Notes on the use of `dae` for design

C. J. Brien

March 23, 2019

Contents

1 Introduction

The R (R Core Team, 2019) package `dae` (Brien, 2018) provides functions useful in the design and anova of experiments. This document describes how to use some of them to produce layouts for experiments and to check some of their properties.

1.1 Functions to be used

The functions in `dae` fall into the following categories and those that will be covered in this document are listed and described:

1. Data

   BIBDWheat.dat Data for a balanced incomplete block experiment.

   Casuarina.dat Data for an experiment with rows and columns from Williams et al. (2002).

   Exp249.mplot.des Systematic, main-plot design for an experiment to be run in a greenhouse.

   Fac4Proc.dat Data for a 2^4 factorial experiment.

   LatticeSquare_t49.des A Lattice square design for 49 treatments.

   McIntyreTMV.dat The design and data from McIntyre (1955) two-phase experiment.

   Oats.dat Data for an experiment to investigate nitrogen response of 3 oats varieties from Yates (1937).

   Sensory3Phase.dat Data for the three-phase sensory evaluation experiment in Brien and Payne (1999).

   Sensory3PhaseShort.dat Data for the three-phase sensory evaluation experiment in Brien and Payne (1999), but with short factor names.

   SPLGrass.dat Data for an experiment to investigate the effects of grazing patterns on pasture composition.

2. Factor manipulation functions

   fac.gen: Data for an experiment to investigate nitrogen response of 3 oats varieties.

   fac.recode: Recodes the levels and values of a factor.

   fac.combine: Combines several factors into one.

   fac.divide: Divides a factor into several individual factors.

3. Design functions

   blockboundaryPlot: Plots a block boundary on a plot produced by designPlot.

   designAnatomy: Given the layout for a design, obtain its anatomy via the canonical analysis of its projectors to show the confounding and aliasing inherent in the design.

   designLatinSqrSys: Generate a systematic plan for a Latin Square design.

   designPlot: A graphical representation of an experimental design using labels stored in a matrix.

   designPlotlabels: Plots labels on a two-way grid using ggplot2.

   designRandomize: Takes a systematic design and randomizes it according to the nesting (and crossing) relationships between the recipient(unit) factors for the randomization.

   no.reps: Computes the number of replicates for an experiment.

   summary.pcanon: Summarizes the anatomy of a design, being the decomposition of the sample space based on its canonical analysis, as produced by designAnatomy. The table produced includes the degrees of freedom and summary statistics of the canonical efficiency factors.

   summary.pcanon: Summarizes the anatomy of a design, as produced by designAnatomy, being the decomposition of the sample space based on its canonical analysis.

4. ANOVA functions

5. Matrix functions

6. Projector and canonical efficiency functions

   efficiencies.pcanon: Produces a list containing the canonical efficiency factors for the joint decomposition of two or more sets of projectors (Brien and Bailey, 2009) obtained using `designAnatomy`.

7. Miscellaneous functions.

Documentation for each of these functions is available from the user manual (`vignette("dae-manual", package="dae")`) and their are some notes that show how to use the functions (`vignette("DesignNotes", package="dae")`).

1.2 The paradigm

Fundamental to the approach in this document, and to using the functions described, is that a single allocation involves allocating a set of *allocated* factors to a set of *recipient* factors. In many designs, this allocation is achieved by randomization. However, sometimes there is systematic allocation or restricted allocation.

2 Single-allocation orthogonal design in R

This class of experiments covers the orthogonal standard or textbook experiments and these experiments must be single phase because they involve a single randomization, in the sense that the randomization can be achieved with a single permutation. Hence there will be two sets of factors, or tiers, one set being allocated to the other set. In `designRandomize`, these are referred to as the allocated and recipient sets of factors. They are also called the unit and treatment factors, respectively.

Firstly, initialize by loading the library that will be used.

```
library(dae)

## Loading required package:  ggplot2

options(width=100)
```

2.1 Two potential designs for a 5×5 grid of plots

Suppose an experiment to investigate five treatments is to be conducted on 25 plots, the 25 plots being arranged in a 5×5 grid. Two possible designs are a randomized complete block design (RCBD) or a Latin square design (LSD). The factor-allocation diagram (Brien et al., 2011) for the RCBD is in Figure 1 and that for the LSD is in Figure 2.

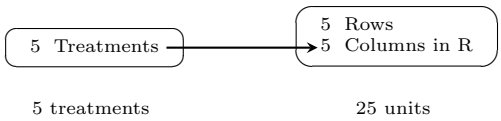


Figure 1: Factor-allocation diagram for an RCBD: treatments are allocated to units; the arrow indicates that the factor Treatments is randomized to Columns; Columns in R indicates that the Columns are considered to be nested within Rows for this randomization; R = Rows.

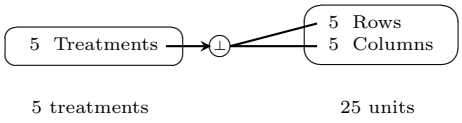


Figure 2: Factor-allocation diagram for a LSD: treatments are allocated to units; the arrow indicates that the allocation is randomized; the '①' at the end of the arrow indicates that an orthogonal design is used; the two lines from '①' indicates that the Treatments are allocated to the combinations of Rows and Columns using the design.

2.1.1 Produce the randomized layout for an RCBD

Use `designRandomize` to randomize the treatments according to an RCBD. The arguments to `designRandomize` that need to be set are (i) `allocated`, (ii) `nested.recipients`, (iii) `recipient`, and optionally, (iv) `seed`. The allocated factors as treatment factors and recipient factors are also referred to as block or unit factors. Often, the recipient factors and the nesting of the recipient factors are specified by `lists`. This will result in the recipient factors being generated in standard order. A systematic arrangement of the allocated factors, corresponding to the values of the recipient factors, needs to be supplied and this can be done via `factor` for a single factor or a `data.frame` for several allocated factors. The name of the allocated factor(s) in the layout produced will be the name of the supplied `factor` or the names of the `data.frame` columns.

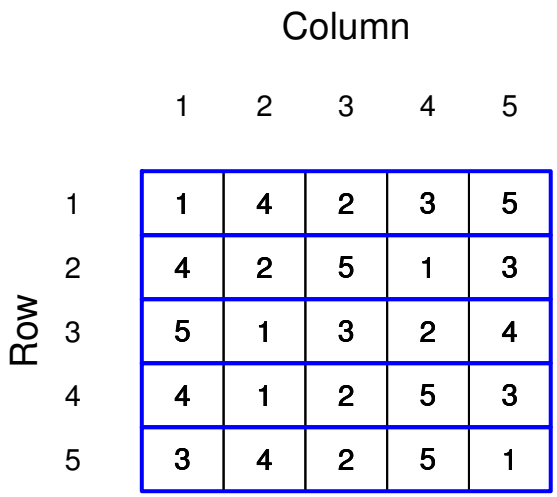
In this example, the allocated factor is Treatments, with 5 levels, and the recipient factors are Rows and Columns, both with 5 levels. Suppose that Rows are to form the blocks.

Use the following R code to obtain and display the layout:

```r
#'## Obtain the layout
Treatments <- factor(rep(1:5, times=5))
RCBD.lay <- designRandomize(allocated = Treatments,
                            recipient = list(Rows=5, Columns=5),
                            nested.recipients = list(Columns = "Rows"),
                            seed = 521814)
#'## Output the layout
RCBD.lay
```

```
##    Rows Columns Treatments
## 1     1       1          1
## 2     1       2          4
## 3     1       3          2
## 4     1       4          3
## 5     1       5          5
## 6     2       1          4
## 7     2       2          2
## 8     2       3          5
## 9     2       4          1
## 10    2       5          3
## 11    3       1          5
## 12    3       2          1
## 13    3       3          3
## 14    3       4          2
## 15    3       5          4
## 16    4       1          4
## 17    4       2          1
## 18    4       3          2
## 19    4       4          5
## 20    4       5          3
## 21    5       1          3
## 22    5       2          4
## 23    5       3          2
## 24    5       4          5
## 25    5       5          1
```

```r
#'## Plot the layout
Trt.mat <- matrix(as.character(RCBD.lay$Treatments), nrow=5, byrow=TRUE)
designPlot(Trt.mat, labels = levels(RCBD.lay$Treatments)[1:5],
           rtitle = "Row", ctitle = "Column",
           chardivisor = 6, rchardivisor = 6, cchardivisor = 6,
           blockdefinition = cbind(1,5), blocklinecolour = "blue")
```



The randomized layout is obtained by permuting (i) Rows and (ii) Columns within Rows. Then the permuted Rows and Columns and the systematic Treatments are sorted so that Rows and Columns are in standard order. The column `.Permutation` shows the permutation used. This and the `.Units` column are only included if `unit.permutation=TRUE`.

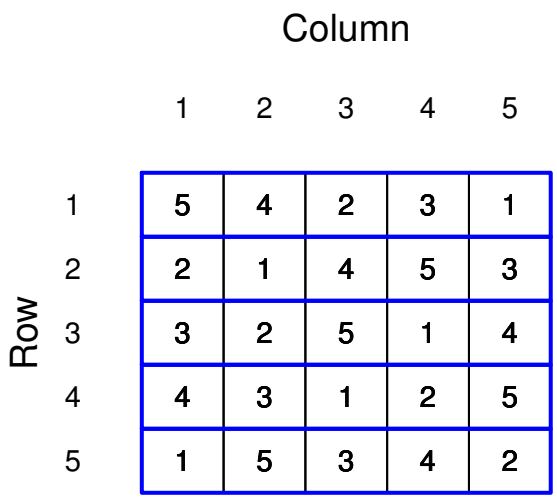
2.1.2 Produce the randomized layout for an LSD

Use `designRandomize` to randomize the treatments according to an LSD, having obtained the systematic design using `designLatinSqrSys`. In this case, for this design Rows and Columns are crossed, there are no nested factors. The layout can be obtained using the following R code:

```
#'## Obtain the layout
Treatments <- factor(designLatinSqrSys(5))
LSD.lay <- designRandomize(allocated = Treatments,
                           recipient = list(Rows=5, Columns=5),
                           seed = 154381)
#'## Output the layout
LSD.lay

##    Rows Columns Treatments
## 1     1       1          5
## 2     1       2          4
## 3     1       3          2
## 4     1       4          3
## 5     1       5          1
## 6     2       1          2
## 7     2       2          1
## 8     2       3          4
## 9     2       4          5
## 10    2       5          3
```

```
## 11    3       1          3
## 12    3       2          2
## 13    3       3          5
## 14    3       4          1
## 15    3       5          4
## 16    4       1          4
## 17    4       2          3
## 18    4       3          1
## 19    4       4          2
## 20    4       5          5
## 21    5       1          1
## 22    5       2          5
## 23    5       3          3
## 24    5       4          4
## 25    5       5          2
```

```
#'## Plot the layout
Trt.mat <- matrix(as.character(LSD.lay$Treatments), nrow=5, byrow=TRUE)
designPlot(Trt.mat, labels = levels(LSD.lay$Treatments)[1:5],
           rtitle = "Row", ctitle = "Column",
           chardivisor = 6, rchardivisor = 6, cchardivisor = 6,
           blockdefinition = cbind(1,5), blocklinecolour = "blue")
```



2.1.3 Check the properties of the designs

The properties of the designs can be investigated using `designAnatomy`.

Because these experiments involve a single randomization, they are two-tiered. That is, there are just two sets of factors involved in the randomization. As we have seen, the first set of factors is the set of allocated (treatment) factors and the second set is the set of recipient (unit) factors. Further there will be a set of projectors

associated with each tier and `designAnatomy` is used to do an eigenanalysis of the relationships between the two sets of projectors. The sets of projectors are specified to `designAnatomy` via model `formulae`, the formula for the recipient factors coming first in the `list` of `formulae`.

For both the RCBD and LSD the two sets of factors are (i) {Rows, Columns} and (ii) {Treatments}. What differs between the two designs is the nesting/crossing relationship between Rows and Columns and this will be expressed in the `formulae`.

Use the commands given below to produce the anatomy for the RCBD and LSD that have been obtained. Note that the 'Mean' source has been omitted from these tables. It can be included using the `grandMean` argument for `designAnatomy`.

```
#'## Anatomy for the RCBD
RCBD.canon <- designAnatomy(formulae = list(unit = ~Rows/Columns,
                                            trt = ~Treatments),
                            data = RCBD.lay)
summary(RCBD.canon)

##
##
## Summary table of the decomposition for unit & trt
##
##  Source.unit   df1 Source.trt df2 aefficiency eefficiency order
##  Rows            4
##  Columns[Rows]  20 Treatments   4      1.0000      1.0000     1
##                    Residual    16

#'## Anatomy for the LSD
LSD.canon <- designAnatomy(formulae = list(unit = ~Rows*Columns,
                                           trt = ~Treatments),
                           data = LSD.lay)
summary(LSD.canon)

##
##
## Summary table of the decomposition for unit & trt
##
##  Source.unit  df1 Source.trt df2 aefficiency eefficiency order
##  Rows           4
##  Columns        4
##  Rows#Columns  16 Treatments   4      1.0000      1.0000     1
##                   Residual    12
```

Get the mixed-model terms for the analysis by reruning the summary function with the `label.swap` argument set to `TRUE`.

```
#'## Term-based anatomy for the RCBD
summary(RCBD.canon, label.swap = TRUE)

##
##
## Summary table of the decomposition for unit & trt
##
##  Source.unit   df1 Source.trt df2 aefficiency eefficiency order
##  Rows            4
##  Columns[Rows]  20 Treatments   4      1.0000      1.0000     1
##                    Residual    16
```

```
#'## Term-based anatomy for the LSD
summary(LSD.canon, label.swap = TRUE)

##
##
## Summary table of the decomposition for unit & trt
##
##  Source.unit  df1 Source.trt df2 aefficiency eefficiency order
##  Rows           4
##  Columns        4
##  Rows#Columns  16 Treatments   4      1.0000      1.0000     1
##                   Residual    12
```

2.1.4 Questions

1. With what unit term is Treatments confounded in these designs and what is the difference in the interpretation of this term between the two different designs?

   Treatments is confounded with the term Rows:Columns. For the RCBD, this term coresponds to the source Columns[Rows], or Rows within Columns. For the LSD, this term corresponds to the source Rows#Columns, or the interaction of Rows-and-Columns.

2. What would determine which of these two designs is used for a particular experiment?

   In a discussion with the researcher, it needs to be determined whether overall Column differences can be ruled out. If they can then the RCBD should be used; otherwise the LSD would be used.

2.2 Split-plot from Yates (1937)

Yates (1937) describes a split-plot experiment that investigates the effects of three varieties of oats and four levels of Nitrogen fertilizer. The varieties are assigned to the main plots using a randomized complete block design with 6 blocks and the nitrogen levels are randomly assigned to the subplots in each main plot. The factor-allocation diagram for the experiment is in Figure 3.

