Alpha Lattice Designs

(1) These are incomplete block designs (sometimes balanced) that are resolvable.

Resolvable: Incomplete blocks group together into "super-blocks" that are complete.

Example of a resolvable design for 9 treatments: Graeco-Latin square (table of these Appendix 6 B)

$$\begin{pmatrix} A & B & C \\ B & C & A \\ C & A & B \end{pmatrix} \begin{pmatrix} A & B & C \\ C & A & B \\ B & C & A \end{pmatrix} \quad trts: T = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

Split up treatments by rows, columns, square 1 letters and square 2 letters (4 superblocks)

T			Letters	
Rows		Cols	Sq 1	Sq 2
$\begin{pmatrix}1\\2\\3\end{pmatrix}\begin{pmatrix}4\\5\\6\end{pmatrix}\begin{pmatrix}7\\8\\9\end{pmatrix}$		$\begin{pmatrix}1\\4\\7\end{pmatrix}\begin{pmatrix}2\\5\\8\end{pmatrix}\begin{pmatrix}3\\6\\9\end{pmatrix}$	$\begin{pmatrix}1\\6\\8\end{pmatrix}\begin{pmatrix}2\\4\\9\end{pmatrix}\begin{pmatrix}3\\5\\7\end{pmatrix}$	$\begin{pmatrix}1\\5\\9\end{pmatrix}\begin{pmatrix}2\\6\\7\end{pmatrix}\begin{pmatrix}3\\4\\8\end{pmatrix}$

Note: (optional) for p a prime number, there are p-1 orthogonal pxp Latin squares that you can construct as shown on page 155. A simplified version follows:

Let i and j represent the row and column number of an entry in square k, k=1 to p-1.

Square k (1,2,…,p-1) has i j entry (1 through p) given by k(i-1) + (j-1) reduced modulo p.

Square 1 (k=1), k(i-1)+(j-1) = (i-1)+(j-1).
row i=1 (i-1=0) has entries 1(0)+(j-1) = j-1 =0 1 2,
row i=2 has entries 1(2-1)+(j-1) = 1+(j-1) =1 2 0,
row i=3 has entries 1(3-1)+(j-1) = 2+(j-1) = 2 0 1.

Square 2 has (i,j,) entry 2(i-1)+(j-1).
Rows: (j-1): 0 1 2, 2+(j-1) :2 0 1, 4+(j-1) :1 2 0

(2) Alpha Lattices are resolvable designs with treatments assigned by cyclic permutations of the treatment numbers.

Example of cyclic permutation, t=5 treatments (0,1,2,3,4) in incomplete blocks of size b=3:
Write down any 3 treatments: (1 3 4)
Keep adding 1 (mod t)

2 4 0
3 0 1
4 1 2
0 2 3
1 3 4

Eventually (t+1=6 steps) you have to get back to the original pattern (include that one only once).

Rows form the incomplete blocks of the lattice. This one is not well balanced. The pairs (0,2) and (0,3) occur in two blocks each whereas the pair (0,4) appears only once.

You can see that cyclic designs exist for any number of treatments and any block size. Their properties (balance and efficiency for comparing means) may be good or not so good.

(3) The alpha cyclic lattices are obtained by cyclic incrementing of initial arrays. The initial arrays can be found in G&G appendix 8A.4 and elsewhere (e.g. Patterson et al 1978 – see text)

The basic case has the restriction that we have t =ks treatments in s blocks of size k. These would generally not be used for a small number of treatments. They are obtained from initial tables by the kind of permutation shown above (please see section 8.7 for more information on properties)

Example: k=4, t=20, so s=5 blocks (of size k=4)
Now we would also want replication so let's use r=3 replicates.

(A) Go to the table appendix 8A.4 (top array because s=5)

This appears:

0 0 0 0 0
0 1 2 3 4
0 4 3 2 1
0 2 4 1 3

(B) We want r=3 replicates so we use the first 3 rows. We have k=4 so we use only the first 4 columns. This is our "alpha array" with elements α_{ij}
0 0 0 0 row 1
0 1 2 3 row 2
0 4 3 2 row 3

From these elements α_{ij}, create the "A-array" by replacing α_{ij} with $a_{ij} = k\alpha_{ij}+(j-1)$ modulo t
[= $4\alpha_{ij}+(j-1)$ modulo 20 in our example].

