Comput. Biol. Med. Vol. 21, No. 1/2, pp. 69–77, 1991
Printed in Great Britain

0010–4825/91 $3.00+.00

BASIC ALGORITHMS FOR RANDOM SAMPLING AND TREATMENT RANDOMIZATION

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

(*Received* 7 *June* 1990; *received for publication* 19 *October* 1990)

Abstract—Five BASIC programs to select random samples from populations or to randomize treatments are presented. Program 1 is used to obtain randomization of any number of treatments in an equal number of positions or test units for any number of replicates. Program 2 produces latin squares of any size for treatment randomization. Program 3 is used to obtain a specific number of randomly selected samples from a population without replacement. Program 4 produces quasi-latin squares that have treatments repeated equally in all rows and columns, with identical treatments either spaced or not. Program 5 can be used with any size grid to place 3–100 treatments in equal proportions and with spacing of identical treatments. Both programs 4 and 5 allow for horizontal and vertical separation between identical treatments at sampling places while still retaining the quality of randomness. These programs should facilitate random sampling and randomization procedures which are required to correctly analyze experiments by the methods of statistical probability.

BASIC Algorithms Randomization Sampling Experimental design

INTRODUCTION

The use of statistical methods for analysis of observational and experimental data is pervasive throughout biology and in fact all of science. One of the most common assumptions in statistical tests for properly determining differences in data sets is that sampling or subject selection were done at "random". This assumption of random selection applies to both parametric (e.g. *z*-test, *t*-test, ANOVA) and non-parametric tests (e.g. chi-square, Wilcoxon, Kruskal–Wallis, Mann–Whitney "U") [1]. In a sampling of statistical text books selected "at random" (actually selected in order of discovery) from the University statistical library, I found that only 5 [1–5] of 17 texts had a brief description of randomization and random sampling using random number tables [1–17]. Furthermore, neither SAS [18] nor SPSS [19] for the IBM personal computer had descriptions, although they did caution the reader to obtain data with proper randomization methods (not described). However, in none of these texts were there specific procedures for randomization of several kinds of sampling procedures, leaving the specific algorithm up to the ingenuity of the experimenter. Textbooks of experimental design and sampling discuss algorithms for treatment randomization and random sampling in more detail [20–25]. However, only a few general algorithms are available and none of these are implemented by computer.

According to McCall [1], "a simple random sample is one in which all elements of the population have an equal probability of being selected". His is one of the few statistical texts above [1–5] which directs one in using a random number table to get, for example, a sample of 50 students from a list of 3000. First, one assigns each student a number between 1 and 3000, then blocks off the table into four-digit columns and reads down the columns until 50 four-digit numbers are found that fall between 0001 and 3000 inclusive. Although this may be useful for some types of experiments, there are obviously a myriad of different situations where other algorithms, often more complex, are required to properly randomize sampling.

A.

```
10 CLS : DEFINT A-Z: PRINT "SIMPLE TREATMENT/POSITION RANDOMIZATION"
20 PRINT "ENTER NUMBER OF TREATMENTS/POSITIONS"; : INPUT N
30 PRINT "ENTER NUMBER OF REPLICATES"; : INPUT R
40 PRINT "ENTER A SPECIFIC NUMBER FOR A SPECIFIC RANDOM";
50 INPUT " SEQUENCE"; RN: DIM A(N, R): DIM B(N): RN = RND(-RN)
60 FOR Z = 1 TO R: FOR I = 1 TO N: B(I) = I: NEXT I: J = N
70 FOR I = 1 TO N: X = INT(RND * J + 1): A(I, Z) = B(X): J = J - 1
80 FOR K = X TO J: B(K) = B(K + 1): NEXT K: NEXT I: NEXT Z
90 PRINT "T     REPS"
100 FOR P = 1 TO N: PRINT CHR$(P + 64); : FOR E = 1 TO R
110 PRINT USING "###"; A(P, E); : NEXT E: PRINT : NEXT P
```

