A quick guide to PLANOR, an R library for the automatic generation of regular fractional factorial designs

Monod H., Bouvier A., Kobilinsky A.

INRA, UR 341, Unité MIAj
Mathématiques et Informatique Appliquées - Jouy
F78352 Jouy en Josas Cedex
France

May 23, 2011

Contents

Contents

1 Introduction

The `PLANOR` R library generates regular fractional factorial designs for a wide and flexible range of user specifications. It is based on algebraic methods of construction and more specifically on the key matrix method [8], described in detail in [5], [6], [2], and more simply in [1]. This method produces so-called *regular* fractional designs in which factorial effects are either estimable independently or completely confounded. The `PLANOR` R library originates from the `PLANOR` software which was written in the APL language by André Kobilinsky. The initial `PLANOR` manual [3] has been adapted to the `PLANOR` R library [4] and gives more details on the theory than this short guide.

`PLANOR` can manage factors with different numbers of levels. It can take into account hierarchical relationships among factors. It is also possible to control the confounding of treatments effects with block effects, as in split-plot or criss-cross experiments. The user provides information on the design factors, on the factorial effects to include in the model and on those to estimate, and on the design size. He or she then asks `PLANOR` to search for one or more designs meeting the requirements. The solutions, if any, are given as a list of design key matrices. Several specific functions then allow to investigate the solutions' properties and to print and store the resulting designs.

This vignette presents the basic usage of PLANOR . A more comprehensive presentation is under preparation, as well as additional package functions. More details are also available through the help functions of the PLANOR package.

2 Fractional designs with 2-level factors

We consider an experiment with four treatment factors and one block factor, all five at two levels. The aim is to estimate the main effects of the treatment factors, assuming that the model also includes the two-factor treatment interactions and the main effect of the block factor. Each block has four units so that the size of the required design is $N = 2^3 = 8$. In this example, the key matrix has three rows and five columns.

After loading the library, the experiment requirements are specified in three parts : *(i)* the factors ; *(ii)* the model and (optionally) the subset of factorial effects to estimate ; *(iii)* the design size. All this information can be provided directly to the `planor.designkey` function, which searches for design key solutions. The optional `block` argument allows to specify that some factors should be considered as block (or nuisance) factors. In PLANOR , the distinction between *treatment* and *block* factors is taken into account when studying confounding and aliasing properties of the fractional designs associated with a given key matrix. The `estimate` argument of the `planor.model` function is optional : by default, it is considered that all terms in the `model` formula must be estimated.

```
> library("planor")
> # ***************** EXAMPLE 1 *****************
> # Four 2-level treatment factors and one 2-level block factor
> # Model: block+(A+B+C+D)^2  -   Estimate: A+B+C+D
> # N = 2^3 = 8 units
> #
> ex1Key <- planor.designkey(factors=c("block","A","B","C","D"),
+                            nlevels=rep(2,5),
+                            block=~block,
+                            model=~block+(A+B+C+D)^2,
+                            estimate=~A+B+C+D,
+                            nunits=2^3)

Determination of ineligible factorial terms
Determination of ineligible pseudofactorial terms
Independent searches for prime(s) :  2
Key-matrix search for prime p = 2
There is  1  predefined column
First visit to column  2
First visit to column  3
First visit to column  4
First visit to column  5
The search is closed: max.sol =  1 solution(s) found
```

During the search, a so-called backtrack algorithm looks successively for new columns to add to the key matrix. Succinct information is given to check the algorithm progress. By default, the search stops as soon as one solution is found.

As an alternative to using `planor.designkey` directly, the user may provide the information on the experiment step by step with the functions `planor.factors` and `planor.model`. The idea is to store the results of these functions in R objects and use them as arguments to `planor.designkey`. This may be convenient, for example, when one wants to explore several possible models and design sizes with the same set of factors.

```
> ex1Fac <- planor.factors( factors=c("block","A","B","C","D"),
+                           nlevels=rep(2,5),
+                           block=~block )
> ex1Mod <- planor.model( model=~block+(A+B+C+D)^2, estimate=~A+B+C+D )
> ex1Key <- planor.designkey(factors=ex1Fac, model=ex1Mod, nunits=2^3)

Determination of ineligible factorial terms
Determination of ineligible pseudofactorial terms
Independent searches for prime(s) :  2
Key-matrix search for prime p = 2
There is  1  predefined column
First visit to column  2
First visit to column  3
First visit to column  4
First visit to column  5
The search is closed: max.sol =  1 solution(s) found
```

