Arrays and DO loops

Shannon Pileggi

STAT 330

OUTLINE

Overview

- `DO` loops can be used to perform a series of statements on an observation any number of times
- they can be handy when creating a data set from scratch or calculating something that happens at regular intervals (ex. annual interest rates)
- Some details:
  - `DO` loops always need to end with the statement `END;`
  - There is an implicit *output* at the end of each data step. If you want to create an observation for each iteration, place `OUTPUT;` inside the `DO` loop.

OUTPUT statement

SAS uses an **implicit** output statement at the **end** of the data step **only** if the data step does not contain the word output. The two codes below are equivalent:

SAS Code

```
DATA example1 ;
    DO i = 1 TO 4 ;
    END ;
RUN ;
```

SAS Code

SAS Code

```
DATA example2 ;
    DO i = 1 TO 4 ;
    END ;
    OUTPUT ;
RUN ;
```

SAS Code

OUTPUT statement

The default position for the **implicit** output statement always right before the RUN; statement - but we can move it!

SAS Code

```
DATA example2 ;
    DO i = 1 TO 4 ;
    END ;
    OUTPUT ;
RUN ;
```

SAS Code

Obs	i
1	5

SAS Code

```
DATA example3 ;
    DO i = 1 TO 4 ;
        OUTPUT ;
    END ;
RUN ;
```

SAS Code

Obs	i
1	1
2	2
3	3
4	4

Index variables

- In the previous example, we used `i` as the *index* variable.
- Index variables don't have to be used in the inner `DO/END` block, but SAS does keep track of its values.
- By default, SAS will increase the index variable by $+1$
- The index variable is increased by the default value at the *bottom* of each loop iteration when SAS encounters the `END;` statement.
- So, at the termination of the loop, the value of the index variable is one increment beyond the stop value.

On your own:

How could you modify the `example3` code to show each iteration of i and the last value of i?

SAS Code

```
DATA example2 ;
    DO i = 1 TO 4 ;
    END ;
    OUTPUT ;
RUN ;
```

SAS Code

Obs	i
1	5

SAS Code

```
DATA example3 ;
    DO i = 1 TO 4 ;
        OUTPUT ;
    END ;
RUN ;
```

SAS Code

Obs	i
1	1
2	2
3	3
4	4

Changing the Loop Increment/Decrement Method

```
do i=0 to 10 by 2;     i=0,2,4,6,8,10
do j=1 to 10 by 2;     j=1,3,5,7,9
do k=1 to 10 by .1;    k=1,1.1,1.2,...,9.9,10
do l=10 to 1 by -1;    l=10,9,8,...,1
do m=2,6,9,11,22;      specify nonstandard index values
do n="A","B","C";      specify character index values
do o=var1,var2,var3;   specify variables
```

What are the final values of the index variables?

A. `DO i = 1 TO 5; ... END;`

B. `DO j = 2 TO 8 by 2; ... END;`

C. `DO k = 10 TO 2 by -2; ... END;`

Nested do loops

SAS Code

```
DATA example5 ;
   DO i = 1 TO 3;
      DO j = "A","B" ;
         x + 1 ;
         OUTPUT ;
      END ;
   END ;
RUN ;
```

SAS Code

Variables Table for DO Loop		
i	j	x
1	A	1
1	B	2
2	A	3
2	B	4
3	A	5
3	B	6

On your own: Predict the output that you would see if you moved the OUTPUT statement to after the first `END;`.

DO loops

DO UNTIL/WHILE

Modified `DO` Loop: DO UNTIL

- `DO UNTIL` is a special loop that **breaks** the iteration process when a *condition* is met
- **SAS Syntax:** `DO UNTIL (`*`condition`*`);`
- Helpful when the number of iterations is not known ahead of time
- **Important note:** SAS checks on the status of the condition **at the bottom** of the loop (when it encounters the `END;` statement).

Which of the following is TRUE?

1. A `DO UNTIL` loop always executes at least once
2. A `DO UNTIL` may never execute

Modified `Do` Loop: DO WHILE

- `DO WHILE` is a special loop that **continues** the iteration process as long as a *condition* is met
- **SAS Syntax:** `do while (condition);`
- Helpful when the number of iterations is not known ahead of time
- **Important note:** SAS checks on the status of the condition **at the start** of the loop.

Which of the following is TRUE?

1. A `DO WHILE` loop always executes at least once
2. A `DO WHILE` may never execute

Example Code

Suppose my annual income is $60,000, and I expect it to increase by 2% every year, and I plan to save 10% of my annual income each year. How many years until I save $500,000?

SAS Code

```
DATA retire ;
        savings = 0 ;
        income = 60000 ;
        year = 0 ;

        DO UNTIL (savings >= 500000) ;
    /*---------------------------*/
    /*    WORKS EQUIVALENTLY     */
    /*DO WHILE (savings < 500000);*/
    /*---------------------------*/
                year = year + 1 ;
                savings = savings + income*.10 ;
                income = income + income*.02 ;
                OUTPUT ;
        END ;
RUN ;
```

SAS Code

On your own:

SAS Code

```
DATA example6 ;
   x = 15 ;
   DO WHILE(x > 12) ;
      x + 1 ;
   END ;
RUN ;
```

SAS Code

What is the value of x at the completion of this DATA step?

