Comput. Biol. Med. Vol. 23, No. 2, pp. 167–176, 1993
Printed in Great Britain

0010-4825/93 $6.00 + .00
© 1993 Pergamon Press Ltd

RANDOMIZATION ALGORITHMS IN BASIC FOR EXPERIMENTAL DESIGN

JOHN A. BYERS
Department of Ecology, Animal Ecology, Lund University, S-223 62 Lund, Sweden

(*Received* 12 *February* 1992; *in revised form* 5 *October* 1992; *received for publication* 15 *October* 1992)

Abstract—Six BASIC programs for randomization of treatments with respect to space and time are presented. Program 1 is used to obtain randomization of several treatments in an equal number of positions for any number of replicates such that identical treatments are not replicated successively in the same position. Program 2 randomly assigns different treatments as specified in a grid of any size either (a) without constraints or (b) so that similar treatments do not occur next to each other either horizontally or vertically, or (c) horizontally, vertically, or diagonally. Program 3 assigns from five to 100 different treatments in equal proportions to a grid of specified size with treatments separated as in Program 2 (c) above. Program 4 randomly assigns any number of different treatments in equal proportions to a Latin square-like grid containing row and column cells with a multiple number of the treatments such that no identical treatments are replicated successively in the same row, column, or both row and column. Program 5 produces Latin squares of letters with unique numbered subscripts randomized. Program 6 makes Greco-Latin squares with an odd number of letters per side. These programs will aid in randomization of treatments within positions and replicates when a degree of uniformity or spacing of treatments is desired in order to increase the power of statistical tests. Examples of program output are discussed for tests with bark beetle pheromone components.

BASIC   Algorithms   Randomization   Sampling   Experimental design
Scolytidae   Pheromone

INTRODUCTION

Textbooks of experimental design and sampling discuss algorithms for treatment randomization and sampling at random [1–9]. However, only a few general algorithms are presented and none of these are implemented by computer. Recently, Byers [10] implemented five algorithms on the personal computer for use in randomization of treatments and for sampling at random. The computer-programmed algorithms in the BASIC language allowed construction of a Latin square of any size, or a quasi-Latin square with equal numbers of replicated treatments for each row and column. The algorithms also included a method for simple randomization of treatments and positions for any number of replicates. In addition, random placement of equal numbers of different treatments within a grid could be done in such a way so that identical treatments were separated both horizontally and vertically from one another by other treatments (spacing of treatments).

In this paper additional algorithms are presented, that are used to randomize treatments such that a degree of uniformity or spacing results among identical treatments. Some of the algorithms here allow even more spacing of identical treatments in a grid to include horizontal, vertical and diagonal separation. Also, certain algorithms can be used with unequal numbers of replicates of several treatment types (including no treatment), with or without the spacing function.

The creation of these algorithms for spacing identical treatments was inspired by experiments with insect pheromones. It seems advisable when testing pheromone component blends in the field to distribute the treatments more uniformly throughout an area if the chemical dispensers have high release rates, so as not to concentrate responding individuals at unnatural densities in certain places. It is also well known that densities of flying insects may vary dramatically over an area [11, 12]. Thus, it is logical to place treatments consisting of traps releasing different blends of behavioral chemicals more uniformly, but still at random, over an area. This would ensure that all treatment blends were fairly presented to the flying insects and also would minimize the risk of confusion by high concentrations of pheromone in certain regions, compared with some purely random placements of blends.

The algorithms here are presented as examples for use in randomization of treatments. However, in many cases it is also possible to use the algorithms for sampling at random but with a degree of uniformity (i.e. ensuring that the area is more regularly sampled, similar in principle to stratified sampling [13]). Another consideration is that the examples of two-dimensional grids (space by space intersection) can be imagined alternatively as a spatial-position by time-period intersection. The easy and efficient use of the six randomization algorithms in BASIC (Figs 1–6) should help improve the design of scientific experiments and allow more efficient analysis of the results.

METHODS

The six programs operate in the BASIC language (BASICA, (C) 1983 or QuickBASIC 4.0, (C) 1982-1987 Microsoft Corp.) on the IBM, PC, XT, AT or compatible computer, although they could be easily modified for other computer brands using other versions of BASIC because of the restricted use of commands and simple output statements. Multiple statements (separated here by colons) can be placed on successive lines if required. However, statements following *IF-THEN* conditional tests must be repositioned with care.

RESULTS AND DISCUSSION

Program 1: *Simple treatment/position randomization with no repeats between replicates*

The program (Fig. 1A) will generate a set of randomly selected treatments (1 to any number of treatments) for the same number of positions for any number of experimental replicates. This algorithm is similar to program 1 described earlier [10] except that no treatment can be repeated between successive replicates. This constraint causes a degree of "spacing", either through time or spatially (if the row by column array represents a grid of positions).

One enters the number of treatments (8), which is equal to the number of positions (8), for any number of replicates (16), and a number which serves as the random seed. The random seed number will determine the starting point in the sequence of random numbers and thus it should not be the same if the experiment is repeated. For each row (treatment) and each column (replicate) the computer picks at random a number (representing the position) from an array (such that the number is no longer available for selection). During subsequent replicates, the algorithm checks each treatment to see if the previous replicate has the same position selected, if so, then further selections at random are tried until a non-matching position is found. Output of the program for eight treatments (A–H) and positions (1–8) replicated 16 times is shown in Fig. 1B.