B.

```
SIMPLE TREATMENT/POSITION RANDOMIZATION
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  8  2  4  3  3  3  8  2  8  2  2  7  6  7  7
B  8  5  8  3  1  5  4  3  7  7  6  6  4  3  2  8
C  6  3  5  1  5  6  7  1  3  5  8  5  1  1  3  4
D  5  6  3  6  4  1  2  4  1  4  4  7  6  5  4  5
E  1  7  6  2  8  8  1  7  8  1  3  8  5  2  6  1
F  7  1  1  5  6  4  5  2  6  6  7  3  3  4  8  6
G  4  4  7  7  7  7  8  6  4  3  1  4  2  7  5  3
H  2  2  4  8  2  2  6  5  5  2  5  1  8  8  1  2
```

Fig. 1. (A) Listing of BASIC program used to randomly assign treatments to positions. (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 algorithms used by the version of BASIC).

Rohlf and Sokal [3] present a ten thousand random digit table which they say "is generally useful for a variety of sampling operations". However, "extended use of the same set of random numbers in the same sampling experiment is ill advised. In such cases, new random numbers should be looked up in a different table or preferably generated on the computer". They suggest that a table of random digits should be entered at random by choosing the page by a "random procedure", and determining the row and column by "blindly pointing" to it and then preceding in some "predetermined fashion, either horizontally or vertically." These examples indicate that random number tables are cumbersome to use as well as time consuming compared to computer generated random numbers. Furthermore, the sequence of random numbers in a table is usually from 2000 [2, 5] to 10 000 [3] and up to a maximum of 100 000 [26], while even a small home computer (ZX-81, Sinclair Research Ltd.) can generate 65 536 different numbers [27]. The random generator in QuickBASIC (MicroSoft®) can generate up to 10 million possible numbers and the sequence does not repeat for at least 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
30 PRINT C
```

The advantage of computer generation of random numbers compared to tables is that the former can be combined with algorithms for specific randomization and sampling applications, without error, and output to printer for a paper copy.

METHODS

The easy and efficient use of the following five randomization algorithms in BASIC (Figs 1–5) may help improve the design and reliability of analysis of scientific experiments. They operate in the BASIC language (BASICA, (C) 1983 or QuickBASIC 4.0,

A.

```
10 CLS : PRINT "LATIN SQUARE": DEFINT A-Z
20 INPUT "ENTER NUMBER OF TREATMENTS"; N: R = N: DIM A(N, R)
30 PRINT "ENTER A SPECIFIC NUMBER FOR A SPECIFIC RANDOM";
40 INPUT " SEQUENCE"; RN: DIM B(N): DIM C(N): DIM D(N)
50 RN = RND(-RN)
60 FOR I = 1 TO N: B(I) = I: C(I) = I: D(I) = I: NEXT I
70 J = N: REM RANDOMIZE ARRAY COLUMN
80 FOR I = 1 TO N
90 X = INT(RND * J + 1): B(I) = C(X): J = J - 1
100 FOR K = X TO J: C(K) = C(K + 1): NEXT K: NEXT I
110 J = N: REM RANDOMIZE ARRAY ROW
120 FOR I = 1 TO N: X = INT(RND * J + 1): C(I) = D(X): J = J - 1
130 FOR K = X TO J: D(K) = D(K + 1): NEXT K: NEXT I
140 FOR Z = 1 TO N: FOR W = 1 TO N: A(Z, W) = B(Z) + C(W)
150 IF A(Z, W) > N THEN A(Z, W) = A(Z, W) - N
160 PRINT USING "###"; A(Z, W); : NEXT: PRINT : NEXT
```

B.

```
LATIN SQUARE
ENTER NUMBER OF TREATMENTS? 8
ENTER A SPECIFIC NUMBER FOR A SPECIFIC RANDOM SEQUENCE? 3
  5  7  8  6  2  3  4  1
  4  6  7  5  1  2  3  8
  8  2  3  1  5  6  7  4
  2  4  5  3  7  8  1  6
  3  5  6  4  8  1  2  7
  1  3  4  2  6  7  8  5
  6  8  1  7  3  4  5  2
  7  1  2  8  4  5  6  3
```

Fig. 2. (A) Listing of BASIC program used to randomly assign treatments in a latin square. (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 algorithms used by the version of BASIC).

A.