Our A-array is thus:
0 1 2 3 ← 0 + (j-1)
0 5 10 15 ← $4\alpha+(j-1)$ for row 2 α values
0 17 14 11 ← $4\alpha+(j-1)$ for row 3 α values

(C) The final step: Generate a set of blocks from each row of the A-array by cyclic permutation, incrementing by k (4 in our case) modulo t. Here are the super-sets (rows are the blocks):

```
* demo Alpha1.sas ;
Data Cyclic;
input t1 t2 t3 t4; k=4; s=5; t=k*s;
set+1;
do i=1 to s;
t1=mod(t1+k,t);
t2=mod(t2+k,t);
t3=mod(t3+k,t);
t4=mod(t4+k,t);
block+1;
output;
end;
cards;
0 1 2 3
0 5 10 15
0 17 14 11
;
proc print noobs;
var t1 t2 t3 t4 block; by set;
proc transpose data=Cyclic out=trans (rename =
(col1=trt) drop=_NAME_ ) ;
var t1-t4; by set block;
data design; set trans; trt=trt+1;
             *start trts at 1 not 0; ;
proc print data=design; by set; run;
proc freq;
table trt*set/norow nocol nopercent; run;
```

```
*********output*************

--------------------------- set=1 ---------------------------

                t1    t2    t3    t4    block

                 4     5     6     7      1
                 8     9    10    11      2
                12    13    14    15      3
                16    17    18    19      4
                 0     1     2     3      5

------------------------ set=2 -----------------------------

                t1    t2    t3    t4    block

                 4     9    14    19       6
                 8    13    18     3       7
                12    17     2     7       8
                16     1     6    11       9
                 0     5    10    15      10

--------------------------- set=3 ---------------------------

                t1    t2    t3    t4    block

                 4     1    18    15      11
                 8     5     2    19      12
                12     9     6     3      13
                16    13    10     7      14
                 0    17    14    11      15
```

Now suppose we have less treatments, say 17. Pick a column (say the second with 1, 5, 9, 13, 17) and eliminate 3 of the treatments (say 1, 5, 9) from all the blocks. Note that we now have unequal block sizes. In practice, picking the smallest ks>t (18>17 in our case) to generate the original sets would be preferred.

Continuing with our demos, Alpha1A.sas generates an “incidence matrix” with a row for each treatment and column for each block. The ijth entry is 1 if treatment i occurs in block j, 0 otherwise. Compute $C=AA'$. The ij^{th} entry here is the number of co-occurrences (in the same block) of treatments i and j.

Block1: (1, 3, 4), Block 2: (2, 3, 4).

$$A=\begin{pmatrix}1&0\\0&1\\1&1\\1&1\end{pmatrix},\quad C=AA'=\begin{pmatrix}1&0\\0&1\\1&1\\1&1\end{pmatrix}\begin{pmatrix}1&0&1&1\\0&1&1&1\end{pmatrix}=\begin{pmatrix}1&0&1&1\\0&1&1&1\\1&1&2&2\\1&1&2&2\end{pmatrix}$$

* Demo Alpha1A.sas;

```
proc iml; reset fw=3;
 use design;
 read all var{trt block} into comb;
 print comb;
 a=shape(0,20,15);
 do i=1 to 60;
 a[comb[i,1],comb[i,2]] = 1;
 end;
 label = {" 1" " 2" " 3" " 4" " 5" " 6" " 7" " 8" " 9" "10"
   "11" "12" "13" "14" "15" "16" "17" "18" "19" "20"};
 print a [rowname=label colname=label]; c = a*a`;
 print c [rowname=label colname=label];
 rep=vecdiag(c);       * extract diagonal;
 print rep;
```

******* partial output (C) **********

	C																		
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
1	3	1	1	1	0	1	0	0	0	0	1	1	0	0	1	1	0	1	0	0
2	1	3	1	1	1	0	1	0	0	0	0	1	0	0	0	1	1	0	1	0
3	1	1	3	1	0	1	0	1	1	0	0	0	1	0	0	0	0	1	0	1
4	1	1	1	3	0	0	1	0	1	1	0	0	1	1	0	0	0	0	1	0
5	0	1	0	0	3	1	1	1	0	1	0	0	0	0	1	1	0	0	1	1
6	1	0	1	0	1	3	1	1	1	0	1	0	0	0	0	1	0	0	0	1
7	0	1	0	1	1	1	3	1	0	1	0	1	1	0	0	0	1	0	0	0
8	0	0	1	0	1	1	1	3	0	0	1	0	1	1	0	0	1	1	0	0
9	0	0	1	1	0	1	0	0	3	1	1	1	0	1	0	0	0	0	1	1
10	0	0	0	1	1	0	1	0	1	3	1	1	1	0	1	0	0	0	0	1
11	1	0	0	0	0	1	0	1	1	1	3	1	0	1	0	1	1	0	0	0
12	1	1	0	0	0	0	1	0	1	1	1	3	0	0	1	0	1	1	0	0
13	0	0	1	1	0	0	1	1	0	1	0	0	3	1	1	1	0	1	0	0
14	0	0	0	1	0	0	0	1	1	0	1	0	1	3	1	1	1	0	1	0
15	1	0	0	0	1	0	0	0	0	1	0	1	1	1	3	1	0	1	0	1
16	1	1	0	0	1	1	0	0	0	0	1	0	1	1	1	3	0	0	1	0
17	0	1	0	0	0	0	1	1	0	0	1	1	0	1	0	0	3	1	1	1
18	1	0	1	0	0	0	0	1	0	0	0	1	1	0	1	0	1	3	1	1
19	0	1	0	1	1	0	0	0	1	0	0	0	0	1	0	1	1	1	3	1
20	0	0	1	0	1	1	0	0	1	1	0	0	0	0	1	0	1	1	1	3

