

Vignette for ‘blocksdesign’ package

Introduction

Comparative experiments in agriculture and biology involve comparisons between unstructured treatments such as varieties or comparisons between factorial treatments such as combinations of different fertilizer types or comparisons between quantitative level treatments such as rates of fertilizer application. Treatment design for unstructured treatments is usually pre-determined by the requirements of the experiment but the best choice of design for a set of factorial or functional treatments can be complex and may require computer methods. A common feature of agricultural and biological experiments is the high background variability of individual plots or units and the effective control of background variability is usually essential for efficient estimation of treatment effects. The most common method for the control of non-treatment variability is to arrange treatments into homogeneous blocks of plots so that the precision of comparison of treatments within individual blocks is improved. Good block design may also require computer methods and the ‘blocksdesign’ package is intended to provide an integrated general purpose design package for both treatment and block design, especially for field and crop experiments.

Treatment designs

Unstructured treatments

Unstructured treatments have no underlying treatment model and the only meaningful comparisons are pairwise differences between individual pairs of treatments. Treatment design for unstructured treatments is chiefly concerned with the choice of individual treatments and individual treatment replication and these choices usually depend on the purposes and economics of a trial. Replication need not be equal for all treatments and often it is desirable to increase the replication on certain individual treatments, for example when certain treatments are controls or standards against which the remaining treatments are to be compared. Sometimes, due say to lack of resources or lack of experimental material, some treatments may be un-replicated (conventionally they have a single ‘replication’). Usually, the choice of replication will be decided by the experimenter on pragmatic grounds and it is important that any good block design algorithm should be able to provide an efficient block design for any arbitrary pattern of treatment replication.

Structured treatments

Structured treatments have an underlying model such as a response surface model for quantitative level factors or a factorial model for qualitative level factors. Response surface designs and factorial designs assume an empirical linear model for treatment effects and efficient design is usually based on the optimization of a design criterion derived from the information matrix of the assumed empirical linear model. The most general design criterion is D-optimality (see Atkinson et. al. 2007), which maximizes the determinant of the design information matrix. D-optimality is the criterion used by ‘blocksdesign’ for the numerical optimization of all general, non-orthogonal, treatment designs.

Treatment design algorithm

The treatment design algorithm used by 'blocksdesign' finds a D-optimal treatment design by selecting an optimal set of treatment combinations from a candidate set of treatments. The candidate set contains all feasible treatment combinations that might occur in the final optimized design and the design algorithm selects those treatment combinations that optimize the treatment information matrix. Treatments are selected from the candidate set with replacement except when the size of the candidate set exactly equals the size of the required design and the treatments model is null in which case the entire candidate set is selected as the treatment design. Allowing the entire candidate set to be selected allows any arbitrary pre-selected design to be used as a treatment design. For example, if a design with arbitrary replication of individual treatments is required, then using that design as the candidate set and using a null treatment model will force the full candidate set to be used as the required design and will preserve the required replication. If selection with replacement is required, it is merely necessary to define a non-null treatments model. If the candidate set is different in size from the required design, selection with replacement is automatic.

Treatment models

The treatments model can be either a single formula for a single design matrix or a compound formula for two or more design matrices. A compound formula contains one or more occurrences of a splitting operator | which splits-off a partial design formula on the left-hand side of each |. Assuming the left-hand part of each split is a well-defined model design formula and replacing all remaining | by + in each partial design formula gives a hierarchical set of design matrices for sequential model fitting. The advantage of sequential model fitting is that it provides improved flexibility for fitting factors or variables of different status or importance and allows a wider range of choices of optimized design for different situations (see examples below). Assuming each formula in a hierarchical set contains at least one variable or factor not contained in any preceding formula, the resulting set of design matrices can be conditionally optimized from the smallest to the largest.

The hierarchical fitting process provides flexibility for treatment design optimization and allows model terms to be fitted in order of importance. For example, if a treatment design has a set of one or more qualitative level factors such as varieties in a field trial together with a number of interacting polynomial fertilizer factors it may be desirable to ensure that all the varieties are as near equally replicated as possible. However, global optimization of a single design matrix is not guaranteed to give an equal or balanced representation of all individual factor levels in a design. To ensure equal or near-equal representation for each variety, it may be desirable to fit the variety design first to ensure a balanced allocation of varieties. Then the full design can be fitted by the full model formula but with the variety factor levels held constant to maintain the required variety design.

Examples

The following examples are constructed by the *design{blocksdesign}* function which is a general-purpose design function for general qualitative or quantitative level factorial models. The design inputs are a candidate set of treatments, a treatments model and a set of block factors. The design size is determined by the length of the block factors, if defined, otherwise by the length of the candidate set. If the number of candidate treatments exactly equals the design size and the *treatments_model* is null, the full candidate set is selected, otherwise selection with replacement is used.