```
10 CLS: PRINT "SELECTION OF NON-REPEATED NUMBERS FROM SAMPLE";
20 PRINT " SET": PRINT "ENTER A SPECIFIC NUMBER ";
30 INPUT "FOR A SPECIFIC RANDOM SEQUENCE"; RN: RN = RND(-RN)
40 INPUT "ENTER LOWEST NUMBER OF SAMPLE SET"; L
50 INPUT "ENTER HIGHEST NUMBER OF SAMPLE SET"; N
60 INPUT "ENTER NUMBER OF SAMPLE NUMBERS TO SELECT"; Q
70 IF Q>N-L+1 THEN PRINT "# OF SAMPLES > THAN RANGE": GOTO 40
80 N = N - L + 1: DIM A(Q): FOR W = 1 TO Q
90 A(W)=INT(RND*N)+L:FOR R=0 TO W-1:IF A(W)=A(R) THEN X=1
100 NEXT R: IF X = 1 THEN X = 0: GOTO 90
110 PRINT A(W); : NEXT W
```

B.

```
SELECTION OF NON-REPEATED NUMBERS FROM SAMPLE SET
ENTER A SPECIFIC NUMBER FOR A SPECIFIC RANDOM SEQUENCE? 3
ENTER LOWEST NUMBER OF SAMPLE SET? 101
ENTER HIGHEST NUMBER OF SAMPLE SET? 200
ENTER NUMBER OF SAMPLE NUMBERS TO SELECT? 10
 192  199  145  173  184  182  142  170  161  177
```

Fig. 3. (A) Listing of BASIC program used to randomly select non-repeated samples from a sample set. (B) Printed output of program on computer screen of an IBM PC or compatible computer (output may differ due to differences in the random generator algorithms used by the version of BASIC).

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
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 DIM A(R, C): DIM B(C): DIM C(C): DIM D(R): DIM E(R)
90 INPUT "ENTER A SPECIFIC NUMBER FOR A SPECIFIC RANDOM SEQUENCE"; RN
100 RN = RND(-RN)
110 FOR I = 1 TO C STEP C / N: M = M + 1: FOR K = 1 TO C / N: V = V + 1
120 C(V) = M: NEXT: NEXT: REM MAKE ARRAY OF TREATMENTS
130 J = C: M = 0: V = 0: REM RANDOMIZE COLUMN TREATMENT ARRAY
140 FOR I = 1 TO C
150 X = INT(RND * J + 1)
160 IF S = 1 AND I > 1 THEN IF B(I - 1) = C(X) THEN 150
170 B(I) = C(X): J = J - 1
180 FOR K = X TO J: C(K) = C(K + 1): NEXT K: NEXT I
200 FOR I = 1 TO R STEP R / N: M = M + 1: FOR K = 1 TO R / N: V = V + 1
210 E(V) = M: NEXT: NEXT: REM MAKE ARRAY OF TREATMENTS
220 J = R: REM RANDOMIZE ROW TREATMENT ARRAY
230 FOR I = 1 TO R
240 X = INT(RND * J + 1)
250 IF S = 1 AND I > 1 THEN IF D(I - 1) = E(X) THEN 240
260 D(I) = E(X): J = J - 1
270 FOR K = X TO J: E(K) = E(K + 1): NEXT K: NEXT I
280 FOR Z = 1 TO R: FOR W = 1 TO C: A(Z, W) = D(Z) + B(W)
290 IF A(Z, W) > N THEN A(Z, W) = A(Z, W) - N
300 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? 9
ENTER NUMBER OF ROWS? 9
ENTER NUMBER OF TREATMENTS? 3
ENTER 1 FOR SPACED TREATMENTS, 2 FOR NO SPACING? 1
ENTER A SPECIFIC NUMBER FOR A SPECIFIC RANDOM SEQUENCE? 7
  1  3  2  1  2  3  2  3  1        2  1  1  3  2  3  1  3  2
  3  2  1  3  1  2  1  2  3        1  3  3  2  1  2  3  2  1
  1  3  2  1  2  3  2  3  1        3  2  2  1  3  1  2  1  3
  3  2  1  3  1  2  1  2  3        3  2  2  1  3  1  2  1  3
  2  1  3  2  3  1  3  1  2        1  3  3  2  1  2  3  2  1
  3  2  1  3  1  2  1  2  3        3  2  2  1  3  1  2  1  3
  2  1  3  2  3  1  3  1  2        1  3  3  2  1  2  3  2  1
  1  3  2  1  2  3  2  3  1        2  1  1  3  2  3  1  3  2
  2  1  3  2  3  1  3  1  2        2  1  1  3  2  3  1  3  2
         SPACED                              NOT SPACED
```

