Designing comparative experiments using R

(Chris Brien and Sam Rogers)

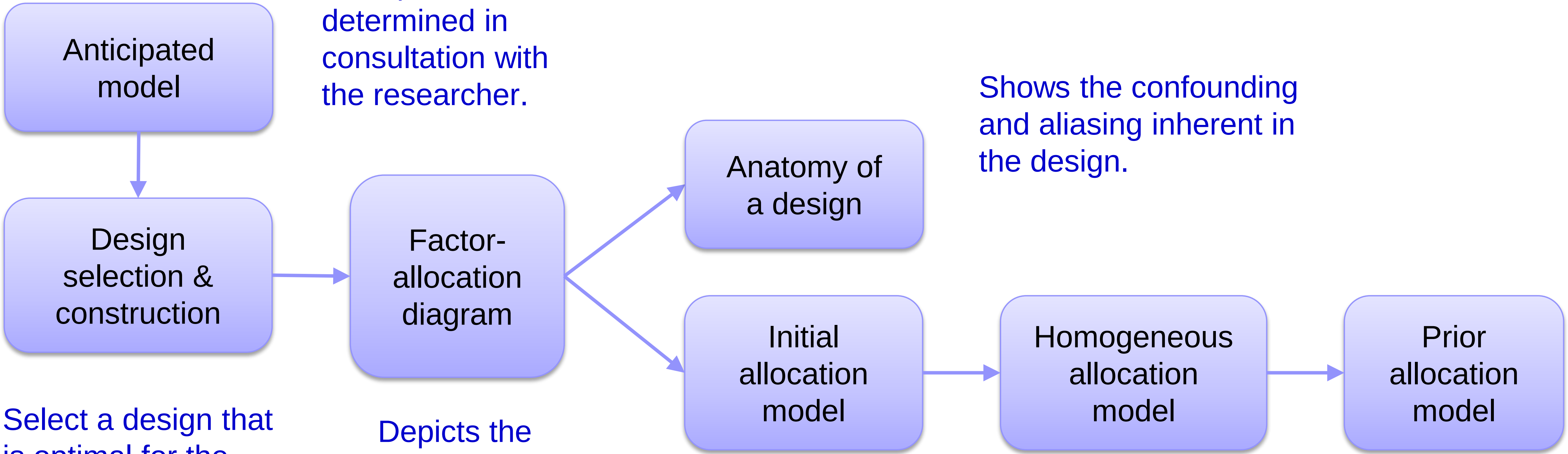
II. Nonorthogonal experimental design

Outline

1. Designing nonorthogonal experiments and the alphabet of efficiency measures.
2. Using the concepts for balanced designs.
3. Using the concepts for unbalanced designs.
   a. A partially balanced incomplete-block design
   b. A wheat experiment from Gilmour et al. (1995)
   c. A plant accelerator design
4. Summary of constructing nonorthogonal designs.
5. What happens when there is missing data?
6. Systematic allocation and pseudoreplication.
7. Summary of confounding and aliasing.

Recall the paradigm for designing experiments

(Brien, 2017)


Anticipated model
Design selection & construction
Factor-allocation diagram
Anatomy of a design
Initial allocation model
Homogeneous allocation model
Prior allocation model

Anticipated model determined in consultation with the researcher.

Shows the confounding and aliasing inherent in the design.

Select a design that is optimal for the anticipated model.

Depicts the allocation.

Model derived from the allocation – should match the anticipated model.

Modified versions of the initial allocation model.

1. Designing nonorthogonal experiments

- For nonorthogonal experiments, getting the initial systematic design is generally more difficult than for orthogonal experiments.
  - Cannot just deploy a standard known design;
  - Will demonstrate a number of approaches.
- Our **`dae`** friends, **`designRandomize`** and **`designAnatomy`**, play the same role as for orthogonal experiments.
  (**`designRandomize`** is not used for spatial designs.)
- How do we know that the design that we have is good?
  - Design optimality is the answer.
  - There is A-, D-, C-, E-, G-, M- and S-optimality. Which one?

Design optimality

- For comparative experiments, A-optimality is the favoured optimality criterion.
  - The definition of A-optimality is that it minimizes the total variance of the predictions or Prediction Error Variance (PEV) (Kiefer, 1959)
  - The PEV is the same as the average variance of pairwise differences (AVPD):
    - when terms to be optimized (e.g. Treatments) are fixed;
    - not when the terms to be optimized are random.
    - when the residual model is not iid i.e. correlated residuals are OK.
- Often suggested that minimum AVPD is the criterion of choice for comparative experiments.
  - So they will be A-optimal if the terms to be optimized are fixed;
  - But what if the terms to be optimized are random?
    - Is AVPD appropriate for random factors?
    - Given the effects are random, conducting inference on a pair of differences is not meaningful.
    - So PEV seems a reasonable measure, but it is not the same as AVPD; nonetheless AVPD is used.
- As previously mentioned, often fixed-model A-optimal designs are sought for comparative experiments:
  - All model terms are assume fixed, except the residuals.

2. Using the concepts for balanced designs

- Suppose have 20 plots arranged in a grid of 4 rows × 5 columns.
- We want to assign 5 lines to the 20 plots.
- Again, what design to use?
  - Completely Randomized CRD,
  - Randomized Complete or Incomplete Block (RCBD or IBD), or
  - Youden Square Design (YSD) (an LSD is impossible)?
- Already know that, irrespective of the design:
  - the unit factors are Rows, Columns and the treatment factor is Lines.
- Suppose that Row and Column differences are probable.
- What is the anticipated model?
  - Lines + Rows + Columns | Rows:Columns — same as for an LSD

2(a) Row and Column differences likely in 4 x 5 grid

- Need a design that allows for Row and Column main effects.
  - Are Rows and Columns crossed or nested? Why?
    - Crossed because expect consistent differences between Rows and between Columns.
- YSD is a design that is optimal for this model:
  - Construct by taking a Latin square and omitting a row.
- Use of **`designRandomize`** (and **`designLatinSqrSys`**) to get a design:

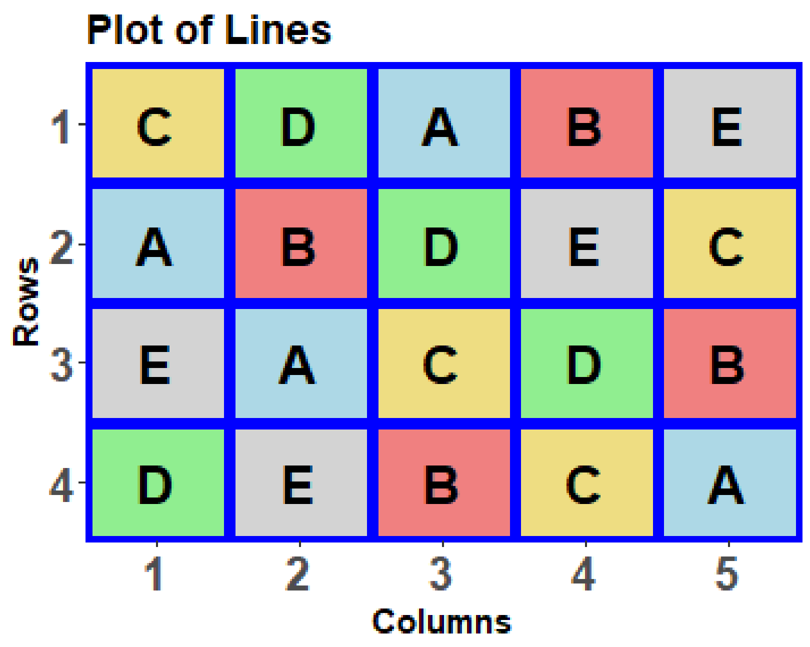
```
b <- 4
t <- 5
> #'## Construct a systematic layout and obtain the randomized layout
> YSD.sys <- cbind(fac.gen(list(Rows=b, Columns=t)),
+                  Lines = factor(designLatinSqrSys(t)[1:(b*t)], labels = LETTERS[1:t]))
> YSD.lay <- designRandomize(allocated = YSD.sys["Lines"],
+                            recipient = YSD.sys[c("Rows", "Columns")],
+                            seed      = 95332)
> #'## Output the layout
> YSD.lay
```

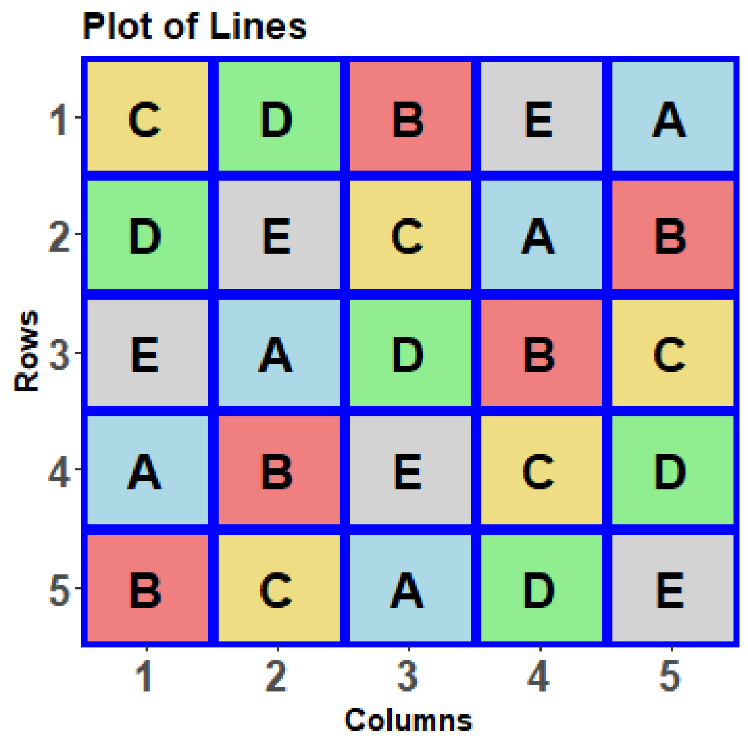
Extract column subsets of **`data.frames`**.

Generate Latin square, but take only first 20 of 25 values (4 rows x 5 columns).

YSD.lay

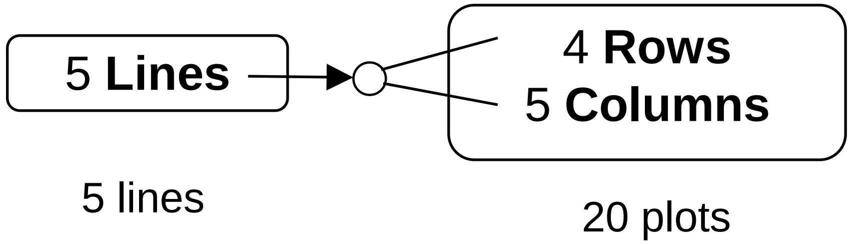
```
   Rows Columns Lines
1     1       1     C
2     1       2     D
3     1       3     A
4     1       4     B
5     1       5     E
6     2       1     A
7     2       2     B
8     2       3     D
9     2       4     E
10    2       5     C
11    3       1     E
12    3       2     A
13    3       3     C
14    3       4     D
15    3       5     B
16    4       1     D
17    4       2     E
18    4       3     B
19    4       4     C
20    4       5     A
```





The initial allocation-based mixed model

- The factor allocation diagram is:

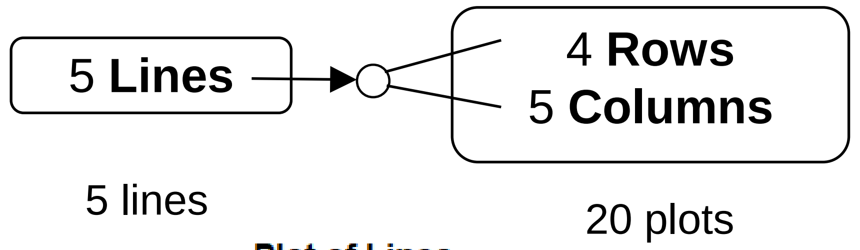


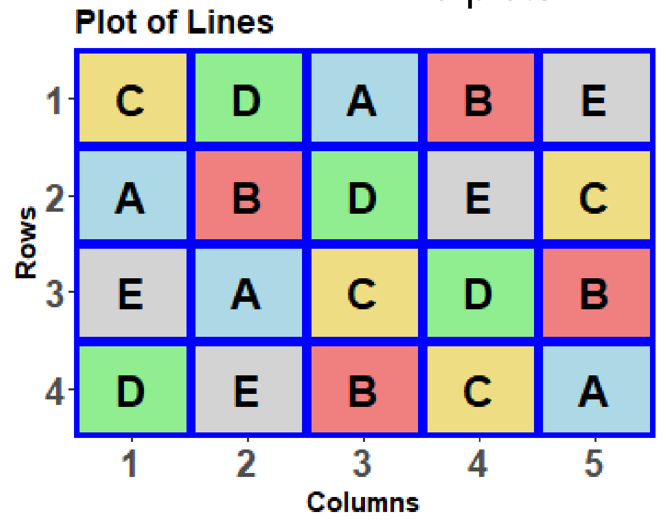
- From the factor allocation diagram, the initial allocation model is:

Lines | Rows + Columns + Rows:Columns

  - This model and the anticipated model are different — here Rows and Columns are random.
  - The Rows and Columns terms could be moved to the fixed model to form a homogeneous allocation model,
    - which in turn may become the prior allocation model.

Working out the confounding





Rows \ Columns	1	2	3	4	5
1	C	D	A	B	E
2	A	B	D	E	C
3	E	A	C	D	B
4	D	E	B	C	A

- What are the recipient (unit) sources?
  - Rows, Columns & Rows#Columns
- Lines will be confounded with which recipient (unit) sources?
  - With Columns & Rows#Columns (cf. LSD).
- Can determine this by investigating the relationships between two sets of projectors, those for lines and those for plots:
  - one source projector for each term in the initial allocation model;
  - $\{\mathbf{Q}_{\mathrm{L}}\}$ and $\{\mathbf{P}_{\mathrm{R}}, \mathbf{P}_{\mathrm{C}}, \mathbf{P}_{\mathrm{R\#C}}\}$.
- Require the eigenvalues of $\mathbf{P}\mathbf{Q}_{\mathrm{L}}\mathbf{P}$ for all 3 **P**s.
- They are calculated and statistical summaries of them are tabulated by **`designAnatomy`**.

Check properties using designAnatomy

```
> YSD.canon <- designAnatomy(formulae = list(plots = ~ Rows*Columns,
+                                            lines = ~ Lines),
+                            data     = YSD.lay)
> summary(YSD.canon)

Summary table of the decomposition for plots & lines (based on adjusted
quantities)

 Source.plots df1 Source.lines df2 aefficiency eefficiency order
 Rows           3
 Columns        4 Lines          4      0.0625      0.0625     1
 Rows#Columns  12 Lines          4      0.9375      0.9375     1
                  Residual       8

The design is not orthogonal
```

but the **order** is one and so the design is balanced

For the first time,

- Lines occurs twice in an analysis;
- neither the aefficiency nor the eefficiency are 1;
- Lines is partially confounded with two sources.

The design’s properties

```
Summary table of the decomposition for plots & lines (based on adjusted
quantities)

 Source.plots df1 Source.lines df2 aefficiency eefficiency order
 Rows           3
 Columns        4 Lines          4      0.0625      0.0625     1
 Rows#Columns  12 Lines          4      0.9375      0.9375     1
                  Residual       8
```

The design is not orthogonal

- All 4 df for Lines are confounded with both `Columns` and `Rows#Columns`;
- None are confounded with `Rows`.

- Thus there are 4 nonzero eigenvalues for $\mathbf{P}_C\mathbf{Q}_L\mathbf{P}_C$ and for $\mathbf{P}_{RC}\mathbf{Q}_L\mathbf{P}_{RC}$:
  - For $\mathbf{P}_C\mathbf{Q}_L\mathbf{P}_C$, all are 0.0625 (1/16);
  - For $\mathbf{P}_{RC}\mathbf{Q}_L\mathbf{P}_{RC}$, all are 0.9375 (15/16);
  - Being 1st-order balanced, the efficiencies sum to 1.
- 15/16 of the information for `Lines` is confounded with `Rows#Columns`.
  - Generally, prefer the intrablock or intrarow-intracolumn efficiency to be greater than, say, 0.75.