Example 1a: Treatment design for 2 varieties x 3 levels of N x 3 levels of K in 12 plots assuming a model with first-order interaction effects. This example optimizes the whole treatment design simultaneously using a single model formula

```
## single treatment model formula;
treatments = expand.grid(Variety = factor(rep(1:2)), N = 1:3, K = 1:3);
variety = "~ Variety";
model = "~ (Variety + N + K)^2 + I(N^2) + I(K^2)";
blocks=data.frame(main=gl(1,12));
design(treatments,blocks,treatments_model=model,searches=10);
```

```
$treatments_model
```

	Treatment.model	Model.DF	D.Efficiency
1	~ (Variety + N + K)^2 + I(N^2) + I(K^2)	8	1.013173

```
$treatments
```

	Variety	N	K	freq
1	1	1	1	1
2	1	1	2	1
3	1	1	3	1
4	1	2	1	1
5	1	3	1	1
6	1	3	2	1
7	1	3	3	1
8	2	1	1	1
9	2	1	3	1
10	2	2	2	1
11	2	3	1	1
12	2	3	3	1

Example 1b: Treatment design for 2 varieties x 3 levels of N x 3 levels of K in 12 plots assuming a model with first-order interaction effects. This example optimizes the variety design first and then optimizes the full treatment design with the variety factor effects held fixed

```
## sequentially fitted treatment model formulae;
model = "~ Variety | (Variety + N + K)^2 + I(N^2) + I(K^2)";
design(treatments,blocks,treatments_model=model,searches=10);
```

```
$treatments_model
```

	Treatment.model	Model.DF	D.Efficiency
1	~ Variety	1	1
2	~ Variety + (Variety + N + K)^2 + I(N^2) + I(K^2)	8	0.9955825

```
$treatments
```

	Variety	N	K	freq
1	1	1	1	1
2	1	1	2	1
3	1	1	3	1
4	1	2	3	1
5	1	3	1	1
6	1	3	3	1
7	2	1	1	1
8	2	1	3	1
9	2	2	2	1
10	2	2	3	1
11	2	3	1	1
12	2	3	3	1

Example 1a gives an unequal 7 + 5 split of variety plots which is likely to be undesirable whereas Example 1b forces a 6 + 6 split. The D-efficiency of the full model in Example 1a is slightly higher than the D-efficiency of the full model in Example 1b but for most practical purposes Example 1b will be the preferred design

Example 2a Second-order response surface for three 3-level factors assuming a 10-point design

```
treats = expand.grid(A = 1:3, B = 1:3, C = 1:3);
block = data.frame(main=gl(1,10));
model = " ~ ( A + B + C)^2 + I(A^2) + I(B^2) + I(C^2)";
design(treats,block,treatments_model=model);
```

	Treatment.model	Model.DF	D.Efficiency
1	~ (A + B + C)^2 + I(A^2) + I(B^2) + I(C^2)	9	0.9262674

```
$treatments
  A B C freq
1 1 1 1 1
2 1 1 3 1
3 1 2 2 1
4 1 3 1 1
5 2 1 2 1
6 2 2 1 1
7 2 3 3 1
8 3 1 1 1
9 3 2 3 1
10 3 3 1 1
```

Example 2a shows the D-optimum choice of 10 treatments from a complete factorial design for three 3-level factors assuming a second-order response surface model. The second-order model has nine parameters (and a mean) therefore a design based on only 10 point is saturated and is not useful for model checking. The D-efficiency has no special meaning as the full candidate set is not a natural choice of design for this treatment model.

Example 2b Same design as 2a but with sequential model fitting.

Input

```
model = " ~ A + I(A^2) | (A + B)^2 + I(B^2) | (A + B + C)^2 + I(C^2)";
repeat {z=design(treats,block,treatments_model=model);
if (z$treatments_model[3,3]>0.92) break}; print(z);
```

	Treatment.model	Model.DF	D.Efficiency
1	~ A + I(A^2)	2	0.9905782
2	~ A + I(A^2) + (A + B)^2 + I(B^2)	5	0.9931412
3	~ A + I(A^2) + (A + B)^2 + I(B^2) + (A + B + C)^2 + I(C^2)	9	0.9262674

```
$treatments
  A B C freq
1 1 1 1 1
2 1 1 3 1
3 1 2 2 1
4 1 3 1 1
5 2 1 2 1
6 2 2 1 1
7 2 3 3 1
8 3 1 1 1
9 3 2 3 1
10 3 3 1 1
```

Example 2b shows sequential fitting of treatment models for the three treatment factors A, B and C in that order. In this example, the D-efficiency of the best full sequential model is equal to the D-efficiency of the full simultaneous model but only after a large number of complete replicate searches.

Nested block designs

In many situations, comparability between treatments can be improved by grouping the experimental units into blocks. Blocks should be as homogeneous as possible and the choice of blocks design can be critical for the success of an experiment. The most basic type of block design is the complete randomized blocks design where each block contains one or more complete replicate sets of treatments. Complete randomized blocks estimate all treatment effects fully within blocks and are usually the best choice for small experiments. However, for large experiments, variability within complete blocks can be large and then it may be beneficial to further sub-divide complete replicate block into smaller nested sub-blocks to improve the precision of the within-sub-block comparisons.

Nested blocks

Complete replicate blocks with a single level of nesting are called resolvable incomplete blocks and are widely used in practical research. Treatment information is estimated both within and between the incomplete blocks and a fully informative analysis requires the combination of within and between-block treatment information using some form of mixed-model analysis, see, for example, Piepho and Edmondson (2018). The aim of good block design is to maximize the precision of estimation of treatment effects and for a single level of nesting block designs can be optimized by maximizing the information content of the incomplete blocks design. Various design criteria have been considered for block designs (see, for example, John & Williams 1998) but the most general design criterion is D-optimality. The D-optimality criterion maximizes the determinant of the design information matrix and is the criterion of choice used by the ‘blocksdesign’ algorithm.

