designit: a flexible engine to generate experiment layouts

Juliane Siebourg-Polster, Iakov Davydov, Guido Steiner, Balazs Banfai

Roche Pharma Research and Early Development
Pharmaceutical Sciences
Roche Innovation Center Basel

21.10.2022

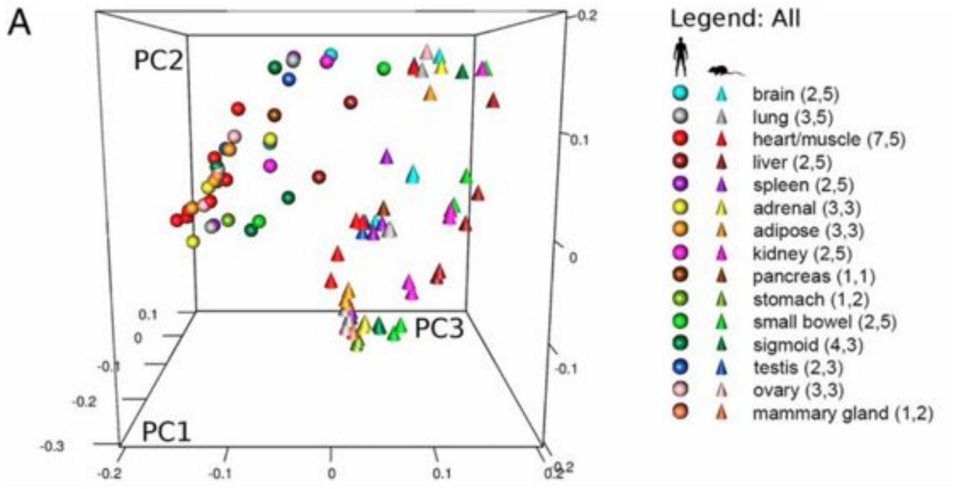
Introduction

Batch effects matter

An ENCODE paper (2014) shows clustering of RNA-Seq samples by **species, not by organs**.

Suspected reason: batch effect.

Differential expression analysis (e.g., comparing mouse and human) will be strongly affected by the batch-effect driving genes.

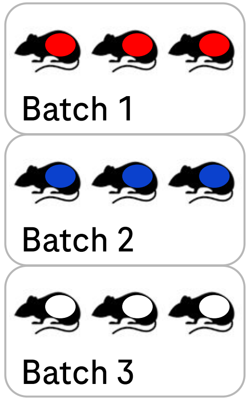


Comparison of human and mouse transcriptomes
Lin et al. PNAS Dec 2014, 111 (48) 17224-17229; DOI: 10.1073/pnas.1413624111

Introduction

Goal

Observation = Biological effect + Systematic error (Including batch effects) + Random error



Worst case scenario: Difference between the **groups** (color) is **fully confounded** by the **batch**.

Given a **limited set of** heterogeneous **samples**, we want to achieve **precise group effect estimates** and **avoid confounding**

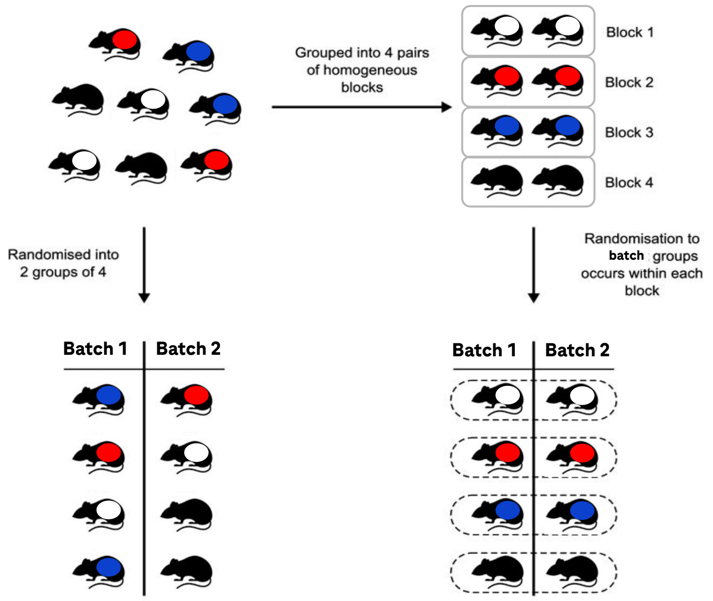
Solution: distributing groups of interest evenly across technical batches or sequences.