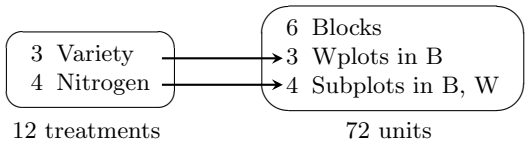


Figure 3: Factor-allocation diagram for an split-plot design: treatments are allocated to units; the arrows indicates that the factors Variety and Nitrogen are randomized to Wplots and Subplots, respectively; Wplots in B indicates that the Wplots are considered to be nested within Blocks for this randomization; Sunplots in B, W indicates that the Subplots are considered to be nested within Blocks and Wplots for this randomization; B = Blocks, W = Wplots

2.2.1 Produce the randomized experimental layout

Use `designRandomize` (and `fac.gen`) to obtain a randomized layout for this experiment and then check its properties, as illustrated in the following R code:

```
#'## Obtain the layout
spld.recip <- fac.gen(list(Blocks=6, Wplots=3, Subplots=4))
spld.nest <- list(Wplots = "Blocks",
                  Subplots = c("Wplots", "Blocks"))
spld.trt <- fac.gen(list(Variety=c("Victory","Golden Rain","Marvellous"),
                         Nitrogen=c(0,0.2,0.4,0.6)), times=6)
```

```
spld.lay <- designRandomize(allocated = spld.trt,
                            recipient = spld.recip,
                            nested.recipients = spld.nest,
                            seed = 235805)
#'## Display design produced
spld.lay
```

```
##    Blocks Wplots Subplots     Variety Nitrogen
## 1       1      1        1  Marvellous      0.4
## 2       1      1        2  Marvellous        0
## 3       1      1        3  Marvellous      0.2
## 4       1      1        4  Marvellous      0.6
## 5       1      2        1     Victory        0
## 6       1      2        2     Victory      0.2
## 7       1      2        3     Victory      0.6
## 8       1      2        4     Victory      0.4
## 9       1      3        1 Golden Rain      0.2
## 10      1      3        2 Golden Rain      0.4
## 11      1      3        3 Golden Rain      0.6
## 12      1      3        4 Golden Rain        0
## 13      2      1        1  Marvellous      0.4
## 14      2      1        2  Marvellous      0.2
## 15      2      1        3  Marvellous        0
## 16      2      1        4  Marvellous      0.6
## 17      2      2        1     Victory      0.2
## 18      2      2        2     Victory        0
## 19      2      2        3     Victory      0.6
## 20      2      2        4     Victory      0.4
## 21      2      3        1 Golden Rain      0.6
## 22      2      3        2 Golden Rain      0.4
## 23      2      3        3 Golden Rain      0.2
## 24      2      3        4 Golden Rain        0
## 25      3      1        1 Golden Rain      0.2
## 26      3      1        2 Golden Rain      0.6
## 27      3      1        3 Golden Rain      0.4
## 28      3      1        4 Golden Rain        0
## 29      3      2        1  Marvellous      0.4
## 30      3      2        2  Marvellous      0.6
## 31      3      2        3  Marvellous        0
## 32      3      2        4  Marvellous      0.2
## 33      3      3        1     Victory      0.4
## 34      3      3        2     Victory      0.2
## 35      3      3        3     Victory        0
## 36      3      3        4     Victory      0.6
## 37      4      1        1  Marvellous        0
## 38      4      1        2  Marvellous      0.4
## 39      4      1        3  Marvellous      0.2
## 40      4      1        4  Marvellous      0.6
## 41      4      2        1 Golden Rain      0.2
## 42      4      2        2 Golden Rain      0.6
## 43      4      2        3 Golden Rain      0.4
## 44      4      2        4 Golden Rain        0
## 45      4      3        1     Victory      0.4
## 46      4      3        2     Victory        0
```

```
## 47      4      3        3     Victory      0.6
## 48      4      3        4     Victory      0.2
## 49      5      1        1 Golden Rain      0.2
## 50      5      1        2 Golden Rain        0
## 51      5      1        3 Golden Rain      0.6
## 52      5      1        4 Golden Rain      0.4
## 53      5      2        1  Marvellous        0
## 54      5      2        2  Marvellous      0.2
## 55      5      2        3  Marvellous      0.6
## 56      5      2        4  Marvellous      0.4
## 57      5      3        1     Victory      0.4
## 58      5      3        2     Victory      0.2
## 59      5      3        3     Victory      0.6
## 60      5      3        4     Victory        0
## 61      6      1        1  Marvellous        0
## 62      6      1        2  Marvellous      0.6
## 63      6      1        3  Marvellous      0.4
## 64      6      1        4  Marvellous      0.2
## 65      6      2        1     Victory      0.4
## 66      6      2        2     Victory      0.2
## 67      6      2        3     Victory        0
## 68      6      2        4     Victory      0.6
## 69      6      3        1 Golden Rain      0.6
## 70      6      3        2 Golden Rain      0.2
## 71      6      3        3 Golden Rain      0.4
## 72      6      3        4 Golden Rain        0

#'## Check its properties
spld.canon <- designAnatomy(formulae = list(unit = ~Blocks/Wplots/Subplots,
                                            trt = ~Variety*Nitrogen),
                            data = spld.lay)
summary(spld.canon, which.criteria = c("aeff", "order"))

##
##
## Summary table of the decomposition for unit & trt
##
##  Source.unit             df1 Source.trt       df2 aefficiency order
##  Blocks                    5
##  Wplots[Blocks]           12 Variety            2      1.0000     1
##                              Residual          10
##  Subplots[Blocks:Wplots]  54 Nitrogen           3      1.0000     1
##                              Variety#Nitrogen   6      1.0000     1
##                              Residual          45
```

The function `fac.gen` is from the package `dae` and generates the factors in the `list` in standard order with the specified numbers of levels or the levels in supplied character or numeric vectors. Its use here is unnecessary, but it does illustrate that, besides a `list`, a `data.frame` can be supplied to the `recipient` argument of `designRandomize`. The `seed` is specified to ensure that the same design is produced whenever `designRandomize` is run with these arguments.

2.2.2 Analysis of variance (anova) for the Yields

After reading in the data, use the `aov` function to produce the anova as shown below. Note the use of the `Error` function to produce two Residual lines, one each for Wplots and Subplots.

```
#'## Read in data for actual experiment
data("Oats.dat")

#'## Analyse by anova
oats.aov <- aov(Yield ~ Nitrogen*Variety +
                  Error(Blocks/Wplots/Subplots), data=Oats.dat)
summary(oats.aov)

##
## Error: Blocks
##           Df Sum Sq Mean Sq F value Pr(>F)
## Residuals  5  15875    3175
##
## Error: Blocks:Wplots
##           Df Sum Sq Mean Sq F value Pr(>F)
## Variety    2   1786   893.2   1.485  0.272
## Residuals 10   6013   601.3
##
## Error: Blocks:Wplots:Subplots
##                  Df Sum Sq Mean Sq F value   Pr(>F)
## Nitrogen          3  20020    6673  37.686 2.46e-12 ***
## Nitrogen:Variety  6    322      54   0.303    0.932
## Residuals        45   7969     177
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The anova table shown here is the same as the anatomy, but in a different format.

2.2.3 Questions

1. In what sense does this design involve a single randomization?

   In the sense that the randomization of both Nitrogen and Variety can be achieved with a single permutation of the units, the subplots.

2. A factorial RCBD would involve randomizing the $3 \times 4 = 12$ treatments to the 12 subplots within each block. What has been achieved in using the split-plot design as compared to a factorial RCBD?

   The precision of the Variety differences has been sacrified to increase the precision of the Nitrogen differences. This is the case because the Residual mean square for Blocks:Wplots is substantially larger than that for Blocks:Wplots:Subplots. If a factorial RCBD had been used, the Residual mean square for Blocks:Plots would be the weighted average of the two Residual mean squares from the split-plot experiment, the weight being the Residual degrees of freedom. That is, the value of the Residual mean square for the factorial RCBD would be between the values for the two Residual mean squares for the splot-plot design.

2.3 An environmental experiment

Suppose an environmental scientist wants to investigate the effect on the biomass of burning areas of natural vegetation. There are available two areas separated by several kilometres for use in the investigation. It is only possible to either burn or not burn an entire area. The area to be burnt is randomly selected and the other area is to be left unburnt as a control. Further, 30 locations in each area are to be randomly sampled and the biomass measured at each location. The factor-allocation diagram for the experiment is in Figure 4.

Obtain the randomized layout for this experiment and check its properties.

```
#'## Obtain the layout
Burn <- factor(rep(1:2, each=30))
```

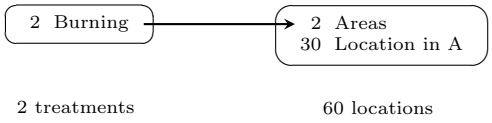


Figure 4: Factor-allocation diagram for the environmental experiment: treatments are allocated to locatiions; the arrow indicates that the factor Burning is randomized to Areas; Locations in A indicates that the LOcations are considered to be nested within Areas; A = Areas.

```
Burn.lay <- designRandomize(allocated = Burn,
                            recipient = list(Areas=2, Samples=30),
                            nested.recipients = list(Samples = "Areas"),
                            seed = 872159)
#'## Check its properties
Burn.canon <- designAnatomy(formulae = list(unit = ~Areas/Samples,
                                            trt = ~Burn),
                            data = Burn.lay)
summary(Burn.canon)

##
##
## Summary table of the decomposition for unit & trt
##
##  Source.unit    df1 Source.trt df2 aefficiency eefficiency order
##  Areas            1 Burn         1      1.0000      1.0000     1
##  Samples[Areas]  58
```

2.3.1 Questions

1. How is the pseudo-replication involved in this experiment manifested in the anatomy?

   Because (i) Areas and Burn are alongside each other in the anova table, (ii) they both have 1 degree of freedom, and (iii) the single canonical eficiency factor is one, then Areas and Burn are completely confounded. That is, the pseudoreplication has resulted in differences between Areas and between Burns being inextricably mixed up.

2. The randomization-based mixed model for the experiment is $\mathbb{E}[Y] = \text{Burn}$ and $\text{Var}[Y] = \text{Areas} + \text{Areas}:\text{Samples}$. What difficulties do you anticipate in attempting to fit this model? How could the model be modified so that a fit can be obtained? Brien and Demétrio (2009) call models formed by removing terms to enable a fit to be achieved 'models of convenience'. What dangers do you foresee in basing conclusions on the fitted model?

   There will be a singularity in the model because Areas is confounded with Burn. A fit could be obtained by removing Areas from the random model. The problem is that a test of Burn would then be based on the ratio of variability in Burn differences to an estimate of the variance of Samples-within-Areas variability. This does not include Areas variability and so the denominator is likely to be underestimated; p-values based from this test are likely to be too small and significant differences are more likely to be declared where there are none as compared to when an estimate of Areas variability is included in the denominator of the F-statistic.

3 Single-allocation, nonorthogonal design in R

This class of experiments covers the nonorthogonal standard or textbook experiments.

3.1 Twenty treatments in an alpha design

The following table gives an alpha design for 20 treatments, taken from Williams et al. (2002, p.128). The design has 3 replicates, each of which contains 5 blocks of 4 plots. It is a resolved design in that each replicate contains a complete set of the treatments.

Table 1: Unrandomized alpha design for 20 treatments

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

The factor-allocation diagram for the experiment is in Figure 5.

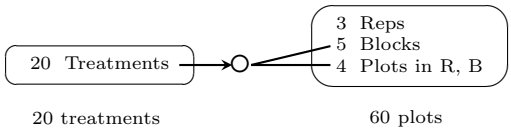


Figure 5: Factor-allocation diagram for the alpha design: treatments are allocated to units; the arrow indicates that the allocation is randomized; the 'O' at the end of the arrow indicates that a nonorthogonal design is used; the two lines from 'O' indicates that the Treaments are allocated to the combinations of Blocks and Plots using the design; Blocks in R indicates that the Blocks are considered to be nested within Reps for this randomization; Plots in R, B indicates that the Plots are considered to be nested within Reps and Blocks for this randomization; R = Reps; B = Blocks.

3.1.1 Produce the randomized layout for the alpha design and check its properties

Use `designRandomize` to obtain the randomized layout and `designAnatomy` to check its properties.

```
#'## Obtain the layout
Treats <- factor(c(1:20,
                   1:5, 7:10,6, 13:15,11,12, 19,20,16:18,
                   1:5, 8:10,6,7, 15,11:14, 17:20,16))
# Note that Treatments has been entered by rows within a replicate
alpha.lay <- designRandomize(allocated = Treats,
                             recipient = list(Reps=3, Plots=4, Blocks=5),
                             nested.recipients = list(Blocks = "Reps",
                                                      Plots = c("Reps", "Blocks")),
                             seed = 918508)
alpha.lay <- with(alpha.lay, alpha.lay[order(Reps,Blocks,Plots), ])

#'## Check its properties
alpha.canon <- designAnatomy(formulae = list(units = ~Reps/Blocks/Plots,
                                             trts = ~Treats),
                             which.criteria = "all",
```

```
                            data = alpha.lay)
summary(alpha.canon, which.criteria = "all")

##
##
## Summary table of the decomposition for units & trts (based on adjusted quantities)
##
##  Source.units       df1 Source.trts df2 aefficiency eefficiency mefficiency sefficiency xefficiency
##  Reps                 2
##  Blocks[Reps]        12 Treats       12      0.2778      0.1667      0.3333      0.0152      0.4167
##  Plots[Reps:Blocks]  45 Treats       19      0.7447      0.5833      0.7895      0.0365      1.0000
##                         Residual     26
##  order dforthog
##
##      2        0
##      3        7
##
##
## The design is not orthogonal
```

The summary table shows us a number of summary statistics calculated from the canonical efficiency factors. They are:

aefficiency: the harmonic mean of the nonzero canonical efficiency factors.

mefficiency: the mean of the nonzero canonical efficiency factors.

eefficiency: the minimum of the nonzero canonical efficiency factors.

sefficiency: the variance of the nonzero canonical efficiency factors.

xefficiency: the maximum of the nonzero canonical efficiency factors.

order: the order of balance and is the number of unique nonzero canonical efficiency factors.

dforthog: the number of canonical efficiency factors that are equal to one.

For this example it can be seen that (i) an average 74.47%, as measured by the harmonic mean, or 78.95%, as measured by the arithmetic mean, of the information about Treats is confounded with the differences between plots within the reps-blocks combinations and (ii) there are 3 different efficiency factors associated with the 19 Treats degrees of freedom estimated from Reps:Blocks:Plots, the smallest of which is 0.5833 and 7 of which are one. In this case, where the treatments are equally replicated, it can be concluded that the mean variance of a normalized treatment contrast is inversely proportional to the harmonic mean of the canonical efficiency factors, that is, to 0.7447.

Get the mixed-model terms for the analysis by reruning the summary function with the `label.swap` argument set to `TRUE`.