Large designs

Although resolvable block designs with a single level of nesting work well for small or moderate numbers of experimental units, a single level of nesting may be inadequate for large experiments such as field variety trials which may involve scores or hundreds of treatments. Small or moderate sized experiments with nested blocks of reasonable size will confound only a small amount of treatment information between blocks and should have good efficiency even when the inter- to intra-block variance ratio is high. For large sized experiments with heterogeneous variability, however, if the nested blocks are small enough to give good within-block homogeneity of variance, the inter-block space will be large and heterogeneous and will confound a substantial amount of treatment information between blocks. In that situation, the efficiency of recovery of inter-block treatment information will be low and the overall efficiency of the block design will be sub-optimal.

Multi-level nesting

Multi-level nesting gives a series of nested blocks where the nested inter-block space at each level of nesting can be assumed to have good homogeneity of variance and where only a small amount of useful treatment information is confounded within each nested inter-block space. In many situations, the variability will decrease with depth of nesting and in those situations top-down optimization should ensure that the minimum possible amount of treatment information is confounded within each inter-block space taken in order from the top down. A mixed model analysis of a multi-level nested block design using modern mixed model design is straightforward and allows the proper weighted combination of treatment information from each inter-block space.

Examples

The following examples use the `blocks{blocksdesign}` function which is a special recursive function for simple multi-level nested block designs for unstructured treatment sets. The function generates designs for treatments with arbitrary levels of replication and arbitrary depth of nesting and each successive set of blocks is optimized within the levels of each preceding set of blocks using conditional D-optimality. Special block designs such as lattice designs or latin or Trojan square designs are constructed algebraically using mutually orthogonal Latin squares (MOLS). Designs based on prime-power MOLS require the ‘crossdes’ package, see Sailer (2013). The block sizes are chosen automatically by the algorithm dependent on the block and treatment design and the block sizes for any particular set of blocks will always be as nearly equal as possible and will never differ by more than one unit. The outputs from the `blocks` function include a data frame showing the allocation of treatments to blocks for each plot of the design and a table showing the achieved D- and A-efficiency factors for each set of nested blocks together with A-efficiency upper bounds, where available. A plan showing the allocation of treatments to blocks in the bottom level of the design is also included in the output. See John and Williams (1998) for a definition of A-efficiency.

Example 3. Two replicates of 128 treatments with two main blocks and four levels of nesting with two nested sub-blocks in each level of nesting.

In this example, the attained efficiencies of the first five levels of nesting are close to or equal to their upper bounds which shows that these nested designs are close to optimal. The agreement for the bottom level of nesting is less good but it is known that the reliability of efficiency bounds decreases as the efficiency decreases so the reliability of the bound for the bottom level of nesting is probably poor. In summary, this example shows that for equi-replicate design with regular block sizes the efficiency of multi-level nesting is close to the expected upper-bound suggesting that there is little loss of efficiency due to the constraints imposed by multi-level blocking.

```
blocks(treatments = 128, replicates = 2, blocks = c(2, 2, 2, 2, 2, 2));
```

```
$blocks_model
  Level Blocks D-Efficiency A-Efficiency A-Bound
1 Level_1     2           1           1           1
2 Level_2     4   0.9891437   0.9844961 0.9883226
3 Level_3     8   0.9677833   0.9548872 0.9601815
4 Level_4    16   0.9264364   0.9007092 0.9057052
5 Level_5    32   0.8419961   0.786915 0.7898712
6 Level_6    64   0.6654802   0.5427813 0.566227
```