Continuing with our demos, Alpha1B.sas generates some data from the alpha lattice to illustrate the analysis. Two datasets are generated. The first has one observation in each (trt,block) combination.

```
*Demo Alpha1B.txt - run Alpha1.txt and Alpha1A.txt first;
%let nblks=15;
%let reps=3;

Data effects; drop i;
array BL(&nblks); array S(&reps);
do i=1 to &reps; S(i) = 8*normal(123); end;
 *random set effects;
do i=1 to &nblks; BL(i) = 10*normal(123); end;
* random block effects;
proc print; title "&reps sets of blocks, &nblks total";
run;
```

```
data new;
array BL(&nblks) BL1-BL&nblks; array s(&reps) S1-S&reps;
set Design;
if _n_=1 then set effects; retain;
Yield = 50 + 2*(trt-10) + BL(block) + S(set) +
5*normal(123);
keep yield trt block set;
proc print data=New; run;

proc mixed data=new; class trt block set;
model yield = trt;
random set block(set);
estimate "1 vs 2 (1st assoc.)" trt -1 1 ;
estimate "1 vs 3 (1st assoc.)" trt -1 0 1 ;
estimate "1 vs 5 (Oth assoc.)"  trt -1 0 0 0 1 ;
estimate "1 vs. 7 (Oth assoc.)" trt -1 0 0 0 0 0 1;
run;

Data clones; set new;
do clone = 1 to 3;
  Y=yield+.5*normal(123);
output; end;
proc mixed data=clones; class trt block set;
model y = trt;
random set trt*set block(set);
estimate "1 vs 2 (1st assoc.)" trt -1 1 ;
estimate "1 vs 3 (1st assoc.)" trt -1 0 1 ;
estimate "1 vs 5 (Oth assoc.)"  trt -1 0 0 0 1 ;
estimate "1 vs. 7 (Oth assoc.)" trt -1 0 0 0 0 0 1;
run;
```

Partial output from first MIXED run (no clones)

Covariance Parameter Estimates

Cov Parm	Estimate
set	182.10
block(set)	29.9684
Residual	24.9976

Type 3 Tests of Fixed Effects

Effect	Num DF	Den DF	F Value	Pr > F
trt	19	26	13.22	<.0001

Estimates

Label	Estimate	Standard Error	DF	t Value	Pr > |t|
1 vs 2 (1st assoc.)	2.7133	4.4601	26	0.61	0.5482
1 vs 3 (1st assoc.)	6.8034	4.4601	26	1.53	0.1392
1 vs 5 (0th assoc.)	7.2214	4.6676	26	1.55	0.1339
1 vs. 7 (0th assoc.)	12.1324	4.6676	26	2.60	0.0152

With the clones we can get a clean estimate of the interaction between the treatments (perhaps varieties) and the sets (perhaps locations).

Covariance Parameter Estimates

Cov Parm	Estimate
set	181.18
trt*set	24.3076
block(set)	30.3991
Residual	0.2266

Type 3 Tests of Fixed Effects

Effect	Num DF	Den DF	F Value	Pr > F
trt	19	38	13.47	<.0001

Estimates

Label	Estimate	Standard Error	DF	t Value	Pr > |t|
1 vs 2 (1st assoc.)	2.9771	4.4083	38	0.68	0.5035
1 vs 3 (1st assoc.)	6.6365	4.4083	38	1.51	0.1405
1 vs 5 (0th assoc.)	7.7117	4.6152	38	1.67	0.1030
1 vs. 7 (0th assoc.)	12.4813	4.6152	38	2.70	0.0102

To see how PROC PLAN can generate the three sets in the alpha array, run the demo Alpha_PLAN.sas.

```
** Demo Alpha_Plan_notes.sas **;
   ******************************************************
  * Example:  Test 20 families (varieties) in 3 sets of   *
  * blocks.  In each set (set=superblock), 5 blocks of 4  *
  * families each. Each set contains all 20 families.     *
   ******************************************************;
 ** block -> number of blocks to print. Try changing to 10;
 ** family -> how many treatments (say families here)
     you want per block;
 ** of ___ how many treatments (families here) there
     are altogether;

proc plan;
  factors block=5 family=4 of 20
  cyclic ( 1 18 15 12) 4;
 run;

 /*
cyclic ( 1 2 3 4 ) 4 ;
cyclic ( 1 6 11 16) 4;
   */
```

(output)

Factor	Select	Levels	Order	Initial Block / Increment
block	5	5	Random	
family	4	20	Cyclic	(1 18 15 12) / 4

block	---family--			
4	1	18	15	12
2	5	2	19	16
3	9	6	3	20
5	13	10	7	4
1	17	14	11	8