Program 2: *Spaced treatments proportioned as specified in any row by column grid*

The treatments can be placed either (a) purely at random, (b) spaced so that identical treatments do not occur vertically or horizontally next to each other, or (c) neither vertically, horizontally or diagonally. This program (Fig. 2A) is used to place different treatments in equal or unequal proportions in a grid. The grid-side dimensions can be varied in size. The program also checks that the number of desired treatments and their replicates do not exceed the number of grid positions. If the total number of different treatment replicates is less than the number of grid positions, then zeros will be placed in the extra positions at random.

A.

```
10 CLS : PRINT "SIMPLE TREATMENT/POSITION RANDOMIZATION"
20 PRINT "WITH NO REPEATS BETWEEN REPLICATES"
30 INPUT "ENTER NUMBER OF TREATMENTS/POSITIONS"; N
40 INPUT "ENTER NUMBER OF REPLICATES"; R
50 INPUT "ENTER A SPECIFIC NUMBER FOR A SPECIFIC RANDOM SEQUENCE"; RN
60 DIM A(N, R): RN = RND(-RN)
70 FOR Z = 0 TO R - 1: W = -1
80 W = W + 1
90 V = -1: X = INT(RND * N) + 1
100 V = V + 1: IF A(V, Z) = X THEN 90
110 IF V = W THEN 130
120 GOTO 100
130 A(W, Z) = X: IF Z > 0 THEN 140 ELSE 160
140 IF A(V, Z) = A(V, Z - 1) THEN 150 ELSE 160
150 FOR EE = 0 TO W: A(EE, Z) = 0: NEXT: W = -1: GOTO 80
160 IF W < N - 1 THEN 80
170 NEXT Z: PRINT "T    REPS"
180 FOR P = 0 TO N - 1: PRINT CHR$(P + 65);
190 FOR E = 0 TO R - 1: PRINT USING "###"; A(P, E);
200 NEXT E: PRINT : NEXT P
```

B.

```
SIMPLE TREATMENT/POSITION RANDOMIZATION
WITH NO REPEATS BETWEEN REPLICATES
ENTER NUMBER OF TREATMENTS/POSITIONS? 8
ENTER NUMBER OF REPLICATES? 16
ENTER A SPECIFIC NUMBER FOR A SPECIFIC RANDOM SEQUENCE? 5
T    REPS
A  3  5  2  1  8  3  5  4  7  6  3  7  5  1  4  6
B  7  1  8  2  6  4  6  1  3  4  7  6  8  5  6  7
C  5  7  6  5  7  2  3  6  1  7  2  5  7  4  8  4
D  2  4  3  8  5  1  7  5  2  3  6  3  2  7  2  1
E  6  3  7  4  2  7  2  3  4  1  8  1  4  3  7  2
F  8  6  1  7  1  6  1  8  5  8  5  4  1  6  1  8
G  4  2  4  6  3  5  8  7  6  2  1  8  6  8  5  3
H  1  8  5  3  4  8  4  2  8  5  4  2  3  2  3  5
```

Fig. 1. (A) Listing of BASIC program used to randomly assign treatments to positions such that no identical treatments are replicated successively in the same position. (B) Printed output of program on computer screen of an IBM PC or compatible computer (output may differ due to differences in the random number generator algorithm used by the version of BASIC).

One enters the number of columns (12) and the number of rows (6) and the program displays the number of grid positions (72). Then the number of different treatments desired is entered (5). For each treatment, a number of replicates is entered (10) that cannot be more than the number of grid positions remaining. A random seed is entered to allow different random selections for different experiments. Output of the program for five treatments (1–5) each replicated 10 times and with 22 positions empty is shown in Fig. 2B.

Program 3: *Five to* 100 *very spaced treatments in equal proportions in any size grid*

This program (Fig. 3A) is similar to program 2 part (c), but it is much more efficient at finding a solution to the spacing of treatments. The grid sides can be of any dimensions but the number of grid positions must coincide with the total number of tests comprised of the different treatments, all equally proportioned. The spacing of identical treatments occurs in all possible directions, vertically, horizontally and diagonally. To illustrate, one can suppose that bark beetles, *Dendroctonus brevicomis*, will be tested for their response to a grid of 120 holes releasing volatile attractants. The holes are drilled in a plywood board and small vials containing chemicals are placed over each hole on the bottom side. Five treatments of 24 vials each are tested: treatment no. 1 is empty, no. 2 has a pine tree compound (myrcene), no. 3 has myrcene and a pheromone component (*exo*-brevicomin) produced only by females, no. 4 has myrcene and a male-produced pheromone component (frontalin), and no. 5 has all three chemicals. These chemicals are synergistic in attracting *D. brevicomis* to a tree or to the board [14]. However, the question is whether it is possible for the beetles to discriminate between the treatment holes by

