Chapter 15 GENERATING DATA WITH DO LOOPS

CONSTRUCTING DO LOOPS

A **DO** loop is used to perform repetitive calculations in the DATA step. **DO** loops make DATA steps concise and transparent. The syntax is

data *dataset\_name*;

do *index\_variable*=*start* to *stop* by *increment*;

SAS statements;

end;

run;

where

* the *index\_variable* stores the value of the current iteration of the Do loop,
* *start* specifies the initial value of *index\_variable*,
* *stop* is the last value that executes the DO loop,
* *increment* specifies an increment value for *index\_variable*.
* Typically *increment*=1, and the BY statement may be omitted in this case.

Example. The program below calculates how much interest was earned each month for a one-year investment.

data earnings;

amount=1000;

rate=0.075/12;

do month=1 to 12;

earned+(amount+earned)\*rate;

end;

run;

proc print noobs;

run;

The variable amount is the dollar amount of the initial investment. The annual interest rate is assumed 7.5% compounded monthly, so the variable rate=0.075/12 represents the interest rate per month. The variable earned is the cumulative interest amount earned.

* Note the syntax for iterative calculation of the variable earned. The general syntax is

*variable\_name* + expression

* At the beginning of the DATA step processing, the data file is created, and the values of amount and rate are assigned. Next, the DO loop starts executing, recalculating the value of earned, and incrementing the value of month. After the last (12th) execution of the DO loop, the index variable month is incremented to 13. Because 13 exceeds the stop value of the index variable, the DO loop terminates. The final value of month=13 is stored in the data file.

The final data set has the form

amount rate month earned

1000 .00625 13 77.6326

The calculated amount of interest earned during a one-year period is $77.63.

* The increment amount is allowed to be negative. Then the starting value must exceed the stopping value. For example,

do count=10 to 0 by -2;

SAS statements;

end;

* It is possible to specify how many times a DO loop executes by listing items in a series. The syntax is

do *index\_variable*=*value1*, *value2*, …;

SAS statements;

end;

To list the values in the series, they must be specified either

* + All numeric, for example, do count=1,2,3,5,8,13,21;
  + All character, for example, do month='Jan','Apr','July','Oct';
  + All variable names (either all numeric variables or all character variables), for example, do quiz=quiz1,quiz2,quiz5,quiz7;

USING OUTPUT STATEMENT IN DO LOOP

To create an observation for each iteration of the DO loop, place an OUTPUT statement inside the DO loop before the END statement. The created data set then contains the values of all variables for each iteration of the DO loop.

Example. In the earnings example, the OUTPUT statement is inserted to create a complete output for all iterations.

data earnings;

amount=1000;

rate=0.075/12;

do month=1 to 12;

earned+(amount+earned)\*rate;

output;

end;

run;

proc print noobs;

run;

The output is

amount rate month earned

1000 .00625 1 6.2500

1000 .00625 2 12.5391

1000 .00625 3 18.8674

1000 .00625 4 25.2354

1000 .00625 5 31.6431

1000 .00625 6 38.0908

1000 .00625 7 44.5789

1000 .00625 8 51.1075

1000 .00625 9 57.6770

1000 .00625 10 64.2874

1000 .00625 11 70.9392

1000 .00625 12 77.6326

NESTING DO LOOPS

DO loops may be executed within a DO loop. Putting a DO loop within a DO loop is called **nesting.** The syntax is

do *index\_variable1*=*start1* to *stop1* by *increment1*;

SAS statements;

do *index\_variable2*=*start2* to *stop2* by *increment2*;

SAS statements;

end;

SAS statements;

end;

* It is important to assign **different index variable names** to the nested loops.

Example. Suppose in the earnings example, a deposit of $1,000 is made every year. The following code calculates the gross capital accumulated during a 2-year period.

data earnings2;

rate=0.075/12;

do year=1 to 2;

capital+1000;

do month=1 to 12;

capital+capital\*rate;

output;

end;

end;

run;

proc print noobs;

run;

The output is

rate year capital month

.00625 1 1006.25 1

.00625 1 1012.54 2

.00625 1 1018.87 3

.00625 1 1025.24 4

.00625 1 1031.64 5

.00625 1 1038.09 6

.00625 1 1044.58 7

.00625 1 1051.11 8

.00625 1 1057.68 9

.00625 1 1064.29 10

.00625 1 1070.94 11

.00625 1 1077.63 12

.00625 2 2090.62 1

.00625 2 2103.68 2

.00625 2 2116.83 3

.00625 2 2130.06 4

.00625 2 2143.38 5

.00625 2 2156.77 6

.00625 2 2170.25 7

.00625 2 2183.82 8

.00625 2 2197.46 9

.00625 2 2211.20 10

.00625 2 2225.02 11

.00625 2 2238.92 12

So, the gross capital accumulated during a 2-year period is $2,238.92.

