Vignette for 'blocksdesign' package

Summary

Block designs are the designs of choice for comparative linear model experiments. Complete replicate blocks are fully efficient for treatment estimation but may not provide good homogeneity of variance. Incomplete blocks can improve homogeneity of variance but a single set of blocks may not be adequate for large numbers of treatments. Multi-level nesting provides a range of block sizes in the same experiment and this vignette discusses designs for hierarchies of multi-level blocks including designs with crossed block factors.

Introduction

Comparative experiments often involve estimation of treatment effects against a background of high non-treatment variability and effective control of background variability is essential for good treatment estimation. The most common method for field trials is the complete randomized blocks design in which every treatment is represented in every block in proportion to its replication. Complete block designs can be effective against a range of nuisance effects such as patchy fertility, row-and-column effects or even the residual effects of previous treatments but for large designs with many treatments, complete randomized blocks may be too large to give good homogeneity of variance. In that situation, incomplete blocks containing fewer than a complete set of treatments in each block can be used to obtain improved homogeneity of variance within blocks. See Bailey (1999 and 2008).

Incomplete block designs with a single level of nesting assume that background variation can be partitioned into components comprising a component of variation within blocks and a component of variation between blocks. However, for experiments with many treatments, the components of variation may occur over a range of scales of measurement and then designs with a single level of nesting may not be adequate to capture all the important sources of non-treatment variability. In that situation, repeated or multi-level nesting over a range of block sizes may be needed to capture background variability over the full range of all scales of measurement.

Comparative experiments in agriculture and biology often involve comparisons between unstructured treatments such as varieties or comparisons between factorial treatments such as different fertilizer types or comparisons between quantitative level treatments such as rates of fertilizer application. Treatment design for an unstructured treatment set 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. Good design of nested incomplete blocks also usually requires computer methods. 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 design

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 perhaps to lack of resources or lack of experimental material, some treatments may need to 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 whatever the choice of treatment replication.

Example

The following example shows a simple basic design for a set of 12 unstructured treatments numbered 1 to12 each with 4 replicates and a single control treatment, numbered 13, with 8 replicates. The example does not specify a block design and the design defaults to four complete randomised blocks each containing treatments 1 – 12 once and treatment 13 twice. Unstructured treatment designs can be constructed either by the "blocksdesign::blocks()" function or by the "blocksdesign::design()". The design() function is more general than the blocks() function and provides more control over the blocks and treatment design but requires a more detailed set of input parameters. In this example, the blocks() function will be used to provide a simple basic block design:

Model

```
blocks(treatments = list(12,1), replicates = list(4,8))
```

Output

```
$Blocks_model
    Level Blocks D-Efficiency A-Efficiency A-Bound
1 Level_0      4            1            1       1
```

The output shows the block efficiency table for the design where the column labelled 'Level' indicates the depth of nesting. In this case, there is just one set of complete randomized blocks without nesting, the nested level is zero. The A-efficiency bound is only available for equi-replicate designs with equal block sizes or for complete orthogonal block designs, as in the above example.

The other design outputs for the blocks() function include $Treatments, showing a table of the treatment replicates, $Design showing a table of the allocation of treatments to blocks and $Plan showing a plan of the allocation of treatments to blocks in the bottom level of blocking.

Structured treatments

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

The 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 the feasible treatment combinations that might occur in the final optimized design and the design algorithm selects those combinations that optimize the treatment information matrix. Treatments can be selected either with or without replacement depending on a resample parameter which can be set to either TRUE or FALSE. If resample is TRUE, selected treatments are replaced in the candidate set, which allows for repeated sampling of the selected treatments, whereas if resample is FALSE, selected treatments are deleted from the candidate set, which means that no treatment can be repeated in the final design more often than it occurs in the original candidate set.

The treatments model is either a single formula or a compound formula that is split by the operator |. Splitting operators define a sequence of partial treatment models with the left hand side of each splitting operator defining a partial model with all preceding splitting operators replaced by the summation operator +. Partial model formulae define partial treatment designs which are fitted and optimized sequentially from left to right. Sequential model fitting provides flexibility for fitting factors or variables of different status or importance.