Fig. 4. (A) Listing of BASIC program used to randomly assign treatments in equal proportions in a latin square-like grid with spacing or not between identical treatments. (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 algorithms used by the version of BASIC).

(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.

RESULTS AND DISCUSSION

Program 1: simple treatment/position randomization

The program (Fig. 1A) will generate a set of randomly selected treatments (one to any number of treatments) for this same number of positions for any number of experimental replicates. For example, Schlyter *et al.* [28] tested eight blends of different chemicals

A.

```
10 CLS:PRINT "3 TO 100 SPACED TREATMENTS IN EQUAL PROPORTIONS";
20 PRINT " IN ANY SIZE GRID":DEFINT A-Z
30 INPUT "ENTER NUMBER OF ROWS";R
40 INPUT "ENTER NUMBER OF COLUMNS";L
50 INPUT "ENTER 3 TO ANY NUMBER OF TREATMENTS";N
60 IF N>=3 AND N<=R*L THEN 80
70 PRINT "NUMBER OF TREATMENTS MUST BE >=3 OR <=";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:IF A(W+1,Z)=A(W,Z) THEN 160
170 IF A(W-1,Z)=A(W,Z) THEN 160
180 IF A(W,Z+1)=A(W,Z) THEN 160
190 IF A(W,Z-1)=A(W,Z) THEN 160
200 FOR D=1 TO N:IF A(W,Z)=D THEN C(D)=C(D)+1
210 NEXT D:NEXT Z:NEXT W
220 M=0
230 M=M+1:IF M=N+1 THEN 320
240 IF C(M)<R*L/N THEN P=M:C(M)=C(M)+1:GOTO 260
250 GOTO 230
260 J=J+1:IF C(J)>R*L/N THEN C(J)=C(J)-1:Q=J:J=0:GOTO 290
270 IF J=N THEN J=0:GOTO 290
280 GOTO 260
290 FOR W=1 TO R:FOR Z=1 TO L:IF T=1 THEN 310
300 IF A(W,Z)=Q THEN GOSUB 340
310 NEXT Z:NEXT W:T=0:GOTO 220
320 FOR W=1 TO R:FOR Z=1 TO L:PRINT USING "###";A(W, Z);
330 NEXT Z:PRINT:NEXT W:END
340 IF A(W+1,Z)=P THEN RETURN
350 IF A(W-1,Z)=P THEN RETURN
360 IF A(W,Z+1)=P THEN RETURN
370 IF A(W,Z-1)=P THEN RETURN
380 A(W,Z)=P:T=1:RETURN
```

B.

```
3 TO 100 SPACED TREATMENTS IN EQUAL PROPORTIONS IN ANY SIZE GRID
ENTER NUMBER OF ROWS? 10
ENTER NUMBER OF COLUMNS? 10
ENTER 3 TO ANY NUMBER OF TREATMENTS? 5
ENTER A SPECIFIC NUMBER FOR A SPECIFIC RANDOM SEQUENCE? 5
  3  5  4  5  4  3  5  4  2  5
  5  2  5  4  1  5  4  2  4  2
  3  1  3  1  3  1  2  1  3  4
  2  4  2  3  1  4  3  4  1  2
  4  2  4  5  2  3  2  5  4  3
  2  1  5  3  1  5  4  1  3  2
  3  5  3  5  3  2  1  4  5  1
  5  3  1  4  2  4  2  1  3  2
  1  2  5  1  3  5  4  5  4  1
  5  3  1  5  1  2  1  4  3  2
```

Fig. 5. (A) Listing of BASIC program used to randomly assign treatments (3–100 different types) in equal proportions in any size grid. (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 algorithms used by the version of BASIC).

found in the hindguts of the European spruce bark beetle, *Ips typographus*, in order to determine which of the compounds were essential for aggregation on the host tree.

