The Magnificent **Do**

Paul M. Dorfman

SAS Consultant

Jacksonville, FL

Q.: What is the DO statement in SAS NOT intended for?

- Doing all kinds of weird stuff with arrays
- Creating a *perpetuum mobile*
- Saving programming keystrokes
- Pandering to GOTO-less police
- Processing sequential files
- Grouping statements for block execution
- Coding for job security

Q: What are these three things?

- Sequence
- Selection
- Repetition

A: The three constructs necessary and sufficient for GOTO-less Programming

---

1. SEQUENCE
   (SAS: Natural Control Flow)

2. SELECTION
   (SAS: If-Then-Else / Select-End)

3. REPETITION
   (SAS: Do-Loop, Implied Loop)

Why Is The Repetition Structure Important?

- Forms basis for program automation
- Code once – execute many times
- Allows iterative instruction modification
- Naturally lends itself to better structured programming
- Provides for nesting periodic processes

Do-Loop: The Anatomy

```
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
Do <Index> = <From, By, To expressions>  While  | Until ( <expression> )  ;
   [TOP]:          Evaluate Index. If Index > To then go to EXIT
                   Evaluate While expression. If True go to EXIT
   - - - - - - - - - - - - - - - - - - - - - - - - -
    [BODY]:        < ... SAS instructions ... >
                   Leave  active? Go to EXIT
                   Continue active? Go to BOTTOM
                    < ... SAS instructions ... >
   - - - - - - - - - - - - - - - - - - - - - - - - -
   [BOTTOM]:       Evaluate Until expression. If True go to EXIT
                   Add By-expression to Index
   Go to TOP
End ;
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
[EXIT]:
```

“Golden Rule”
of Programming Repetition Structures
<broken all the time>

Only

Loop-Modified

<What’s that?>

Instructions Should Be Coded

Inside a Repetition Construct

Q.: What is a loop-modified instruction?
A.: Instruction whose effect may change as a result of the iterative process.

E.G.:

```
                 ----------------------------------------
                 |  Do J = 1 To N ;                      |
Unmodified       |     If J = 1 Then Put 'Beginning...' ;|
  Modified       |     Set DSN ;                         |
Unmodified       |     NewVar = Date() ;                 |
  Modified       |     If Not Mod(J,1e3) Then Put ‘Going...' ;|
  Modified       |     A (J + Offset) = B (J) ;          |
Unmodified       |     If J = N Then Put 'Over.' ;       |
                 |  End ;                                |
                 ----------------------------------------
```

Sequential File Reading / Processing: 3GLs

Explicit file-reading loop only, e.g. in COBOL:

```
PERFORM WITH TEST AFTER
   READ FILE AT END SET EOF TO TRUE
            NOT AT END PERFORM PROCESS-RECORD
            <... Other COBOL sentences ...>
END-PERFORM.
```

Sequential File Reading / Processing: SAS

1. Implied “observation loop”:
   <abused>

```
Data ... ;
  .............
  Set [Merge, Update, Input] ... ;
  .............
Run ;
```

---

2. Explicit Do-loop:
   <underused>

```
Data ... ;
  .............
  Do Until ( EoF ) ;
    Set [Merge, Update, Input]  End = EoF ... ;
    ...........
  End ;
  .............
  Stop ;
Run ;
```

Implied Loop in Do-Loop Terms

```
< Populate all valued retains at compile >

Do Internal_Counter = 1 By +1 ;
   < Initialize non-retains to missing ... >
   _N_ = Internal_Counter ;
   _Error_ = 0 ;
   < ... SAS statements ... >
   < SET, MERGE, INPUT, UPDATE ... > ... ;
      If < buffer-empty > Then Do ;
         If _Error_ NOT =  0 Then Put _All_ ;
         LEAVE ;
      End ;
   < ... SAS statements ... >
      If < DELETE-statement-active >  Then CONTINUE ;
      If < RETURN-statement-active > Then Do ;
          If < no-OUTPUT-statement-elsewhere > Then OUTPUT ;
             CONTINUE ;
      End ;
      If < STOP-active > Then LEAVE ;
      If < no-OUTPUT-statement-elsewhere > Then OUTPUT ;
      If _Error_ NOT =  0 Then Put _All_ ;
End ;
```

Implied Loop vs. Explicit Loop: Single File Processing

Given a SAS data set ACCOUNTS:

- Write a **header** to an external file OUT with current date formatted as YYYY-MM-DD (at position 1).
- Read a credit card account from a SAS data set ACCOUNTS and select only observations containing VISA numbers (they begin with 4). Write each selected account to OUT at position 1.
- After ACCOUNTS has been processed, write a **trailer**, with the date formatted as YYYY-MM-DD (positions 1-10) and total number of records in the file, excluding the header and trailer, with leading zeroes (positions 11-20).