A.

```
10 CLS : PRINT "SPACED TREATMENTS PROPORTIONED AS SPECIFIED IN ANY ROW";
20 PRINT "*COLUMN GRID"
30 PRINT "ENTER 1: TREATMENTS PLACED PURELY AT RANDOM";
40 PRINT "ENTER 2: SAME TREATMENTS NOT ALLOWED HORIZONTALLY OR VERTIC";
50 PRINT "ALLY": PRINT "ENTER 3: SAME TREATMENTS NOT ALLOWED HORIZONTA";
60 CLEAR : INPUT "LLY, VERTICALLY OR DIAGONALLY"; SP: SP = INT(SP)
70 IF SP >= 1 OR SP <= 3 THEN TR = 0: GOTO 80 ELSE 30
80 INPUT "ENTER NUMBER OF COLUMNS"; C
90 INPUT "ENTER NUMBER OF ROWS"; R:
100 PRINT "TREATMENTS MUST BE LESS THAN COL.*ROWS = "; C * R: CR = C * R
110 INPUT "ENTER NUMBER OF TREATMENTS"; N: DIM TR(N + 1): DIM TN(N + 1)
120 IF N > CR OR N < 1 THEN PRINT "INCORRECT NUMBER OF TREATMENTS": GOTO 30
130 IF N < 5 AND SP > 1 THEN PRINT "TREATMENTS OF <5 MIGHT NOT BE SPACED"
140 FOR W = 1 TO N: PRINT "> ENTER NUMBER OF REPLICATES FOR TREATMENT #"; W;
150 INPUT TR(W): TR = TR(W) + TR: PRINT "#LEFT="; CR - TR;
160 IF TR > CR THEN PRINT "TOO MANY REPLICATES - WON'T WORK": GOTO 30
170 NEXT: PRINT : IF CR - TR = 0 THEN 190
180 PRINT CR - TR; "CELLS WITH NO TREATMENT": TR(N + 1) = CR - TR
190 TN(N + 1) = TR(N + 1): DIM A(R + 1, C + 1)
200 FOR W = 0 TO R + 1: A(W, 0) = 500: NEXT
210 FOR W = 0 TO C + 1: A(0, W) = 500: NEXT
220 FOR W = 1 TO N + 1: TN(W) = TR(W): NEXT
230 INPUT "ENTER A SPECIFIC NUMBER FOR A SPECIFIC RANDOM SEQUENCE"; RN
240 X = RND(-RN): IF TR(N + 1) > 0 THEN NC = N + 1 ELSE NC = N
250 Q = INT(RND * NC) + 1: A$ = INKEY$: IF A$ = CHR$(27) THEN END
260 IF TR(Q) < 1 THEN GOTO 250
270 H = INT(RND * R) + 1: J = INT(RND * C) + 1
280 TRY = TRY + 1: IF TRY = 100 THEN GOSUB 290: GOTO 250 ELSE 320
290 FOR P = 1 TO N: TR(P) = TN(P): NEXT: TRY = 0: CT = 0
300 TY = TY + 1: LOCATE , 1: PRINT "[Esc] TO BREAK: TRY #"; TY;
310 FOR W = 1 TO R: FOR Z = 1 TO C: A(W, Z) = 0: NEXT: NEXT: RETURN
320 IF A(H, J) > 0 THEN GOTO 250
330 IF SP = 1 THEN 390
340 IF A(H - 1, J) = Q OR A(H + 1, J) = Q THEN 250
350 IF A(H, J - 1) = Q OR A(H, J + 1) = Q THEN 250
360 IF SP = 2 THEN 390
370 IF A(H - 1, J - 1) = Q OR A(H + 1, J - 1) = Q THEN 250
380 IF A(H - 1, J + 1) = Q OR A(H + 1, J + 1) = Q THEN 250
390 TR(Q) = TR(Q) - 1: TRY = 0
400 A(H, J) = Q: CT = CT + 1: IF Q = N + 1 THEN A(H, J) = 0
410 IF CT <> CR THEN 250
420 PRINT : FOR W = 1 TO R: FOR Z = 1 TO C
430 PRINT USING "###"; A(W, Z); : NEXT Z: PRINT : NEXT W
```

B.

```
SPACED TREATMENTS PROPORTIONED AS SPECIFIED IN ANY ROW*COLUMN GRID

ENTER 1: TREATMENTS PLACED PURELY AT RANDOM
ENTER 2: SAME TREATMENTS NOT ALLOWED HORIZONTALLY OR VERTICALLY

ENTER 3: SAME TREATMENTS NOT ALLOWED HORIZONTALLY, VERTICALLY OR
DIAGONALLY? 3
ENTER NUMBER OF COLUMNS? 12
ENTER NUMBER OF ROWS? 6
TREATMENTS MUST BE LESS THAN COL.*ROWS =  72
ENTER NUMBER OF TREATMENTS? 5
> ENTER NUMBER OF REPLICATES FOR TREATMENT # 1 ? 10
#LEFT= 62 > ENTER NUMBER OF REPLICATES FOR TREATMENT # 2 ? 10
#LEFT= 52 > ENTER NUMBER OF REPLICATES FOR TREATMENT # 3 ? 10
#LEFT= 42 > ENTER NUMBER OF REPLICATES FOR TREATMENT # 4 ? 10
#LEFT= 32 > ENTER NUMBER OF REPLICATES FOR TREATMENT # 5 ? 10
#LEFT= 22
 22 CELLS WITH NO TREATMENT
ENTER A SPECIFIC NUMBER FOR A SPECIFIC RANDOM SEQUENCE? 1

  4  1  3  5  1  0  3  0  3  4  0  3
  2  5  4  0  2  5  0  5  0  0  1  4
  3  1  2  0  0  1  4  0  1  4  5  2
  5  4  0  5  3  0  2  0  2  0  3  0
  2  3  2  4  1  4  0  3  1  0  1  4
  1  0  0  5  0  2  0  5  0  2  5  3
```

Fig. 2. (A) Listing of BASIC program used to randomly assign any number of different treatments in specified proportions in any size grid. The treatments can be placed at random with (1) no constraints, or with spacing so that similar treatments do not occur either (2) horizontally or vertically to each other or (3) horizontally, vertically, or diagonally to each other. (B) Printed output of program on computer screen of an IBM PC or compatible computer (output may differ due to differences in the random number generator algorithm used by the version of BASIC).