To combine or not combine information?

- **`Lines`**, being confounded with **`Columns`** and with **`Rows#Columns`**, there are available two estimates of the **`Lines`** effects:
  - It is expected that those estimated from **`Columns`** differences would have greater variability than those estimated from **`Rows#Columns`**. Why?
- Should these two sets of estimates be combined?
  - In this case, not a lot would be lost by relying on the intrarow-intracolumn estimates: actually, only 1/16 of the information.
  - The advantage is that the more variable inter-column estimates do not contaminate the less variable intrarow-intracolumn estimates.
- In the context of mixed modelling,
  - The combined estimates are produced when **`Columns`** is random.
  - The intrarow-intracolumn estimates are produced when **`Columns`** is fixed.
  - That is, in deciding whether **`Columns`** is fixed or random, consider whether intrablock or combined estimates of **`Lines`** are required.

What if you don’t know what design to use here?

- Look up Cochran and Cox (1957) [C&C] – but they are called incomplete Latin squares, or use **agicolae** (De Mendiburu, 2019).
  - However, you have to know what you the design that you need.
- Use computer searching: **CycDesigN**, **SAS** or **od**.
  - Both the standalone software **CycDesigN** and the **R** package **od** (Butler, 2019) search for a design that minimizes the average variance of pairwise differences (AVPD).
    - **CycDesigN** searches for fixed-model A-optimal designs;
    - **od** searches for mixed-model A-optimal designs;
  - Provided the terms being optimized (treatments) are fixed, these designs are A-optimal because the AVPD equals the PEV.
    - Otherwise, they may not be A-optimal.
  - The harmonic mean of the efficiency factors, the A-efficiency, is proportional to the PEV when the only random term is the residual (or identity) term.
  - **SAS** searches for a D-optimal design.
    - Minimizes the volume of the confidence ellipsoid of estimates (not necessarily A-optimal).
    - The product of the reciprocals of the efficiency factors is minimized.
    - D-optimal designs are used when response curve parameters are to be estimated.

Using od to obtain an optimal design

- The **`od`** function has the following arguments:
  - **`fixed`**, **`random`** and **`residual`** are formulae for specifying the mixed model.
  - **`permute`** is a formula with a single term that is to be optimized by swapping values for the term between rows of its design matrix.
  - **`swap`** is a formula for specifying a term for restricting the permutes to be within its levels.
  - **`search`** specifying a search strategy: **`random`**, **`tabu`** (records rejected designs), **`randomwalk`** (as for random, but accepts a non-improving design with proability *P*) and **`tabu+rw`** (combined).
  - **`maxit`** gives the number of tabu loops or random interchanges.
  - **`start.values`** allows one to specify the values of variance parameters, without beginning a search.
  - **`data`** is a data.frame containing an initial design (obligatory as used to resolve terms in formulae).

Using od to obtain an optimal 4 x 5 grid design

```
> #'### Initialize with a randomized RCBD layout
> R4C5.ini <- cbind(fac.gen(list(Rows=b, Columns=t)),
+                   Lines = factor(rep(1:t, times = b), labels = LETTERS[1:t]))
> R4C5.ini <- designRandomize(allocated         = R4C5.ini["Lines"],
+                             recipient         = R4C5.ini[c("Rows", "Columns")],
+                             nested.recipients = list(Columns = "Rows"),
+                             seed              = 7851)
> #'### Get the od design
> R4C5.od <- od(fixed   = ~ Rows + Columns + Lines,
+               permute = ~ Lines,
+               search  = "tabu", maxit = 25,
+               data    = R4C5.ini)
Done set up; elapsed =   0.00
Initial A-value = 0.952475 (5 A-equations; rank C 4)
A-value after tabu loop 1 is 0.569492
A-value after tabu loop 2 is 0.558333
A-value after tabu loop 3 is 0.533333
...
A-value after tabu loop 25 is 0.533333
Hash table size 30
Final A-value after 25 tabu iterations: 0.533333
Done optimise; elapsed =   0.02
```

```
> R4C5.lay <- R4C5.od$design
> #'### Independently calculate the A-measure
> (designAmeasures(mat.Vpredicts(target = ~ Lines -1,
+                                fixed  = ~ Rows + Columns,
+                                design = R4C5.lay)))
          all
all 0.5333333
```

- The AVPD for the row-column design (0.53) is almost half that for the RCBD (0.95).

The od design

```
> #'### Randomize design according to the plots structure
> R4C5.lay <- designRandomize(allocated  = R4C5.lay["Lines"],
+                             recipient  = R4C5.lay[c("Rows", "Columns")],
+                             seed       = 65460)
```

- This randomization ensures a valid randomization.
  - That is, a randomization that is randomly selected from all possible randomizations.

```
> #'### Calculate the A-measure of the randomized design
> (designAmeasures(mat.Vpredicts(target = ~ Lines -1,
+                                fixed  = ~ Rows + Columns,
+                                design = R4C5.lay)))
          all
all 0.5333333
```

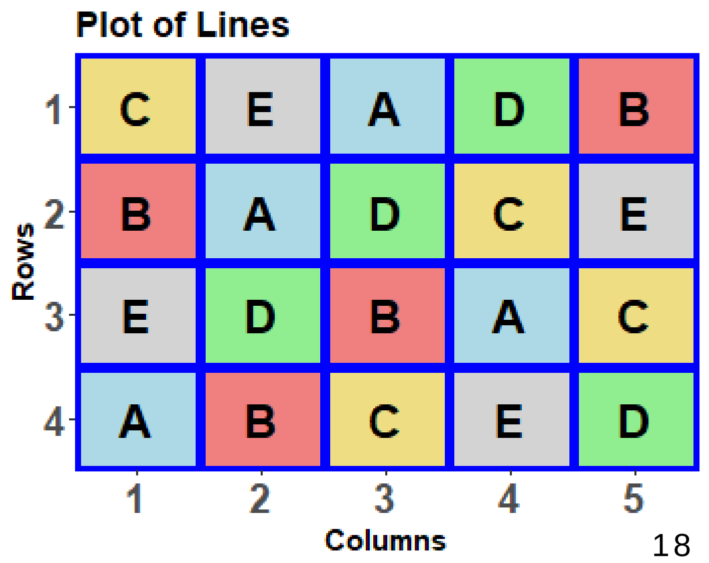
- No change in the AVPD.

The anatomy of the od design

```
> #'### Check properties of the od layout
> R4C5.canon <- designAnatomy(formulae = list(plots = ~ Rows*Columns,
+                                             lines = ~ Lines),
+                                data     = R4C5.lay)
> summary(R4C5.canon)

Summary table of the decomposition for plots & lines (based on adjusted quantities)

 Source.plots df1 Source.lines df2 aefficiency eefficiency order
 Rows           3
 Columns        4 Lines          4      0.0625      0.0625     1
 Rows#Columns  12 Lines          4      0.9375      0.9375     1
                  Residual       8

The design is not orthogonal
```

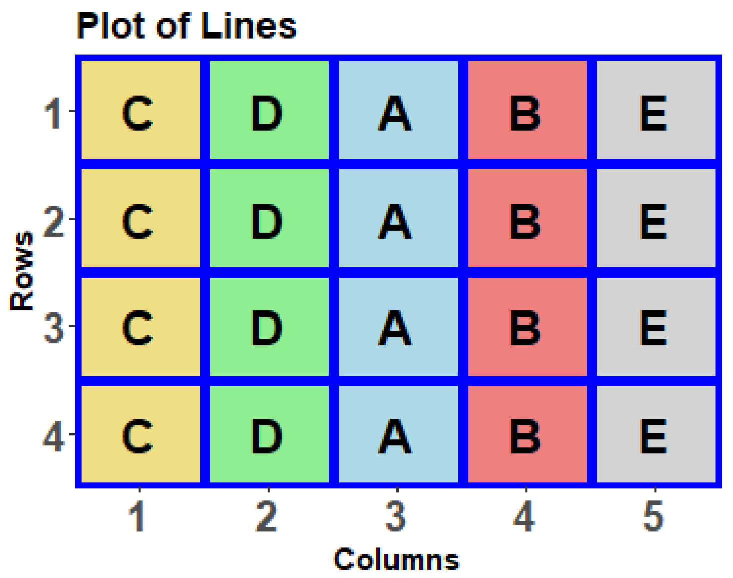
- Same as the Youden square anatomy.

Each treatment occurs in 4 out of 5 columns and so the design is a YSD.



Can choose the wrong starting design

```
> #'## Try a starting design in which row-column randomization is used on a systematic design
> R4C5.ini <- designRandomize(allocated  = data.frame(Lines =
+                                                      factor(rep(1:t, times = b),
+                                                             labels = LETTERS[1:t])),
+                              recipient = list(Rows=b, Columns=t
+                              seed      = 95332)
```



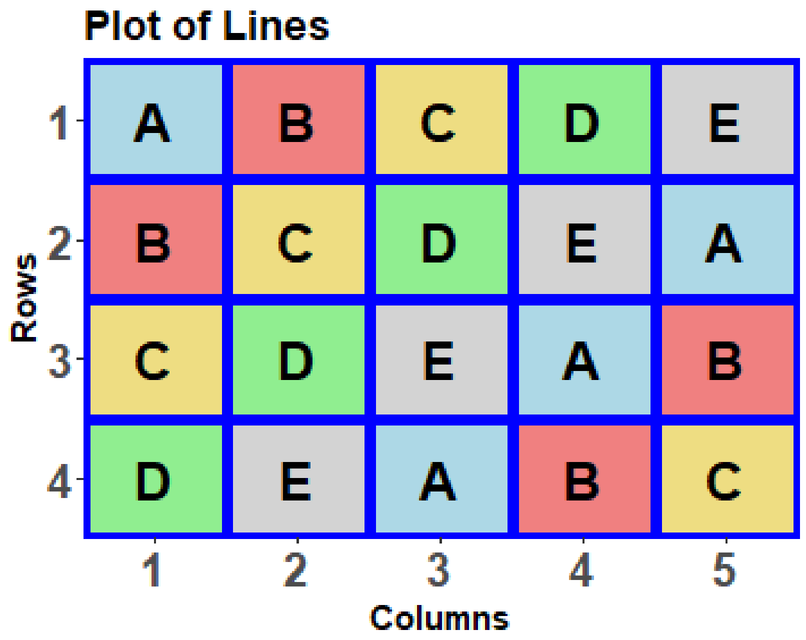
```
> #'### Get the od design
> R4C5.od <- od(fixed   = ~ Rows + Columns + Lines,
+               permute = ~ Lines,
+               search  = "tabu", maxit = 25,
+               data    = R4C5.ini)
Done set up; elapsed =   0.00
Error in od(fixed = ~Rows + Columns + Lines,
permute = ~Lines, search = "tabu",  :
  Disconnected design of order 4
```

- At least some DF for the fixed **`permute`** term are confounded with other fixed terms:
  - All DF **`Lines`** with **`Columns`** here;
- Solution:
  - Choose a connected design (full Lines df at least partially confounded with **`Rows#Columns`**.
  - Or, use a mixed model (not here).

Optimal systematic design in, optimal systematic design out

```
YSD.sys <- cbind(fac.gen(list(Rows=b, Columns=t)),
                 Lines = factor(designLatinSqrSys(t)[1:(b*t)], labels = LETTERS[1:t]))
R4C5.sys.od <- od(fixed   = ~ Rows + Columns + Lines,
                  permute = ~ Lines,
                  search  = "tabu", maxit = 25,
                  data    = YSD.sys)
plotR4C5(R4C5.sys.od$design)
```

Take home message:
od produces an optimal design,
not a randomized design.



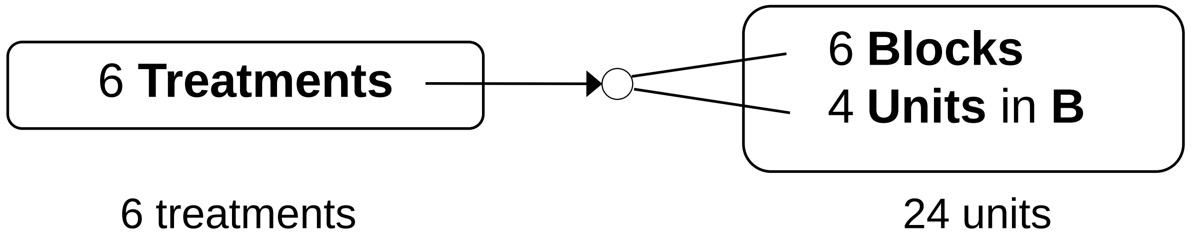
Some points to remember in using od

- The treatment terms cannot be confounded with fixed unit terms:
  - e.g. **`Lines`** confounded with **`Columns`**.
- **`od`** does not necessarily produce a properly randomized design:
  - That is, one randomly selected from all possible randomizations;
  - Supply a systematic optimal design: **`od`** will return it unmodified;
  - Can use **`designRandomize`** after **`od`** when independent errors are assumed; otherwise before **`od`**.
- The computed A-value (AVPD) can be checked, or the value under an alternative model calculated, with **`designAmeasures(mat.Vpredicts(…))`**.
  - **`mat.Vpredicts`** calculates the predictions variance matrix and **`designAmeasures`** calculates the AVPD from the matrix.
- Some designs are optimal under both fixed and random units terms:
  - orthogonal, balanced (incomplete-) block, (most generalized) Youden square and the lattice square designs are A-optimal under fixed and mixed models.

3. Using the concepts for unbalanced designs

3(a) A partially balanced incomplete-block design (PBIBD) from C&C (p.379)

- This design is suitable for a situation in which:
  - the number of treatments is 6,
  - each treatment is to be replicated 4 times,
  - the anticipated model is Treatments | Blocks + Blocks:Units, and
  - the number of units per block restricted to 4.



	Blocks					
Units	I	II	III	IV	V	VI
1	1	2	3	4	5	6
2	4	5	6	1	2	3
3	2	3	1	5	6	4
4	5	6	4	2	3	1

PBIBD randomized layout

```
> #'## Input the systematic design
> b <- 6
> k <- 4
> t <- 6
> PBIBD2.sys <- cbind(fac.gen(list(Blocks = b, Units = k)),
+                     Treatments = factor(c(1,4,2,5,
+                                           2,5,3,6,
+                                           3,6,1,4,
+                                           4,1,5,2,
+                                           5,2,6,3,
+                                           6,3,4,1)))
> #'## Randomize the systematic design
> PBIBD2.lay <- designRandomize(allocated         = PBIBD2.sys["Treatments"],
+                               recipient         = PBIBD2.sys[c("Blocks", "Units")],
+                               nested.recipients = list(Units = "Blocks"),
+                               seed               = 98177)
```

PBIBD properties

```
>#'## Compute the anatomy
> PBIBD2.canon <- designAnatomy(formulae = list(unit = ~ Blocks/Units,
+                                               trt  = ~ Treatments),
+                               data     = PBIBD2.lay)
> summary(PBIBD2.canon, which.criteria = c('aeff', 'xeff', 'eeff','order', 'dforth'))

Summary table of the decomposition for unit & trt (based on adjusted quantities)

 Source.unit   df1 Source.trt df2 aefficiency xefficiency eefficiency order dforthog
 Blocks          5 Treatments   2      0.2500      0.2500      0.2500     1        0
                   Residual     3
 Units[Blocks]  18 Treatments   5      0.8824      1.0000      0.7500     2        3
                   Residual    13

The design is not orthogonal
```

- What are the eigenvalues for $\mathbf{P}_{BU}\mathbf{Q}_{L}\mathbf{P}_{BU}$?
  - Three are one and two are 0.75 for a harmonic mean of 0.8824.
- That 88% of Lines information confounded with Units[Blocks] is good.