Introduction
Blocking & Randomization

- Randomization: Not sufficient
  - Too limited sample sizes to avoid grouping by chance
  - Experimental constraints might not allow for fully random layouts

- Blocking: arranging experimental units in groups that are similar to one another to remove biological or technical effects of no primary interest

"**Block** what you can and **randomize** what you cannot." (G. Box, 1978)


Grouped into 4 pairs of homogeneous blocks
Block 1
Block 2
Block 3
Block 4
Randomised into 2 groups of 4
Randomisation to batch groups occurs within each block
Batch 1
Batch 2
Batch 1
Batch 2

Adapted from Lazic 2016

designit

The package at a glance

bedapub.github.io/designit/

To **avoid batch or gradient effects confounding** in complex experiments, designit is an R package that offers flexible ways to **allocate a given set of samples to experiment layouts**.

It's strength is that it implements a very **general framework** that can easily be customized and extended to fit specific constrained layouts.

Data structure: `BatchContainer` class

- R6 object storing:
  - Experiment dimensions (batches, plates, cages)
  - Sample annotation
  - Scoring functions for sample distribution

Main function: `optimize_design()`

- Optimizes the layout with user defined
  - Scores for sample distribution
  - Optimization protocols
  - Sample shuffling functions
- Returns improved design and optimization trace

Sample batching

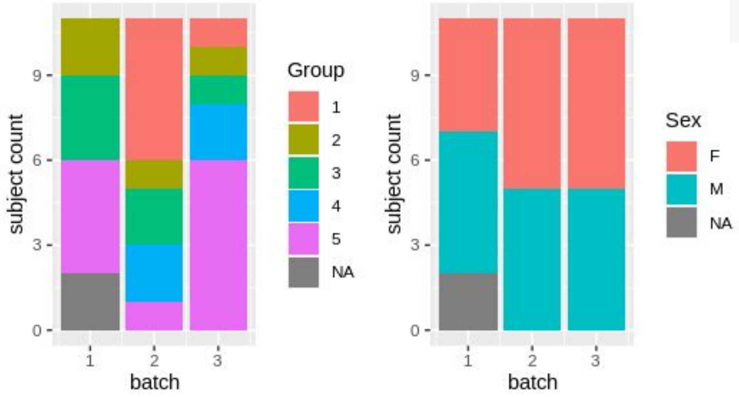
Setup

- Assign 31 samples to 3 equally sized batches
- Balance by:
  - treatment group (higher priority)
  - sex (lower priority)

```r
bc <- BatchContainer$new(
  dimensions = list("batch" = 3, "location" = 11),
)
bc$scoring_f <- list(
  group = osat_score_generator(batch_vars = "batch",
                               feature_vars = "Group"),
  sex = osat_score_generator(batch_vars = "batch",
                             feature_vars = "Sex")
)
assign_random(bc, subject_data)
```

Order matters!

Batch composition before optimization



```r
bc$get_samples()
```

batch	location	SubjectID	Group	Sex
1	1	NA	NA	NA
1	2	P32	5	M
1	3	P10	3	F
...	...	...	...	...
3	9	P31	3	F
3	10	P33	5	M
3	11	P24	5	F

Sample batching

Optimization

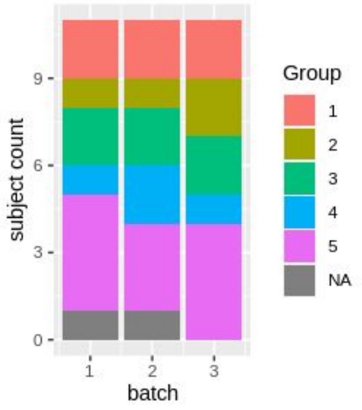
- Assign 31 samples to 3 equally sized batches
- Balance by:
  - treatment group (higher priority)
  - sex (lower priority)