A.

```
10 CLS : PRINT "5 TO 100 VERY SPACED TREATMENTS (SAME TREATMENTS NEVER"
20 PRINT "NEXT TO EACH OTHER) IN EQUAL PROPORTIONS IN ANY SIZE GRID"
30 DEFINT A-Z: INPUT "ENTER NUMBER OF ROWS"; R
40 INPUT "ENTER NUMBER OF COLUMNS"; L
50 INPUT "ENTER 5 TO 100 TREATMENTS"; N
60 IF N >= 5 AND N <= R * L AND N <= 100 THEN 80
70 PRINT "TREATMENTS MUST BE FROM 5 TO 100 AND <"; R * L: GOTO 30
80 IF INT(R * L / N) = R * L / N THEN 110
90 PRINT "ROWS*COLUMNS NOT EVENLY DIVISIBLE"
100 PRINT "BY NUMBER OF TREATMENTS, TRY AGAIN": GOTO 30
110 DIM A(R + 1, L + 1): DIM C(101)
120 PRINT "ENTER A SPECIFIC NUMBER FOR A SPECIFIC RANDOM ";
130 INPUT "SEQUENCE"; RN: RN = RND(-RN)
140 FOR W = N + 1 TO 100: C(W) = R * L / N: NEXT
150 FOR W = 1 TO R: FOR Z = 1 TO L
160 A(W, Z) = INT(RND * N) + 1
170 IF A(W + 1, Z) = A(W, Z) OR A(W + 1, Z - 1) = A(W, Z) THEN 160
180 IF A(W - 1, Z) = A(W, Z) OR A(W - 1, Z - 1) = A(W, Z) THEN 160
190 IF A(W, Z + 1) = A(W, Z) OR A(W - 1, Z + 1) = A(W, Z) THEN 160
200 IF A(W, Z - 1) = A(W, Z) OR A(W + 1, Z + 1) = A(W, Z) THEN 160
210 FOR D = 1 TO N: IF A(W, Z) = D THEN C(D) = C(D) + 1
220 NEXT D: NEXT Z: NEXT W
230 M = 0
240 M = M + 1: IF M = N + 1 THEN 340
250 IF C(M) < R * L / N THEN P = M: C(M) = C(M) + 1: GOTO 270
260 GOTO 240
270 J = J + 1
280 IF C(J) > R * L / N THEN C(J) = C(J) - 1: Q = J: J = 0: GOTO 310
290 IF J = N THEN J = 0: GOTO 310
300 GOTO 270
310 FOR W = 1 TO R: FOR Z = 1 TO L: IF T = 1 THEN 330
320 IF A(W, Z) = Q THEN GOSUB 360
330 NEXT Z: NEXT W: T = 0: GOTO 230
340 FOR W = 1 TO R: FOR Z = 1 TO L: PRINT USING "###"; A(W, Z);
350 NEXT Z: PRINT : NEXT W: END
360 IF A(W + 1, Z) = P OR A(W + 1, Z - 1) = P THEN RETURN
370 IF A(W - 1, Z) = P OR A(W - 1, Z - 1) = P THEN RETURN
380 IF A(W, Z + 1) = P OR A(W - 1, Z + 1) = P THEN RETURN
390 IF A(W, Z - 1) = P OR A(W + 1, Z + 1) = P THEN RETURN
400 A(W, Z) = P: T = 1: RETURN
```

B.

```
5 TO 100 VERY SPACED TREATMENTS (SAME TREATMENTS NEVER
NEXT TO EACH OTHER) IN EQUAL PROPORTIONS IN ANY SIZE GRID
ENTER NUMBER OF ROWS? 8
ENTER NUMBER OF COLUMNS? 15
ENTER 5 TO 100 TREATMENTS? 5
ENTER A SPECIFIC NUMBER FOR A SPECIFIC RANDOM SEQUENCE? 5
  2  5  4  2  4  3  5  4  2  5  2  5  4  1  5
  4  3  1  3  5  2  1  3  1  3  1  3  2  3  4
  1  2  5  2  4  3  4  5  2  5  4  5  1  5  1
  5  3  1  3  1  5  1  3  1  3  2  3  2  4  2
  2  4  2  4  2  3  4  2  4  5  1  4  1  5  3
  1  3  1  5  1  5  1  3  1  2  3  2  3  2  4
  4  5  4  2  4  2  4  5  4  5  4  5  4  1  3
  2  3  1  5  1  5  1  3  1  3  2  3  2  5  4
```

Fig. 3. (A) Listing of BASIC program used to randomly assign five to 100 different treatments in equal proportions in a grid of any size with spacing so that similar treatments do not reside either horizontally, vertically, or diagonally to each other. (B) Printed output of program on computer screen of an IBM PC or compatible computer (output may differ due to differences in the random number generator algorithm used by the version of BASIC).

entering certain holes and becoming trapped in the vials? It may be desirable to even out the variation of position with respect to treatment on the board.

To construct such a test grid, one enters the number of rows (8), the number of columns (15), and the number of treatments (5) as well as the random seed. Since it is not possible to space (vertically, horizontally, and diagonally) four or fewer treatments in equal proportions, the program requires at least five treatments. The algorithms are similar to those described earlier ([10], Program 5) except that diagonal separation has been added. As can be seen in Fig. 3B, each treatment type is represented more uniformly over the board.

