Identifying, randomizing, canonically analyzing and formulating mixed models for designs for comparative experiments using **R**

C. J. Brien and S. Rogers

November 16, 2019

Biometrics by the Botanic Gardens, Adelaide, December 2019

This document describes how to use functions from the `R` (R Core Team, 2019) packages `dae` (Brien, 2019) and `od` (Butler, 2019) to produce layouts for experiments and to check some of their properties.

Contents

1 Installed software

The following software should be installed on your computer:

- `R` (3.5.x or later preferable)
- RStudio
- Packages (you can check the version using the `packageVersion` function.)
  - dae (Version 3.1-16 or later from CRAN or `http://chris.brien.name/rpackages`)
  - od (Version 2.0.0 from `http://mmade.org`)

2 Programme

09:00–10:00: Concepts in experimental design: Experiment description, randomization by permutation based on the nesting an crossing, canonical analysis of a design and formulating allocation-based mixed models for orthogonal designs, including those with multiple errors.

10:00–10:45: Orthogonal experimental design in `R`: participants use `dae` to generate orthogonal designs for experiments and apply the concepts learned in the previous session.

11:15–12:15: Nonorthogonal experimental design: Using the concepts in the context of balanced and unbalanced experiments; canonical efficiency factors and the alphabet of efficiency measures; the effects of covariates, missing observations, systematic allocation and pseudoreplication.

12:15–13:00 Nonorthogonal experimental design in `R`: using `dae` and `od` to produce nonorthogonal designs for experiments.

13:00–13:45 Lunch

13:45–14:15 Nonorthogonal experimental design in `R` (continued)

14:15–15:15 Advanced experimental design: multiphase and p-rep designs.

15:45–17:00 Using `R` for advanced experimental design: more practice in using `dae` and `od` to construct experimental designs.

3 Packages and the functions to be used

3.1 dae

The package `dae` provides functions useful in the design and anova of experiments (Brien, 2019). There are 84 functions that fall into the following categories and those that will be used in this course are described:

1. Data

   BIBDWheat.dat Data for a balanced incomplete block experiment.

   Cabinet1.des A design for one of the growth cabinets in an experiment with 50 lines and 4 harvests.

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

   Exp249.munit.des Systematic, main-unit 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: Generate all combinations of several factors and, optionally, replicate them.

   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

   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.

   designBlocksGGPlot: Adds block boundaries to a plot produced by designGGPlot.

   designGGPlot: A graphical representation of an experimental design based on labels stored in a data.frame 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.

   efficiencies.pcanon: Extracts the canonical efficiency factors from a pcanon.object produced by designAnatomy.

4. ANOVA functions

5. Matrix functions

6. Projector and canonical efficiency functions

7. Miscellaneous functions.

3.2 od

The package `od` generates optimal experimental designs (Butler, 2019). It does this based on an *anticipated* mixed model and obtains a design that minimizes the average variance of pairwise differences (AVPD). It has 16 functions; those that will be used in this course are as follows:

od: Generates optimal designs for comparative experiments under a general linear mixed model.

od.options: Sets or displays various options that affect the behaviour of od.

Documentation for each of these functions is available from the user manual for the relevant package. In general this can be found in the `doc` subdirectory of the directory in which the package is installed or from the `help` for the function once the package has been installed. For the latter, to see the manual for package `foo`, enter `help(package="foo")` and click on the link `User guides, package vignettes and other documentation`.

For `dae`, the manual is available via `vignette("dae-manual", package="dae")` and there are some notes that show how to use the functions that are available via `vignette("DesignNotes", package="dae")`.

4 Notation used for mixed models

The general form for a mixed model is:

$$\mathbf{Y} = \mathbf{X}\boldsymbol{\beta} + \mathbf{Z}\mathbf{u} + \mathbf{e},$$

where $\boldsymbol{\beta}$ is the vector of fixed parameters, $\mathbf{u}$ is the vector of random effects, and $\mathbf{e}$ is the vector of residuals corresponding to each observation. The matrices $\mathbf{X}$ and $\mathbf{Z}$ are the design matrices for the fixed and random effects, respectively. Generally, $\mathbf{X}$ and $\boldsymbol{\beta}$ are conformably partitioned so that there is a separate submatrix and subvector for each fixed term. Similarly, $\mathbf{Z}$ and $\mathbf{u}$ are conformably partitioned according to the random terms.

A mixed model is expressed in symbolic form by list of the fixed terms, followed by a '|', and then a list of the random terms. Terms contributing to the residual variation are underlined.

5 Session 1: Orthogonal experimental design in R

This class of experiments covers the orthogonal standard or textbook experiments, those that 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, an allocated set that is allocated to a recipient set. These two sets are also referred to as the unit and treatment factors, respectively.

Firstly, initialize by loading the `dae` library. Also check the version that is loaded.

```
library(dae)
packageVersion("dae")
```

5.1 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.

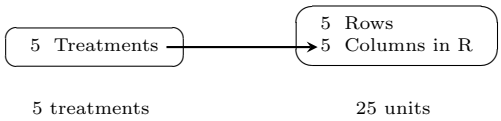

Figure 1: Factor-allocation diagram for an RCBD: treatments are allocated to units; the arrow indicates that the factor Treatments is randomized to Columns; Columns in R indicates that the Columns are considered to be nested within Rows for this randomization; R = Rows.

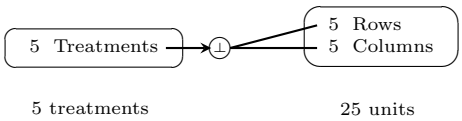

Figure 2: Factor-allocation diagram for an LSD: treatments are allocated to units; the arrow indicates that the allocation is randomized; the '⊥' at the end of the arrow indicates that an orthogonal design is used; the two lines from '⊥' indicates that the Treatments are allocated to the combinations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

5.1.1 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) `recipient`, (iii) `nested.recipients`, and optionally, (iv) `seed`. The allocated factors are also referred to as treatment factors and the recipient factors as block or unit factors. A systematic arrangement of the allocated factors, corresponding to the values of the recipient factors, needs to be supplied and there are a number of ways of doing this.

Our general approach is to set up a systematic design in a `data.frame` to separate this aspect of constructing a design from the randomizing of a design. The naming convention used is that the name of the `data.frame` ends in `.sys`. This `data.frame` should contain the values of both the recipient and the allocated `factors`, the latter in a systematic order that is appropriate for the design. The `dae` function `fac.gen` will be used to generate the values of the recipient `factors` in standard order and often will also be used to generate the values of the allocated `factors`.

Then the allocated and recipient factors are supplied to `designRandomize` by subsetting the columns of the `data.frames` to just the appropriate factors for each argument. Note that the Treatments could also be supplied as a factor and the recipient factors can be specified directly to the `recipient` argument as a `list`, e.g. `list(Rows=b, Columns=t)`. A `data.frame` containing the recipient and randomized allocated factors is produced and, in these notes, the name for the `data.frame` with the randomized layout will end in `.lay`.

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.