\$Plan

	Level_1	Level_2	Level_3	Level_4	Level_5	Level_6	Blocks.Plots:	1	2	3	4
1	Blocks_1	Blocks_1	Blocks_1	Blocks_1	Blocks_1	Blocks_1		79	24	33	14
2	Blocks_1	Blocks_1	Blocks_1	Blocks_1	Blocks_1	Blocks_2		29	42	34	115
3	Blocks_1	Blocks_1	Blocks_1	Blocks_1	Blocks_2	Blocks_1		18	41	94	5
4	Blocks_1	Blocks_1	Blocks_1	Blocks_1	Blocks_2	Blocks_2		26	21	30	39
5	Blocks_1	Blocks_1	Blocks_1	Blocks_1	Blocks_2	Blocks_1		36	112	82	72
6	Blocks_1	Blocks_1	Blocks_1	Blocks_2	Blocks_1	Blocks_2		81	110	106	78
7	Blocks_1	Blocks_1	Blocks_1	Blocks_2	Blocks_2	Blocks_1		65	113	23	63
8	Blocks_1	Blocks_1	Blocks_1	Blocks_2	Blocks_2	Blocks_2		99	8	125	84
9	Blocks_1	Blocks_1	Blocks_2	Blocks_1	Blocks_1	Blocks_1		52	58	117	127
10	Blocks_1	Blocks_1	Blocks_2	Blocks_1	Blocks_1	Blocks_2		11	116	19	10
11	Blocks_1	Blocks_1	Blocks_2	Blocks_1	Blocks_2	Blocks_1		91	126	59	75
12	Blocks_1	Blocks_1	Blocks_2	Blocks_1	Blocks_2	Blocks_2		104	107	4	64
13	Blocks_1	Blocks_1	Blocks_2	Blocks_2	Blocks_1	Blocks_1		85	98	28	123
14	Blocks_1	Blocks_1	Blocks_2	Blocks_2	Blocks_1	Blocks_2		49	120	16	118
15	Blocks_1	Blocks_1	Blocks_2	Blocks_2	Blocks_2	Blocks_1		71	100	67	102
16	Blocks_1	Blocks_1	Blocks_2	Blocks_2	Blocks_2	Blocks_2		68	17	95	77
17	Blocks_1	Blocks_2	Blocks_1	Blocks_1	Blocks_1	Blocks_1		87	53	2	69
18	Blocks_1	Blocks_2	Blocks_1	Blocks_1	Blocks_1	Blocks_2		61	108	76	66
19	Blocks_1	Blocks_2	Blocks_1	Blocks_1	Blocks_2	Blocks_1		96	88	15	128
20	Blocks_1	Blocks_2	Blocks_1	Blocks_1	Blocks_2	Blocks_2		31	54	92	62
21	Blocks_1	Blocks_2	Blocks_1	Blocks_2	Blocks_1	Blocks_1		101	25	37	109
22	Blocks_1	Blocks_2	Blocks_1	Blocks_2	Blocks_1	Blocks_2		89	73	111	43
23	Blocks_1	Blocks_2	Blocks_1	Blocks_2	Blocks_2	Blocks_1		22	86	97	13
24	Blocks_1	Blocks_2	Blocks_1	Blocks_2	Blocks_2	Blocks_2		74	3	83	90
25	Blocks_1	Blocks_2	Blocks_2	Blocks_1	Blocks_1	Blocks_1		56	1	38	105
26	Blocks_1	Blocks_2	Blocks_2	Blocks_1	Blocks_1	Blocks_2		45	60	103	47
27	Blocks_1	Blocks_2	Blocks_2	Blocks_1	Blocks_2	Blocks_1		57	121	44	35
28	Blocks_1	Blocks_2	Blocks_2	Blocks_1	Blocks_2	Blocks_2		93	6	27	7
29	Blocks_1	Blocks_2	Blocks_2	Blocks_2	Blocks_1	Blocks_1		46	9	55	20
30	Blocks_1	Blocks_2	Blocks_2	Blocks_2	Blocks_1	Blocks_2		114	32	122	40
31	Blocks_1	Blocks_2	Blocks_2	Blocks_2	Blocks_2	Blocks_1		119	80	124	50
32	Blocks_1	Blocks_2	Blocks_2	Blocks_2	Blocks_2	Blocks_2		51	70	12	48
33	Blocks_2	Blocks_1	Blocks_1	Blocks_1	Blocks_1	Blocks_1		64	16	101	94
34	Blocks_2	Blocks_1	Blocks_1	Blocks_1	Blocks_1	Blocks_2		54	9	61	22
35	Blocks_2	Blocks_1	Blocks_1	Blocks_1	Blocks_2	Blocks_1		81	85	20	41
36	Blocks_2	Blocks_1	Blocks_1	Blocks_1	Blocks_2	Blocks_2		117	8	47	7
37	Blocks_2	Blocks_1	Blocks_1	Blocks_2	Blocks_1	Blocks_1		36	56	69	31
38	Blocks_2	Blocks_1	Blocks_1	Blocks_2	Blocks_1	Blocks_2		19	32	98	124
39	Blocks_2	Blocks_1	Blocks_1	Blocks_2	Blocks_2	Blocks_1		105	25	123	42
40	Blocks_2	Blocks_1	Blocks_1	Blocks_2	Blocks_2	Blocks_2		84	91	90	18
41	Blocks_2	Blocks_1	Blocks_2	Blocks_1	Blocks_1	Blocks_1		108	30	33	109
42	Blocks_2	Blocks_1	Blocks_2	Blocks_1	Blocks_1	Blocks_2		77	38	58	128
43	Blocks_2	Blocks_1	Blocks_2	Blocks_1	Blocks_2	Blocks_1		63	55	70	1
44	Blocks_2	Blocks_1	Blocks_2	Blocks_1	Blocks_2	Blocks_2		126	89	112	67
45	Blocks_2	Blocks_1	Blocks_2	Blocks_2	Blocks_1	Blocks_1		100	97	28	60
46	Blocks_2	Blocks_1	Blocks_2	Blocks_2	Blocks_1	Blocks_2		96	119	2	99
47	Blocks_2	Blocks_1	Blocks_2	Blocks_2	Blocks_2	Blocks_1		72	13	11	39
48	Blocks_2	Blocks_1	Blocks_2	Blocks_2	Blocks_2	Blocks_2		48	45	14	107
49	Blocks_2	Blocks_2	Blocks_1	Blocks_1	Blocks_1	Blocks_1		86	118	29	15
50	Blocks_2	Blocks_2	Blocks_1	Blocks_1	Blocks_1	Blocks_2		127	35	37	82
51	Blocks_2	Blocks_2	Blocks_1	Blocks_1	Blocks_2	Blocks_1		102	110	27	50
52	Blocks_2	Blocks_2	Blocks_1	Blocks_1	Blocks_2	Blocks_2		24	46	88	75
53	Blocks_2	Blocks_2	Blocks_1	Blocks_2	Blocks_1	Blocks_1		121	66	103	95
54	Blocks_2	Blocks_2	Blocks_1	Blocks_2	Blocks_1	Blocks_2		122	52	4	3
55	Blocks_2	Blocks_2	Blocks_2	Blocks_2	Blocks_2	Blocks_1		78	68	21	74
56	Blocks_2	Blocks_2	Blocks_1	Blocks_2	Blocks_2	Blocks_2		40	79	23	62
57	Blocks_2	Blocks_2	Blocks_2	Blocks_1	Blocks_1	Blocks_1		120	43	44	106
58	Blocks_2	Blocks_2	Blocks_2	Blocks_1	Blocks_1	Blocks_2		26	71	12	53
59	Blocks_2	Blocks_2	Blocks_2	Blocks_1	Blocks_2	Blocks_1		73	104	80	92
60	Blocks_2	Blocks_2	Blocks_2	Blocks_1	Blocks_2	Blocks_2		5	6	116	113
61	Blocks_2	Blocks_2	Blocks_2	Blocks_2	Blocks_1	Blocks_1		111	114	51	34
62	Blocks_2	Blocks_2	Blocks_2	Blocks_2	Blocks_1	Blocks_2		57	87	65	83
63	Blocks_2	Blocks_2	Blocks_2	Blocks_2	Blocks_2	Blocks_1		49	10	125	76
64	Blocks_2	Blocks_2	Blocks_2	Blocks_2	Blocks_2	Blocks_2		59	93	17	115

