Designing comparative experiments using R

(Chris Brien and Sam Rogers)

III. Multiphase and partially replicated (*p*-rep) designs

Outline

1. Multiphase designs.
2. Partially replicated designs.
3. Summary.

1. Multiphase designs

- Brien (2017) gives a review, including published applications.
- Three introductory papers are Brien and Bailey (2006), Brien et al. (2011) and Brien (2019).
- “Normal” two-phase experiments (Brien et al., 2011,Section 4) involve a single-randomization in each phase.
  - This implies that a design is required for each phase.
  - The object of the second phase is to evaluate material produced in the first phase and one or more response variables are measured in the second phase.
  - There may also be response variables from the first-phase.
  - The phase is the period of time during which a set of units are engaged in producing their outcome: material and/or response variables.
  - One phase might overlap another phase.

1.1 A simple two-phase athlete training experiment

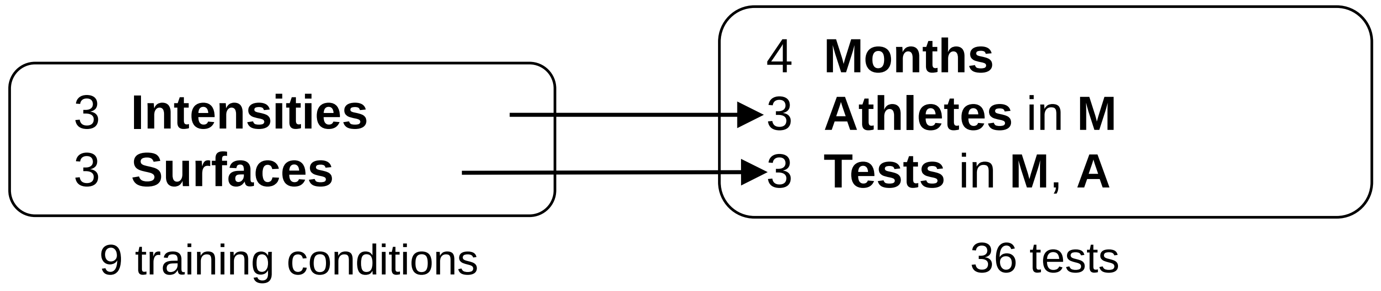
Brien, Harch, Correll, Bailey (2011)

- Suppose in a simple two-phase athlete training experiment:
  - in addition to heart rate taken immediately upon completion of a test,
  - the free haemoglobin is to be measured using blood specimens taken from the athletes after each test, which are to be transported to the lab for analysis.
- The experiment consists of a test phase and a laboratory phase:
  - Test phase: 36 tests involving 3 athletes in each of 4 months; heart rate is measured and a blood specimen taken.
    - The unit is a test taken by an athlete.
    - The outcomes are the heart rate, a response variable, and a blood sample, material for the second phase.
  - Laboratory phase: each month 3 blood samples are taken to the laboratory for analysis.
    - The unit is a blood specimen.
    - The outcome is the free haemoglobin in the blood specimen, a response variable.

First phase: athlete testing

Brien, Harch, Correll, Bailey (2011)

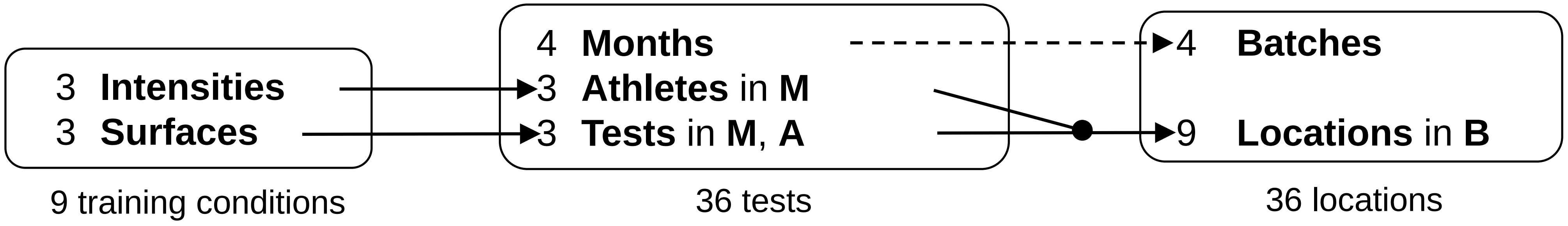
- Recall from the standard athlete training experiment in Session 1 that 9 training conditions are to be investigated:
  - combinations of 3 surfaces and 3 intensities of training.
- In each of the 4 Months of testing:
  - 3 endurance athletes are recruited.
  - Each athlete undergoes 3 tests, separated by 7 days, under 3 different training conditions.
- A split-unit design was employed to allocate Intensities and Surfaces.



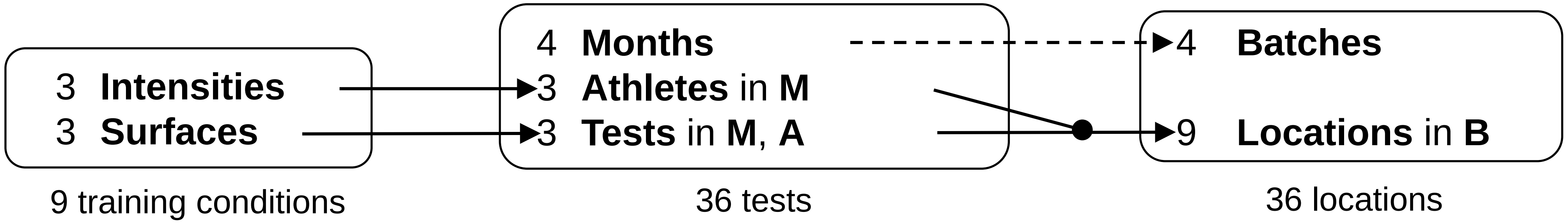
- For the second phase the 36 tests need to be allocated for analysis in the laboratory.

Second laboratory phase

- A restriction in the second phase:
  - The blood specimens from the first phase need to be processed as soon as possible (not held for 4 months).
  - Thus, the 9 specimens collected each month are to be processed together.
- Suppose that it is decided to process them in a random order,
  - That is, it is assumed that there is no systematic trend across the processing of the 9 samples so that a nested second-phase design is required.

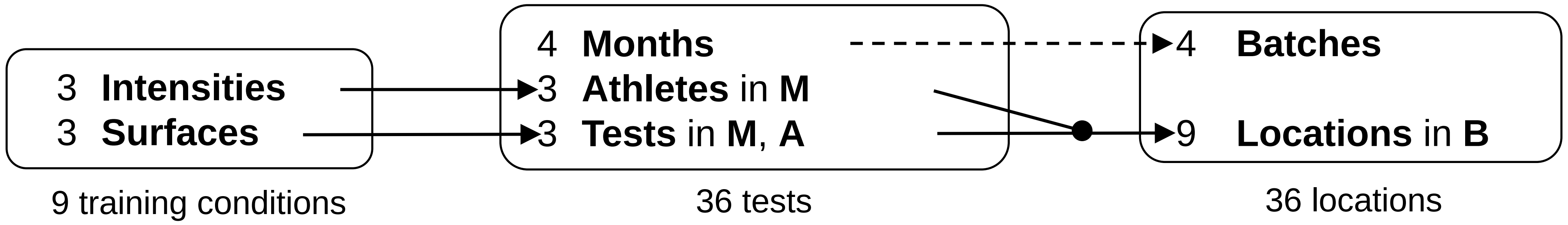


A simple two-phase athlete training experiment (cont'd)



- It is two-phase with three sets of objects:
  - training conditions, tests and locations:
    - training conditions are allocated in the first-phase and the second-phase i.e. only ever allocated.
    - tests are recipients factors in the second phase and are allocated factors in the second phase i.e. different roles in the two phases.
    - locations are recipient factors in the second phase i.e. only ever recipient.

A simple two-phase athlete training experiment (cont'd)

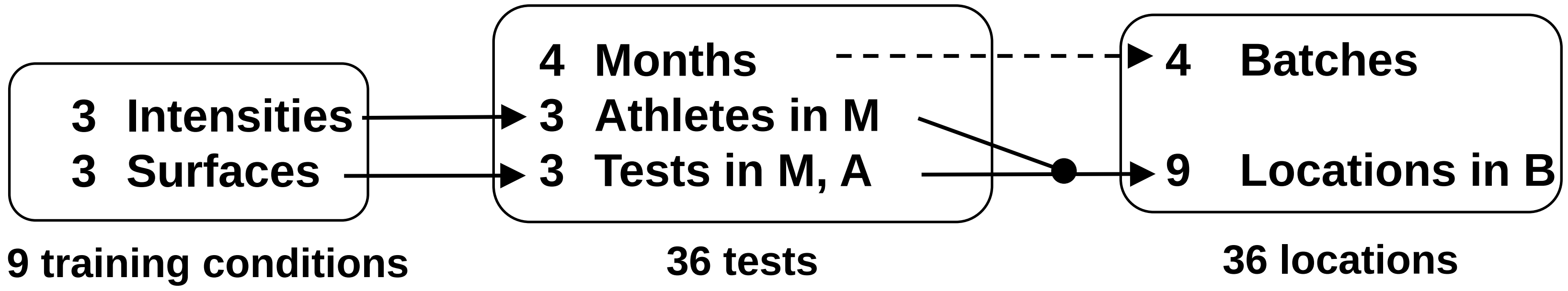


- It is described as involving two composed allocations, one of two types of allocations in a chain:
  - Training conditions are allocated to tests and tests to locations;
- Here, the second phase begins during the first phase.
- Have not allowed for an overall, processing-order effect.
  - More about that soon.

Randomization in the second phase

- **Principle 7** (Allocate all and randomize in laboratory) (Brien et al, 2011):
  - The laboratory-phase design should *always* allocate *all* the first-phase unit factors, as well as any laboratory treatments, to the laboratory units, using randomization wherever possible.
- As is the case for any randomization, randomizing the lab phase:
  - Guards against unanticipated systematic effects.
  - Justifies the form of the variance matrix used for the experiment.
  - Required for a valid estimate of error.
- Additionally, for a second (lab) phase, randomizing
  - compensates for unfortunate randomizations in the first phase.
- However, have seen that practical problems can limit randomization.
- But, are there other reasons for not randomizing the second phase?
  - Does it make it difficult to estimate first-phase phenomena?
  - For example, spatial correlation, linear trends, unequal variances?

The design species for a normal two-phase design



- The four design species for allocating sets of objects when there is an allocation in each of the two phases (Brien, 2019):
  - *First-phase design:* allocated and recipient objects from the first phase
    - (training conditions and tests);
  - *Second-phase design:* first- and second-phase recipient objects
    - (tests and locations);
  - *Cross-phase design:* first-phase allocated objects and second-phase recipient objects
    - (training conditions and locations);
  - *Two-phase or combined design:* all three sets of objects.
- **`designTwophaseAnatomies`** produces the four anatomies for them.

The anatomy for the first phase design (from Session 1)

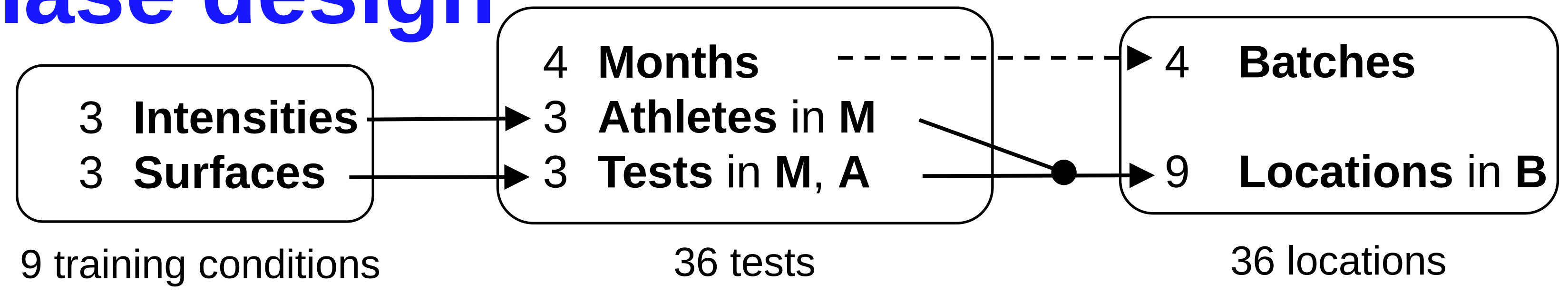
```
> split.canon <- designAnatomy(formulae = list(tests = ~Months/Athletes/Tests,
+                                              cond  = ~Intensities*Surfaces),
+                              data      = split.lay)
> summary(split.canon, which.criteria="none")
```

Formulae for recipient and allocated.