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:

```
b <- 5
t <- 5
#'## Set up a systematic design
RCBD.sys <- cbind(fac.gen(generate = list(Rows=b, Columns=t)),
                  fac.gen(generate = list(Treatments = LETTERS[1:t]),
                          times = b))
#'## Obtain the layout
RCBD.lay <- designRandomize(allocated         = RCBD.sys["Treatments"],
                            recipient         = RCBD.sys[c("Rows", "Columns")],
                            nested.recipients = list(Columns = "Rows"),
                            seed              = 1134)

#'## Output the layout
RCBD.lay
#'## Plot the layout
designGGPlot(RCBD.lay, labels = "Treatments", cellalpha = 0.75,
             blockdefinition = cbind(1,t))
```

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. The `seed` is specified to ensure that the same design is produced whenever `designRandomize` is run with these arguments.

5.1.2 Produce the randomized layout for an LSD

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

```
b <- 5
t <- 5
#'## Set up a systematic design
LSD.sys <- cbind(fac.gen(list(Rows=b, Columns=t)),
                 Treatments = factor(designLatinSqrSys(t), labels = LETTERS[1:t]))
#'## Obtain the layout
LSD.lay <- designRandomize(allocated = LSD.sys["Treatments"],
                           recipient = LSD.sys[c("Rows", "Columns")],
                           seed      = 141)
#'## Output the layout
LSD.lay
#'## Plot the layout
designGGPlot(LSD.lay, labels = "Treatments", cellalpha = 0.75,
             blockdefinition = cbind(1,1))
```

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. The `seed` is specified to ensure that the same design is produced whenever `designRandomize` is run with these arguments.

5.1.3 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` for `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 anatomies (skeleton anova tables) for the RCBD and LSD that have been obtained. Note that the 'Mean' source has been omitted from these tables, but can be included using `grandMean = TRUE` when calling `designAnatomy`.

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

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

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

5.1.4 Questions

1. What is the advantage of specifying a `seed` in `designRandomize`?
2. With what unit source is Treatments confounded in these designs and what is the difference in the interpretation of these sources?
3. What would determine which of these two designs is used for a particular experiment?

5.2 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.

5.2.1 Produce the randomized experimental layout

Use `fac.gen` to obtain a systematic layout and then `designRandomize` to obtain a randomized layout for this experiment. Check the properties of the design, as illustrated in the following `R` code:

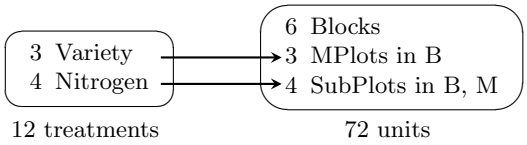

Figure 3: Factor-allocation diagram for a split-plot design: treatments are allocated to units; the arrows indicates that the factors Variety and Nitrogen are randomized to MPlots and Subplots, respectively; MPlots in B indicates that the MPlots are considered to be nested within Blocks for this randomization;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

```
Oats.sys <- cbind(fac.gen(list(Blocks=6, MPlots=3, SubPlots=4)),
                  fac.gen(list(Variety=c("Victory","Golden Rain", "Marvellous"),
                               Nitrogen=c(0,0.2,0.4,0.6)), times=6))
Oats.lay <- designRandomize(allocated         = Oats.sys[c("Variety", "Nitrogen")],
                            recipient         = Oats.sys[c("Blocks", "MPlots", "SubPlots")],
                            nested.recipients = list(MPlots = "Blocks",
                                                     SubPlots = c("MPlots", "Blocks")),
                            seed              = 235805)
#'## Plot design produced, first combining Variety and Nitrogen so plot on 2 lines per cell
Oats.lay$Treatments <- with(Oats.lay, fac.combine(list(Variety, Nitrogen),
                                                  combine.levels = TRUE, sep = "\n"))
designGGPlot(Oats.lay, labels = "Treatments",
             row.factors = c("Blocks", "MPlots"), column.factors = "SubPlots",
             cellfillcolour.column = "Variety", cellalpha = 0.75,
             blockdefinition = c(1,4))

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

5.2.2 Questions

1. In what sense does this design involve a single randomization?
2. What is the initial allocated mixed model for this design? Is it equivalent to a randomization model?
3. 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?

5.3 Split-unit design from Mead (1990)

Mead (1990, Example 14.1) describes an experiment to investigate the effects of grazing patterns on pasture composition. It is available in `dae` as `SPLGrass.dat`.

The design for the experiment is a split-unit design. The main units are arranged in 3 Rows $\times$ 3 Columns. Each main unit is split into 2 SubRows x 2 SubColumns.

The factor Period, with levels 3, 9 and 18 days, is assigned to the main units using a 3×3 Latin square. The two-level factors Spring and Summer are assigned to split-units using a criss-cross design that is randomized within each main unit. The levels of each of Spring and Summer are two different grazing patterns in its season. The response variable is `Main.Grass`.

Use `data(SPLGrass.dat)` to load the design (and the data) and then investigate the properties of the design using `designAnatomy`.

5.3.1 Questions

1. Describe the confounding that is inherent in this design.
2. Draw a factor-allocation diagram for this experiment.
3. What is the initial allocated mixed model for this design?

5.4 A design for the petrol additives experiment

Box et al. (2005, Section 4.4) describes a car emission experiment that investigates 4 additives. It involves 4 cars being driven by 4 drivers. Here we investigate increasing the replication by repeating the experiment on two occasions. Suppose that the 4 cars differ between occasions.

In a `data.frame` called `LSRepeat.sys`, generate a systematic design using two 4×4 Latin squares for allocating the 4 Additives to the 32 tests, being the combinations of the 2 Occasions x 4 Drivers x 4 Cars. Make sure that a Latin square is used for each Occasion.

Now a comparison is made of two different ways of randomizing this design. Firstly, we retain the factors Occasions, Drivers and Cars from the systematic design. The factor-allocation diagram is in Figure 4.

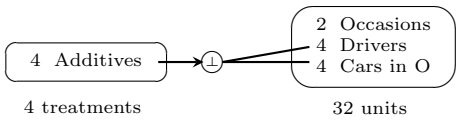

Figure 4: Factor-allocation diagram for repeated LSDs: treatments are allocated to units; the arrow indicates that the allocation is randomized; the '⊥' at the end of the arrow indicates that an orthogonal design is used; the two lines from '⊥' indicates that the Additives are allocated to the combinations of Drivers and Cars within Occasions using the design.

```
#'## Obtain a randomized layout with Cars nested within Occasions
LSRepeat2b.lay <- designRandomize(allocated         = LSRepeat.sys["Additives"],
                                  recipient         = LSRepeat.sys[c("Occasions", "Drivers",
                                                                     "Cars")],
                                  nested.recipients = list(Cars="Occasions"),
                                  seed              = 194)

#'## Plot the layout
designGGPlot(LSRepeat2b.lay, row.factors = "Cars", column.factors = c("Occasions", "Drivers"),
             labels = "Additives", cellalpha = 0.75, blockdefinition = cbind(4,4))

#'## Get the anatomy of the layout
LSRepeat2b.canon <- designAnatomy(formulae = list(unit = ~ (Occasions/Cars)*Drivers,
                                                  trt  = ~ Additives),
                                  data     = LSRepeat2b.lay)
summary(LSRepeat2b.canon)
```

Secondly, we use only Drivers and Cars to do the randomization, but still attempt to include Occasions in the analysis. The new factor-allocation diagram is in Figure 5.

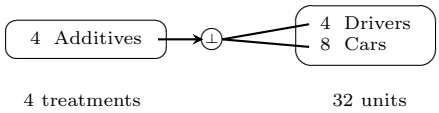

Figure 5: Factor-allocation diagram for repeated LSDs: treatments are allocated to units; the arrow indicates that the allocation is randomized; the '⊥' at the end of the arrow indicates that an orthogonal design is used; the two lines from '⊥' indicates that the Additives are allocated to the combinations of Drivers and Cars using the design.