Example 4. Two replicates of 50 treatments (1 to 50) and 50 replicates of one treatment (51) with two main replicate blocks of size 75 and 25 blocks of size 3 nested within each main block. Treatment 51 is a control treatment and should occur once in every nested block.

Inputs

```
blocks(treatments=c(50,1),replicates=c(2,50),blocks=c(2,25));
```

Outputs

```
$blocks_model
  Level Blocks D-Efficiency A-Efficiency A-Bound
1 Level_1     2             1             1      <NA>
2 Level_2    50    0.6358266    0.5909988      <NA>
```

\$Plan

```
  Level_1 Level_2 Blocks.Plots:  1  2  3
1 blocks_1 blocks_1          37 12 51
2 blocks_1 blocks_2          20  1 51
3 blocks_1 blocks_3          38 51 17
4 blocks_1 blocks_4           9 51 32
5 blocks_1 blocks_5          19 26 51
6 blocks_1 blocks_6          51 50 48
7 blocks_1 blocks_7          31 43 51
8 blocks_1 blocks_8          22 51  8
9 blocks_1 blocks_9          51 18 10
10 blocks_1 blocks_10         51 36 11
11 blocks_1 blocks_11         29 51 13
12 blocks_1 blocks_12         51  2 42
13 blocks_1 blocks_13         47 16 51
14 blocks_1 blocks_14          6 51  4
15 blocks_1 blocks_15         39 28 51
16 blocks_1 blocks_16         33 51 27
17 blocks_1 blocks_17          7 40 51
18 blocks_1 blocks_18         51 25 45
19 blocks_1 blocks_19         24 46 51
20 blocks_1 blocks_20         44 51 35
21 blocks_1 blocks_21         51 21 30
22 blocks_1 blocks_22         51  3 23
23 blocks_1 blocks_23         34 51 15
24 blocks_1 blocks_24         51 49 41
25 blocks_1 blocks_25         51 14  5
26 blocks_2 blocks_1          41  4 51
27 blocks_2 blocks_2          10 29 51
28 blocks_2 blocks_3          50 16 51
29 blocks_2 blocks_4           6 51 21
30 blocks_2 blocks_5          24 31 51
31 blocks_2 blocks_6          20 51 22
32 blocks_2 blocks_7          32 51 47
33 blocks_2 blocks_8          45 51  3
34 blocks_2 blocks_9          46 19 51
35 blocks_2 blocks_10         38 28 51
36 blocks_2 blocks_11         51 26 40
37 blocks_2 blocks_12         23 51 11
38 blocks_2 blocks_13         51 35 33
39 blocks_2 blocks_14         48 18 51
40 blocks_2 blocks_15          1 51 15
41 blocks_2 blocks_16          5 27 51
42 blocks_2 blocks_17         17 42 51
43 blocks_2 blocks_18         13 51 25
44 blocks_2 blocks_19         36 44 51
45 blocks_2 blocks_20          2 51 43
46 blocks_2 blocks_21         49 51 34
47 blocks_2 blocks_22         12 51 14
48 blocks_2 blocks_23          7 30 51
49 blocks_2 blocks_24         39 51  9
50 blocks_2 blocks_25         37  8 51
```

Factorial block designs

Sometimes it can be advantageous to use a fully crossed factorial block design. In field trials, for example, factorial row-and-column blocks are sometimes used to accommodate physical rows and columns in a field layout. Factorial blocks are often assumed to fit a simple additive main effects model but additivity of main effects is a very strong assumption and may not be fully valid for blocks with many crossed levels.

Polynomial block models

One approach with spatial designs such as row-and-column field trials is to fit a model with low-order polynomial interactions between rows and columns (see Edmondson R. N. 1993). The 'blocksdesign' package assumes block effects are qualitative therefore polynomial block effects cannot be fitted either by the 'blocksdesign::blocks' function or by the 'blocksdesign::design' function. However, an alternative approach is to fit the dual of the design regarding the factorial row and column blocks and the linear rows by linear columns interaction effect as treatments and the original treatments as blocks. As the original treatments are additive, the dual blocks are also additive and can therefore be optimized by the blocks design algorithm. The following example first sets-up a skeleton 6 x 6 Latin square using the data frame LS_grid and then defines a Latin square treatment model with a linear row by linear column interaction effect using the treatments model formula.

