SCstatisticalprogramming.com

Constructing DO Loops

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 lets you write concise DATA steps that are easier to change and debug, and also can greatly reduce the number of statements required for a repetitive calculation.

To construct a DO loop, you use the DO and END statements along with other SAS statements.

General form, simple iterative DO loop:

```
DO index-variable = start TO stop BY increment;
        SAS statements
END;
```

where the start, stop, and increment values

- are set upon entry into the DO loop
- cannot be changed during the processing of the DO loop
- can be numbers, variables, or SAS expressions.

The END statement terminates the loop.

Additional Note: The value of the index variable can be changed within the loop.

```
DO index-variable=start TO stop BY increment;
   SAS statements
END;
```

When creating a DO loop with the iterative DO statement, you must specify <u>an index variable</u>. The index variable stores the value of the current iteration of the DO loop. You may use any valid SAS name.

Next, specify the conditions that execute the DO loop. A simple specification contains a start value, a stop value, and an increment value for the DO loop.

<u>The start value</u> specifies the initial value of the index variable.

<u>The TO clause</u> specifies the stop value. <u>The stop value</u> is the last index value that executes the DO loop.

<u>The optional BY clause</u> specifies an increment value for the index variable. Typically, you want the DO loop to increment by 1 for each iteration. If you do not specify a BY clause, the default increment value is 1.

```
DO index-variable=start TO stop BY increment;
   SAS statements
END;
```

For example, the specification below increments the index variable by 1, resulting in quiz values of 1, 2, 3, 4, and 5:

```
do quiz=1 to 5;
```

By contrast, the following specification increments the index variable by 2, resulting in rows values of 2, 4, 6, 8, 10, and 12:

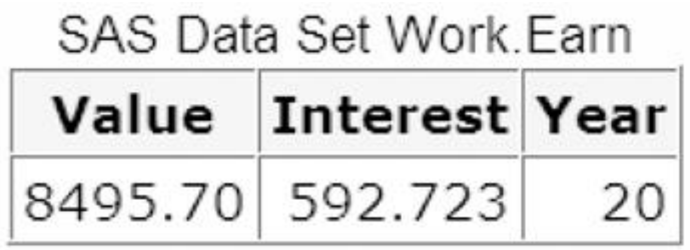
```
do rows=2 to 12 by 2;
```

Constructing DO Loops (Cont.)

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;
       Year+1;
   end;
run;
```



SAS Data Set Work.Earn

Value	Interest	Year
8495.70	592.723	20

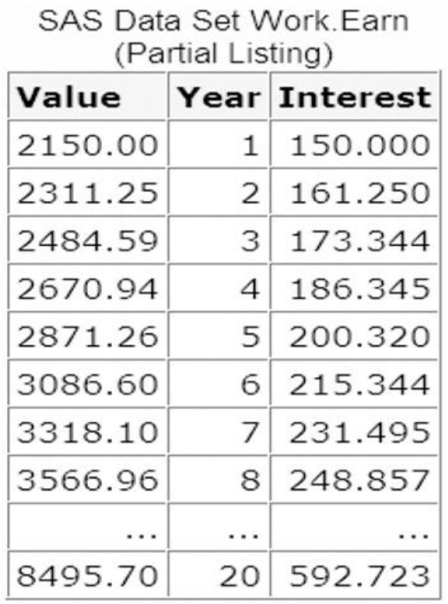
The sum statement Year+1 accumulates the number of iterations of the DO loop and stores the total in the new variable Year. The final value of Year is then stored in the data set, whereas the index variable counter is dropped. The data set has one observation.

Explicit OUTPUT Statements

To create an observation for each iteration of the DO loop, place 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.

The previous example created one observation because it used automatic output at the end of the DATA step. In the following example, the OUTPUT statement overrides automatic output, so the DATA step writes 20 observations.

```
data work.earn;
   Value=2000;
   do Year=1 to 20;
      Interest=value*.075;
      value+interest;
      output;
   end;
run;
```



SAS Data Set Work.Earn
(Partial Listing)

Value	Year	Interest
2150.00	1	150.000
2311.25	2	161.250
2484.59	3	173.344
2670.94	4	186.345
2871.26	5	200.320
3086.60	6	215.344
3318.10	7	231.495
3566.96	8	248.857
...	...	...
8495.70	20	592.723

SCstatisticalprogramming.com

Decrementing DO Loops

You can decrement a DO loop's index variable by specifying a negative value for the BY clause. For example, the specification in this iterative DO statement decreases the index variable by 1, resulting in values of 5, 4, 3, 2, and 1.

```
DO index-variable=5 to 1 by -1;
   SAS statements
END;
```

When you use a negative BY clause value, the start value must always be greater than the stop value in order to decrease the index variable during each iteration.

Specifying a Series of Items

You can also specify how many times a DO loop executes by listing items in a series.

General form, DO loop with a variable list:

```
DO index-variable = value1, value2, value3…;
          SAS statements
END;
```

where values can be character or numeric.

When the DO loop executes, it executes once for each item in the series. The index variable equals the value of the current item. You must use commas to separate items in the series.

SCstatisticalprogramming.com

To list items in a series, you must specify either

all numeric values

```
DO index-variable = 2,5,9,13,27;
    SAS statements
END;
```

all character values, with each value enclosed in quotation marks

```
DO index-variable = 'MON','TUE','WED','THR','FRI';
    SAS statements
END;
```

all variable names—the index variable takes on the values of the specified variables.

```
DO index-variable = win,place,show;
    SAS statements
END;
```

Variable names must represent either all numeric or all character values. Do not enclose variable names in quotation marks.