In both cases, the key matrix solution is stored in the object `ex1Key`. If needed, detailed properties of the solution can be obtained by two different functions. The `summary` function prints the key matrix and the defining relationships of the fractions that can be generated with this key matrix. More detailed information on the aliasing between factorial effects is given by the function `alias`.

```
> summary(ex1Key)

********** Prime  2  design **********

--- Solution  1  for prime  2  ---

DESIGN KEY MATRIX
    block A B C D
*U*     1 0 1 0 1
*U*     0 1 1 0 0
*U*     0 0 0 1 1

TREATMENT EFFECTS CONFOUNDED WITH THE MEAN
1 = A B C D

BLOCK-and-TREATMENT EFFECTS CONFOUNDED WITH THE MEAN
1 = block A B
1 = block C D

WEIGHT PROFILES
Treatment effects confounded with the mean: 4^1
Treatment effects confounded with block effects: 2^2
Treatment pseudo-effects confounded with the mean: 4^1
Treatment pseudo-effects confounded with block effects: 2^2

> alias(ex1Key)

********** Prime  2  design **********

--- Solution  1  for prime  2  ---

UNALIASED TREATMENT EFFECTS
```

```
A ; B ; C ; D

ALIASED TREATMENT EFFECTS
A C = B D
A D = B C

TREATMENT EFFECTS CONFOUNDED WITH BLOCK EFFECTS
block = A B = C D

UNALIASED BLOCK EFFECTS
nil


--- Synthesis on the aliased treatment effects for prime  2  ---

     unaliased trt.aliased blc.aliased
[1,]         4           4           2
```

Last but not least, a design can be generated by the function `planor.design`. The design itself is the object in slot `design` of the more complex object generated by `planor.design`. An option allows the design to be randomized, according to a block structure formula that the user must specify (option `randomize`).

```
> ex1Des <- planor.design(ex1Key)

Extraction of a design key from an object of class listofkeyrings

> print(ex1Des@design)

  block A B C D
1     1 1 1 1 1
2     1 1 1 2 2
3     1 2 2 1 1
4     1 2 2 2 2
5     2 1 2 1 2
6     2 1 2 2 1
7     2 2 1 1 2
8     2 2 1 2 1

> ex1Rand <- planor.design(ex1Key, randomize=~block/UNITS)

Extraction of a design key from an object of class listofkeyrings

> print(ex1Rand@design)

  UNITS block A B C D
1     1     1 1 2 1 2
2     2     1 1 2 2 1
3     3     1 2 1 1 2
4     4     1 2 1 2 1
5     5     2 1 1 2 2
6     6     2 2 2 1 1
7     7     2 2 2 2 2
8     8     2 1 1 1 1
```

3 Fractional designs with 3-level factors

We keep the same example but with 3-level factors and a few more options. The results are not shown for sake of brevity.

```
> # ***************** EXAMPLE 2 *****************
> # Four 3-level treatment factors and one 3-level block factor
> # Model: block+(A+B+C+D)^2  -   Estimate: A+B+C+D
> # N = 3^3 = 27 units
> #
> ex2Key <- planor.designkey(factors=c(LETTERS[1:4],"block"),
+                            nlevels=rep(3,5),
+                            block=~block,
+                            model=~block+(A+B+C+D)^2,
+                            estimate=~A+B+C+D,
+                            nunits=3^3, base=~A+B+C, max.sol=2)
> summary(ex2Key)
> summary(ex2Key)
> ex2Des <- planor.design(ex2Key[2])
```

Two optional arguments of `planor.designkey` have been used, first to specify that A, B and C should be used as basic factors, and second to ask for two solutions whereas the default is one. Both solutions are examined by `summary` and the second one, say, is chosen by the user to generate a factorial design. When *basic* factors are specified, they are used to generate and identify the experimental units [3]. As a consequence, all combinations of the basic factors are guaranteed to be included in the design. When relevant, using basic factors is recommended because it can speed up the search.