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

treatment the computer picks at random a number (representing the position) from an array (such that the number is no longer available for selection) until all replicates have positions selected for the treatments. Output of the program for eight treatments (A–H) and positions (1–8) replicated 16 times is shown in Fig. 1B. Each of the blends (A–H) would then be placed in the designated position (1–8) for each replicate time period (usually for a few hours or one day) and catch recorded for each treatment/position during each replicate. The advantage of this method is that fewer traps need to be attended while the disadvantage is that the replicates often occur under different environmental conditions (since the first and last replicates could be separated by 16 days in this case).

Program 2: latin square

This program (Fig. 2A) generates any size latin square in which all treatments each occur once in each row and column. It is well known that the spatial distribution of flying bark beetles varies greatly as observed with monitor traps [29, 30]. A latin square arrangement can be used in an attempt to even out the variation between trap positions so that treatment effects can be properly observed. The tests of Schlyter *et al.* [28] could also have been performed using an 8×8 latin square with each column representing a replicate time and each row a position and the number corresponding to each of the 8 treatments. Alternatively, the experiment could have been done simultaneously using 64 traps in a grid.

The number of either the columns, rows, or treatments, which are all equal, is entered along with the random seed number. The algorithm uses the method of randomizing numbers by treatment/position in program 1 for two such arrays of treatments (column and row). The column and row arrays (e.g. 87356412 and 57862341) can then be used to calculate a unique latin square of (8 column × 8 row) cell elements. Each column and row intersection, cell(c, r), of the latin square can be obtained from the sum of the numbers in the respective column and row arrays (e.g. cell(1, 1) = 8 + 5 = 13). However, if the sum is more than the number of columns (in this case 8) then the sum has the number of columns subtracted (e.g. cell(1, 1) = 13 − 8 = 5). This method was first presented by Stevens [31] and Bose [32]. Output (Fig. 2B) is shown with the replicates suggested across the rows, the positions across the columns, and the treatments represented by the number in each cell. The rows × columns can also be considered only as positions with eight possible treatments.

Program 3: selection of non-repeated numbers from a sample set

This program (Fig. 3A) selects a certain number of integer numbers at random from within a sample set beginning and ending at specified numbers such that no two selected numbers are the same. To illustrate, one can suppose that 1000 trees have been tagged in 10 plots of 100 trees each. One of the plots is selected for further study (trees 101–200) and it is desired that 10 trees be selected at random for intensive sampling of bark beetle attack patterns.

One enters the lower (101) and upper numbers (200) of the sample set and the number of selections as well as the random seed. The computer then picks a random number from the sample set and checks to see if this number matches any previously selected numbers. If a match occurs, then the current number is discarded and another number selected, which, if no match is found, then proceeds to the next selection until the number of requested selections has been obtained. The computer outputs these numbers (Fig. 3B) in the sequence of selection so that they can be used for both random sequences or as random samples. The advantage of the algorithm over the one used in program 1 is that a memory array need not hold the population, which may then be very large, but instead need only hold the numbers selected.

Program 4: any number of treatments equally proportioned in a 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 treatments are replicated equally to the extent allowed by the places in the column and row. Identical treatments may occur side by side or can be spaced so that no vertical or horizontal positions can be filled by the same treatment. The columns and the rows must be evenly divisible by the number of treatments. Byers *et al.* [33] released the aggregation pheromone components of the bark beetle *Pityogenes chalcographus* at three rates over two orders of magnitude in the forest. Again, in order to reduce variation due to spatial heterogeneity in the forest one can proportion the treatments among the columns and rows (either through time or space).

One enters the number of columns (9), rows (9), 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 input. The algorithm is similar to that used in program 2 in which a row and a column array are randomized, but in this case with appropriate repetitions of the treatments. In the option for the spacing of identical treatments, the algorithm must check 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 the two arrays are randomized then the addition of the arrays proceeds as in program 2. Output (Fig. 4B) shows the treatments evenly proportioned across the rows (replicates) and columns (positions). The rows × columns can also be considered as positions only.

