Vignette for 'blocksdesign' package

Introduction

Comparative experiments in agriculture and biology often involve the estimation of treatment effects against a background of high non-treatment variability. Treatments can be unstructured such as comparisons between individual varieties or regimes or they can have complex factorial structures such as factorial combinations of different fertilizer types. Unstructured treatments usually assume that all pairwise treatment comparisons are of equal importance but structured treatments usually require a treatment model for the effects of the different treatment components. Effective treatment comparison requires effective control of background variability and most agricultural and biological experiments use replicated treatments divided into blocks for the control of background variability. The aim of good block design is to group treatments into blocks that are as homogeneous as possible so that within-blocks treatment comparisons are made with improved precision relative to between-blocks treatment comparisons. Good experimental design requires both good treatment design and good block design and the 'blocksdesign'package is intended to provide an integrated general purpose design package for treatment and block designs especially for field and crop experiments.

Treatments designs

Treatment design for unstructured treatment sets is chiefly concerned with the choice of individual treatments and choice of replication and these choices usually depend on the purposes and economics of a trial and are not amenable to optimization methods. Structured treatments sets such as factorial designs or response surface designs (see Piepho and Edmondson 2018), however, usually depend on empirical linear models for interpretation and 'blocksdesign' can optimize the choice of treatments for any empirical linear treatment model for any combination of factorial or polynomial treatment model. The treatment design is selected from a candidate set of treatments by choosing a fixed number of treatments that optimize the information matrix of the treatments design using the D-optimality criterion (see Atkinson et. al. 2007). Treatments are selected from the candidate set with replacement unless the size of the candidate set exactly equals the size of the required design in which case the treatments design can be selected either with or without replacement. Selection without replacement allows the full candidate set to be selected and allows any arbitrary treatment set with any arbitrary replication to be used as the treatment design.

Blocks designs

Often experimental units will have high background variability and then proper account must be taken of plot-to-plot variability. In many situations, comparability between treatments can be improved by

grouping the experimental units into blocks that are as homogeneous as possible and the choice of blocks design can be critical for the success of an experiment. The most basic type of are complete randomized blocks 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, the variability within complete blocks can be large and then it may be beneficial to further sub-divide each complete replicate block into smaller nested sub-blocks to improve the precision of the within-sub-block treatment 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) and the R mixed model software package lme4 (Bates et. al. 2014). 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. Various design criteria have been considered for block design (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' package.

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, however, if the block size of a single set of nested blocks is small enough to give good within-block homogeneity of variance, the inter-block space will be large and will confound a substantial amount of treatment information between blocks. Furthermore, the inter-block space will be large and may be highly heterogeneous. In these circumstances, the efficiency of the inter-block analysis will be low leading to a less than optimal recovery of inter-block information.

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 of the nested inter-block space will decrease with depth of nesting and in these situations top-down optimization should ensure that the minimum possible amount of treatment information is confounded within each nested inter-block space. 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 nested inter-block space.

Factorial block designs

Sometimes it can be advantageous to use a double blocking system in which one set of blocks is crossed with a second set of blocks in a fully factorial block design. In field trials, the two sets of blocks are often called row blocks and column blocks, respectively, and often coincide with physical rows and columns in the design layout. Double blocking can be valuable for controlling block effects in two dimensions simultaneously and crossed blocks designs are often assumed to fit a simple additive main effects model with additive row and additive column effects. However, additivity of block effects is a very strong assumption and unlikely to be valid for crossed block factors with many levels such as field designs with long rows, long columns and single plot row-by-column intersections. Crossed block designs with fewer levels per factor and with blocks of two or more plots in each interaction provide more flexibility for modelling factorial block effects but require assumptions at the design stage about the relative importance of possible block interaction effects versus block main effects.