A.

```
10 CLS : PRINT "ANY # TREATMENTS EQUALLY PROPORTIONED IN LATIN ";
20 PRINT "SQUARE-LIKE GRID": PRINT "SPACED OR NOT SPACED": DEFINT A-Z
30 INPUT "ENTER NUMBER OF COLUMNS"; C: INPUT "ENTER NUMBER OF ROWS"; R
40 INPUT "ENTER NUMBER OF TREATMENTS"; N: G = N * 10: IF N < 2 THEN 30
50 IF INT(R / N) = R / N AND INT(C / N) = C / N THEN 70
60 PRINT "COLUMNS OR ROWS NOT EVENLY DIVISIBLE BY TREATMENTS": GOTO 30
70 INPUT "ENTER 1 FOR SPACED TREATMENTS, 2 FOR NO SPACING"; S
80 IF S = 2 THEN 100 ELSE PRINT STRING$(50, 95)
90 INPUT "SPACING: ENTER 1 FOR COLUMN, 2 FOR ROW, 3 FOR BOTH"; U
100 DIM A(R, C): DIM B(C): DIM C(C): DIM D(R): DIM E(R)
110 INPUT "ENTER A SPECIFIC NUMBER FOR A SPECIFIC RANDOM SEQUENCE"; RN
120 RN = RND(-RN): M = 0: V = 0
130 FOR I = 1 TO C STEP C / N: M = M + 1: FOR K = 1 TO C / N: V = V + 1
140 C(V) = M: NEXT: NEXT: REM MAKE ARRAY OF TREATMENTS
150 J = C: M = 0: V = 0: T = 0: REM RANDOMIZE COLUMN TREATMENT ARRAY
160 FOR I = 1 TO C
170 X = INT(RND * J + 1): T = T + 1: IF T > G THEN EXIT FOR
180 IF U <> 1 THEN IF S = 1 AND I > 1 THEN IF B(I - 1) = C(X) THEN 170
190 B(I) = C(X): J = J - 1
200 FOR K = X TO J: C(K) = C(K + 1): NEXT K: NEXT I: IF T > G THEN 120
210 FOR I = 1 TO R STEP R / N: M = M + 1: FOR K = 1 TO R / N: V = V + 1
220 E(V) = M: NEXT: NEXT: REM MAKE ARRAY OF TREATMENTS
230 J = R: REM RANDOMIZE ROW TREATMENT ARRAY
240 FOR I = 1 TO R
250 X = INT(RND * J + 1): T = T + 1: IF T > G THEN EXIT FOR
260 IF U <> 2 THEN IF S = 1 AND I > 1 THEN IF D(I - 1) = E(X) THEN 250
270 D(I) = E(X): J = J - 1
280 FOR K = X TO J: E(K) = E(K + 1): NEXT K: NEXT I: IF T > G THEN 120
290 FOR Z = 1 TO R: FOR W = 1 TO C: A(Z, W) = D(Z) + B(W)
300 IF A(Z, W) > N THEN A(Z, W) = A(Z, W) - N
310 PRINT USING "###"; A(Z, W); : NEXT: PRINT : NEXT
```

B.

```
ANY # TREATMENTS EQUALLY PROPORTIONED IN LATIN SQUARE-LIKE GRID

SPACED OR NOT SPACED
ENTER NUMBER OF COLUMNS? 6
ENTER NUMBER OF ROWS? 6
ENTER NUMBER OF TREATMENTS? 3
ENTER 1 FOR SPACED TREATMENTS, 2 FOR NO SPACING? (2,1,1,1)

__________________________________________________
SPACING: ENTER 1 FOR COLUMN, 2 FOR ROW, 3 FOR BOTH? ( ,1,2,3)

ENTER A SPECIFIC NUMBER FOR A SPECIFIC RANDOM SEQUENCE? 1
2  2  3  3  1  1    2  2  3  3  1  1    3  1  2  3  1  2    2  3  1  2  3  1
3  3  1  1  2  2    3  3  1  1  2  2    1  2  3  1  2  3    3  1  2  3  1  2
1  1  2  2  3  3    1  1  2  2  3  3    1  2  3  1  2  3    2  3  1  2  3  1
2  2  3  3  1  1    2  2  3  3  1  1    2  3  1  2  3  1    1  2  3  1  2  3
3  3  1  1  2  2    3  3  1  1  2  2    2  3  1  2  3  1    3  1  2  3  1  2
1  1  2  2  3  3    1  1  2  2  3  3    3  1  2  3  1  2    1  2  3  1  2  3
  Not Spaced          Column Spaced        Row Spaced          Both Spaced
```

Fig. 4. (A) Listing of BASIC program used to randomly assign any number of different treatments in equal proportions in a Latin square-like grid of any size. Treatments may be spaced (or not) such that no identical treatments are replicated successively in the same (1) column, (2) row, or (3) both column and row. The input values are shown in parentheses in the order necessary to generate the four possible spacings. (B) Printed output of program on computer screen on an IBM PC or compatible computer for the four possible spacings. Only one of the possible spacings will actually be output during any program run, but all four are shown here (output may differ due to differences in the random number generator algorithm used by the version of BASIC).

Program 4: Any number of treatments equally proportioned in Latin square-like grid with spacing or not between identical treatments

This program (Fig. 4A) provides a column by row array of positions or (positions × replicates) in which a certain number of different treatments are replicated equally to the extent allowed by the number of places in the column and row. Identical treatments may occur side by side or can be spaced so that no vertical, or horizontal, or vertical and horizontal positions can be filled by the same treatment.