* To drop the variables rate, year, and month after the calculations are done, type

in the above code

data earnings2 (drop=rate year month);

USING THE DO UNTIL STATEMENT

Sometimes it is desired to execute a DO loop until a certain condition is reached. It can happen that the number of iterations that the DO loop executes is not known a priori. In this case, the **DO UNTIL** statement is used in the DO loop. The syntax is

do until(*expression*);

SAS statements;

end;

* The loop is executed until expression is **true**.
* The expression is evaluated **at the bottom of the loop**, hence a DO UNTIL loop always executes **at least once**.
* If expression is **true**, the loop is **not executed again**.

Example. Suppose a deposit of $1,000 is made every year, and it is desired to calculate the minimum number of years it takes to accumulate at least $15,000 on the account. The annual interest rate is 0.075 compounded yearly. The following program uses the variable year to count the number of iterations in the DO UNTIL loop, because there is no index variable in this loop.

data account (drop=rate);

rate=0.075;

do until(capital>=15000);

capital+1000;

capital+capital\*rate;

year+1;

output;

end;

run;

proc print noobs;

run;

The output is

capital year

1075.00 1

2230.63 2

3472.92 3

4808.39 4

6244.02 5

7787.32 6

9446.37 7

11229.85 8

13147.09 9

15208.12 10

So, it takes 10 years to accumulate at least $15,000 on the account, and the gross amount in 10 years will be $15,208.12.

USING THE DO WHILE STATEMENT

The DO WHILE statement is used to execute a DO loop while the expression is true. The syntax is

do while(*expression*);

SAS statements;

end;

* The expression is evaluated **at the top of the loop**. If expression is **false**, the DO WHILE loop **never executes**.

Example. In the above example, a DO WHILE loop may be used in place of the DO UNTIL loop. The outputs are identical. The code is

data account (drop=rate);

rate=0.075;

**do while(capital<15000);**

capital+1000;

capital+capital\*rate;

year+1;

output;

end;

run;

proc print noobs;

run;

USING UNTIL CONDITION WITH THE DO STATEMENT

A DO loop with index variable may be combined with **UNTIL** statement to execute the loop the specified number of times or until an expression holds, whichever occurs first. The syntax is

do *index\_variable*=*start* to *stop* by *increment* until(*expression*);

SAS statements;

end;

Example. An annual 7.5% rate is compounded monthly. A deposit of $1,000 is made every year. The following code calculates the minimum number of years and months it takes to accumulate at least $15,000 on the account.

data account (drop=rate);

rate=0.075/12;

do until(capital>=15000);

capital+1000;

do month=1 to 12 until(capital>=15000);

capital+capital\*rate;

end;

year+1;

end;

year=year-1;

run;

proc print noobs;

run;

The output is

capital month year

15056.81 8 9

So, it takes 9 years and 8 months to reach $15,056.81 on the account.

* Note that to obtain the correct number of years, it is necessary to use the statement

year=year-1; after the DO UNTIL loop, because once the inner loop stops executing, the statement year+1; in the outer loop is still executed, increasing the correct number of years by 1.

USING WHILE CONDITION WITH THE DO STATEMENT

Alternatively to the UNTIL statement, a WHILE statement may be used in conjunction with a DO loop. The syntax is

do *index\_variable*=*start* to *stop* by *increment* while(*expression*);

SAS statements;

end;

Example. In the previous example, the same answer may be obtained by using a WHILE statement. The code is

data account (drop=rate);

rate=0.075/12;

do until(capital>=15000);

capital+1000;

do month=1 to 12 while(capital<15000);

capital+capital\*rate;

end;

year+1;

end;

year=year-1;

month=month-1;

run;

proc print noobs;

run;

* Note that now both variables year and month must be decreased by one to obtain the correct values, because for month=9 the expression in the WHILE statement is false and the inner loop stops executing.