The following lines also work; they illustrate that the basic factors need not be part of the model but they must have been declared in `planor.factors`.

```
> ex2Fac <- planor.factors(factors=c(LETTERS[1:4], "block", "BASE"),
+                          nlevels=rep(3,6) )
> ex2Mod <- planor.model(model=~block+(A+B+C+D)^2,
+                        estimate=~A+B+C+D )
> ex2Key <- planor.designkey(factors=ex2Fac,
+                            model=ex2Mod,
+                            nunits=3^3,
+                            base=~A+B+BASE,
+                            max.sol=2)
```

4 Asymmetric fractional factorial designs

A regular fractional factorial design is called mixed or asymmetric when the numbers of levels of the factors involve several different prime numbers. The asymmetric designs constructed in `PLANOR` consist of the cross products of designs based on each prime. This does not allow for a great flexibility in terms of confounding, but it enlarges the scope of situations that can be addressed.

```
> # Four treatment factors at 6, 6, 4, 2 levels and one 6-level block factor
> # Model: block+(A+B+C+D)^2 ; Estimate: A+B+C+D\n")
> # N = 144 = 2^4 x 3^2 experimental units
> mixKey <- planor.designkey(factors=c( LETTERS[1:4], "block"),
+                            nlevels=c(6,6,4,2,6),
+                            block=~block,
```

```
+                              model=~block+(A+B+C+D)^2,
+                              estimate=~A+B+C+D,
+                              nunits=144,
+                          base=~A+B+D, max.sol=2)

Determination of ineligible factorial terms
Determination of ineligible pseudofactorial terms
Independent searches for prime(s) :  2 3
Key-matrix search for prime p = 2
There are  3  predefined columns
First visit to column  4
First visit to column  5
First visit to column  6
The search is closed: max.sol =  2 solution(s) found
Key-matrix search for prime p = 3
There are  2  predefined columns
First visit to column  3
The search is closed: max.sol =  2 solution(s) found

> summary(mixKey)

********** Prime  2  design **********

--- Solution  1  for prime  2  ---

DESIGN KEY MATRIX
    A_1 B_1 D C_1 C_2 block_1
A_1   1   0 0   1   0       1
B_1   0   1 0   1   0       1
D     0   0 1   1   0       0
*U*   0   0 0   0   1       0

TREATMENT EFFECTS CONFOUNDED WITH THE MEAN
1 = A_1 B_1 D C_1

BLOCK-and-TREATMENT EFFECTS CONFOUNDED WITH THE MEAN
1 = A_1 B_1 block_1
1 = D C_1 block_1

WEIGHT PROFILES
Treatment effects confounded with the mean: 4^1
Treatment effects confounded with block effects: 2^2
Treatment pseudo-effects confounded with the mean: 4^1
Treatment pseudo-effects confounded with block effects: 2^2

--- Solution  2  for prime  2  ---

DESIGN KEY MATRIX
    A_1 B_1 D C_1 C_2 block_1
A_1   1   0 0   1   0       1
B_1   0   1 0   1   0       0
D     0   0 1   1   0       1
*U*   0   0 0   0   1       0

TREATMENT EFFECTS CONFOUNDED WITH THE MEAN
```

```
1 = A_1 B_1 D C_1

BLOCK-and-TREATMENT EFFECTS CONFOUNDED WITH THE MEAN
1 = A_1 D block_1
1 = B_1 C_1 block_1

WEIGHT PROFILES
Treatment effects confounded with the mean: 4^1
Treatment effects confounded with block effects: 2^2
Treatment pseudo-effects confounded with the mean: 4^1
Treatment pseudo-effects confounded with block effects: 2^2


********** Prime  3  design **********

--- Solution  1  for prime  3  ---

DESIGN KEY MATRIX
    A_2 B_2 block_2
A_2   1   0       1
B_2   0   1       1

TREATMENT EFFECTS CONFOUNDED WITH THE MEAN
nil

BLOCK-and-TREATMENT EFFECTS CONFOUNDED WITH THE MEAN
1 = A_2^2  B_2^2  block_2

WEIGHT PROFILES
Treatment effects confounded with the mean: none
Treatment effects confounded with block effects: 2^1
Treatment pseudo-effects confounded with the mean: none
Treatment pseudo-effects confounded with block effects: 2^1

--- Solution  2  for prime  3  ---

DESIGN KEY MATRIX
    A_2 B_2 block_2
A_2   1   0       2
B_2   0   1       1

TREATMENT EFFECTS CONFOUNDED WITH THE MEAN
nil

BLOCK-and-TREATMENT EFFECTS CONFOUNDED WITH THE MEAN
1 = A_2 B_2^2  block_2

WEIGHT PROFILES
Treatment effects confounded with the mean: none
Treatment effects confounded with block effects: 2^1
Treatment pseudo-effects confounded with the mean: none
Treatment pseudo-effects confounded with block effects: 2^1
```

```
> mixPlan <- planor.design(key=mixKey, select=c(1,1), randomize=~block/UNITS)
```

```
Extraction of a design key from an object of class listofkeyrings

> reorder <- order(mixPlan@design$block,mixPlan@design$UNITS)
> mixPlan@design <- mixPlan@design[reorder,]
> print(mixPlan@design[1:25,])

    UNITS A B D C block
1       1 6 6 1 2     1
6       6 6 6 2 4     1
8       8 5 4 1 2     1
10     10 6 6 2 3     1
15     15 3 3 1 2     1
17     17 4 5 1 2     1
19     19 1 2 2 3     1
24     24 5 4 1 1     1
26     26 1 2 1 2     1
28     28 2 1 1 1     1
33     33 5 4 2 3     1
35     35 3 3 2 3     1
109   109 4 5 2 4     1
114   114 4 5 1 1     1
116   116 2 1 2 3     1
118   118 3 3 1 1     1
123   123 6 6 1 1     1
125   125 3 3 2 4     1
127   127 4 5 2 3     1
132   132 2 1 1 2     1
134   134 2 1 2 4     1
136   136 5 4 2 4     1
141   141 1 2 2 4     1
143   143 1 2 1 1     1
2       2 4 3 2 2     2
```