```
#'## Obtain the terms for the design
summary(alpha.canon, which.criteria = "all", label.swap = TRUE)

##
##
## Summary table of the decomposition for units & trts (based on adjusted quantities)
##
##  Source.units       df1 Source.trts df2 aefficiency eefficiency mefficiency sefficiency xefficiency
##  Reps                 2
##  Blocks[Reps]        12 Treats       12      0.2778      0.1667      0.3333      0.0152      0.4167
```

```
##  Plots[Reps:Blocks]  45 Treats       19      0.7447      0.5833      0.7895      0.0365      1.0000
##                         Residual     26
##  order dforthog
##
##      2        0
##      3        7
##
##
## The design is not orthogonal
```

3.1.2 Questions

1. What is the randomization-based mixed model for this experiment?

   The trts term (Source.trts) provides the fixed term and the units terms (Source.units) provide the random terms. That is, Treats is assumed fixed and Reps, Blocks and Plots are assumed random. Hence, the symbolic mixed model is $\mathbb{E}[Y] = \text{Treats}$ *and* $\text{Var}[Y] = \text{Reps} + \text{Reps:Blocks} + \text{Reps:Blocks:Plots}$.

2. In a mixed-model analysis, which unit terms might you fit as fixed terms? Why?

   Reps is a definite candidate for the following reasons. Firstly, Reps has only two degrees of freedom and it will be difficult to estimate a variance component for it. Secondly, one does not want to estimate Treats from Reps (there is no Treat information between Reps).

3.2 Balanced incomplete block design from Joshi (1987)

Joshi (1987) gives an experiment to investigate six varieties of wheat that employs a balanced incomplete block design with 10 blocks, each consisting of three plots. The factor-allocation diagram for the experiment is in Figure 6.

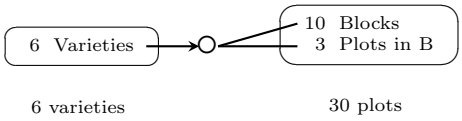


Figure 6: Factor-allocation diagram for the balanced incomplete block design: treatments are allocated to units; the arrow indicates that the allocation is randomized; the '○' at the end of the arrow indicates that a nonorthogonal design is used; the two lines from '○' indicates that the Varieties are allocated to the combinations of Blocks and Plots using the design; Plots in B indicates that the Plots are considered to be nested within Blocks for this randomization; B = Blocks.

3.2.1 Input the Yields and check properties of the design

Use the following R code to input the data for the experiment and check its properties.

```
#'## Input the design and data
data("BIBDWheat.dat")

#'## Check the properties of the design
bibdwheat.canon <- designAnatomy(formulae = list(units = ~Blocks/Plots,
                                                 trts = ~Varieties),
                                 data = BIBDWheat.dat)
summary(bibdwheat.canon)

##
##
```

```
## Summary table of the decomposition for units & trts (based on adjusted quantities)
##
##  Source.units  df1 Source.trts df2 aefficiency eefficiency order
##  Blocks          9 Varieties     5      0.2000      0.2000     1
##                    Residual      4
##  Plots[Blocks]  20 Varieties     5      0.8000      0.8000     1
##                    Residual     15
##
## The design is not orthogonal
```

From this it is clear that 80% of the information about Varieties is available from the Block:Plots source; that is, 80% of the Varieties information is confounded with differences between plots within blocks. Of course, the remaining 20% is confounded with Blocks.

3.2.2 Anova for the Yields

```
#'## Perfom an anova
summary(aov(Yield ~ Varieties + Error(Blocks/Plots), data = BIBDWheat.dat))

##
## Error: Blocks
##           Df Sum Sq Mean Sq F value Pr(>F)
## Varieties  5  196.6   39.32   0.582  0.718
## Residuals  4  270.4   67.59
##
## Error: Blocks:Plots
##           Df Sum Sq Mean Sq F value Pr(>F)
## Varieties  5 1156.4  231.29   4.021 0.0163 *
## Residuals 15  862.9   57.53
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

3.3 A design with rows and columns from Williams (2002)

Williams et al. (2002, p.144) provides an example of a resolved, Latinized, row-column design with four rectangles (blocks) each of six rows by ten columns. The experiment investigated differences between 60 provenances of a species of Casuarina tree, these provenances coming from 18 countries; the trees were inoculated prior to planting at two different times, time of inoculation being assigned to the four replicates so that each occurred in two replicates. At 30 months, diameter at breast height (Dbh) was measured.

The factor-allocation diagram for the experiment is in Figure 7.

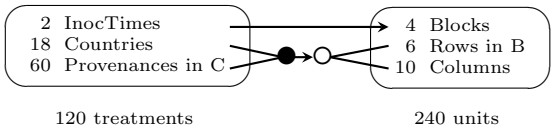


Figure 7: Factor-allocation diagram for the row-and-column design: treatments are allocated to units; the arrows indicates that the allocations are randomized; the 'O' at the end of the lower arrow indicates that a nonorthogonal design is used; the two lines from 'O' indicates that the Countries and Provenances are allocated to the combinations of Rows and Columns using the design; Rows in B indicates that the Rows are considered to be nested within Blocks for this randomization; B = Blocks.

3.3.1 Input the design and check the properties of the design

Use the following R code to input the design and check its properties.

```
#'## Input the design
data("Casuarina.dat")
#'## Check the properties of the design
Casuarina.canon <- designAnatomy(formulae = list(units = ~(Reps/Rows)*Columns,
                                                 trts = ~InocTime*(Countries+Provenances)),
                                 data = Casuarina.dat)

## Warning in projs.2canon(CombinedSets$Q[[ntiers]], struct[[ktier]]$Q): Provenances[Countries]
and Countries are partially aliased in Rows[Reps]
## Warning in projs.2canon(CombinedSets$Q[[ntiers]], struct[[ktier]]$Q): Provenances[Countries]
and Countries are partially aliased in Reps#Columns
## Warning in projs.2canon(CombinedSets$Q[[ntiers]], struct[[ktier]]$Q): Provenances[Countries]
and Countries are partially aliased in Rows#Columns[Reps]
## Warning in projs.2canon(CombinedSets$Q[[ntiers]], struct[[ktier]]$Q): InocTime#Countries and
Countries are partially aliased in Rows#Columns[Reps]
## Warning in projs.2canon(CombinedSets$Q[[ntiers]], struct[[ktier]]$Q): InocTime#Countries and
Provenances[Countries] are partially aliased in Rows#Columns[Reps]
## Warning in projs.2canon(CombinedSets$Q[[ntiers]], struct[[ktier]]$Q): InocTime#Provenances[Countries]
and Countries are partially aliased in Rows#Columns[Reps]
## Warning in projs.2canon(CombinedSets$Q[[ntiers]], struct[[ktier]]$Q): InocTime#Provenances[Countries]
and Provenances[Countries] are partially aliased in Rows#Columns[Reps]
## Warning in projs.2canon(CombinedSets$Q[[ntiers]], struct[[ktier]]$Q): InocTime#Provenances[Countries]
and InocTime#Countries are partially aliased in Rows#Columns[Reps]

summary(Casuarina.canon, which = c("aeff", "eeff", "order", "dforth"))

##
##
## Summary table of the decomposition for units & trts (based on adjusted quantities)
##
##  Source.units       df1 Source.trts                     df2 aefficiency eefficiency order dforthog
##  Reps                 3 InocTime                          1      1.0000      1.0000     1        1
##                         Residual                          2
##  Rows[Reps]          20 Countries                        17      0.0145      0.0018    17        0
##                         Provenances[Countries]           3      0.1622      0.1326     3        0
##  Columns              9 Countries                        9      0.0137      0.0028     9        0
##  Reps#Columns        27 Countries                       17      0.0134      0.0012    17        0
##                         Provenances[Countries]          10      0.2320      0.1596    10        0
##  Rows#Columns[Reps] 180 Countries                       17      0.7611      0.5588    17        0
##                         Provenances[Countries]          42      0.6851      0.3429    42        0
##                         InocTime#Countries              17      0.6808      0.4735    17        0
##                         InocTime#Provenances[Countries] 42      0.5516      0.2009    42        0
##                         Residual                        62
##
## Table of (partial) aliasing between sources derived from the same formula
##
##  Source                          df Alias                 In                 aefficiency
##  Provenances[Countries]           3 Countries             Rows[Reps]              0.1622
##  Provenances[Countries]          10 Countries             Reps#Columns            0.2320
##  Provenances[Countries]          42 Countries             Rows#Columns[Reps]      0.6851
##  InocTime#Countries              17 Countries             Rows#Columns[Reps]      0.6808
```

```
##  InocTime#Countries              17 Provenances[Countries] Rows#Columns[Reps]      0.6808
##  InocTime#Provenances[Countries] 42 Countries              Rows#Columns[Reps]      0.5516
##  InocTime#Provenances[Countries] 42 Provenances[Countries] Rows#Columns[Reps]      0.5516
##  InocTime#Provenances[Countries] 42 InocTime#Countries     Rows#Columns[Reps]      0.5516
##  eefficiency order dforthog
##       0.1326     3        0
##       0.1596    10        0
##       0.3429    42        0
##       0.4735    17        0
##       0.4735    17        0
##       0.2009    42        0
##       0.2009    42        0
##       0.2009    42        0
##
## The design is not orthogonal
```

Firstly, note that `designAnatomy` has automatically detected that Provenances is nested within Countries, even though Provenances has 60 unique levels: the sources for these two terms are Countries and Provenances[Countries] and these have 17 and 42 degrees of freedom when estimated from Rows # Columns [Reps], respectively. The total of these degrees of freedom is 59, one less than the number of Provenances, as expected.

Secondly, the partial aliasing evident in this design reflects a lack of (structure) balance between the treatment sources within each units source. This is an undesirable, but unavoidable, feature of the design for this experiment.

4 Multiphase experiments in R

This class of experiments differs from those previously presented in that they often employ two or more randomizations or allocations, each to a different type of unit. As a result, there will be three or more sets of factors, or tiers, to deal with; further, when there are three sets of factors, three formula will need to be supplied to `designAnatomy`.

4.1 Athletic examples based on Brien et al. (2011)

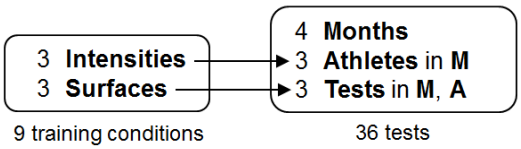
Brien et al. (2011) give several designs for an athletic experiment that illustrate the basic principles to be employed in designing multiphase experiments. Here designs for three different scenarios are considered.

4.1.1 A standard single-phase athlete training experiment

First, a split-plot experiment in which the performance of an athlete when subject to nine different training conditions is investigated. The nine training conditions are the combinations of three surfaces and three intensities of training. Also, assume that the prime interest is in surface differences, with intensities included to observe the surfaces over a range of intensities. The experiment is to involve 12 athletes, three per month for four consecutive months; each athlete undergoes three tests. The heart rate of the athlete is to be taken immediately upon completion of a test.

A split-plot design is to be employed for the experiment: the three intensities are randomized to the three athletes in each month and the three surfaces are randomized to the three tests that each athlete is to undergo. The randomization diagram is shown in Figure 8. Generate a randomized layout for the experiment.

Figure 8: Randomization diagram for the single-phase athlete training experiment




```
#'## Generate a layout for a standard athlete training experiment
eg1.lay <- designRandomize(allocated = fac.gen(list(Intensities = 3, Surfaces = 3),
                                               times = 4),
                           recipient = list(Months = 4, Athletes = 3, Tests = 3),
                           nested.recipients = list(Athletes = "Months",
                                                    Tests = c("Months", "Athletes")),
                           seed = 2598)
eg1.lay
```

```
##    Months Athletes Tests Intensities Surfaces
## 1       1        1     1           2        3
## 2       1        1     2           2        2
## 3       1        1     3           2        1
## 4       1        2     1           3        2
## 5       1        2     2           3        1
## 6       1        2     3           3        3
## 7       1        3     1           1        1
## 8       1        3     2           1        2
## 9       1        3     3           1        3
## 10      2        1     1           2        1
## 11      2        1     2           2        2
```

```
## 12      2        1     3           2        3
## 13      2        2     1           1        3
## 14      2        2     2           1        2
## 15      2        2     3           1        1
## 16      2        3     1           3        1
## 17      2        3     2           3        3
## 18      2        3     3           3        2
## 19      3        1     1           2        1
## 20      3        1     2           2        3
## 21      3        1     3           2        2
## 22      3        2     1           3        2
## 23      3        2     2           3        3
## 24      3        2     3           3        1
## 25      3        3     1           1        2
## 26      3        3     2           1        3
## 27      3        3     3           1        1
## 28      4        1     1           1        3
## 29      4        1     2           1        2
## 30      4        1     3           1        1
## 31      4        2     1           2        1
## 32      4        2     2           2        2
## 33      4        2     3           2        3
## 34      4        3     1           3        1
## 35      4        3     2           3        3
## 36      4        3     3           3        2
```

```
#'## Check properties of the design
eg1.canon <- designAnatomy(formulae = list(tests = ~Months/Athletes/Tests,
                                           cond = ~Intensities*Surfaces),
                           data = eg1.lay)
summary(eg1.canon, which.criteria="none")
```

```
##
##
## Summary table of the decomposition for tests & cond
##
##  Source.tests           df1 Source.cond          df2
##  Months                   3
##  Athletes[Months]         8 Intensities            2
##                             Residual               6
##  Tests[Months:Athletes]  24 Surfaces               2
##                             Intensities#Surfaces   4
##                             Residual              18
```

Question

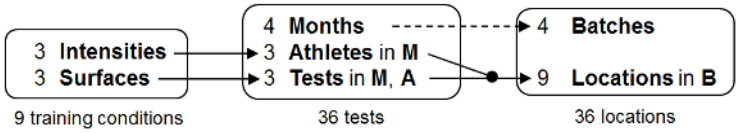
1. Why was a split-plot design chosen for this experiment?

   Because it is likely that differences between tests within an athlete will be smaller than differences between athletes within a month. Hence, becasue the prime interest is in Surfaces, they are assigned to tests within an athlete and will have better precision than Intensities, which have been assigned to the more variable athletes within a month.

4.1.2 A simple two-phase athlete training experiment

Suppose that, in addition to heart rate taken immediately upon completion of a test, the free haemoglobin is to be measured using blood specimens taken from the athletes after each test and transported to the laboratory for analysis. That is, a second laboratory phase is required to obtain the new response. In this phase, because the specimens become available monthly, the batch of specimens for one month are to be processed, in a random order, before those for the next month are available. The allocation diagram for this experiment is in Figure 9, the dashed line indicating that Months are systematically allocated to Batches. The randomizations in this diagram are composed (Brien and Bailey, 2006) and is one of the two types of randomizations in a chain (Bailey and Brien, 2015). This means that the second-phase randomization only need to consider how the tests factors are to be assigned to locations; training conditions can be ignored in determining the second-phase design.