PBIBD with unique Units levels

AUnits = All Units

```
> PBIBD2.lay$AUnits <- with(PBIBD2.lay, fac.combine(list(Blocks,Units)))
> levels(PBIBD2.lay$AUnits)
 [1] "1"  "2"  "3"  "4"  "5"  "6"  "7"  "8"  "9"  "10" "11" "12" "13" "14" "15" "16" "17" "18"
[19] "19" "20" "21" "22" "23" "24"
> #'### Blocks + AUnits
> PBIBD2U.canon <- designAnatomy(formulae = list(unit = ~ Blocks + AUnits,
+                                                trt  = ~ Treatments),
+                                data     = PBIBD2.lay)
> summary(PBIBD2U.canon, which.criteria = c('aeff', 'xeff', 'eeff','order', 'dforth'))

Summary table of the decomposition for unit & trt (based on adjusted quantities)

 Source.unit    df1 Source.trt df2 aefficiency xefficiency eefficiency order dforthog
 Blocks           5 Treatments   2      0.2500      0.2500      0.2500     1        0
                    Residual     3
 AUnits[Blocks]  18 Treatments   5      0.8824      1.0000      0.7500     2        3
                    Residual    13
The design is not orthogonal
> #'### Blocks/AUnits
> PBIBD2U.canon <- designAnatomy(formulae = list(unit = ~ Blocks/AUnits,
+                                                trt  = ~ Treatments),
+                                data     = PBIBD2.lay)
```

Produces exactly the same anatomy.

Using od to get an A-optimal design

```
> #'### Initialize with a randomized layout
> PBIBD.ini <- cbind(fac.gen(list(Blocks=b, Units=k)),
+                    Treatments = factor(rep(1:t, times = b*k/t), labels = LETTERS[1:t]))
> PBIBD.ini <- designRandomize(allocated         = PBIBD.ini["Treatments"],
+                              recipient         = PBIBD.ini[c("Blocks", "Units")],
+                              nested.recipients = list(Units = "Blocks"),
+                              seed              = 4794)
> #'### Get the od design
> PBIBD.od <- od(fixed   = ~ Blocks + Treatments,
+                permute = ~ Treatments,
+                search  = "tabu", maxit = 25,
+                data    = PBIBD.ini)
Done set up; elapsed =   0.00
Initial A-value = 0.566667 (6 A-equations; rank C 5)
A-value after tabu loop 1 is 0.559487
A-value after tabu loop 2 is 0.559487
A-value after tabu loop 3 is 0.559487
…
Final A-value after 25 tabu iterations: 0.559487
Done optimise; elapsed =   0.02
```

```
> (designAmeasures(
+   mat.Vpredicts(target = ~ Treatments -1,
+                 fixed  = ~ Blocks,
+                 design = PBIBD2.lay)))
          all
all 0.5666667
```

AVPD for C&C design.

PBIBD od randomization

```
> PBIBD.lay <- PBIBD.od$design
> #'## Randomize the od design
> PBIBD.lay <- designRandomize(allocated         = PBIBD.lay["Treatments"],
+                              recipient         = PBIBD.lay[c("Blocks", "Units")],
+                              nested.recipients = list(Units = "Blocks"),
+                              seed              = 13332)
```

PBIBD od design properties

```
> #'### Check properties of the od layout
> PBIBD.canon <- designAnatomy(formulae = list(plots = ~ Blocks/Units,
+                                              trts  = ~ Treatments),
+                              data      = PBIBD.lay)
> summary(PBIBD.canon, which.criteria = c('aeff', 'xeff', 'eeff','order', 'dforth'))

Summary table of the decomposition for plots & trts (based on adjusted quantities)

 Source.plots  df1 Source.trts df2 aefficiency xefficiency eefficiency order dforthog
 Blocks          5 Treatments    4      0.0937      0.1875      0.0625     2        0
                   Residual      1
 Units[Blocks]  18 Treatments    5      0.8937      1.0000      0.8125     3        1
                   Residual     13

The design is not orthogonal
```

- The **od** design is (nearer) A-optimal, with
  - a higher A-efficiency than the PBIBD2 (0.8937 versus 0.8824),
  - three rather than two different efficiency factors,
  - the range of the efficiency value is less (min of 0.75 versus 0.8125).

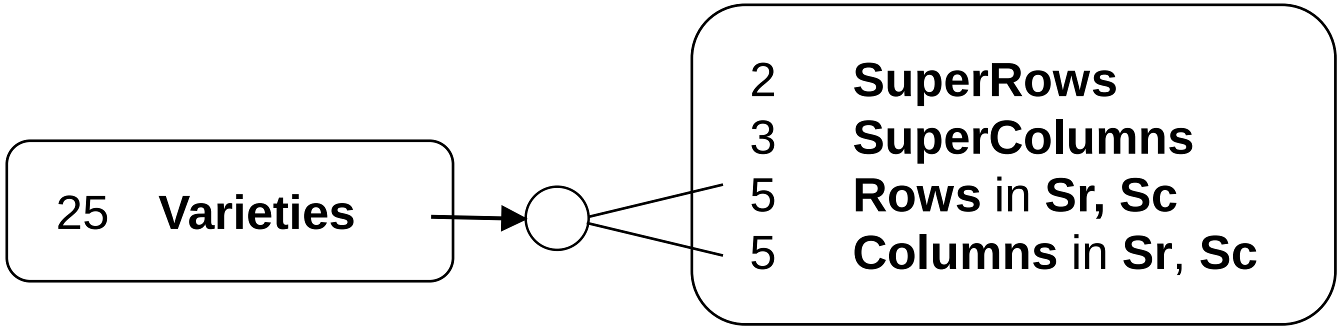
This shows that, in contrast to a BIBD, a PBIBD of order 2 is not necessarily A-optimal.
Which one to use?

- The PBIBD2 will have only 2 SEM values and so 3 SEDs.
- The **od** design would have 6 SEDs, but they would cover a narrower range.

More than A-value to consider.

3(b) A wheat experiment (Gilmour et al., 1995)

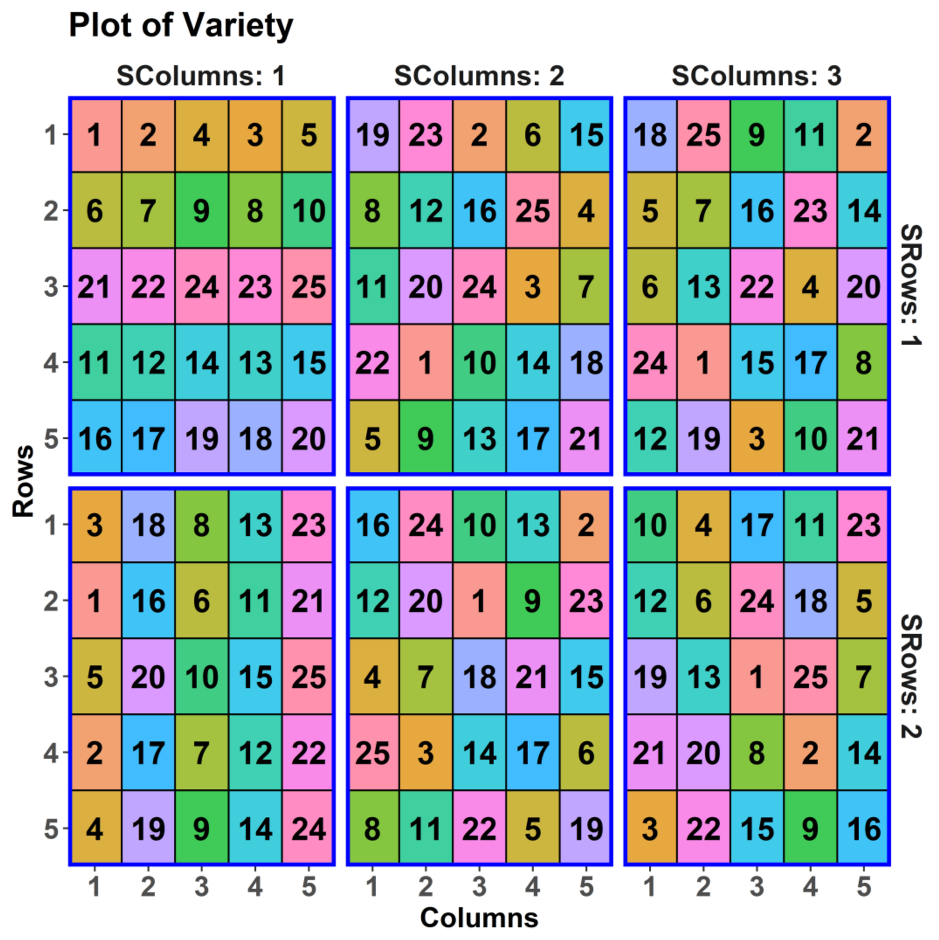
- Investigates 25 varieties of wheat.
- A balanced lattice square on a 10 $\times$ 15 grid from C&C.
- Six reps, each 5 rows $\times$ 5 columns
- It is an example in the **`asreml`** manual, and the **`asremlPlus`** manual and the Wheat vignette: **`vignette(package = 'asremlPlus')`**.
- Factor-allocation diagram



25 varieties

150 plots

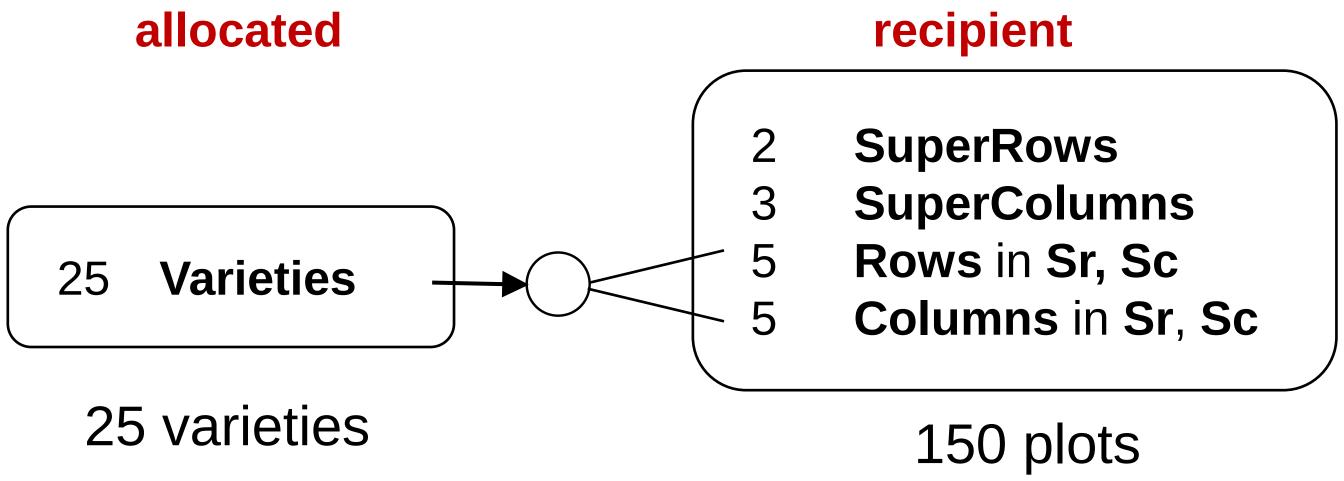
Sr = SuperRows, Sc = SuperColumns



SRows	Rows	SC1 C1	SC1 C2	SC1 C3	SC1 C4	SC1 C5	SC2 C1	SC2 C2	SC2 C3	SC2 C4	SC2 C5	SC3 C1	SC3 C2	SC3 C3	SC3 C4	SC3 C5
1	1	1	2	4	3	5	19	23	2	6	15	18	25	9	11	2
1	2	6	7	9	8	10	8	12	16	25	4	5	7	16	23	14
1	3	21	22	24	23	25	11	20	24	3	7	6	13	22	4	20
1	4	11	12	14	13	15	22	1	10	14	18	24	1	15	17	8
1	5	16	17	19	18	20	5	9	13	17	21	12	19	3	10	21
2	1	3	18	8	13	23	16	24	10	13	2	10	4	17	11	23
2	2	1	16	6	11	21	12	20	1	9	23	12	6	24	18	5
2	3	5	20	10	15	25	4	7	18	21	15	19	13	1	25	7
2	4	2	17	7	12	22	25	3	14	17	6	21	20	8	2	14
2	5	4	19	9	14	24	8	11	22	5	19	3	22	15	9	16

How is this design to be randomized?

The wheat experiment — models



- Allocated ⇒ fixed;
  Recipient ⇒ random.
- Take all combinations of the factors within a panel, subject to the restriction that a nested factor cannot occur without its nesting factor.

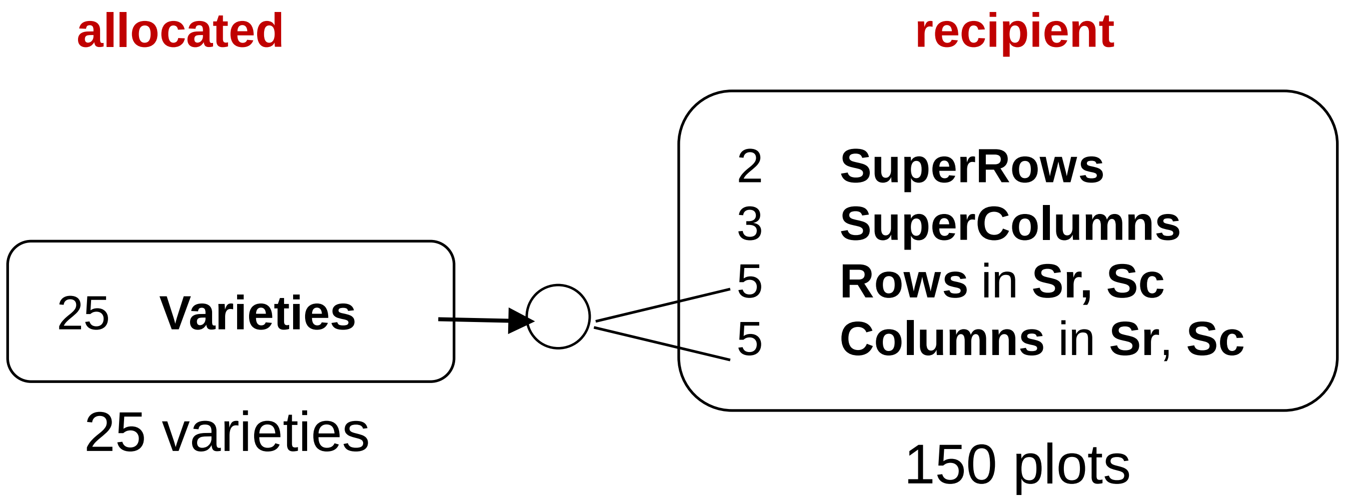
- Initial allocation model:
  - Varieties | SRows + SColumns + SRows:SColumns +
    SRows:SColumns:Rows + SRows:SColumns:Columns +
    <u>SRows:SColumns:Rows:Columns</u>.

A randomization model

An identity (or residual) term – it uniquely indexes the units.

- The balanced lattice square is A-optimal for this model.
- No term for differences between whole rows and whole columns, because not randomized by them.
  - If had, then structure (SRows/Rows) * (SColumns/Columns), not (SRows*SColumns) / (Rows*Columns).

The wheat experiment – models (revised)



- Initial allocation model:
  - Varieties | SRows*SColumns + SRows:SColumns:Rows + SRows:SColumns:Columns + SRows:SColumns:Rows:Columns.
- Homogeneous allocation model:
  - Might make SRows*SColumns fixed.
- Prior allocation model:
  - Varieties | SRows*SColumns + SRows:SColumns:Rows + SRows:SColumns:Columns + units + ar1(SRows:Rows):ar1(SColumns:Columns).

nugget variance

spatial residual correlation

- The prior allocation model is not a randomization model, but a randomization-based model.