1. 12
2. 15
3. 16
4. this loop executes infinitely

On your own:

SAS Code

```
DATA example7 ;
   x = 0 ;
   /*ENTER LINE HERE*/
      x = x + 1 ;
      x_sq = x**2 ;
      OUTPUT ;
   END ;
RUN ;
```

SAS Code

Suppose I wanted to generate the sequence of numbers $1^2, 2^2, 3^2, 4^2$. Which line of code would achieve this?

1. `DO UNTIL(x < 4);`
2. `DO WHILE (x < 4);`
3. `DO UNTIL(x = 5);`
4. `DO WHILE (x = 5);`

DO loops

DO UNTIL/WHILE

Data

SAS Code

```
DATA grades;
   INPUT name $ exam1 exam2 exam3;
   DATALINES;
   Shannon      96    82    83
   Lex          92    81    68
   Becky        92    75    73
   Lora         94    65    70
   Susan        91    77    85
   Hunter       76    72    86
   Ulric        98    71    80
   Richann      90    60    60
   Tim          97    94   100
   Ronald        .    77    60
   ;
RUN;
```

SAS Code

Motivating Example

An *inefficient* way to convert all test scores to letter grades:

SAS Code

```
DATA grades2 ;
        SET grades ;
        *FOR EXAM1;
        IF exam1 = . THEN letter1 = " " ;
        ELSE IF exam1 <60 THEN letter1 = "F";
        ELSE IF 60 <= exam1 < 70 THEN letter1 = "D";
        ELSE IF 70 <= exam1 < 80 THEN letter1 = "C";
        ELSE IF 80 <= exam1 < 90 THEN letter1 = "B";
        ELSE IF 90 <= exam1 THEN letter1 = "A";
        /*Repeat code chunk for exam2*/
        /*Repeat code chunk for exam3*/
RUN ;
```

SAS Code

About arrays

- **Purpose of SAS Arrays:** An array is a temporary grouping of variables under a **single name**.
  - must be all character or all numeric
  - can be existing variables or new variables that you would like to create
- Helps reduce the number of required statements to process variables
- Can simplify the maintenance of DATA step programs
- NOTE: Arrays only exist during the data step, although the variables they work with may be a part of the data set.

Defining an Array

- **SAS Syntax:** `ARRAY` *`array_name`* `(dimension)` *`elements`*`;`
- The name of the array must not be a SAS keyword or an existing variable

```
array scores (3) exam1 exam2 exam3 ;
```

- The array `scores` contains the variables `exam1`, `exam2`, and `exam3`.
- To reference a variable using an array call the array name and then the appropriate subscript,

  i. e., `scores(2)` refers to `exam2`

Example code

An *efficient* way to convert all test scores to letter grades:

SAS Code

```
DATA grades2 ;
   SET grades ;

   ARRAY scores (*) exam: ;

   ARRAY letters (3) $ ;

   DO i = 1 TO DIM(SCORES) ;
      IF scores(i) = . THEN letters(i) = " " ;
      ELSE IF scores(i) <60 THEN letters(i) = "F";
      ELSE IF 60 <= scores(i) < 70 THEN letters(i) = "D";
      ELSE IF 70 <= scores(i) < 80 THEN letters(i) = "C";
      ELSE IF 80 <= scores(i) < 90 THEN letters(i) = "B";
      ELSE IF 90 <= scores(i) THEN letters(i) = "A";
   END ;
RUN ;
```

SAS Code

Shortcuts for listing variables

- Numbered range lists are for variables that start with the same characters and end with consecutive numbering
  - `Var6 Var7 Var8` is the same as `Var6-Var8`
- Name range lists depend on the position of the variables in the data set
  - `Cat Cow Pig Dog` is the same as `Cat -- Dog` (Assuming that this is the position order in the data set)
  - How can you check the order? Use `PROC contents`!
- You can use these shortcuts inside of functions
  - e.g., `X = mean(of Var1-Var5)`
- When variables have the same prefix, you can also call all variables with that prefix
  - e.g., `Y = sum(of Var:)`

```
array assignments (?)  hw1 hw2 hw3 hw4;
```

What belongs within the parentheses of this array statement?

1. `hoemwork`
2. `homework*`
3. `1-4`
4. `4`

A more complex array

SAS Code

```
DATA grades2 ;
   SET grades ;

   ARRAY scores (*) exam: ;

   ARRAY letters (3) $ ;

   ARRAY letter_values (6) $ (" " "F" "D" "C" "B" "A");

   ARRAY grcuts (6) (0 60 70 80 90 100);

   DO i = 1 TO DIM(SCORES) ;

   DO j = 2 to 6 ;
       IF grcuts(j-1) <= scores(i) <= grcuts(j)
       THEN letters(i) = letter_values(j) ;
   END ;
   END ;
RUN ;
```