The algorithm starts by decomposing the factors into pseudofactors that all have a prime number of levels. Then it performs a similar decomposition of the `model` and `estimate` terms. After these initial steps, separate key-matrix searches are performed, one for each prime involved in the problem. The prime decompositions are automatic and transparent to the user. The recomposition when generating a design is transparent too. In contrast, most information on the search process and on the fraction properties are given according to the prime decompositions.

5 Split-plot designs

In a split-plot experiment, there are two treatment factors `variety` and `fert`, say, at m and n levels respectively. The block structure consists of r blocks each containing m sub-blocks of size n and the factor `variety` is constrained to be constant within sub-blocks.

In an orthogonal split-plot design, each variety occupies one sub-block of each block, and each sub-block contains the n distinct levels of factor `fert`. In `PLANOR` , this design can be constructed by defining the block structure as a cross between a `block` and a `subblock` factor. The `hierarchy` argument is used to specify that `variety` must be constant within the combinations of `block` and `subblock`. Two model-estimate pairs are given to the `listofmodels` argument. First, the main effect of `fert` and the interaction between `fert` and `variety` must be estimable when blocks and sub-blocks are included in the model. Second, the main effect of `variety` must be estimable between sub-blocks, that is, when blocks but not sub-blocks are included in the model. The command below calculates the design key of a split-plot design with $r = 2$, $n = 2$, $m = 2$.

```
> splitKey <- planor.designkey(factors=list(block=1:2,
+                                 subblock=1:2,
+                                 variety=LETTERS[1:2],
+                                 fert=c("organic","mineral")),
+                               block=~block+subblock,
+                               hierarchy=list(~variety/(block*subblock)),
+                               listofmodels=
+                               list(c( ~block*subblock+variety*fert, ~fert+fert:variety),
+                                    c( ~block+variety,               ~variety)),
+                               nunits=2*2*2,
+                               base=~block+subblock)

Determination of ineligible factorial terms
Determination of ineligible pseudofactorial terms
Independent searches for prime(s) :  2
Key-matrix search for prime p = 2
There are  2  predefined columns
First visit to column  3
First visit to column  4
The search is closed: max.sol =  1 solution(s) found

> summary(splitKey)

********** Prime  2  design **********

--- Solution  1  for prime  2  ---

DESIGN KEY MATRIX
         block subblock variety fert
block        1        0       0    0
subblock     0        1       1    0
*U*          0        0       0    1

TREATMENT EFFECTS CONFOUNDED WITH THE MEAN
nil

BLOCK-and-TREATMENT EFFECTS CONFOUNDED WITH THE MEAN
1 = subblock variety

WEIGHT PROFILES
Treatment effects confounded with the mean: none
Treatment effects confounded with block effects: 1^1
Treatment pseudo-effects confounded with the mean: none
Treatment pseudo-effects confounded with block effects: 1^1

> alias(splitKey)

********** Prime  2  design **********

--- Solution  1  for prime  2  ---

UNALIASED TREATMENT EFFECTS
nil

ALIASED TREATMENT EFFECTS
```

```
nil

TREATMENT EFFECTS CONFOUNDED WITH BLOCK EFFECTS
nil

UNALIASED BLOCK EFFECTS
block ; subblock


--- Synthesis on the aliased treatment effects for prime  2  ---

     unaliased trt.aliased blc.aliased
[1,]         0           0           0
```

```
> print(planor.design(splitKey, randomize=~block/subblock/UNITS)@design)

Extraction of a design key from an object of class listofkeyrings
  UNITS block subblock variety    fert
1     1     1        1       A mineral
2     2     1        1       A organic
3     3     1        2       B organic
4     4     1        2       B mineral
5     5     2        1       A organic
6     6     2        1       A mineral
7     7     2        2       B organic
8     8     2        2       B mineral
```

An alternative command to get the split-plot is given below. The main difference is that the `subblock` factor now takes rm levels and is considered as nested in `block` rather than crossed with it.

```
> splitKey <- planor.designkey(factors=list(block=1:2,
+                                 subblock=1:4,
+                                 variety=LETTERS[1:2],
+                                 fert=c("organic","mineral")),
+                               block=~block+subblock,
+                               hierarchy=list( ~block/subblock, ~variety/subblock),
+                               listofmodels=
+                               list(c( ~subblock+variety*fert, ~fert+fert:variety),
+                                    c( ~block+variety,              ~variety)),
+                               nunits=2*2*2,
+                               base=~subblock)

Determination of ineligible factorial terms
Determination of ineligible pseudofactorial terms
Independent searches for prime(s) :  2
Key-matrix search for prime p = 2
There are  2  predefined columns
First visit to column  3
First visit to column  4
First visit to column  5
The search is closed: max.sol =  1 solution(s) found
```

```
> print(planor.design(splitKey, randomize=~block/subblock/UNITS)@design)
```

```
Extraction of a design key from an object of class listofkeyrings
  UNITS subblock block variety    fert
1     1        1     1       B mineral
2     2        1     1       B organic
3     3        2     1       A organic
4     4        2     1       A mineral
5     5        3     2       B mineral
6     6        3     2       B organic
7     7        4     2       A mineral
8     8        4     2       A organic
```

6 Fractional designs with nested factors and a complex block structure

We now consider an experiment with concrete and more complex specifications. This example stems from an experiment to study the cleaning of surfaces by a robot, see [3], example 3 on page 3. There are five treatment factors at 2 levels. The block structure consists of four plates with 2 rows and 4 columns per plate, resulting in 32 experimental units. In addition, the design must cope with experimental constraints between treatment and block factors. The treatment factors concentration (`conc`) and temperature (`Tact`) must remain constant within a plate. The treatment factors denoted by `nsoil` and `qsoil` must remain constant within each column of each plate. Only treatment factor rugosity (`Rug`) can be modified freely between experimental units.

