SAS – USING ARRAYS – A FIRST EXAMPLE

Often it is necessary to perform a similar operation on several variables within an observation. Arrays are used in the SAS data step to reduce the amount of code that has to be written to accomplish these types of tasks. In this example a number of variables contain information measured in miles, and the desire is to convert this information to kilometres. An array is used to reduce the amount of code that needs to be written to accomplish this task.

In the example below: the SAS dataset **distancesInMiles** has information on the distance measured in miles that people drive to work (variable: distanceToWork), the distance people drive to a park (variable: distanceToPark), and the distance people drive to shop (variable: distanceToShop). The program creates a new dataset, **distancesInKM**, which contains this information in kilometres.

This program illustrates how to convert the value of all three variables (distanceToWork, distanceToPark, distanceToShop) into kilometres with minimal coding by using arrays and a DO loop.

Please refer to the "Code Notes" section below for an explanation of what the code is doing in those sections identified by a number in a **red-filled** circle.

```
/* Using arrays to reduce coding */

data distancesInMiles ;
      input distanceToWork distanceToPark distanceToShop ;
      datalines ;
      23.4 0.9 2.4
      11.2 35.0 41.2
      0.5 14.2 6.8
      ;
run ;
```

Step 1:
Create the SAS dataset:
distancesInMiles

```
proc print data = distancesInMiles ;
      title "Distances in Miles " ;
run ;
```

Step 2:
Print the dataset to see the distances as they are recorded in miles

SAS – USING ARRAYS – A FIRST EXAMPLE

1, 2, 3

```
data distancesInKM ;
     set distancesInMiles;
     array howFar distanceToWork distanceToPark distanceToShop ;

     /* convert all distances to kilometres */

     do index = 1 to 3 ;
           howFar(index) = howFar(index) * 1.609 ;
     end ;

     drop index ;
run ;
```

Step 3:
Create a new dataset, **distancesInKM**, that has all the data from the dataset **distancesInMiles**, but the values are converted to kilometres.

```
proc print data = distancesInKM ;
     title "Distances in Kilometres" ;
run ;
```

Step 4:
Print the dataset **distancesInKM** to see the distances as they have been converted to kilometres.

Output produced by this program's PROCS (PROCEDURES):

```
                 Distances in Miles

               distance     distance     distance
       Obs      ToWork       ToPark       ToShop

        1        23.4          0.9          2.4
        2        11.2         35.0         41.2
        3         0.5         14.2          6.8
```

Output produced by Step 2
(Print the dataset to see the distances as they are recorded in miles.)

SAS – USING ARRAYS – A FIRST EXAMPLE

```
                 Distances in Kilometres

              distance     distance     distance
       Obs     ToWork       ToPark       ToShop

        1     37.6506       1.4481       3.8616
        2     18.0208      56.3150      66.2908
        3      0.8045      22.8478      10.9412
```

Output produced by Step 4 (Print the dataset **distancesInKM** to see the distances as they have been converted to kilometres.)

Code Notes: (explanation of what the code is doing in those areas identified by a number in a red-filled circle)

1. The ARRAY statement does two things: (1) it provides one name for a group of variables (in this case the group will be called: **howFar**), and (2) it indicates what variables will be in the group known as howFAR (in this case there will be three variables in the group: distanceToWork, distanceToPark, and distanceToShop).

   Note that all variables in an array must be of the same type – in this case they are all numeric.

   Once the ARRAY statement is executed, SAS will understand that **howFAR(1)** (spoken as the *1st variable in the group howFAR*) is the variable distanceToWork, while **howFar(2)** (spoken as the *2nd variable in the group howFar*) is the variable distanceToPark, and **howFar(3)** (spoken as the *3rd variable in the group howFar*) is the variable distanceToShop. This understanding is what enables the coding efficiency described in Code Note 2 below.

2. The DO loop causes the statement: **`howFAR(index) = howFar(index) * 1.609 ;`** to be executed 3 times, with the value of the variable **index** incrementing each time, starting with an initial value of 1.

   For example, the first time through the DO loop the statement is understood by SAS to be:
   `howFAR(1) = howFar(1) * 1.609 ;`

   The second time through the DO loop the statement is understood by SAS to be:
   `howFAR(2) = howFar(2) * 1.609 ;`

The third time through the DO loop the statement is understood by SAS to be:

```
     howFAR(3) = howFar(3) * 1.609 ;
```

The DO loop code block is thus equivalent to these three SAS statements:

```
distanceToWork = distanceToWork *  1.609 ;
distanceToPark = distanceToPark *  1.609 ;
distanceToShop = distanceToShop * 1.609 ;
```

3. The DROP statement just tells SAS not to keep the variable **index** in the dataset that is being created.

Tips:

- The stopping value of the DO loop index cannot be greater than the number of variables in the group (there are 3 variables in the group howFAR in this case – hence the 3 in the statement: `DO INDEX = 1 to 3 ;`). In this example, trying to reference howFAR(4) (spoken as the *4th variable in the group howFar*) would generate an error and the data step would stop executing.

- Always examine the SAS log for any error messages or warnings.