Figure 9: Allocation diagram for the two-phase athlete training experiment



Using the following R code, obtain a layout for the second phase and check the properties of the layout. In doing this, the first-phase layout is randomized. However, because Months is not randomized to Batches, the argument `except` in `designRandomize` is used to effect the systematic allocation.

```
#'## Generate a layout for a simple two-phase athlete training experiment
#'## Phase 1 - the split-plot design that has already been generated.
#'## Phase 2 - randomize tests (and training conditions) to locations,
#'##           but Months assigned systematicaly to Batches
#'##           so except Batches from the randomization
eg2.lay <- designRandomize(allocated = eg1.lay,
                           recipient = list(Batches = 4, Locations = 9),
                           nested.recipients = list(Locations = "Batches"),
                           except = "Batches",
                           seed = 71230)
eg2.lay
```

```
##    Batches Locations Months Athletes Tests Intensities Surfaces
## 1        1         1      1        2     3           3        3
## 2        1         2      1        1     2           2        2
## 3        1         3      1        2     2           3        1
## 4        1         4      1        3     1           1        1
## 5        1         5      1        3     2           1        2
## 6        1         6      1        1     1           2        3
## 7        1         7      1        2     1           3        2
## 8        1         8      1        1     3           2        1
## 9        1         9      1        3     3           1        3
## 10       2         1      2        3     1           3        1
## 11       2         2      2        2     2           1        2
## 12       2         3      2        1     3           2        3
## 13       2         4      2        1     2           2        2
## 14       2         5      2        3     2           3        3
## 15       2         6      2        2     1           1        3
## 16       2         7      2        2     3           1        1
## 17       2         8      2        3     3           3        2
## 18       2         9      2        1     1           2        1
```

```
## 19       3         1      3        1     1           2        1
## 20       3         2      3        3     1           1        2
## 21       3         3      3        2     3           3        1
## 22       3         4      3        2     2           3        3
## 23       3         5      3        2     1           3        2
## 24       3         6      3        3     3           1        1
## 25       3         7      3        3     2           1        3
## 26       3         8      3        1     2           2        3
## 27       3         9      3        1     3           2        2
## 28       4         1      4        2     3           2        3
## 29       4         2      4        2     1           2        1
## 30       4         3      4        1     1           1        3
## 31       4         4      4        1     2           1        2
## 32       4         5      4        1     3           1        1
## 33       4         6      4        3     1           3        1
## 34       4         7      4        2     2           2        2
## 35       4         8      4        3     2           3        3
## 36       4         9      4        3     3           3        2
```

```
#'## Check properties of the design
eg2.canon <- designAnatomy(formulae = list(locs = ~Batches/Locations,
                                           tests = ~Months/Athletes/Tests,
                                           cond = ~Intensities*Surfaces),
                           data = eg2.lay)
summary(eg2.canon, which.criteria="none")
```

```
##
##
## Summary table of the decomposition for locs, tests & cond
##
##  Source.locs        df1 Source.tests           df2 Source.cond          df3
##  Batches              3 Months                   3
##  Locations[Batches]  32 Athletes[Months]         8 Intensities            2
##                                                   Residual               6
##                         Tests[Months:Athletes]  24 Surfaces               2
##                                                   Intensities#Surfaces   4
##                                                   Residual              18
```

Questions

1. What would be the allocation-based mixed model for this experiment, an allocation-based mixed model having the same terms as the randomization-based mixed model that would apply if all the allocations had been made by randomizing. Do you anticipate any problem in fitting it?

   The allocation-based mixed model is formed by treating all training-conditions factors as fixed and the remaining factors as random. Hence, the symbolic mixed model is $\mathbb{E}[Y] = \text{Intensities}+\text{Surfaces}+\text{Intensities}:\text{Surfaces}$ *and* $\text{Var}[Y] = \text{Months}+\text{Months}:\text{Athletes}+\text{Months}:\text{Athletes}:\text{Tests}+\text{Batches}+\text{Batches}:\text{Locations}$. *The problem in fitting it would be that Months and Batches are confounded so that the variance model is singular.*

2. Compare the units for the two phases in this experiment?

   A unit in the first phase is a test conducted on an athlete in a particular month; in the second phase, a unit is a location of a test within a batch. That is, the unit in the first phase is an athlete's test and in the second phase is a blood specimen in a lab location.

3. What are the outcomes for the two phases for this experiment?

The outcome for the first phase is the heart rate for a test and a blood specimen from the test; the outcome for the second phase, is the free haemglobin measured at a location.

4.1.3 Allowing for lab processing order in the athletic training example

This experiment differs from the last in the second phase: the second phase now requires a row-column design. However, one cannot consider a design for just Months, Athletes and Tests and ignore Intensities and Surfaces, as was done in the previous design. Indeed prime consideration needs to be given to Intensities and Surfaces.

Rosemary Bailey has manually constructed a design to assign two three-level factors to 4 rows × 9 columns, such that the split-plot nature of the first phase design is retained. One begins by forming an incomplete block design of 9 blocks each of 4 plots for two three-level treatment factors. Then this is converted to a row-column design by using the algorithm associated with Hall's Marriage Theorem (Bailey, 2008, Section 11.10) to rearrange the treatments within a block so that there is a complete set of training conditions in each row. Finally, the columns have to be rearranged into three groups to have the same number of each Intensity within a group of 3 Locations and rearrange the columns within the groups so that the order of the Surfaces within a Batch is the same for all 3 groups of Locations. The systematic design is given in Table 2.

To complete the design for the experiment the Athletes and Tests in each Month have to be associated with the Intensities and Surfaces combinations. For example, the Athlete for the first Month that was assigned level 1 of Intensities is associated with the first three combinations in the first row of Table 2; then the levels of Tests for this Athlete is associated with the Surface levels according to which Surface was assigned to which Test in the first phase. This is repeated for all combinations in the table.

If one considers how the Athletes are to be assigned to the combinations in Table 2, it is clear that there will be a group of Athletes assigned to each group of three Locations. We could introduce a pseudofactor A_1 for Athletes that has level 1 for the 4 Athletes assigned to Locations 1–3, level 2 for those assigned to Locations 4–6 and level 3 for those assigned to Locations 7–9. This pseudofactor is confounded with Locations; that is, two of the Months:Athletes degrees of freedom are confounded with Locations. However, the randomization does not assign the levels of the pseudofactor to Locations groups and so it only needs to be included in the analysis if one wants to separately identify it as a source there. It has not been included in what follows.

Table 2: Row-column design for assigning 3 Intensities × 3 Surfaces to 4 Batches × 9 Locations in the second-phase of the two-phase athlete training experiment†

		Locations								
		1	2	3	4	5	6	7	8	9
Batches	1	1,2	1,3	1,1	2,2	2,3	2,1	3,2	3,3	3,1
	2	1,3	1,1	1,2	2,3	2,1	2,2	3,3	3,1	3,2
	3	2,1	2,2	2,3	3,1	3,2	3,3	1,1	1,2	1,3
	4	3,1	3,2	3,3	1,1	1,2	1,3	2,1	2,2	2,3

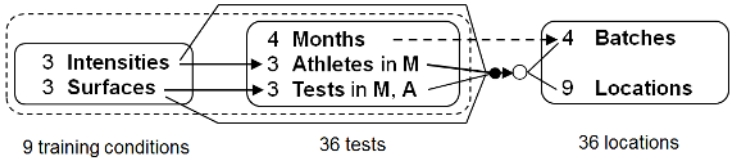
†The pair of numbers in a cell of the table is the Intensity level followed by the Surfaces level to be be applied to the Batch-Location combination for that cell.

The allocation diagram, using this design, is in Figure 10. In this diagram, the training conditions and tests panels are surrounded by a dashed rectangle and lines go from the training conditions sources to the lines from the test sources. This indicates that the result of the allocation in the first phase needs to be explicitly taken into account in the second-phase allocation. The randomization involved has been called a randomized-inclusive randomization (Brien and Bailey, 2006) and is one of the two types of randomizations in a chain (Bailey and Brien, 2015). Because Batches and Locations are crossed, the second phase randomization is achieved by independently permuting the Batches and Locations.

Use the following R code to obtain a layout for the new second phase design.

```
#'## Generate systematic design proposed by RAB for Intensities and Surfaces
#'## It is based on two 3x3 mutually orthogonal Latin squares
```

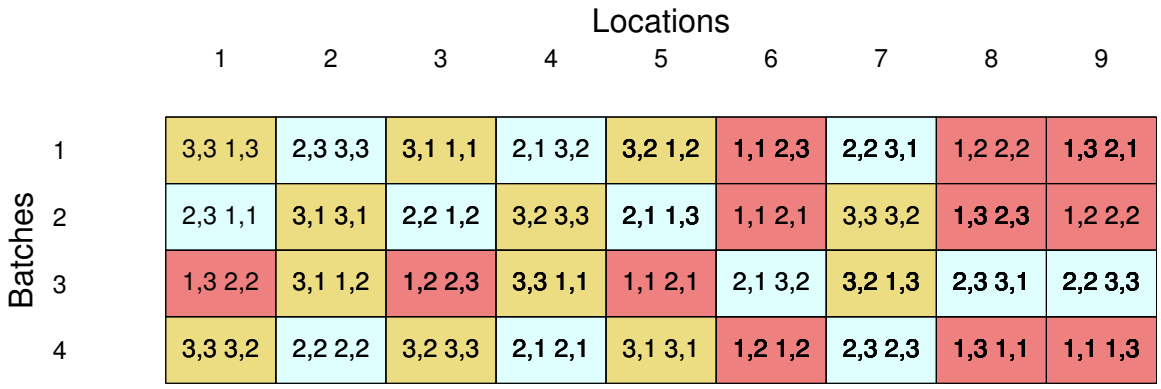
Figure 10: Allocation diagram for the two-phase athlete training experiment with a row-column design for the second phase




```
#'## that are extended by adding fourth rows
ELS3x3 <- c(1:3, 1:3, 2,3,1, 3,1,2)
eg3.sys <- cbind(fac.gen(list(Batches = 4, Locations = 9)),
                 data.frame(Intensities = factor(rep(ELS3x3, each=3)),
                            Surfaces = factor(c(rep(c(2,3,1), 3), rep(c(3,1,2), 3),
                                              rep(1:3, 6)))))


#'## Second phase design
#'## Generate a systematic layout for Months, Athletes, Tests, Intensities and Surfaces
eg3.sys$Months <- eg3.sys$Batches
eg3.sys <- merge(eg1.lay, eg3.sys) #merges on commmon factors Months, Intensities and Surfaces
eg3.sys  <- with(eg3.sys, eg3.sys[order(Batches,Locations),])
eg3.sys <- eg3.sys[c("Months", "Athletes", "Tests", "Intensities", "Surfaces")]

#'## Allocate the second phase
eg3.lay <- designRandomize(allocated   = eg3.sys,
                           recipient = list(Batches = 4, Locations = 9),
                           except = "Batches",
                           seed = 243526)
eg3.lay

##    Batches Locations Months Athletes Tests Intensities Surfaces
## 1        1         1      1        3     3           1        3
## 2        1         2      1        2     3           3        3
## 3        1         3      1        3     1           1        1
## 4        1         4      1        2     1           3        2
## 5        1         5      1        3     2           1        2
## 6        1         6      1        1     1           2        3
## 7        1         7      1        2     2           3        1
## 8        1         8      1        1     2           2        2
## 9        1         9      1        1     3           2        1
## 10       2         1      2        2     3           1        1
## 11       2         2      2        3     1           3        1
## 12       2         3      2        2     2           1        2
## 13       2         4      2        3     2           3        3
## 14       2         5      2        2     1           1        3
## 15       2         6      2        1     1           2        1
## 16       2         7      2        3     3           3        2
## 17       2         8      2        1     3           2        3
## 18       2         9      2        1     2           2        2
## 19       3         1      3        1     3           2        2
## 20       3         2      3        3     1           1        2
## 21       3         3      3        1     2           2        3
```

```
## 22       3         4      3         3     3           1         1
## 23       3         5      3         1     1           2         1
## 24       3         6      3         2     1           3         2
## 25       3         7      3         3     2           1         3
## 26       3         8      3         2     3           3         1
## 27       3         9      3         2     2           3         3
## 28       4         1      4         3     3           3         2
## 29       4         2      4         2     2           2         2
## 30       4         3      4         3     2           3         3
## 31       4         4      4         2     1           2         1
## 32       4         5      4         3     1           3         1
## 33       4         6      4         1     2           1         2
## 34       4         7      4         2     3           2         3
## 35       4         8      4         1     3           1         1
## 36       4         9      4         1     1           1         3
```

```
#'## Plot the layout - or see file that includes the output
A.mat <- matrix(as.character(eg3.lay$Athletes), nrow=4, byrow=TRUE)
comb.fac <- with(eg3.lay, fac.combine(list(AT = fac.combine(list(Athletes, Tests),
                                                            combine=TRUE, sep=","),
                                           IS = fac.combine(list(Intensities, Surfaces),
                                                            combine=TRUE, sep=",")),
                                      combine=TRUE, sep=" "))
comb.mat <- matrix(as.character(comb.fac), nrow=4, byrow=TRUE)
#Output the row and column labels
designPlot(A.mat, plotlabels = FALSE, rtitle = "Batches", ctitle = "Locations",
           chardivisor = 6, rchardivisor = 6, cchardivisor = 6)
#Colour according to athlete
cell.colours <- c("lightcoral","lightcyan","lightgoldenrod")
for (i in 1:3)
  designPlot(A.mat, labels=i, plotlabels = FALSE, cellfillcolour = cell.colours[i],
             plotcellboundary = TRUE, new=FALSE,
             chardivisor = 6, rchardivisor = 6, cchardivisor = 6)
#Add the Athletes, Tests, Intensities, Surfaces combinations
designPlot(comb.mat, new=FALSE, chardivisor = 6, rchardivisor = 4, cchardivisor = 4)
```



Check the properties of the design.

```
#'## Check properties of the design
eg3.canon <- designAnatomy(formulae = list(locs = ~Batches*Locations,
                                           tests = ~Months/Athletes/Tests,
                                           cond = ~Intensities*Surfaces),
                           data = eg3.lay)
summary(eg3.canon, which.criteria =c("aefficiency", "order"))

##
##
## Summary table of the decomposition for locs, tests & cond (based on adjusted quantities)
##
##  Source.locs       df1 Source.tests           df2 Source.cond          df3 aefficiency order
##  Batches             3 Months                   3                            1.0000     1
##  Locations           8 Athletes[Months]         2 Intensities              2      0.0625     1
##                        Tests[Months:Athletes]   6 Surfaces                 2      0.0625     1
##                                                   Intensities#Surfaces     4      0.2500     1
##  Batches#Locations  24 Athletes[Months]         6 Intensities              2      0.9375     1
##                                                   Residual                 4      1.0000     1
##                        Tests[Months:Athletes]  18 Surfaces                 2      0.9375     1
##                                                   Intensities#Surfaces     4      0.7500     1
##                                                   Residual                12      1.0000     1
##
## The design is not orthogonal
```

It is clear that Months:Athletes and Months:Athletes:Tests are not orthogonal to Locations and Batches:Locations, because the former sources are confounded with both of the latter sources. To examine the nature of the nonorthogonality, the anatomy for just the tests and locations tiers is obtained.

```
#"### Examine the nonorthogonality between locations and tests
eg3.locstests.canon <- designAnatomy(formulae = list(locs = ~Batches*Locations,
                                                     tests = ~Months/Athletes/Tests),
                                     data = eg3.lay)
summary(eg3.locstests.canon, which.criteria =c("aefficiency", "order"))

##
##
## Summary table of the decomposition for locs & tests
##
##  Source.locs       df1 Source.tests           df2 aefficiency order
##  Batches             3 Months                   3      1.0000     1
##  Locations           8 Athletes[Months]         2      1.0000     1
##                        Tests[Months:Athletes]   6      1.0000     1
##  Batches#Locations  24 Athletes[Months]         6      1.0000     1
##                        Tests[Months:Athletes]  18      1.0000     1
```

Questions

1. What do you conclude about the confounding of Months:Athletes and Months:Athletes:Tests with Locations?

   Since all efficiency factors are one, it is concluded that the 8 degrees of freedom for Months:Athletes has been split into two orthogonal parts, one with 2 degrees of freedom which is confounded with Batches and the other with 6 degrees of freedom which is confounded with Batches:Locations. The source Months:Athletes:Tests has been similarly partitioned.