```
#'## Obtain a randomized layout
LSRepeat.D8.sys <- LSRepeat.sys
LSRepeat.D8.sys$Cars <- with(LSRepeat.D8.sys, fac.combine(list(Occasions, Cars)))
LSRepeat.D8.sys <- with(LSRepeat.D8.sys, LSRepeat.D8.sys[order(Drivers,Cars),])
LSRepeat2b.D8.lay <- designRandomize(allocated = LSRepeat.D8.sys["Additives"],
                                     recipient  = LSRepeat.D8.sys[c("Drivers", "Cars")],
                                     seed       = 149)

#'## Plot the layout
designGGPlot(LSRepeat2b.D8.lay, row.factors = "Drivers", column.factors = "Cars",
             labels = "Additives", cellfillcolour.column = "Additives",
             cellalpha = 0.75, blockdefinition = cbind(4,8))

#'## Get the anatomy of the layout
LSRepeat2.D8.canon <- designAnatomy(formulae = list(unit = ~ Drivers*Cars,
                                                    trt  = ~ Additives),
                                    data     = LSRepeat2b.D8.lay)
summary(LSRepeat2.D8.canon)

#'## Add Occasions to the analysis
LSRepeat2b.D8.lay$Occasions <- fac.recode(LSRepeat2b.D8.lay$Cars, rep(1:2, each=4))
LSRepeat2b.D8.lay
LSRepeat2b.D8.canon <- designAnatomy(formulae = list(unit = ~ (Occasions + Cars)*Drivers,
                                                     trt  = ~ Additives),
                                     data      = LSRepeat2b.D8.lay)
summary(LSRepeat2b.D8.canon)
```

5.4.1 Questions

1. The Residual degrees of freedom for a single 4×4 Latin square are 6. Has the use of two 4×4 Latin squares had the desired effect of increasing the Residual df? What other advantage does the use of two Latin squares have over the use of a single Latin square?
2. What is the difference between the two randomizations?
3. How do the two anatomies that include Occasions differ?
4. What effect does including Occasions#Drivers have on the anatomy?

6 Session 2: Nonorthogonal experimental design in R

This class of experiments covers the nonorthogonal standard or textbook experiments and these experiments must be single phase because they involve a single randomization.

Firstly, initialize by loading the libraries that will be used and setting the output width.

```
library(dae, quietly = TRUE)
library(od)
packageVersion("od")
options(width=100)
```

6.1 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	1	2	3	4	5	1	2	3	4	5	1	2	3	4	5
2	6	7	8	9	10	7	8	9	10	6	8	9	10	6	7
3	11	12	13	14	15	13	14	15	11	12	15	11	12	13	14
4	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 6.

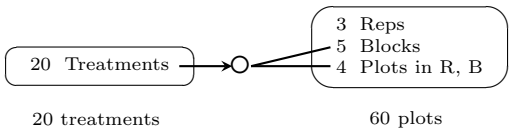

Figure 6: Factor-allocation diagram for the alpha design: treatments are allocated to units; the arrow indicates that the allocation is randomized; the 'O' at the end of the arrow indicates that a nonorthogonal design is used; the two lines from 'O' indicate that the Treatments are allocated to the combinations 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.

6.1.1 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.

```
#'## Set up the systematic design
# Note that Treatments has been entered by rows within a replicate
alpha.sys <- cbind(fac.gen(list(Reps=3, Plots=4, Blocks=5)),
                   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)))
#'## Obtain the layout
alpha.lay <- designRandomize(allocated         = alpha.sys["Treats"],
```

```
                              recipient         = alpha.sys[c("Reps", "Plots", "Blocks")],
                              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")
```

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 Plots[Reps:Blocks], 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 (A), that is, to 0.7447. In particular, $AVPD = 2/(rA)$.

```
AVPD <- designAmeasures(mat.Vpredicts(target = ~ Treats - 1,
                                      fixed  = ~ Reps/Blocks,
                                      design = alpha.lay))[[1]]
Aeff <- summary(alpha.canon, which.criteria = "aeff")$decomp$aefficiency[3]
(measures <- c(AVPD, Aeff, 2/(3*Aeff)))
```

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

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

6.1.2 Questions

1. What is the randomization-based mixed model for this experiment?
2. In a mixed-model analysis, which unit terms might you fit as fixed terms? Why?

6.2 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 7.

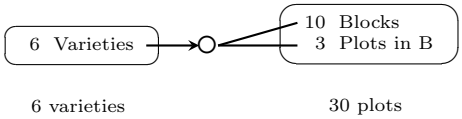

Figure 7: Factor-allocation diagram for the balanced incomplete-block design: treatments are allocated to units; the arrow indicates that the allocation is randomized; the '○' at the end of the arrow indicates that a nonorthogonal design is used; the two lines from '○' indicates that the Varieties are 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.

6.2.1 Load the design and check its 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)
```

From this it is clear that 80% of the information about Varieties is available from the Plots[Blocks] 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.

Calculate the AVPD and check that $AVPD = 2/(rA)$

```
AVPD <- designAmeasures(mat.Vpredicts(target = ~ Varieties - 1,
                                      fixed  = ~ Blocks,
                                      design = BIBDWheat.dat))[[1]]
Aeff <- summary(bibdwheat.canon, which.criteria = "aeff")$decomp$aefficiency[3]
(measures <- c(AVPD, Aeff, 2/(5*Aeff)))
```

6.2.2 Questions

1. What is the value of xefficiency for Varieties when confounded with Plots[Blocks] for this design? Why?
2. How many nonzero eigenvalues does $\mathbf{Q}_{\mathrm{V}}\mathbf{Q}_{\mathrm{BP}}\mathbf{Q}_{\mathrm{V}}$ have?

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

Williams et al. (2002, p.144) provide an example of a tree experiment that 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. The design used was a split-unit design comprised of four rectangles each of six rows by ten columns; the rectangles are located next to each other so that they are contiguous along the rows. The two inoculation times were randomized to the rectangles (main units). The provenances were randomized to the subunits using a resolved, latinized, row-column design, the rectangles forming replicates of the Provenances. The latinization was by columns and was necessary because differences between Columns

(across Reps) was anticipated; it served to avoid multiple occurrences of a provenance in a column. At 30 months, diameter at breast height (Dbh) was measured.

The factor-allocation diagram for the experiment is in Figure 8.

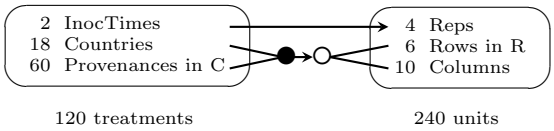

Figure 8: Factor-allocation diagram for the row-and-column design: treatments are allocated to units; the arrows indicates that the allocations are randomized; the two lines leading to the '●' indicate that it is is the combinations of Countries and Provenances that is allocated; the '○' at the end of the lower arrow indicates that a nonorthogonal design is used; the two lines from '○' 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 Reps for this randomization; R = Reps.