Single File Processing:
Implied Loop vs. Explicit Do-Loop

```
Data _Null_ ;
  Retain Date ;
  If _N_ = 1 Then Do ;
    Date = Date () ;
    Put @1 Date YYMMDD10. ;
  End ;
  If EoF Then Put @ 1 Date YYMMDD10.
                    @11 N z10.  ;
  Set ACCOUNTS End = EoF ;
  If ACCTNO NE: '4' Then Delete ;
  N ++ 1 ;
  Put @1 ACCTNO $16. ;
 Run ;
```

Single File Processing:
Implied Loop vs. Explicit Do-Loop

```
Data _Null_ ;
  Retain Date ;

  If _N_ = 1 Then Do ;
    Date = Date () ;
    Put @1 Date YYMMDD10. ;
  End ;
  If EoF Then Put @ 1 Date YYMMDD10.
                      @11 N z10.  ;

  Set ACCOUNTS End = EoF ;
  If ACCTNO NE: '4' Then Delete ;
  N ++ 1 ;
  Put @1 ACCTNO $16. ;
 Run ;
```

```
Data _Null_ ;
  Date = Date () ;
  Put @1 Date YYMMDD10. ;

  Do Until ( EoF ) ;
    Set ACCOUNTS End = EoF ;
    If ACCTNO NE: '4' Then Continue ;
    N ++ 1 ;
    Put @1 ACCTNO $16. ;
  End ;

  Put @  1 Date YYMMDD10.
       @11 N Z10.  ;
  Stop ;
 Run ;
```

Explicit Do-Loop Multiple File Processing

```
Data ... ;
      <...do whatever SAS stuff you need before reading file(s)... >
   Do ... Until ( EoF1 ) ;
      Set [Merge, Input...] <File(s)> End = EoF1 ;
      <...process file 1...>
   End ;
      <...do SAS stuff after file 1...>
   Do ... Until ( EoF2 ) ;
      Set [Merge, Input...] <File(s)> End = EoF2 ;
      <...process file 2...>
   End ;
      <...do SAS stuff after file2...>
   Do ... Until ( EoF3 ) ;
      Set [Merge, Input...] <File(s)> End = EoF3 ;
      <...process file 3...>
   End ;
   <...do SAS stuff after file3...>
        ..............................
   <...more explicit Do-loops if need be...>
        ..............................
   <...do whatever SAS stuff you need before terminating step...>
   STOP ;
Run ;
```

Q.: I want my _N_ and _Error_ and stuff in the log !!!
A.: OK ...

```
Data ... ;
   ................
   Do _N_ = 1 By +1 Until ( EoF ) ;
      _Error_ = 0 ;
      ................
      Set A End = EoF ;
      ................
      If _Error_ Then Put _All_ ;
   End ;
   ................
   Stop ;
Run ;
```

Implied Loop and Explicit Do-Loop as a Control-break Team (The DoW-Loop)

Q: DoW-what ???
A: Not an industry term ... it is Whitlock... Mea culpa, Ian...

```
Data ... ;
  <...stuff done before each break_event...> ;

  Do <Index Specs> Until ( Break_Event ) ;
       Set [Merge, Update, Input, ...] ;

       <...stuff done for each incoming record...> ;

  End ;

  <...stuff done after each break-event... > ;

Run ;
```

Q.: What Is a Break-Event?

- Generally: Encountering any cardinal expression value (e.g. missing) in an iteration
- Most often: Last record in a by-group

The DoW-Loop: Example

```
Data B  ( Keep = ID Prod Summa Count Mean) ;
   Prod = 1 ;
   Do Count = 1 By +1 Until ( Last.ID ) ;
      Set A ;
      By  ID ;
      If Missing (Var) Then Continue ;
      Prod      = Prod * Var ;
      MeanCount = Sum (MeanCount, 1) ;
      Summa     = Sum (Summa, Var) ;
   End ;
   If MeanCount Then Mean = Sum / MeanCount ;

   * Here, 1 record per group is written automatically ;
Run ;
```

Q.: So what is the big DoW-deal?
A.: It is all in programming LOGIC

- Actions taken before, between and after break events naturally separated by the program in the stream-of-the-consciousness manner.

RULES:

- If an action is to be done before the group is processed, simply code it before the DOW-loop. It is NOT necessary to predicate this action by the <IF FIRST.ID> condition.

- If the action is to be done for each record, code it inside the loop.

- If is has to be done after the group, like computing an average and outputting summary values, code it after the DOW-loop.

Nesting DoW-Loops (Multi-Level Control-Break)

```
<...Initialize level X...>
Do X_cnt = 1 By 1 Until ( Last.X) ;
   <...Initialize level Y...>
   Do Y_cnt = 1 By 1 Until (Last.Y) ;
      <...Initialize level Z...>
      Do Z_cnt = 1 By 1 Until (Last.Z) ;
         Set XYZ ;
         By X Y Z ;
         <...Aggregate at level Z...>
      End ;
      <...Report    at level Z...>
      <...Aggregate at level Y...>
   End ;
   <...Report    at level Y...>
   <...Aggregate at level X...>
End ;
<...Report at level X...>
```

Conclusion

DO

IT !