2. Are the designs proposed for this experiment first-order balanced?

   The design is first-order balanced, because the order of the efficiency factors is one for all confounded sources.

3. What has been the cost of allowing for order of processing in the lab? Is the cost acceptable? Why?

   The cost has been that some information about Months:Athletes, along with Intensities, and some information about Months:Athletes:Tests, along with Surfaces and Intensities:Surfaces, has been confounded with Locations. The cost is acceptable, because the amount of information lost on the main effects is only 6.25% and on the interaction is 25%. The latter will be recovered in a REML-based mixed model analysis. However, the Resdiual degrees of freedom for Months:Athletes has been reduced from 6 to 4 and for Months:Athletes:Tests from 18 to 14. While the latter is unlikely to be seriously deleterious, the former is of concern.

4.2 McIntyre's (1955) two-phase example

McIntyre (1955) reports an investigation of the effect of four light intensities on the synthesis of tobacco mosaic virus in leaves of tobacco *Nicotiana tabacum* var. Hickory Pryor. It is a two-phase experiment: the first phase is a treatment phase, in which the four light treatments are randomized to the tobacco leaves, and the second phase is an assay phase, in which the tobacco leaves are randomized to the half-leaves of assay plants.

In the first phase, four successive leaves at defined positions on the stem were taken from each of eight plants of comparable age and vigour that had been inoculated with the virus. Arbitrarily grouping the plants into two sets of four, the four treatments were applied to the leaves, which had been separated from the plants and were sustained by flotation on distilled water, in a Latin square design for each set with tobacco plants as columns and leaf positions as rows; see Figure 12.

In the second phase, virus content of each tobacco leaf was assayed by expressing sap and inoculating half leaves of the assay plants, *Datura stramonium*, on which countable lesions would appear. Lots of eight sap samples were formed from pairs of tobacco plants, the pairs being comprised of a plant from each set in the treatment phase. The eight samples from a lot were assigned to four assay plants using one of four 4×4 Graeco-Latin square designs, with the leaves from a single tobacco plant assigned using one of the alphabets and the second tobacco plant using the other (see Figure 13). Actually, this design is a semi-Latin square (Bailey, 1992).

The randomization diagram for the experiment in Figure 11. Unfortunately, the randomization for this experiment was not described by McIntyre (1955). Because there are multiple squares in both phases, there are several possible randomizations depending on the effects anticipated as possible in the experiment. As shown by the nesting relations in the randomization diagram, I have assumed that NicPlant is randomized within Sets and Posn randomized across Sets. Similarly, I have assumed that DatPlant was randomized with Lot and AssPosn across Lot. In the randomization diagram, N_1 is a factor for the pairs of tobacco plants formed by taking a plant from each set in the first phase.

Figure 11: Randomization diagram for McIntyre's (1955) two-phase experiment

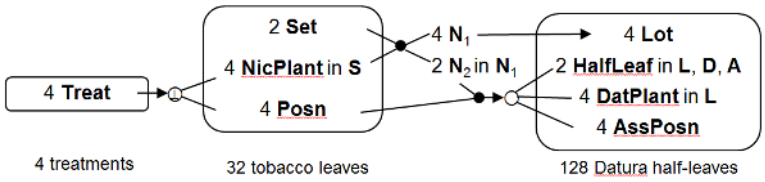


Figure 12: Layout for the first phase of McIntyre's (1955) experiment†

Nicotiana Plants

Leaf Position	1	2	3	4	Leaf Position	1	2	3	4
1	a	b	c	d		a	b	c	d
	1	5	9	13		17	21	25	29
2	b	a	d	c		c	d	a	b
	2	6	10	14		18	22	26	30
3	c	d	a	b		d	c	b	a
	3	7	11	15		19	23	27	31
4	d	c	b	a		b	a	d	c
	4	8	12	16		20	24	28	32

†The letter in each cell refers to the light intensity to be applied to the unit and the number to the unit.

Figure 13: Layout for the second phase of McIntyre's (1955) experiment†

Datura Plants

Assay Leaf Position	1	2	3	4	Assay Leaf Position	5	6	7	8
1	1	2	3	4		5	6	7	8
	17	20	18	19		23	22	24	21
2	2	1	4	3		8	7	6	5
	18	19	17	20		22	23	21	24
3	3	4	1	2		7	8	5	6
	19	18	20	17		21	24	22	23
4	4	3	2	1		6	5	8	7
	20	17	19	18		24	21	23	22

Datura Plants

Assay Leaf Position	9	10	11	12	Assay Leaf Position	13	14	15	16
1	9	10	11	12		13	14	15	16
	28	25	27	26		30	31	29	32
2	10	9	12	11		16	15	14	13
	27	26	28	25		31	30	32	29
3	11	12	9	10		15	16	13	14
	26	27	25	28		32	29	31	30
4	12	11	10	9		14	13	16	15
	25	28	26	27		29	32	30	31

†The numbers in the cell refer to the units from the first phase (tobacco leaves) to be assigned to the two half-leaves of the assay plant; they are in standard order for Set, then NicPlant followed by Position.

4.2.1 Check the properties of the randomized layout

Load the data and use `designAnatomy` to check the properties of the design.

```
#'## Load data
data("McIntyreTMV.dat")
#'## Check properties of the design
TMV.canon <- designAnatomy(formulae = list(assay = ~((Lot/DatPlant)*AssPosn)/HalfLeaf,
                                           test = ~(Set/NicPlant)*Posn,
                                           trt = ~Treat),
                           data=McIntyreTMV.dat)
summary(TMV.canon, which.criteria=c("aeff", "ord"))

##
##
## Summary table of the decomposition for assay, test & trt (based on adjusted quantities)
##
##  Source.assay                    df1 Source.test        df2 Source.trt df3 aefficiency order
##  Lot                               3 NicPlant[Set]        3                     1.0000     1
##  DatPlant[Lot]                    12
##  AssPosn                           3
##  Lot#AssPosn                       9
##  DatPlant#AssPosn[Lot]            36 Posn                   3                     0.5000     1
##                                      Set#Posn               3                     0.5000     1
##                                      NicPlant#Posn[Set]    18 Treat         3      0.5000     1
##                                                               Residual     15      0.5000     1
##                                      Residual              12
##  HalfLeaf[Lot:DatPlant:AssPosn]   64 Set                    1                     1.0000     1
##                                      NicPlant[Set]          3                     1.0000     1
##                                      Posn                   3                     0.5000     1
##                                      Set#Posn               3                     0.5000     1
##                                      NicPlant#Posn[Set]    18 Treat         3      0.5000     1
##                                                               Residual     15      0.5000     1
##                                      Residual              36
##
## The design is not orthogonal
```

4.2.2 Questions

1. Is the variance matrix for this experiment based on two sets of terms that are orthogonal?

   The variance matrix for this experiment is based on the factors in the tobacco leaves and Datura half-leaves tiers. The terms derived from the factors in these two tiers are not orthogonal. In particular, Set:Posn and Set:NicPlant:Posn are partially confounded with both Lot:DatPlant:AssPosn and Lot:DatPlant:AssPosn:HalfLeaf.

2. What are the advantages and disadvantages of a mixed -model analysis of the data from this experiment, as opposed to an anova?

   The advantage of a mixed-model analysis is that combined estimates will be provided for Set:Posn, Set:NicPlant:Posn, and Treat. The disadvantages are (i) that not all random terms are well-estimated, some having small degrees of freedom, and cause problems in fitting the model, and (ii) the Wald F-statistics are only approximately distributed as F-distributions. On the other hand, an anova is not applicable because of the nonorthogonality between the sets of terms making up the variance matrix; at least some F-ratios will not be independently distributed.

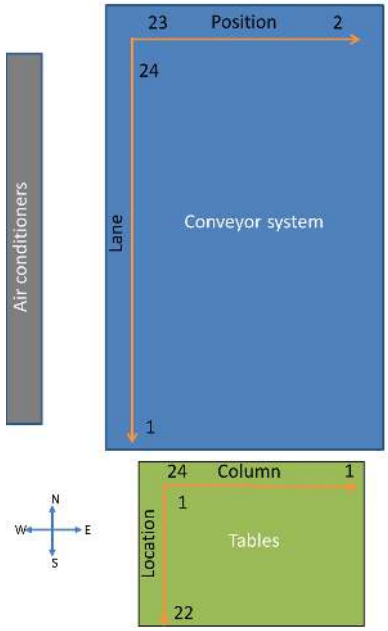
4.3 A Plant Accelerator experiment with a split-plot design

This experiment involves the investigation of 75 wheat lines, of which 73 are Nested Association Mapping (NAM) wheat lines and the other two are two check lines, Scout and Gladius. It was conducted in 2014 in the Plant Accelerator, a facility in Adelaide with 4 Smarthouses. A Smarthouse is a large greenhouse with two areas within it: (i) a Table area at the southern end and (ii) a Conveyor area at the northern end — see Figure 14. The conveyor system has the capability of automatically moving and imaging around 500 pots per day. There are air conditioners placed down the western side of the Smarthouse, which creates a trend from west to east. Further, there is a north-south trend due to changes in light intensity (Brien et al., 2013).

The experiment involves two phases: the table and conveyor phases. The table phase is the establishment phase in which plants are germinated in pots on the tables where they undergo an early growth stage. In the conveyor phase, having placed the pots in carts on the conveyor system, the plants are automatically imaged and watered daily, being moved to a processing station by the conveyor system for this.

This experiment has a single plant per pot and these will be arranged in a 24×22 grid in both phases: 24 columns $\times$ 22 locations in the table phase and 24 lanes $\times$ 22 (2–23) positions, as shown in Figure 14. However, the 24 columns in the table phase run east-west and the 24 lanes in the conveyor phase run north-south. Because there are systematic trends in both phases to be accounted for in the analysis, the same layout will be used in both phases, but the table layout will be rotated clockwise through 90°. That is, Locations 1–22 will be in Positions 2–23, respectively, and the Column will be placed in the Lane with the same number.

Figure 14: Schematic of Smarthouse for the Plant Accelerator experiment

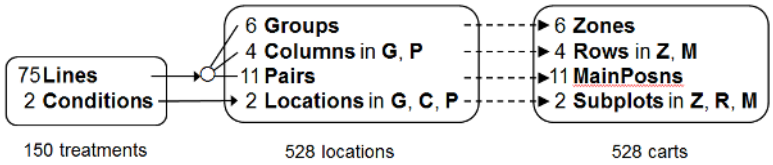


The design employed for the experiment is a split-plot design in which two consecutive pots/carts form a main plot. The main-plot design uses a blocked design with rows and columns generated with `DiGGer` (Coombes, 2009). It assigns Lines to main plots, the Lines being unequally replicated; Scout and Gladius each occur on 12 main plots (24 carts), 21 randomly-selected NAM lines each occur on 4 main plots (8 carts) and the remaining 52 NAM lines each occur on 3 main plots (6 carts). The subplot design merely randomizes Conditions (0 mM NaCl, 100 mM NaCl) to the two carts in each main plot.

In the main-plot design, the blocks are, in the table phase, 6 Groups of 4 Columns and, in the conveyor phase, 6 Zones of 4 Rows (lanes). However, while the generated design is based on crossed rows and columns, it is known from past experience that, while there are differences between Zones, there are not differences between Rows within Zones (Brien et al., 2013) and none are anticipated between Columns within Groups on the tables. The columns of the main-plot design are indexed by 11 Pairs in the table phase and 11 MainPosns in the conveyor phase. The design generated with `DiGGer` (Coombes, 2009) will be rerandomized so that the Lines are

randomized to 4 Columns within each Groups-Pairs combination and the 11 sets of Lines assigned by `DiGGer` to the 11 Pairs will be rerandomized to Pairs. The allocation diagram is shown in Figure 15.