For example, the following example constructs a treatment design for 4 varieties with 3 levels of N and 3 levels of K assuming a degree-2 fertilizer response surface. Model 1 fits all the treatment terms simultaneously using a single optimisation:

Model 1

```
treatments = expand.grid(Variety = factor(1:4), N = 1:3, K = 1:3)
blocks = data.frame(main = gl(2,12))
model = " ~ (Variety + N + K)^2  + I(N^2) + I(K^2)"
design(treatments, blocks, treatments_model = model, searches=10)
```

Output (model 1)

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

```
13        3 1 3    1
14        3 2 1    1
15        3 3 1    1
16        3 3 3    1
17        4 1 1    1
18        4 1 2    1
19        4 1 3    1
20        4 2 1    1
21        4 2 3    1
22        4 3 1    1
23        4 3 2    1
24        4 3 3    1


$Treatments_model
                              Treatment.model Model.DF D.Efficiency
1  ~ (Variety + N + K)^2  + I(N^2) + I(K^2)       14     1.052045
```

Model 2 fits the variety effects and the fertilizer effects sequentially with the four variety levels fitted first and the fertilizer effects fitted conditionally on the fixed variety effects.

Model 2

```
treatments = expand.grid(Variety = factor(1:4), N = 1:3, K = 1:3)
blocks=data.frame(main = gl(2,12))
model = " ~ Variety | (Variety + N + K)^2  + I(N^2) + I(K^2)"
design(treatments, blocks, treatments_model = model, searches=10)


$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 1    1
10       2 2 3    1
11       2 3 1    1
12       2 3 3    1
13       3 1 1    1
14       3 1 3    1
15       3 2 2    1
16       3 2 3    1
17       3 3 1    1
18       3 3 3    1
19       4 1 1    1
20       4 1 2    1
21       4 1 3    1
22       4 3 1    1
23       4 3 2    1
24       4 3 3    1
```

```
$Treatments_model
                                          Treatment.model Model.DF D.Efficiency
1                                              ~ Variety         3            1
2  ~ Variety + (Variety + N + K)^2  + I(N^2) + I(K^2)       14     1.042662
```

For Model 1 the numbers of factor combinations of the four variety levels vary between 5 and 8, which few research workers would regard as acceptable, whereas for Model 2 each variety is equally replicated. The overall D-efficiency factor of Model 2 is slightly less than the overall D-efficiency of Model 1 but the model now has an equal division of the 24 factor combinations into 6 combination for each level of variety which most research workers would regard as a necessary requirement for a good design. NB the ^2 square operator is an interaction operator unless 'protected' by the I() operator.

Simple nested block designs

In many situations, comparability between treatments can be improved by grouping 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.

Multi-level nesting

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.

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 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. 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.

The blocks() function of 'blocksdesign' 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). 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.

The following examples show a simple recursively nested block design for four replicates of 100 treatments with four complete main blocks for the top level of blocks and two levels of nesting where the first level of nesting has 10 sub-blocks of size 10 nested within each main block and the second level of nesting has two sub-sub-blocks of size 5 nested within each nested sub-block.

Model

```
blocks(100, 4, list(4,10,2))$Blocks_model
```

Block efficiencies

```
    Level Blocks D-Efficiency A-Efficiency A-Bound
1 Level_0      4            1            1       1
2 Level_1     40      0.90067      0.89189 0.89189
3 Level_2     80      0.78476      0.75946 0.76327
```

The A-efficiency of the Level_1 blocks is optimal because the Level_1 blocks are lattice blocks based on a pair of orthogonal 10 x 10 Latin squares. The A-efficiency of the Level_2 blocks is close to the theoretical upper A-bound which shows that the constraints of multi-level nesting have not significantly reduced the efficiency of the Level_2 blocks design.

Factorial nested block designs

Sometimes it can be advantageous to use a fully crossed factorial blocks design in field trials. For example, factorial row-and-column blocks are sometimes used to accommodate physical row and column effects in a field layout. Factorial block designs 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. For that reason, the "blocksdesign" algorithm can fit block main effects and block 2-factor interaction effects weighted by an assumed model for the relative

importance of main effects versus 2-factor interaction effects, assuming that all model effects are estimable.