Factorial block interactions

Assume that the block intersections of a crossed blocks design contain two or more plots and that the crossed blocks interaction effects are estimable. It is straightforward to optimize the main effects of a crossed block design by using a sequential fitting approach but once the row and column blocks are fixed it is not possible to optimize the rows-by-columns blocks by a simple swapping algorithm. Let $(\boldsymbol{T}, \boldsymbol{X}, \boldsymbol{Z})$ represent the model matrix for a crossed blocks design where $\boldsymbol{T}$ is the treatments design matrix and the block contrasts are partitioned into two orthogonalized sets $\boldsymbol{X}$ and $\boldsymbol{Z}$ such that $\boldsymbol{X}$ represents the main effects of the blocks design and $\boldsymbol{Z}$ represents the rows-by-columns interaction effects Then the 'blocksdesign' algorithm optimizes the following weighted blocks adjusted treatment information matrix determinant for $0 \leq \boldsymbol{w} \leq 1$:

$$Det\big(\, Info(w)\big) = \; \left|\boldsymbol{T}'\big(\boldsymbol{I} - \boldsymbol{X}\boldsymbol{X}' - \boldsymbol{Z}\boldsymbol{Z}'\boldsymbol{w}^{\boldsymbol{2}}\big)\boldsymbol{T}\,\right| \qquad (1)$$

If $w = 0$ the information matrix in (1) is the usual additive crossed blocks model whereas if $w = 1$, the information matrix is just the full factorial interaction blocks model. However, for intermediate weighting, $0 < w < 1$, the information matrix (1) down-weights the block interaction effects $\boldsymbol{Z}\boldsymbol{Z}'$by the square of the weighting parameter w. The best choice of down-weighting parameter w will depend on the design efficiencies of the various factorial block effects and is best found by trial error at the design stage, see examples below.

Design optimization

The 'blocksdesign' package has two design optimization functions:

blocks: a special recursive function for nested block designs for unstructured treatment sets. The function generates designs for treatments with arbitrary levels of replication and arbitrary depth of nesting where 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, 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 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.

design: a general purpose function for unstructured or general qualitative or quantitative factorial treatment models. The design inputs include a candidate set of treatments, a treatments model and the required blocks design. If the candidate set and the required design size differ, the treatment design is always selected from the candidate with replacement to optimize the information matrix of the required treatment model. If the number of candidate treatments exactly equals the design size, the full candidate set of treatments will normally provide the treatment design but a parameter 'fullset' can be set to FALSE if selection with replacement is required. After the treatment design has been selected, the blocks design algorithm builds the blocks design by sequentially adding blocks factors where each block factor is optimized conditional on all previous block factors. If the design has estimable crossed block factor interaction effects the blocks main effects and interaction effects are optimized as shown in equation (1). Outputs include a data frame of the block and treatment factors for each plot and a table showing the achieved D-efficiency factors for each set of nested or crossed blocks. Fractional factorial efficiency factors based on the generalized variance of the complete factorial design are also shown (see the design documentation for more details).

Example designs for unstructured treatment sets

Example 1 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 should always attain the theoretical A-efficiency upper bound

Example 2 Four replicates of 12 treatments and 16 replicates of one additional treatment in 4 complete blocks with 4 sub-blocks nested in each main block

Inputs
blocks(treatments = c(12, 1), replicates = c(4, 1), blocks = c(4, 4))

Outputs

```
$blocks_model
    Level Blocks D-Efficiency A-Efficiency A-Bound
1 Level_1      4            1            1    <NA>
2 Level_2     16    0.8059274          0.8    <NA>


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

This design should be identical with Example 1 except that a single control treatment (16 replicates) has been added to each of the 16 nested blocks. This design is constructed algorithmically by 'blocksdesign' but in this example it would have been simpler (and more reliable) to add the control treatment to each block in Example 1 by hand. In other situations the replication pattern and the blocks pattern may not match so conveniently and then the algorithmic method is likely to be more efficient and more reliable than a simple heuristic method for adding control treatments. Upper efficiency upper bounds are not available because the design is not equally replicated but the efficiency factors can be useful for comparing different optimizations to ensure that that an optimal or near-optimal design has been found.

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

Inputs
blocks(128, 2, c(2, 2, 2, 2, 2, 2))

Outputs