Figure 15: Allocation diagram for the Plant Accelerator experiment



4.3.1 Produce the layout

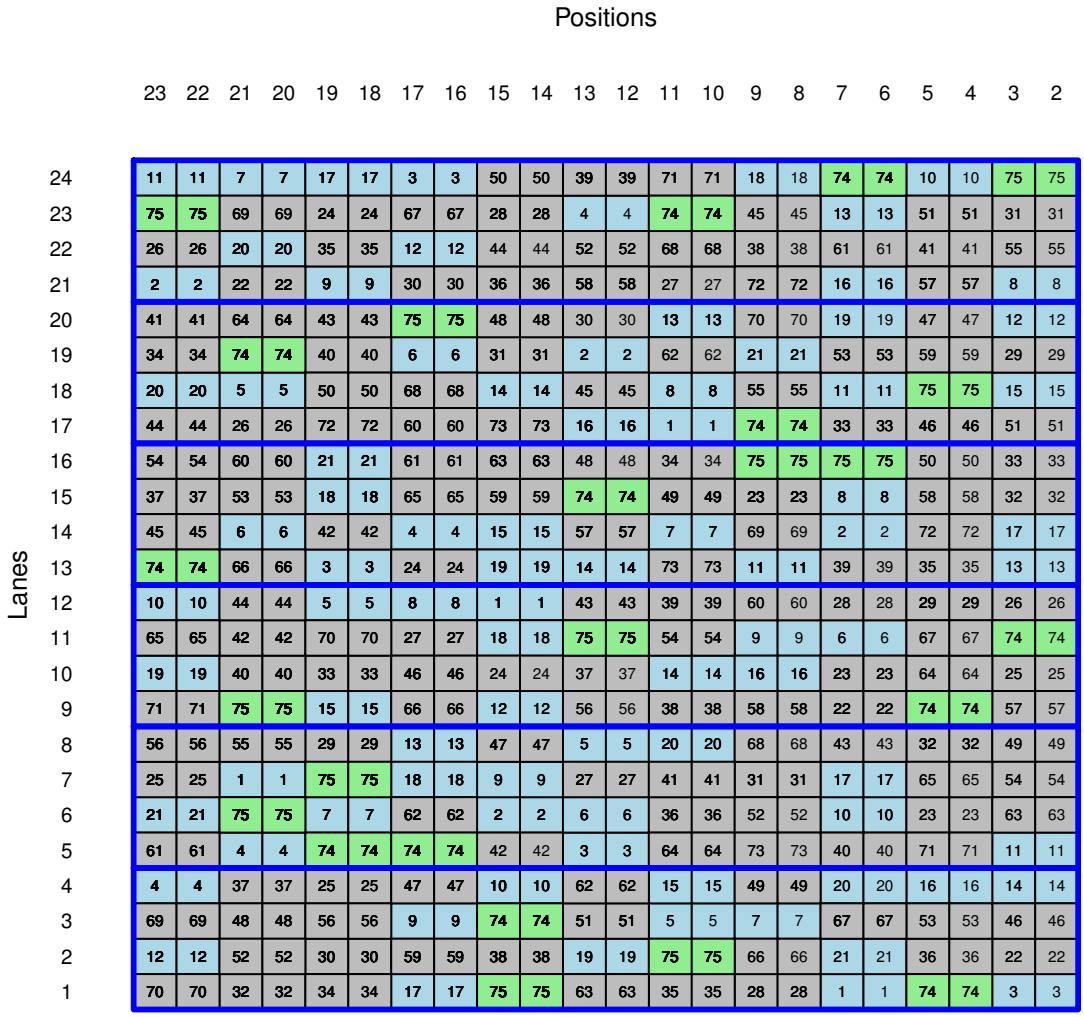
Use the following instructions to load the main-plot design produce with `DiGGer` and check its properties using `designAnatomy`.

```
#'## Load the main-plot design - it has Lines in row-column order
data("Exp249.mplot.des")
Exp249.mplot.des$Blocks <- factor(rep(1:6, each = 44))

#'## Check its properties
Exp249.mplot.canon <- designAnatomy(formulae = list(cart = ~(Blocks/Rows)*Cols,
                                                    treat = ~Lines),
                                    data = Exp249.mplot.des)
summary(Exp249.mplot.canon)

##
##
## Summary table of the decomposition for cart & treat (based on adjusted quantities)
##
##  Source.cart       df1 Source.treat df2 aefficiency eefficiency order
##  Blocks              5 Lines          5      0.1498      0.1422     5
##  Rows[Blocks]       18 Lines         18      0.2685      0.1813    18
##  Cols               10 Lines         10      0.2102      0.1769    10
##  Blocks#Cols        50 Lines         50      0.1154      0.0142    50
##  Rows#Cols[Blocks] 180 Lines         74      0.5816      0.2088    74
##                        Residual     106
##
## The design is not orthogonal
```

Expand main-plot design to produce the split-plot design, including a three-level factor Checks that compares Scout, Gladius and the mean of the NAM lines. Perhaps, produce a plot of the allocation of the Lines.

```
#'## Expand design to rerandomize lines and to assign conditions to locations
Exp249.recip <- list(Groups = 6, Columns = 4, Pairs = 11, Locations = 2)
Exp249.nest  <- list(Columns = c("Groups", "Pairs"),
                     Locations = c("Groups", "Columns", "Pairs"))
Exp249.alloc <- data.frame(Lines = factor(rep(Exp249.mplot.des$Lines, each=2),
                                          levels=1:75),
                           Checks = fac.recode(rep(Exp249.mplot.des$Lines, each=2),
                                               newlevels=c(rep(3, 73), 1 , 2),
                                               labels = c("NAM","Scout","Gladius")),
                           Conditions = factor(rep(1:2, times=264),
                                               labels = c('0 NaCl','100 NaCl')))
```

```
Exp249.lay <- designRandomize(allocated = Exp249.alloc,
                              recipient = Exp249.recip,
                              nested.recipients = Exp249.nest,
                              seed = 51412)



#'## Add second-phase factors
#'## (to which the first-phase factors have been systematically allocated)
Exp249.lay <- cbind(fac.gen(list(Lanes = 24, Positions = 2:23)),
                    fac.gen(list(Zones = 6, Rows = 4, MainPosn = 11, Subplots = 2)),
                    Exp249.lay)

#'## Plot the assignment of Lines in the second-phase design - or see file that includes the output
SPL.Line.mat <- matrix(as.numfac(Exp249.lay$Lines), ncol=22, byrow=T)
colnames(SPL.Line.mat) <- 2:23
rownames(SPL.Line.mat) <- 1:24
SPL.Line.mat <- SPL.Line.mat[24:1, 22:1]
designPlot(SPL.Line.mat, labels = 1:21, cellfillcolour = "lightblue", new=TRUE,
           plotlabels = TRUE, rtitle = "Lanes", ctitle = "Positions",
           chardivisor = 4, rchardivisor = 3, cchardivisor = 3, plotcellboundary = TRUE)
designPlot(SPL.Line.mat, labels = 22:73, cellfillcolour = "grey", new=FALSE,
           plotlabels = TRUE, chardivisor = 4, rchardivisor = 3, cchardivisor = 3,
           plotcellboundary = TRUE)
designPlot(SPL.Line.mat, labels = 74:75, cellfillcolour = "lightgreen", new=FALSE,
           plotlabels = TRUE, chardivisor = 4, rchardivisor = 3, cchardivisor = 3,
           plotcellboundary = TRUE,
           blocksequence = TRUE, blockdefinition = cbind(4,22),
           blocklinewidth = 3, blocklinecolour = "blue")
```



4.3.2 Check the properties of the design

The maximal allocation-based mixed model is E[Y] = (Checks + Lines) * Conditions, with Checks nested within Lines, and var[Y] = (Zones * MainPosn) / Rows / Subplots + (Groups * Pairs) / Columns / Locations. Use the `designAnatomy` to check the properties of the design for an analysis of data from an experiment based on this design.

```
#'## Check design properties
Exp249.canon <- designAnatomy(formulae = list(carts = ~(Zones*MainPosn)/Rows/Subplots,
                                              tables = ~(Groups*Pairs)/Columns/Locations,
                                              treats = ~(Checks + Lines) * Conditions),
                              data = Exp249.lay)
```

```
## Warning in projs.2canon(CombinedSets$Q[[ntiers]], struct[[ktier]]$Q): Lines[Checks] and Checks
are partially aliased in MainPosn&Pairs
## Warning in projs.2canon(CombinedSets$Q[[ntiers]], struct[[ktier]]$Q): Lines[Checks] and Checks
are partially aliased in Zones#MainPosn&Groups#Pairs
## Warning in projs.2canon(CombinedSets$Q[[ntiers]], struct[[ktier]]$Q): Lines[Checks] and Checks
are partially aliased in Rows[Zones:MainPosn]&Columns[Groups:Pairs]
```

```
summary(Exp249.canon)
```

```
##
##
## Summary table of the decomposition for carts, tables & treats (based on adjusted quantities)
##
##  Source.carts                   df1 Source.tables                    df2 Source.treats            df3
##  Zones                            5 Groups                             5 Lines[Checks]              5
##  MainPosn                        10 Pairs                             10 Checks                     2
##                                                                          Lines[Checks]               8
##  Zones#MainPosn                  50 Groups#Pairs                      50 Checks                     2
##                                                                          Lines[Checks]              48
##  Rows[Zones:MainPosn]           198 Columns[Groups:Pairs]            198 Checks                     2
##                                                                          Lines[Checks]              72
##                                                                          Residual                  124
##  Subplots[Zones:MainPosn:Rows] 264 Locations[Groups:Pairs:Columns] 264 Conditions                   1
##                                                                          Checks#Conditions           2
##                                                                          Lines#Conditions[Checks]   72
##                                                                          Residual                  189
##  aefficiency eefficiency order
##       0.1498      0.1422     5
##       0.0033      0.0031     2
##       0.2094      0.1809     8
##       0.2111      0.2049     2
##       0.1142      0.0145    48
##       0.7854      0.7792     2
##       0.6640      0.2632    66
##       1.0000      1.0000     1
##       1.0000      1.0000     1
##       1.0000      1.0000     1
##       1.0000      1.0000     1
##       1.0000      1.0000     1
##
## Table of (partial) aliasing between sources derived from the same formula
##
##  Source        df Alias  In                                          aefficiency eefficiency order
##  Lines[Checks]  8 Checks MainPosn&Pairs                                   0.2094      0.1809     8
##  Lines[Checks] 48 Checks Zones#MainPosn&Groups#Pairs                      0.1142      0.0145    48
##  Lines[Checks] 72 Checks Rows[Zones:MainPosn]&Columns[Groups:Pairs]       0.6640      0.2632    66
##
## The design is not orthogonal
```

Because, there is a one-to-one correspondence between the tables and carts sources, omit the tables formula and rerun — it will make the anova table more readable.

```
#'## Check design properties, with tables omitted
Exp249.canon <- designAnatomy(formulae = list(carts = ~(Zones*MainPosn)/Rows/Subplots,
```

```
                                              treats = ~(Checks + Lines) * Conditions),
                            data = Exp249.lay)

## Warning in projs.2canon(CombinedSets$Q[[ntiers]], struct[[ktier]]$Q): Lines[Checks] and Checks
are partially aliased in MainPosn
## Warning in projs.2canon(CombinedSets$Q[[ntiers]], struct[[ktier]]$Q): Lines[Checks] and Checks
are partially aliased in Zones#MainPosn
## Warning in projs.2canon(CombinedSets$Q[[ntiers]], struct[[ktier]]$Q): Lines[Checks] and Checks
are partially aliased in Rows[Zones:MainPosn]

summary(Exp249.canon)

##
##
## Summary table of the decomposition for carts & treats (based on adjusted quantities)
##
##  Source.carts                   df1 Source.treats            df2 aefficiency eefficiency order
##  Zones                            5 Lines[Checks]              5      0.1498      0.1422     5
##  MainPosn                        10 Checks                     2      0.0033      0.0031     2
##                                     Lines[Checks]              8      0.2094      0.1809     8
##  Zones#MainPosn                  50 Checks                     2      0.2111      0.2049     2
##                                     Lines[Checks]             48      0.1142      0.0145    48
##  Rows[Zones:MainPosn]           198 Checks                     2      0.7854      0.7792     2
##                                     Lines[Checks]             72      0.6640      0.2632    66
##                                     Residual                 124
##  Subplots[Zones:MainPosn:Rows] 264 Conditions                  1      1.0000      1.0000     1
##                                     Checks#Conditions          2      1.0000      1.0000     1
##                                     Lines#Conditions[Checks]  72      1.0000      1.0000     1
##                                     Residual                 189
##
## Table of (partial) aliasing between sources derived from the same formula
##
##  Source        df Alias  In                    aefficiency eefficiency order
##  Lines[Checks]  8 Checks MainPosn                    0.2094      0.1809     8
##  Lines[Checks] 48 Checks Zones#MainPosn             0.1142      0.0145    48
##  Lines[Checks] 72 Checks Rows[Zones:MainPosn]       0.6640      0.2632    66
##
## The design is not orthogonal
```

4.3.3 Examine the properties of the design for an alternative analysis

However, rather than fit the allocation-based model, because it is known from past experience that once a linear trend for MainPosn has been fitted there are no deviations from this trend, the term xMainPosn is used to fit the trend; the term `xMainPosn` is a centred, linear covariate for MainPosn. Use the `designAnatomy` to check the properties of the design for an analysis based on a modified model, in which MainPosn in the random model has been replaced by xMainPosn in the fixed model, Zones:MainPosn has been omitted and Zones:MainPosn:Rows has been replaced by Zones:Mainplots.

```
#'## Add factors and variates for new analysis
Exp249.lay <- within(Exp249.lay,
                     { xMainPosn <- as.numfac(MainPosn)
                       xMainPosn <- -(xMainPosn - mean(xMainPosn))
                       Mainplots <- fac.combine(list(Rows,MainPosn))
                     })
```

```
head(Exp249.lay)
```

```
##   Lanes Positions Zones Rows MainPosn Subplots Groups Columns Pairs Locations Lines  Checks
## 1     1         2     1    1        1        1      1       1     1         1     3 Gladius
## 2     1         3     1    1        1        2      1       1     1         2     3 Gladius
## 3     1         4     1    1        2        1      1       1     2         1    74     NAM
## 4     1         5     1    1        2        2      1       1     2         2    74     NAM
## 5     1         6     1    1        3        1      1       1     3         1     1 Gladius
## 6     1         7     1    1        3        2      1       1     3         2     1 Gladius
##   Conditions Mainplots xMainPosn
## 1   100 NaCl         1         5
## 2     0 NaCl         1         5
## 3   100 NaCl         2         4
## 4     0 NaCl         2         4
## 5   100 NaCl         3         3
## 6     0 NaCl         3         3
```

```
#'## Check properties if only linear trend fitted
Exp249.canon <- designAnatomy(formulae = list(cart = ~ Zones/Mainplots/Subplots,
                                              treat = ~ xMainPosn +
                                                (Checks + Lines) * Conditions),
                              data = Exp249.lay)
summary(Exp249.canon)
```

```
##
##
## Summary table of the decomposition for cart & treat (based on adjusted quantities)
##
##  Source.cart               df1 Source.treat             df2 aefficiency eefficiency order
##  Zones                       5 Lines[Checks]              5      0.1500      0.1426     5
##  Mainplots[Zones]          258 xMainPosn                  1      1.0000      1.0000     1
##                                Checks                     2      1.0000      1.0000     1
##                                Lines[Checks]             72      0.9879      0.8437     6
##                                Residual                 183
##  Subplots[Zones:Mainplots] 264 Conditions                 1      1.0000      1.0000     1
##                                Checks#Conditions          2      1.0000      1.0000     1
##                                Lines#Conditions[Checks]  72      1.0000      1.0000     1
##                                Residual                 189
##
## Table of (partial) aliasing between sources derived from the same formula
##
##  Source                    df  Alias     In    aefficiency eefficiency order
##  Checks                     2 xMainPosn treat      0.9995      0.9990     2
##  Lines[Checks]             74 xMainPosn treat      0.9963      0.7851     2
##  Checks#Conditions          5 xMainPosn treat      0.9998      0.9990     2
##  Lines#Conditions[Checks] 149 xMainPosn treat      0.9982      0.7851     2
##
## The design is not orthogonal
```

The Table of (partial) aliasing shows that all `treat- sources are partially aliased with \verb`xMainPosn=, although they are not far from being orthogonal.

We have been able to check what information is available about Lines and Conditions after adjustment for the linear trend. In practice, a spline term might be needed to account for nonlinearity in the trend.

4.3.4 Questions

1. What advantages accrue from randomizing Lines within Groups:Pairs (Zones:MainPosn) as compared to the original DiGGer design, in which they are randomized to Columns within Groups (Lanes within Zones) and to Pairs (MainPosn)?

