RANDOMISATION

FOUNDATIONS AND R REFRESHER

Gavin Simpson
gavin@anivet.au.dk
Aarhus University

Wednesday, 21 October 2026

TODAY

Time	Component
12:30–12:40	Retrieval questions
12:40–13:15	Randomisation teaching
13:15–13:40	R refresher and demonstration
13:40–13:55	Break
13:55–16:00	Workshop, second break, debrief and exit

RETRIEVE

For a feed trial in which every pig in a pen receives the same diet:

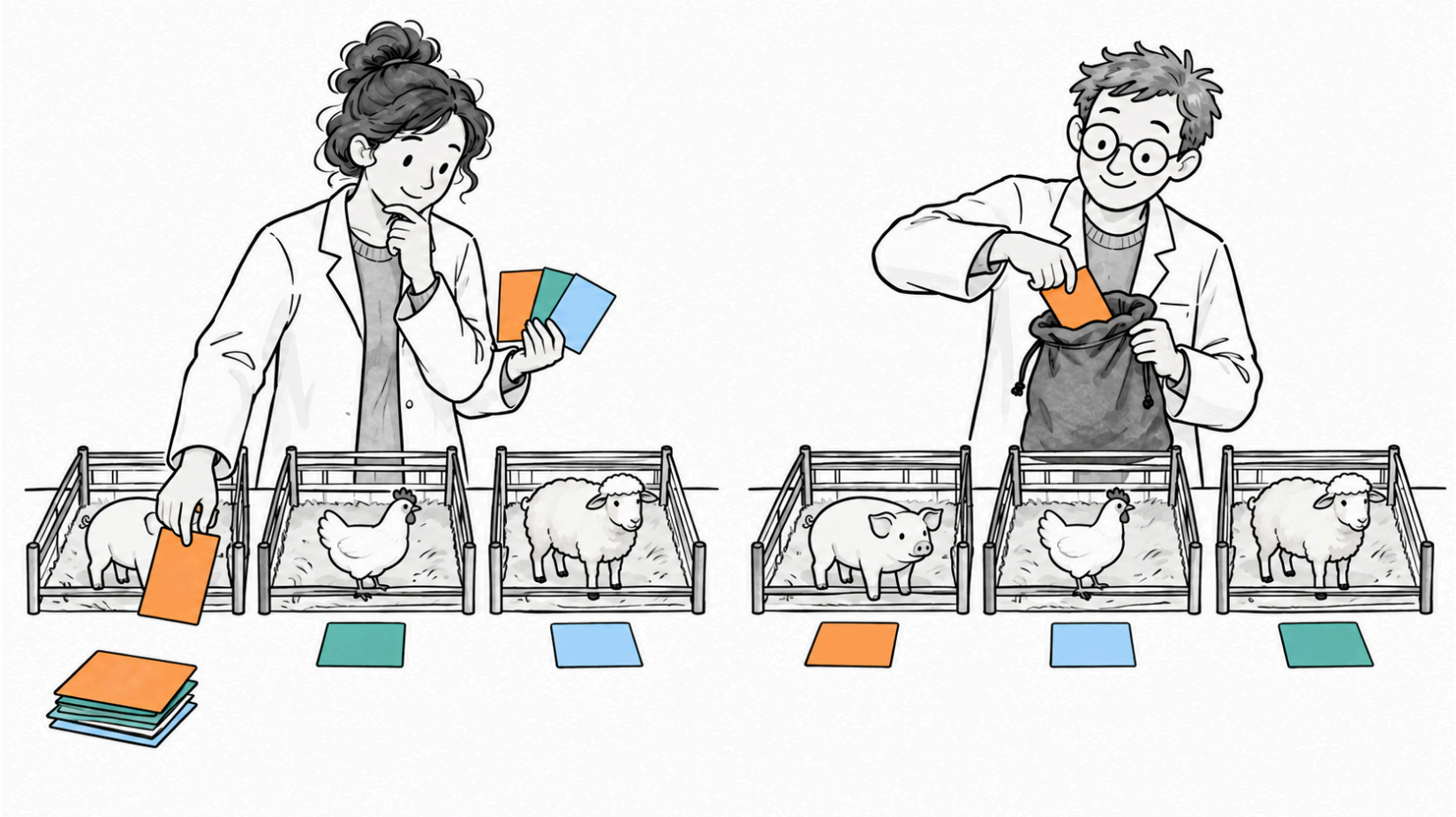
1. What is the experimental unit?
2. What is a treatment level?
3. What counts as replication of a diet?
4. When would diet be confounded with room?

