ABSTRACT: Most high level programming languages utilize the array data structure, which allows the programmer to group several related variables into one variable name. With version 5, SAS Institute has introduced into all its operating environments an array structure more consistent with that of other languages by providing for explicit subscripting of arrays. This makes it possible to access an array element directly by specifying in a subscript after the array name which element is being sought. Since arithmetic calculations can be performed on the subscript variable, several array elements in any order can be accessed in a single statement. This direct access approach makes SAS code considerably easier to write, follow, and modify, features which are very important in programs intended for repeated usage. This paper will demonstrate the advantages of using explicitly-subscripted arrays in the calculation of bioavailability parameters. These values are required routinely by the pharmaceutical industry in the testing of new drugs.

An array is a data structure that is used to hold several related variables. The advantage to this structure is that the variables can all be processed the same way in one step simply by referring to the array name.

In version 5, explicitly-subscripted arrays (which are similar to those in high level languages like Pascal and FORTRAN) are now available in all operating environments in addition to SAS's standard implicitly-subscripted arrays. The differences between these two types are as follows:

The statement defining an explicitly-subscripted array contains the array name, a value indicating the number of elements, and the array elements. The analogous statement for an implicitly-subscripted array gives the array name, an optional index variable, and the array elements.

Explicit: ARRAY NAME [5] VAR1 - VAR5;
Implicit: ARRAY NAME [1] VAR1 - VAR5;

To access a specific element of an explicitly-subscripted array, simply append the desired subscript value to the array name. For contrast, to access a specific element of an implicitly-subscripted array, one must pursue the round-about method of assigning a value to the index variable, and then using the array name in a separate statement.

Explicit: NAME [3] = ...
Implicit: I = 3;
NAME = ...;

If one needs to access all elements of an array in turn, the iterative DO statement must be used for the explicit types whereas either the iterative DO or the DO OVER statement can be used for implicitly-subscripted arrays.

Explicit: DO I = 1 TO N;
Implicit: DO I = 1 TO N; or DO OVER NAME;

There are several advantages to using the explicitly-subscripted arrays:

- Array elements can be accessed directly.
- The code is easier to follow.
- In a single statement, one can travel through an array in either direction by performing arithmetic calculations on the subscript variable.

A current disadvantage to SAS's explicitly-subscripted arrays (except in AOS/VS, PRIMOS, and VMS systems) is the lack of a multi-dimension capability. However, this feature is expected in a future release.

To compare the use of SAS's explicitly and implicitly-subscripted arrays, these bioavailability calculations have been chosen:

- CMAX (maximum concentration of a compound in the patient's serum)
- TMAX (time at which maximum concentration is reached)
- AUC (area under the curve of concentration versus time).

These parameters are required routinely by the pharmaceutical industry in the testing of new drugs. Consequently, it is important that the programs calculating their values be as clear and direct as possible to facilitate repeated usage involving potential minor modifications of the code.

***** DETERMINATION OF CMAX AND TMAX *****;
Explicit

*** DEFINE ARRAYS.
***
ARRAY CONC [8] DRUG1 - DRUG8;
ARRAY TIMES [8] TIME1 - TIME8;

*** INITIALIZE CMAX AND TMAX TO ***;
*** FIRST VALUES OF ARRAYS.
***
CMAX = CONC [1];
TMAX = TIMES[1];

*****
*** CHECK REMAINING ARRAY ELEMENTS ***;
*** TO SEE IF GREATER. IF SO, SET ***
*** CMAX AND TMAX EQUAL TO THE ***
*** GREATER VALUES. ***;
DO I = 2 TO 8;
   IF (CONC[I] GT CMAX) THEN DO;
    CMAX = CONC[I];
    TMAX = TIMES[I];
   END; *IF*;
END; *DO*;

Implicit

*** DEFINE CONCENTRATION AND TIME ***;
*** ARRAYS.
ARRAY CONC [I) DRUG1 - DRUGS;
ARRAY TIMES [I] TIME1 - TIME8;
*** INITIALIZE CMAX AND TMAX TO ***;
*** FIRST DATA POINT.
I = 1;
CMAX = CONC;
TMAX = TIMES;
*** CHECK REMAINING ARRAY ELEMENTS ***;
*** TO SEE IF GREATER. IF SO, SET ***
*** CMAX AND TMAX EQUAL TO THE ***
*** GREATER VALUES. ***;
DO I = 2 TO 8;
   IF (CONC GT CMAX) THEN DO;
    CMAX = CONC;
    TMAX = TIMES;
   END; *IF*;
END; *DO*;

A missing value can be estimated as the average
of its adjacent values.