The wheat experiment - properties

Based on initial or homogeneous allocation model, with SRows*SColumns combined.

```
> Wheat.canon <- designAnatomy(formulae = list(units = ~ (SRows:SColumns)/(Rows*Columns),
+                                               trt   = ~ Variety),
+                                 data      = Wheat.dat)
> summary(Wheat.canon, which.criteria = c("aeff", "order"))
Summary table of the decomposition for units & trt (based on adjusted quantities)
 Source.units                    df1 Source.trt df2 aefficiency order
 SRows:SColumns                    5
 Rows[SRows:SColumns]             24 Variety     24      0.1667     1
 Columns[SRows:SColumns]          24 Variety     24      0.1667     1
 Rows#Columns[SRows:SColumns]     96 Variety     24      0.6667     1
                                     Residual    72
The design is not orthogonal
```

However, Gilmour et al. (1995) and Butler et al. (2018) have ignored SRows:SColumns (Reps). What happens?

```
> Wheat.RC.canon <- designAnatomy(formulae = list(units = ~ ARows*AColumns,
+                                                  trt   = ~ Variety),
+                                    data      = Wheat.dat)
> summary(Wheat.RC.canon)
Summary table of the decomposition for units & trt (based on adjusted quantities)
Source.units   df1 Source.trt df2 aefficiency eefficiency order
 ARows           9 Variety      8      0.1667      0.1667     1
                   Residual     1
 AColumns       14 Variety     12      0.1667      0.1667     1
                   Residual     2
 ARows#AColumns 126 Variety    24      0.8452      0.6732    18
                   Residual   102
```

ARow = SRows:Rows; (A = All)
AColumn = SColumns:Columns.

Not randomization-based:

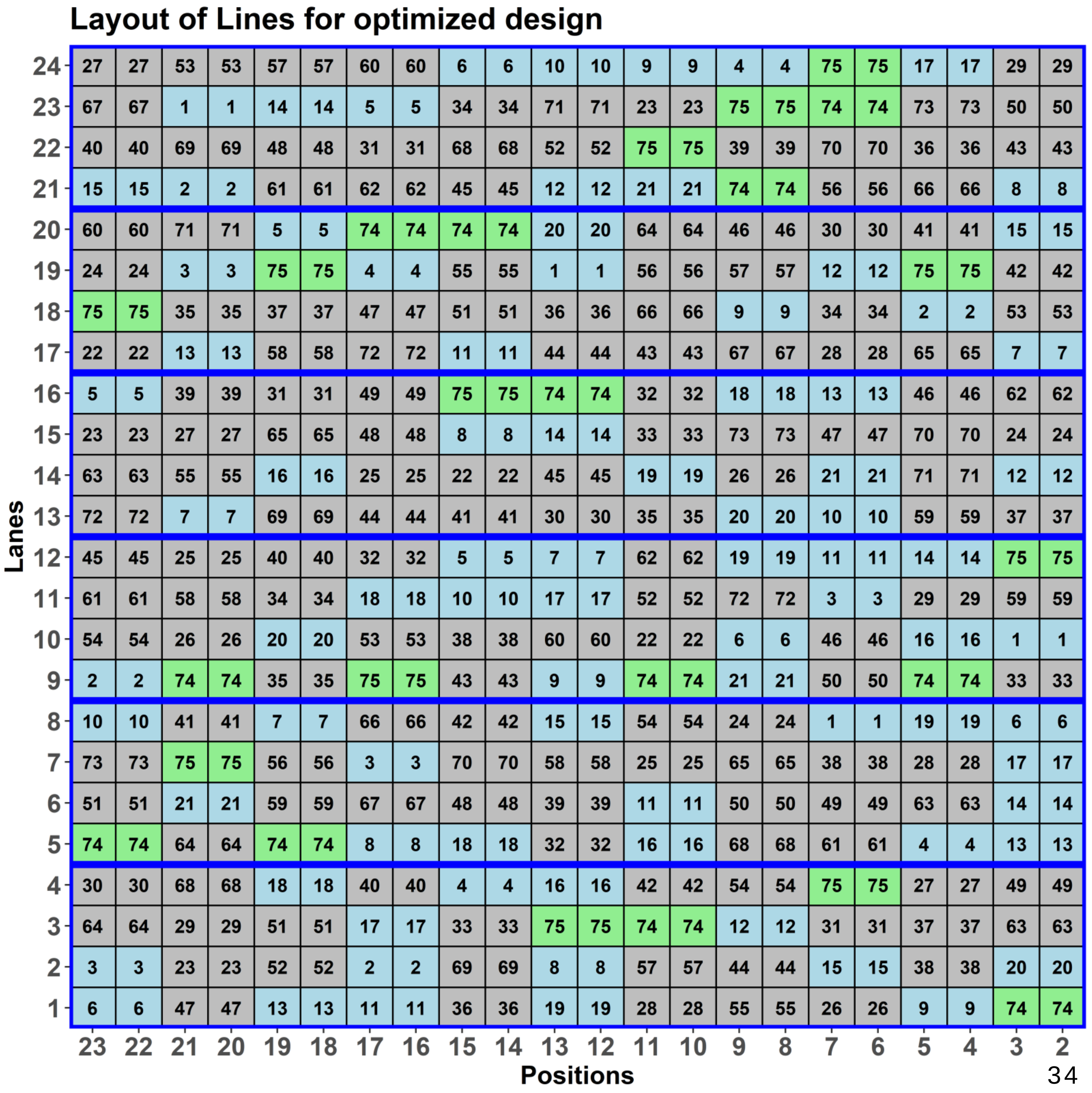
- Pushes down the 53 DF of the first 3 sources from the lattice:
  - into all units sources;
  - some in ARow#ACol Residual.
- More Variety information confounded with Row#Col, but...

A-optimality of the design

- The resolved design has the advantage that SRows*Scolumns (Replicate) differences do not contribute to the variability of the Varieties.
- It is the A-optimal resolved design.
- It is not the A-optimal row-column design, i.e. under the model:
  - ARows + AColumns + Varieties | Rows:Columns.
- Nor is it A-optimal for the prior allocation model with :
  - Varieties | SRows*SColumns + SRows:SColumns:Rows + SRows:SColumns:Columns + units + ar1(SRows:Rows):ar1(SColumns:Columns).
- For these alternative models, use **od** to search for (near) A-optimal designs.

3(c) A Plant Accelerator (PA) design

- Split-unit design from **od**.
- 75 lines assigned to main units (2 carts) using a blocked, row-column design:
  - 6 blocks of 4 Lanes;
  - 21 NAM lines (blue) on 4 main units each;
  - 52 NAM lines (grey) on 3 main units each;
  - Scout & Gladius (green) on 12 main units each.
- 2 Conditions randomized to pairs of carts (not shown).

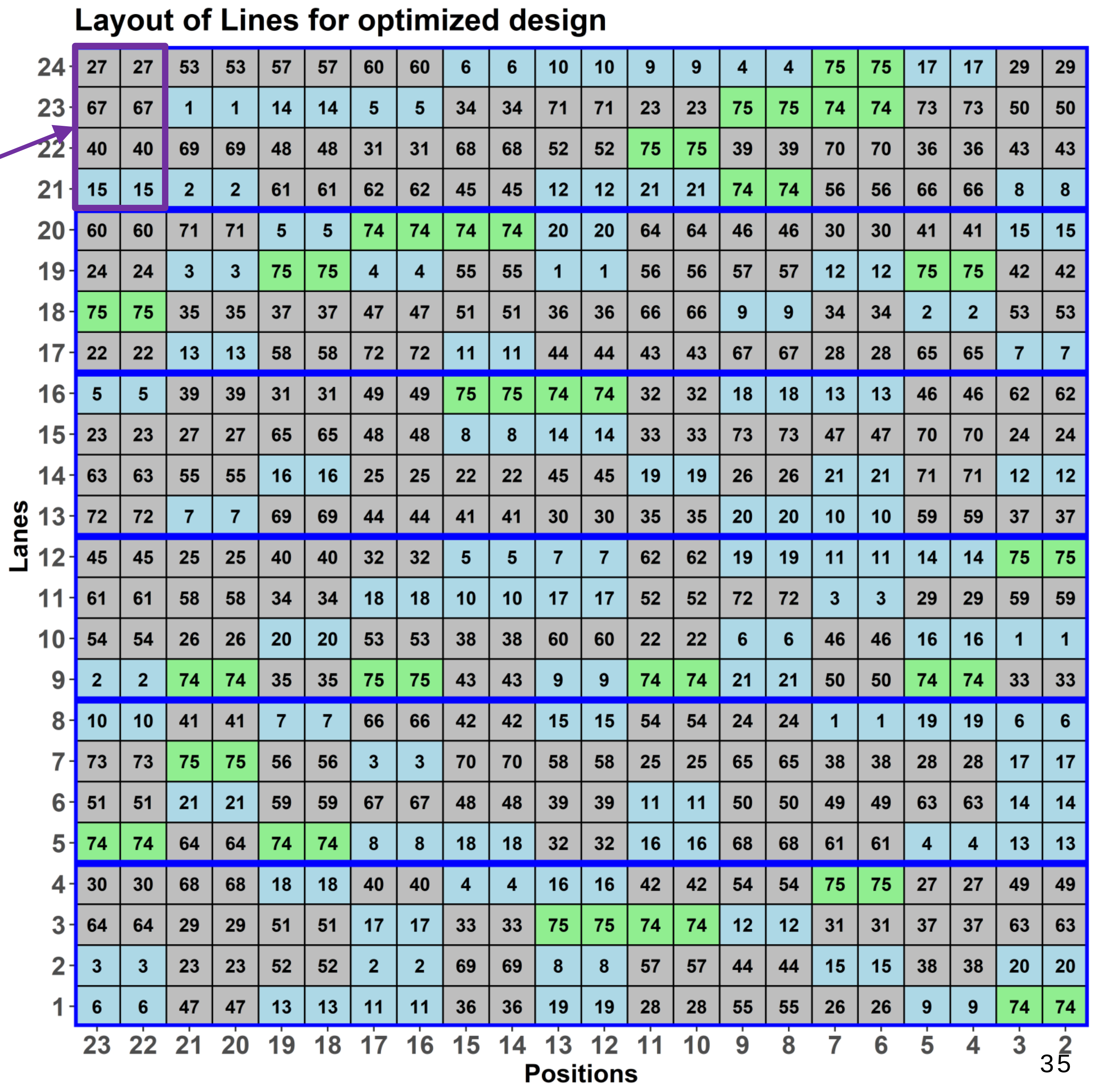

Layout of Lines for optimized design

Lanes \ Positions	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2
24	27	27	53	53	57	57	60	60	6	6	10	10	9	9	4	4	75	75	17	17	29	29
23	67	67	1	1	14	14	5	5	34	34	71	71	23	23	75	75	74	74	73	73	50	50
22	40	40	69	69	48	48	31	31	68	68	52	52	75	75	39	39	70	70	36	36	43	43
21	15	15	2	2	61	61	62	62	45	45	12	12	21	21	74	74	56	56	66	66	8	8
20	60	60	71	71	5	5	74	74	74	74	20	20	64	64	46	46	30	30	41	41	15	15
19	24	24	3	3	75	75	4	4	55	55	1	1	56	56	57	57	12	12	75	75	42	42
18	75	75	35	35	37	37	47	47	51	51	36	36	66	66	9	9	34	34	2	2	53	53
17	22	22	13	13	58	58	72	72	11	11	44	44	43	43	67	67	28	28	65	65	7	7
16	5	5	39	39	31	31	49	49	75	75	74	74	32	32	18	18	13	13	46	46	62	62
15	23	23	27	27	65	65	48	48	8	8	14	14	33	33	73	73	47	47	70	70	24	24
14	63	63	55	55	16	16	25	25	22	22	45	45	19	19	26	26	21	21	71	71	12	12
13	72	72	7	7	69	69	44	44	41	41	30	30	35	35	20	20	10	10	59	59	37	37
12	45	45	25	25	40	40	32	32	5	5	7	7	62	62	19	19	11	11	14	14	75	75
11	61	61	58	58	34	34	18	18	10	10	17	17	52	52	72	72	3	3	29	29	59	59
10	54	54	26	26	20	20	53	53	38	38	60	60	22	22	6	6	46	46	16	16	1	1
9	2	2	74	74	35	35	75	75	43	43	9	9	74	74	21	21	50	50	74	74	33	33
8	10	10	41	41	7	7	66	66	42	42	15	15	54	54	24	24	1	1	19	19	6	6
7	73	73	75	75	56	56	3	3	70	70	58	58	25	25	65	65	38	38	28	28	17	17
6	51	51	21	21	59	59	67	67	48	48	39	39	11	11	50	50	49	49	63	63	14	14
5	74	74	64	64	74	74	8	8	18	18	32	32	16	16	68	68	61	61	4	4	13	13
4	30	30	68	68	18	18	40	40	4	4	16	16	42	42	54	54	75	75	27	27	49	49
3	64	64	29	29	51	51	17	17	33	33	75	75	74	74	12	12	31	31	37	37	63	63
2	3	3	23	23	52	52	2	2	69	69	8	8	57	57	44	44	15	15	38	38	20	20
1	6	6	47	47	13	13	11	11	36	36	19	19	28	28	55	55	26	26	9	9	74	74

The anticipated model

Zones-MainPositions cell

- Zones + Lines + Conditions + Lines:Conditions | MainPositions + Zones:MainPositions + Zones:MainPositions:Rows + Zones:MainPositions:Rows:Carts.
  - Zones are the blocks of 4 Lanes;
  - MainPositions are the columns of pairs of carts;
  - Rows are the 4 lanes within a Zone;
  - Zones:MainPositions:Rows are the main units.
  - Carts are the pairs of Carts within a Zones:MainPositions:Rows combination; they are the sub-units.



Layout of Lines for optimized design

Lanes	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2
24	27	27	53	53	57	57	60	60	6	6	10	10	9	9	4	4	75	75	17	17	29	29
23	67	67	1	1	14	14	5	5	34	34	71	71	23	23	75	75	74	74	73	73	50	50
22	40	40	69	69	48	48	31	31	68	68	52	52	75	75	39	39	70	70	36	36	43	43
21	15	15	2	2	61	61	62	62	45	45	12	12	21	21	74	74	56	56	66	66	8	8
20	60	60	71	71	5	5	74	74	74	74	20	20	64	64	46	46	30	30	41	41	15	15
19	24	24	3	3	75	75	4	4	55	55	1	1	56	56	57	57	12	12	75	75	42	42
18	75	75	35	35	37	37	47	47	51	51	36	36	66	66	9	9	34	34	2	2	53	53
17	22	22	13	13	58	58	72	72	11	11	44	44	43	43	67	67	28	28	65	65	7	7
16	5	5	39	39	31	31	49	49	75	75	74	74	32	32	18	18	13	13	46	46	62	62
15	23	23	27	27	65	65	48	48	8	8	14	14	33	33	73	73	47	47	70	70	24	24
14	63	63	55	55	16	16	25	25	22	22	45	45	19	19	26	26	21	21	71	71	12	12
13	72	72	7	7	69	69	44	44	41	41	30	30	35	35	20	20	10	10	59	59	37	37
12	45	45	25	25	40	40	32	32	5	5	7	7	62	62	19	19	11	11	14	14	75	75
11	61	61	58	58	34	34	18	18	10	10	17	17	52	52	72	72	3	3	29	29	59	59
10	54	54	26	26	20	20	53	53	38	38	60	60	22	22	6	6	46	46	16	16	1	1
9	2	2	74	74	35	35	75	75	43	43	9	9	74	74	21	21	50	50	74	74	33	33
8	10	10	41	41	7	7	66	66	42	42	15	15	54	54	24	24	1	1	19	19	6	6
7	73	73	75	75	56	56	3	3	70	70	58	58	25	25	65	65	38	38	28	28	17	17
6	51	51	21	21	59	59	67	67	48	48	39	39	11	11	50	50	49	49	63	63	14	14
5	74	74	64	64	74	74	8	8	18	18	32	32	16	16	68	68	61	61	4	4	13	13
4	30	30	68	68	18	18	40	40	4	4	16	16	42	42	54	54	75	75	27	27	49	49
3	64	64	29	29	51	51	17	17	33	33	75	75	74	74	12	12	31	31	37	37	63	63
2	3	3	23	23	52	52	2	2	69	69	8	8	57	57	44	44	15	15	38	38	20	20
1	6	6	47	47	13	13	11	11	36	36	19	19	28	28	55	55	26	26	9	9	74	74

Positions

Check properties of the PA design

```
> PA.canon <- designAnatomy(formulae = list(carts = ~ (Zones*MainPositions)/Rows/Carts,
+                                           trts = ~ Lines * Conditions),
+                           data = PA.lay)
> summary(PA.canon, which=c("aeff", "eeff", "order", "dfor"))

Summary table of the decomposition for carts & trts (based on adjusted quantities)

 Source.carts                    df1 Source.trts      df2 aefficiency eefficiency order dforthog
 Zones                             5 Lines              5      0.1497      0.1254     5        0
 MainPositions                    10 Lines             10      0.2101      0.1724    10        0
 Zones#MainPositions              50 Lines             50      0.1209      0.0193    50        0
 Rows[Zones:MainPositions]       198 Lines             74      0.6764      0.2746    66        9
                                     Residual         124
 Carts[Zones:MainPositions:Rows] 264 Conditions         1      1.0000      1.0000     1        1
                                     Lines#Conditions  74      1.0000      1.0000     1       74
                                     Residual         189
The design is not orthogonal
```

- The information about **`Lines`** confounded with **`Rows[Zones:MainPositions]`** is low.
  - However, all 74 df for Lines confounded with it and so Lines is connected.
- It is anticipated that the differences between **`MainPositions`** can be described in terms of a linear trend across **`MainPositions`** and that **`Zones:MainPositions`** can be ignored.
  - Could optimize for linear trend by replacing **`MainPositions`** with a centred numeric covariate, say **`xMainPosn`**.
  - This tends to push extra replicates to the first and last MainPositions, which is not optimal for curved trends.
  - So optimize for factor **`MainPositions`** and check properties for numeric covariate **`xMainPosn`**.