Layout is in **`split.lay`**.

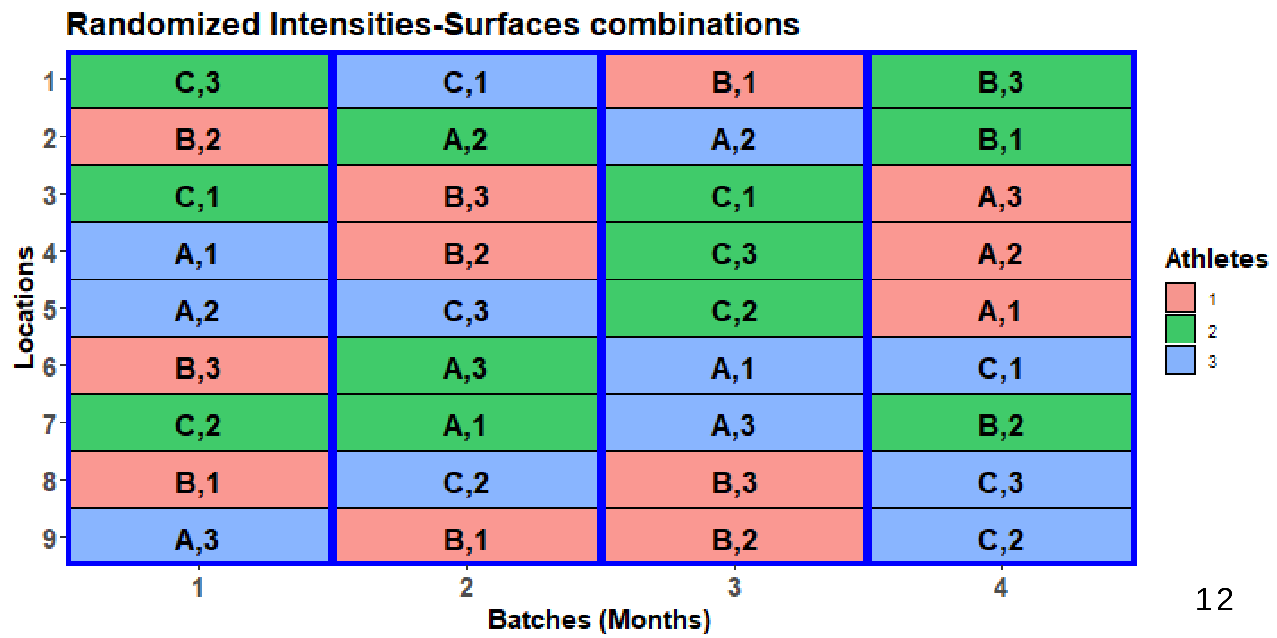
```
Summary table of the decomposition for tests & cond

 Source.tests               df1 Source.cond            df2
 Months                       3
 Athletes[Months]             8 Intensities              2
                                Residual                 6
 Tests[Months:Athletes]      24 Surfaces                 2
                                Intensities#Surfaces     4
                                Residual                18
```

Construct two-phase design



- Have to randomize tests (and training conditions) to locations

```
> eg1.lay <- designRandomize(allocated         = split.lay,
+                            recipient         = list(Batches = 4, Locations = 9),
+                            nested.recipients = list(Locations = "Batches"),
+                            except            = "Batches",
+                            seed = 71230)
```



Locations	1	2	3	4
1	🟩 C,3	🟦 C,1	🟥 B,1	🟩 B,3
2	🟥 B,2	🟩 A,2	🟦 A,2	🟩 B,1
3	🟩 C,1	🟥 B,3	🟩 C,1	🟥 A,3
4	🟦 A,1	🟥 B,2	🟩 C,3	🟥 A,2
5	🟦 A,2	🟦 C,3	🟩 C,2	🟥 A,1
6	🟥 B,3	🟩 A,3	🟦 A,1	🟦 C,1
7	🟩 C,2	🟩 A,1	🟦 A,3	🟩 B,2
8	🟥 B,1	🟦 C,2	🟥 B,3	🟦 C,3
9	🟦 A,3	🟥 B,1	🟥 B,2	🟦 C,2

Check properties of the multiphase design

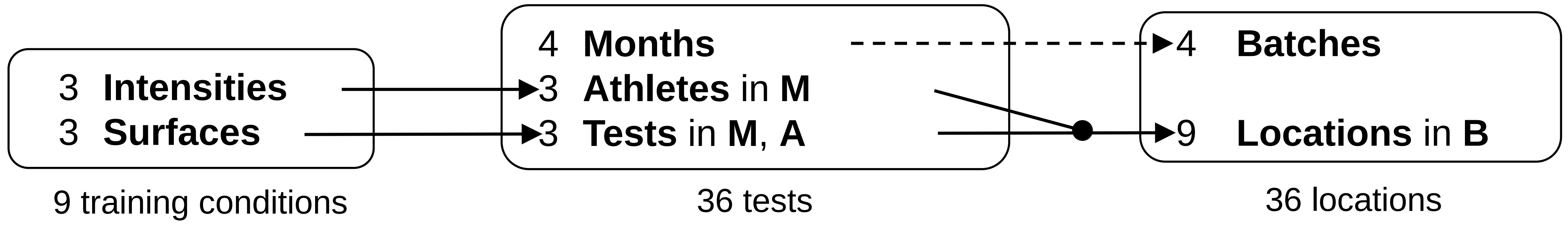
```
> eg1.canon <- designAnatomy(formulae = list(locs = ~ Batches/Locations,
+                                            test = ~ Months/Athletes/Tests,
+                                            cond = ~ Intensities*Surfaces),
+                            data     = eg1.lay)
> summary(eg1.canon, which.criteria=c("aeff", "order"))
```

Three formulae reflecting the factor-allocation diagram (no limit on the number).

```
Summary table of the decomposition for locs, test & cond

 Source.locs        df1 Source.test              df2 Source.cond          df3 aefficiency order
 Batches              3 Months                     3                                1.0000     1
 Locations[Batches]  32 Athletes[Months]           8 Intensities            2       1.0000     1
                                                     Residual               6       1.0000     1
                        Tests[Months:Athletes]    24 Surfaces               2       1.0000     1
                                                     Intensities#Surfaces   4       1.0000     1
                                                     Residual              18       1.0000     1
```

- All sources are orthogonal and all, except **`Months`**, are confounded with **`Locations[Batches]`**.
- Note also that there are no residuals for **`Batches`** or **`Locations[Batches]`**.
  - They are exhaustively confounded, which will always be the case when the numbers of objects are equal for two consecutive tiers.
- Question that remains: what mixed model to fit?

Initial allocation model



- Initial allocation model (like the anatomy, reflects the factor allocation diagram):
  - Intensities + Surfaces + Intensities:Surfaces |
    Months + Months:Athletes + Months:Athletes:Tests +
    Batches + Batches:Locations.
- However, this model will not fit because of confounding between tests and locations.

Prior allocation model

```
Summary table of the decomposition for locs, test & cond

 Source.locs          df1 Source.test              df2 Source.cond           df3 aefficiency order
 Batches                3 Months                     3                                1.0000     1
 Locations[Batches]    32 Athletes[Months]           8 Intensities             2      1.0000     1
                                                       Residual                6      1.0000     1
                          Tests[Months:Athletes]    24 Surfaces                2      1.0000     1
                                                       Intensities#Surfaces    4      1.0000     1
                                                       Residual               18      1.0000     1
```

- Need to remove
  - One of Months and Batches, and make the retained term fixed:
  - Locations:Batches or both Months:Athletes and Months:Athletes:Tests.
- Must retain Months:Athletes, otherwise it would be pooled, either into Months:Athletes:Tests or Batches:Locations.
- The prior allocation model is the model for the first phase and is a model of convenience:
  - Months + Intensities + Surfaces + Intensities:Surfaces | Months:Athletes + Months:Athletes:Tests.
- The very important point is that, while they are not in the model, the lab terms contribute to those that are e.g. Months is not just due to Months differences, but is also due to Batches variance.

1.2 Allowing for lab order in the athletic experiment

Brien (2017)

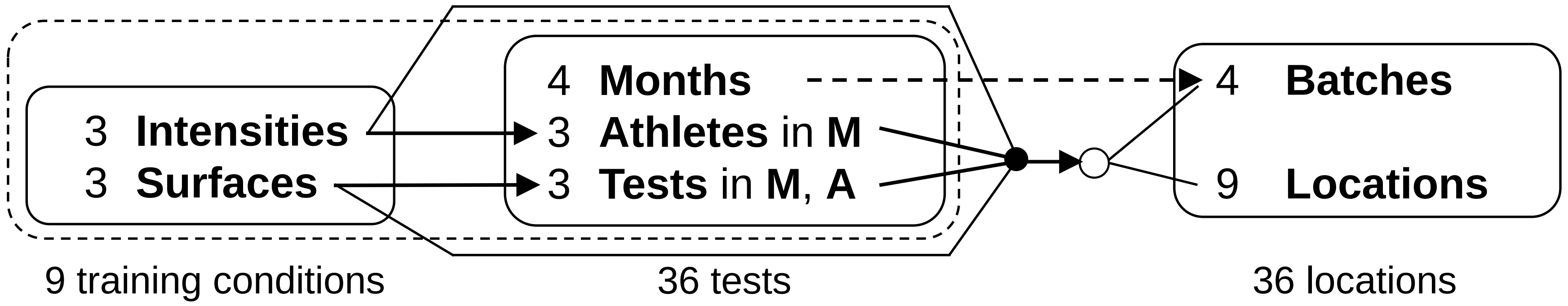
- If it is expected that there will be consistent differences between locations across the months, then the initial allocation model would be:
  - Intensities + Surfaces + Intensities:Surfaces | Months + Months:Athletes + Months:Athletes:Tests + Batches + Locations + Batches:Locations.
  - That is, Batches and Locations are now crossed (similar to RCBD versus LSD).
  - A row-column design is required for the crossed, second phase.

The Locations term has been added.

