Chapter 14

Generating Data With DO Loops

Overview

You can execute SAS statements repeatedly by placing them in a DO loop. DO loops can execute any number of times in a single iteration of the DATA step.

Using DO loops we can

- Read data
- Generate data
- Execute statements conditionally
- Write concise DATA steps

DO Loop Syntax

General form:

DO *index-variable = start* **TO** *stop*< **BY** *increment*>;
SAS statement block
END;

- **Start** value specifies the initial value of the index variable.
- The **TO** clause specifies the **stop** value. The stop value is the last index value that executes the DO loop.
- The optional **BY** clause specifies an **increment** value for the index variable. Typically, the DO loop increments by 1 for each iteration. If the BY clause is omitted, the default increment value is **1.**
- The *start, stop*, and *increment* values are set upon entry into the DO loop and cannot be changed during the processing of the DO loop. They can be numbers, variables, or SAS expressions.
- The **END** statement terminates the loop.

DO Loop Example

```
data _NULL_;
   sum=0;
   do i=1 to 7 by 2; * specify an index variable and the conditions;
      sum+i;          * i=1, 3, 5, 7;
      put sum=;
   end;
run;
```

```
sum=1
sum=4
sum=9
sum=16
```

Reduce Code with DO Loops

Sum the interest earned each month for a one-year investment

```
data earnings;
    set master;
    earned=0;
    earned+(amount+earned)*(rate/12);
    earned+(amount+earned)*(rate/12);
    earned+(amount+earned)*(rate/12);
    earned+(amount+earned)*(rate/12);
    earned+(amount+earned)*(rate/12);
    earned+(amount+earned)*(rate/12);
    earned+(amount+earned)*(rate/12);
    earned+(amount+earned)*(rate/12);
    earned+(amount+earned)*(rate/12);
    earned+(amount+earned)*(rate/12);
    earned+(amount+earned)*(rate/12);
    earned+(amount+earned)*(rate/12);
    run;
```

```
data earnings;
    set master;
    earned=0;
    do month=1 to 12;
        earned+(amount+earned)*(rate/12);
    end;
run;
```

DO Loop Execution

During compilation, the program data vector is created for the data set.

Program Data Vector

_N_	Amount	Rate	month	Earned
•	•	•	•	•

When the DATA step executes, the values of Amount and Rate are assigned.

Program Data Vector

_N_	Amount	Rate	month	Earned
1	1000	0.075	•	•

Next, the DO loop executes. During each execution of the DO loop, the value of Earned is calculated and is added to its previous value; then the value of month is incremented. On the twelfth execution of the DO loop, the value of month is incremented to 12 and the value of Earned is 77.6326.

Program Data Vector

_N_	Amount	Rate	month	Earned
1	1000	0.075	12	77.6326

SAS Data Set Finance.Earnings

Amount	Rate	month	Earned
1000	0.075	13	77.6326

Counting Iterations of DO Loops

In some cases, it is useful to create an index variable to count and store the number of iterations in the DO loop. Then you can drop the index variable from the data set.

```
data work.earn (drop=counter);
  Value=2000;
  do counter=1 to 20;
    Interest=value*.075;
    value+interest;    /*a sum statement*/
    Year+1;            /*a sum statement*/
  end;
run;
```

SAS Data Set Work.Earn

Value	Interest	Year
8495.70	592.723	20

Explicit OUTPUT Statements

You can create an observation for each iteration of the DO loop by placing an OUTPUT statement inside the loop. By default, every DATA step contains an implicit OUTPUT statement at the end of the step. But placing an explicit OUTPUT statement in a DATA step overrides automatic output, causing SAS to add an observation to the data set only when the explicit OUTPUT statement is executed.

```
data work.earn (drop=counter);
  Value=2000;
  do counter=1 to 20;
    Interest=value*.075;
    value+interest;   /*a sum statement*/
    Year+1;           /*a sum statement*/
    output;
 end;
run;
```

Value	Interest	Year
2150	150	1
2311.25	161.25	2
2484.59375	173.34375	3
2670.938281	186.344531	4
2871.258652	200.320371	5
3086.603051	215.344399	6
3318.09828	231.495229	7
3566.955651	248.857371	8
3834.477325	267.521674	9
4122.063124	287.585799	10
4431.217859	309.154734	11
4763.559198	332.341339	12
5120.826138	357.26694	13
5504.888098	384.06196	14
5917.754706	412.866607	15
6361.586309	443.831603	16
6838.705282	477.118973	17
7351.608178	512.902896	18
7902.978791	551.370613	19
8495.7022	592.723409	20

Decrementing DO Loops

You can decrement a DO loop's index variable by specifying a negative value for the BY clause

```
Data backwardsbyone;
        do i =20 to 0 by -2;
                output;
        end;
Run;

Proc print data=backwardsbyone nobs;
        title 'Counting backwards by 2';
Run;
```

Counting backwards by 2

i
20
18
16
14
12
10
8
6
4
2
0

Specifying a Series of Items

Rather than specifying start, stop and increment values in a DO statement, you can tell SAS how many times to execute a DO loop by listing items in a series. The list items must be all numeric, or all character, or all variables.

```
data fourseasons;
        do Season =  'spring', 'summer', 'fall', 'winter';
                select(Season);
                  when ('spring') Result='Beautiful!';
                  when ('winter') Result='Cold!';
                  when ('summer') Result='Hot!';
                  when ('fall') Result='Harvest!';
                end;
                output;
        end;
proc print data=fourseasons noobs;
  title 'The four seasons are:';
run;
```

The four seasons are:

Season	Result
spring	Beautiful!
summer	Hot!
fall	Harvest!
winter	Cold!