6.3.1 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
load("Casuarina.dat.rda")
#'## Check the properties of the design
Casuarina.canon <- designAnatomy(formulae = list(units = ~ (Reps/Rows)*Columns,
                                                 trts  = ~ InocTime*(Countries+Provenances)),
                                 data       = Casuarina.dat)
summary(Casuarina.canon, which = c("aeff", "eeff", "order", "dforth"))
```

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.

6.3.2 Questions

1. What is it about the design that makes it resolved for Provenances?
2. What is the disadvantage of allocating InocTimes to Reps?

6.4 A resolved design for the wheat experiment that is near-A-optimal under a mixed model

Gilmour et al. (1995) provides an example of a wheat experiment for 25 Varieties in which a balanced lattice design was employed.

The factor-allocation diagram for the experiment is in Figure 9.

In the lectures it was stated that, while the design is optimal for a fixed model, it is not optimal for a mixed model. In this exercise, a search will be made for a resolved design that is near-A-optimal under a mixed model.

6.4.1 Input the design and check the properties of the design

Use the following `R` code to input the design, plot it and check its properties.

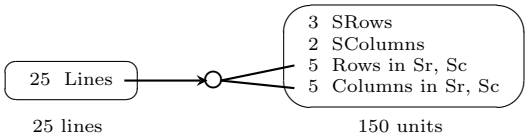

Figure 9: Factor-allocation diagram for the row-and-column design: treatments are allocated to units; the arrows indicates that the allocations are randomized; the 'O' at the end of the lower arrow indicates that a nonorthogonal design is used; the two lines from 'O' indicates that the Lines are allocated to the combinations of Rows and Columns using the design; Rows (Columns) in Sr, Sc indicates that the Rows (Columns) are considered to be nested within SRows and SColumns for this randomization; Sr = S(uper)Rows; Sc = S(uper)Columns.

```
#'## Get the design
library(asremlPlus)
data(Wheat.dat)
latt.lay <- cbind(fac.gen(list(ARows = 10, AColumns = 15)),
                  fac.gen(list(SRows = 2, Rows = 5, SColumns = 3, Columns =5)),
                  Wheat.dat["Variety"])
#'## Plot the design
#+ "LattDesign"
library(scales)
cell.colours <- hue_pal()(25)
designGGPlot(latt.lay, labels = "Variety",
             row.factors = c("SRows", "Rows"), column.factors = c("SColumns", "Columns"),
             colour.values = cell.colours, cellalpha = 0.75, size = 6,
             blockdefinition = cbind(5,5))
#'## Check the properties of the design
latt.canon <- designAnatomy(formulae = list(units = ~ (SRows:SColumns)/(Rows*Columns),
                                            trt   = ~ Variety),
                            data     = latt.lay)
summary(latt.canon, which.criteria = c("aeff", "order"))
```

6.4.2 Search for a near-A-optimal design

Use od to search for a near-A-optimal design under a mixed model. In this case the "tabu+rw" search method is to be used. Further, the od options are to be set to values that I have found by trail-and-error to be successful. The options are

P: the probability of accepting a non-improving design; the default is P=0.005.

localSearch: the number of steps in the random walk local search strategy of the "tabu+rw" search option; the default is 10000.

tabuStop: if the number of consecutive tabu loops with no change in the objective function exceeds tabuStop, then tabu optimization terminates (the default is 4).

```
#'## Set od options
maxit <- 50
search <- "tabu+rw"
od.options(P = 0.10, localSearch = 10000, tabuStop = 100)
#'## Set up the values of the variance components and autocorrelation for the random terms
params <- c(2.5, 1, 0.1, 0.1, 0.5, 1, 0.6, 0.4)
names(params) <- c("g.sRR", "g.sCC", "g.sRsCR", "g.sRsCC", "g.u", "g.aRaC", "rho.R", "rho.C")
#'### Set the values in od
```

```
Wheat.start <- od(fixed    = ~ SRows*SColumns + Variety,
                  random   = ~ SRows:Rows + SColumns:Columns +
                                  SRows:SColumns:(Rows + Columns) + units,
                  residual = ~ ar1(ARows):ar1(AColumns),
                  permute  = ~ Variety, swap = ~ SRows:SColumns,
                  data     = latt.lay, start.values = TRUE)
vp.table <- Wheat.start$vparameters.table
vp.table$Value <- params
print(vp.table)
#'### Generate the near-A-optimal design
Wheat.od <- od(fixed    = ~ SRows*SColumns + Variety,
               random   = ~ SRows:Rows + SColumns:Columns +
                            SRows:SColumns:(Rows + Columns) + units,
               residual = ~ ar1(ARows):ar1(AColumns),
               permute  = ~ Variety, swap = ~ SRows:SColumns,
               G.param  = vp.table, R.param = vp.table,
               maxit    = maxit, search = search,
               data     = latt.lay)
Wheat.lay <- Wheat.od$design
Wheat.lay$unit <- factor(1:nrow(Wheat.lay))
```

6.4.3 Checking the properties of the designs

Now calculate the A-measure for the original lattice design and the near-optimal design produce by `od`. Also, produce the anatomy for the near-optimal design.

```
#'### Calculate the A-measure for the lattice design under a mixed model
latt.lay$unit <- factor(1:nrow(latt.lay))
(A.latt <- designAmeasures(mat.Vpredicts(target = ~ Variety - 1,
                                         fixed  = ~ SRows*SColumns - 1,
                                         random = ~ SRows:Rows + SColumns:Columns +
                                                    SRows:SColumns:(Rows + Columns) + unit - 1,
                                         G      = as.list(params[1:5]),
                                         R      = kronecker(mat.ar1(params["rho.R"], 10),
                                                            mat.ar1(params["rho.C"], 15)),
                                         design = latt.lay))[[1]])
#'### Check the A-value for the near-optimal design
(A.wht <- designAmeasures(mat.Vpredicts(target = ~ Variety - 1,
                                        fixed  = ~ SRows*SColumns - 1,
                                        random = ~ SRows:Rows + SColumns:Columns +
                                                   SRows:SColumns:(Rows + Columns) + unit - 1,
                                        G      = as.list(params[1:5]),
                                        R      = kronecker(mat.ar1(params["rho.R"], 10),
                                                           mat.ar1(params["rho.C"], 15)),
                                        design = Wheat.lay))[[1]])
(A.wht/A.latt)
#'## Check the properties of the design
Wheat.canon <- designAnatomy(formulae = list(unit = ~ (SRows:SColumns)/(Rows*Columns),
                                             trt  = ~ Variety),
                             data     = Wheat.lay)
summary(Wheat.canon, which.criteria = c("aeff", "meff", "xeff", "eeff", "order"))
```

6.4.4 Questions

1. How do the AVPD values calculated by `od` and those calculated using `designAmeasures` and `mat.Vpredicts` compare?

2. Summarize the differences between the original balanced lattice design and the `od` design. Is the increased precision of the `od` design worthwhile?

6.5 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 10.

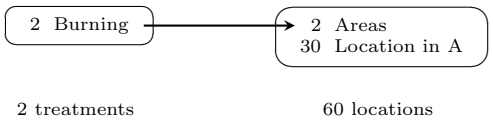

Figure 10: Factor-allocation diagram for the environmental experiment: treatments are allocated to locations; the arrow indicates that the factor Burning is randomized to Areas; Locations in A indicates that the Locations are considered to be nested within Areas; A = Areas.

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

```
#'## Obtain the layout
Burn.sys <- cbind(fac.gen(list(Areas=2, Locations=30)),
                  Burn = factor(rep(c("Burn", "NoBurn"), each=30)))
Burn.lay <- designRandomize(allocated         = Burn.sys["Burn"],
                            recipient         = Burn.sys[c("Areas", "Locations")],
                            nested.recipients = list(Locations = "Areas"),
                            seed              = 872159)

#'## plot the design
designGGPlot(Burn.lay, labels = "Burn", row.factors = "Locations", column.factors = "Areas")

#'## Check its properties
Burn.canon <- designAnatomy(formulae = list(unit = ~Areas/Locations,
                                            trt  = ~Burn),
                            data     = Burn.lay)
summary(Burn.canon)
```

6.5.1 Questions

1. How is the pseudo-replication involved in this experiment manifested in the skeleton anova table?

2. The randomization-based mixed model for the experiment is Burn | Areas + Areas:Locations. 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 of convenience?

7 Session 3: Using R for advanced experimental design

Firstly, initialize by loading the libraries that will be used and setting the output width.