Design considerations

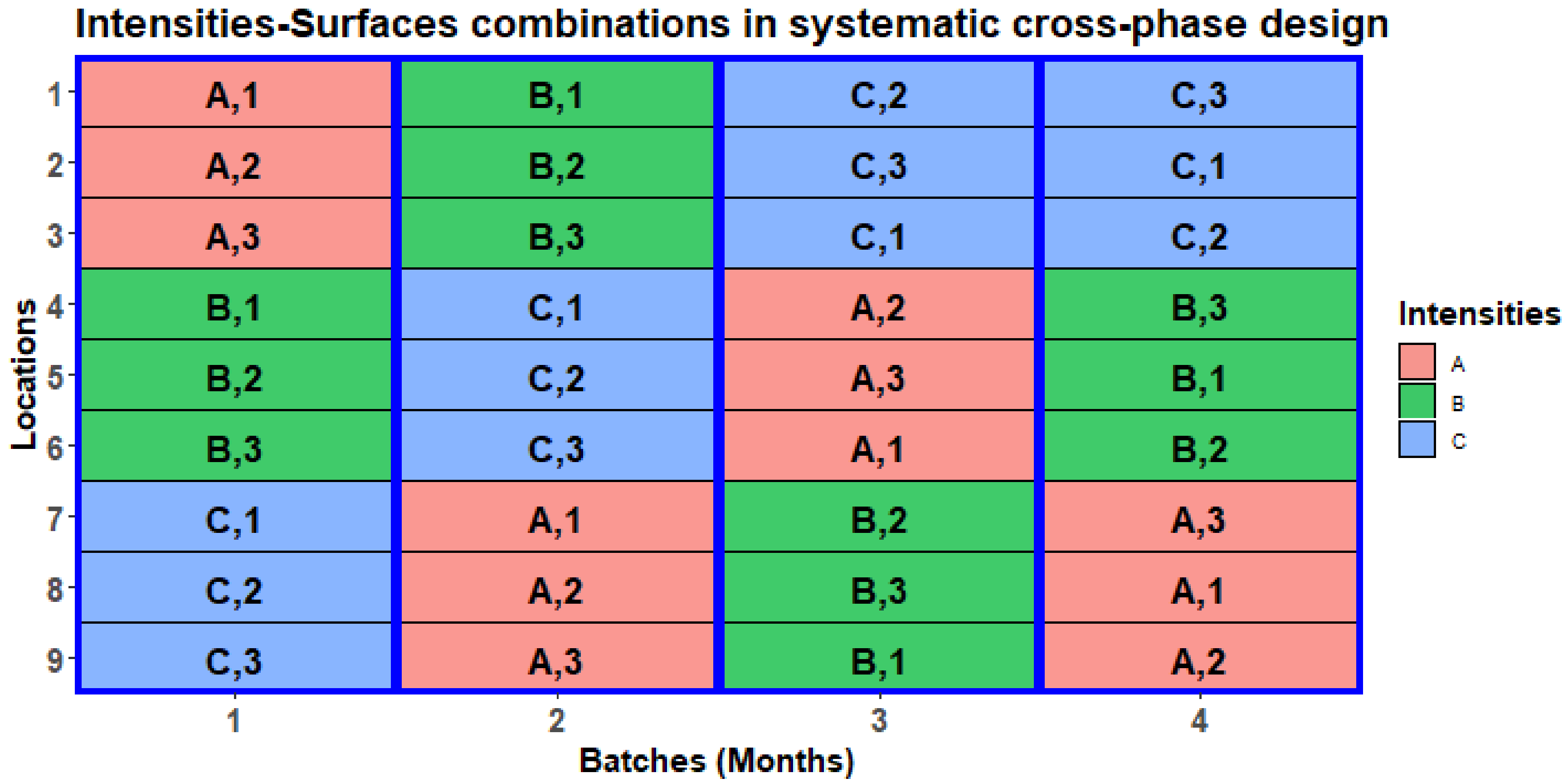
- To produce a good two-phase design, the allocation of the first phase to locations cannot ignore Intensities and Surfaces: a good cross-phase design is needed.
  - In the previous design, they could be ignored because everything within Months was being randomized to Locations within Batches.
- In addition, the second-phase design cannot be ignored because the split-unit nature of the first-phase design must be taken into account.
  - Because of the time constraints Months must be associated with Batches.
  - Within a month, assigning Athletes to triples of consecutive Locations is consistent with the use of a split-unit design in the first-phase.
  - Tests can then be assigned to the locations within a triple.
- Thus the cross-phase design must efficiently assign Intensities to Location triples and Surfaces to the Locations within a triple.

The factor allocation diagram



Systematic cross-phase design

- A balanced factorial design (Hinkelmann & Kempthorne, 2005, section 12.5).



Intensities-Surfaces combinations in systematic cross-phase design

Locations	Batches (Months) 1	2	3	4
1	A,1	B,1	C,2	C,3
2	A,2	B,2	C,3	C,1
3	A,3	B,3	C,1	C,2
4	B,1	C,1	A,2	B,3
5	B,2	C,2	A,3	B,1
6	B,3	C,3	A,1	B,2
7	C,1	A,1	B,2	A,3
8	C,2	A,2	B,3	A,1
9	C,3	A,3	B,1	A,2

Intensities

🟥	A
🟩	B
🟦	C

- A 3 × 4 extended Latin square (LS + column of repeats) is used to allocate Intensities to the triples (colours & letters);
- A 3 × 4 extended Latin square is used for a Locations triple × Batches; the same extended Latin square is used for all 3 triples.
- To ensure no repeat Intensities-Surfaces combinations for a Location, the repeated columns for the two ELSs must be associated with different Batches.
- The Intensities and Surfaces are arranged in a split-unit pattern.

Construct a systematic second-phase design and randomize it

```
> #'## Generate a systematic cross-phase design for Intensities and Surfaces
> eg2.phx.sys <- cbind(fac.gen(list(Batches = 4, Locations = 9)),
+                      data.frame(Intensities = factor(rep(c(designLatinSqrSys(3), c(3,2,1)),
+                                                           each = 3), labels = LETTERS[1:3]),
+                                 Surfaces     = factor(c(rep(1:3, times = 3),
+                                                          rep(1:3, times = 3),
+                                                          rep(c(2,3,1), times = 3),
+                                                          rep(c(3,1,2), times = 3)))))
> #'## Generate a systematic two-phase design by bringing in first-phase recipient factors
> eg2.phx.sys$Months <- eg2.phx.sys$Batches
> eg2.sys <- merge(split.lay, eg2.phx.sys) #merge on commmon factors Months, Intensities & Surfaces
> eg2.sys  <- with(eg2.sys, eg2.sys[order(Batches,Locations),])
> #'## Allocate the second phase
> eg2.lay <- designRandomize(allocated = eg2.sys[c("Months", "Athletes", "Tests",
+                                                  "Intensities", "Surfaces")],
+                            recipient = eg2.sys[c("Batches", "Locations")],
+                            except    = "Batches",
+                            seed      = 243526)
```

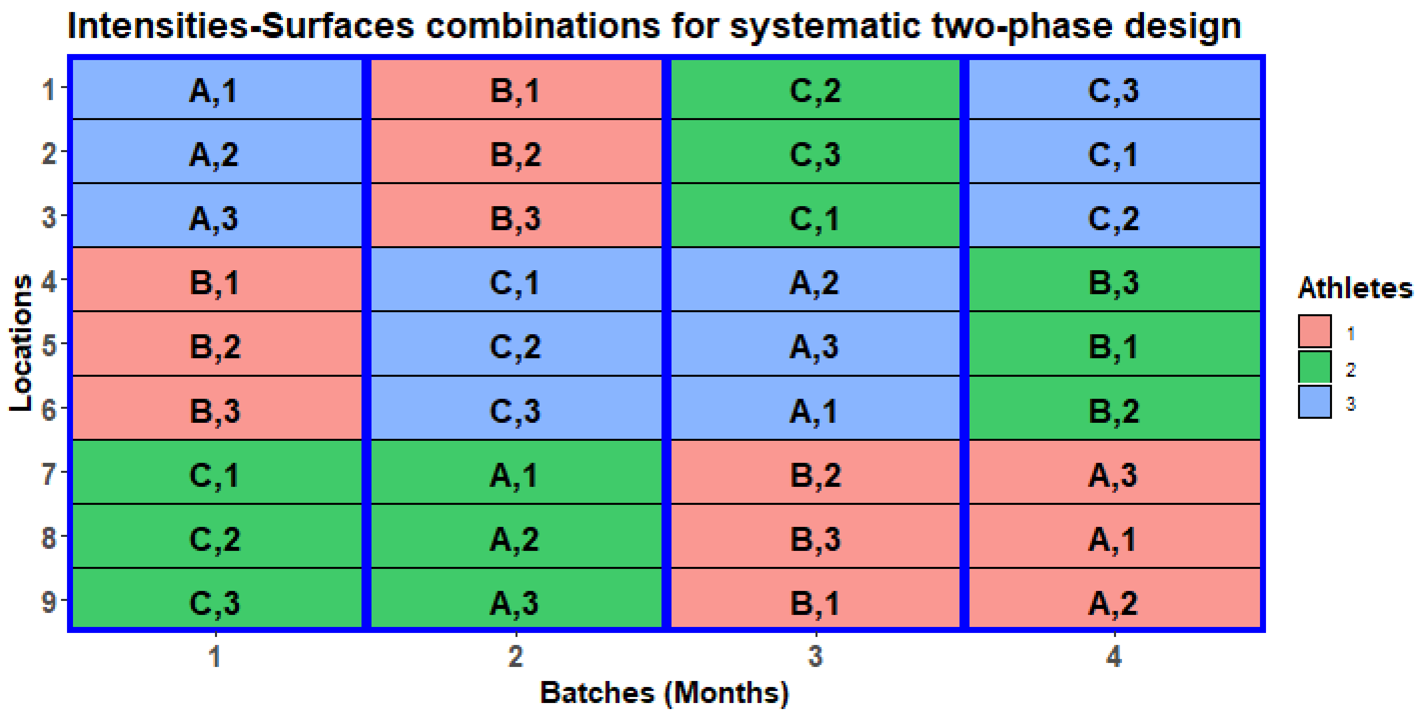
ELSD

An ELSD

Don’t randomize Batches.

Systematic versus randomized designs

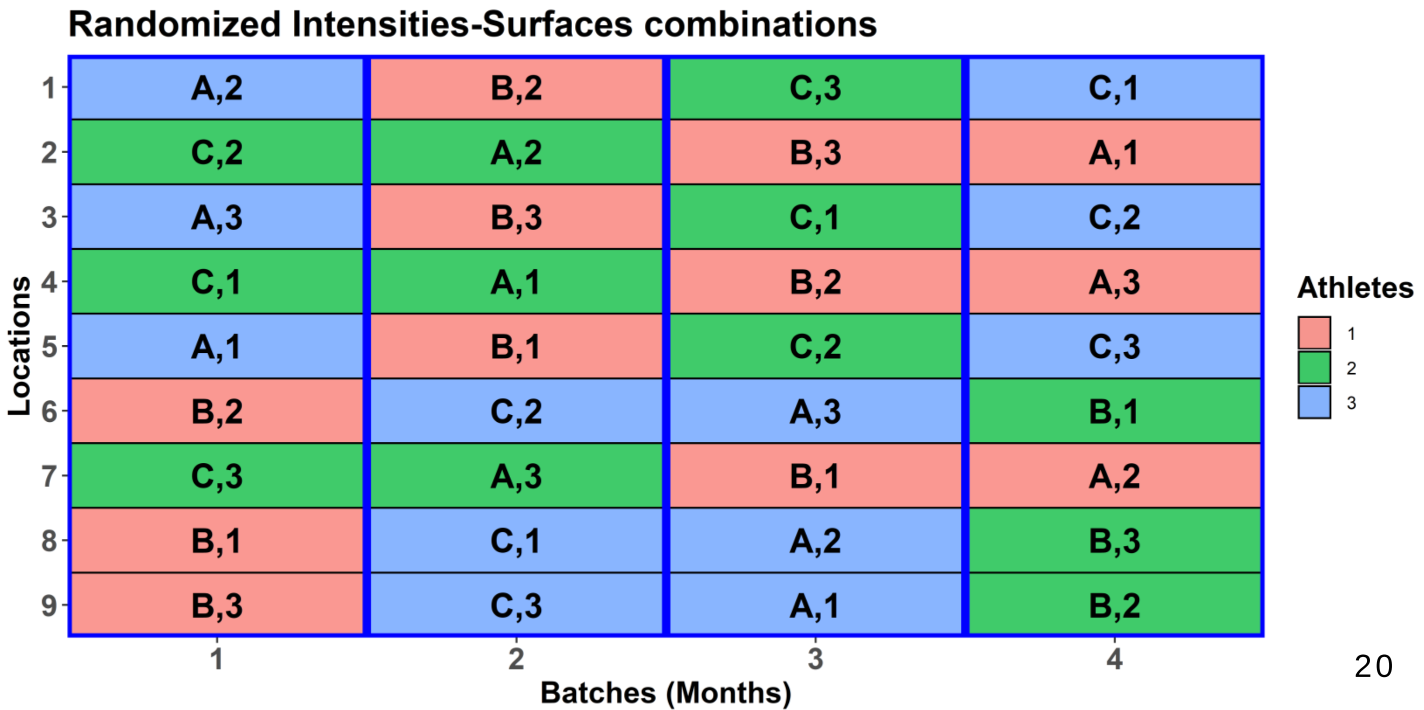
- The randomized design is obtained from the systematic design by permuting:
  - its rows (Locations),
  - but not its columns (Batches).


