A SAS macro to generate patient/subject random allocation schedules for clinical studies, including blocked or unblocked trials, is presented. The macro can be used to generate allocation schedules for a total of 12 experimental units. For example, the allocation schedule for a study of n experimental units (patients, subjects, etc.) requires n/t of the numbers 1, 2, ..., n randomly assigned to treatment group 1, n/t assigned to treatment group 2, etc. Studies are blocked by certain factors which require each subgroup of size b (such that n is a multiple of b) to be equally divided among the t treatments. The random allocation schedule described above is equivalent to a truly random allocation schedule. Theoretical justification of the "randomness" of the rank transformation is given and verified by a Monte Carlo experiment.

**Examples**

**Example 1.** Suppose we need a random allocation schedule for a total of 12 experimental units which are to receive 1 of 3 treatments. We want blocks of size 6 and the allocation numbers (ANs) to start with 1. Table 1 shows the results of executing the algorithm with n = 12, t = 3, b = 6 and s = 1 as described.

<table>
<thead>
<tr>
<th>AN</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
</tr>
<tr>
<td>2</td>
<td>.57</td>
<td>.43</td>
<td>.79</td>
<td>.57</td>
<td>.03</td>
<td>.41</td>
<td>.63</td>
<td>.81</td>
<td>.17</td>
<td>.88</td>
<td>.58</td>
<td>.09</td>
</tr>
<tr>
<td>3</td>
<td>Block(s)</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>R(x_i)</td>
<td>5</td>
<td>3</td>
<td>5</td>
<td>4</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>5</td>
<td>2</td>
<td>6</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>Group(k)</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>6</td>
<td>AN</td>
<td>5</td>
<td>3</td>
<td>5</td>
<td>4</td>
<td>1</td>
<td>2</td>
<td>10</td>
<td>11</td>
<td>8</td>
<td>12</td>
<td>9</td>
</tr>
<tr>
<td>7</td>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**THEORETICAL JUSTIFICATION**

In order to assure that this algorithm yields a truly random allocation schedule, we must show that the probability of any particular schedule given n, t, b, and s is the multiplicative inverse of the number of possible schedules. Without loss of generality we can take s = 1 since the schedule for any s is the same as the schedule for s = 1 with s = 1 added to each AN. Also, we can take b = n since the algorithm applies to each group of size b. Therefore, all we need show is that given a random sample of size n from the uniform distribution on (0,1), the probability of any given rank-order occurring is the same as that of any other rank-order occurring, i.e., 1/n.

It can easily be proved by induction that

\[ dx_n = \int_{x_{n-1}}^{x_n} \frac{1}{n!} dx_i \]

Taking s = 1, equation (1) is equivalent to

\[ P(x_1 < x_2 < \ldots < x_n) = \frac{1}{n!} \]

The probability of any rank-order is \( n! \) by appropriately changing the order of the \( x_i \)'s in equation (1).

**EMPIRICAL JUSTIFICATION**

Since a random sample from a uniform distribution yields a random rank-order of the sample, the randomness of allocation schedules generated by the algorithm described above depends on that of the uniform sample obtained (from SAS function UNIFORM). In order to empirically justify that the algorithm yields random allocation schedules, a Monte Carlo experiment was run.

A random sample of size 4 from the uniform distribution on (0,1) has a total of 24 possible
These are equally likely. To verify this empirically, 500 goodness of fit chi squares with 23 degrees of freedom were each calculated from counts of the rank-orders resulting from 24 random samples of size 4 from SAS UNIFORM function. These 500 chi squares represented a random sample from the chi square distribution with 23 degrees of freedom and were tested for goodness of fit, Ho: $\mu = 23$, and Ho: $\sigma^2 = 46$ (recall that the mean and variance of the chi square distribution are the d.f. and twice the d.f., respectively). This was done for 10 sets of 500 such squares. Some tail categories of the goodness of fit chi squares were trimmed because they severely inflated the chi square. In each of the 10 runs, the fit was very good except in the extreme upper tail categories in some cases (categories were 1, 2, 3, ..., max chi square observed). All means were not significantly different (p > .05) from 23, and only 2 of 10 variances was significantly different (p < .05) from 46. The Monte Carlo results are shown in Table 2.