The fitted treatments design is an orthogonal design with 11 degrees of freedom which shows that the rows, columns and linear rows by linear columns contrasts are orthogonal and that the dual treatment design is indeed a Latin square. Finally, the dual block design is optimized conditional on the treatments design which means that the allocation of the actual treatments to the Latin square design is optimized after allowing for the presence of a linear row by linear column interaction effect. A substantial number of repeat optimizations was needed to find a stable maximum optimum but after a sufficient number of optimizations a Latin square design with a fixed linear rows by linear columns interaction was found with a D-efficiency factor of about 94% (92% A-efficiency) which means that the best treatment design is about 94% D-efficient (92% A-efficient) for treatment effects after allowing for a linear rows by linear columns interaction effect.

Example 5 Six varieties in a 6 x 6 row-and-column design with polynomial linear row by linear column interaction effect.

Inputs

```
LS_grid = expand.grid(rows=factor(1:6), cols=factor(1:6));
blocks = data.frame(varieties=factor(rep(1:6,6)));
lin_rows = as.numeric(levels(LS_grid$rows)) [LS_grid$rows];
lin_cols = as.numeric(levels(LS_grid$cols)) [LS_grid$cols];
latin_sq = "~ rows | cols + lin_rows:lin_cols ";
d = design(LS_grid,blocks,latin_sq,searches=2000);
```

Outputs

```
$treatments_model
      Treatment model Model DF D-Efficiency
1      ~ rows           5      1
2 ~ rows + cols + lin_rows:lin_cols 11      1

$blocks_model
      Blocks levels D-Efficiency A-Efficiency Interactions levels D-Efficiency A-Efficiency
varieties      5 0.9404422 0.9193536 varieties      5 0.9404422 0.9193536
```

\$design			
	varieties	rows	cols
1	1	4	2
2	2	2	4
3	3	4	1
4	4	4	3
5	5	3	3
6	6	2	5
7	1	3	4
8	2	4	6
9	3	6	6
10	4	3	6
11	5	1	4
12	6	6	3
13	1	6	5
14	2	3	5
15	3	1	5
16	4	2	2
17	5	5	1
18	6	5	2
19	1	2	1
20	2	5	3
21	3	3	2
22	4	5	5
23	5	6	2
24	6	3	1
25	1	5	6
26	2	1	2
27	3	2	3
28	4	1	1
29	5	4	5
30	6	1	6
31	1	1	3
32	2	6	1
33	3	5	4
34	4	6	4
35	5	2	6
36	6	4	4

Factorial block interactions

Assuming that the rows-by-columns intersection blocks of a row-and-column field trial are estimable, another approach is to fit a factorial blocks effects model for the blocks design with block main effects and block interaction effects weighted according to their relative importance. Let T represent the treatment effects matrix of a factorial crossed blocks design and let B represent an orthogonalized matrix of factorial block effects. Assuming that all block effects are estimable, the information matrix of the factorial blocks design model is $T'(I - BB')T$. Maximization of the determinant of this matrix gives a D-optimal design for the factorial blocks design model.

The blocks matrix B can be partitioned into a set of columns, B_1 , representing the block factor main effects and a set of columns, B_2 , representing the block interaction effects and letting w represent a weighting factor between 0 and 1, a weighted crossed blocks interaction effects matrix can be written:

$$\text{Information} = T'(I - B_1B_1' - B_2B_2'w^2)T \quad (1)$$

For $w = 0$ the information matrix in (1) is the usual additive crossed blocks model whereas if $w = 1$, the information matrix is a full factorial blocks model. Intermediate values of w , down-weight the block interaction effects B_2B_2' by the square of the weighting parameter w . The best choice of w will

be unknown at the design stage but the effects of different choices on the attained efficiency factors of the various factorial block effects can be found by trial error at the design stage, as shown in examples 6a, 6b and 6c below.

Example 6 Crossed blocks design for 4 replicates of 12 treatments with 4 rows and 4 columns and blocks of size 3 nested within each row-by-column intersection.

6a) Weighting = 0 giving an additive main effects design for rows and columns

```
treatments = factor(1:12);
blocks = data.frame(Rows = gl(4,12), cols = gl(4,3,48));
design(treatments,blocks,searches=200,weighting=0)$blocks_model;
```

	Blocks	Levels	D_Effic	A_Effic	Interactions	Int_levs	Int_D_Effic	Int_A_Effic
1	Rows	4	1	1	Rows	4	1.000000	1.000000
2	cols	4	1	1	Rows*Cols	16	0.682796	0.6316198

6b) Weighting = 0.5 giving a compromise design for rows, columns and rows:columns blocks

```
design(treatments,blocks,searches=200,weighting=0.5)$blocks_model;
```

	Blocks	Levels	D_Effic	A_Effic	Interactions	Int_levs	Int_D_Effic	Int_A_Effic
1	Rows	4	1	1	Rows	4	1.000000	1.000000
2	cols	4	1	1	Rows*Cols	16	0.7176709	0.7096774

6c) Weighting = 1 giving a fully crossed rows-by-columns design

```
design(treatments,blocks,searches=200,weighting=1)$blocks_model;
```

	Blocks	Levels	D_Effic	A_Effic	Interactions	Int_levs	Int_D_Effic	Int_A_Effic
1	Rows	4	1.000000	1.000000	Rows	4	1.000000	1.000000
2	cols	4	0.9144364	0.9034965	Rows*Cols	16	0.7176709	0.7096774

The row blocks are always added before the column blocks and are always orthogonal irrespective of the choice of w . The column blocks are added after the row blocks and the efficiency of the column blocks and the row-by-column interaction blocks will depend on the value of weighting factor w .