Linear trend across MainPositions

Replace 11 **MainPositions** with 44 **MainUnits** and include a covariate.

```
> PA.xMainPosn.canon <- designAnatomy(list(cart=~ Zones/MainUnits/Carts,
+                                          treat=~ xMainPosn + Lines * Conditions),
+                                     data = PA.lay)
> summary(PA.xMainPosn.canon, which=c("aeff", "eeff", "order", "dfor"))

Summary table of the decomposition for cart & treat (based on adjusted quantities)

 Source.cart             df1 Source.treat     df2 aefficiency eefficiency order dforthog
 Zones                     5 Lines              5      0.1500      0.1255     5        0
 Mainunits[Zones]        258 xMainPosn          1      1.0000      1.0000     1        1
                             Lines             74      0.9879      0.8217     6       69
                             Residual         183
 Carts[Zones:Mainunits]  264 Conditions         1      1.0000      1.0000     1        1
                             Lines#Conditions  74      1.0000      1.0000     1       74
                             Residual         189

Table of (partial) aliasing between sources derived from the same formula

 Source            df  Alias     In    aefficiency eefficiency order dforthog
 Lines             74 xMainPosn treat      0.9960      0.7687     2       73
 Lines#Conditions 149 xMainPosn treat      0.9980      0.7687     2      148

The design is not orthogonal
```

Clearly, Lines is not orthogonal to a linear trend in **xMainPosn.** The aliasing is moderate (23% of one Lines df is lost).

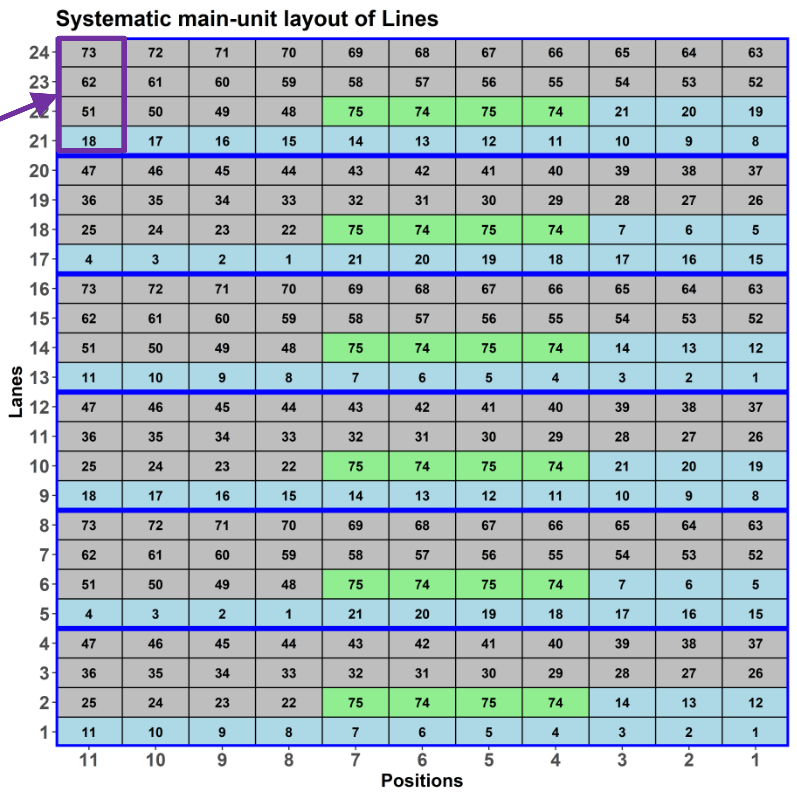
More importantly the majority of the rest of the information about Lines, is available from **Rows[Zones:MainPositions]** (main units); (at least 82%; on average 98.8%).

Using od to get a design — initial main-unit design

Zones-MainPositions cell

- Aim to balance between 6 Zones the numbers of
  - RILs (1:21) replicated 4 times (blue),
  - Parents (74:75) replicated 12 times (green),
  - RILs (22:73) replicated 3 times (grey).

```
> b <- 6
> r <- 4
> c <- 11
> maxit <-25
> search <- "tabu“
> main.sys <- cbind(fac.gen(list(Zones = b, Rows = r, MainPositions = c)),
+                   Lines = factor(c(1:14, 74:75, 74:75, 22:47,        #Z1
+                                    15:21, 1:7, 74:75, 74:75, 48:73,  #Z2
+                                    8:21, 74:75, 74:75, 22:47,        #Z3
+                                    1:14, 74:75, 74:75, 48:73,        #Z4
+                                    15:21, 1:7, 74:75, 74:75, 22:47,  #Z5
+                                    8:21, 74:75, 74:75, 48:73)))      #Z6
> #'### Randomize systematic design
> main.sys <- designRandomize(allocated = main.sys["Lines"],
+                             recipient = main.sys[c("Zones", "Rows", "MainPositions")],
+                             nested.recipients = list(Rows = c("Zones", "MainPositions")),
+                             seed = 71598)
```

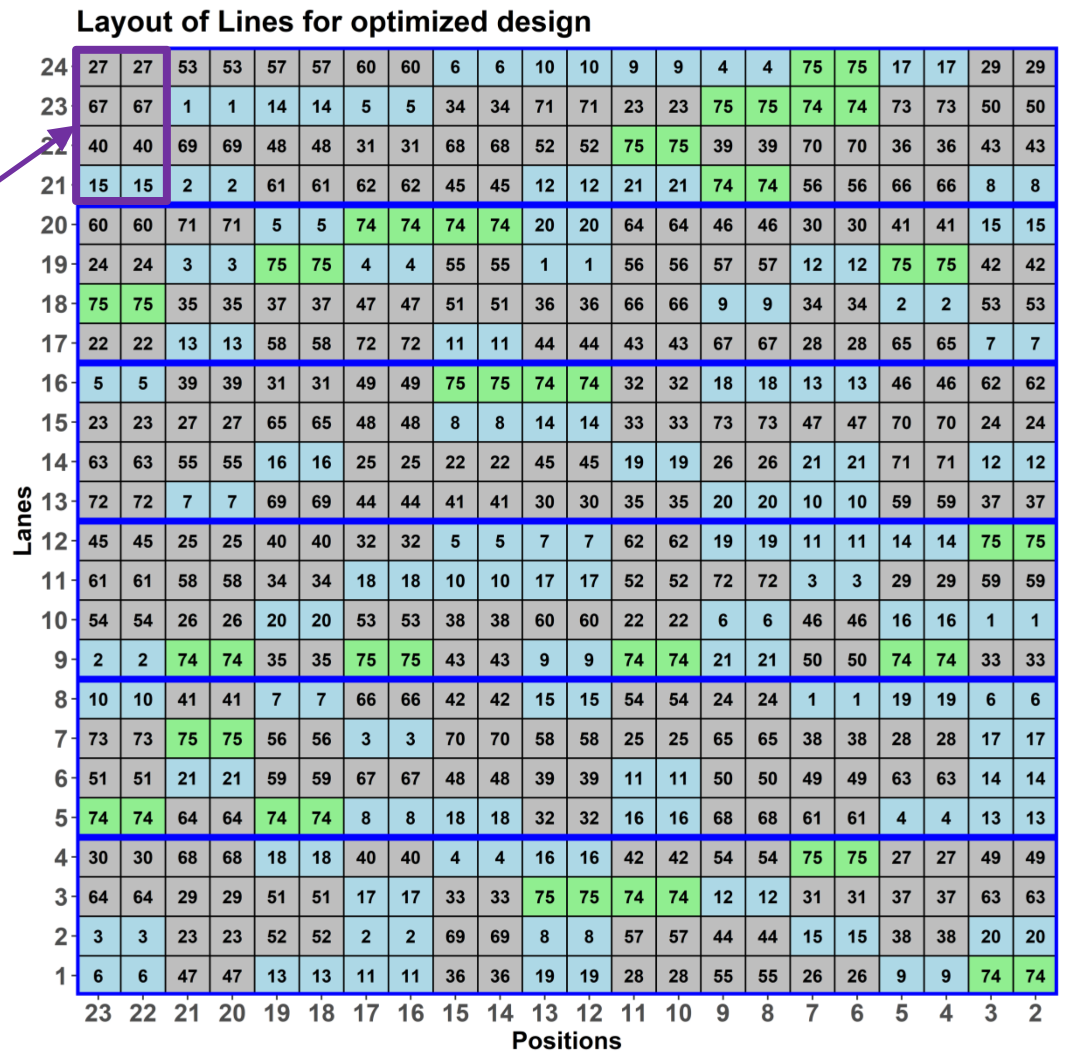


Lanes	11	10	9	8	7	6	5	4	3	2	1
24	73	72	71	70	69	68	67	66	65	64	63
23	62	61	60	59	58	57	56	55	54	53	52
22	51	50	49	48	75	74	75	74	21	20	19
21	18	17	16	15	14	13	12	11	10	9	8
20	47	46	45	44	43	42	41	40	39	38	37
19	36	35	34	33	32	31	30	29	28	27	26
18	25	24	23	22	75	74	75	74	7	6	5
17	4	3	2	1	21	20	19	18	17	16	15
16	73	72	71	70	69	68	67	66	65	64	63
15	62	61	60	59	58	57	56	55	54	53	52
14	51	50	49	48	75	74	75	74	14	13	12
13	11	10	9	8	7	6	5	4	3	2	1
12	47	46	45	44	43	42	41	40	39	38	37
11	36	35	34	33	32	31	30	29	28	27	26
10	25	24	23	22	75	74	75	74	21	20	19
9	18	17	16	15	14	13	12	11	10	9	8
8	73	72	71	70	69	68	67	66	65	64	63
7	62	61	60	59	58	57	56	55	54	53	52
6	51	50	49	48	75	74	75	74	7	6	5
5	4	3	2	1	21	20	19	18	17	16	15
4	47	46	45	44	43	42	41	40	39	38	37
3	36	35	34	33	32	31	30	29	28	27	26
2	25	24	23	22	75	74	75	74	14	13	12
1	11	10	9	8	7	6	5	4	3	2	1

The balancing is not essential, but an attempt to ensure that a balanced design is considered.

Recall the anticipated model

Zones-MainPositions cell


Layout of Lines for optimized design

Lanes	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2
24	27	27	53	53	57	57	60	60	6	6	10	10	9	9	4	4	75	75	17	17	29	29
23	67	67	1	1	14	14	5	5	34	34	71	71	23	23	75	75	74	74	73	73	50	50
22	40	40	69	69	48	48	31	31	68	68	52	52	75	75	39	39	70	70	36	36	43	43
21	15	15	2	2	61	61	62	62	45	45	12	12	21	21	74	74	56	56	66	66	8	8
20	60	60	71	71	5	5	74	74	74	74	20	20	64	64	46	46	30	30	41	41	15	15
19	24	24	3	3	75	75	4	4	55	55	1	1	56	56	57	57	12	12	75	75	42	42
18	75	75	35	35	37	37	47	47	51	51	36	36	66	66	9	9	34	34	2	2	53	53
17	22	22	13	13	58	58	72	72	11	11	44	44	43	43	67	67	28	28	65	65	7	7
16	5	5	39	39	31	31	49	49	75	75	74	74	32	32	18	18	13	13	46	46	62	62
15	23	23	27	27	65	65	48	48	8	8	14	14	33	33	73	73	47	47	70	70	24	24
14	63	63	55	55	16	16	25	25	22	22	45	45	19	19	26	26	21	21	71	71	12	12
13	72	72	7	7	69	69	44	44	41	41	30	30	35	35	20	20	10	10	59	59	37	37
12	45	45	25	25	40	40	32	32	5	5	7	7	62	62	19	19	11	11	14	14	75	75
11	61	61	58	58	34	34	18	18	10	10	17	17	52	52	72	72	3	3	29	29	59	59
10	54	54	26	26	20	20	53	53	38	38	60	60	22	22	6	6	46	46	16	16	1	1
9	2	2	74	74	35	35	75	75	43	43	9	9	74	74	21	21	50	50	74	74	33	33
8	10	10	41	41	7	7	66	66	42	42	15	15	54	54	24	24	1	1	19	19	6	6
7	73	73	75	75	56	56	3	3	70	70	58	58	25	25	65	65	38	38	28	28	17	17
6	51	51	21	21	59	59	67	67	48	48	39	39	11	11	50	50	49	49	63	63	14	14
5	74	74	64	64	74	74	8	8	18	18	32	32	16	16	68	68	61	61	4	4	13	13
4	30	30	68	68	18	18	40	40	4	4	16	16	42	42	54	54	75	75	27	27	49	49
3	64	64	29	29	51	51	17	17	33	33	75	75	74	74	12	12	31	31	37	37	63	63
2	3	3	23	23	52	52	2	2	69	69	8	8	57	57	44	44	15	15	38	38	20	20
1	6	6	47	47	13	13	11	11	36	36	19	19	28	28	55	55	26	26	9	9	74	74

Positions

- Zones + Lines + Conditions + Lines:Conditions | MainPositions + Zones:MainPositions + Zones:MainPositions:Rows + Zones:MainPositions:Rows:Carts.
- Several random terms.
  - By default **od** assumes that the variance component for Zones:MainPositions:Rows:Carts is one and the rest are 0.1 times it.
    - That is, other than the residual, the components are small.
    - Suppose this is OK, except that MainPositions is likely to be 0.5.
    - Zones fixed is equivalent to assuming that the variance component is infinite.

Using od to set variance parameters for the main-unit design

```
> #'### Set variance parameters
> main.ini <- od(fixed = ~ Zones + Lines,
+                random = ~ MainPositions + Zones:(Rows + MainPositions),
+                permute = ~ Lines,
+                start.values = TRUE,
+                data = main.sys)
> vp.table <- main.ini$vparameters.table
> vp.table$Value[1] <- 0.5
> (vp.table)
           Component Value
1      MainPositions   0.5
2         Zones:Rows   0.1
3 Zones:MainPositions   0.1
4            units!R   1.0
```

With this argument, returns an object that includes a variance parameter table.

MainPositions will be set to 0.5, as desired.

Using od to get a near-A-optimal main-unit design

```
> #'### Optimize
> main.od <- od(fixed = ~ Zones + Lines,
+               random = ~ MainPositions + Zones:(Rows + MainPositions),
+               permute = ~ Lines,
+               maxit = maxit, search = search,
+               G.param = vp.table,
+               data = main.sys)
Done set up; elapsed =   0.00
Initial A-value = 1.035331 (75 A-equations; rank C 74)
A-value after tabu loop 1 is 0.707094
A-value after tabu loop 2 is 0.706797
A-value after tabu loop 3 is 0.706732
...
A-value after tabu loop 25 is 0.706413
Hash table size 678
Final A-value after 25 tabu iterations: 0.706413
Done optimise; elapsed =   4.65
> main.lay <- main.od$design
```

Need to supply the variance parameter table to the `G.param` argument.