```
library(dae)
library(od)
options(width=100)
```

7.1 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 two different multiphase scenarios are considered, both being based on a first-phase that is the testing phase and employs a split-unit design.

7.1.1 A standard single-phase athlete training experiment

First, a split-unit design is generated for an experiment in which the performance of an athlete when subject to nine different training conditions is tested. 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 factor-allocation diagram is shown in Figure 11. Generate a randomized layout for the experiment.

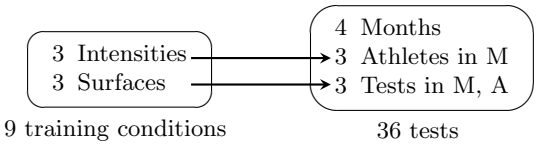

Figure 11: Factor-allocation diagram for the standard athlete training experiment: training conditions are randomized to tests; the two left-hand arrows indicate that the levels of Intensities and Surfaces are randomized to Athletes and Tests, respectively; M = Months; A = Athletes.

```
#'## Phase 1: Construct a systematic layout and generate a randomized layout for the first phase
split.sys <- cbind(fac.gen(list(Months = 4, Athletes = 3, Tests = 3)),
                   fac.gen(list(Intensities = LETTERS[1:3], Surfaces = 3),
                           times = 4))
split.lay <- designRandomize(allocated         = split.sys[c("Intensities", "Surfaces")],
                             recipient         = split.sys[c("Months", "Athletes", "Tests")],
                             nested.recipients = list(Athletes = "Months",
                                                      Tests = c("Months", "Athletes")),
                             seed = 2598)

#'## Plot the design
#+ "SplitDes_v2"
split.lay <- within(split.lay,
                    Conditions <- fac.combine(list(Intensities, Surfaces),
                                              combine.levels = TRUE))
plt <- designGGPlot(split.lay, labels = "Conditions",
                    row.factors = "Tests", column.factors = c("Months", "Athletes"),
```

```
                    cellalpha = 0.75, size = 6,
                    blockdefinition = rbind(c(3,1)), blocklinecolour = "darkgreen",
                    printPlot = FALSE)
designBlocksGGPlot(plt, nrows = 3, ncolumns = 3, blockdefinition = rbind(c(3,3)))

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

Question

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

7.1.2 A simple two-phase athlete training experiment

Multiphase experiments differ from those previously presented in that they 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`.

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 factor-allocation diagram for this experiment is in Figure 12, 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.

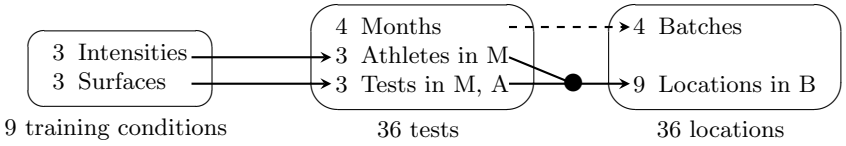

Figure 12: Factor-allocation diagram for the two-phase athlete training experiment: training conditions are randomized to tests and tests are allocated to locations; the two left-hand arrows indicate that the levels of Intensities and Surfaces are randomized to Athletes and Tests, respectively; the dashed arrow indicates that Months are systematically allocated to Batches; the '●' indicates that the combinations of the levels of Athletes and Tests are randomized to the Locations; M = Months; A = Athletes; B = Batches.

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
eg1.lay <- designRandomize(allocated         = split.lay,
                           recipient         = list(Batches = 4, Locations = 9),
                           nested.recipients = list(Locations = "Batches"),
```

```
                            except            = "Batches",
                            seed              = 71230)
eg1.lay

#'## Plot the layout
#+ Athlete_eg1lay
eg1.lay$Conditions <- with(eg1.lay, fac.combine(list(Intensities, Surfaces),
                                                combine=TRUE, sep=","))
designGGPlot(eg1.lay, labels = "Conditions",
             row.factors = "Locations", column.factors = "Batches",
             cellfillcolour.column = "Athletes", cellalpha = 0.75,  size = 6,
             title = "Randomized Intensities-Surfaces combinations",
             blockdefinition = rbind(c(9,1)),
             ggplotFuncs = list(xlab("Batches (Months)"),
                                theme(legend.position = "right")))
```

Check the properties of the design.

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

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?
2. Compare the units for the two phases in this experiment?
3. What are the outcomes for the two phases for this experiment?

7.1.3 Allowing for lab processing order in the athletic training example

Brien (2017) discusses a design, and its properties, that differs in the second phase from that described in Section 7.1.2: it assumes that lab processing order within a batch is important and so 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. That is, a suitable cross-phase design for allocating Intensities and Surfaces to Batches and Locations is needed. However, the second-phase design that allocates Months, Athletes and Tests to Batches and Locations has to be considered in that it must account for the split-unit nature of the first-phase design.

For the second-phase design, the Months are associated with Batches. Then each triple of consecutive locations in a batch are associated with a single athlete, one of those for the month associated with the batch. This leaves tests to be assigned to locations within triples. Thus, the cross-phase design will need to allocate efficiently an intensity to a location triple and surface to the locations within a triple.

The cross-phase design is a balanced factorial design (Hinkelmann and Kempthorne, 2005, Section 12.5) and can be constructed using two extended Latin squares (ELS) as follows:

1. a 3×4 ELS, formed from a 3×3 Latin square by repeating one of its columns, will be used to allocate Intensities to the 3 Locations triples $\times$ 4 Months.
2. A 3×4 ELS will be used to allocate Surfaces to the 3 Locations $\times$ 4 Months within a triple; the same ELS is used for the three triples.

3. To ensure no repeat Intensities-Surfaces combinations for a Location, the two Batches to which the repeated columns of the ELS for Intensities are assigned must be different from the two Batches to which repeated columns of the ELS for Surfaces are assigned.

The factor-allocation diagram, for this design, is in Figure 13. 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 randomizations involved have been called randomized-inclusive randomizations (Brien and Bailey, 2006) and are 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. A design with the same properties had been previously constructed by Rosemary Bailey (pers. comm.).

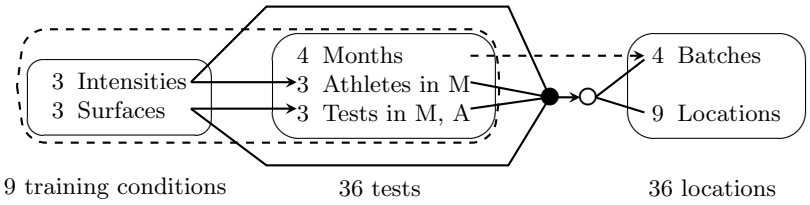

Figure 13: Factor-allocation diagram for the two-phase athlete training experiment with a row-column design for the second phase: training conditions are randomized to tests, then training conditions and tests are randomized to locations; the '●' indicates that the observed combinations of the levels of Intensities, Surfaces, Athletes and Tests are randomized to locations; the '○' indicates that a nonorthogonal design was used in this randomization to the combinations of the levels of Batches and Locations; the dashed arrow indicates that Months were systematically allocated to Batches; the dashed oval indicates that all factors from the first phase form a pseudotier and all are actively involved in determining the allocation to locations; M = Months and A = Athletes.

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