To begin with, we show how to specify user-defined factor levels, by providing a list to the `factors` argument of `planor.factors`. Then, experimental constraints are specified through the `hierarchy` argument of `planor.factors`.

```
> # ************ ROBOT1A EXAMPLE *************
> # Block structure: 4 plates / (2 rows x 4 columns)
> # Treatments: 4 2-level factors
> # Hierarchy 1: conc constant in plate
> # Hierarchy 2: Tact constant in plate
> # Hierarchy 3: nsoil constant in plate x column
> # Hierarchy 4: qsoil constant in plate x column
> # N = 32 units
> #
> robotFac <- planor.factors( factors=list(
+                             conc=c(1,3),
+                             Tact=c(15,30),
+                             nsoil=c("curd","Saint-Paulin"),
+                             qsoil=c("0.01g","0.10g"),
+                             Rug=c(0.25,0.73),
+                             plate=1:4,
+                             row=1:2,
+                             col=1:4),
+                            hierarchy=list(~conc/plate,
+                            ~Tact/plate,
+                            ~nsoil/(plate*col),
+                            ~qsoil/(plate*col)))
```

This example requires several model-estimate combinations. The main model-estimate pair contains all the treatment factorial effects but no block effect. It guarantees that all treatment combinations will be present in the design, since all treatment factorial effects are required to be estimable in the model with no block effect. The second model-estimate pair (`listofmodels` argument) ensures that the `Rug` factor is orthogonal to block factors.

```
> robotMod <- planor.model( model=~nsoil*qsoil*Rug*conc*Tact,
+                           listofmodels=list(c(~plate+row+col+Rug, ~Rug)) )
```

The base option of the planor.designkey function is used here to impose that experimental units be associated with the combinations of the block factors.

```
> robotKey <- planor.designkey(factors = robotFac, model = robotMod,
+     nunits = 32, base = ~plate + row + col)

Determination of ineligible factorial terms
Determination of ineligible pseudofactorial terms
Independent searches for prime(s) :  2
Key-matrix search for prime p = 2
There are  5  predefined columns
First visit to column  6
First visit to column  7
First visit to column  8
First visit to column  9
First visit to column  10
The search is closed: max.sol =  1 solution(s) found

> summary(robotKey[1])

Extraction of a design key from an object of class listofkeyrings

********** Prime  2  design **********

DESIGN KEY MATRIX
        plate_1 plate_2 row col_1 col_2 conc Tact nsoil qsoil Rug
plate_1       1       0   0     0     0    1    0     0     0   1
plate_2       0       1   0     0     0    0    1     0     0   0
row           0       0   1     0     0    0    0     0     0   1
col_1         0       0   0     1     0    0    0     1     0   0