Weighted factorial treatment models

Let $\mathbf{T}$ be a matrix of contrasts for a set of treatments factors and let $\mathbf{B_1}$ and $\mathbf{B_2}$ be contrast matrices for the additive effects of two sets of crossed block factors. Let $\mathbf{B_{2:1}}$ be the matrix of two-factor interactions between $\mathbf{B_1}$ and $\mathbf{B_2}$. Then the full block and treatment design matrix is:

$$\mathbf{T} + \mathbf{B_1} + \mathbf{B_2} + \mathbf{B_{2:1}}$$

Assume $\mathbf{B} = (\mathbf{B_1}, \mathbf{B_2}, \mathbf{B_{2:1}})$ is of full rank and assume a singular value decomposition $\mathbf{B} = \mathbf{QR}$ where $\mathbf{Q} = (\mathbf{Q_1}, \mathbf{Q_2}, \mathbf{Q_{2:1}})$ is a conformal orthogonal basis for $\mathbf{B}$. Since $\mathbf{Q'Q} = \mathbf{I}$ and $(\mathbf{R'R})^{-1} = \mathbf{R^{-1}R'^{-1}}$, the block adjusted treatment information matrix $\mathbf{T'}\left(\mathbf{I} - \mathbf{B}(\mathbf{B'B})^{-1}\mathbf{B'}\right)\mathbf{T}$ can be re-written as:

$$\mathbf{T'}(\mathbf{I} - \mathbf{Q_1}\mathbf{Q_1}' - \mathbf{Q_2}\mathbf{Q_2}' - \mathbf{Q_{2:1}}\mathbf{Q_{2:1}}')\mathbf{T}$$

The "blocksdesign" algorithm optimizes a weighted treatment information matrix:

$$\mathbf{T'}\left(\mathbf{I} - \mathbf{Q_1}\mathbf{Q_1}' - \mathbf{Q_2}\mathbf{Q_2}' - w^2 \times \mathbf{Q_{2:1}}\mathbf{Q_{2:1}}'\right)\mathbf{T}$$

where the two-factor interaction term $\mathbf{Q_{2:1}}\mathbf{Q_{2:1}}'$ is down-weighted by an arbitrary scalar w^2 for $0 \leq w \leq 1$.

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

The following three examples show crossed blocks design for 4 replicates of 12 treatments with 4 main rows and 4 main columns. and blocks of size 3 nested within each row-by-column intersection. The row blocks are added first and will always comprise complete replicate blocks. The column blocks are added after the row blocks and the efficiency of the column block main effects and the row-by-column interaction block effects will depend on the choice of weighting. The three examples show the effects of three different choices of weighting parameter on the relative importance of the 2-factor row-by-column interaction effects.

In Model 1 the weighting is zero and additive column block effects are orthogonal and are estimated with full efficiency but the rows-by-columns block interaction effects are estimated with low efficiency.

In Model 2 the weighting is 0.5 and the rows-by-columns block effects are estimated with improved efficiency relative to Model 1 while the additive column block effects still remain orthogonal and are still estimated with full efficiency.

In Model 3 the weighting is 1 and the rows-by-columns block interaction effects are estimated with the maximum possible efficiency. However, the additive column blocks are no longer orthogonal and are estimated with efficiency less than one. The Model 3 design is equivalent to an incomplete block

design with four complete main blocks (rows) and with four sub-blocks (row-by-column interaction blocks) nested within each main block.

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

Model 1

```
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
```

Output

```
  Blocks Levels D_Effic A_Effic Interactions Int_levs Int_D_Effic Int_A_Effic
1   Rows      4       1       1         Rows        4    1.000000   1.0000000
2   Cols      4       1       1    Rows*Cols       16    0.682796   0.6316198
```

---

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

Model 2

```
treatments = factor(1:12)
blocks = data.frame(Rows = gl(4,12), Cols = gl(4,3,48))
design(treatments,blocks,searches=200,weighting=0.5)$blocks_model
```

Output

```
  Blocks Levels D_Effic A_Effic Interactions Int_levs Int_D_Effic Int_A_Effic
1   Rows      4       1       1         Rows        4   1.0000000   1.0000000
2   Cols      4       1       1    Rows*Cols       16   0.7176709   0.7096774
```

---

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

Model 3

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

Output

```
  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 0.9144364 0.9034965    Rows*Cols       16   0.7176709   0.7096774
```

In this example, Model 2 gives a fully orthogonal set of main column blocks and fully efficient rows-by-columns design compared with Model 3 which has full weighting for row-by-column effects. Usually, for crossed blocks designs, not all rows, columns and rows-by-columns can be optimized simultaneously but this example is a special design called a Trojan square (Edmondson 1998) which has the property that the main rows and columns blocks design and the rows-by-columns interaction design can all be optimized simultaneously. This exemplifies the utility of the weighting method for designs with estimable crossed blocks interaction effects. 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.

