Westat

Improving lives through research

Using Blaise for Implementing a Complex Sampling Algorithm

By Linda Gowen and Ed Dolbow, Westat Inc.

Introduction

A new in-person household survey using Blaise as the tool to perform computer aided interviewing (CAI), required a complicated sampling algorithm.

The system architects wanted to explore the idea of programming the algorithm in Blaise verses using other software to do it.

Reasons for using Blaise

- Eliminate the need to license, install and maintain separate sampling software
- Eliminate integrating Blaise system with sampling software.

New In-Person Household Survey

- The Survey gathers the information about a household including:
  - A Household member roster
  - Demographics
  - Personal characteristics
  - An extended adult survey
  - An extended youth survey
  - A supplemental adult survey

Sampling Algorithm Requirements

- 2 levels of sampling

  First - Select up to 2 youths and 2 adults to administer extended interviews.

  Second - Select a portion of the adult extended interviews to administer additional survey questions in the adult supplement survey.

- Apply a sampling rate for each person based on a combination of the person's characteristics(age, race, smoking status, etc). There are over 60 different sampling rates given the different combinations.

Sampling Algorithm Requirements, continued

- Apply adjustments to sampling rates when the initial household respondent reported a person's information differently than the person reports themselves in the extended interview.
- Generate random numbers, to the thousandth place precision (0.001), for both the household, as well as, each person in the household.
- Sort household members from lowest to highest by their random number.

Westat

Blaise Solution

- The 2 requirements we needed to explore for the complete solution was the **random** function and **sorting routine**.
- We had confidence the other requirements could easily be met using Blaise.

Westat

Random Function

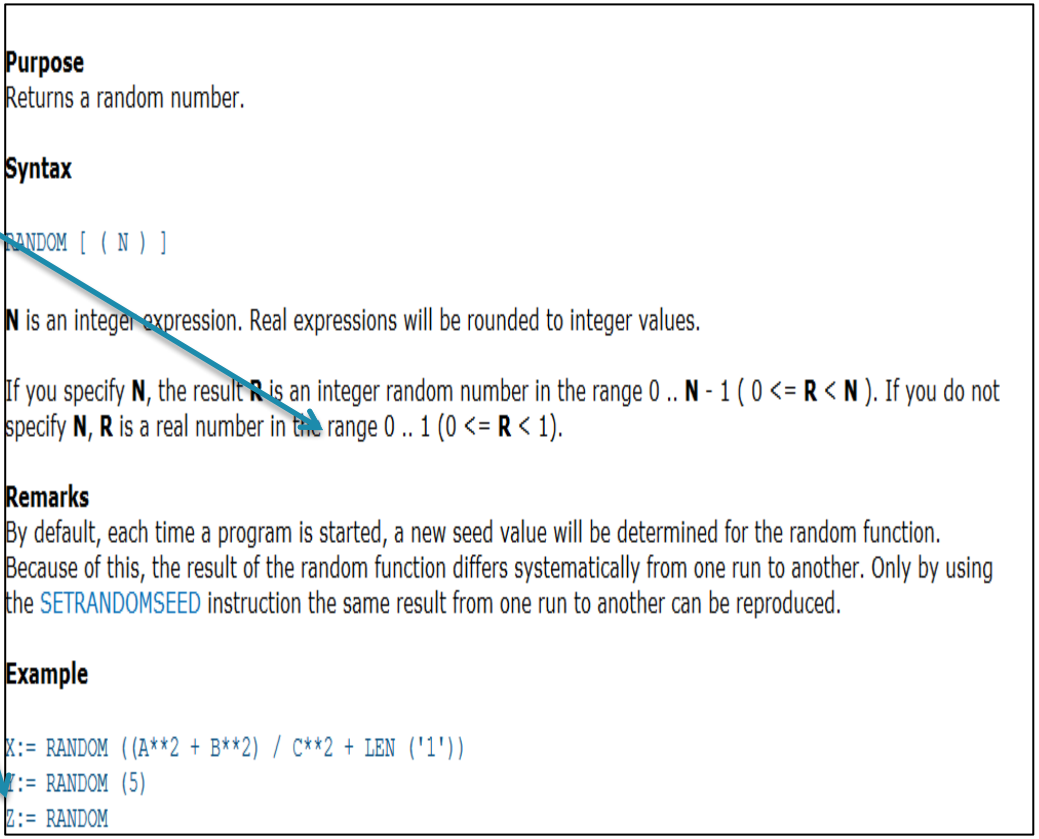
- Features we like
  - Thousandth place precision
  - Very simple to use

- Fields Definition

```
RAND1 (RAND1)
        {English text}
        "Person Random Number generated following
        a uniform distribution between 0 and 1."
        :  0.000..1.000,EMPTY
```

Note: The precision of the actual number is greater that the thousandth place, so the resulting number is rounded. So both 0 and 1 are generated.

From Blaise Help:



Purpose
Returns a random number.

Syntax

```
RANDOM [ ( N ) ]
```

N is an integer expression. Real expressions will be rounded to integer values.

If you specify **N**, the result **R** is an integer random number in the range 0 .. **N** - 1 (0 <= **R** < **N**). If you do not specify **N**, **R** is a real number in the range 0 .. 1 (0 <= **R** < 1).