Program 5: three to 100 spaced treatments in equal proportions in any size grid

This program (Fig. 5A) is used to more evenly space treatments in a grid of positions so that the same treatments never stand either "horizontally or vertically" next to each other, although they may stand diagonally. This constraint tends to space treatments in a more uniform distribution even though they are still fairly random. A trivial case of alternation occurs with just two treatments. The program can also be used for spaced sampling. For example, a grid of 20 × 20 could be alloted 10 "treatments" (40 positions for each) of which the positions of treatment five are sampled. The program can be modified (lines 10, 110 and 140) to allow even more than 100 treatment types.

To illustrate, Byers *et al.* [34] caught the bark beetle *Tomicus piniperda* when attracted to traps containing four treatments of host pine logs with either no beetles, males only, females only, or both sexes as well as a blank trap. In Scandinavia, *T. piniperda* seeks hosts once a year over a relatively short period of time (often for only one or two days) compared to many other bark beetles. Thus, it would be desirable to test all treatments and replicates simultaneously in a grid. Furthermore, one is more likely to authenticate a hypothesis if the adverse effects of area-wide variations in beetle density, discussed above, are minimized by spacing the treatments in the grid for a more uniform, but still random, distribution throughout the area.

One enters the number of rows (10), columns (10), and treatment types (5) as well as the random seed. If the number of rows times columns is not evenly divisible by the number of treatments then equal proportions cannot be obtained so the computer would request new input. Once the input conditions are satisfied, the program picks integer random numbers, from one to the number of treatments, inclusive, for each column position of a row. Each treatment number selected is compared to the respective grid position values above and below as well as to the sides. If any match is found then another number is randomly selected and used in its place. The adjacent locations are again compared until no match occurs. The sum of this type of treatment is incremented by one to obtain running totals of each treatment type, and then the next column position is processed, or if at the row's end, then the first position of the next row. After all positions have a treatment selected for them it is probable that, by chance, the treatment types are not equally proportioned.

Thus, another algorithm (beginning on line 220, Fig. 5A) uses the running totals of treatment types to determine a treatment type which has less occurrences in the grid than an even share. The computer then searches the grid to find a treatment type which has more occurrences and replaces it with the first treatment type which so far is under-represented, but only if the adjacent comparisons do not match, otherwise another prospective location is found and the procedure repeated. Once a suitable location is found and treatments replaced, the running totals are appropriately changed and the whole process repeated if necessary until all running totals of the treatment types are equal, whereupon the program ends. Output (Fig. 5B) for a 10×10 grid of five treatments gives 20 of each treatment type, and none of these are horizontally or vertically adjacent to identical types. This spacing of treatments, while retaining a degree of randomness, would be desirable when it is expected that positional effects on variation may be larger than treatment differences. Thus, one does not want a particular set of treatments only in one area but in a diversity of areas as accomplished by program 5, or the latin square type programs (programs 2 and 4).

Experimental design software packages using expert systems are just now becoming available for aiding the experimenter [35]. For a more thorough discussion of the use of latin square and other randomization methods there are several textbooks on experimental design and sampling [20–25]. It is hoped that these five BASIC algorithms for random sampling and randomization of treatments will be extensively used by biologists to improve the validity of our scientific inferences and conclusions.

SUMMARY

Five 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 randomized blocks, latin squares, simple sampling without replacement, and some factorial blocks.

Acknowledgements—This work was supported by a cooperative research grant from the U.S.A. (NSF-INT-8503520, Department of Entomological Sciences, University of California, Berkeley, CA) and Sweden (STU-84-3937). I thank my colleagues, Olle Anderbrant and Fredrik Schlyter, as well as several anonymous reviewers, for improvements to the manuscript.