Intensities-Surfaces combinations for systematic two-phase design

Locations	1	2	3	4
1	🟦 A,1	🟥 B,1	🟩 C,2	🟦 C,3
2	🟦 A,2	🟥 B,2	🟩 C,3	🟦 C,1
3	🟦 A,3	🟥 B,3	🟩 C,1	🟦 C,2
4	🟥 B,1	🟦 C,1	🟦 A,2	🟩 B,3
5	🟥 B,2	🟦 C,2	🟦 A,3	🟩 B,1
6	🟥 B,3	🟦 C,3	🟦 A,1	🟩 B,2
7	🟩 C,1	🟩 A,1	🟥 B,2	🟥 A,3
8	🟩 C,2	🟩 A,2	🟥 B,3	🟥 A,1
9	🟩 C,3	🟩 A,3	🟥 B,1	🟥 A,2

Batches (Months)

Athletes
🟥 1
🟩 2
🟦 3


Randomized Intensities-Surfaces combinations

Locations	1	2	3	4
1	🟦 A,2	🟥 B,2	🟩 C,3	🟦 C,1
2	🟩 C,2	🟩 A,2	🟥 B,3	🟥 A,1
3	🟦 A,3	🟥 B,3	🟩 C,1	🟦 C,2
4	🟩 C,1	🟩 A,1	🟥 B,2	🟥 A,3
5	🟦 A,1	🟥 B,1	🟩 C,2	🟦 C,3
6	🟥 B,2	🟦 C,2	🟦 A,3	🟩 B,1
7	🟩 C,3	🟩 A,3	🟥 B,1	🟥 A,2
8	🟥 B,1	🟦 C,1	🟦 A,2	🟩 B,3
9	🟥 B,3	🟦 C,3	🟦 A,1	🟩 B,2

Batches (Months)

Athletes
🟥 1
🟩 2
🟦 3

Anatomy of the two-phase design allowing for lab processing order

```
Summary table of the decomposition for locs, test & cond (based on adjusted quantities)

 Source.locs       df1 Source.test              df2 Source.cond            df3 aefficiency order
 Batches             3 Months                     3                                 1.0000     1
 Locations           8 Athletes[Months]           2 Intensities              2      0.0625     1
                       Tests[Months:Athletes]     6 Surfaces                 2      0.0625     1
                                                    Intensities#Surfaces     4      0.2500     1
 Batches#Locations  24 Athletes[Months]           6 Intensities              2      0.9375     1
                                                    Residual                 4      1.0000     1
                       Tests[Months:Athletes]    18 Surfaces                 2      0.9375     1
                                                    Intensities#Surfaces     4      0.7500     1
                                                    Residual                12      1.0000     1

The design is not orthogonal
```

Most of the information about Intensities and Surfaces is confounded with **`Batches#Locations`**.

The Residual for Intensities has been reduced from 6 to 4 df.

The design is balanced

Prior allocation model

```
Summary table of the decomposition for locs, test & cond (based on adjusted quantities)

 Source.locs       df1 Source.test             df2 Source.cond          df3 aefficiency order
 Batches             3 Months                    3                             1.0000     1
 Locations           8 Athletes[Months]          2 Intensities            2    0.0625     1
                       Tests[Months:Athletes]    6 Surfaces               2    0.0625     1
                                                   Intensities#Surfaces   4    0.2500     1
 Batches#Locations  24 Athletes[Months]          6 Intensities            2    0.9375     1
                                                   Residual               4    1.0000     1
                       Tests[Months:Athletes]   18 Surfaces               2    0.9375     1
                                                   Intensities#Surfaces   4    0.7500     1
                                                   Residual              12    1.0000     1
```

- Same exhaustive confounding issues as for the nested second-phase design.
- Must retain Locations and Months:Athletes to prevent undesirable pooling.
- One possible prior allocation model is the model for the first phase plus Locations:
  - Months + Intensities + Surfaces + Intensities:Surfaces | Months:Athletes + Months:Athletes:Tests + Locations.
- Again this is a model of convenience and does not portray all the sources of variation affecting the response variables for this experiment

Using od to construct a design

- Split-plot designs
  - involve a two-step process to optimize:
    1. optimize the main-unit factors;
    2. optimize the sub-unit factors, given the main-unit optimization;
  - Only optimizes main effects.

Optimizing the main-unit, cross-phase design

```
> #'## Optimize the main-unit, cross-phase design, based on assigning Intensities to Locations tripletss
> #'### Set up a randomized starting design
> eg2.main.ini <- cbind(fac.gen(list(Batches = 4, Triplets = 3)),
+                       fac.gen(list(Intensities = LETTERS[1:3]), times = 4))
> eg2.main.ini <- designRandomize(allocated         = eg2.main.ini[c("Intensities")],
+                                 recipient         = eg2.main.ini[c("Batches","Triplets")],
+                                 nested.recipients = list(Triplets = "Batches"),
+                                 seed              = 61461)
> #'### Use od to optimize the main-unit design
> eg2.main.od <- od(fixed   = ~ Batches + Triplets + Intensities,
+                   permute = ~ Intensities, swap = ~ Batches,
+                   maxit   = maxit, search = "tabu",
+                   data    = eg2.main.ini)
Mon Oct  7 15:00:29 2019
Initial A-value = 0.727273 (3 A-equations; rank C 2)
A-value after tabu loop 1 is 0.533333
A-value after tabu loop 2 is 0.533333
…
A-value after tabu loop 50 is 0.533333
Hash table size 4
Final A-value after 50 tabu iterations: 0.533333
> eg2.main.des <- eg2.main.od$design
```

Set up an RCBD for **`Intensities`** as a starting design.

Use **`swap`** to keep the design resolved for **`Batches`**.

Optimizing the sub-unit, cross-phase design

```
> #'## Optimize the sub-unit, cross-phase design, based on assigning Surfaces to Locations within triplets
> #'### Set up a randomized starting design
> eg2.ini <- cbind(fac.gen(list(Surfaces = 3), times = 12),
+                  fac.gen(list(Batches = 4, Triplets = 3, Locations = 3)))
> eg2.ini <- designRandomize(allocated         = eg2.ini["Surfaces"],
+                            recipient         = eg2.ini[c("Batches", "Triplets", "Locations")],
+                            nested.recipients = list(Locations = c("Batches", "Triplets")),
+                            except            = c("Batches", "Triplets"),
+                            seed              = 65435)
> eg2.ini$Locations <- with(eg2.ini, fac.combine(list(Triplets, Locations)))
> eg2.ini <- merge(eg2.ini, eg2.main.des[c("Batches", "Triplets", "Intensities")])
> #'### Use od to optimize the sub-unit design
> eg2.od <- od(fixed   = ~ Batches*Triplets + Locations + Surfaces,
+              permute = ~ Surfaces, swap = ~ Batches:Triplets,
+              maxit   = maxit, search = search,
+              data    = eg2.ini)
Mon Oct  7 15:02:14 2019
Initial A-value = 0.191781 (3 A-equations; rank C 2)
A-value after tabu loop 1 is 0.177778
A-value after tabu loop 2 is 0.177778
…
A-value after tabu loop 50 is 0.177778
Final A-value after 50 iterations: 0.177778
```

A starting-design design with **`Surfaces`** randomized within **`Batches-Triplets`**.

Add the main-unit design.

Use **`swap`** to only interchange within **`Batches-Triplets`**, so keeping main-unit design.

Produce the two-phase design based on the od designs

Combine cross-phase and first-phase designs.

```
> eg2.des <- eg2.od$design
> split.lay$Batches <- split.lay$Months
> eg2.lay <- merge(eg2.des, split.lay)
> eg2.lay <- with(eg2.lay, eg2.lay[order(Batches, Locations),])
> eg2.lay <- designRandomize(allocated = eg2.lay[c("Months","Athletes","Tests",
+                                              "Intensities","Surfaces")],
+                            recipient = eg2.lay[c("Batches","Locations")],
+                            except    = "Batches",
+                            seed      = 87620)
> #'## Check properties of the design
> eg2.canon <- designAnatomy(formulae = list(locs = ~ Batches*Locations,
+                                            test = ~ Months/Athletes/Tests,
+                                            cond = ~ Intensities*Surfaces),
+                            data     = eg2.lay)
Warning messages:
1: In projs.2canon(CombinedSets$Q[[ntiers]], struct[[ktier]]$Q) :
  Intensities#Surfaces and Surfaces are partially aliased in Locations&Tests[Months:Athletes]
2: In projs.2canon(CombinedSets$Q[[ntiers]], struct[[ktier]]$Q) :
  Intensities#Surfaces and Surfaces are partially aliased in Batches#Locations&Tests[Months:Athletes]
```

Randomize the first-phase design to the second-phase units.

The interaction and **`Surfaces`** main effects are not orthogonal.

Properties of the od-based two-phase design

```
Summary table of the decomposition for locs, test & cond (based on adjusted quantities)

 Source.locs       df1 Source.test            df2 Source.cond          df3 aefficiency order
 Batches             3 Months                   3                               1.0000     1
 Locations           8 Athletes[Months]         2 Intensities            2      0.0625     1
                       Tests[Months:Athletes]   6 Surfaces               2      0.0625     1
                                                  Intensities#Surfaces   3      0.1121     3
                                                  Residual               1      1.0000     1
 Batches#Locations  24 Athletes[Months]         6 Intensities            2      0.9375     1
                                                  Residual               4      1.0000     1
                       Tests[Months:Athletes]  18 Surfaces               2      0.9375     1
                                                  Intensities#Surfaces   4      0.6559     4
                                                  Residual              12      1.0000     1

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

 Source               df Alias    In                                        aefficiency order
 Intensities#Surfaces 3  Surfaces Locations&Tests[Months:Athletes]               0.1121     3
 Intensities#Surfaces 4  Surfaces Batches#Locations&Tests[Months:Athletes]       0.6559     4

The design is not orthogonal
```

The design is unbalanced and the interaction efficiency is low (cf 0.75 for manual construction).

Properties of the four species of od-based designs

- Used **`designTwophaseAnatomies`** to output the 4 species of designs for a two-phase design.
  - The anatomy of the two-phase design has been presented.
  - The second-phase anatomy is:

```
Summary table of the decomposition for locs & test

 Source.locs       df1 Source.test              df2 aefficiency eefficiency order
 Batches             3 Months                     3      1.0000      1.0000     1
 Locations           8 Athletes[Months]           2      1.0000      1.0000     1
                       Tests[Months:Athletes]     6      1.0000      1.0000     1
 Batches#Locations  24 Athletes[Months]           6      1.0000      1.0000     1
                       Tests[Months:Athletes]    18      1.0000      1.0000     1
```

  - It shows that the allocation of second-phase units (tests) to first-phase units (locations) is orthogonal.
  - This is desirable because it means that the variance matrix is relatively straightforward.