TWO MEANINGS OF RANDOM

Random sampling	Random allocation
Selects units from a population	Assigns available units to treatments
Supports generalisation to the sampled population	Supports a fair treatment comparison
Concerns who or what enters the study	Concerns which treatment each unit receives

> Today we randomise treatment allocation. We do not claim the 24 pens are a random sample of all pig pens.

WHAT IS RANDOMISED?



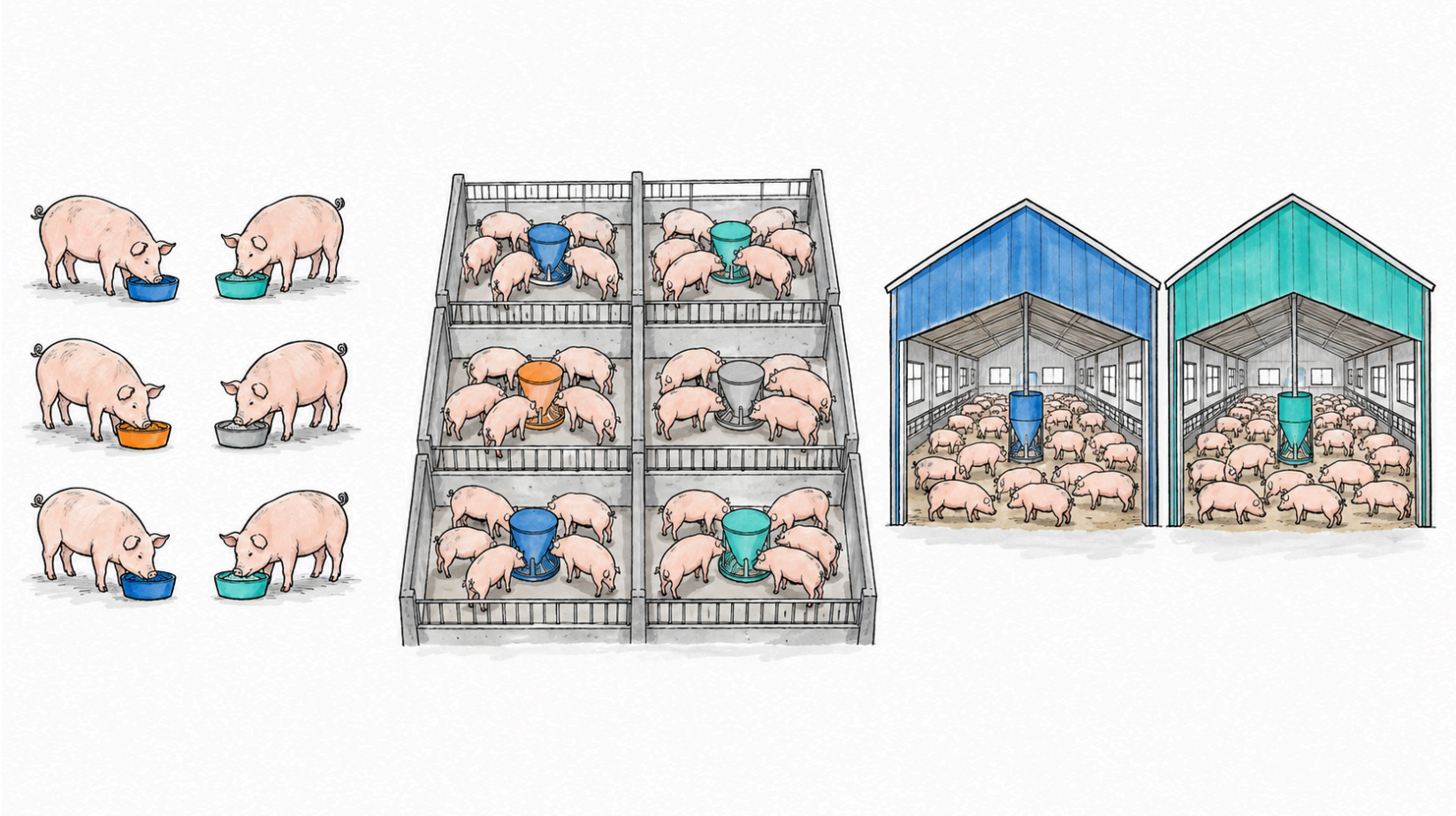
THE RANDOMISATION UNIT

The **experimental unit** is the smallest unit that can receive a treatment independently of other units.

- A diet delivered to a whole pen randomises **pens**
- Measuring six pigs in a pen does not create six allocations
- Recording one row per pen keeps the design visible

