

DO - END GROUPS

A plain **IF - THEN** statement allows for only one result to occur when the **IF** condition is true.

Notice what happens in the following.

EXAMPLE 1

SAS Code:

```
DATA EXP1;  
  INFILE 'A:\EXP.DAT';  
  INPUT SOIL $ TRT COUNT1 COUNT2;  
  
  IF TRT = 2 THEN NEW = COUNT1 + 5;  
                  TOTAL = COUNT1 + COUNT2;  
                  RATIO = COUNT1/COUNT2;
```

Here the intent is to calculate some new variables (NEW, TOTAL, & RATIO) only for treatment 2.

However, the result of this code will calculate TOTAL and RATIO for all treatments. Grouping all 3 requests so that they will be done only when TRT=2, requires a set of **DO - END** statements. Think of these as parentheses that bound groups of statements.

EXAMPLE 2

SAS Code:

```
DATA EXP1;  
  INFILE 'A:\EXP.DAT';  
  INPUT SOIL $ TRT COUNT1 COUNT2;  
  
  IF TRT = 2 THEN DO;  
    NEW = COUNT1 + 5;  
    TOTAL = COUNT1 + COUNT2;  
    RATIO = COUNT1/COUNT2;  
  END;
```

ITERATIVE DO - END GROUPS

There are occasions when we need to do some operations again and again, iteratively. A special form of the **DO - END** structure exists for this.

EXAMPLE 3

SAS Code:

```
DATA RANDNUM;  
  DO INDEX = 1 TO 100;  
    RAND = RANUNI(0);  
    OUTPUT;  
  END;  
  .  
  .  
  .  
more code
```

This code will generate 100 uniform random numbers using the SAS function RANUNI(.). The idea is to get random numbers repeatedly, 100 times. This is done by creating a new variable, **INDEX** (any valid name will do), which is initially set equal to 1. A random number is then generated and output.

INDEX is incremented to 2 and another random number is generated and output. This process is continued until **INDEX** passes 100, where the 'loop' is stopped and SAS continues with the code following the loop. Note that there are other similar forms of this type of conditional **DO - END** structure such as **DO - WHILE** and **DO - UNTIL**.

Suppose we have plant weights taken sequentially over time (repeated measures). Each time period has been entered as a separate variable. The data file has 11 variables:

ITERATIVE DO - END GROUPS & ARRAYS (cont.)

Plant	wt1	wt2	wt3	wt10	
1	13.8840	14.197	43.4681	...	125.594
2	23.0968	3.119	55.2602	...	138.973
3	19.9821	22.066	11.1744	...	156.626
4	23.2994	64.761	47.2603	...	133.944
.
.
.

However, to work with the data, we will need just 3 variables, **plant**, **time** and **weight**. To do this we need to reformat the data such that rows become columns. While this could be done by hand, a quicker method is to use the SAS **ARRAY** statement.

An **ARRAY** in SAS is nothing more than a **list** of variables. Once we have this list, the iterative **DO - END** structure can be used to point to positions within the list.

EXAMPLE 4

SAS Code:

```
DATA PLANTS;  
  INFILE 'A:\GROWTH.DAT';  
  INPUT PLANT WT1 WT2 WT3 ... WT10 FIRSTOBS=2;  
  ARRAY WTS{10} WT1 - WT10;  
  DO TIME = 1 TO 10;  
    WEIGHT = WTS{TIME};  
    OUTPUT;  
  END;
```

ITERATIVE DO - END GROUPS & ARRAYS (cont.)

When the data is read in, a list (ARRAY) called **WTS** is setup. This can be any valid name that is not already used in the dataset. The number of items in the list is given in the braces ({ & }). Following this the items to be placed in the list are given in the order they should appear. In this case, the variables WT1 through WT10 are given in chronological order. (Notice that this is abbreviated as WT1 - WT10). It is called an indexed list and can be used this way when the variable names follow a sequential numbering scheme such as WT1, WT2, WT3, etc. The **DO - END** loop creates a variable called **TIME** which is initialized to 1. The loop itself then creates another new variable, **WEIGHT**, equal to the first member of the list **WTS{1}**. An explicit output is then used to put these new values, as well as the original data, into the dataset. **TIME** is then incremented to 2 and the 2nd member of the list is set to **WEIGHT** and output.

This continues until the 10th member of the list is output after which the next line is read from the data file and then the process is repeated. Notice that each line of the data file produces 10 observations in the SAS dataset. If you had 20 plants to begin with, the dataset should end up with 200 observations. Always check the SAS LOG screen or print out the data to insure that things are working as you intended!