Remarks
By default, each time a program is started, a new seed value will be determined for the random function. Because of this, the result of the random function differs systematically from one run to another. Only by using the SETRANDOMSEED instruction the same result from one run to another can be reproduced.

Example

```
X:= RANDOM ((A**2 + B**2) / C**2 + LEN ('1'))
Y:= RANDOM (5)
Z:= RANDOM
```

Random Number Generator Results

- We conducted an analysis in SAS on a dataset of 15,800 records generated by our Blaise program. We wanted to see if the random numbers were evenly distributed.

- Our conclusion is Blaise generated random numbers perfectly.

Mean	0.500607
Std Deviation	0.28960
Median	0.500000
Variance	0.08387
Mode	0.146000
Range	1.00000
Interquartile Range	0.50400

Westat

Bubble Sort in Blaise

- We decided to program a simple bubble sort to sort random numbers. We used the code below as an example.

```
For I:=1 TO 10 DO  {simple sorting algorithm}
    For j:=1 to 9 DO
        IF SortedArray[J] > SortedArray[j+1]  THEN
          k:= SortedArray[J]
          SortedArray[J]:= SortedArray[j+1]
          SortedArray[j+1]:=k
        ENDIF
    ENDDO
ENDDO
```

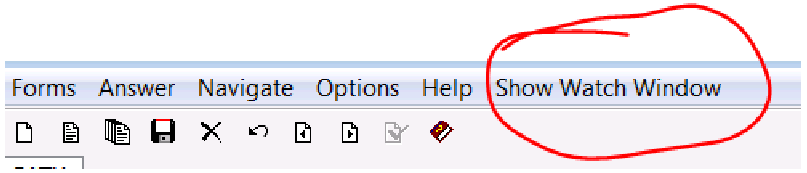
-

Westat

Testing the Sort Program

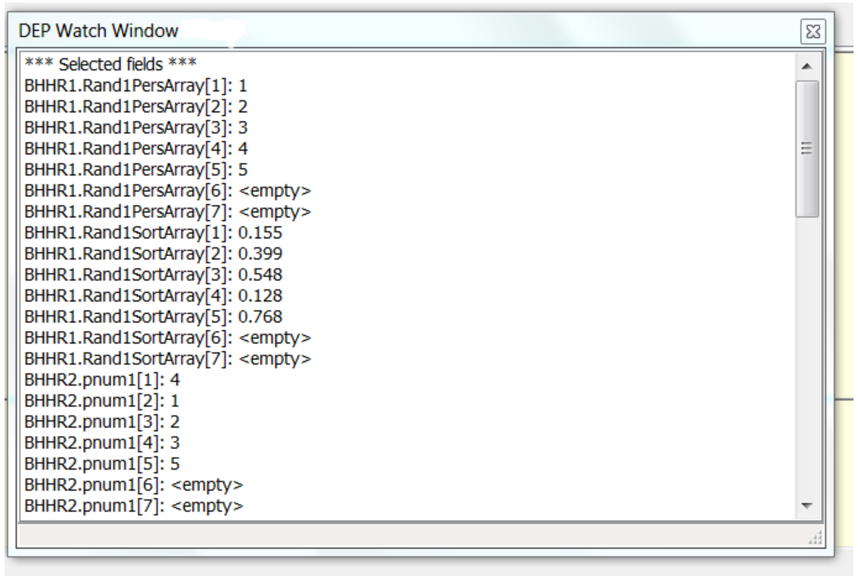
- We tested the sort by using the DEP watch window.

- To activate the DEP Watch Window, the option “/!” from the command line when calling the DEP program. The button to turn off/on the DEP watch screen will be activated in the Blaise Data Entry Screen.



Seeing Sorting Results in the DEP watch window

- This screen shot shows 5 persons in RAND1PersArray, the random number generated for each of the persons stored in RAND1SortArray and the person numbers sorted in the Pnum1 array.


DEP Watch Window

```
*** Selected fields ***
BHHR1.Rand1PersArray[1]: 1
BHHR1.Rand1PersArray[2]: 2
BHHR1.Rand1PersArray[3]: 3
BHHR1.Rand1PersArray[4]: 4
BHHR1.Rand1PersArray[5]: 5
BHHR1.Rand1PersArray[6]: <empty>
BHHR1.Rand1PersArray[7]: <empty>
BHHR1.Rand1SortArray[1]: 0.155
BHHR1.Rand1SortArray[2]: 0.399
BHHR1.Rand1SortArray[3]: 0.548
BHHR1.Rand1SortArray[4]: 0.128
BHHR1.Rand1SortArray[5]: 0.768
BHHR1.Rand1SortArray[6]: <empty>
BHHR1.Rand1SortArray[7]: <empty>
BHHR2.pnum1[1]: 4
BHHR2.pnum1[2]: 1
BHHR2.pnum1[3]: 2
BHHR2.pnum1[4]: 3
BHHR2.pnum1[5]: 5
BHHR2.pnum1[6]: <empty>
BHHR2.pnum1[7]: <empty>
```

Westat®

Conclusion

Everyone involved with the project, was pleasantly surprised at how well the sampling algorithm was implemented in Blaise both in performance and results. It was a huge advantage to stay with the same technology used by the data collection instruments. This way we avoided the additional complexities of maintaining and integrating different software and managing and paying for additional software licenses.