> If individual pigs cannot receive different diets independently, pigs must not be shuffled as though they could.

SAME QUESTION, DIFFERENT UNITS



A COMPLETE DESIGN

A **completely randomised single-factor design** has:

- one experimental factor
- two or more treatment levels
- independent experimental units
- every permitted allocation determined by a random process

In our case, the factor is **diet**, with four levels assigned to 24 pens.

BALANCE IS DIFFERENT

Balance means equal replication of treatment levels.

Randomisation means chance determines which unit receives which label.

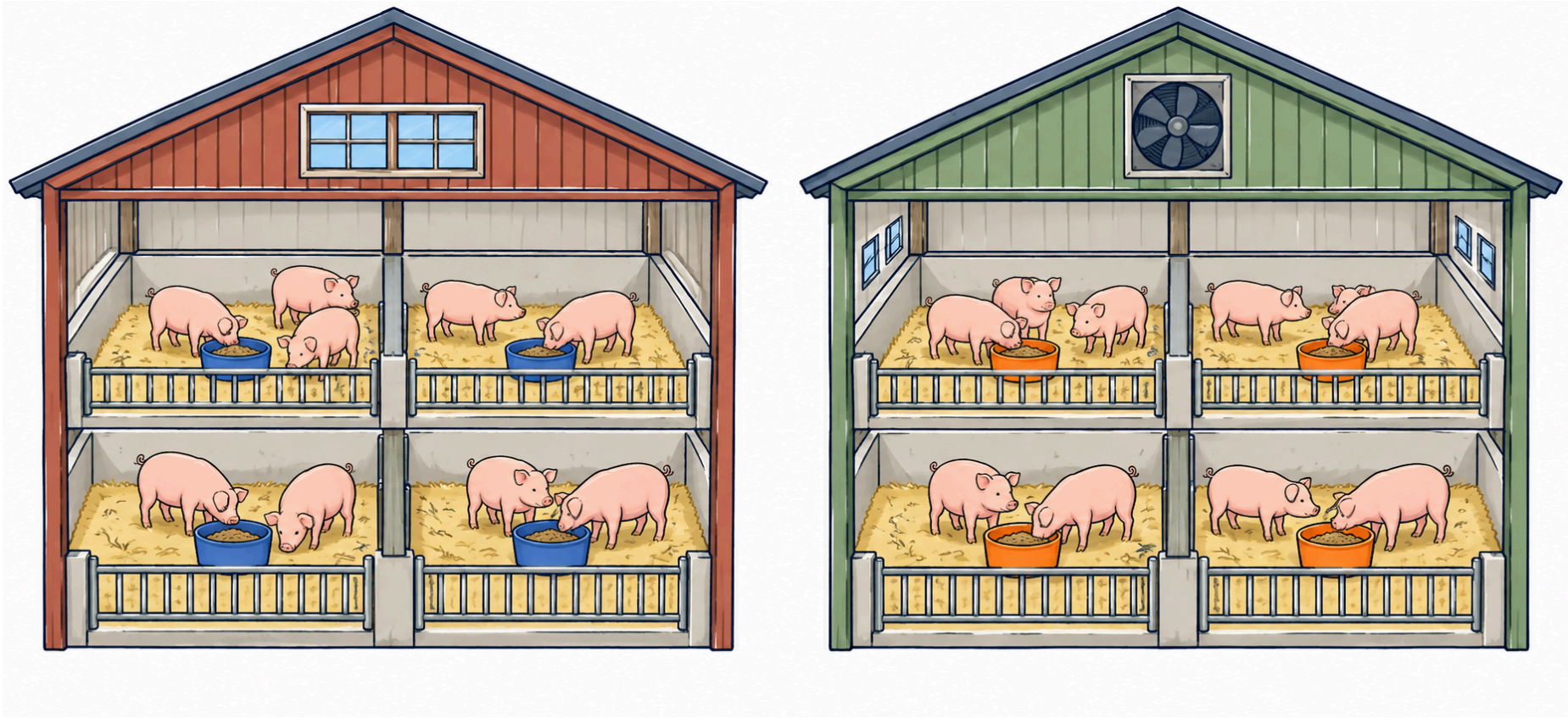
Allocation	Balanced?	Randomised?
Six labels per diet, shuffled	Yes	Yes
Control, A, B, C repeated	Yes	No
24 independent draws with replacement	Not guaranteed	Yes