Example 6a has $w = 0$ therefore the main column blocks are optimized with efficiency 1. The rows-by-columns blocks are not optimized in this design.

Example 6c has $w = 1$ therefore the rows-by-columns blocks are optimized with D-efficiency 0.7176709 and A-efficiency 0.7096774. The column blocks are not optimized in this design.

Example 6b has $w = 0.5$ and both the column blocks and the rows-by-columns blocks are optimized simultaneously.

Usually for crossed blocks designs, not all rows, columns and rows-by-columns can be optimized simultaneously but Example 6 is a special design called a Trojan square (Edmondson 1998) which has special optimality properties. In the general case, trial and error methods can be used to find a good choice of weighting that gives a good compromise design with good efficiencies on all the required block structures.

Some additional examples

Example 7 Four replicates of 12 treatments in 4 complete blocks with 4 sub-blocks nested in each main block (this corresponds to a rectangular lattice design see Plan 10.10 Cochran and Cox 1957)

Inputs

```
blocks(treatments = 12, replicates = 4 ,blocks = c(4, 4));
```

Outputs

```
$blocks_model
  Level Blocks D-Efficiency A-Efficiency A-Bound
1 Level_1     4           1           1           1
2 Level_2    16    0.7176709    0.7096774 0.7096774
```

\$Plan

```
  Level_1 Level_2 Blocks.Plots:  1  2  3
1 Blocks_1 Blocks_1           4  6  5
2 Blocks_1 Blocks_2           1  3 12
3 Blocks_1 Blocks_3           8 10 11
4 Blocks_1 Blocks_4           7  9  2
5 Blocks_2 Blocks_1           7  6 12
6 Blocks_2 Blocks_2           3  5 10
7 Blocks_2 Blocks_3          11  4  9
8 Blocks_2 Blocks_4           8  2  1
9 Blocks_3 Blocks_1          11  3  7
10 Blocks_3 Blocks_2           8 12  4
11 Blocks_3 Blocks_3           6  2 10
12 Blocks_3 Blocks_4           9  5  1
13 Blocks_4 Blocks_1           8  5  7
14 Blocks_4 Blocks_2           4  2  3
15 Blocks_4 Blocks_3          10 12  9
16 Blocks_4 Blocks_4           6  1 11
```

This design is constructed algebraically from MOLS and because it is a balanced rectangular lattice will always attain the theoretical A-efficiency upper bound.

Example 8a Four replicates of 12 treatments with 4 main replicate rows and 4 main replicate columns and 3 sub-column blocks nested in each main column.

This design extends Example 6 by nesting 3 sub-columns in each main column to give a physical layout for an experiment with four rows and 12 columns where the 12 columns are arranged in sets of four replicate main columns blocks. The main rows, main columns and main intersection block efficiencies are unchanged from Example 6 because the sub-column blocks have been optimized by swaps made *within* the main row-by-column intersection blocks.

Inputs

```
treatments = factor(rep(1:12,4));
blocks = data.frame(Rows = gl(4,12), Cols = gl(4,3,48), subCols = gl(12,1,48));
design(treatments,blocks,searches=200)$blocks_model;
```

Outputs

	Blocks	Levels	D_Effic	A_Effic	Interactions	Int_levs	Int_D_Effic	Int_A_Effic
1	Rows	4	1.0000000	1.0000000	Rows	4	1.0000000	1.0000000
2	Cols	4	1.0000000	1.0000000	Rows*Cols	16	0.7176709	0.7096774
3	subCols	12	0.8053142	0.7925806	Rows*Cols*subCols	48	0.0000000	0.0000000

Example 8b Four replicates of 12 treatments with 4 main blocks and 3 sub-blocks in each main block

The efficiency factors for the sub-column blocks in Example 8a can be compared with the efficiencies of a simple nested block design for 4 replicates of 12 treatments with 4 main blocks and 3 sub-blocks in each main block. This example shows that the sub-column blocks of Example 8a are fully efficient compared with a simple nested blocks design with blocks of the same size so, for this special case, there is no loss of efficiency due to the additional constraints of the row and column layout.

Inputs

```
blocks(12,4,c(4,3))$blocks_model;
```

Outputs

Level	Blocks	D-Efficiency	A-Efficiency	A-Bound
1	Level_1	4	1	1
2	Level_2	12	0.8053142	0.7925806 0.8133803

Example 9 Two replicates of 272 treatments in a 16 x 34 design with nested rows and columns

Inputs

```
data(durban);
durban=durban[c(3,1,2,4,5)];
durban=durban[do.call(order,durban),];
treatments=data.frame(gen=durban$gen);
Reps = factor(rep(1:2,each=272));
Rows = factor(rep(1:16,each=34));
Col1 = factor(rep(rep(1:4,c(9,8,8,9)),16));
Col2 = factor(rep(rep(1:8,c(5,4,4,4,4,4,5)),16));
Col3 = factor(rep(1:34,16));
blocks = data.frame(Reps,Rows,Col1,Col2,Col3);
design(treatments,blocks,searches=1)$blocks_model;
```