One enters the number of columns (6), rows (6), and treatments (3) as well as the random seed number. The program checks to see if the number of treatments can divide evenly into both the rows and the columns, if not, then new input must be entered. The option of spacing or not is also entered. The algorithm is similar to that described earlier [10] in which the first step is to randomize both a row and a column array that each have the appropriate repetitions of the different treatments. However, in the option for the spacing of identical treatments, the algorithm must check the row array and/or the column array to see if the previous selection in the array is the same treatment, if so then another selection is made until a different treatment is found. Once these two arrays are randomized (with or without spacing) then their addition proceeds. The value for each row by array intersection cell is the sum of the appropriate cells in the two arrays, unless the sum is greater than N (the number of different treatments), in which case the value is the sum minus N. Output (Fig. 4B) shows the treatments evenly proportioned across the rows (replicates) and columns (positions) for the four cases: no spacing, spacing within columns, spacing within rows, and spacing both vertically and horizontally.

Program 5: *Latin square with randomized non-repeated subscripts*

The program (Fig. 5A) first constructs a Latin square of letters using the algorithm described in program 4 above. Then for each letter (A, B, C . . .) an array of numbers is randomized as were the initial row and column arrays when constructing the Latin square. When the appropriate letter (e.g. A) is found in a row then the next successive number in the number array is used as the letter's subscript. After all rows have been searched then a new number array is randomized and the next letter (e.g. B) is searched for in each row until all letters have been dealt with.

One enters the number of treatments (10) which correspond to the various letters and the number of subscripts for each letter, as well as the random seed. For example, one may test for behavioural synergism between two compounds taken from four candidate pheromone components. There are 6 possible combinations plus 4 single components that can compose 10 different blends (treatments A–J). One also could release 10 different rates of the volatile components (subscripts 1–10). The rows can represent 10 different positions in the field and each column can be each of 10 test days. The catch on traps containing the blends would then be analysed for differences with respect to blend and dosage.

The Latin square algorithm used here can generate all 12 squares of 3×3 letters, but not all possible Latin squares of more letters than three per side. For Latin squares of side $N = 4$ (4×4) the algorithm generates $N!(N-1)! = 144$ different squares, for a 5×5, 2880 squares, and for a 6×6, 86 400 squares [9]. There are actually 576 different 4×4 and 161 280 different 5×5 squares [4, 8]. Thus not all possible Latin squares can be obtained with the algorithm, but there is a sufficiently large sample from which to select a choice at random such that no significant experimental bias should result.

Program 6: *Greco-Latin square*

The program (Fig. 6A) will generate a Greco-Latin square in which every Latin letter occurs once in each row and once in each column, and each Greek letter (represented here by numbers instead) occurs once for each row and once for each column. In addition, each Latin letter and Greek letter (number) occur only once together in the square. Thus, for a Greco-Latin square of 5×5 there are 25 combinations of Latin letters and numbers which can be placed in 25 positions. This means that there are 25! (about 10^{25}) permutations out of which some Greco-Latin squares must be found.

Kempthorne [9] says a Greco-Latin square of side 12 exists, although he gives no solution or reference. He does say that Greco-Latin squares of even numbered sides have not been enumerated except as he shows for a side of 4 (and possibly for a side of 12). It can be seen that in Latin squares of sides $N = 3, 4, 5$ or more the number of 45° diagonals made from two or more quadrants number $N + j$, where j is $0, 1, 2, 3, \ldots$, so that squares of odd numbered sides have an even j while even numbered sides have an odd j. The 4-sided ($j = 1$) square is a special case, while Greco-Latin squares with even numbered sides of 6 or more (j is odd and 3 or more) do not seem to exist.

A.

```
10 CLS : DEFINT A-Z
20 PRINT "LATIN SQUARE WITH RANDOMIZED NON-REPEATED SUBSCRIPTS"
30 INPUT "ENTER NUMBER OF TREATMENTS"; N: R = N: IF N > 26 THEN 30
40 PRINT "ENTER A SPECIFIC NUMBER FOR A SPECIFIC RANDOM";
50 INPUT " SEQUENCE"; RN: DIM B(N): DIM C(N): DIM D(N)
60 RN = RND(-RN): DIM A$(N, N)
70 FOR I = 1 TO N: B(I) = I: C(I) = I: D(I) = I: NEXT I
80 J = N: REM RANDOMIZE ARRAY COLUMN
90 FOR I = 1 TO N
100 X = INT(RND * J + 1): B(I) = C(X): J = J - 1
110 FOR K = X TO J: C(K) = C(K + 1): NEXT K: NEXT I
120 J = N: REM RANDOMIZE ARRAY ROW
130 FOR I = 1 TO N: X = INT(RND * J + 1): C(I) = D(X): J = J - 1
140 FOR K = X TO J: D(K) = D(K + 1): NEXT K: NEXT I
150 FOR R = 1 TO N: FOR C = 1 TO N: T = B(R) + C(C)
160 IF T > N THEN T = T - N
170 A$(R, C) = CHR$(64 + T): NEXT C: NEXT R
180 FOR L = 1 TO N: J = N
190 FOR I = 1 TO N: C(I) = I: NEXT I
200 FOR I = 1 TO N
210 X = INT(RND * J + 1): B(I) = C(X): J = J - 1
220 FOR K = X TO J: C(K) = C(K + 1): NEXT K: NEXT I
230 T = 0: C$ = CHR$(L + 64)
240 FOR R = 1 TO N: T = T + 1
250 FOR C = 1 TO N: IF B(T) > 9 THEN S$ = " " ELSE S$ = "  "
260 IF A$(R, C) = C$ THEN A$(R, C) = A$(R, C) + MID$(STR$(B(T)), 2) + S$
270 NEXT C: NEXT R: NEXT L
280 FOR R = 1 TO N: FOR C = 1 TO N
290 PRINT A$(R, C); : NEXT C: PRINT : NEXT R
```