   The anatomy for the `DiGGer` *design shows us that all 74 degrees of freedom are estimable in Blocks:Rows:Columns with average efficiency 0.582 and minimum efficiency 0.209. Compared to this, the anatomy for the rerandomized design show that the NAM lines are estimable from Zones:MainPosn:Rows, the source equivalent to Blocks:Rows:Columns, with average efficiency 0.664 and minimum efficiency 0.263. Also, the Residual degrees of freedom for Blocks:Rows:Columns have increased from 106 degrees of freedom in the original design to 124 degrees of freedom for Zones:MainPosn:Rows in the rerandomized design. That is, one can expect the estimation of the Lines predictions and their standard errors to be more precise for the rerandomized design.*

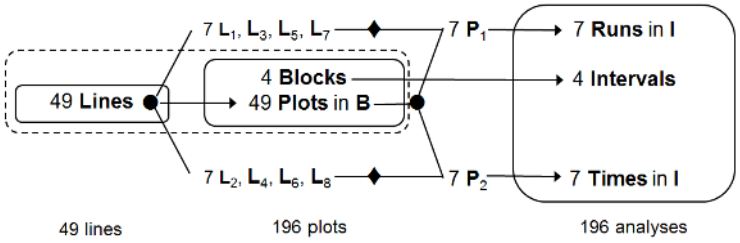
2. What effect does the use of a linear trend, as opposed to a set of effects, have on the analysis?

   The efficiency for Lines has increased further so that the minimum is now 0.844 and the Residual degrees of freedom for Zones:MainPosn:Rows now stands at 183. This allows one to consider ignoring information not estimable from Zones:MainPosn:Rows, while predictions will be adjusted for the trend across MainPosn.

4.4 Two-phase, wheat experiment with 49 lines

The first, or field, phase of a wheat trial for 49 lines is laid out as an RCBD with four blocks. The produce from the field trial is processed in the second, or laboratory, phase and the design employed is a balanced lattice square for 49 treatments that involves 4 replicates each consisting of a 7×7 square. In the laboratory phase there are four intervals each of which consists of 7 runs of a machine. In a run, samples are processed at seven consecutive times. This experiment is Example 2.2 from Bailey and Brien (2015), where its anova with expected mean squares is given. Its randomization diagram is in Figure 16.

Figure 16: Randomization diagram for the two-phase wheat variety experiment



4.4.1 Produce randomized layout for both phases and check its properties

```
#'## Generate a layout for the field phase
Lines  <- factor(rep(1:49, times=4))
field.lay <- designRandomize(allocated = Lines,
                             recipient = list(Blocks = 4, Plots = 49),
                             nested.recipients = list(Plots = "Blocks"),
                             seed = 82522)
field.lay

##     Blocks Plots Lines
## 1        1     1    48
## 2        1     2    10
## 3        1     3    23
```

```
## 4        1     4    31
## 5        1     5    36
## 6        1     6    11
## 7        1     7    18
## 8        1     8    40
## 9        1     9    12
## 10       1    10    43
## 11       1    11    30
## 12       1    12     7
## 13       1    13    29
## 14       1    14    20
## 15       1    15    46
## 16       1    16    28
## 17       1    17    37
## 18       1    18    21
## 19       1    19    34
## 20       1    20     4
## 21       1    21    24
## 22       1    22    32
## 23       1    23    41
## 24       1    24     6
## 25       1    25    38
## 26       1    26    45
## 27       1    27    22
## 28       1    28    42
## 29       1    29    19
## 30       1    30     2
## 31       1    31    33
## 32       1    32    13
## 33       1    33     9
## 34       1    34    15
## 35       1    35     5
## 36       1    36    16
## 37       1    37    35
## 38       1    38     8
## 39       1    39    49
## 40       1    40     1
## 41       1    41     3
## 42       1    42    39
## 43       1    43    26
## 44       1    44    14
## 45       1    45    27
## 46       1    46    17
## 47       1    47    44
## 48       1    48    25
## 49       1    49    47
## 50       2     1    35
## 51       2     2    21
## 52       2     3    11
## 53       2     4    42
## 54       2     5    44
## 55       2     6    19
## 56       2     7    29
```

```
## 57       2     8    45
## 58       2     9    13
## 59       2    10    49
## 60       2    11    30
## 61       2    12    18
## 62       2    13    37
## 63       2    14    23
## 64       2    15    16
## 65       2    16    22
## 66       2    17    14
## 67       2    18    12
## 68       2    19     6
## 69       2    20    41
## 70       2    21    28
## 71       2    22     9
## 72       2    23    17
## 73       2    24     5
## 74       2    25    31
## 75       2    26    40
## 76       2    27     4
## 77       2    28    32
## 78       2    29     1
## 79       2    30    43
## 80       2    31     2
## 81       2    32    48
## 82       2    33    38
## 83       2    34    27
## 84       2    35    39
## 85       2    36    26
## 86       2    37    46
## 87       2    38     3
## 88       2    39     8
## 89       2    40    15
## 90       2    41    10
## 91       2    42    24
## 92       2    43    36
## 93       2    44    25
## 94       2    45    34
## 95       2    46    47
## 96       2    47    33
## 97       2    48     7
## 98       2    49    20
## 99       3     1     8
## 100      3     2    23
## 101      3     3    40
## 102      3     4    16
## 103      3     5    13
## 104      3     6     7
## 105      3     7     1
## 106      3     8    26
## 107      3     9    15
## 108      3    10     6
## 109      3    11    14
```

```
## 110      3    12     4
## 111      3    13    28
## 112      3    14    30
## 113      3    15    25
## 114      3    16    31
## 115      3    17    12
## 116      3    18    19
## 117      3    19    20
## 118      3    20    39
## 119      3    21     3
## 120      3    22    42
## 121      3    23    38
## 122      3    24    17
## 123      3    25    10
## 124      3    26    46
## 125      3    27    34
## 126      3    28    35
## 127      3    29    29
## 128      3    30    44
## 129      3    31    18
## 130      3    32    24
## 131      3    33    21
## 132      3    34    47
## 133      3    35    22
## 134      3    36    33
## 135      3    37    41
## 136      3    38     2
## 137      3    39     9
## 138      3    40    48
## 139      3    41     5
## 140      3    42    37
## 141      3    43    43
## 142      3    44    32
## 143      3    45    49
## 144      3    46    45
## 145      3    47    27
## 146      3    48    36
## 147      3    49    11
## 148      4     1    24
## 149      4     2    40
## 150      4     3    39
## 151      4     4    28
## 152      4     5    46
## 153      4     6    20
## 154      4     7    11
## 155      4     8    10
## 156      4     9    36
## 157      4    10    33
## 158      4    11    18
## 159      4    12    12
## 160      4    13     9
## 161      4    14    38
## 162      4    15    43
```

```
## 163      4    16    42
## 164      4    17    23
## 165      4    18     6
## 166      4    19    13
## 167      4    20    14
## 168      4    21    29
## 169      4    22     4
## 170      4    23    19
## 171      4    24    30
## 172      4    25     2
## 173      4    26    44
## 174      4    27    26
## 175      4    28    35
## 176      4    29    17
## 177      4    30     7
## 178      4    31    32
## 179      4    32    37
## 180      4    33    31
## 181      4    34     8
## 182      4    35    21
## 183      4    36    25
## 184      4    37    15
## 185      4    38    34
## 186      4    39    48
## 187      4    40    27
## 188      4    41     1
## 189      4    42    16
## 190      4    43    49
## 191      4    44    41
## 192      4    45    22
## 193      4    46     3
## 194      4    47     5
## 195      4    48    45
## 196      4    49    47
```

```
#'## Generate laboratory phase
#'### Load a systematic balanced lattice square
data("LatticeSquare_t49.des")
#'### Form systematic design
#'### Add Intervals to field layout,  merge the data frames and sort into lab phase order
field.lay$Intervals <- field.lay$Blocks
lab.alloc <- merge(LatticeSquare_t49.des, field.lay)
lab.alloc <- with(lab.alloc, lab.alloc[order(Intervals, Runs,Times),])
lab.alloc <- lab.alloc[c("Blocks","Plots","Lines")] #Reduce columns in lab.alloc
#'### Randomize the design
lab.lay <- designRandomize(allocated = lab.alloc,
                           recipient = list(Intervals = 4, Runs = 7, Times = 7),
                           nested.recipients = list(Runs = "Intervals",
                                                    Times = "Intervals"),
                           seed = 141797)
lab.lay
```

```
##     Intervals Runs Times Blocks Plots Lines
## 1           1    1     1      4    41     1
```

```
## 2           1    1     2      4    43    49
## 3           1    1     3      4    36    25
## 4           1    1     4      4    13     9
## 5           1    1     5      4    10    33
## 6           1    1     6      4    44    41
## 7           1    1     7      4    29    17
## 8           1    2     1      4     5    46
## 9           1    2     2      4    14    38
## 10          1    2     3      4    35    21
## 11          1    2     4      4    47     5
## 12          1    2     5      4    45    22
## 13          1    2     6      4    24    30
## 14          1    2     7      4    19    13
## 15          1    3     1      4    33    31
## 16          1    3     2      4    17    23
## 17          1    3     3      4    18     6
## 18          1    3     4      4     3    39
## 19          1    3     5      4    20    14
## 20          1    3     6      4    37    15
## 21          1    3     7      4    49    47
## 22          1    4     1      4    40    27
## 23          1    4     2      4    23    19
## 24          1    4     3      4    26    44
## 25          1    4     4      4    28    35
## 26          1    4     5      4    46     3
## 27          1    4     6      4     7    11
## 28          1    4     7      4     9    36
## 29          1    5     1      4    42    16
## 30          1    5     2      4    34     8
## 31          1    5     3      4     2    40
## 32          1    5     4      4     1    24
## 33          1    5     5      4    39    48
## 34          1    5     6      4    30     7
## 35          1    5     7      4    31    32
## 36          1    6     1      4    12    12
## 37          1    6     2      4    22     4
## 38          1    6     3      4    21    29
## 39          1    6     4      4     6    20
## 40          1    6     5      4    32    37
## 41          1    6     6      4    48    45
## 42          1    6     7      4     4    28
## 43          1    7     1      4    16    42
## 44          1    7     2      4    38    34
## 45          1    7     3      4     8    10
## 46          1    7     4      4    15    43
## 47          1    7     5      4    11    18
## 48          1    7     6      4    27    26
## 49          1    7     7      4    25     2
## 50          2    1     1      2    23    17
## 51          2    1     2      2    26    40
## 52          2    1     3      2    34    27
## 53          2    1     4      2    17    14
## 54          2    1     5      2    30    43
```

```
## 55          2     1     6       2    27     4
## 56          2     1     7       2    11    30
## 57          2     2     1       2     4    42
## 58          2     2     2       2    22     9
## 59          2     2     3       2     8    45
## 60          2     2     4       2    28    32
## 61          2     2     5       2     6    19
## 62          2     2     6       2    16    22
## 63          2     2     7       2    19     6
## 64          2     3     1       2     7    29
## 65          2     3     2       2    38     3
## 66          2     3     3       2    35    39
## 67          2     3     4       2    36    26
## 68          2     3     5       2     9    13
## 69          2     3     6       2    15    16
## 70          2     3     7       2    10    49
## 71          2     4     1       2    32    48
## 72          2     4     2       2    40    15
## 73          2     4     3       2    31     2
## 74          2     4     4       2    33    38
## 75          2     4     5       2    44    25
## 76          2     4     6       2     1    35
## 77          2     4     7       2    18    12
## 78          2     5     1       2     3    11
## 79          2     5     2       2    45    34
## 80          2     5     3       2     2    21
## 81          2     5     4       2    29     1
## 82          2     5     5       2    13    37
## 83          2     5     6       2    46    47
## 84          2     5     7       2    42    24
## 85          2     6     1       2    24     5
## 86          2     6     2       2    21    28
## 87          2     6     3       2    39     8
## 88          2     6     4       2     5    44
## 89          2     6     5       2    25    31
## 90          2     6     6       2    20    41
## 91          2     6     7       2    12    18
## 92          2     7     1       2    14    23
## 93          2     7     2       2    37    46
## 94          2     7     3       2    47    33
## 95          2     7     4       2    49    20
## 96          2     7     5       2    48     7
## 97          2     7     6       2    41    10
## 98          2     7     7       2    43    36
## 99          3     1     1       3    24    17
## 100         3     1     2       3    20    39
## 101         3     1     3       3     6     7
## 102         3     1     4       3    30    44
## 103         3     1     5       3    17    12
## 104         3     1     6       3    27    34
## 105         3     1     7       3    35    22
## 106         3     2     1       3     1     8
## 107         3     2     2       3    14    30
```

```
## 108         3    2     3      3    34    47
## 109         3    2     4      3    22    42
## 110         3    2     5      3    21     3
## 111         3    2     6      3    15    25
## 112         3    2     7      3    19    20
## 113         3    3     1      3    10     6
## 114         3    3     2      3    13    28
## 115         3    3     3      3    23    38
## 116         3    3     4      3    36    33
## 117         3    3     5      3    43    43
## 118         3    3     6      3     4    16
## 119         3    3     7      3    49    11
## 120         3    4     1      3    28    35
## 121         3    4     2      3     7     1
## 122         3    4     3      3    31    18
## 123         3    4     4      3     5    13
## 124         3    4     5      3     2    23
## 125         3    4     6      3    46    45
## 126         3    4     7      3     3    40
## 127         3    5     1      3    26    46
## 128         3    5     2      3    18    19
## 129         3    5     3      3    29    29
## 130         3    5     4      3    32    24
## 131         3    5     5      3    37    41
## 132         3    5     6      3    11    14
## 133         3    5     7      3    38     2
## 134         3    6     1      3     8    26
## 135         3    6     2      3    40    48
## 136         3    6     3      3    39     9
## 137         3    6     4      3    12     4
## 138         3    6     5      3    33    21
## 139         3    6     6      3    48    36
## 140         3    6     7      3    16    31
## 141         3    7     1      3    42    37
## 142         3    7     2      3    25    10
## 143         3    7     3      3    47    27
## 144         3    7     4      3     9    15
## 145         3    7     5      3    44    32
## 146         3    7     6      3    41     5
## 147         3    7     7      3    45    49
## 148         4    1     1      1    40     1
## 149         4    1     2      1    10    43
## 150         4    1     3      1    38     8
## 151         4    1     4      1    34    15
## 152         4    1     5      1    13    29
## 153         4    1     6      1     5    36
## 154         4    1     7      1    27    22
## 155         4    2     1      1    24     6
## 156         4    2     2      1     1    48
## 157         4    2     3      1    32    13
## 158         4    2     4      1    14    20
## 159         4    2     5      1    19    34
## 160         4    2     6      1    23    41
```

```
## 161         4    2     7      1    45    27
## 162         4    3     1      1    35     5
## 163         4    3     2      1    49    47
## 164         4    3     3      1     9    12
## 165         4    3     4      1    29    19
## 166         4    3     5      1    31    33
## 167         4    3     6      1     8    40
## 168         4    3     7      1    43    26
## 169         4    4     1      1    20     4
## 170         4    4     2      1    15    46
## 171         4    4     3      1     6    11
## 172         4    4     4      1     7    18
## 173         4    4     5      1    22    32
## 174         4    4     6      1    42    39
## 175         4    4     7      1    48    25
## 176         4    5     1      1    12     7
## 177         4    5     2      1    39    49
## 178         4    5     3      1    44    14
## 179         4    5     4      1    18    21
## 180         4    5     5      1    37    35
## 181         4    5     6      1    28    42
## 182         4    5     7      1    16    28
## 183         4    6     1      1    30     2
## 184         4    6     2      1    47    44
## 185         4    6     3      1    33     9
## 186         4    6     4      1    36    16
## 187         4    6     5      1    11    30
## 188         4    6     6      1    17    37
## 189         4    6     7      1     3    23
## 190         4    7     1      1    41     3
## 191         4    7     2      1    26    45
## 192         4    7     3      1     2    10
## 193         4    7     4      1    46    17
## 194         4    7     5      1     4    31
## 195         4    7     6      1    25    38
## 196         4    7     7      1    21    24

#'## Check properties of the design
wheat.canon <- designAnatomy(formulae = list(lab = ~Intervals/(Runs*Times),
                                             field = ~Blocks/Plots,
                                             treats = ~Lines),
                             data = lab.lay)
summary(wheat.canon, which.criteria =c("aefficiency", "order"))

##
##
## Summary table of the decomposition for lab, field & treats (based on adjusted quantities)
##
##  Source.lab            df1 Source.field  df2 Source.treats df3 aefficiency order
##  Intervals               3 Blocks          3                       1.0000     1
##  Runs[Intervals]        24 Plots[Blocks]  24 Lines          24      0.2500     1
##  Times[Intervals]       24 Plots[Blocks]  24 Lines          24      0.2500     1
##  Runs#Times[Intervals] 144 Plots[Blocks] 144 Lines          48      0.7500     1
##                                              Residual       96      1.0000     1
```

```
##
## The design is not orthogonal
```