Outputs

Blocks	levels	D-Efficiency	A-Efficiency	Interactions	levels	D-Efficiency	A-Efficiency
Reps	2	1.0000000	1.0000000	Reps	2	1.0000000	1.0000000
Rows	16	0.9647882	0.9507384	Reps*Rows	16	0.9647882	0.9507384
Col1	4	0.9955180	0.9944849	Reps*Rows*Col1	64	0.8437030	0.7812191
Col2	8	0.9853798	0.9800796	Reps*Rows*Col1*Col2	128	0.6768033	0.5419516
Col3	34	0.9207093	0.8915016	Reps*Rows*Col1*Col2*Col3	544	0.0000000	0.0000000

This example shows an alternative blocking system for a real experimental design. The original design (see Durban et al 2003) was a simple additive row-and-column design with rows comprising 34 plots and columns comprising 16 plots. Examination of the data (not shown here) suggests that these assumptions were highly unrealistic and that even after eliminating additive row and column effects the treatment adjusted residuals for each individual row were far from homogeneous. The example design shows the efficiency factors for a nested blocks design with three levels of nesting within columns which would have provided additional control of trends within rows. For comparison, the efficiency factors of the actual treatments design (see data set ‘durban’) assuming the block model shown above can be found from the following analysis:

Inputs

```
blockEfficiencies(treatments,blocks);
```

Outputs

Blocks	levels	D-Efficiency	A-Efficiency	Interactions	levels	D-Efficiency	A-Efficiency
Reps	2	1	1	Reps	2	1	1
Rows	16	0.9640685	0.9479535	Reps*Rows	16	0.9640685	0.9479535
Col1	4	0.9922603	0.9888487	Reps*Rows*Col1	64	0.8407711	0.7710131
Col2	8	0.9826103	0.9752815	Reps*Rows*Col1*Col2	128	0.6748665	0.5368455
Col3	34	0.9161031	0.8809427	Reps*Rows*Col1*Col2*Col3	544	<NA>	<NA>

Every efficiency measure in the first analysis is an improvement on the corresponding efficiency measure in the second analysis. The first design gives more protection against unforeseen or unpredicted trends or patterns in the spatial layout of the design and therefore should provide a more robust design for a practical experiment.

Example 10 Second-order model for a 1/3rd fraction of five qualitative 3-level factors in 3 blocks of size 27

This example is a classical regular fraction of a second-order model for five 3-level factors arranged in 3 blocks each of size 27. An orthogonal design is easily constructed algebraically so it provides a useful test of the algorithmic method. The algorithm quickly constructs a 1/3rd fraction of a full factorial design for the required model but not all such fractions can be divided into three orthogonal blocks and increasing the number of searches will not always help because once the algorithm finds an orthogonal treatment fraction it uses that fraction exclusively when searching for an orthogonal block design. For that reason, the example shows how to repeatedly re-construct the entire design until the required orthogonal block design is found.

Inputs

```
treatments = expand.grid(F1 = factor(1:3), F2 = factor(1:3), F3 = factor(1:3),
F4 = factor(1:3), F5 = factor(1:3));
blocks=data.frame(main=gl(3,27));
model = " ~ (F1 + F2 + F3 + F4 + F5)^2";
repeat {
z=design(treatments,blocks,treatments_model=model,searches=5);
if ( z$blocks_model[1,3] == 1) break };
print(z);
```

Outputs

```
$treatments_model
```

```
Treatment.model1          D.Efficiency
~ (F1 + F2 + F3 + F4 + F5)^2          1
```

```
$blocks_model
```

Blocks	levels	D-Efficiency	A-Efficiency	Interactions	levels	D-Efficiency	A-Efficiency
main	2	1	1	main	2	1	1

References

- Atkinson, A.C, Donev, A.N. & Tobias, R. D. (2007). Optimum Experimental Designs, with SAS. Oxford, Oxford University Press.
- Bates, D., Maechler, M., Bolker, B., Walker, S. (2015). Fitting Linear Mixed-Effects Models Using lme4. *Journal of Statistical Software*, 67(1), 1-48. doi:10.18637/jss.v067.i01.
- Durban, M., Hackett, C., McNicol, J., Newton, A., Thomas, W., & Currie, I. (2003). The practical use of semi-parametric models in field trials, *Journal of Agric. Biological and Envir. Stats.*, 8, 48-66.
- Edmondson R. N. (1993). Systematic Row-and-column Designs Balanced for Low Order Polynomial Interactions between Rows and Columns. *J. R. Statist. Soc. B* (1993) 55, No. 3, pp. 707-723
- Edmondson, R. N. (1998). Trojan square and incomplete Trojan square designs for crop research. *Journal of Agricultural Science, Cambridge* (1998), 131, 135–142.
- John, J. A & Williams, E. R. (1998). *Cyclic and Computer Generated Designs*. 2nd Edition, Chapman and Hall.
- Martin Oliver Sailer (2013). crossdes: Construction of Crossover Designs. R package version 1.1-1. <https://CRAN.R-project.org/package=crossdes>
- Piepho, Hans-Peter & Edmondson R. N. (2018). A tutorial on the statistical analysis of factorial experiments with qualitative and quantitative treatment factor levels. *Journal of Agronomy and Crop Science*, 204, 429-455.