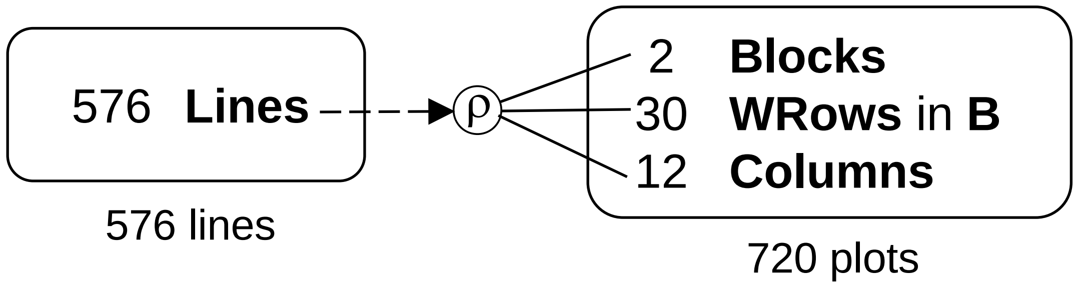
2. Partially replicated (*p*-rep) designs

- These designs were introduced by Cullis et al. (2006)
- They are a variation of the augmented designs, introduced by Federer in 1956.
- An augmented design is one in which a design is used to allocate replicated treatments and these are then augmented with unreplicated treatments.
- The particular features of a *p*-rep design are:
  - Both the unreplicated and replicated treatments are new genotypes; in augmented designs, the unreplicated treatments are usually controls or check varieties;
  - The *p*-rep designs are spatially-optimized.

2.1 A field experiment — a single-phase *p*-rep

(Cullis, Smith & Coombes, 2006)

- 576 Lines on 60 rows × 12 columns.



Dashed line because Lines are allocated to the plots factors, but not using classic randomization.

- 144 Lines are to be duplicated — $p = 0.25$.
- Local spatial correlation is expected and a spatial design is needed.
- The initial allocation model is:
  - Lines | Blocks + Blocks:WRows + Columns + Blocks:Columns + Blocks:WRows:Columns.
- The prior allocation model is:
  - Blocks | Lines + Blocks:WRows + Columns + Blocks:Columns + units + ar1(Blocks:WRows):ar1(Columns).

- **`Lines`** and **`Blocks`** interchanged between fixed-random model;
- Autocorrelation for **`Rows`** and **`Columns`** is added;
- **`units`** is added for nugget variance.

A field *p*-rep — variance parameters

- The prior allocation model:
  - Blocks | Lines + Blocks:WRows + Columns +
    Blocks:Columns + units + ar1(Blocks:WRows):ar1(Columns).
- To search for a spatially-optimized design using **od** need to specify values for the variance parameters.
- The general way to do this is to
  i. set the residual (or identity) term component to 1: $\phi_{\mathrm{BRC}} = 1$;
  ii. Use γ to denote the ratio of each component to the residual: $\gamma_i = \phi_i \,/\, \phi_{\mathrm{BRC}}$.
- Suppose past experience tells us that the following are reasonable values (Smith et al, 2006, p.405):
  - $\gamma_{\mathrm{L}} = 1$, $\gamma_{\mathrm{BR}} = 0.5$, $\gamma_{\mathrm{C}} = 0.1$, $\gamma_{\mathrm{BC}} = 0.05$, $\gamma_{\mathrm{u}} = 0.5$, $\phi_{\mathrm{BRC}} = 1$, $\rho_{\mathrm{BR}} = 0.6$, $\rho_{\mathrm{C}} = 0.4$.
  - The magnitude of ϕ_{L} equals that of ϕ_{BRC}; γ_{u} is the nugget variance; the ρs are the first-order autocorrelation parameters.

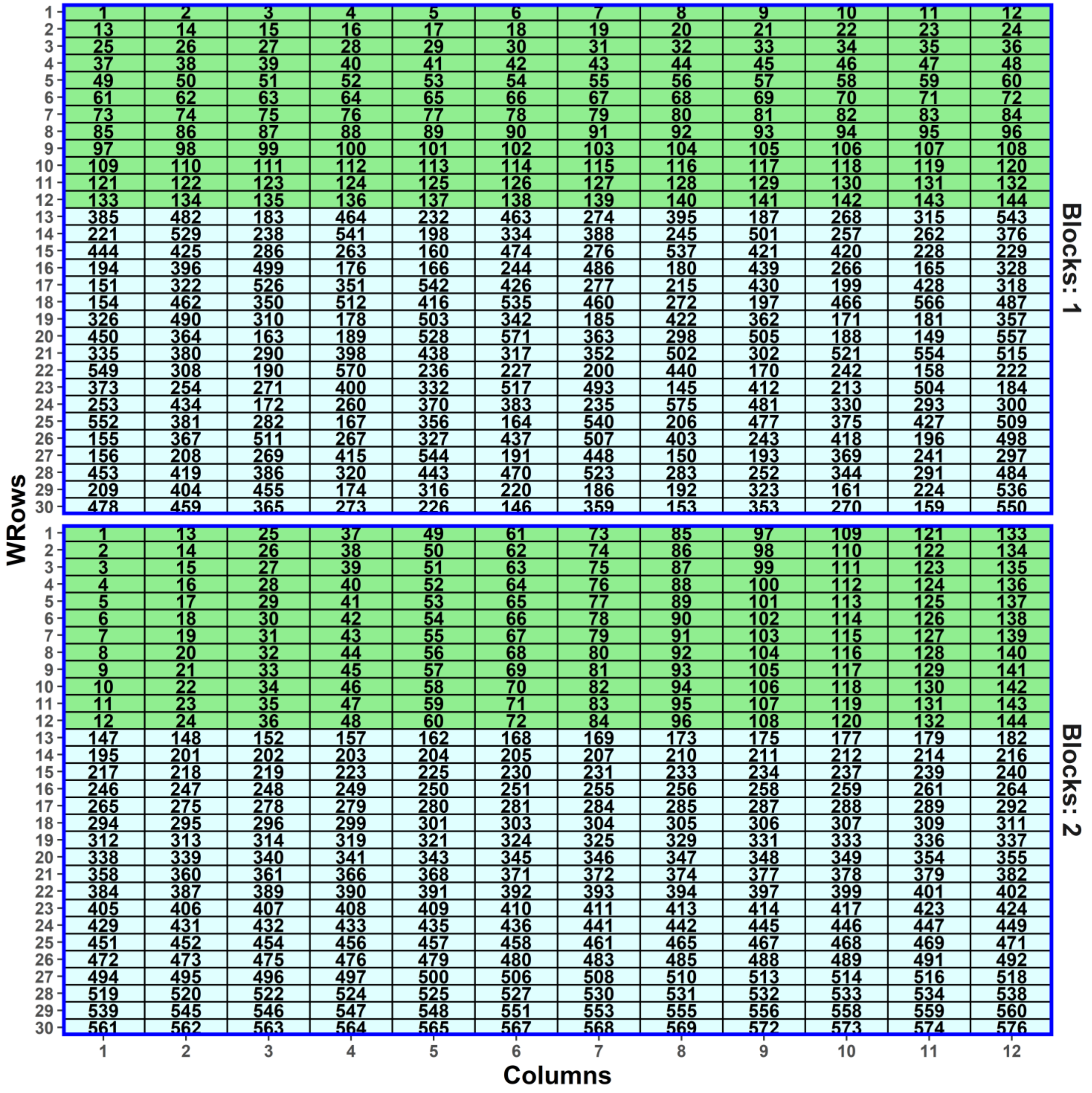
A field *p*-rep — setting up

```
> #'## Set up constants
> g <- 576    # no. genotypes
> ndup <- 144 # no. duplicated genotypes
> b <- 2      # no. blocks
> r <- 60     # no. rows
> c <- 12     # no. columns
> n <- r*c    # no. Plots
>
> #'## Set up variance parameters
> g.L <- 1
> g.BR <- 0.5
> g.C <- 0.1
> g.BC <- 0.05
> g.u <- 0.5
> g.BRC <- 1.0
> rho.R <- 0.6
> rho.C <- 0.4
> params <- c(g.L, g.BR, g.C, g.BC, g.u, g.BRC, rho.R, rho.C)
> names(params) <- c("g.L", "g.BR", "g.C", "g.BC", "g.u", "g.BRC", "rho.R", "rho.C")
```

```
> #'## Set od options
> maxit <- 50
> search <- "tabu+rw"
> od.options(P = 0.10, localSearch = 10000, tabuStop = 100)
```

A field *p*-rep — initial design

```
> #'## Generate a simple lattice for Lines 1:144
> #'
> #' 1:144 are replicated twice 145:g are replicat
> latt.mat <- matrix(1:ndup, nrow = 12, ncol = 12)
> blk1.lines <- sample((ndup+1):g, (g-ndup)/2)
> blk2.lines <- ((ndup+1):g)[!((ndup+1):g %in% blk
> latt.lay <- fac.gen(list(Blocks = 2, WRows = 30,
> latt.lay <- within(latt.lay,
+                      Lines <- factor(c(latt.mat,
+                                        t(latt.ma

> #'## Randomize the initial design
> latt.lay <- designRandomize(allocated         =
+                            recipient         =
+                            nested.recipients =
+                            seed              = (
> latt.lay <- within(latt.lay,
+                    Rows <- fac.combine(list(Bl
```



Blocks	WRows	1	2	3	4	5	6	7	8	9	10	11	12
1	1	1	2	3	4	5	6	7	8	9	10	11	12
1	2	13	14	15	16	17	18	19	20	21	22	23	24
1	3	25	26	27	28	29	30	31	32	33	34	35	36
1	4	37	38	39	40	41	42	43	44	45	46	47	48
1	5	49	50	51	52	53	54	55	56	57	58	59	60
1	6	61	62	63	64	65	66	67	68	69	70	71	72
1	7	73	74	75	76	77	78	79	80	81	82	83	84
1	8	85	86	87	88	89	90	91	92	93	94	95	96
1	9	97	98	99	100	101	102	103	104	105	106	107	108
1	10	109	110	111	112	113	114	115	116	117	118	119	120
1	11	121	122	123	124	125	126	127	128	129	130	131	132
1	12	133	134	135	136	137	138	139	140	141	142	143	144
1	13	385	482	183	464	232	463	274	395	187	268	315	543
1	14	221	529	238	541	198	334	388	245	501	257	262	376
1	15	444	425	286	263	160	474	276	537	421	420	228	229
1	16	194	396	499	176	166	244	486	180	439	266	165	328
1	17	151	322	526	351	542	426	277	215	430	199	428	318
1	18	154	462	350	512	416	535	460	272	197	466	566	487
1	19	326	490	310	178	503	342	185	422	362	171	181	357
1	20	450	364	163	189	528	571	363	298	505	188	149	557
1	21	335	380	290	398	438	317	352	502	302	521	554	515
1	22	549	308	190	570	236	227	200	440	170	242	158	222
1	23	373	254	271	400	332	517	493	145	412	213	504	184
1	24	253	434	172	260	370	383	235	575	481	330	293	300
1	25	552	381	282	167	356	164	540	206	477	375	427	509
1	26	155	367	511	267	327	437	507	403	243	418	196	498
1	27	156	208	269	415	544	191	448	150	193	369	241	297
1	28	453	419	386	320	443	470	523	283	252	344	291	484
1	29	209	404	455	174	316	220	186	192	323	161	224	536
1	30	478	459	365	273	226	146	359	153	353	270	159	550
2	1	1	13	25	37	49	61	73	85	97	109	121	133
2	2	2	14	26	38	50	62	74	86	98	110	122	134
2	3	3	15	27	39	51	63	75	87	99	111	123	135
2	4	4	16	28	40	52	64	76	88	100	112	124	136
2	5	5	17	29	41	53	65	77	89	101	113	125	137
2	6	6	18	30	42	54	66	78	90	102	114	126	138
2	7	7	19	31	43	55	67	79	91	103	115	127	139
2	8	8	20	32	44	56	68	80	92	104	116	128	140
2	9	9	21	33	45	57	69	81	93	105	117	129	141
2	10	10	22	34	46	58	70	82	94	106	118	130	142
2	11	11	23	35	47	59	71	83	95	107	119	131	143
2	12	12	24	36	48	60	72	84	96	108	120	132	144
2	13	147	148	152	157	162	168	169	173	175	177	179	182
2	14	195	201	202	203	204	205	207	210	211	212	214	216
2	15	217	218	219	223	225	230	231	233	234	237	239	240
2	16	246	247	248	249	250	251	255	256	258	259	261	264
2	17	265	275	278	279	280	281	284	285	287	288	289	292
2	18	294	295	296	299	301	303	304	305	306	307	309	311
2	19	312	313	314	319	321	324	325	329	331	333	336	337
2	20	338	339	340	341	343	345	346	347	348	349	354	355
2	21	358	360	361	366	368	371	372	374	377	378	379	382
2	22	384	387	389	390	391	392	393	394	397	399	401	402
2	23	405	406	407	408	409	410	411	413	414	417	423	424
2	24	429	431	432	433	435	436	441	442	445	446	447	449
2	25	451	452	454	456	457	458	461	465	467	468	469	471
2	26	472	473	475	476	479	480	483	485	488	489	491	492
2	27	494	495	496	497	500	506	508	510	513	514	516	518
2	28	519	520	522	524	525	527	530	531	532	533	534	538
2	29	539	545	546	547	548	551	553	555	556	558	559	560
2	30	561	562	563	564	565	567	568	569	572	573	574	576

This is a resolved, augmented design – the replicates of the duplicated Lines are in different blocks and the unduplicated lines are added to the block design.