```
$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_2 Blocks_1 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_1 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
```

This design shows the capability of blocksdesign for repeated or recursive nesting of very large block designs. The main replicate blocks comprise 128 plots and these main blocks are recursively split into pairs of sub-blocks at each of 5 levels of nesting. The optimization criterion is D-optimality but the $blocks_model output shows both the optimized D-efficiency and the A-efficiency for each level of nesting. As the design is equally replicated and has equal block sizes, A-efficiency upper bounds exist for each level of nesting and the $blocks_model efficiency factors show that the blocks design at each

level of nesting is quite close to the theoretical upper bound except possibly for the bottom level of nesting. However, it is known that the estimated upper bounds become unreliable as the efficiency of the design is reduced therefore the bottom level bound of 0.566227 is probably not very useful.

For a more useful test, it is better to compare the block efficiencies of the multi-level nested design with the corresponding efficiencies of a set of nested blocks of the same size but from a single-level nested design. The following example shows the efficiencies of blocks of size four from a single-level nested blocks design with 32 blocks of size 4 nested in replicate blocks of size 128:

Example 3b. Two replicates of 128 treatments with 2 main blocks and 32 blocks of size 4 nested in each main block.

Inputs
blocks(128, 2, c(2, 32))

```
Outputs
$blocks_model
    Level Blocks D-Efficiency A-Efficiency  A-Bound
1 Level_1      2            1            1        1
2 Level_2     64    0.6656609    0.5441637 0.566227
```

The D and A-efficiencies from the bottom level of the multi-level design were 0.6654802 and 0.5427813, respectively, which shows that the reduction in efficiency of the bottom level blocks of the multi-level design due to the constraints imposed by the higher level blocks of the design was small especially for the D-optimality criterion which was the criterion used for design optimization.

Example designs for general block and treatment designs

Example 4 Four replicates of 12 treatments with 4 main replicate rows and 4 main replicate columns.

Inputs

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

Outputs

```
Blocks levels D-Efficiency A-Efficiency Interactions levels D-Efficiency A-Efficiency
Rows      4            1            1         Rows      4            1            1
Cols      4            1            1    Rows*Cols     16    0.7176709    0.7096774
```

This example shows a row and column block design for 4 replicates of 12 treatments with blocks of size 3 in each row-by-column intersection. The design has efficiency 1 for the rows and columns blocks and this shows that the design has a complete set of treatments in each row and each column block. The default weighting of 0.5 was used which gave w = 0.25 in equation (1) and this weighting gave D and A-efficiencies of 0.7176709 and 0.7096774, respectively, for the 16 row-by-column intersection blocks. The design() function does not show upper bounds for the efficiency factors but for an equireplicate design with equal sized blocks, an A-efficiency upper bound can be found by using the blocksdesign::A_bound(n, v, b) function where n is the total number of plots, v is the total number of treatments and b is the total number of blocks. Using this function gives: A_bound(48,12,16) = 0.7096774 for the intersection blocks which shows that the design in Example 1 attained the upper efficiency bound in each of the Rows, Columns and Rows*Columns block structures. The example is a special design called a Trojan square and normally not all three block

structures for this type of crossed block design can be optimized simultaneously. Nevertheless, the example shows the utility of the weighting method for dealing with crossed row and column type designs with blocks of plots nested in the rows-by-columns intersections.

Example 5 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.

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-Efficiency A-Efficiency      Interactions levels D-Efficiency A-Efficiency
Rows      4            1            1            Rows      4            1            1
Cols      4            1            1         Rows*Cols     16    0.7176709    0.7096774
subCols  12    0.8053142    0.7925806 Rows*Cols*subCols     48         <NA>         <NA>
```

This example extends the design in Example 4 by nesting 3 sub-columns in each main column to give a physical layout for an experiment with four rows and 16 columns where the 16 columns are arranged in four replicate main columns blocks each containing 3 nested sub-columns. The main rows, main columns and main intersection block efficiencies are unchanged from example 4 because the sub-column blocks have been optimized by swaps made *within* the main row-by-column intersection blocks. The efficiency factors for the sub-column blocks can be compared with the efficiencies of a simple nested block design for 4 replicates of 48 treatments with 4 main blocks and 3 sub-blocks in each main block by using the blocks(12,4,c(4,3))$blocks_model which gives D and A-efficiency factors of 0.8053142 and 0.7925806 respectively. In this example, the sub-column blocks are fully efficient compared with a simple nested blocks design with blocks of the same size and there is no loss of efficiency due to the additional constraints of the row and column layout of the design. Note that the main rows and the sub-columns intersect in single plots therefore is no estimate of variability and no efficiency factors for the main rows by sub-columns interactions.