Additional examples

Durban et.al. (2003) discussed an experiment with two replicates of 272 spring barley varieties arranged in an array of 34 columns (east-west) and 16 rows (north-south) subject to the constraint that rows 1-8 contained one complete set of treatment replicates and rows 9-16 contained the other. They showed that an analysis based on a conventional additive row-and-column model was inadequate due to residual trends within rows Under these circumstances, it seems natural to model rows and columns using nested column blocks.

The original barley variety trial was designed as a simple row-and-column design with 16 rows and 34 columns but, for the purposes of this example, it is reasonable to assume a set of nested column blocks imposed on the original design and to assume interacting row-and-column blocks.

Model 1

The original design had 16 rows and 34 columns and these are modelled here by a row factor Rows with 16 levels crossed with three nested columns factors Col1, Col2 and Col3 with 4, 8 and 34 levels, respectively. Assuming 34 plots within each row, not all the nested column block factors can be of equal width and two column blocks from each of Col1, Col2 and Col3 must be an extra plot wide.

Input

```
treatments = factor(rep(1:272,2))
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)
```

Output

```
First-order model effects
            First_order_model effects D.Efficiency A.Efficiency
                       (Reps)       1      1.00000      1.00000
                 (Reps+Rows)      15      0.96492      0.95091
            (Reps+Rows+Col1)      18      0.96060      0.94593
       (Reps+Rows+Col1+Col2)      22      0.95084      0.93288
 (Reps+Rows+Col1+Col2+Col3)      48      0.88785      0.84983
```

```
Second-order model effects
            Second_order_model effects D.Efficiency A.Efficiency
                     (Reps)^2       1      1.00000      1.00000
                 (Reps+Rows)^2      15      0.96492      0.95091
             (Reps+Rows+Col1)^2      63      0.84423      0.78183
       (Reps+Rows+Col1+Col2)^2     127      0.67750      0.54116
 (Reps+Rows+Col1+Col2+Col3)^2     543      0.00000      0.00000
```

Model 2

The original design had 34 plots in each row and not all columns in each set of columns were of equal width. A simpler column blocks design can be based on a design with 32 plots in each row but this design requires 17 rows to accommodate the two replicates of 272 treatments. The following design shows a 'Reps' factor with 3-levels, which splits the design into rows 1:8 with 256 plots, row 9 with 32 plots and rows 10:17 with 256 plots. The design algorithm allocates 256 treatments, all different, to rows 1:8, 32 treatments, all different, to row 9 and 256 treatments, all different, to rows 10:17.

Input

```
treatments = factor(rep(1:272,2))
Reps = factor(c(rep(1,256),rep(2,32),rep(3,256)))
Rows = factor(rep(1:17,each=32))
Col1 = factor(rep(rep(1:4,each=8,17)))
Col2 = factor(rep(rep(1:8,each=4,17)))
Col3 = factor(rep(1:32,17))
blocks = data.frame(Reps,Rows,Col1,Col2,Col3)
d=design(treatments, blocks, searches=1)
# incidences to check the treatment partition between the three levels of Reps
table(d$Design[,1],d$Design[,6])
```

Output

```
           First_order_model effects D.Efficiency A.Efficiency
1                     (Reps)       2      0.99756      0.99665
2                 (Reps+Rows)      16      0.96256      0.94786
3             (Reps+Rows+Col1)      19      0.95825      0.94290
4       (Reps+Rows+Col1+Col2)      23      0.94852      0.92994
5 (Reps+Rows+Col1+Col2+Col3)      47      0.89042      0.85338
```

```
           Second_order_model effects D.Efficiency A.Efficiency
1                     (Reps)^2       2      0.99756      0.99665
2                 Reps+Rows)^2      16      0.96256      0.94786
3             (Reps+Rows+Col1)^2      67      0.83372      0.76597
4       (Reps+Rows+Col1+Col2)^2     135      0.65676      0.51247
5(Reps+Rows+Col1+Col2+Col3)^2     543      0.00000      0.00000
```

Row 9 of Model will be split between the two replicate sets with 8 plots from row 9 combined with rows 1:8 to give replicate 1 and eight plots from row 9 combined with rows 10:17 to give replicate 2. Overall, Model 2 should give a smaller maximum pairwise SED for all possible treatment comparisons than Model 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.

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.