col_2         0       0   0     0     1    0    0     0     1   0

TREATMENT EFFECTS CONFOUNDED WITH THE MEAN
1 = plate_1 conc
1 = plate_2 Tact
1 = col_1 nsoil
1 = col_2 qsoil
1 = plate_1 plate_2 conc Tact
1 = col_1 col_2 nsoil qsoil
1 = plate_1 row Rug
1 = row conc Rug
1 = plate_1 col_1 conc nsoil
1 = plate_2 col_1 Tact nsoil
1 = plate_1 col_2 conc qsoil
1 = plate_2 col_2 Tact qsoil
1 = plate_1 plate_2 row Tact Rug
1 = plate_1 plate_2 col_1 conc Tact nsoil
1 = plate_1 plate_2 col_2 conc Tact qsoil
1 = plate_1 col_1 col_2 conc nsoil qsoil
1 = plate_2 col_1 col_2 Tact nsoil qsoil
1 = plate_2 row conc Tact Rug
1 = plate_1 row col_1 nsoil Rug
```

```
1 = row col_1 conc nsoil Rug

BLOCK-and-TREATMENT EFFECTS CONFOUNDED WITH THE MEAN
nil

WEIGHT PROFILES
Treatment effects confounded with the mean: 2^4 3^4 4^5 5^9 6^5 7^3 8^1
Treatment effects confounded with block effects: none
Treatment pseudo-effects confounded with the mean: 2^4 4^6 3^2 5^6 6^4 8^1 7^6 9^2
Treatment pseudo-effects confounded with block effects: none
```

```
> robotDes <- planor.design(robotKey[1], randomize = ~plate/(row *
+     col))
```

```
Extraction of a design key from an object of class listofkeyrings
```

```
> print(robotDes@design)
```

```
   plate row col conc Tact        nsoil qsoil  Rug
1      1   1   1    3   15         curd 0.10g 0.25
2      1   1   2    3   15 Saint-Paulin 0.01g 0.25
3      1   1   3    3   15         curd 0.01g 0.25
4      1   1   4    3   15 Saint-Paulin 0.10g 0.25
5      1   2   1    3   15         curd 0.10g 0.73
6      1   2   2    3   15 Saint-Paulin 0.01g 0.73
7      1   2   3    3   15         curd 0.01g 0.73
8      1   2   4    3   15 Saint-Paulin 0.10g 0.73
9      2   1   1    1   15 Saint-Paulin 0.01g 0.25
10     2   1   2    1   15         curd 0.10g 0.25
11     2   1   3    1   15         curd 0.01g 0.25
12     2   1   4    1   15 Saint-Paulin 0.10g 0.25
13     2   2   1    1   15 Saint-Paulin 0.01g 0.73
14     2   2   2    1   15         curd 0.10g 0.73
15     2   2   3    1   15         curd 0.01g 0.73
16     2   2   4    1   15 Saint-Paulin 0.10g 0.73
17     3   1   1    3   30         curd 0.10g 0.73
18     3   1   2    3   30 Saint-Paulin 0.10g 0.73
19     3   1   3    3   30 Saint-Paulin 0.01g 0.73
20     3   1   4    3   30         curd 0.01g 0.73
21     3   2   1    3   30         curd 0.10g 0.25
22     3   2   2    3   30 Saint-Paulin 0.10g 0.25
23     3   2   3    3   30 Saint-Paulin 0.01g 0.25
24     3   2   4    3   30         curd 0.01g 0.25
25     4   1   1    1   30 Saint-Paulin 0.10g 0.73
26     4   1   2    1   30         curd 0.01g 0.73
27     4   1   3    1   30 Saint-Paulin 0.01g 0.73
28     4   1   4    1   30         curd 0.10g 0.73
29     4   2   1    1   30 Saint-Paulin 0.10g 0.25
30     4   2   2    1   30         curd 0.01g 0.25
31     4   2   3    1   30 Saint-Paulin 0.01g 0.25
32     4   2   4    1   30         curd 0.10g 0.25
```