How does the mixed-model design compare with a fixed-model design for the mixed model?

```
> main.fix.od <- od(fixed = ~ Zones*MainPositions + Zones:Rows + Lines,
+                   permute = ~ Lines,
+                   maxit = maxit, search = search,
+                   data = main.sys)
Done set up; elapsed =   0.00
Initial A-value = 1.664428 (75 A-equations; rank C 73)
A-value after tabu loop 1 is 0.929683
…
A-value after tabu loop 25 is 0.912422
Hash table size 555
Final A-value after 25 tabu iterations: 0.912422
Done optimise; elapsed =   4.62
> #'### Calculate A-measure under mixed model
> main.fix.lay <- main.fix.od$design
> designAmeasures(mat.Vpredicts(target = ~ Lines - 1,
+                               fixed = ~ Zones -1,
+                               random = ~ MainPositions + Zones:(Rows + MainPositions) - 1,
+                               G = as.list(vp.table$Value[-4]),
+                               design = main.fix.lay))
          all
all 0.7455769
```

This compares with 0.706043, and is 1.06 times the mixed-model design.
The sed would only be slightly inflated (3%).

Expand the main-unit design to a split-unit design

```
> #'### Expand main-unit design to add Carts with Conditions
> PA.sys <- cbind(fac.gen(list(Zones = b, Rows = r, MainPositions = c, Carts = 2)),
+                 data.frame(Lines      = factor(rep(main.lay$Lines, each=2), levels=1:75),
+                            Conditions = factor(rep(1:2, times=264),
+                                                labels = c('0 NaCl','100 NaCl'))))
> #'### Randomize the whole design
> PA.lay <- designRandomize(allocated = PA.sys[c("Lines", "Conditions")],
+                           recipient = PA.sys[c("Zones", "Rows", "MainPositions",
+                                                "Carts")],
+                           nested.recipients = list(Rows = c("Zones", "MainPositions"),
+                                                Carts = c("Zones", "Rows",
+                                                          "MainPositions")),
+                           seed = 51412)
> PA.lay <- cbind(fac.gen(list(Lanes = nlanes, Positions = posns)),
+                 PA.lay)
> #'### Add factors and variates
> PA.lay <- within(PA.lay,
+                 {
+                   xMainPosn <- as.numfac(MainPositions)
+                   xMainPosn <- -(xMainPosn - mean(xMainPosn))
+                   MainUnits <- fac.combine(list(Rows, MainPositions))
+                 })
```

Repermute **Rows**

Permute **Carts** within **Zones-Rows-MainPositions** cell

4. Summary of constructing nonorthogonal designs

- More difficult to identify the systematic design for a nonorthogonal design.
  - Not just a matter of using a standard, well-known design.
- Still use **`designRandomize`** to ensure a valid randomization and **`designAnatomy`** to check the properties of any design.
- For this, it remains necessary to:
  - Divide factors based on allocation of factors (as well as fixed/random).
  - Identify the crossing and nesting, which depends not only on the innate relationships, but also the model employed to account for anticipated variation.
- Numeric covariates introduce partial aliasing (nonorthogonality between allocated terms).

Degrees of balance

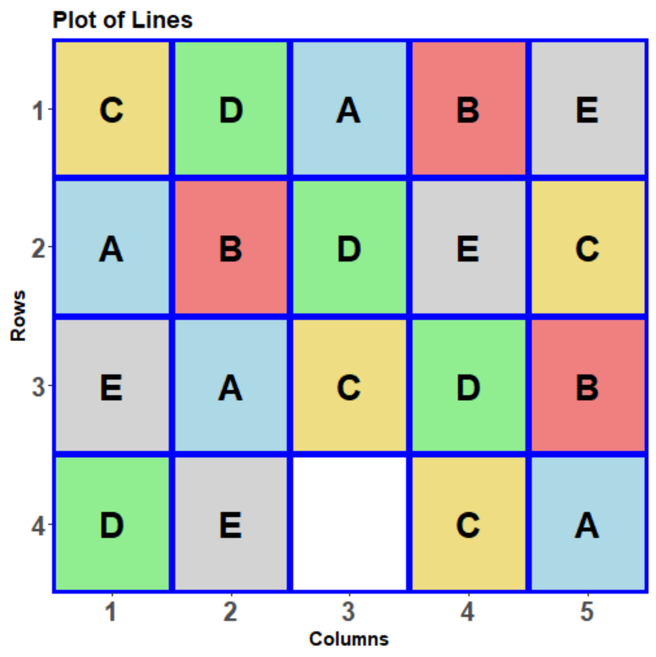
- Three degrees of balance have been encountered in the designs presented:
  1. **Orthogonal, and so balanced**: all canonical efficiency factors (nonzero eigenvalues) are one;
  2. **Balanced, but nonorthogonal**: some canonical efficiency factors are not one, however, they take just one value for (i) any recipient source or (ii) any allocated source when confounded with a particular recipient source.
  3. **Unbalanced and so must be nonorthogonal**: the canonical efficiency factors for at least one source of type (i) or (ii) above take more than one value.
- As we go down this list:
  - the degree of balance decreases and the complexity of the analysis increases;
- One of the great advantages of balanced designs is that all the standard errors of estimates of contrasts for a source of type (ii) will be equal.
  - All contrasts are treated equally.
  - Easier to present the results.
  - However, not always achievable.

Identifying an optimal design

- Several methods available for selecting an optimal design:
  - Deploy a standard design, like a randomized complete-block or split-unit design, known to be optimal
    – **`designRandomize`** can be used to obtain layouts for these.
  - Manually constructing a design, including the use of design keys for factorial experiments (Patterson & Bailey, 1978), given enough knowledge of combinatorics.
  - Consult a catalogue of designs (e.g. Cochran and Cox, 1957; Hinkelmann & Kempthorne, 2005; **`agricolae`** , de Mendiburu, 2019).
  - Computer generation of designs:
    - **`CycDesigN`**, **`od`**, **`SAS`**, **`JMP`**.

5. What happens when there is missing data?

- Suppose the 18th plot in the YSD is lost.
- How does this affect the design's properties?



```
> #'## Set up a layout with a single missing value
> YSD.miss1.lay <- YSD.lay
> YSD.miss1.lay$Lines[18] <- NA
> #'## Get the anatomy of the layout
> YSD.miss1.canon <- designAnatomy(formulae = list(plots = ~ Rows*Columns,
+                                                  lines = ~ Lines),
+                                  data = na.omit(YSD.miss1.lay))
```

Need an **NA** for the plot, but cannot have an **NA** for **designAnatomy**.

The anatomy for a missing value

```
> summary(YSD.miss1.canon, which.criteria = c("aeff", "xeff", "eeff", "order"))
Summary table of the decomposition for plots & lines (based on adjusted quantities)

 Source.plots df1 Source.lines df2 aefficiency xefficiency eefficiency order
 Rows           3 Lines          1      0.0500      0.0500      0.0500     1
                  Residual       2                                          
 Columns        4 Lines          4      0.0444      0.1968      0.0189     3
 Rows#Columns  11 Lines          4      0.8948      0.9663      0.7681     3
                  Residual       7                                          

Table of (partial) aliasing between sources derived from the same formula
 Source  df Alias In    aefficiency xefficiency eefficiency order
 Columns 4  Rows  plots      0.9870      1.0000      0.9500     2

The design is not orthogonal
```

`Columns` is not orthogonal to **`Rows`**;
it is partially aliased with **`Rows`**;
`Columns` is orthogonalized to **`Rows`**,
losing 1.3% in the process.

The anatomy for a missing value

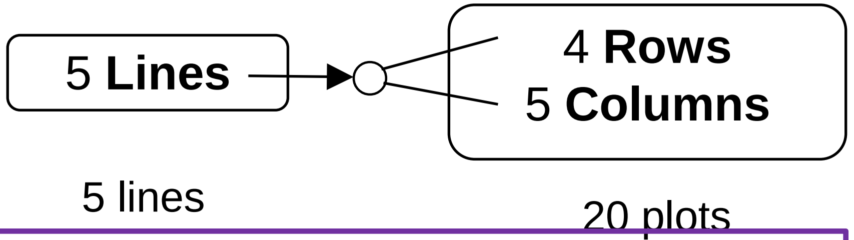
```
> summary(YSD.miss1.canon, which.criteria = c("aeff", "xeff", "eeff", "order"))
Summary table of the decomposition for plots & lines (based on adjusted quantities)

 Source.plots df1 Source.lines df2 aefficiency xefficiency eefficiency order
 Rows           3 Lines          1      0.0500      0.0500      0.0500     1
                  Residual       2
 Columns        4 Lines          4      0.0444      0.1968      0.0189     3
 Rows#Columns  11 Lines          4      0.8948      0.9663      0.7681     3
                  Residual       7
The design is not orthogonal
```

`Lines` confounding is no longer balanced.

Nor is it orthogonal to **`Rows`**.

Less **`Lines`** information confounded with **`Rows#Columns`** (cf 0.9375 for the YSD).

Confounding versus aliasing



```
Summary table of the decomposition for plots & lines (based on adjusted quantities)
 Source.plots df1 Source.lines df2 aefficiency xefficiency eefficiency order
 Rows           3 Lines          1      0.0500      0.0500      0.0500     1
                  Residual       2
 Columns        4 Lines          4      0.0444      0.1968      0.0189     3
 Rows#Columns  11 Lines          4      0.8948      0.9663      0.7681     3
                  Residual       7
```

```
Table of (partial) aliasing between sources derived from the same formula
 Source  df Alias In    aefficiency xefficiency eefficiency order
 Columns 4  Rows  plots      0.9870      1.0000      0.9500     2
```

- Aliasing refers to nonorthogonality between sources in the same tier (panel):
  - i.e. both allocated or both recipient sources; e.g. **Rows** and **Columns**.
- **Confounding** refers to nonorthogonality between sources from different tiers (panels):
  - i.e. an allocated and a recipient source; e.g. **Lines** and **Rows**.

Confounding versus aliasing

- Confounding and aliasing are about the relationships between sources.
- An allocated (recipient) source can be aliased or partially aliased with another allocated (recipient) source.
  - An aliased source is one that when the term for it is fitted, there is no information about its source available.
  - A partially aliased source only loses some of its information to sources for previously fitted terms, e.g. **`Columns`** is partially aliased with **`Rows`**.
  - Aliasing is to be avoided if possible, although sometimes it is purposefully employed (e.g. alias potentially small three-factor treatment interactions with treatment main effects in fractional factorial experiments).
- An allocated source can be confounded or partially confounded with a recipient source.
  - It is confounded with a recipient source when all information about it is associated with that recipient source.
  - If only part of the information is associated with the recipient source, then it is partially confounded with the recipient source.
  - Confounding or partial confounding is unavoidable in experiments.
  - Confounding is preferred to partial confounding, if it is achievable (and provided there is a Residual for the recipient source).

Confounding examples

In an RCBD, **`Lines`** is confounded with **`Columns[Rows]`**, i.e. all information about **`Lines`** is associated with the recipient source **`Columns[Rows]`**.

- Confounding

```
> summary(RCBD.canon)
 Source.plots df1 Source.lines df2 aefficiency eefficiency order
 Rows           3
 Columns[Rows] 20 Lines          4      1.0000      1.0000     1
                  Residual      16
```

- Partial Confounding

```
 Source.plots df1 Source.lines df2 aefficiency xefficiency eefficiency order
 Rows           3 Lines          1      0.0500      0.0500      0.0500     1
                  Residual       2
 Columns        4 Lines          4      0.0444      0.1968      0.0189     3
 Rows#Columns  11 Lines          4      0.8948      0.9663      0.7681     3
                  Residual       7
```

In a YSD with a missing value, **`Lines`** is partially confounded with **`Rows`**, **`Columns`** and **`Columns[Rows]`**, i.e. some information about **`Lines`** is associated with all recipient sources.

A missing treatment

- How does a missing treatment affect the properties of the design?

```
> #'## Set up a layout with a missing Line
> YSD.missA.lay <- YSD.lay
> YSD.missA.lay$Lines[YSD.missA.lay$Lines == "A"] <- NA
```

The anatomy for a missing treatment

```
> #'## Get the anatomy of the layout
> YSD.missA.canon <- designAnatomy(formulae = list(plots = ~ Rows*Columns,
+                                                  lines = ~ Lines),
+                                   data = na.omit(YSD.missA.lay))
> summary(YSD.missA.canon, which.criteria = c("aeff", "xeff", "eeff", "order"))

Summary table of the decomposition for plots & lines (based on adjusted quantities)
 Source.plots df1 Source.lines df2 aefficiency xefficiency eefficiency order
 Rows           3
 Columns        4 Lines          3      0.0909      0.0909      0.0909     1
                  Residual       1
 Rows#Columns   8 Lines          3      0.9091      0.9091      0.9091     1
                  Residual       5

Table of (partial) aliasing between sources derived from the same formula
 Source  df Alias In    aefficiency xefficiency eefficiency order
 Columns 4  Rows  plots      0.9362      1.0000      0.9167     2

The design is not orthogonal
```

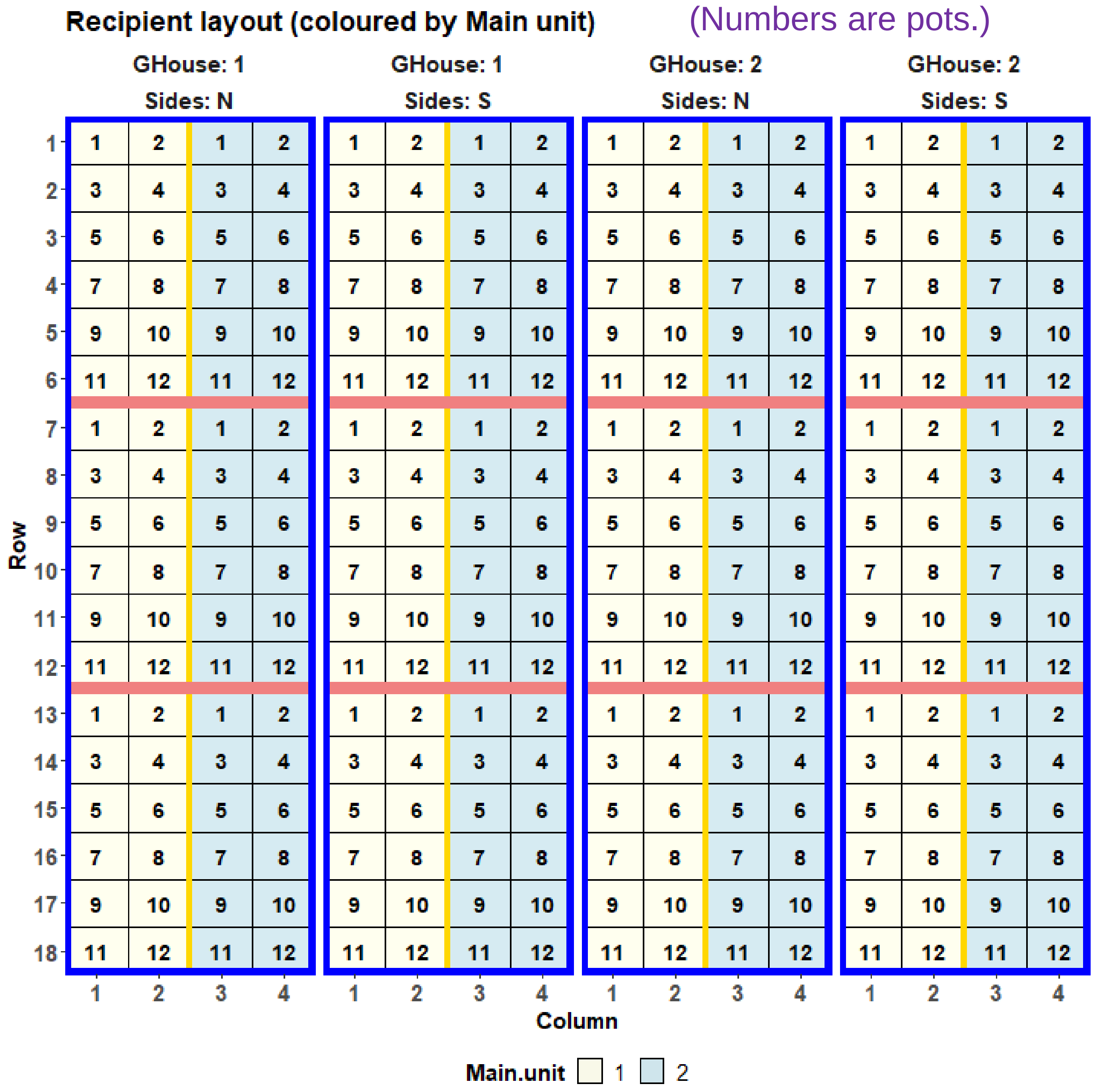
What has been the effect of the missing treatment?