```
#'## Generate a systematic cross-phase design for Intensities and Surfaces
#' It is based on (i) an extended Latin square (ELS) for allocating Intensities to
#' Locations triples x Batches and (ii) the same ELS for each triple, the ELSD being used to
#' allocate Surfaces to the three Locations within each triple by four Batches.
#' The Batches to which the repeated columns of the ELSD for Intensities are assigned must be
#' different from the Batches to which repeated columns of the ELSD for Surfaces are assigned.
#+ Athlete_eg2sys_v3
eg2.phx.sys <- cbind(fac.gen(list(Batches = 4, Locations = 9)),
                     data.frame(Intensities = factor(rep(c(designLatinSqrSys(3), c(3,2,1)),
                                                         each = 3), labels = LETTERS[1:3]),
                                Surfaces = factor(c(rep(1:3, times = 3),
                                                    rep(1:3, times = 3),
                                                    rep(c(2,3,1), times = 3),
                                                    rep(c(3,1,2), times = 3)))))
eg2.phx.sys$Conditions <- with(eg2.phx.sys, fac.combine(list(Intensities, Surfaces),
                                                        combine.levels = TRUE))
designGGPlot(eg2.phx.sys, labels = "Conditions",
             row.factors = "Locations", column.factors = "Batches",
             cellfillcolour.column = "Intensities", cellalpha = 0.75, size = 6,
             title = "Intensities-Surfaces for systematic cross-phase design",
             blockdefinition = rbind(c(9,1)),
             ggplotFuncs = list(xlab("Batches (Months)"),
                                theme(legend.position = "right")))
```

```
#'## Second phase design
#'## Generate a systematic two-phase design by bringing in first-phase recipient factors
eg2.phx.sys$Months <- eg2.phx.sys$Batches
eg2.sys <- merge(split.lay, eg2.phx.sys) #merge on commmon factors Months, Intensities & Surfaces
eg2.sys  <- with(eg2.sys, eg2.sys[order(Batches,Locations),])
designGGPlot(eg2.sys, labels = "Conditions",
             row.factors = "Locations", column.factors = "Batches",
             cellfillcolour.column = "Athletes", cellalpha = 0.75,  size = 6,
             title = "Intensities-Surfaces for systematic two-phase design ",
             blockdefinition = rbind(c(9,1)),
             ggplotFuncs = list(xlab("Batches (Months)"),
                                theme(legend.position = "right")))
#'## Allocate the second phase
eg2.lay <- designRandomize(allocated = eg2.sys[c("Months", "Athletes", "Tests",
                                                 "Intensities", "Surfaces")],
                           recipient = eg2.sys[c("Batches", "Locations")],
                           except    = "Batches",
                           seed      = 243526)
head(eg2.lay)
#'## Plot the layout
#+ Athlete_eg2lay_v3
eg2.lay$Conditions <- with(eg2.lay, fac.combine(list(Intensities, Surfaces),
                                                combine=TRUE, sep=","))
designGGPlot(eg2.lay, labels = "Conditions",
             row.factors = "Locations", column.factors = "Batches",
             cellfillcolour.column = "Athletes", cellalpha = 0.75,  size = 6,
             title = "Randomized Intensities-Surfaces combinations",
             blockdefinition = rbind(c(9,1)),
             ggplotFuncs = list(xlab("Batches (Months)"),
                                theme(legend.position = "right")))
```

Check the properties of the design.

```
#'## 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 =c("aefficiency", "order"))
```

It is clear that Athletes[Months] and Tests[Months:Athletes] 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 skeleton anova for just the tests and locations tiers is obtained.

```
#"### Examine the nonorthogonality between locations and tests
eg2.locstests.canon <- designAnatomy(formulae = list(locs  = ~ Batches*Locations,
                                                     tests = ~ Months/Athletes/Tests),
                                     data     = eg2.lay)
summary(eg2.locstests.canon, which.criteria =c("aefficiency", "order"))
```

Questions

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

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

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

7.2 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 15.

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 16). Actually, this design is a semi-Latin square (Bailey, 1992).

The factor-allocation diagram for the experiment is in Figure 14. 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 factor-allocation diagram, I have assumed that randomization to NicPlant was within Sets and to Posn was across Sets. Similarly, I have assumed that randomization to DatPlant was within Lot and to AssPosn across Lot. In the factor-allocation diagram, N_1 is a factor for the pairs of tobacco plants formed by taking a plant from each set in the first phase.

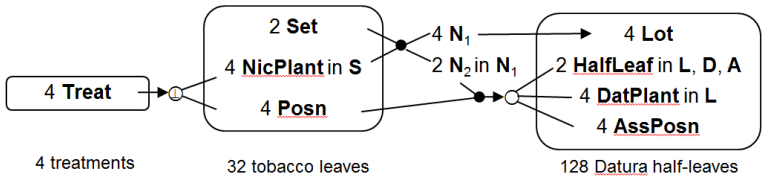

Figure 14: Factor-allocation diagram for McIntyre's (1955) two-phase experiment: treatments are randomized to tobacco leaves and tobacco leaves are randomized to Datura half-leaves; the arrow to the '⏚', the '⏚' and the two lines from the '⏚' indicate that Treat is randomized to the combinations of NicPlant and Posn using an orthogonal design; N_1 is a pseudofactor indexing the pairs of tobacco plants formed by taking a plant from each set in the first phase and N_2 is a pseudofactor indexing the tobacco plants within the pairs formed by taking a plant from each set in the first phase; N_1 is randomized to Lot in the second phase; the combinations of N_2 and Posn is randomized to the combinations of HalfLeaf, DatPlant and AssPosn using a nonorthogonal design, the latter indicated by the 'O'; S = Set; L = Lot; D = DatPlant; A = AssPosn.

Figure 15: 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 16: 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.

7.2.1 Check the properties of the randomized layout

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

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

7.2.2 Questions

1. Summarize the properties of the four design species for this example.
2. Is the variance matrix for this experiment based on two sets of terms that are orthogonal?
3. What are the advantages and disadvantages of a mixed-model analysis of the data from this experiment, as opposed to an anova?

7.3 A p-rep design for a field experiment with 576 Lines

A field experiment is to be conducted on a grid of 60 rows × 12 columns. Of the 576 Lines, 144 are to be duplicated and the remaining 432 are to be unreplicated. In the lecture, a lattice design was used as a starting design. Here a randomized complete-block design will be used. The factor-allocation diagram is in Figure 17.

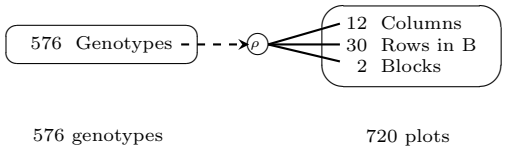

Figure 17: Factor-allocation diagram for the p-rep design for a field experiment with 576 Lines: genotypes are allocated to plots; the dashed arrow on the left indicates that the allocation of Genotypes is not randomized; the 'ρ' at the end of the arrow indicates that Genotypes are allocated to combinations of the levels of Blocks, Rows and Columns, using a design that takes into account correlation between plots; B = Blocks.

7.3.1 Generate the starting design and check the properties of the design

Use the following `R` code to generate a randomized complete-block design and to check its properties.