A field *p*-rep — setting variance parameters in od

```
> #'## Use od to generate the p-rep starting with the simple lattice - with units and autocorrelation
> prepuar1.latt.od <- od(fixed        = ~ Blocks,
+                        random       = ~ Lines + Rows + Columns/Blocks + units,
+                        residual     = ~ ar1(Rows):ar1(Columns),
+                        permute      = ~ Lines, swap = ~ Blocks,
+                        start.values = TRUE,
+                        data         = latt.lay)
> vp.table <- prepuar1.latt.od$vparameters.table
> vp.table$Value <- params
> vp.table
                Component Value
1                   Lines  1.00
2                    Rows  0.50
3                 Columns  0.10
4         Columns:Blocks  0.05
5                   units  0.50
6          Rows:Columns!R  1.00
7    Rows:Columns!Rows!cor  0.60
8 Rows:Columns!Columns!cor  0.40
```

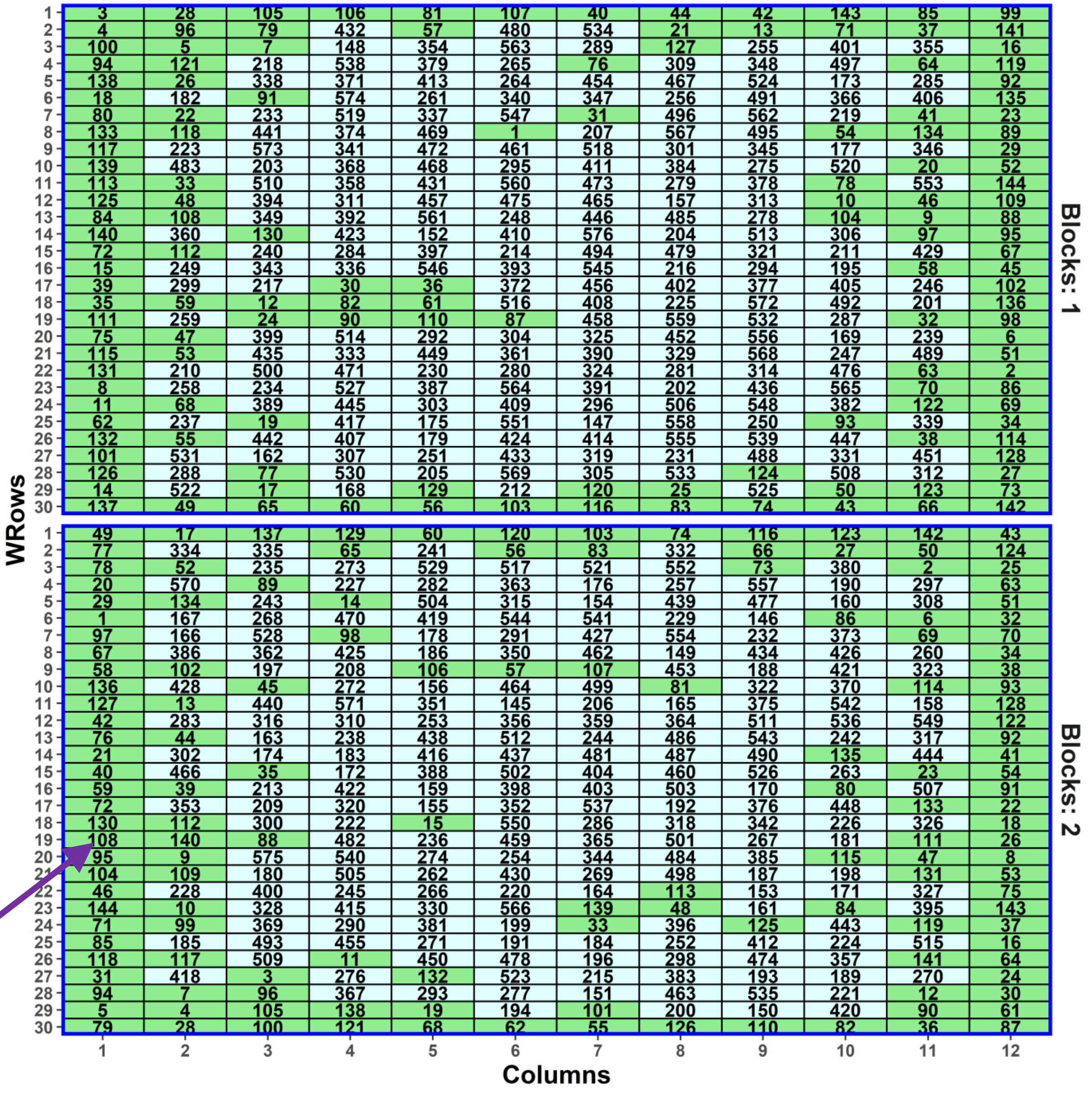
Note **`Rows`** used rather than **`Blocks:Rows`**: **`ar1`** requires a single factor; the two terms are equivalent as a random term.

`swap` restricts interchanges to be within **`Blocks`** and so ensures that the design remains resolved.

A field *p*-rep — generating the design

```
> prepuar1.latt.od <- od(fixed    = ~ Blocks,
+                        random   = ~ Lines + Rows + Co
+                        residual = ~ ar1(Rows):ar1(Col
+                        permute  = ~ Lines, swap = ~ E
+                        G.param  = vp.table, R.param =
+                        maxit    = maxit, search = sea
+                        data     =l att.lay)
Fri Sep  6 16:52:46 2019
Initial A-value = 1.026048 (576 A-equations; rank C 576
A-value after tabu loop 1 is 1.016726
A-value after tabu loop 2 is 1.016552

…

A-value after tabu loop 49 is 1.016174
A-value after tabu loop 50 is 1.016174
Hash table size 2477
Final A-value after 50 iterations: 1.016174
> prepuar1.latt.lay <- prepuar1.latt.od$design
```

Note that all the border plots are duplicated **`Lines`**; the same does not occur when an RCBD is used as the starting design (in Prac).



Blocks: 1

	1	2	3	4	5	6	7	8	9	10	11	12
1	3	28	105	106	81	107	40	44	42	143	85	99
2	4	96	79	432	57	480	534	21	13	71	37	141
3	100	5	7	148	354	563	289	127	255	401	355	16
4	94	121	218	538	379	265	76	309	348	497	64	119
5	138	26	338	371	413	264	454	467	524	173	285	92
6	18	182	91	574	261	340	347	256	491	366	406	135
7	80	22	233	519	337	547	31	496	562	219	41	23
8	133	118	441	374	469	1	207	567	495	54	134	89
9	117	223	573	341	472	461	518	301	345	177	346	29
10	139	483	203	368	468	295	411	384	275	520	20	52
11	113	33	510	358	431	560	473	279	378	78	553	144
12	125	48	394	311	457	475	465	157	313	10	46	109
13	84	108	349	392	561	248	446	485	278	104	9	88
14	140	360	130	423	152	410	576	204	513	306	97	95
15	72	112	240	284	397	214	494	479	321	211	429	67
16	15	249	343	336	546	393	545	216	294	195	58	45
17	39	299	217	30	36	372	456	402	377	405	246	102
18	35	59	12	82	61	516	408	225	572	492	201	136
19	111	259	24	90	110	87	458	559	532	287	32	98
20	75	47	399	514	292	304	325	452	556	169	239	6
21	115	53	435	333	449	361	390	329	568	247	489	51
22	131	210	500	471	230	280	324	281	314	476	63	2
23	8	258	234	527	387	564	391	202	436	565	70	86
24	11	68	389	445	303	409	296	506	548	382	122	69
25	62	237	19	417	175	551	147	558	250	93	339	34
26	132	55	442	407	179	424	414	555	539	447	38	114
27	101	531	162	307	251	433	319	231	488	331	451	128
28	126	288	77	530	205	569	305	533	124	508	312	27
29	14	522	17	168	129	212	120	25	525	50	123	73
30	137	49	65	60	56	103	116	83	74	43	66	142

Blocks: 2

	1	2	3	4	5	6	7	8	9	10	11	12
1	49	17	137	129	60	120	103	74	116	123	142	43
2	77	334	335	65	241	56	83	332	66	27	50	124
3	78	52	235	273	529	517	521	552	73	380	2	25
4	20	570	89	227	282	363	176	257	557	190	297	63
5	29	134	243	14	504	315	154	439	477	160	308	51
6	1	167	268	470	419	544	541	229	146	86	6	32
7	97	166	528	98	178	291	427	554	232	373	69	70
8	67	386	362	425	186	350	462	149	434	426	260	34
9	58	102	197	208	106	57	107	453	188	421	323	38
10	136	428	45	272	156	464	499	81	322	370	114	93
11	127	13	440	571	351	145	206	165	375	542	158	128
12	42	283	316	310	253	356	359	364	511	536	549	122
13	76	44	163	238	438	512	244	486	543	242	317	92
14	21	302	174	183	416	437	481	487	490	135	444	41
15	40	466	35	172	388	502	404	460	526	263	23	54
16	59	39	213	422	159	398	403	503	170	80	507	91
17	72	353	209	320	155	352	537	192	376	448	133	22
18	130	112	300	222	15	550	286	318	342	226	326	18
19	108	140	88	482	236	459	365	501	267	181	111	26
20	95	9	575	540	274	254	344	484	385	115	47	8
21	104	109	180	505	262	430	269	498	187	198	131	53
22	46	228	400	245	266	220	164	113	153	171	327	75
23	144	10	328	415	330	566	139	48	161	84	395	143
24	71	99	369	290	381	199	33	396	125	443	119	37
25	85	185	493	455	271	191	184	252	412	224	515	16
26	118	117	509	11	450	478	196	298	474	357	141	64
27	31	418	3	276	132	523	215	383	193	189	270	24
28	94	7	96	367	293	277	151	463	535	221	12	30
29	5	4	105	138	19	194	101	200	150	420	90	61
30	79	28	100	121	68	62	55	126	110	82	36	87

Canonical analysis of the design: investigating its anatomy

- Want to look at the relationships of the lines sources to the plots sources.
- The plots sources:
  - Blocks + Rows[Blocks] + Columns + Blocks#Columns +Rows#Columns[Blocks].
- The lines source:
  - Lines.
- Using **dae**:

- A is the harmonic mean of the efficiency factors.
- **M is the mean of the efficiency factors.**
- E is the minimum of the efficiency factors.
- **dforth is the number of efficiency factors equal to one.**
- Order is the number of unique efficiency factors.