6. Systematic allocation and pseudoreplication

- It happens that randomization is not always desirable or possible.
- A grapevine experiment is to be run in two greenhouses:
  - One greenhouse is to be kept at ambient temperature and the other is to be cooled;
  - Of the two greenhouses, one is naturally warmer than the other and so needs to be the warm greenhouse.
  - So randomization is not desirable.
- Within each greenhouse, two salinity treatments (control and saline) are to be applied to 12 varieties.
- The combinations of Heat, Salinity and Varieties are to be replicated 6 times.

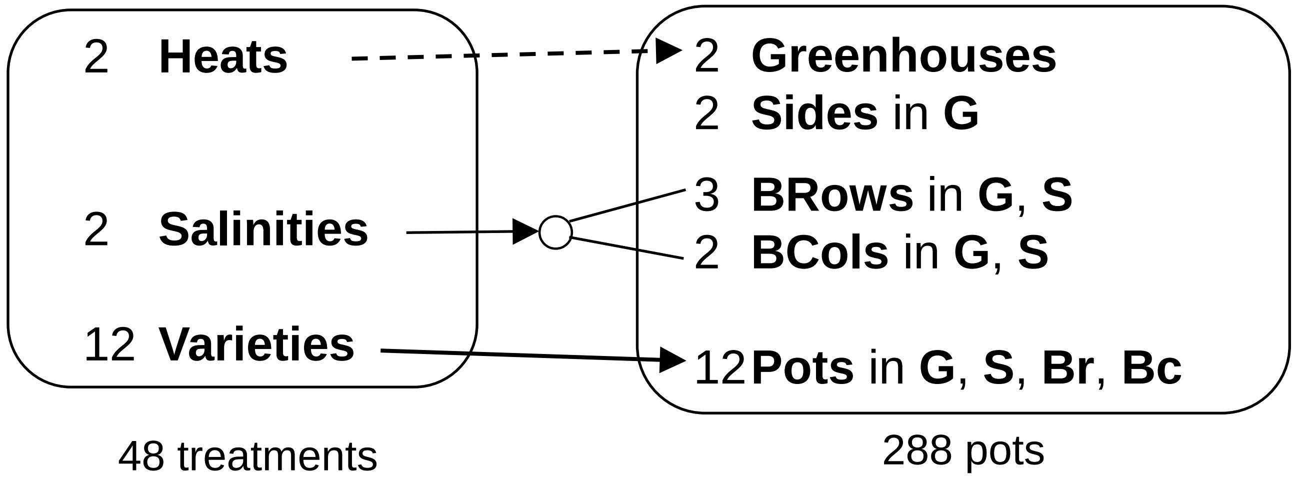
Grapevine design

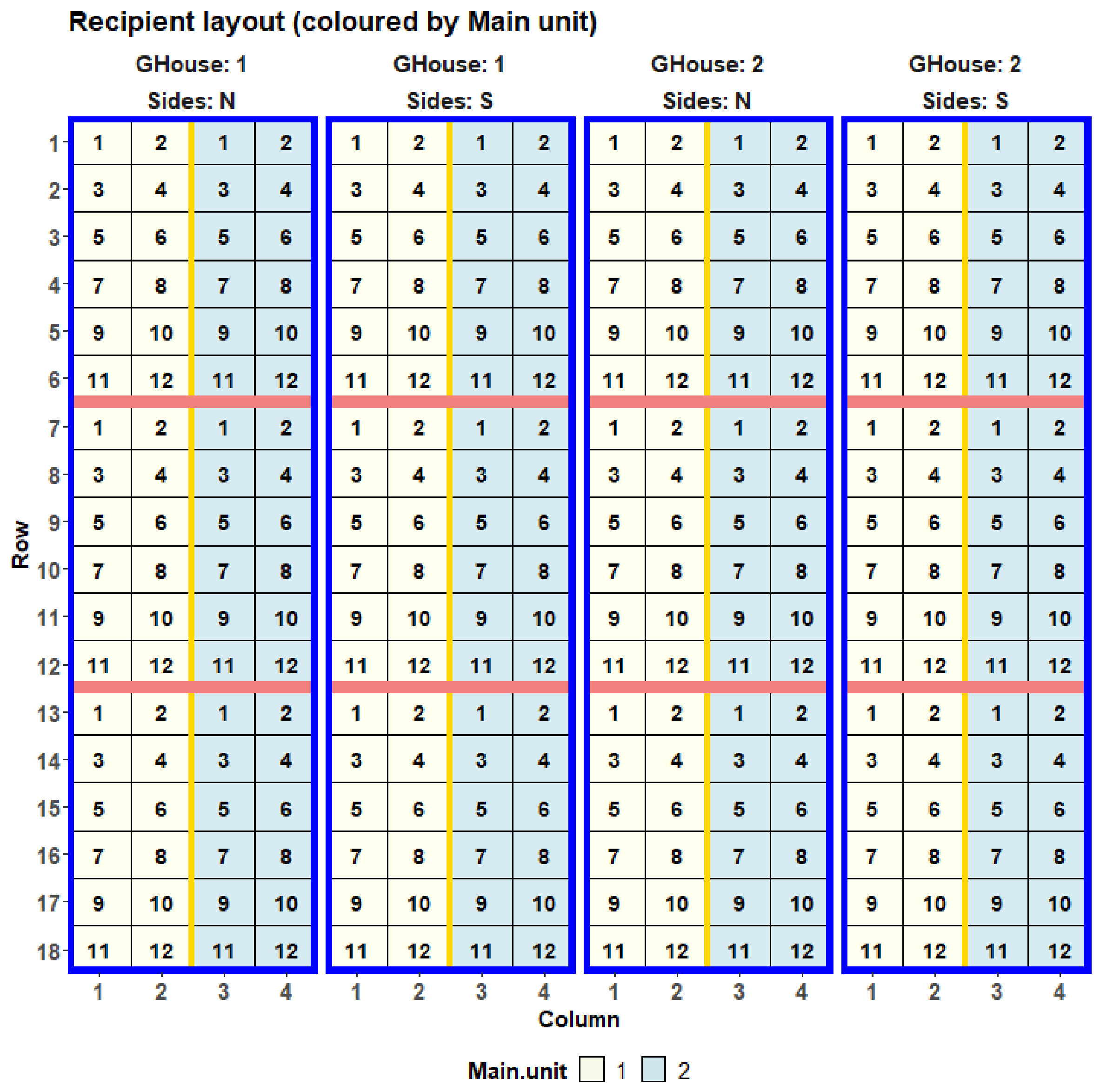
- Within each greenhouse:
  - There are 2 Sides (blue rectangles), with 6 main units per Side (pink and yellow lines separates main units).
  - A split-unit design is to be used to assign
    - Salinities to main units;
    - Varieties to 12 pots (subunits) in each main unit.
  - Split-unit design because:
    - Large differences between Salinities;
    - Variety differences are the most important.



Grapevine design

- Anticipated model:
  - Heat * Salinity * Varieties | Ghouses + Ghouses:Sides + Ghouses:Sides:BRows + Ghouses:Sides:BCols +
    Main units Ghouses:Sides:Brows:BCols +
    Subunits Ghouses:Sides:Brows:Bcols:Pots.



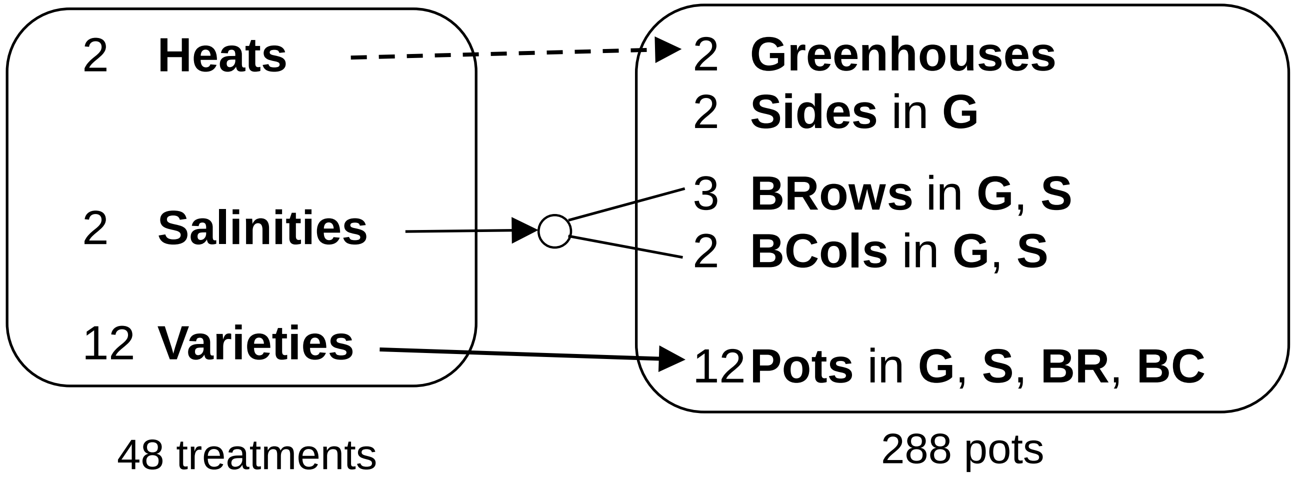


Row	GHouse: 1 Sides: N 1	2	3	4	GHouse: 1 Sides: S 1	2	3	4	GHouse: 2 Sides: N 1	2	3	4	GHouse: 2 Sides: S 1	2	3	4
1	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2
2	3	4	3	4	3	4	3	4	3	4	3	4	3	4	3	4
3	5	6	5	6	5	6	5	6	5	6	5	6	5	6	5	6
4	7	8	7	8	7	8	7	8	7	8	7	8	7	8	7	8
5	9	10	9	10	9	10	9	10	9	10	9	10	9	10	9	10
6	11	12	11	12	11	12	11	12	11	12	11	12	11	12	11	12
7	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2
8	3	4	3	4	3	4	3	4	3	4	3	4	3	4	3	4
9	5	6	5	6	5	6	5	6	5	6	5	6	5	6	5	6
10	7	8	7	8	7	8	7	8	7	8	7	8	7	8	7	8
11	9	10	9	10	9	10	9	10	9	10	9	10	9	10	9	10
12	11	12	11	12	11	12	11	12	11	12	11	12	11	12	11	12
13	1	2	1	2	1	2	1	2	1	2	1	2	1	2	1	2
14	3	4	3	4	3	4	3	4	3	4	3	4	3	4	3	4
15	5	6	5	6	5	6	5	6	5	6	5	6	5	6	5	6
16	7	8	7	8	7	8	7	8	7	8	7	8	7	8	7	8
17	9	10	9	10	9	10	9	10	9	10	9	10	9	10	9	10
18	11	12	11	12	11	12	11	12	11	12	11	12	11	12	11	12

Main.unit ◻ 1 ▨ 2

To balance Salinity, a 3 $\times$ 2 extended Latin square design, based on 2 $\times$ 2 Latin squares is to be used in each Side.

Generating the systematic design in R



Generate Heat and Varieties in standard order; the 12 works as if a factor with 12 levels occurs in this position.

Generate the recipient factors indexing the pots.

Extra rows.

```
> split.sys <- cbind(fac.gen(list(GHouse = 2, Sides = c("N", "S"),
+                                  BRows = 3, BCols = 2, Pots = 12)),
+                    fac.gen(list(Heat = c("Warm", "Cool"), 12, Varieties = 12)),
+                    Salinity = factor(rep(c(designLatinSqrSys(2),1,2,
+                                            designLatinSqrSys(2, start = c(2,1)),2,1),
+                                          each = 12, times = 2),
+                                      labels = c("Control", "Na")))
```

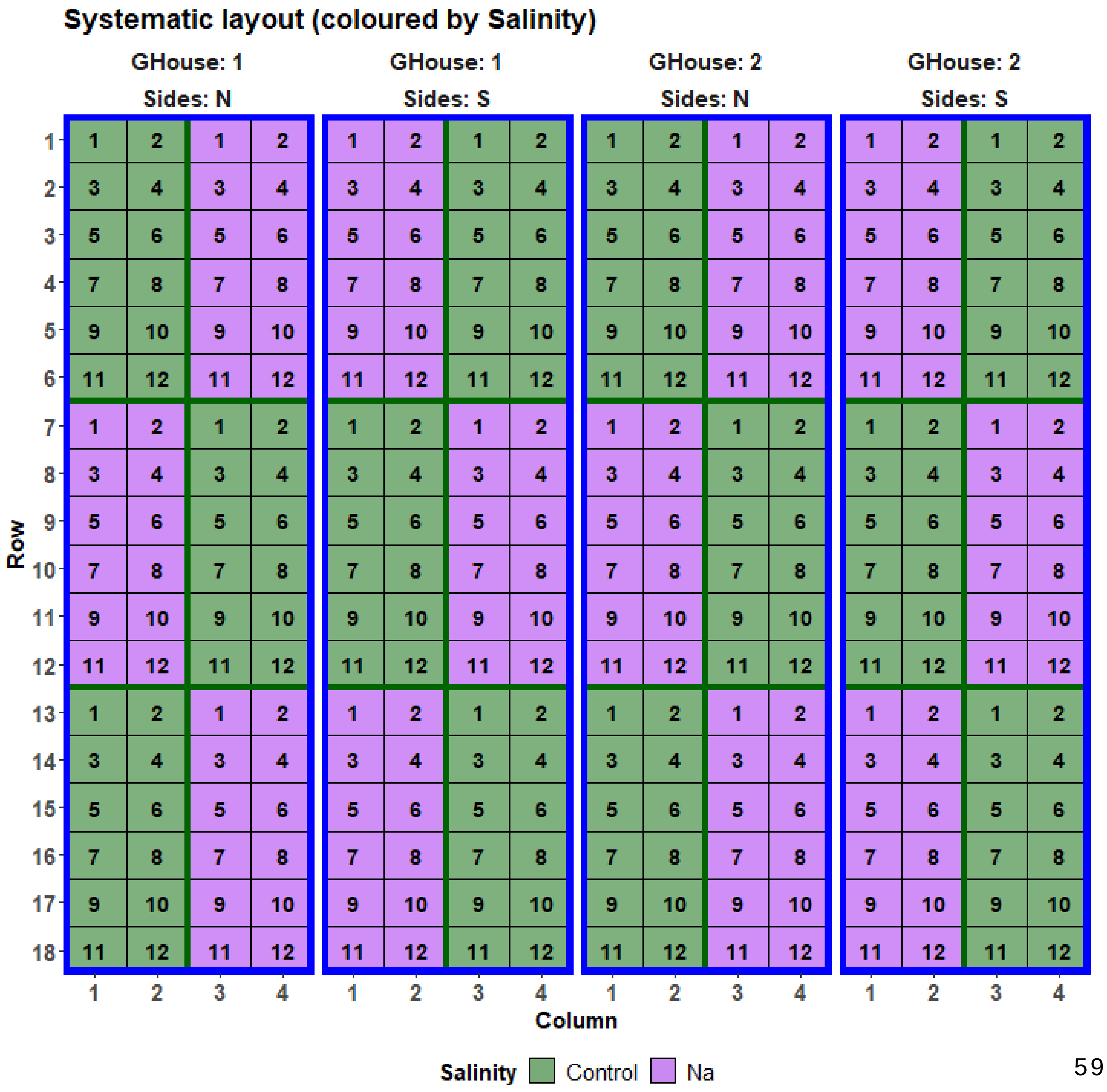
Salinity has to be assigned using Extended Latin Squares (ELS).