```
#'# This script generates a p-rep design for 576 lines, 144 of which are replicated and Lines are rando
#'### It is the first-phase design of a two-phase design a la Smith et al. (2006)

#'## Set up constants
g <- 576    # no. genotypes
ndup <- 144 # no. duplicated genotypes
b <- 2      # no. blocks
r <- 60     # no. rows
c <- 12     # no. columns
n <- r*c    # no. plots

#'## Generate an RCBD
#' 1:144 are replicated twice 145:g are replicated once
```

```
blk1.lines <- sample((ndup+1):g, (g-ndup)/2)  #randomly select half undup Lines for Block 1
blk2.lines <- ((ndup+1):g)[!((ndup+1):g %in% blk1.lines)] #rest in Block 2
rcbd.sys <- cbind(fac.gen(list(Blocks = 2, Plots = 360)),
                  Lines = factor(c(1:ndup, blk1.lines,
                                   1:ndup, blk2.lines)))
rcbd.lay <- designRandomize(allocated         = rcbd.sys["Lines"],
                            recipient         = rcbd.sys[c("Blocks", "Plots")],
                            nested.recipients = list(Plots = "Blocks"),
                            seed              = 12471)
rcbd.lay <- cbind(fac.gen(list(Rows = 60, Columns = 12)),
                  rcbd.lay)
rcbd.lay <- within(rcbd.lay, WRows <- fac.recode(Rows, rep(1:30, times=2), levels=1:30))
#'### Check properties
rcbd.canon <- designAnatomy(formulae = list(plot = ~ (Blocks + Rows)*Columns,
                                            trt  = ~ Lines),
                            data     = rcbd.lay)
summary(rcbd.canon, which.criteria = c("aeff", "meff", "eeff", "order", "dfor"))
```

7.3.2 Search for a near-A-optimal design

Use od to search for a near-A-optimal design under a mixed model.

```
#'## Set od options
maxit <- 25
search <- "tabu+rw"
od.options(P = 0.10, localSearch = 10000, tabuStop = 100)

#'## Set up variance parameters (based on Smith et al (2006, p.405))
g.L   <- 1
g.BR  <- 0.5
g.C   <- 0.1
g.BC  <- 0.05
g.u   <- 0.5
g.BRC <- 1.0
rho.R <- 0.6
rho.C <- 0.4
params <- c(g.L, g.BR, g.C, g.BC, g.u, g.BRC, rho.R, rho.C)
names(params) <- c("g.L", "g.BR", "g.C", "g.BC", "g.u", "g.BRC", "rho.R", "rho.C")

#'## Use od to generate the p-rep starting with the RCBD - with units and autocorrelation
prepuar1.rcbd.od<- od(fixed        = ~ Blocks,
                      random       = ~ Lines + Rows + Columns/Blocks + units,
                      residual     = ~ ar1(Rows):ar1(Columns),
                      permute      = ~ Lines, swap = ~ Blocks,
                      start.values = TRUE,
                      data         = rcbd.lay)
vp.table <- prepuar1.rcbd.od$vparameters.table
vp.table$Value <- params
vp.table
prepuar1.rcbd.od<- od(fixed   = ~ Blocks,
                      random   = ~ Lines + Rows + Columns/Blocks + units,
                      residual = ~ ar1(Rows):ar1(Columns),
                      permute  = ~ Lines, swap = ~ Blocks,
```

```
                     G.param  = vp.table, R.param = vp.table,
                     maxit    = maxit, search = search,
                     data     = rcbd.lay)
prepuar1.rcbd.lay <- prepuar1.rcbd.od$design

#'### Plot the design
#+ Breed576opt
prepuar1.rcbd.lay$Replication <- fac.recode(prepuar1.rcbd.lay$Lines,
                                            rep(1:2, c(ndup, (g-ndup))))
designGGPlot(prepuar1.rcbd.lay, labels = "Lines",
             row.factors = c("Blocks", "WRows"), column.factors = "Columns",
             cellfillcolour.column = "Replication",
             colour.values = c("lightgreen", "lightcyan"),
             axis.text.size = 10, blockdefinition = cbind(30,12),
             title = NULL)
```

```
#'### Check properties
prepuar1.rcbd.canon <- designAnatomy(formulae = list(plot = ~ (Blocks + Rows)*Columns,
                                                     trt  = ~ Lines),
                                     data     = prepuar1.rcbd.lay)
summary(prepuar1.rcbd.canon, which.criteria = c("aeff", "meff", "eeff", "order", "dfor"))
```

In the values for the variance parameters, γ_{BC} was set to 0.05, thus indicating that it was thought to be small. The question then arises as to what would be the effect of leaving out the term. To check this recalculate the AVPD without it and redo the anatomy with the source omitted.

```
prepuar1.rcbd.lay$unit <- factor(1:nrow(prepuar1.rcbd.lay)) #factor for ASReml units
(designAmeasures(mat.Vpredicts(target = ~ Lines -1,
                               Gt     = 1,
                               fixed  = ~ Blocks,
                               random = ~ Rows + Columns + unit - 1,
                               G      = as.list(params[c("g.BR", "g.C", "g.u")]),
                               R      = kronecker(mat.ar1(params["rho.R"], r),
                                                  mat.ar1(params["rho.C"], c)),
                               design = prepuar1.rcbd.lay)))[[1]]
prepBCout.canon <- designAnatomy(formulae = list(plot = ~ (Blocks + Rows) + Columns +
                                                         Blocks:Rows:Columns,
                                                 trt  = ~ Lines),
                                 data     = prepuar1.rcbd.lay)
summary(prepBCout.canon, which.criteria = c("aeff", "meff", "eeff", "order", "dfor"))
```

7.3.3 Questions

1. How do the plots of the p-rep designs obtained from the balanced lattice and randomized complete-block designs compare?
2. The A-value for the design obtained from the balanced lattice was 1.016146. How does the design generated from the randomized complete-block design compare?
3. Summarize the differences between the original balanced lattice design and the `od` design. Is the increased precision of the `od` design worthwhile?
4. Is this design connected under a fixed model? How can you tell?

7.4 A two-phase p/q-rep design for a field experiment with 576 Lines

In Section 7.3, a design was constructed for a field experiment to be conducted on a grid of 60 rows × 12 columns. Of the 576 Lines, 144 were duplicated and the remaining 432 were unreplicated. This field experiment is the first-phase of the experiment, the second phase being a milling phase in which samples of grain are taken from the plots to be milled so that quality characteristics of the grain can be ascertained.

The factor-allocation diagram for the two-phase experiment is in Figure 18.

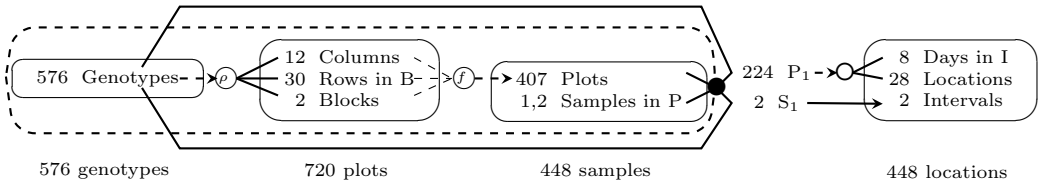

Figure 18: Factor-allocation diagram for a two-phase p/q-rep design for a field experiment with 576 Lines: genotypes are allocated to plots, plots are selected to produce samples and samples are allocated to locations; the dashed arrow on the left indicates that the allocation of Genotypes is not randomized; the '(ρ)' at the end of the arrow indicates that Genotypes are allocated to combinations of the levels of Blocks, Rows and Columns, using a design that takes into account correlation between plots; the '(f)' indicates the selection of a fraction of the levels of Blocks, Rows and Columns; the dashed lines signify that the selection is purposeful; the dashed oval encircling the three panels on the left indicates that a pseudotier of all factors is formed in allocating samples to locations because it uses all the information about the first-phase factors; the levels of the pseudofactor S_1 groups together the samples that are to be assigned to the same interval; the pseudofactor P_1 indexes the Plots that are to occur within the same interval; the dashed arrow ending at the 'O' indicates that Plots within P_1 are systematically allocated, the 'O' indicates that the design is nonorthogonal and the two lines leaving it indicate that the Plots are assigned to the combinations of the levels of Days and Locations within an Interval; B = Blocks; P = Plots; I = Intervals; D = Days.

7.4.1 Select the samples and assign them systematically to the milling phase

Use the following `R` code to select the samples from the field experiment for the milling phase, plot it and check its properties.