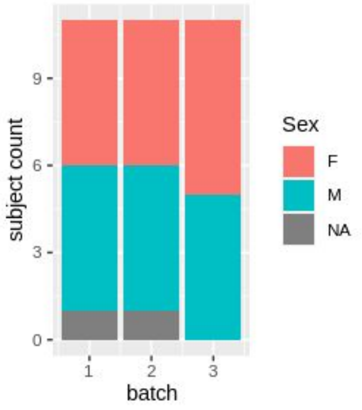
```
trace <- optimize_design(
  bc,
  n_shuffle = 1,
  acceptance_func =
    ~ accept_leftmost_improvement(..., tolerance = 0.01),
  max_iter = 150,
  quiet = TRUE
)
```

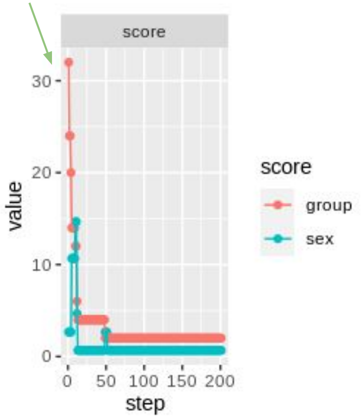
Prioritize scoring variables in given order

```
trace$plot()
```

Batch composition after optimization







batch	location	SubjectID	Group	Sex
1	1	NA	NA	NA
1	2	P01	1	F
1	3	P10	3	F
...	...	...	...	...
3	9	P29	5	F
3	10	P33	5	M
3	11	P12	3	F

Plate layouts

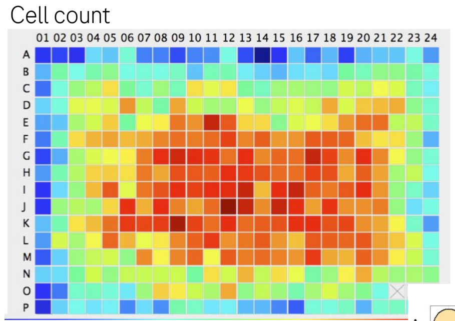
Spatial confounding

Assays are often performed in well plates (24, 96, 384)

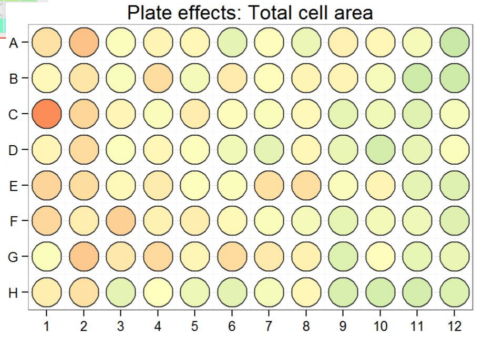
Observed effects

- Edge effects (bad plate sealing)
- Gradients (non-equal temperature distribution)
- Row / column effects (pipetting issues)

Since plate effects often cannot be avoided, we aim to distribute sample groups of interest evenly across the plate and adjust for the effect computationally.