B.

```
LATIN SQUARE WITH RANDOMIZED NON-REPEATED SUBSCRIPTS
ENTER NUMBER OF TREATMENTS? 10
ENTER A SPECIFIC NUMBER FOR A SPECIFIC RANDOM SEQUENCE? 1
F4  B6  J10 G4  A8  D6  I8  E7  H1  C7
G1  C2  A5  H4  B9  E2  J5  F2  I6  D3
I9  E3  C1  J6  D4  G7  B3  H10 A1  F1
J2  F9  D10 A7  E8  H6  C3  I2  B7  G5
C6  I3  G8  D5  H5  A9  F10 B8  E6  J3
B10 H3  F3  C9  G9  J4  E1  A10 D8  I5
H7  D2  B5  I7  C8  F7  A6  G10 J1  E10
D7  J9  H9  E4  I1  B1  G2  C4  F8  A2
E5  A4  I10 F5  J7  C5  H2  D1  G6  B2
A3  G3  E9  B4  F6  I4  D9  J8  C10 H8
```

Fig. 5. (A) Listing of BASIC program used to create a Latin square with up to 26 letters per side and 26 unique subscripts per letter. (B) Printed output of program on computer screen on an IBM PC or compatible computer (output may differ due to differences in the random number generator algorithm used by the version of BASIC).

The program first finds a general solution for a Greco-Latin square with an odd number of letters per side of 3 or more. For example, for a 5×5 square, this is done by beginning the first row with ABCDE. The last letter of the row, E, starts on the next row down and arrangement of letters is subsequently ordered (EABCD). This pattern is shifted for the third (DEABC) and all rows resulting in a Latin square. The same procedure is done with numbers but in reverse (right to left) so that the first row is 54321 and the second row is 43215. The two Latin squares of letters and numbers are superimposed and result in a Greco-Latin square. The algorithm does not work for squares with sides with an even number of rows/columns.

By switching two of the columns one obtains another Greco-Latin square. Similarly, two rows can be switched to get two different squares. This principle is then used to switch at random various rows and columns to get many different Greco-Latin squares. It was found that the number of random switches of rows or columns needed to be at least as large as the number of rows/columns in order to effect a significant randomization of the initial pattern. With a 4×4 square it is possible to make six different swaps of the rows for a total of seven different squares. Also, six ($N = 6$) different swaps of the columns allow a grand total of $1 + [\Sigma_{j=1}^{N}(j-1)]^2 = 37$ different Greco-Latin squares including the original one. For a 5×5 there are 101 squares, for a 7×7 there are 442

A.

```
10 CLS : PRINT "GRECO-LATIN SQUARE": DEFINT A-Z
20 INPUT "ENTER 4 OR AN ODD NUMBER OF TREATMENTS"; N: R = N
30 IF N < 3 OR (N <> 4 AND INT(N / 2) = N / 2) OR N > 26 THEN 20
40 PRINT "ENTER A SPECIFIC NUMBER FOR A SPECIFIC RANDOM";
50 INPUT " SEQUENCE"; RN: RN = RND(-RN)
60 REM FIND ONE SOLUTION TO A GRECO-LATIN SQUARE OF SIDE N
70 DIM A(N, N): DIM B(N, N): DIM A$(N, N): IF N = 4 THEN 120
80 FOR R = 1 TO N: FOR C = 1 TO N: P = P + 1: A(R, C) = P
90 B(R, N + 1 - C) = P: IF C = N THEN P = P - 1
100 IF P = N THEN P = 0
110 NEXT C: NEXT R: GOTO 160: REM DATA FOR SIDE OF 4
120 DATA 1,1,2,2,3,3,4,4,2,4,1,3,4,2,3,1
130 DATA 3,2,4,1,1,4,2,3,4,3,3,4,2,1,1,2
140 FOR R = 1 TO N: FOR C = 1 TO N: READ A(R, C), B(R, C)
150 NEXT C: NEXT R
160 FOR W = 1 TO N: REM RANDOMIZE ROWS & COLUMNS N TIMES
170 RN = INT(RND * N) + 1: K = INT(RND * N) + 1
180 FOR C = 1 TO N: SWAP A(RN, C), A(K, C)
190 SWAP B(RN, C), B(K, C): NEXT C: REM ROWS SWAPPED
200 RN = INT(RND * N) + 1: K = INT(RND * N) + 1
210 FOR R = 1 TO N: SWAP A(R, RN), A(R, K)
220 SWAP B(R, RN), B(R, K): NEXT R: REM COLUMNS SWAPPED
230 NEXT W: REM END OF RANDOMIZATION OF ROWS AND COLUMNS
240 FOR R = 1 TO N: FOR C = 1 TO N
250 A$(R, C) = CHR$(A(R, C) + 64) + MID$(STR$(B(R, C)), 2)
260 A$(R, C) = A$(R, C) + STRING$(4 - LEN(A$(R, C)), 32)
270 PRINT A$(R, C); : NEXT C: PRINT : NEXT R
```