****** INTERPOLATE A MISSING VALUE. ******;

Explicit

DO I = 2 TO 7;
   IF (CONC[I] = .)
      THEN CONC[I] = (CONC[I-1] + CONC[I+1]) / 2;
   END; *DO*;

Implicit

*** DEFINE 3 CONCENTRATION ARRAYS SO ***;
*** THAT CURRENT, FORMER, AND LATTER ***;
*** DATA POINTS CAN BE ACCESSED. ***;
ARRAY C_CURRENT [I) DRUG1 - DRUG7;
ARRAY C_FORMER [I) DRUG1 - DRUG8;
ARRAY C_LATTER [I) DRUG3 - DRUG8;
DO OVER C_CURRENT;
   IF (C_CURRENT = .)
      THEN C_CURRENT = (C_FORMER + C_LATTER) / 2;
   END; *DO*;

Partial areas under the curve will be determined
by averaging the values of two adjacent
concentration points and then computing the area
as that average concentration times the
time displacement. The total area under the curve will be the sum of the seven partial
areas.

****** CALCULATE AREA UNDER CONC. ******;
****** VS TIME CURVE. ******;

Explicit

*** CREATE ARRAY TO HOLD EACH PARTIAL ***;
*** AREA UNDER CURVE. ***;
ARRAY PAUC [I] PAUC1 - PAUC7;
*** INITIALIZE TOTAL AUC VARIABLE ***;
*** TO ZERO. ***;
AUC = 0;
*** CALCULATE TOTAL AUC. ***;
DO I = 1 TO 7;
   PAUC[I] = (CONC[I] + CONC[I+1]) / 2
      * ABS(TIMES[I] - TIMES[I+1]);
   AUC = AUC + PAUC[I];
END; *DO*;

Implicit

*** DEFINE 2 CONCENTRATION ARRAYS AND ***;
*** 2 TIME ARRAYS SO THAT PRESENT AND ***;
*** NEXT DATA POINTS CAN BE ACCESSED. ***;
ARRAY C_PRESNT [I) DRUG1 - DRUG7;
ARRAY C_NEXT [I) DRUG1 - DRUG8;
ARRAY T_PRESNT [I) TIME1 - TIME7;
ARRAY T_NEXT [I) TIME2 - TIME8;
*** CREATE ARRAY TO HOLD EACH PARTIAL ***;
*** AUC. ***;
ARRAY PAUC[I] PAUC1 - PAUC7;
*** INITIALIZE TOTAL AUC VARIABLE TO ***;
*** ZERO. ***;
AUC = 0;
*** CALCULATE TOTAL AUC. ***;
DO OVER PAUC;
   PAUC = (C_PRESNT + C_NEXT) / 2
      * ABS(T_PRESNT - T_NEXT);
   AUC = AUC + PAUC;
END; *DO*;

The interpolation and AUC calculation examples
convincingly demonstrate the power of explicit
array subscripts. With explicit subcripting,
one need merely perform the necessary calcula-
tions on the subscript variable to move back
and forth through the array. Using implicit
subscripts, however, necessitates defining sev-
eral arrays that are offset from each other by
one data point so that the entire arrays can be
added or subtracted.

In conclusion, explicitly-subscripted arrays
offer the programmer direct access to array
elements which simplifies array statements and
increases clarity of the code. The resulting
programs are easier to follow and modify, fea-
tures that are extremely valuable in software
designed for repeated usage.

SAS® is the registered trademark of SAS
Institute Inc., Cary, NC, USA.

Patricia L. Gerend, Syntex Research
A4-200, 3401 Hillview Avenue
Palo Alto, CA 94304