NOT RANDOM ALLOCATION

- **Alternation:** first unit Control, next unit Diet A
- **Baseline choice:** give the lightest animals the experimental diet
- **Repeating pattern:** Control, A, B, C down every room
- **Convenience:** use whichever treatment is easiest to deliver next

These rules are predictable or depend on a unit characteristic. None uses a random process.

CONFOUNDED BY CONSTRUCTION



WHY RANDOMISE?

Random allocation:

- removes deliberate or unconscious choice from assignment
- gives uncontrolled unit differences a chance to be spread among treatments
- provides the basis for probability-based analysis
- makes the allocation procedure auditable when it is recorded

> Randomisation protects the allocation process; it does not certify that the realised groups are identical.

WHAT IT CANNOT PROMISE

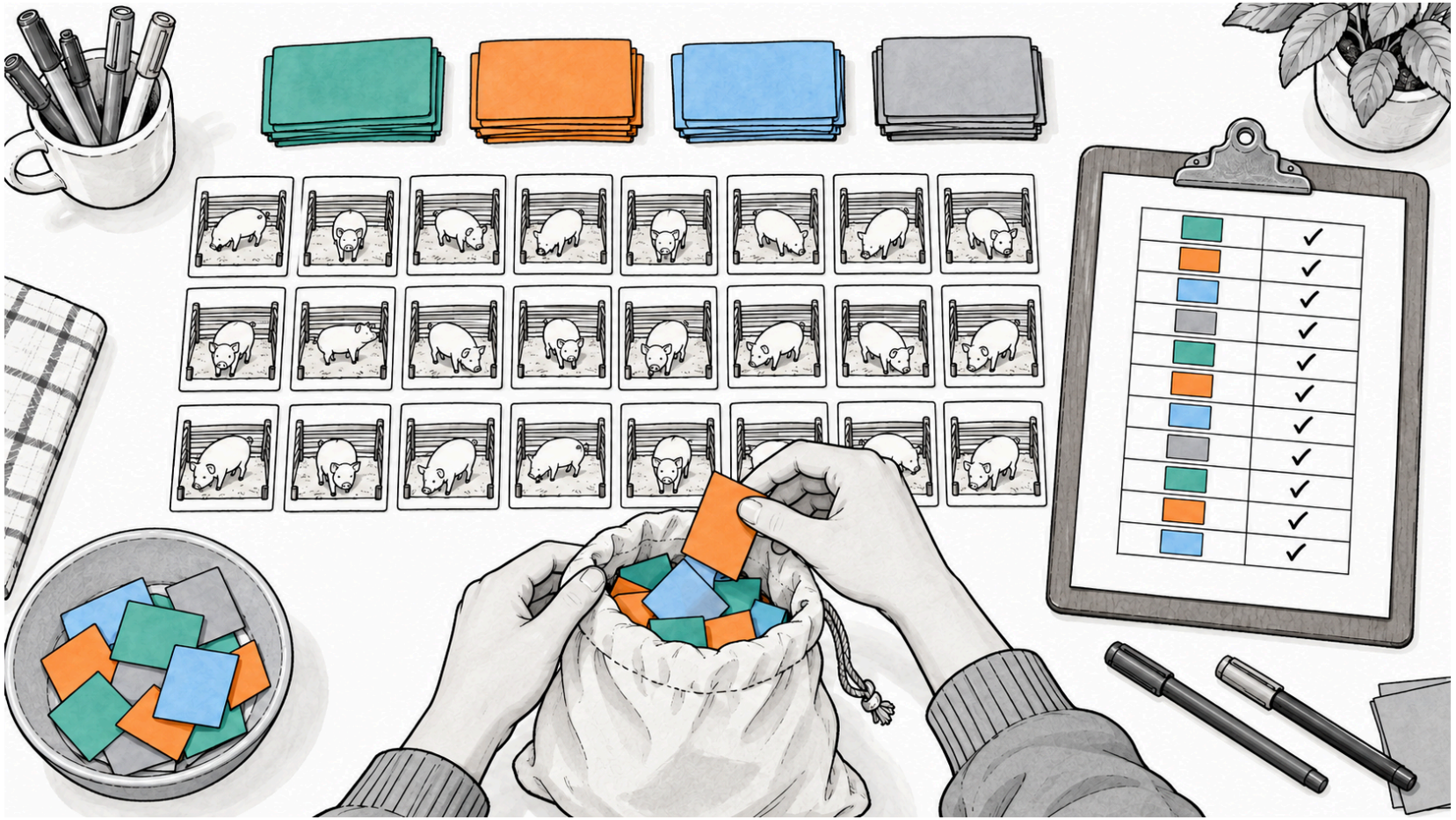
One random allocation may still produce:

- more heavy animals in one treatment
- more of one treatment in a particular room
- a visually surprising run of neighbouring assignments

It also cannot repair:

- too few independent experimental units
- poor measurement
- treatment contamination or interference
- biased selection into the study

KEEP THE RECORD



THE UNIT TABLE

```
library("tibble")
library("dplyr")
library("ggplot2")

pens <- tibble(
  pen_id = sprintf("P%02d", 1:24),
  room = rep(c("North", "Central", "South"), each = 8),
  position = rep(1:8, times = 3)
)

pens
```

TREATMENT LABELS

```
treatments <- rep(
  c("Control", "Diet A", "Diet B", "Diet C"),
  each = 6
)

tibble(diet = treatments) |>
  count(diet)
```

```
# A tibble: 4 × 2
  diet        n
  <chr>   <int>
1 Control     6
2 Diet A      6
3 Diet B      6
4 Diet C      6
```

The vector contains exactly six copies of each treatment label.

DRAW ONCE

```
set.seed(123)

allocation <- pens |>
  mutate(diet = sample(treatments))

allocation
```

- `set.seed()` fixes the random-number stream
- `sample()` permutes the 24 labels
- `mutate()` attaches one label to each pen

CHECK THE TOTAL

```
allocation |>
  count(diet)
```

```
# A tibble: 4 × 2
  diet        n
  <chr>   <int>
1 Control     6
2 Diet A      6
3 Diet B      6
4 Diet C      6
```

Checking is part of producing the allocation—not an optional extra.