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

A field *p*-rep — anatomy

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

 Source.plot          df1 Source.trt df2 aefficiency mefficiency eefficiency order dforthog
 Blocks                 1 Lines        1      0.6000      0.6000      0.6000     1        0
 Rows[Blocks]          58 Lines       58      0.7849      0.8000      0.4840    58        0
 Columns               11 Lines       11      0.7713      0.7818      0.5731    11        0
 Blocks#Columns        11 Lines       11      0.8126      0.8182      0.6727    11        0
 Rows#Columns[Blocks] 638 Lines      575      0.4135      0.8877      0.0061    82      494
                          Residual    63
The design is not orthogonal
```

- All of the plots sources are orthogonal (no aliasing).

- A is the harmonic mean of the efficiency factors.
- **M is the mean of the efficiency factors.**
- E is the minimum of the efficiency factors.
- **dforth is the number of efficiency factors equal to one.**
- Order is the number of unique efficiency factors.

- A lot of information about some Lines contrasts in other than **`Rows#Columns[Blocks]`** (plots).
- Not a unique decomposition, but **`Rows#Columns[Blocks]`** decomposition is.
- Concentrate on the last Lines source, where all 575 Lines df are partially confounded.

A field *p*-rep — anatomy

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

 Source.plot          df1 Source.trt df2 aefficiency mefficiency eefficiency order dforthog
 Blocks                 1 Lines        1      0.6000      0.6000      0.6000     1        0
 Rows[Blocks]          58 Lines       58      0.7849      0.8000      0.4840    58        0
 Columns               11 Lines       11      0.7713      0.7818      0.5731    11        0
 Blocks#Columns        11 Lines       11      0.8126      0.8182      0.6727    11        0
 Rows#Columns[Blocks] 638 Lines      575      0.4135      0.8877      0.0061    82      494
                          Residual    63
```

Lines efficiencies							
≤0.1	0.1-0.2	0.2-0.3	0.3-0.4	0.4-0.5	0.5-0.6	…	1
23	21	17	12	7	1		494

- A is the harmonic mean of the efficiency factors.
- M is the mean of the efficiency factors (the sum is the Fisher information for the design and is a component of (M,S optimality).
- dforth is the number of efficiency factors equal to one.

- A lot (86%) of orthogonal df in Plots.
- But, a lot of efficiencies close to 0 in Plots, which is to be expected for for *p*-rep designs
  — distorts A so M better?

The effect on the anatomy of assuming that Blocks#Columns is zero

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

 Source.plot          df1 Source.trt df2 aefficiency mefficiency eefficiency order dforthog
 Blocks                 1 Lines        1      0.6000      0.6000      0.6000     1        0
 Rows[Blocks]          58 Lines       58      0.7849      0.8000      0.4953    58        0
 Columns               11 Lines       11      0.7729      0.7818      0.6024    11        0
 Blocks#Rows#Columns  649 Lines      575      0.5168      0.9033      0.0119    71      505
                          Residual    74

The design is not orthogonal
```

- More lines information in Blocks#Rows#Columns and more Residual df.
- Still some information about Lines almost orthogonal to Blocks#Rows#Columns.
- AVPD = 1.014ϕ_{BRC}. (minor change – was 1.016ϕ_{BRC}.)

Calculating the A-measure (AVPD) using `designAmeasures` and `mat.Vpredicts` from dae

- The model arguments of the **od** call

```
> prepuar1.latt.od<- od(fixed    = ~ Blocks,
+                      random   = ~ Lines + Rows + Columns/Blocks + units,
+                      residual = ~ ar1(Rows):ar1(Columns),
+                      permute  = ~ Lines, …)
```

- Corresponding **`designAmeasures`** call

```
> prepuar1.latt.lay$unit <- factor(1:nrow(prepuar1.latt.lay)) #factor for ASReml units
> (designAmeasures(mat.Vpredicts(target = ~ Lines -1,
+                                Gt     = 1,
+                                fixed  = ~ Blocks,
+                                random = ~ Rows + Columns/Blocks + unit - 1,
+                                G      = as.list(params[c("g.BR", "g.C", "g.BC", "g.u")]),
+                                R      = kronecker(mat.ar1(params["rho.R"], r),
+                                                   mat.ar1(params["rho.C"], c)),
+                                design = prepuar1.latt.lay)))[[1]]
[1] 1.016151
```

Matches **`permute`**.

As in **od** call, minus **`Lines`**.

As in **`vp.table`**.

As for **`residual`**.

- To calculate without **`Columns:Blocks`**, drop "**`/Blocks`**" and "**`g.BC`**".
- What happens if ar1 and nugget variance are dropped from **od** call?

Comparing spatial and nonspatial designs

Design	Nonspatial A	Spatial A	aefficiency	mefficiency	eefficiency	dforthog
Nonspatial	0.988486	1.018665	0.5168	0.8877	0.0264	494
Spatial	0.988857	1.016151	0.4180	0.8877	0.0080	494

- Both designs are equally suitable for nonspatial data.
- The difference between the designs for spatial data is very small.
  - The only differences are in aefficiency and eefficiency.
  - The nonspatial design is slightly better because the range of the efficiency factors is less.

Comparing canonical analysis and A-measures (AVPD)

- Canonical analysis
  - Shows the anatomy of the design: where the information is in the design and the nonorthogonality that is present.
  - Do not need to specify the variance parameter values and not dependent on them.
  - Does not account for spatial correlation and nonlinear trends.
  - Limited relationship with AVPD
    - When target is fixed, variance-components-only model and equally replicated, aefficiency is directly related to AVPD, otherwise it is not.
  - Only useful for characterizing a design, rather than searching for an optimal design.
- AVPD
  - Is a measure of the precision in the experiment that gives equal weight to all contrasts, and is used by **od**, but is not the same as PEV.
  - Need to specify the variance parameter values because depends on them.

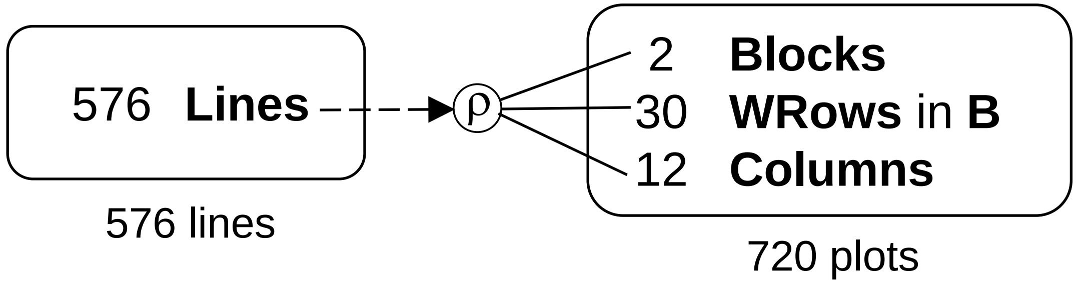
2.2 Partially replicated designs in two phases

- Smith et al. (2006) give examples of experiments that employ designs in which both phases employ partially replicated designs in both phases:
  - They are dubbed *p*/*q*-rep designs.
  - That is, *p*% of the lines are replicated in the first phase and *q*% of the plots with unreplicated lines are replicated in the second phase.
- We will produce a design for a an experiment with p = 0.25 and q = 0.10.
  - Previous example is a *p*-rep design for a field experiment, with p = 0.25.
  - It will be extended to include a milling phase.

The first phase design— a *p*-rep field experiment

(Cullis, Smith & Coombes, 2006)

- 576 Lines on 60 rows × 12 columns.

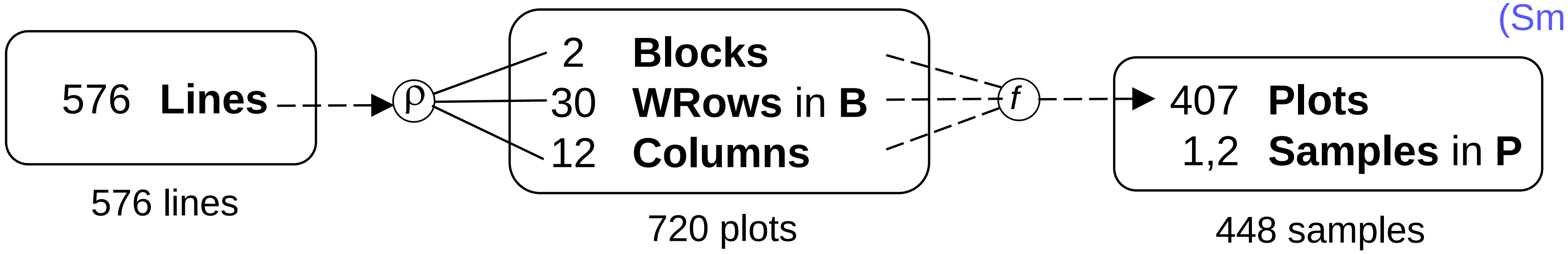


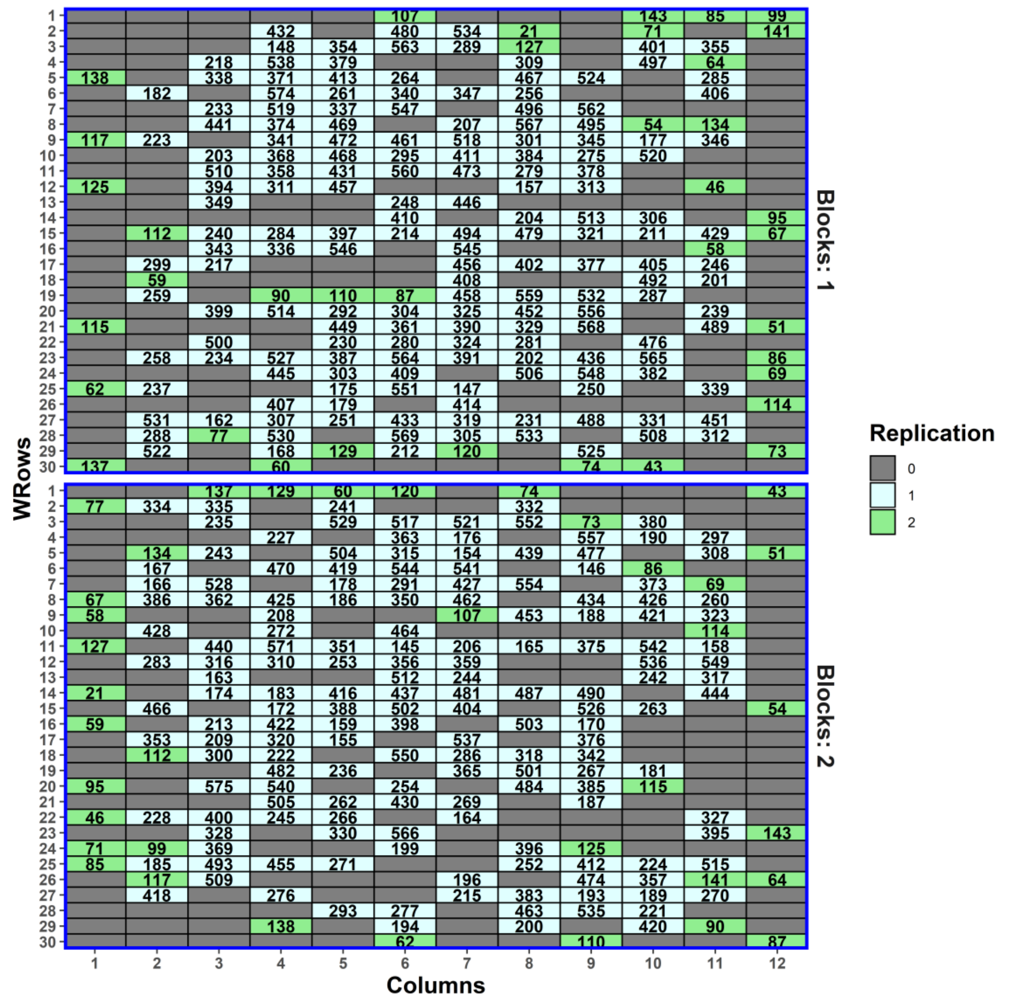
Dashed line because Lines are allocated to the plots factors, but not using classic randomization.

- 144 Lines are to be duplicated — $p = 0.25$.
- A spatially optimized design was used to allocate lines to plots.
- Suppose that samples of grain from the field experiment are to be taken to the laboratory for milling and analysis in the laboratory.
  - After the field experiment 370 lines have been identified for processing in the milling phase.

Sampling plots for the milling (second) phase

(Smith, Lim & Cullis, 2006)





- Take 333 unduplicated and 37 duplicated lines on to milling phase (= 370 lines on 407 plots).
- Of the 333 unduplicated Lines, 41 are duplicated (2 samples required) in the milling phase — q = 0.10 (of plots).
- What will happen here as compared to previous design?
  - Answer: Blocks, Rows and Columns will no longer be orthogonal — unit terms are partially aliased (cf. confounding).
  - Also, Lines confounding will change.

First-phase anatomy for the fraction (without `Blocks#Columns`)