B.

```
GRECO-LATIN SQUARE
ENTER 4 OR AN ODD NUMBER OF TREATMENTS? 7
ENTER A SPECIFIC NUMBER FOR A SPECIFIC RANDOM SEQUENCE? 1
G1  B6  A7  C5  D4  F2  E3
D5  F3  E4  G2  A1  C6  B7
E6  G4  F5  A3  B2  D7  C1
C4  E2  D3  F1  G7  B5  A6
B3  D1  C2  E7  F6  A4  G5
A2  C7  B1  D6  E5  G3  F4
F7  A5  G6  B4  C3  E1  D2
```

Fig. 6. (A) Listing of BASIC program used to create a Greco-Latin square of up to 26 letters per side and subscripts per side. Both letters and subscripts form a Latin square and all cells have unique combinations of letters and subscripts. (B) Printed output of program on computer screen on an IBM PC or compatible computer (output may differ due to differences in the random number generator algorithm used by the version of BASIC).

squares. The general algorithm does not work for even numbered sides, but a Greco-Latin solution exists for a side of four as shown earlier [9]. This unique solution is used with the row and column randomization algorithm to obtain the 37 possible Greco-Latin squares.

Program output for a Greco-Latin square is shown in Fig. 6B. It can be supposed that here one wanted to test the effects of seven pheromone blends each at seven dosages in seven areas of the forest on seven days. The Greco-Latin square would allow a more powerful analysis since all treatments and dosages were tested at each position and on each day, although each position or day did not have identical treatments/dosages. Several texts discuss the advantages and drawbacks of using Latin and Greco-Latin squares as well as the statistical analysis of variance [1, 3, 4, 6, 8, 9].

The random generator in QuickBASIC (Microsoft®) can generate up to 10 million possible numbers and the sequence does not repeat for about 16.7 million selections, as tested with the following BASIC lines:

```
10 A=RND(-3):T=RND
20 C=C+1:R=RND:IF R<>T THEN 20 ELSE PRINT C
```

The advantage of computer generation of random numbers compared to the use of tables is that the computer can be combined with specific algorithms for randomization and sampling applications. The computer can process the algorithms at high speed, without error, and output the results on paper via a printer. Printed paper copy can be obtained in two ways, (1) by printing the screen with the 'Print Scrn' key and (2) by substituting LPRINT statements for the PRINT ones shown in the figures.

The experimental designs presented here that employ a degree of uniform spacing among treatments are more appropriate for use than traditional statistical analyses only when there are logical or pragmatic reasons, as in the examples above. Thus it seems advisable to consult a statistician when considering these specialized designs. A compiled program version is available from the author (send a formatted disk and return mailer).

SUMMARY

Six algorithms implemented in the BASIC computer language are presented for use in randomizing treatments and/or sampling during the design and implementation of experiments. The convenience of the use of personal computers for generating random numbers necessary for use in the algorithms is discussed. Methods for generation of sampling plans and treatment protocol are described for blocks of grid cells with or without separation of identical treatments in adjacent cells. True Latin squares or quasi-Latin squares (in which a row or column may have a multiple number of the same treatment), with or without identical treatment separation in adjacent cells (horizontally, vertically, or both) are also described. Latin squares with randomized subscripts as well as Greco-Latin squares can also be generated rapidly.

Acknowledgements—This work was supported by a research grant from the Swedish Agricultural and Forest Research Council, SJFR. Olle Anderbrant and Fredrik Schlyter were helpful in the review of the manuscript.

REFERENCES

1. V. L. Anderson and R. A. McLean, *Design of Experiments*. Marcel Dekker, New York (1974).
2. W. G. Cochran, *Sampling Techniques*. Wiley, New York (1963).
3. W. G. Cochran and G. M. Cox, *Experimental Designs*. Wiley, New York (1957).
4. B. E. Cooper, *Statistics for Experimentalists*. Pergamon Press, London (1969).
5. D. R. Cox, *Planning of Experiments*. Wiley, New York (1958).
6. M. N. Das and N. C. Giri, *Design and Analysis of Experiments*. Wiley, New Delhi (1979).
7. W. E. Deming, *Some Theory of Sampling*. Dover, New York (1950).
8. W. T. Federer, *Experimental Design*. MacMillan, New York (1955).
9. O. Kempthorne, *The Design and Analysis of Experiments*. Wiley, New York (1952).
10. J. A. Byers, Basic algorithms for random sampling and treatment randomization, *Comput. Biol. Med.* **21**, 69 (1991).
11. J. A. Byers, O. Anderbrant and J. Löfqvist, Effective attraction radius: a method for comparing species attractants and determining densities of flying insects. *J. chem. Ecol.* **15**, 749 (1989).
12. T. L. Payne, J. E. Coster, J. V. Richerson, E. R. Hart, R. L. Hedden and L. J. Edson, Reducing variation in field tests of behavioral chemicals for the southern pine beetle. *J. Georgia Entomol. Soc.* **13**, 85 (1978).
13. B. D. Ripley, *Spatial Statistics*. Wiley, New York (1981).
14. D. L. Wood, L. E. Browne, B. Ewing, K. Lindahl, W. D. Bedard, P. E. Tilden, K. Mori, G. B. Pitman and P. R. Hughes, Western pine beetle: specificity among enantiomers of male and female components of an attractant pheromone. *Science* **192**, 896 (1976).

About the Author—John A. Byers received his B.Sc. (1971) and M.Sc. (1973) in Entomology from Colorado State University, and Ph.D. (1978) in Entomology from the University of California at Berkeley. He is currently a Docent (Associate Professor) of Animal Ecology in the Pheromone Research Group, Lund University, Sweden. His main research interests are in insect behaviour and chemical ecology and in computer simulation of behavioral and ecological models.