FORTTRAN IV subroutines to generate fully crossed random sequences of multidimensional stimuli with repetition limits on each dimension

A. P. WHITE and C. THOMPSON
Durham University, Durham DH1 3LE, England

Some psychological experiments require that multidimensional stimuli be presented to subjects in blocks arranged so that each possible combination of values occurs an equal number of times. Sometimes, it is also required to impose certain sequential constraints on the stimulus sequence. Programs exist to carry out these tasks (e.g., Hammond, 1977), but they usually work by first generating sequences at random that have the required frequencies and then checking them for the existence of undesired sequential contingencies. Sequences that fail to meet the applied criteria are merely rejected. In cases where the sequential constraints are severe, this can lead to very long search times being required and even absolute failure.

Description. The subroutines described here allow the user to generate sequences with the required frequency characteristics and then to check for the occurrence of repetitions of the same value on each of the dimensions. Unlike the types of program mentioned above, if a set of sequences fails to meet the sequential criteria, an attempt is made to reorder the sequences. This feature renders it more likely that acceptable sequences will be found within a reasonably short search time.

In order to describe the functioning of these subroutines, it is best to conceive of the output in terms of an m by n matrix, where m is the number of stimulus dimensions and n is the number of trials required.

To start with, the user's main program calls one of four subroutines: IRAN1, IRAN2, IRAN3, or IRAN4. These randomize one, two, three, or four variables, respectively, so that each possible combination of variables occurs an equal number of times. The values

for each variable are stored in vectors that are passed back to the main program. Thus, IRAN1 produces one such vector, IRAN2 produces two vectors, and so on. These vectors constitute the rows of the matrix described earlier.

If run limits are required on any of the variables, the main program can again call one of four subroutines: RUNL1, RUNL2, RUNL3, or RUNL4. These scan, respectively, one, two, three, or four vectors, in parallel, from left to right (i.e., beginning to end), looking for sequences of consecutive identical values that are longer than the limits specified by the user. If such a sequence is found, an attempt is made to find an acceptable position in the matrix for the column containing the offending element, in that part of the matrix that has already been examined. If such a position cannot be found, then this column is removed to the far right-hand end of the matrix, to be examined again later.

It may be that, after removing a column from Position K to the right-hand end of the matrix, the new column to occupy Position K is also rejected. Clearly, the situation may arise in which none of the remaining columns can be fitted into that part of the matrix that has already been examined. When this happens, the subroutine admits defeat, sets a switch, and returns control to the main program, which the user would probably require to generate another series of vectors, with one of the first series of subroutines.

The subroutines RUNL2, RUNL3, and RUNL4 also have the facility that the matrix can be checked for the existence of repetitions of entire column vectors. If the sequence of consecutive identical columns is longer than a limit specified by the user, the previously described procedure is put into operation for rearranging the columns.

Performance. In order to make comparisons between this method of generating sequences and the conventional method, a series of time tests was conducted using IRAN3. For each set of parameters employed, IRAN3 was used to generate a number of sequences

Table 1
Comparison of Efficiency of Methods for Generating Three-Dimensional Sequences with Repetition Limits on Each Dimension, Using an IBM 370/168

Length of Sequence	Levels per Dimension	Run Limits	New Method						Old Method			
			Succes- ses Timed	Successful*		Number Failed	CPU Time**	Mean Time per Success†	Succes- ses Timed	Number Failed	CPU Time**	Mean Time per Success†
8	2,2,2	1,2,2	2000	15	1985	319	6.83	3.42	200	34837	13.34	66.70
16	2,2,2	1,2,2	2000	0	2000	242	28.41	14.21	2	131804	85.29	42650
32	2,2,2	1,2,2	2000	0	2000	587	156.60	78.30	2		>200	>100000
32	2,2,2	2,2,2	2000	0	2000	6	38.69	19.35	2		>200	>100000
32	2,2,2	3,2,2	2000	0	2000	0	28.56	14.28	2		>200	>100000
60	3,4,5	4,4,4	2000	1192	808	0	17.02	8.56	2000	1317	9.04	4.52

*Number succesful with and without rearrangement. **In seconds. †In milliseconds.

that were checked for acceptability and rejected if they failed to meet the specified criteria. The CPU times taken were compared with those required to generate sequences with the same constraints, using RUNL3 to rearrange unacceptable sequences. No check for column repetitions was employed. The tests were run on an IBM 370/168. The results are summarized in Table 1.

The first three rows of the table show that, for severe run limits (1, 2, 2), the new method is more efficient than the old and becomes increasingly so as the required sequence length is increased. In fact, the old method did not succeed in generating two sequences in 200 sec when the sequence length was 32. The third, fourth, and fifth rows in the table show that, as the sequence length is held constant at 32 and the run limits are progressively relaxed, the new method becomes more rapid, although the old method continues to fail to produce sequences in the time allowed. The final row of the table shows that, for a very easy sequence,

where the initial failure rate is about 40%, the old method is approximately twice as fast as that using the rearrangement technique. This suggests that this technique is best used only when the parameters are such that it is difficult to find an acceptable sequence by the conventional method.

Computer and Language. All the subroutines are written in a subset of FORTRAN IV and have been run on an IBM 370/168 and on an IBM 1130.

Availability. Instructions and listings are available without cost from C. Thompson, Department of Psychology, Durham University, Durham DH1 3LE, England.

REFERENCE

- HAMMOND, H. V. A program to generate random sequences of multidimensional stimuli with constraints on first-order and second-order contingencies. *Behavior Research Methods & Instrumentation*, 1977, 9, 549-550.

(Accepted for publication July 18, 1978.)