<table>
<thead>
<tr>
<th>Run</th>
<th>p</th>
<th>$\delta^2$</th>
<th>p: Goodness of Fit</th>
<th>Tail Categories Excluded</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>23.5</td>
<td>.13</td>
<td>55.5</td>
<td>.001</td>
</tr>
<tr>
<td>2</td>
<td>23.3</td>
<td>.25</td>
<td>43.1</td>
<td>.025</td>
</tr>
<tr>
<td>3</td>
<td>23.0</td>
<td>.25</td>
<td>49.6</td>
<td>.025</td>
</tr>
<tr>
<td>4</td>
<td>23.0</td>
<td>.25</td>
<td>50.9</td>
<td>.029</td>
</tr>
<tr>
<td>5</td>
<td>23.2</td>
<td>.25</td>
<td>44.2</td>
<td>.028</td>
</tr>
<tr>
<td>6</td>
<td>23.0</td>
<td>.25</td>
<td>46.4</td>
<td>.028</td>
</tr>
<tr>
<td>7</td>
<td>23.0</td>
<td>.25</td>
<td>47.2</td>
<td>.026</td>
</tr>
<tr>
<td>8</td>
<td>23.1</td>
<td>.25</td>
<td>46.1</td>
<td>.025</td>
</tr>
<tr>
<td>9</td>
<td>23.6</td>
<td>.06</td>
<td>45.0</td>
<td>.025</td>
</tr>
<tr>
<td>10</td>
<td>22.7</td>
<td>.28</td>
<td>44.6</td>
<td>.025</td>
</tr>
</tbody>
</table>

*All in upper tail

This empirical evidence gives us confidence that the use of the SAS UNIFORM function and the algorithm presented herein will yield valid random allocation schedules in practice.

**USER'S INSTRUCTIONS**

The SAS macro ALLOC is used to generate random allocation schedules as described above. Following the macro in a SAS program, the user need only add one line of SAS instructions for each schedule desired. The line to be added is

```
DATA ALLOCIN; S = s; N = n; T = t; BS = b; NAPL = 1; OUTPUT; ALLOC
```

where

- $s$ = the smallest allocation number desired (the starting number),
- $n$ = the total number of allocation numbers required,
- $t$ = the number of treatments (groups),
- $b$ = the block size (number of allocation numbers per block), and
- $l$ = the number of allocation numbers to be printed per line of output.

For subsequent allocation schedules, this line can be repeated once for each additional schedule desired. If titles are needed, they can be inserted immediately following the added line.

The call for ALLOC must be moved to follow the title statement (see DISPLAY 1). When titles are used, the user should use "OPTIONS NOCENTER;" prior to calling ALLOC because ALLOC does not center its output.

Note that this macro provides only balanced allocation schedules, therefore, the following restrictions apply:
1) $n$ must be a multiple of $t$,
2) $n$ must be a multiple of $b$, and
3) $b$ must be a multiple of $t$.

The macro is listed in lines 100-8500 of DISPLAY 1 and contains documentation of the programming conventions. Lines 8600-9000 of DISPLAY 1 give samples of user added instructions for examples of schedules. DISPLAY 2 shows the random allocation schedules that result from execution of the SAS program in DISPLAY 1.

**ACKNOWLEDGEMENTS**

I am grateful to TOM COOK for pointing out the reason for an error I had made in my initial Monte Carlo verification of the algorithm. I also thank MARK KERSTEN for her typing this manuscript and the preliminary draft.

**REFERENCE**

DISPLAY 1

MACRO ALLOC DATA ALLOCIN; SET ALLOCOUT;

* THIS MACRO PRODUCES BALANCED RANDOM ALLOCATION SCHEDULES.
* INPUT IS A DATASET ALLOCIN WITH ONE OBSERVATION AND 4 VARIABLES
* IN THIS ORDER: STARTING ALLOCATION NUMBER (AN)
* NUMBER OF AN'S NEEDED
* THRESHOLD OF TREATMENT GROUPS
* BS-BLOCK SIZE
* NAPL=NUMBER OF AN'S PRINTED PER LINE OF OUTPUT
* THE REMAINING COMMENTS DOCUMENT THE PROGRAM

; * CHECK FOR IMBALANCED INPUT, SET ERROR FLAG, AND WARN USER
; IF NAPL#8 THEN NAPL=8; IF MOD(N,Ti) NE 0 THEN DO; FLAG=1;
; PUT 'ERROR: H NOT A MULTIPLE OF T, TRY AGAIN'; END;
; IF MOD(N,BS) NE 0 THEN DO; FLAG=1;
; PUT 'ERROR: H NOT A MULTIPLE OF T, TRY AGAIN'; END;
; IF MOD(ES,Ti) NE 0 THEN DO; FLAG=1;
; PUT 'ERROR: H NOT A MULTIPLE OF T, TRY AGAIN'; END;
; PROC MATRIX; FETCH Z DATA=ALLOCIN;

EXIT: IF Z(1,6)=1 THEN DO; E=JI1.4.1); OUTPUT E OUT=ALLOCOUT; END;

