The Theory of Designed Experiments

9. Complex Block Structures

UNESP-Botucatu, August 2010

Nested Block Structures

In a simple block design, most of the information comes from within block comparisons, so we should design the experiment to ensure maximum information at this level.

This automatically determines what information is available at the inter-block level and so no modifications are possible to the design principles we have already established.

Nested Block Structures

However if the blocks themselves fall into groups or are discretizations of a continuous variable such as space or time, it might be sensible to restrict the randomization at this level, i.e. when randomizing blocks to block labels. The blocks might be grouped together in superblocks.

If the blocks are arranged so that each superblock contains each treatment exactly once, the design is said to be *resolved*. (Actually, usually a block design is said to be *resolvable* if it *can* be arranged in this way, but this is a purely mathematical concept.)

Nested Block Structures

It is not necessarily the case that we should use a resolved design even if one exists. Two views:

- Choose the best block design, then arrange blocks in superblocks to get the most informative analysis ignoring blocks;
- Keep replicates in separate superblocks and, conditional on this, arrange treatments to blocks to get the most informative intra-block analysis.

The latter is more commonly done in practice, but the former seems more logical in most experiments, since it maximizes the amount of information in the most informative stratum.

Example

A variety screening experiment is to be conducted to compare 20 varieties in 8 blocks, each containing 5 plots.

A good incomplete block design, obtained by computer search, is

			Block				
I	II	III	IV	V	VI	VII	VIII
1	3	1	4	6	7	2	3
2	5	4	7	12	8	5	11
9	9	6	10	13	12	10	15
11	14	8	15	14	16	13	16
18	19	19	18	20	17	17	20

Nested Block Structures

In field experiments, and in many other experiments, blocks will be used for management purposes, e.g. a block or blocks will be harvested together, as well as for reducing variation.

This may make it convenient to use superblocks in the above experiment which, however, is not resolvable. We *might* prefer to force the design to be resolved and select a design from the restricted class of resolved incomplete block designs.

Searches for resolved designs are even simpler, since the interchanges are restricted to be between blocks in the same superblock. We did this to obtain the following design.

Nested Block Structures

	Superblock							
	I				II			
Block	I	II	III	IV	V	VI	VII	VIII
	1	2	3	4	1	2	3	4
	5	6	7	8	6	7	8	5
	9	10	11	12	11	12	9	10
	13	14	15	16	16	13	14	15
	17	18	19	20	20	17	18	19

Note the (generalized) cyclic structure of this design. Until recently, cyclic methods were used to construct such designs. They usually allow reasonably good designs to be found, but are not guaranteed to produce optimal designs.

Multi-way blocking

Example An experiment is to be conducted to compare 8 different cattle feeds for their effect on the milk yield of cows. 8 animals are available and they can be used 4 times each. Washout periods are allowed between measuring the feeds to ensure that there is no carry over effect between periods. We have two obvious blocking factors - animals and time periods.

Multi-way blocking

It may be important to allow for two (or more) sources of variation among experimental units in many types of experiment, e.g.

- In two- (or more-)colour microarray experiments, there might be differences between frames and between colours
- In field or controlled environment experiments there might be spatial variation in both directions.
- In human experiments, age and sex might both be important.
- In laboratory experiments, time and piece of equipment might both be blocking factors.

Multi-way blocking

If the number of rows and the number of columns are both the same as the number of treatments, then a Latin square design is appropriate.

More often multiple Latin squares are appropriate. There are two different situations:

- Separate squares have separate rows and separate columns. The unit structure is `Squares/(Rows*Columns)`.
- Separate squares have separate rows but the same columns - Latin rectangles. The unit structure is `Rows*Columns`.

Note that the same mathematical object, e.g. a set of 4 Latin squares, can give a suitable design for different experimental structures.

Multi-way blocking

If the unit structure does not allow us to use a Latin square, we can consider using a “nearly-Latin” square, e.g. by deleting a single row or column from a Latin square. Such designs preserve the optimality properties of Latin squares.

More generally, if we can have complete blocks in one direction, we should simply match these with an optimal block design in the other direction.

Multi-way blocking

We will find a design for our initial example.

First get a good block design for animals as blocks.

Animal							
I	II	III	IV	V	VI	VII	VIII
1	1	1	1	2	2	2	3
2	3	3	4	3	4	5	4
4	5	6	5	5	7	6	6
6	8	7	7	7	8	8	8

Multi-way blocking

Rearrange to get each treatment in each period.

		Animal							
		I	II	III	IV	V	VI	VII	VIII
Period	I	1	8	6	4	2	7	5	3
	II	2	1	7	5	3	8	6	4
	III	4	3	1	7	5	2	8	6
	IV	6	5	3	1	7	4	2	8

Multi-way blocking

Say we had only 6 animals.

A good design for columns

Animal					
I	II	III	IV	V	VI
1	1	1	2	3	4
2	2	3	3	4	5
4	5	6	6	5	6
8	7	8	7	7	8

and a good design for rows

Period	I	1	2	3	4	5	6
	II	1	2	3	4	7	8
	III	1	2	5	6	7	8
	IV	3	4	5	6	7	8

can be combined to produce a good row-column design

		Animal					
		I	II	III	IV	V	VI
Period	I	1	2	3	6	4	5
	II	8	7	1	2	3	4
	III	2	1	8	7	5	6
	IV	4	5	6	3	7	8

Multi-way blocking

The separate construction of an optimal row design and an optimal column design is not guaranteed to find the optimal row-column design, except when one of the blocking systems can be orthogonal.

The most general methods of construction are again algorithmic. Interchange algorithms can be run which are the same as those for single blocking, except that now all interchanges must be allowed, not just those between units in different blocks.

The criteria can be the same as those for blocking.

General row-column designs tend to be less “nice” than general block designs, e.g. it is harder to achieve balance or near balance.

Multi-way blocking

It is possible, but not common in practice, to have more than two blocking classifications.

Example: 4 insecticides for fruit trees are to be compared on 16 trees for their effects. It is thought possible that trees which had the same treatment last year will give similar responses, as well as those in rows and columns. A Graeco-Latin square is appropriate.

		Column							
		I		II		III		IV	
Row	I	I	1	II	2	III	3	IV	4
	II	III	2	IV	1	I	4	II	3
	III	IV	3	III	4	II	1	I	2
	IV	II	4	I	3	IV	2	III	1

More general structures

Example (from Ch. 3): In a hen house, 32 chicken cages are arranged in 4 stacks of 4$\times$2 cages. 8 treatments are to be compared. The unit structure is `(Stack*Tier)/Cage`.

One possibility, known as a "*Trojan square*" uses the same mathematical structure as the Graeco-Latin square.

	Stack							
Tier	I		II		III		IV	
I	1	5	2	6	3	7	4	8
II	3	6	4	5	1	8	2	7
III	4	7	3	8	2	5	1	6
IV	2	8	1	7	4	6	3	5

More general structures

This design is doubly resolved, i.e. there is a complete set of treatments in each row and a complete set in each column.

This might not be the best way to construct the design. Alternatively, choose an optimal design for blocks of size 2, then rearrange blocks in rows and columns to optimize with respect to rows and columns.

In this case, the Trojan square is optimal for unstructured treatments, but not for factorial treatments (with several sensible optimality criteria).

However, with unstructured treatments, there are other optimal block designs which cannot be made doubly resolvable.

Hence, we should try to find *all*, or at least many, optimal block designs and then arrange each of them in rows and columns.