Acknowledgements

This vignette was typed using the Sweave package (Leisch, 2002[7]).

References

[1] S. Cliquet, C. Durier & A. Kobilinsky – "Principle of a fractional factorial design for qualitative and quantitative factors: application to the production of *Bradyrhizobium japonicum* in culture media", *Agronomie* **14** (1994), p. 569–587.

[2] A. Kobilinsky – "Les plans factoriels", in *Plans d'expériences: applications à l'entreprise* (J. Droesbeke, J. Fine & G. Saporta, éds.), Technip, Paris, 1997, p. 69–209 (Chapter 3).

[3] — , "PLANOR : program for the automatic generation of regular experimental designs. version 2.2 for Windows", Tech. report, MIA Unit, INRA Jouy en Josas, 2005.

[4] A. Kobilinsky, A. Bouvier & H. Monod – "PLANOR : an R library for the automatic generation of regular fractional factorial designs. Version 1.0", Tech. report, MIA Unit, INRA Jouy en Josas, 2011.

[5] A. Kobilinsky & H. Monod – "Experimental design generated by group morphisms: an introduction", *Scand. J. Statist.* **18** (1991), p. 119–134.

[6] — , "Juxtaposition of regular factorial designs and the complex linear model", *Scand. J. Statist* **22** (1995), p. 223–254.

[7] F. Leisch – "Sweave: Dynamic generation of statistical reports using literate data analysis", in *Compstat 2002 — Proceedings in Computational Statistics* (W. Härdle & B. Rönz, éds.), Physica Verlag, Heidelberg, 2002, ISBN 3-7908-1517-9, p. 575–580.

[8] H. Patterson & R. Bailey – "Design keys for factorial experiments", *Appl. Statist.* **27** (1978), p. 335–343.