REFERENCES

1. R. B. McCall, *Fundamental Statistics for Psychology*. Harcourt, Brace and World, New York (1970).
2. E. A. Robinson, *Statistical Reasoning and Decision Making*. Goose Pond Press, Houston, TX (1981).
3. F. J. Rohlf and R. R. Sokal, *Statistical Tables*. Freeman, San Francisco, CA (1969).
4. M. R. Spiegel, *Theory and Problems of Statistics*. McGraw-Hill, New York (1961).
5. T. H. Wonnacot and R. J. Wonnacot, *Introductory Statistics for Business and Economics*. Wiley, New York (1972).
6. A. A. Afifi and S. P. Azen, *Statistical Analysis: A Computer Oriented Approach*. Academic Press, New York (1972).
7. R. M. Bethea, B. S. Duran and T. L. Boullion, *Statistical Methods for Engineers and Scientists*. Marcel Dekker, New York (1985).
8. J. Gani, *Perspectives in Probability and Statistics*. Academic Press, New York (1975).
9. S. J. Haberman, *Analysis of Qualitative Data*, Vols I and II. Academic Press, New York (1978).
10. E. A. Hanushek and J. E. Jackson, *Statistical Methods for Social Scientists*. Academic Press, New York (1977).
11. R. Langley, *Practical Statistics: Simply Explained*. Dover, New York (1968).
12. E. L. Lehmann and H. J. M. D'Abrera, *Nonparametrics: Statistical Methods Based on Ranks*. McGraw-Hill, New York (1975).
13. W. Mendenhall, *Introduction to Probability and Statistics*. Wadsworth, Belmont, CA (1969).
14. S. Siegel, *Nonparametric Statistics: For the Behavioral Sciences*. McGraw-Hill, New York (1956).
15. R. R. Sokal and F. J. Rohlf, *Biometry: The Principles and Practice of Statistics in Biological Research*. Freeman, San Francisco, CA (1969).
16. H. G. Tucker, *An Introduction to Probability and Mathematical Statistics*. Academic Press, New York (1962).
17. S. Zacks, *Parametric Statistical Inference*. Pergamon Press, Oxford (1981).
18. SAS Institute, *SAS Procedures Guide for Personal Computers*, 6th Edn. SAS Institute, Cary, NC (1985).
19. M. J. Norusis, *SPSS/PC+ for the IBM PC/XT/AT*. SPSS, Chicago, IL (1986).

20. W. G. Cochran, *Sampling Techniques*. Wiley, New York (1963).
21. W. G. Cochran and G. M. Cox, *Experimental Designs*. Wiley, New York (1957).
22. D. R. Cox, *Planning of Experiments*. New York (1958).
23. M. N. Das and N. C. Giri, *Design and Analysis of Experiments*. Wiley, New York (1979).
24. W. E. Deming, *Some Theory of Sampling*. Dover, New York (1950).
25. O. Kempthorne, *The Design and Analysis of Experiments*. Wiley, New York (1952).
26. R. A. Fisher and F. Yates, *Statistical Tables for Biological, Agricultural and Medical Research*. Oliver and Boyd, Edinburgh (1948).
27. S. Vickers, *ZX81 BASIC Programming*. Sinclair Research Ltd., London (1980).
28. F. Schlyter, G. Birgersson, J. A. Byers, J. Löfqvist and G. Bergström, Field response of spruce bark beetle, *Ips typographus*, to aggregation pheromone candidates. *J. Chem. Ecol.* **13**, 701 (1987).
29. 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).
30. 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).
31. W. L. Stevens, The completely orthogonalised square. *Ann. Eug.* **9**, 82 (1938).
32. R. C. Bose, On the application of Galois fields to the problem of the construction of Hyper-Graeco-Latin squares. *Sankhya* **3**, 323 (1938).
33. J. A. Byers, G. Birgersson, J. Löfqvist and G. Bergström, Synergistic pheromones and monoterpenes enable aggregation and host recognition by a bark beetle. *Naturwissenschaften* **75**, 153 (1988).
34. J. A. Byers, B. S. Lanne, J. Löfqvist, F. Schlyter and G. Bergström, Olfactory recognition of host-tree susceptibility by pine shoot beetles. *Naturwissenschaften* **72**, 324 (1985).
35. P. Marsh, Designing experiments using IBM-PC's. *Statistician's Microcomputer Users' Group* **5**, 3 (1988).

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) from the University of California at Berkeley in Entomology. He is currently a Docent (Associate Professor) of Animal Ecology in the Insect Pheromone Group, Lund University, Sweden. His main research interests are in insect behavioral and chemical ecology and in computer simulation of behavioral and ecological models.