WHEN statement, only the first WHEN statement is executed.

Executing Statements Using the SELECT Group

❖ The SELECT group:

When a select-expression is NOT specified,

```
SELECT <(~~select-expression~~)>;
      WHEN-1 (when-expression-1
         <..., when-expression-n>) statement;
<... WHEN-n (when-expression-1
         <..., when-expression-n>) statement;>
    <OTHERWISE statement;>
END;
```

- ❑ only the when-expression is evaluated and generates a value of TRUE or FALSE.
- ❑ If it is TRUE for a WHEN statement, the corresponding statement is executed.

Executing Statements Using the SELECT Group

Program 2.12:

```
data hearing2_9;
    set hearing;
    length ethnic $ 10;
    select (race);
        when ("W", "H") ethnic = "white";
        when ("B", "A") ethnic = "non-white";
    end;
    select (preg);
        when (1) do;
            trial = "A";
            drug = "Treatment";
        end;
        when (0) do;
            trial = "B";
            drug = "placebo";
        end;
        otherwise;
    end;
    ...
```

If the result of all SELECT-WHEN comparisons is false and no OTHERWISE statement is present, SAS will issue an error message

Executing Statements Using the SELECT Group

Program 2.12:

```
data hearing2_9;
     set hearing;
     length ethnic $ 10;
     select (race);
          when ("W", "H") ethnic = "white";
          when ("B", "A") ethnic = "non-white";
     end;
     select (preg);
          when (1) do;
               trial = "A";
               drug = "Treatment";
          end;
          when (0) do;
               trial = "B";
               drug = "placebo";
          end;
          otherwise;
     end;
     ...
```

DO group

OTHERWISE +null statement This means that if PREG is other than 1 or 0, the TRIAL and DRUG variables will be assigned missing values

Executing Statements Using the SELECT Group

Program 2.12 (continue):

```
    ...
    select(hearing);
        when ("yes") group = 1;
        when ("no") group = 2;
        otherwise group = 3;
    end;
    select;
        when (income > 100000) highincome = 1;
        when (income > .) highincome = 0;
        otherwise;
    end;
run;
```

SELECT-EXPRESSION is not used

Modifying the IF-THEN/ELSE Statement with the Assignment Statement

❖ The assignment statement:

```
variable=expression;
```

```
if age>26 then age_hi = 1;
else age_hi = 0;
```

```
age_hi = age>26;
```

Modifying the IF-THEN/ELSE Statement with the Assignment Statement

- The assignment statement:

```
variable=expression;
```

```
if age>30 then agegroup = 3;
else if age>20 then agegroup = 2;
else if age>. then agegroup = 1;
```

```
agegroup =(age>.)+(age>20)+(age>30);
```

Modifying the IF-THEN/ELSE Statement with the Assignment Statement

Program 2.13:

```
data hearing2_10;
    set hearing;
    agegroup = (age>.) + (age>20) + (age>30);
run;

title 'Check if AGEGROUP is created correctly';
proc means data=hearing2_10 n nmiss min max maxdec=2 missing;
    class agegroup;
    var age;
run;
```

Modifying the IF-THEN/ELSE Statement with the Assignment Statement

Program 2.13:

```
               Check if AGEGROUP is created correctly

                           The MEANS Procedure

                        Analysis Variable : Age

                      N                  N
     agegroup       Obs        N      Miss              Minimum              Maximum
 --------------------------------------------------------------------------------------
            0         1        0         1                    .                    .

            1        10       10         0                15.00                20.00

            2        12       12         0                23.00                30.00

            3        11       11         0                31.00                36.00

 --------------------------------------------------------------------------------------
```