```
> summary(designAnatomy(formulae   = list(plot = ~ ((Blocks/WRows)*Cols)/Samp,
+                                         trt  = ~ Lines),
+                       keep.order = TRUE, data = layout),
+         which.criteria = c("ae", "me", "ee", "dfor"))

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

 Source      df Alias         In   aefficiency mefficiency eefficiency dforthog
 Cols        11 Blocks        plot      0.9992      0.9992      0.9908       10
 Cols        11 WRows[Blocks] plot      0.9230      0.9249      0.8374        0
 Blocks#Cols 22 WRows[Blocks] plot      0.9210      0.9240      0.8151        0
```

- The terms are fitted in the order **`Blocks`**, **`WRows[Blocks]`** and **`Columns`** (see next slide).
- Eleven df for **`Columns`** is aliased with **`Blocks`** and **`WRows[Blocks]`** but 92.3% of the information is retained.
- The analysis will depend on whether **`Columns`** is fitted first or not, but not greatly given the high **`aefficiency`**.

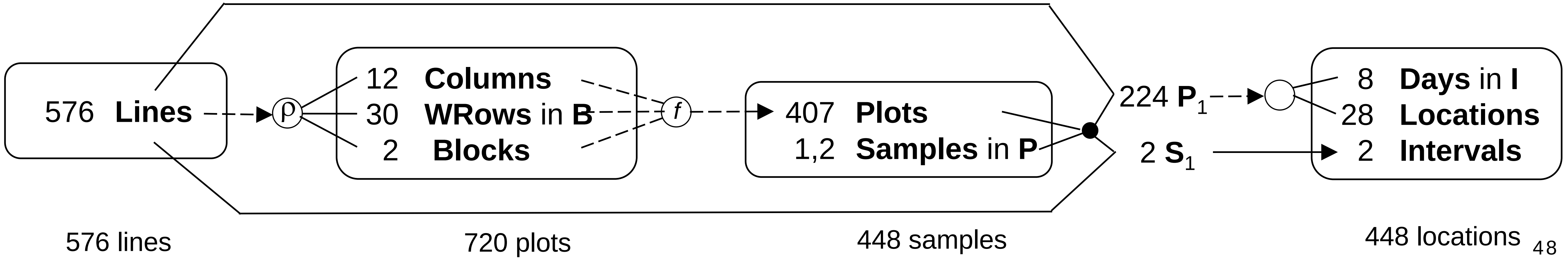
First-phase anatomy for the fraction

```
Source.plot                df1 Source.trt df2 aefficiency mefficiency eefficiency dforthog
Blocks                       1 Lines        1      0.8348      0.8348      0.8348        0
WRows[Blocks]               58 Lines       58      0.7742      0.9013      0.0950       26
Cols                        11 Lines       11      0.8347      0.8760      0.5187        0
Blocks#Cols                 11 Lines       11      0.8231      0.8695      0.4784        0
WRows#Cols[Blocks]         325 Lines      325      0.4602      0.9129      0.0177      288
Samp[Blocks:WRows:Cols]     41
```

- Not unique, but the **`WRows#Cols[Blocks]`** strata is.
- Of the 369 Lines df, 325 are estimable in **`WRows#Cols[Blocks]`**, including 288 (78%) only there.
- There are 44 **`Lines`** df estimable elsewhere, with 26 of these orthogonally confounded with **`WRows[Blocks]`**.
  - Thus for **`Lines`** fixed, the design is disconnected for all plot terms fixed except the last two.
  - Would be connected if all plots terms (except **`Blocks`**) random (needed for **`od`**).
- The mefficiency for the 369 Lines df in **`WRows#Cols[Blocks]`** is 0.8040 (= 0.9129 x 325 / 369).
- **`Samp[Blocks:WRows:Cols]`** (Error) has full 41 df.

Milling-phase allocation for the *p*/*q*-rep design

- There are 448 (407 + 41) samples and so 448 time-locations for milling required:
  - Take 16 days divide them into 2 intervals.
  - Each day there are 28 time-locations for milling.
- Samples are assigned to locations using two pseudofactors, S_1 and P_1:
  - The 448 samples are assigned to the 2 levels of S_1 so that milling duplicates have different levels and, as far as is possible, so do plots from different blocks;
  - The 224 plots in each level of S_1 are assigned to the 224 levels of the pseudofactor P_1 in Rows-Columns order:
    - The 224 plots are comprised of those (i) for the 41 lines that are milling-duplicated, (ii) from the same block for the 37 lines that are field duplicated, and (iii) for 183 lines that are from the same block as (ii) or rows nearby.
  - S_1 is randomized to Intervals and P_1 is systematically allocated to the Days-Locations combinations, the design being nonorthogonal



Check properties of the multiphase design

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

- Note three formulae supplied.
- Have used **`designTwophaseAnatomies`** and this will produce the four species of designs for a two-phase design:
  - The first-phase design for the fraction is not used for the analysis of first-phase responses (e.g. grain yield).

Anatomy of the second-phase design

```
Summary table of the decomposition for lab & plot (based on adjusted quantities)

 Source.lab      df1 Source.plot         df2 aefficiency mefficiency eefficiency dforthog
 Int               1 Blocks                1      0.6386      0.6386      0.6386        0
 Days[Int]        14 Blocks                1      0.2827      0.2827      0.2827        0
                     WRows[Blocks]        13      0.6963      0.7831      0.2458        0
 Locn             27 Blocks                1      0.0061      0.0061      0.0061        0
                     WRows[Blocks]        26      0.0130      0.0974      0.0023        0
 Int#Locn         27 Blocks                1      0.0037      0.0037      0.0037        0
                     WRows[Blocks]        26      0.0140      0.0903      0.0027        0
 Days#Locn[Int] 378 Blocks                1      0.0689      0.0689      0.0689        0
                     WRows[Blocks]        58      0.2760      0.7074      0.0234        0
                     Cols                 11      0.8298      0.8336      0.7439        0
                     Blocks#Cols          11      0.8251      0.8304      0.7058        0
                     WRows#Cols[Blocks] 297      0.4358      0.8991      0.0142      256
```

A lot of **`Blocks`** confounded here.

`Blocks#Cols` mainly confounded here.

The estimable df for **`Wrows#Cols[Blocks]`** has gone from 638 (first-phase) to 325 (fraction) to 297.

Much of **`WRows[Blocks]`** and **`Cols`** confounded here.

Anatomy of the two-phase design

```
Source.lab     df1 Source.plot        df2 Source.trt df3 aefficiency mefficiency eefficiency dforthog
Int              1 Blocks               1 Lines        1      0.6696      0.6696      0.6696        0
Days[Int]       14 Blocks               1 Lines        1      0.6679      0.6679      0.6679        0
                   WRows[Blocks]       13 Lines       13      0.8261      0.8449      0.5362        0
Locn            27 Blocks               1 Lines        1      0.8062      0.8062      0.8062        0
                   WRows[Blocks]       26 Lines       26      0.8135      0.8248      0.6160        0
Int#Locn        27 Blocks               1 Lines        1      0.8050      0.8050      0.8050        0
                   WRows[Blocks]       26 Lines       26      0.8187      0.8279      0.6432        0
Days#Locn[Int] 378 Blocks               1 Lines        1      0.4723      0.4723      0.4723        0
                   WRows[Blocks]       58 Lines       58      0.7908      0.8443      0.3224        0
                   Cols                11 Lines       11      0.8309      0.8597      0.5507        0
                   Blocks#Cols         11 Lines       11      0.8304      0.8587      0.5412        0
                   WRows#Cols[Blocks] 297 Lines      297      0.2940      0.8207      0.0101      219
```

- Just 297 of the total 369 df for `Lines` is estimable from `Wrows#Cols[Blocks]`.
- In all 66.1 % (0.8207 * 297 / 369) of the `Lines` information is estimable here.
- A lot of `Lines` information is confounded with the variation from other field and milling phase sources of variation.

Substituting a linear Locations term

```
> #'## Look at the effect of substituting a linear Columns term for the Column variation term
> ph2sys.lin.canon <- designAnatomy(formulae   = list(lab   = ~ Int:Days + xLocn +
+                                                                  Int:Days:Locn,
+                                                           plot = ~ (Rows*Cols)/Samp,
+                                                           trt  = ~ Lines),
+                                              keep.order = TRUE, data = layout)
> print(summary(ph2sys.lin.canon, which.criteria = c("ae", "me", "ee", "dfor")))
```

Linear term for **Locations**.

Pool to simplify the analysis.

```
Summary table of the decomposition for lab, plot & trt (based on adjusted quantities)

 Source.lab       df1 Source.plot     df2 Source.trt df3 aefficiency mefficiency eefficiency dforthg
 Int:Days          15 Rows             15 Lines       15      0.7852      0.8214      0.4469       0
 xLocn              1 Rows              1 Lines        1      0.8095      0.8095      0.8095       0
 (Int:Days)#Locn 431 Rows              59 Lines       59      0.7021      0.8452      0.0802      13
                      Cols             11 Lines       11      0.8257      0.8659      0.5129       0
                      Rows#Cols       333 Lines      333      0.3245      0.8903      0.0066     283
                      Samp[Rows:Cols]  28                     1.0000      1.0000      1.0000      28
```

- Just 333 of the total 638 df for **Rows#Cols** and of the total 369 df for **Lines** is estimable here.
- Now 80.3% (0.8903 * 333 / 369) of the **Lines** information is estimable here (cf. 66.1% & 0.8207 with R#C included).
- Also 28 of the 41 df for **Samples[Rows:Cols]** (Error df) is available.

Summary

- Here, dividing the factors based on allocation of factors results in three sets of factors: only ever allocated; allocated and recipient; and only ever recipient.
- For a two-phase experiment there are four species of design: first-phase; second-phase; cross-phase; two-phase.
- The same methods of design selection apply, but need to consider three designs and how they combine.
- Again, **`designRandomize`** can be used to randomize the experiment and **`designAnatomy`** can be used to check the properties of the design, irrespective of the nonorthogonality and the number of tiers e.g. *p*/*q*-rep designs.
  - can be slow when the number of observations is large (several hundreds).

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). Multiphase experiments with at least one later laboratory phase. II. Nonorthogonal designs. *Australian & New Zealand Journal of Statistics,* **61**, 234-268.
- Brien, C. J., & Bailey, R. A. (2006). Multiple randomizations (with discussion). *Journal of the Royal Statistical Society, Series B (Statistical Methodology),* **68**, 571-609.
- Brien, C. J., Harch, B. D., Correll, R. L., & Bailey, R. A. (2011). Multiphase experiments with at least one later laboratory phase. I. Orthogonal designs. *Journal of Agricultural, Biological, and Environmental Statistics,* **16**, 422-450.
- Cochran, W. G., & Cox, G. M. (1957). *Experimental Designs* (2nd ed.). New York: Wiley.
- Cullis, B. R., Smith, A. B. & Coombes, N. E. (2006) On the design of early generation variety trials with correlated data. *Journal of Agricultural, Biological and Environmental Statistics*, **11**, 381-393.
- Federer, W. T. (1956). Augmented (or hoonuiaku) designs. *Hawaiian Planters' Record*, **55**, 191-208.
- Hinkelmann, K., & Kempthorne, O. (2005). *Design and analysis of experiments Vol. 2 Advanced experimental design*. Hoboken, N.J.: Wiley-Interscience.