Vebjorn Ljosa https://biology.stackexchange.com/q/13



Scott Warchal https://rpubs.com/Swarchal/phenoScreen

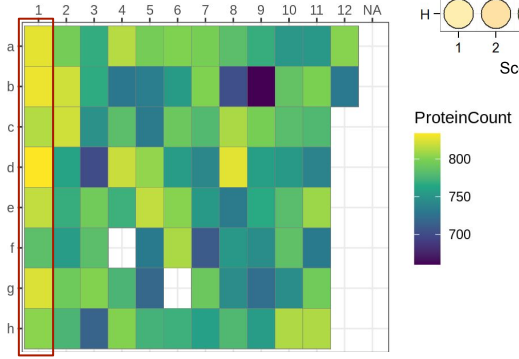


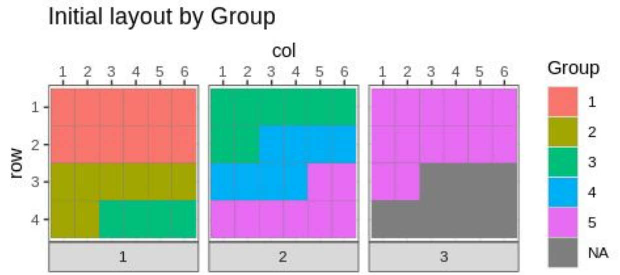
Plate layouts

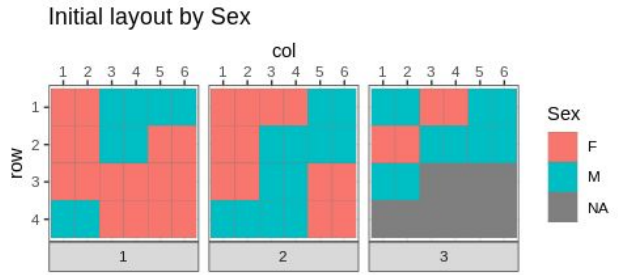
Setup

- Assume previous batches are 24-well plates
- Within plate optimization & across plate blocking
- Balanced by:
  - treatment group (higher priority)
  - sex (lower priority)

```r
bc <- BatchContainer$new(
  dimensions = list("plate" = 3, "row" = 4, "col" = 6),
)
assign_in_order(bc, dat)
```

```r
plot_plate(bc, plate = plate, row = row, column = col,
           .color = Group, title = "Initial layout by Group")
plot_plate(bc, plate = plate, row = row, column = col,
           .color = Sex, title = "Initial layout by Sex")
```





2-step optimization

- Across plates optimization using osat score as before
- **Within plate** optimization using distance based sample scoring function

Plate layouts

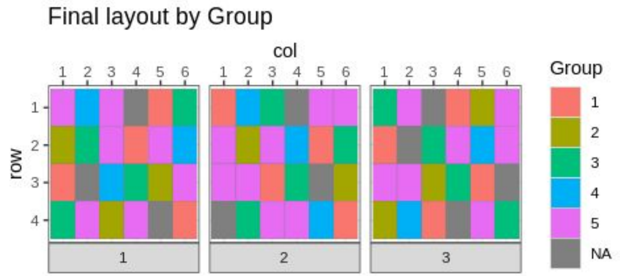
Spatial arrangement

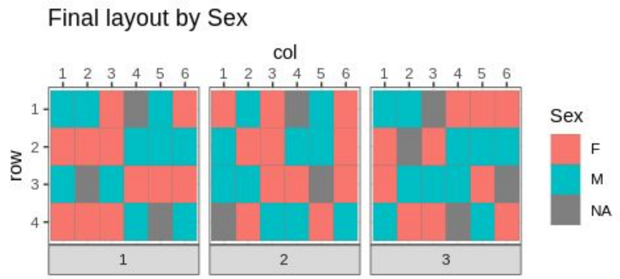
- Assume previous batches are 24-well plates
- Within plate optimization & across plate blocking
- Balanced by:
  - treatment group (higher priority)
  - sex (lower priority)

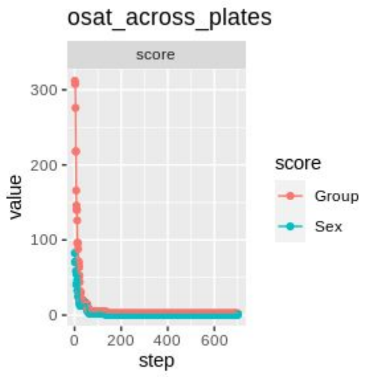
```
traces <- optimize_multi_plate_design(
  bc,
  across_plates_variables = c("Group", "Sex"),
  within_plate_variables = c("Group"),
  plate = "plate", row = "row", column = "col",
  n_shuffle = 2,
  max_iter = 2000
)
```

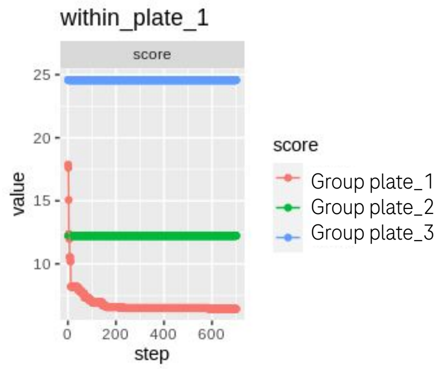
Step 1

Step 2, independent for each plate









Glimpse on complex application

Full in-vivo study layout

Goal:

- Assign 3 treatment conditions to 59 animals, representing 2 relevant strains
- Distribute animals to cages
- Avoid confounding by sex, weight and age

Constraints:

- Cages host ideally 3 animals (preferably 2-5)
- Strain, Sex and Treatment must be homogeneous within a cage
- Don’t put males from different litters same cage; litter mixing is possible for females!
- Average weight and age composition comparable between treatment groups and cages
- Avoid animals with identical ear markings in same cage (if

Animal list

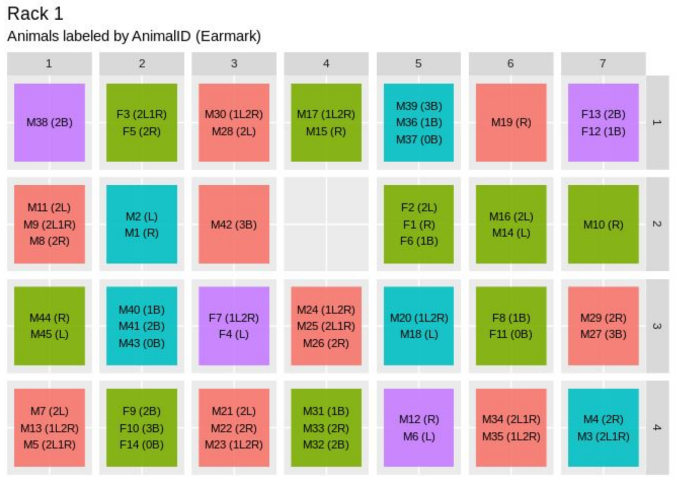
Strain	Sex	BirthDate	n
Strain A	F	NA	7
Strain A	M	NA	22
Strain B	F	2021-03-01	4
Strain B	F	2021-04-12	2
Strain B	F	2021-05-24	1

...

Treatment list

Treatment	Strain	Sex	n
Treatment 1	Strain A	M	10
Treatment 1	Strain B	M	10
Treatment 2	Strain A	F	5
Treatment 2	Strain A	M	5

...



Glimpse on more complex application

3 successive design steps

Assign treatments to animals → Allocate animals to cages → Arrange cages in racks

Given animal annotation and treatments, optimize assignment **balancing e.g. strain, sex, weight, age, litter across treatments** using fitting categorical / numerical scoring functions.

With treatment assignment, and given cage size range, optimize cage allocation wrt. variables

- **uniform** within cages (e.g. **treatment, strain, sex, litter**),
- **balanced** across cages (e.g. **age, body weight**)

With allocated cages, distribute cages in rack(s) **balancing** distribution of e.g **treatment, strain and sex across rack** rows and columns.

Conclusion

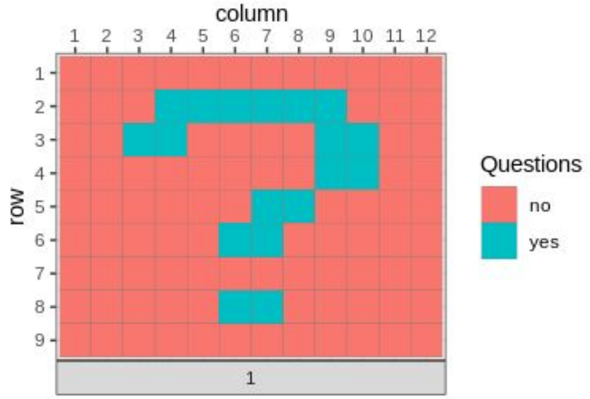
- `designit` aims to be general and adaptable
  - One framework to address simple batching as well as complex multi-step procedures
  - Extendable: custom scoring-functions, acceptance-criteria and shuffling-procedures can be passed to `optimize_design` by the user
- Includes functions and vignettes for frequently used layouts such as plates.

bedapub.github.io/designit/

Acknowledgements

- Martha Serrano
- Sabine Wilson
- David Jitao Zhang
- Konrad Rudolph
- Solveig Badillo
- Fabian Birzele

Thank you



Doing now what patients need next