Working with Data

IF – THEN Statements

Some time, you want an assignment statement to apply to some observations under some conditions, but not others. We call it conditional logic, you need to use IF-THEN statements.

IF condition THEN action;

Working with Data

```
IF Age>=25 THEN Group = '25+';

IF year < 1975 and Model = 'Mustang' THEN Status =
'Classic';

IF Model = 'Miata' THEN DO;
        Make = 'Mazda';
            Seat = 2;
                        END;
```

Working with Data

I want to create a categorical variable name “Cat1” which includes “age” and “income” variables.

Cat1 = 1 if age less than 30 years old and income Less or equal $30,000;

Cat1 = 2 if age less than 30 years old and income More than $30,000;

Cat1 = 3 if age>=30 and income <= 30000;

Cat1 = 4 if age>=30 and income >30000.

Working with Data

if Age<**30** and income<=**30000** then cat1=**1**;

if Age<**30** and income>**30000** then cat1=**2**;

if age>=**30** and income<=**30000** then cat1=**3**;

if age>=**30** and income>**30000** then cat1=**4**;

Working with Data

** have to use IF THEN/ELSE**;

if Age<**30** and income<=**30000** then cat2=**1**;

else if Age<**30** and income<=**70000** then cat2=**2**;* any age < 30 and income 30000-70000 ; without "else" cat2=2 will replace cat2=1;

else if age>=**30** and income<=**30000** then cat2=**3**;

else if age>=**30** and income<=**70000** then cat2=**4**;

Working with Data

IF THEN; DO END;

```
if age<30 then do;
        if income<=30000 then cat2=1;
        if income>30000 then cat2=2;
        end;

else if age>=30 then do;
        if income<=30000 then cat2=3;
        if income>30000 then cat2=4;
        end;
```

Working with Data

Missing values

- For numeric data, a missing value is designed as a .

  Missing value is always smallest value. Missing value is always smaller than any number including 0 and negative number. The order is . < -n < 0 < +n.

```
IF .<height <60 THEN ……
```

- For character, a missing value is designed as ‘ ’ or “ ”

```
IF class =“ ” and Score<4 THEN Group = “Score<4 with missing class ”;
```

Working with Data

```
DATA ageunder 20;
SET allages;
        IF   . <age <20;* excluding the missing value;
        IF    .< diff < - 4; * excluding the missing value;
```

Working with Data

Creating and Redefining Variables. SAS is very flexible and uses a common sense approach to these tasks. You can create and redefine variables with assignment statements using this basic form:

Variable = expression;

BMI = (Weight***0.45**)/(height * **0.025**)****2**;

"+" – addition; "-" – subtraction; "/" – division; "*" – multiplication;

"**" – exponentiation; and more ……

More functions shows in the table – Selected SAS Function.

Subsetting SAS Data

If you want to use some of the observations in a data set and exclude the rest, The most common way is to do this is with a subsetting IF statement in a DATA step.

```
DATA all;
SET subdata;
IF  Conditions;
```

Subsetting SAS Data

If you don’t like to use the Subsetting IF statement, there is another alternative way, The DELETE statement. DELETE statement do the opposite of subsetting IF statement. While the subseting IF statement tells SAS which observation to include, the DALETE statement tells SAS which observations to exclude.

```
DATA all;
SET subset;
IF Conditions THEN DELETE;
```

Using First.*variable* and Last.*variable* Statements to sebsetting data

In addition to the variables you create in your SAS data set, SAS create a few mores automatic variables. You don't see these variables because they are temporary and not saved with your data, but you can use them as any other variables in your data set in the DATA step.

When you are using BY statement in DATA step, The First.*variable* and Last.*variable* are automatic created. The first occurrence of a new value for that variable will have a value of 1, and 0 for others. The Last occurrence of a new value for that variable will have a value of 1, and 0 for others.

SAS Functions

SAS system provides an extensive library of "built-in" function. We will select some character handling and numeric functions.

Function Syntax:

Function-name (argument-1, argument2);

**Where the arguments can be: constants, variables, expressions, other functions*

SAS Functions

Numeric functions:

- *Numbers:*

INT, LOG, LOG10, MAX, Mean, MIN, ROUND, SUM, SQRT, and other such as:

LAG, DIF: The DATA step provides two functions, LAG and DIF, for accessing previous values of a variable or expression. These functions are useful for computing lags and differences of series.

- *Dates:*

DAY, MDY, MONTH, QTR, TODAY

Day=DAY(DateDue);

Month=MONTH(DateDue);

Quarter=QTR (DateDue);

DaysOverDue=Today() - DateDue;

DateClose=MDY(**9**,**6**,**2010**);

SAS Functions

Character functions:

- ||: Combine two characters into one. a='Three', b='Months';
  x=a||b; x='Three Months';

- LEFT: Left aligns a SAS character expression a=' happy'; x=LEFT(a); x='happy'

- LENGTH: a="happy"; x=length (a); x=5; * not count trailing blank.

- SUBSTR: SUBSTR (arg, position, n), a='Today is a Happy Day'; x=substr(a,11,9);
  x='Happy Day';

- TRIM: Trim(arg) removes trailing blanks from character expression
  A=' Happy'; b='Life'; x=trim(a)||b; x='Happy Life';

- UPCASE: a='MyLife'; x=UPCASE(a); x=MYLIFE;

SAS Functions

Character functions – Cont.:

- TRANSLATE: a=’my cat can’; x=(a, ‘r’, ‘c’); x=my rat ran’;

- COMPBL: Converting multiple blanks to single blank.

  Address = ’ 799 Locks Way, Augusta, GA, 30907, USA ’;

  CorrectedAddress = compbl(address);

  CorrectedAdress=’799 Locks Way, Augusta, GA, 30907, USA’;

- COMPRESS: remove any space or specific characters from a string.

  Phone = (201) 555-77 99; phone1=compress(phone); phine2=compress(phone, ‘(-)’;

  Phone1=(201)555-7799, phone2=2015557799

- SCAN: Extract part characters (e.g. last name) from your character variable,

  SCAN (char_var, n, ‘list of delimiters’);

Return the *n*th “word” from the char_var, where a “word” is defined as anything between two delimiters. If n is negative, the scan will proceed from right to lift.