Example 6 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$gen)
Reps = factor(rep(1:2,each=272))
Rows = factor(rep(1:16,each=34))
Col1 = factor(rep(rep(1:4,c(9,8,8,9)),16))
Col2 = factor(rep(rep(1:8,c(5,4,4,4,4,4,4,5)),16))
Col3 = factor(rep(1:34,16))
blocks = data.frame(Reps,Rows,Col1,Col2,Col3)
design(treatments,blocks,searches=1)$blocks_model
```

Outputs

```
Blocks levels D-Efficiency A-Efficiency         Interactions levels D-Efficiency A-Efficiency
Reps      2           1           1                       Reps    2           1           1
Rows     16   0.9647882   0.9507384                  Reps*Rows   16   0.9647882   0.9507384
Col1      4    0.995518   0.9944849               Reps*Rows*Col1   64   0.8437223    0.781362
Col2      8   0.9853798   0.9800809      Reps*Rows*Col1*Col2  128   0.6762391   0.5377895
Col3     34   0.9207087   0.8914904 Reps*Rows*Col1*Col2*Col3  544        <NA>        <NA>
```

This example explores an alternative blocking system for a real experimental design. The original design (see 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 7 Second-order model for a 1/3rd fraction of five qualitative 3-level factors in 3 blocks of size 27

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.model                     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
```

This example shows how the blocksdesign::design() function can fractionate and blocks factorial treatment designs. The 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 easily 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.

Example 8 Second-order response surface design for three 3-level factors assuming a 10 point design

Input

```
treatments = expand.grid(A = 1:3, B = 1:3, C = 1:3)
blocks=data.frame(main=gl(1,10))
model = " ~ ( A + B + C)^2 + I(A^2) + I(B^2) + I(C^2)"
design(treatments,blocks,treatments_model=model,searches=5)

$treatments_model
 Treatment.model                                 D.Efficiency
~ ( A + B + C)^2 + I(A^2) + I(B^2) + I(C^2)    0.9262674

$design
   main A B C
1     1 1 2 2
2     1 2 3 1
3     1 2 1 2
4     1 1 1 3
5     1 2 2 3
6     1 3 3 3
7     1 1 1 1
8     1 3 2 1
9     1 3 1 3
10    1 1 3 3
```

This example 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 therefore a design based on only 10 point is almost saturated and may not be useful for model checking. Nevertheless, this choice of factorial combinations will give the maximum information on the assumed model. The D-efficiency factor is the ratio of the D-optimum efficiency of a design based on the full candidate set versus the D-optimum efficiency of the optimized design and has no special meaning as the full candidate set is not a natural choice of design for this treatment model. Nevertheless, the standardisation makes it simple to compare the efficiency of different optimizations and normally the design with the largest treatment design efficiency factor will be the preferred choice of design

Comment on quantitative nuisance factors

The ‘blocksdesign’ algorithm distinguishes between treatment factors and block factors because the treatments are the factors of interest whereas block factors are merely nuisance factors which need to be allowed for in the model fitting process. Therefore it makes sense to optimize the treatment factors

first before the block factors to ensure maximum precision on the treatment factors irrespective of whether the block factors prove useful or not. The blocksdesign' algorithm fits only qualitative block factors for factorial block effects but, if required, it would be possible to include quantitative polynomial nuisance effects as part of the 'treatment' effects model. However, that would require including the quantitative polynomial nuisance factors in the treatments candidate treatment set and it is not clear if selection with replacement would necessarily provide a suitable fraction for the actual required treatment model. Until further study is undertaken, the best option for robust trend or smoothing models is to use a multi-level nested block design with a sufficiently complex block structure to provide for good local control of variability at the scale required for trend or smoothing models.

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.

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.