APTS Design of Experiments - Practical 1

September 2020

Dave Woods (D.Woods@southampton.ac.uk (mailto:D.Woods@southampton.ac.uk) ; http://www.southampton.ac.uk/~davew (http://www.southampton.ac.uk/~davew))
Statistical Sciences Research Institute, University of Southampton

In this practical session, you will use the `FrF2` package to investigate some aspects of (fractional) factorial designs. Feel free to also use the practical sessions to understand and reproduce results from the notes.

1. **Fractional factorials and fold-over**

(a). Consider a 16-run experiment to investigate nine factors. Use `FrF2` to generate a fractional factorial design defined by

$$I = x_1x_2x_3x_5 = x_1x_2x_4x_6 = x_1x_3x_4x_7 = x_2x_3x_4x_8 = x_1x_2x_3x_4x_9$$

Examine the aliasing scheme of this design. Obtain the alias strings for each main effect.

(b). To *delias* main effects, we can **foldover** a fractional factorial design. A foldover design consists of adding a second set of n runs to the design obtained by "folding over" the original runs; that is, we obtain the second design by multiplying one or more columns of the original design by -1. A complete fold-over generates this design by multiplying all columns by -1.

The function `fold.design` in the `FrF2` package generates foldover designs. Check its syntax in the help file, and then use this function to generate a complete foldover of our original 16-run design.

What do you think will have happened to the alias structure of the new design? Use the `alias` function to check your intuition.

[You should be able to use `design.info()$aliased` but there seems to be a bug in this function that leads to an incorrect alias structure being generated.]

(c). We can also generate partial foldover designs, by only folding over on subsets of columns (ie only multiplying some columns by -1). Use `fold.design` to generate a design by folding over on x_5 only. Then examine the aliasing structure for the generated design. What has happened to the aliasing for the main effects and two-factor interactions involving x_5 ?

2. **Blocking and split-plot designs**

As briefly discussed in the notes, blocks are used when the experimental units can be separated into groups, where it is anticipated that, after the application of a treatment, two units within a group (or block) may give more similar results than two units from different groups.

If we assign the treatments from a factorial design at random to blocks, we may alias important factorial effects with blocks (aliasing between factorial effects and blocks is usually called *confounding*). A better scheme for blocking a factorial design into $b = 2^g$ blocks, each of size 2^{k-g} is to choose some higher

order interactions to confound with blocks, and then assign the treatments so the contrasts for these interactions (ie the columns of the X matrix) are constant within block.

Essentially, we are splitting the full factorial design into b fractions, but rather than choosing only one of the fractions to run, we run all b of them, each in a different block. We confound all the effects in the defining relation of the fraction with blocks, but all other effects are estimable.

(a). `FrF2` can be used to find blocked factorial designs, using the argument `blocks`. Start by setting `blocks=4` to find a design with $n = 32$ runs, $k = 5$ factors and $b = 4$ blocks. You can use `design.info()$aliased.with.blocks` to see what factorial effects have been confounded with blocks.

(b). You can also select which factorial effects are confounded with blocks by setting the `blocks` argument equal to a list of factorial effects (the same as the `generator` argument for fractional factorial designs). Set `blocks = list(c(2, 3, 4, 5), c(1, 2, 3, 4))`, and compare the results to the previous design found (new for 2019: you will also need to set `alias.block.2fis = TRUE`). What other factorial effect is confounded with blocks?

(c). We can also block fractional factorial designs in a similar way by picking factorial effects to confound with blocks. Of course now, any other effects aliased with the chosen effects will also be confounded with blocks (making the choice a bit trickier).

Let's now block a half-fraction of the $k = 5$ full factorial into four blocks, using `FrF2` where we specify both the `generators` and `blocks`. Pick a factorial effect to define the fraction and factorial effects to define the blocks. Study the aliasing scheme of the resulting design to understand the consequences of you choices on the effects that can be estimated.

(d). For some experiments, one or more of the factors might be "hard-to-change"; that is, it is desirable to restrict the randomisation so the values of these factors are changed less often. A simple example might be the temperature of a oven in a food processing experiment. The time it takes for the oven to heat up or cool down makes it necessary to limit the number of temperature changes. A design with at least one hard-to-change factor is called a *split-plot* design, as there are now two levels of experimental units: groups of units (whole plots) which will all be assigned the same value of the hard-to-change factors, and individual units within these groups (sub plots) to which the values of the other factors will be assigned.

Split-plot designs can be found via blocking (fractional) factorial designs by confounding the main effects of the whole-plot factors with blocks. Using `FrF2`, we can explicitly find split-plot designs using the `WPs` and `nfac.WP` arguments to set the number of whole plots and number of whole-plot factors, respectively.

Use `FrF2` to find a split-plot design with $n = 32$ runs, $k = 5$ factors with two whole-plot factors (`FrF2` assumes these will be the first two factors). The values of the two whole-plot factors should be constant within whole-plots.

Split-plot designs are usually analysed using a mixed-effects model, with random effects associated with the whole-plot factors. You can do this using `lmer` from the `lme4` package (see APTS Statistical Modelling). Generate some random data and fit a mixed model to your whole-plot design. What do you notice about the standard errors of the different coefficients?

(Hint - the model formula should be something like `y ~ (x1 + x2 + x3 + x4 + x5)^2 + (1 | wp)`, where `wp` is a column identifying the different whole plots in the experiment).