GENERATE RANDOM SAMPLE FROM UNIFORH (0,1)
RANK WITHIN EACH BLOCK AND MAKE AN'S
TITLE1 ALLOCATION SCHEOULE; TITlE2 FOR EXAMPLE 1
LOC ERROR FLAG SET, SKIP CALCULATIONS. GO TO NEXT STEP
IN THIS ORDER! S=STARTING ALLOCATION NUMBER (AN)
INPUT IS A DATASET ALLOCIN
DO I ,. 1 TO MS; INDEX=(I-1)*BS+RANK(A(I,INDEX)); END;
A!l,INDEX)=(A(I,index)+NAPL*Z(I,5));
A=(A-I,N,0);

* GENERATE RANDOM SAMPLE FROM UNIFORM (0,1)
* DO I = 1 TO N; A(I,1)=UNIFORM(6); END;

* RANK WITHIN EACH BLOCK AND MAKE AN'S
* DO I = 1 TO N; HS=INDEX(I)*(-1)*BS+BS*MS);
* A(I,INDEX)=INDEX+HS=RANK(A(I,INDEX)); END;

* ASSIGN STARTING NUMBER AND ORDER THE AN'S FOR OUTPUT
* A=1+SHAPE(A,T);
* DO I = 1 TO N; A(I,1)=SHAPE(A(I,1)); A(I,2)=SHAPE(A(I,2));
* DO K = 1 TO N; ROW(A(I)); A(Row(A(I),1)); Bعرف(A(I),1)); END; END;

* RESHAPE INTO VECTOR FOR OUTPUT AND CALC MAX DIGITS PER AN
* A=SHAPE(A,1); NAPL=UNIFORM(1,NAPL); NAPL=UNIFORM(1,1,0);
* H=INT(NAPL*0.90000001+LOG10(MAX(A(I)))), MP=UNIFORM(1,1,0);

* FORM OUTPUT MATRIX WITH CONTROLS AND OUTPUT FOR PRINTING
* H=MAX(1,MP)|A|MP; OUTPUT H OUT=ALLOUT;

* IF ERROR FLAG SET, SKIP ERRORS FOR PRINTING SECTION
* EXIT: IF Z(1,6)=1 THEN DO; E=JI1.4.1); OUTPUT E OUT=ALLOUT; END;
DATA _NULL_ SET ALLOCOUT;

* PRINT HEADING FOR EACH TREATMENT GROUP
* IF MOD(COL,15,0) THEN DO; G=COL/15,0; PUT // 81 'GROUP'; BY G 15.0; END;

* CALC NUMBER OF SPACES TO SKIP FOR UNIFORM PRINTING OF AN'S
* N=COL-INT(NAPL*0.0000001+LOG10(COL3));

* CHECK FOR FIRST AN IN EACH TREATMENT GROUP AND SET COUNTER
* IF MOD(COL,15,0) THEN NH=1;

* PRINT AN EACH PUT STTH DEPENDING ON NUMBER OF AN'S PER LINE OF OUTPUT
* N=11 IF N NE COL THEN PUT AN COLS 0;
* ELSE DO: PUT -H COLS 0; NH=RETAIN 0;

* DELETE DATASETS TO CLEAR WORK SPACE
* PROC DELETE DATA=ALLOCIN ALLOUT;

500 600 700 800 900 1000 1100 1200 1300 1400 1500 1600 1700 1800 1900 2000
100 200 300 400 500 600 700 800 900 1000 1100 1200 1300 1400 1500 1600 1700 1800 1900 2000

DISPLAY 2

ALLOCATION SCHEDULE FOR EXAMPLE 1

GROUP 1
3 6
10 11

GROUP 2
4 5
8 12

GROUP 3
1 2
7 9

ANOTHER EXAMPLE
S=101 N=150 T5 BS=10 NAPL=10

GROUP 1
104 107 111 114 124 125 131 135 145 158
158 165 167 172 160 167 173 169 178 189
206 208 211 220 223 237 235 236 242 249

GROUP 2
108 110 118 120 121 123 136 140 141 146
157 160 162 166 176 178 182 190 194 197
202 210 217 219 225 226 231 233 241 244

GROUP 3
101 109 113 119 129 130 137 146 148
153 159 160 164 170 179 182 187 192 199
205 207 215 218 221 226 234 240 253 265

GROUP 4
103 109 112 116 120 126 132 138 142 149
151 152 164 170 175 177 186 190 198 205
209 212 216 222 224 236 238 257 250

GROUP 5
102 206 215 217 222 227 231 236 246 247
156 255 261 263 272 275 283 286 193 195
201 204 213 216 229 237 239 249 256 269