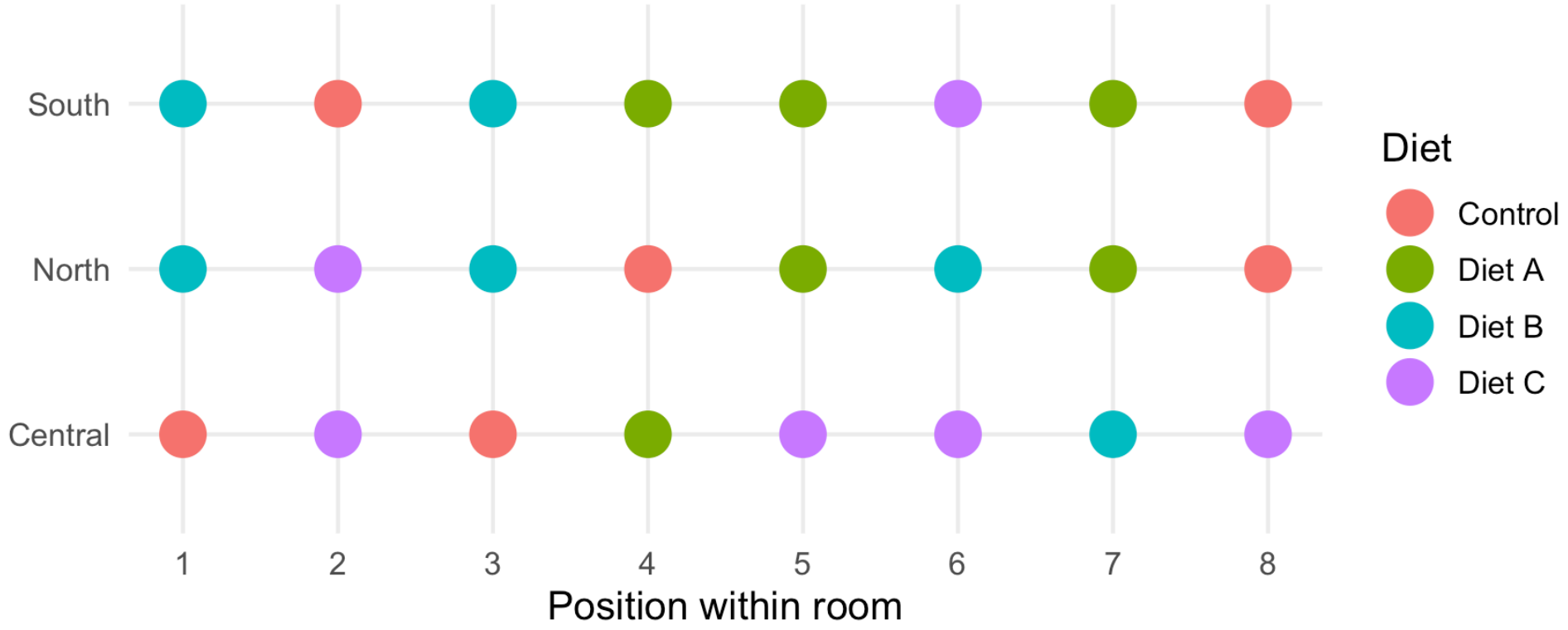
CHECK THE ROOMS

```
allocation |>
  count(room, diet)
```

```
# A tibble: 12 × 3
   room    diet        n
   <chr>   <chr>   <int>
 1 Central Control     2
 2 Central Diet A      1
 3 Central Diet B      1
 4 Central Diet C      4
 5 North   Control     2
 6 North   Diet A      2
 7 North   Diet B      3
 8 North   Diet C      1
 9 South   Control     2
10 South   Diet A      3
11 South   Diet B      2
12 South   Diet C      1
```

> What does overall balance conceal?

SEE THE LAYOUT



GROUPED OPERATIONS

```
allocation |>
  group_by(room) |>
  count(diet) |>
  ungroup()
```

- `group_by()` makes later operations act within each room
- `ungroup()` removes that instruction when it is no longer needed

BEFORE THE BREAK

You should now be able to:

- identify the unit being randomised
- distinguish random allocation from random sampling
- distinguish balance from randomisation
- generate and audit an allocation in R
- explain why a valid randomisation can look uneven

SOURCES

Ruxton, Graeme D., and Nick Colegrave. 2016. *Experimental Design for the Life Sciences*. 4th ed. Oxford: Oxford University Press. Chapter 7, sections 7.1–7.4, pp. 101–108. ISBN 978-0-19-871735-5. Publisher page.

R Core Team. 2026. *R: A Language and Environment for Statistical Computing*. Vienna: R Foundation for Statistical Computing. R project.

Wickham, Hadley, et al. 2019. "Welcome to the tidyverse." *Journal of Open Source Software* 4 (43): 1686. doi:10.21105/joss.01686.