```
#'# This script systematically assigns sampled plots from the first-phase.
#'### It is based on an example from Smith et al. (2006)

#'### Select lines for milling phase
samplines <- c(sample(1:ndup, 37),        #select the 37 duplicated field lines
               sample((ndup+1):g, 333))    #select the 333 unduplicated field lines
milldup <- sample(samplines[38:370], 41)  #and from them Select the 41 plots to duplicate in milling ph
#'### Construct revised data.frame
ph2samp.lay <- with(prepuar1.rcbd.lay, prepuar1.rcbd.lay[Lines %in% samplines, ])
ph2samp.lay <- rbind(ph2samp.lay, prepuar1.rcbd.lay[prepuar1.rcbd.lay$Lines %in% milldup, ])
rownames(ph2samp.lay) <- NULL
ph2samp.lay <- within(ph2samp.lay,
                      {
                        Samples <- factor(rep(1:2, c(407, 41)))
                        Lines <- factor(Lines)
                      })
ph2samp.lay <- with(ph2samp.lay, ph2samp.lay[order(Samples, Rows, Columns), ])

#'## Construct line numbers for different types of duplication
ph2lines <- levels(ph2samp.lay$Lines)
n2 <- length(ph2lines)
undup <- ph2lines[!(ph2lines %in% c(1:37, milldup))]
groups <- list(flddup = c(1:n2)[samplines %in% c(1:37)],
               milldup = c(1:n2)[samplines %in% milldup],
```

```
              undup = c(1:n2)[samplines %in% undup])

#'## Plot the sampled plots
#+ "Samples_v6"
fullgrid <-  merge(fac.gen(list(Blocks = 2, WRows = 30, Columns = 12)),
                   ph2samp.lay, all.x = TRUE)
fullgrid$Replication <- 1
fullgrid$Replication[as.numfac(fullgrid$Lines) < 145 ] <- 2
fullgrid$Replication[is.na(fullgrid$Lines)] <- 0
fullgrid$Replication <- factor(fullgrid$Replication)
designGGPlot(fullgrid, labels = "Lines",
             row.factors = c("Blocks","WRows"), column.factors = "Columns",
             cellfillcolour.column = "Replication",
             colour.values = c("grey50","lightcyan","lightgreen"),
             axis.text.size = 10, blockdefinition = cbind(30,12),
             title = NULL,
             ggplotFuncs = list(theme(legend.position = "right")))

#'## Check properties of sampled subset
ph1.canon <- designAnatomy(formulae = list(plot = ~ ((Blocks/WRows)*Columns)/Samples,
                                           trt  = ~ Lines),
                           data     = ph2samp.lay)
print(summary(ph1.canon, which.criteria = c("ae", "me", "ee", "dfor")))


#'## Allocate samples systematically - confounds field and milling dups
ph2sys.lay <- ph2samp.lay
ph2sys.lay$Intervals <- ph2sys.lay$Samples
ph2sys.lay[!(ph2sys.lay$Lines %in% milldup), "Intervals"][184:366] <- 2
ph2sys.lay <- with(ph2sys.lay, ph2sys.lay[order(Intervals, Rows, Columns), ])
ph2sys.lay <- cbind(ph2sys.lay, fac.gen(list(Days=8, Locations=28), times = 2))
ph2sys.lay <- within(ph2sys.lay,
                     {
                       xLocn <- as.numeric(Locations)
                       xLocn <- xLocn - mean(unique(xLocn))
                     })
```

7.4.2 Check the properties of the p/q-rep design

```
#'## Look at properties of the design
layout <- ph2sys.lay
names(layout)[match(c("Intervals", "Locations", "Columns","Samples"), names(layout))] <-
  c("Int", "Locn", "Cols","Samp")
designTwophaseAnatomies(formulae = list(lab = ~ (Int/Days)*Locn,
                                        plot = ~ ((Blocks/WRows)*Cols)/Samp,
                                        trt  = ~ Lines),
                        which.criteria = c("ae", "me", "ee", "dfor"),
                        keep.order = TRUE, data = layout)
```

7.4.3 Substituting a linear Locations term for arbitrary Locations differences

```
#'## Substituting xLocn for Locations (and pooling Blocks and WRows to reduce the table)
ph2sys.lin.canon <- designAnatomy(formulae   = list(lab  = ~ Int/Days + xLocn +
                                                             Int:Days:Locn,
                                                   plot = ~ (Rows*Cols)/Samp,
                                                   trt  = ~ Lines),
                                  keep.order = TRUE,
                                  data       = layout)
print(summary(ph2sys.lin.canon, which.criteria = c("ae", "me", "ee", "dfor")))
```

7.4.4 Questions

1. Where is most of the information about Rows confounded in the two-phase design?
2. What are the effects on the analysis of being able to describe the Locations differences in terms of a linear trend term instead of arbitrary differences between Locations?

References

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

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. (2017) Multiphase experiments in practice: A look back. *Australian & New Zealand Journal of Statistics*, *59*(4), 327–352.

Brien, C. J. (2019) *dae: functions useful in the design and ANOVA of experiments*. URL http://CRAN.R-project.org/package=dae, (R package version 3.1-16, accessed November 15, 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 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.

Butler, D. G. (2019) *od: generate optimal experimental designs*. URL https://mmade.org/, (R package version 2.0.0, to be made available).

Gilmour, A. R., R. Thompson, and B. R. Cullis (1995) Average information reml: An efficient algorithm for variance parameter estimation in linear mixed models. *Biometrics*, *51*, 1440–1450.

Hinkelmann, K. and O. Kempthorne (2005) *Design and Analysis of Experiments*, Volume 2. of *Wiley Series in Probability and Statistics*. Hoboken, N.J.: Wiley-Interscience.

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.

Mead, R. (1990) *The Design of Experiments*. Cambridge: Cambridge University Press.

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*.