Two Latin squares with different starting rows.

Pair of ELS designs repeated twice, one for each GHouse.

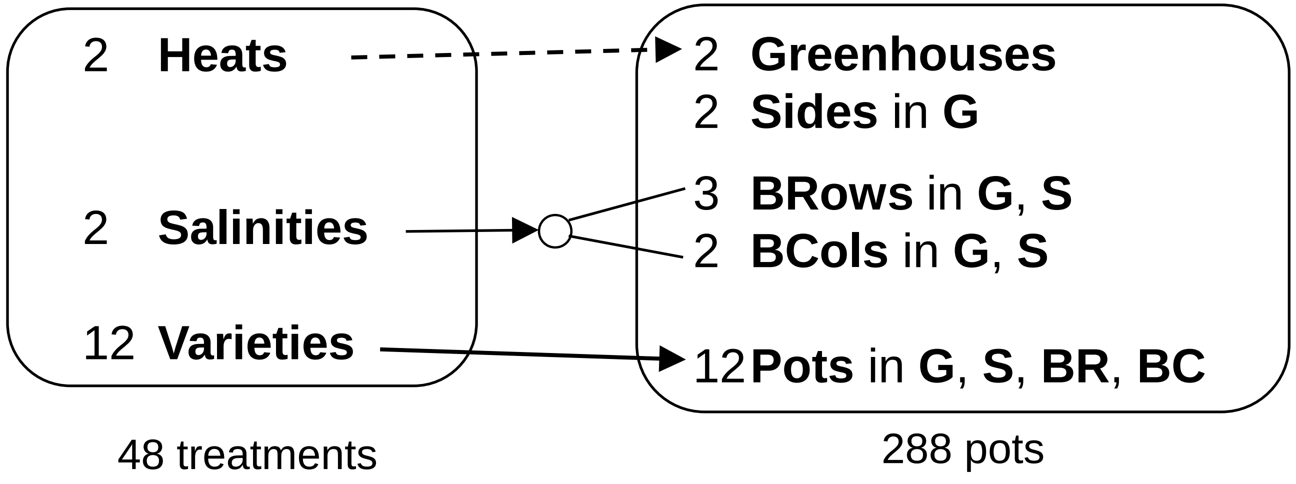
Systematic grapevine design

- Numbers are **`Varieties`**.



Randomizing the grapevine design

- Use **`designRandomize`** from **`dae`** to randomize the systematic layout.
  - The randomization is determined by the nesting relationships between the **`recipient`** factors.



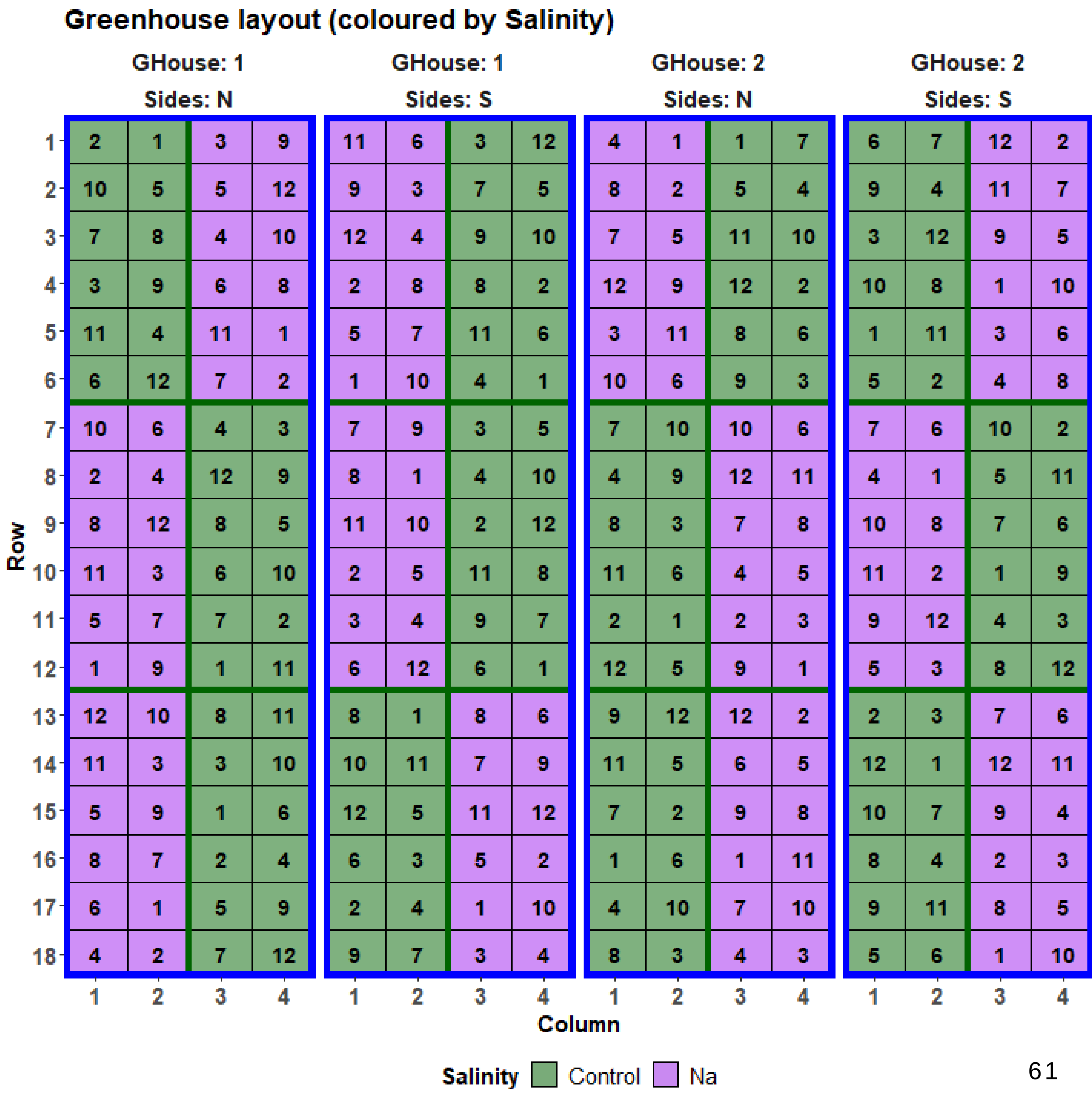
```
> split.lay <- designRandomize(allocated = split.sys[c("Heat", "Salinity", "Varieties")],
+                              recipient = split.sys[c("GHouse", "Sides",
+                                                      "BRows", "BCols", "Pots")],
+                              nested.recipients = list(Sides = "GHouse",
+                                                       BRows = c("Sides", "GHouse"),
+                                                       BCols = c("Sides", "GHouse"),
+                                                       Pots = c("Sides", "GHouse",
+                                                                "BRows", "BCols")),
+                              except = "GHouse",
+                              seed = 64131)
```

The **`except`** option allows the generation of a design, in which Heat is systematically allocated, while Salinity and Lines are randomized.

The **`nested.recipients`** specifies the nesting shown in the **`pots`** panel; factors not nested are assumed to be crossed.

Grapevine design

- Warm has been systematically assigned to the first Greenhouse.
- Within a Side there are 2 columns (BCols) of main units:
  - One has 2 main units with Control and the other 2 with Na.
- A complete set of the 12 Varieties is randomized within each main unit.



Row	GHouse: 1 Sides: N				GHouse: 1 Sides: S				GHouse: 2 Sides: N				GHouse: 2 Sides: S			
	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
1	2	1	3	9	11	6	3	12	4	1	1	7	6	7	12	2
2	10	5	5	12	9	3	7	5	8	2	5	4	9	4	11	7
3	7	8	4	10	12	4	9	10	7	5	11	10	3	12	9	5
4	3	9	6	8	2	8	8	2	12	9	12	2	10	8	1	10
5	11	4	11	1	5	7	11	6	3	11	8	6	1	11	3	6
6	6	12	7	2	1	10	4	1	10	6	9	3	5	2	4	8
7	10	6	4	3	7	9	3	5	7	10	10	6	7	6	10	2
8	2	4	12	9	8	1	4	10	4	9	12	11	4	1	5	11
9	8	12	8	5	11	10	2	12	8	3	7	8	10	8	7	6
10	11	3	6	10	2	5	11	8	11	6	4	5	11	2	1	9
11	5	7	7	2	3	4	9	7	2	1	2	3	9	12	4	3
12	1	9	1	11	6	12	6	1	12	5	9	1	5	3	8	12
13	12	10	8	11	8	1	8	6	9	12	12	2	2	3	7	6
14	11	3	3	10	10	11	7	9	11	5	6	5	12	1	12	11
15	5	9	1	6	12	5	11	12	7	2	9	8	10	7	9	4
16	8	7	2	4	6	3	5	2	1	6	1	11	8	4	2	3
17	6	1	5	9	2	4	1	10	4	10	7	10	9	11	8	5
18	4	2	7	12	9	7	3	4	8	3	4	3	5	6	1	10

Properties of the grapevine design

Matches the **`nested.recipients`**

```
> split.canon <- designAnatomy(formulae = list(tests = ~ GHouse/Sides/(BRows*BCols)/Pots,
+                                              cond = ~ Heat*Salinity*Varieties),
+                               data = split.lay)
> summary(split.canon, which.criteria=c("aeff", "order"))
Summary table of the decomposition for tests & cond (based on adjusted quantities)

 Source.tests                  df1 Source.cond              df2 aefficiency order
 GHouse                          1 Heat                       1      1.0000     1
 Sides[GHouse]                   2                                                
 BRows[GHouse:Sides]             8                                                
 BCols[GHouse:Sides]             4 Salinity                   1      0.1111     1
                                   Heat#Salinity              1      0.1111     1
                                   Residual                   2                   
 BRows#BCols[GHouse:Sides]       8 Salinity                   1      0.8889     1
                                   Heat#Salinity              1      0.8889     1
                                   Residual                   6                   
 Pots[GHouse:Sides:BRows:BCols] 264 Varieties                11      1.0000     1
                                   Heat#Varieties            11      1.0000     1
                                   Salinity#Varieties        11      1.0000     1
                                   Heat#Salinity#Varieties   11      1.0000     1
                                   Residual                 220                   

The design is not orthogonal
```

`Heat` is confounded with **`GHouse`**, an effect of the pseudoreplication.

`Salinity` (& **`Heat#Salinity`**) are not orthogonal, but most information is confounded with **`Brows#Bcols[GH:S]`**. (Design property)

All **`Varieties`** effects are orthogonal.

But all orders are one and so it is balanced.

Prior allocation model for the grapevine design

```
Summary table of the decomposition for tests & cond (based on adjusted quantities)

 Source.tests                  df1 Source.cond                df2 aefficiency order
 GHouse                          1 Heat                         1      1.0000     1
 Sides[Ghouse]                   2
 BRows[GHouse:Sides]             8
 BCols[GHouse:Sides]             4 Salinity                     1      0.1111     1
                                   Heat#Salinity                1      0.1111     1
                                   Residual                     2
 BRows#BCols[GHouse:Sides]       8 Salinity                     1      0.8889     1
                                   Heat#Salinity                1      0.8889     1
                                   Residual                     6
 Pots[GHouse:Sides:BRows:BCols] 264 Varieties                  11      1.0000     1
                                   Heat#Varieties              11      1.0000     1
                                   Salinity#Varieties          11      1.0000     1
                                   Heat#Salinity#Varieties     11      1.0000     1
                                   Residual                   220

The design is not orthogonal
```

The confounding of **Heat** and **GHouse**, is exhaustive in that all **GHouse** information is mixed up with **Heat** differences (there is no Residual **GHouse** Residual).

That is **Heat** and **GHouse** are inextricably mixed up together so that one cannot say which part of any difference associated with either factor is due one or other of the factors.

It means that the initial allocation model will not fit.

- To have a prior allocation model that will fit, one of GHouse and Heat must be removed.

Prior allocation model for the grapevine design

- GHouse is the obvious choice so that Heat and its interactions are retained.
  - Heat * Salinity * Varieties | ~~Ghouses +~~ Ghouses:Sides + Ghouses:Sides:BRows + Ghouses:Sides:BCols + Ghouses:Sides:Brows:BCols + Ghouses:Sides:Brows:Bcols:Pots.

```
Summary table of the decomposition for tests & cond (based on adjusted quantities)

 Source.tests                    df1 Source.cond             df2 aefficiency order
 GHouse:Sides                      3 Heat                      1      1.0000     1
                                     Residual                  2
 BRows[GHouse:Sides]               8
 BCols[GHouse:Sides]               4 Salinity                  1      0.1111     1
                                     Heat#Salinity             1      0.1111     1
                                     Residual                  2
 BRows#BCols[GHouse:Sides]         8 Salinity                  1      0.8889     1
                                     Heat#Salinity             1      0.8889     1
                                     Residual                  6
 Pots[GHouse:Sides:BRows:BCols] 264 Varieties                11      1.0000     1
                                     Heat#Varieties           11      1.0000     1
                                     Salinity#Varieties       11      1.0000     1
                                     Heat#Salinity#Varieties  11      1.0000     1
                                     Residual                220
```

This model is a ”model of convenience”: it gives a fit. However, it does not contain all the pertinent sources of variation in the experiment.

This revised anatomy shows that Sides variability will be used for judging overall Heat differences; this is very likely to be an underestimate of the variability affecting Heat differences.

7. Summary of confounding and aliasing

- In comparative experiments,
  - there is always some confounding; and
  - there may be some aliasing.
- All allocation, be it systematic, haphazard, spatial or randomized, results in confounding:
  - **`designAnatomy`** does not distinguish between different types of allocation.
  - Properly replicated treatments can be systematically allocated.
    - The danger with systematic replication is that it will be confounded with any systematic trends associated with the factors to which it is allocated.
- Pseudoreplication manifests as exhaustive confounding.
- Numeric covariates introduce partial aliasing (nonorthogonality between allocated terms).
- Missing values introduce partial aliasing and confounding.

Practical session for *Nonorthogonal experimental design in R*

1. Using **dae** and **od** to obtain randomized layouts for orthogonal designs.
   i. An alpha design
   ii. A BIBD.
   iii. A nonorthogonal row-column design for a Casuarina trial.
   iv. A 25-line wheat experiment from Gilmour et al. (1995).
   v. A small environmental experiment.
2. Again, you have only to follow the script that has been given.
3. There are some questions for you to answer about each design.

References

- Brien, C. J. (2017). Multiphase experiments in practice: A look back. *Australian & New Zealand Journal of Statistics,* **59**, 327-352.
- Brien, C. J. (2019). *dae: functions useful in the design and ANOVA of experiments*. R package version 3.1-16. URL http://cran.at.r-project.org/package=dae.
- Butler, D. G. (2019) *od: generate optimal experimental designs*. (R package version 2.0.0, to be made available) https://mmade.org/.
- Butler, D. G., Cullis, B. R., Gilmour, A. R., Gogel, B. J., & Thompson, R. (2018). *ASReml-R reference manual (Version 4)*. Hemel Hempstead: VSN International Ltd.
- Cochran, W. G., & Cox, G. M. (1957). *Experimental Designs* (2nd ed.). New York: Wiley.
- De Mendiburu, F. (2018). *agricolae: statistical procedures for agricultural research*. R package version 1.3-1. URL http://cran.at.r-project.org/package=agricolae.
- Gilmour, A. R., Thompson, R., & Cullis, B. R. (1995). Average Information REML: An Efficient Algorithm for Variance Parameter Estimation in Linear Mixed Models. *Biometrics,* **51**, 1440-1450.
- Kiefer, J. (1959). Optimum Experimental Designs. *Journal of the Royal Statistical Society, Series B (Methodological),* **21**, 272-319.
- Hinkelmann, K., & Kempthorne, O. (2005). *Design and analysis of experiments Vol. 2 Advanced experimental design*. Hoboken, N.J.: Wiley-Interscience.
- Patterson, H. D., & Bailey, R. A. (1978). Design keys for factorial experiments. *Journal of the Royal Statistical Society, Series C (Applied Statistics),* **27**, 335-343.