Nesting DO Loops

Putting a DO loop within another DO loop is called nesting. The DATA step below computes the value of a 20-year investment that earns 7.5% annual interest, compounded monthly. The same amount of capital is to be added each year. The program must perform the calculation for each month during each of the 20 years. To do this, you can include the monthly calculations within another DO loop that executes 20 times.

```
data work.earn;
  do year=1 to 20;
    Capital+2000;
    do month=1 to 12;
      Interest=capital*(.075/12);
      capital+interest;
      output;
    end;
  end;
run;
```

year	Capital	month	Interest
1	2012.5	1	12.5
1	2025.078125	2	12.578125
1	2037.734863	3	12.656738281
1	2050.470706	4	12.735842896
1	2063.286148	5	12.815441914
1	2076.181687	6	12.895538426
1	2089.157822	7	12.976135541
1	2102.215058	8	13.057236388
1	2115.353903	9	13.138844115
1	2128.574864	10	13.220961891
1	2141.878457	11	13.303592903
1	2155.265198	12	13.386740358
2	4181.235605	1	25.970407486
2	4207.368328	2	26.132722532
2	4233.66438	3	26.296052048
2	4260.124782	4	26.460402374
2	4286.750562	5	26.625779888
2	4313.542753	6	26.792191013
2	4340.502395	7	26.959642207
2	4367.630535	8	27.12813997
2	4394.928226	9	27.297690845
2	4422.396527	10	27.468301413
2	4450.036506	11	27.639978297
2	4477.849234	12	27.812728161
3	6518.335792	1	40.486557712
3	6559.07539	2	40.739598698
3	6600.069612	3	40.99422119
3	6641.320047	4	41.250435072

Iteratively Processing Observations from a Data Set

Compare how much each CD will earn at maturity from several institutions with an investment of $5,000. The DATA step below creates a new data set, Work.Compare, that contains the added variable, Investment.

SAS Data Set Finance.CDRates

Institution	Rate	Years
MBNA America	0.0817	5
Metropolitan Bank	0.0814	3
Standard Pacific	0.0806	4

```
data work.compare(drop=i);
  set finance.cdrates;
  Investment=5000;
  do i=1 to years;
    investment+rate*investment;
  end;
run;
```

SAS Data Set Work.Compare

Institution	Rate	Years	Investment
MBNA America	0.0817	5	7404.64
Metropolitan Bank	0.0814	3	6323.09
Standard Pacific	0.0806	4	6817.57

Notice that the data set variable Years is used as the stop value in the iterative DO statement. As a result, the DO loop executes the number of times specified by the current value of Years.

Conditionally Executing DO Loops

- The iterative DO statement specifies a fixed number of iterations for the DO loop.
- There are times when you want to execute a DO loop until a condition is reached or while a condition exists, but you don't know how many iterations are needed.
- The **DO WHILE** and **DO UNTIL** statements enable you to execute DO loops based on whether a condition is true or false.

DO UNTIL Loops

General form:

```
DO UNTIL (expression);
    SAS statements
END;
```

where *expression* is a valid SAS expression enclosed in parentheses.

- **The expression is not evaluated until the bottom of the loop**, so a DO UNTIL loop always executes at least once.
- When the expression is evaluated as true, the DO loop stops.

How many years will it take to have $50,000 if you deposit $2,000 each year into an account that earns 10% interest?

```
data work.invest;
  do until(Capital>=50000);
    capital+2000;
    capital+capital*.10;
    Year+1;
    output;
  end;
run;
```

Capital	Year
2200.00	1
4620.00	2
7282.00	3
10210.20	4
13431.22	5
16974.34	6
20871.78	7
25158.95	8
29874.85	9
35062.33	10
40768.57	11
47045.42	12
53949.97	13

DO WHILE Loops

General form:

```
DO WHILE (expression);
    SAS statements
END;
```

where *expression* is a valid SAS expression enclosed in parentheses.

- You can use the DO WHILE statement to execute a DO loop while the expression is true.
- **The DO WHILE expression is evaluated at the top of the DO loop.** If the expression is false the first time it is evaluated, the DO loop never executes.

How many years will it take to have $50,000 if you deposit $2,000 each year into an account that earns 10% interest?

```
data work.invest;
  do while(Capital<50000);
    capital+2000;
    capital+capital*.10;
    Year+1;
    output;
  end;
run;
```

Capital	Year
2200.00	1
4620.00	2
7282.00	3
10210.20	4
13431.22	5
16974.34	6
20871.78	7
25158.95	8
29874.85	9
35062.33	10
40768.57	11
47045.42	12
53949.97	13

Using Conditional Clauses with the Iterative DO Statement

This iterative DO statement enables you to execute the DO loop until Capital is greater than or equal to 50000 or until the DO loop executes ten times, whichever occurs first.

```
data work.invest;
  do year=1 to 10 until(Capital>=50000);
    capital+2000;
    capital+capital*.10;
  end;
  if year=11 then year=10;
run;
```

SAS Data Set Work.Invest

Capital	Year
35062.33	10

```
data work.invest;
  do year=1 to 10 until(Capital>=50000);
    capital+4000;
    capital+capital*.10;
  end;
  if year=11 then year=10;
run;
```

SAS Data Set Work.Invest

Capital	Year
50317.91	8

Creating Samples

Because a DO loop performs iterative processing, it provides an easy way to draw sample observations from a data set. The following example samples every tenth observation of the 5,000 observations in Factory.Widgets.

```
data work.subset;
  do sample=10 to 5000 by 10;
    set factory.widgets point=sample;
    output;
  end;
  stop;
run;
```

The DATA step reads the observations in Factory.Widgets identified by the POINT= option. The values of the POINT= option are provided by the DO loop, which starts at 10 and goes to 5,000 in increments of 10. The data set Work.Subset contains 500 observations.