Given, the nonorthogonality of Blocks:Plots evident in the anatomy, redo the table with just the lab and field tiers to investigate.

```
#'## Check confounding of field sources with lab sources
wheat.labfield.canon <- designAnatomy(formulae = list(lab = ~Intervals/(Runs*Times),
                                                      field = ~Blocks/Plots),
                                      data = lab.lay)
summary(wheat.labfield.canon, which.criteria =c("aefficiency", "order"))

##
##
## Summary table of the decomposition for lab & field
##
##  Source.lab            df1 Source.field  df2 aefficiency order
##  Intervals               3 Blocks          3      1.0000     1
##  Runs[Intervals]        24 Plots[Blocks]  24      1.0000     1
##  Times[Intervals]       24 Plots[Blocks]  24      1.0000     1
##  Runs#Times[Intervals] 144 Plots[Blocks] 144      1.0000     1
```

4.4.2 Questions

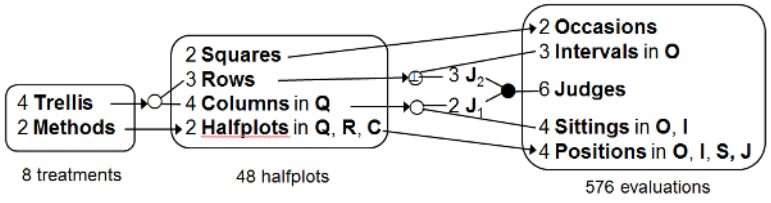
1. Is the variance matrix for this experiment based on two sets of terms that are orthogonal??

   Because all plots sources are confounded orthogonally with analyses sources, the variance matrix is indeed based on two sets of terms that are orthogonal.

4.5 Elaborate, two-phase, sensory experiment

Brien and Payne (1999) describe a two-phase sensory experiment, of which the first, or field, phase is a viticultural experiment and the second, or evaluation, phase involves the assessment of wine made from the produce of the first phase plots. In the field phase, two adjacent Youden squares are used to assign trellis treatments to the plots, a plot being a row-column combination within a square. Each plot is divided into two halfplots and two methods of pruning assigned at random to them. In the second phase, the halfplots from the field phase are randomized, using two Latin squares and an extended Youden design, to glasses in positions on a table for evaluation by judges. This experiment is Example 1.2 from Bailey and Brien (2015), where its anova, along with expected mean squares, is given. Its randomization diagram is in Figure 17.

Figure 17: Randomization diagram for the two-phase sensory experiment



4.5.1 Check the properties of the randomized layout

Load the layout and use `designAnatomy` to check the properties of the design.

```
#'## Load the layout
data("Sensory3PhaseShort.dat")

#'## Examne the properties of the design
sensory.canon <- designAnatomy(formulae = list(eval = ~((Occ/Int/Sit)*Jud)/Posn,
                                               field = ~(Row*(Sqr/Col))/Hplot,
                                               treats = ~Trel*Meth),
                               data = Sensory3PhaseShort.dat)
summary(sensory.canon, which.criteria =c("aefficiency", "order"))

##
##
## Summary table of the decomposition for eval, field & treats (based on adjusted quantities)
##
##  Source.eval           df1 Source.field       df2 Source.treats df3 aefficiency order
##  Occ                     1 Sqr                 1                         1.0000     1
##  Int[Occ]                4
##  Sit[Occ:Int]           18 Col[Sqr]            6 Trel            3      0.0370     1
##                                                  Residual        3      0.3333     1
##                            Residual            12
##  Jud                     5
##  Occ#Jud                 5
##  Int#Jud[Occ]           20 Row                 2                         1.0000     1
##                            Row#Sqr             2                         1.0000     1
##                            Residual            16
##  Sit#Jud[Occ:Int]       90 Col[Sqr]            6 Trel            3      0.0741     1
##                                                  Residual        3      0.6667     1
##                            Row#Col[Sqr]        12 Trel            3      0.8889     1
##                                                  Residual        9      1.0000     1
##                            Residual            72
##  Posn[Occ:Int:Sit:Jud] 432 Hplot[Row:Sqr:Col]  24 Meth            1      1.0000     1
##                                                  Trel#Meth       3      1.0000     1
##                                                  Residual       20      1.0000     1
##                            Residual           408
##
## The design is not orthogonal
```

Note that 1/3 of Sqr:Col is partially confounded with Occ:Int:Sit and 2/3 with Occ:Int:Sit:Jud. Also, 1/9 of Trel is partially confounded with Sqr:Col and 8/9 with Row:Sqr:Col. The canonical efficiency factor for Trel in the two Sqr:Col sources is obtained by multiplying the canonical efficiency of 1/9 for Trel with that for the particular Sqr:Col source, yielding canonical efficiencies of 1/27 and 2/27.

4.5.2 Questions

1. Which is the nonorthogonal source amongst the field sources (`Source.field`) and what is its interblock and intrablock efficiency factors?

   The only nonorthogonal field source is Sqr.Col. Its interblock efficiency factor is 1/3 *and its intrablock effciciency factor is* 2/3.

2. How would an intrablock analysis be achieved using, say, regression software?

   To achieve an intrablock analysis requires careful specification of the order of fitting terms; a nonorthogonal source should not estimated until after all nonorthogonal terms with which it is confounded, except the last, have been estimated. For this experiment, terms should be fitted in the following order:

*Occ*Jud + Row + Occ:Int/(Int + Sit) + Sqr.Col + Trel + Row:Sqr:Col + Occ:Int:Sit:Jud + Meth + Trel:Meth + Row:Sqr:Col:Hplot.*

This will leave a Residual that corresponds to Occ:Int:Sit:Jud:Posn.

5 Power and sample size for designed experiments

In power and sample size calculations, in addition to specifying delta, sigma, power and alpha, one has to supply a number of quantities that vary with the design of the experiment. The following table summarizes these for the common designs, giving the degrees of freedom of the denominator as a function of r, the pure replication of the treatments. Note that rm is the number of replicates in means being compared. For treatment means, this will be the product of r and a multiple, m, for the product of the number of levels of factors not involved in means being compared.

Design	m	rm	*df.num* (ν_1)	*df.denom* (ν_2)
CRD	1	r	$t-1$	$t(r-1)$
RCBD	1	b	$t-1$	$(t-1)(b-1)$
LSD	1	$r(=t)$	$r-1$	$(r-1)(r-2)$
Factorial				
A	b	br	$a-1$	CRD $ab(r-1)$
B	a	ar	$b-1$	RCBD $(ab-1)(r-1)$
A:B	1	r	$(a-1)(b-1)$	LSD $(r-1)(r-2)$
Standard split-plot				
A	b	br	$a-1$	$(a-1)(r-1)$
B	a	ar	$b-1$	$(b-1)(r-1)$
A:B	1	r	$(a-1)(b-1)$	$a(b-1)(r-1)^{\dagger}$

†only approximate for effects not at the same level of A

5.1 Computing the power for given sample size

The function `power.exp` from the `dae` library is used for computing the power in detecting the difference between means for some, not necessarily proper, subset of the factors from a designed experiment.

The usage and arguments for this function are:

`power.exp(rm`5,df.num=1, df.denom=10, delta=1, sigma=1, alpha=0.05, print=FALSE)=

`rm:` the number of observations used in computing a mean.

`df.num:` the degrees of freedom of the numerator of the F for testing the term involving the means;

`df.denom:` the degrees of freedom of the denominator of the F for testing the term involving the means;

`delta:` the true difference between a pair of means;

`sigma:` population standard deviation;

`alpha:` the significance level to be used;

`print:` T or F to have or not have a table of power calculation details printed out.

Note that the values given for the arguments in the above expression for power.exp are the default values assigned to the arguments if they are not set in a call to the function.

5.2 Example: Penicillin yield

Consider the penecillin example taken from Box et al. (2005). Suppose it was expected that the minimum difference between a pair of treatment means is 5 and that $\alpha = 0.05$. In the analysis of variance for this experiment, the Residual MSq was 18.83 so we will take $\sigma^2 \approx 20$. Also, $r = 5$ and $m = 1$. The `power.exp` call to compute the power, and the resulting output, is given below. Note that alpha is not set in this call and so the default value of 0.05 will be used. Also, the expressions `3 * (rm - 1)` and `sqrt(20)` will be evaluated prior to the call to the function. To get the correct value of `rm` used in evaluating the expression `3 * (rm - 1)`, `rm` needs to be set prior to calling `power.exp`.

```
rm <- 5
power.exp(rm=rm, df.num=3, df.denom=3*(rm-1), delta=5, sigma=sqrt(20), print=TRUE)

##   rm df.num df.denom alpha delta    sigma lambda      powr
## 1  5      3       12  0.05     5 4.472136  3.125 0.2159032
## [1] 0.2159032
```

That is, the power of the experiment is just over 0.2.

5.3 Computing the sample size to achieve specified power

The function no.reps from the dae library is used to compute the required number of pure replicates, r, of the treatments in a designed experiment to achieve a specified power in detecting a difference between the means for some, not necessarily proper, subset of the treatment factors. The usage and arguments for this function are as follows:

```
no.reps(multiple=1, df.num=1,
        df.denom=expression((df.num+1)*(r-1)),
        delta=1, sigma=1, alpha=0.05, power =0.8,
        tol = 0.025, print=FALSE)
```

`multiple`: the multiplier, m, which when multiplied by the number of pure replicates of a treatment, r, gives the number of observations (rm) used in computing means for the treatment factor subset; m is the replication arising from other treatment factors. However, for single treatment factor experiments the subset can only be the treatment factor and $m = 1$;

`df.num`: the degrees of freedom of the numerator of the F for testing the term involving the treatment factor subset;

`df.denom`: an expression for the degrees of freedom of the denominator of the F for testing the term involving the treatment factor subset it must involve r, the number of pure replicates, can involve other arguments to `no.reps` such as `multiple` and `df.num`, and must be enclosed in an expression function so that it is not evaluated when `no.reps` is called but will be evaluated as different values of r are tried during execution of `no.reps`;

`delta`: the true difference between a pair of means for some, not necessarily proper, subset of the treatment factors;

`sigma`: population standard deviation;

`alpha`: the significance level to be used;

`power`: the minimum power to be achieved;

`tol`: the maximum difference tolerated between the power required and the power computed in determining the number of replicates;

`print`: T or F to have or not have a table of power calculation details printed out.

5.4 Example II.1.Penicillin yield (continued)

We now determine the number of replicates required to achieve a power of 0.80 in detecting $\Delta = 5$ with $\alpha = 0.05$. We continue to take $\sigma^2 \approx 20$. The use of `no.reps` and the resulting output is as follows:

```
no.reps(multiple=1, df.num=3, df.denom=expression(df.num*(r-1)), delta=5,
        sigma=sqrt(20), power=0.8, print=FALSE)
```

```
## $nreps
## [1] 19
##
## $power
## [1] 0.8055926
```

That is, 19 reps will achieve a little more than the required power of 0.80.

References

Bailey, R. A. (1992) Efficient semi-latin squares. *Statistica Sinica*, *2*, 413–437.

Bailey, R. A. (2008) *Design of Comparative Experiments*. Cambridge: Cambridge University Press.

Bailey, R. A. and C. J. Brien (2015) Randomization-based models for multitiered experiments: I. A chain of randomizations. accepted for the Annals of Statistics.

Box, G. E. P., W. G. Hunter, and J. S. Hunter (2005) *Statistics for Experimenters*. (2nd ed.) New York: Wiley.

Brien, C. J. (2018) *dae: functions useful in the design and ANOVA of experiments*. URL http://CRAN.R-project.org/package=dae, (R package version 3.0-23, accessed March 16, 2019).

Brien, C. J. and R. A. Bailey (2006) Multiple randomizations (with discussion). *Journal of the Royal Statistical Society, Series B (Statistical Methodology)*, *68*, 571–609.

Brien, C. J. and R. A. Bailey (2009) Decomposition tables for experiments. I. A chain of randomizations. *The Annals of Statistics*,. accepted for publication.

Brien, C. J., B. Berger, H. Rabie, and M. Tester (2013) Accounting for variation in designing greenhouse experiments with special reference to greenhouses containing plants on conveyor systems. *Plant Methods*, *9*, 5. Available from http://www.plantmethods.com/content/9/1/5.

Brien, C. J. and C. G. B. Demétrio (2009) Formulating mixed models for experiments, including longitudinal experiments. *The Journal of Agricultural, Biological and Environmental Statistics*, *14*, 253–280.

Brien, C. J., B. D. Harch, R. L. Correll, and R. A. Bailey (2011) Multiphase experiments with at least one later laboratory phase. I. Orthogonal designs. *Journal of Agricultural, Biological and Environmental Statistics*, *16*, 422–450.

Brien, C. J. and R. W. Payne (1999) Tiers, structure formulae and the analysis of complicated experiments. *The Statistician*, *48*, 41–52.

Coombes, N. E. (2009) *Digger: design search tool in R*. URL: http://www.austatgen.org/files/software/downloads/, (accessed June 3, 2015).

Joshi, D. D. (1987) *Linear Estimation and Design of Experiments*. New Delhi: Wiley Eastern.

McIntyre, G. A. (1955) Design and analysis of two phase experiments. *Biometrics*, *11*, 324–334.

R Core Team (2019) *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing. URL: <http://www.r-project.org>.

Williams, E. R., A. C. Matheson, and C. E. Harwood (2002) *Experimental Design and Analysis for Tree Improvement*. (2nd ed.) Melbourne, Australia: CSIRO.

Yates, F. (1937) The design and analysis of factorial experiments. *Imperial Bureau of Soil Science Technical Communication*, *35*.