16 Experimental Design

R. Heller · Date: 2026-05-11 · Last changed: 2026-10-30

Study design across the spectrum: observational designs and STROBE, randomised trials, missing-data mechanisms and multiple imputation, mixed-effects models, DAGs and confounding, propensity scores, the g-methods family (IV, DiD, RDD, HTE), compartmental models for outbreaks, and the pre-registration / SAP / replication-crisis literature.

This chapter contains **30 method pages** and **13 labs**. If you are not sure which method to read, return to Chapter 0 and follow the decision tree to the right node.

16.1 Method pages

Method	Source slug
2^k Factorial Designs	`factorial-2k`
3^k Factorial Designs	`factorial-3k`
Balanced Incomplete Block Designs	`balanced-incomplete-block`
Box-Behnken Designs	`box-behnken-design`
Central Composite Designs	`central-composite-design`
Completely Randomised Design	`completely-randomized-design`
Constrained Mixture Designs	`constrained-mixture`
Crossover Designs	`crossover-designs`
Desirability Functions	`desirability-function`
Fractional Factorial Designs	`fractional-factorial-design`
Graeco-Latin Square Designs	`graeco-latin-square`
Latin Square Designs	`latin-square`
Method of Steepest Ascent	`steepest-ascent`
Optimal Designs: D, A, I	`optimal-design-d-a-i`
Orthogonal Arrays	`orthogonal-arrays`
Plackett-Burman Designs	`plackett-burman`
Power Analysis for DOE	`power-in-doe`
Principles of Blocking	`blocking-principles`
Randomisation in Design	`randomization-in-design`
Randomised Complete Block Design	`randomised-complete-block`
Repeated-Measures Designs	`repeated-measures-design`
Resolution and Aliasing	`resolution-aliasing`
Response Surface Methodology	`response-surface-methodology`
Robust Parameter Design	`robust-parameter-design`

Method	Source slug
Signal-to-Noise Ratios	`signal-to-noise-ratio`
Simplex Centroid Designs	`mixture-centroid`
Simplex Lattice Mixture Designs	`mixture-designs-simplex`
Split-Plot Designs	`split-plot-design`
Strip-Plot Designs	`strip-plot-design`
Taguchi Methods	`taguchi-methods`

16.2 Labs

Lab
Observational designs and STROBE
Randomised controlled trials
Adaptive, non-inferiority, equivalence trials
Bench and translational design
MCAR, MAR, MNAR
Multiple imputation with mice
Linear mixed models with lme4
GLMMs and GEE
DAGs with dagitty and ggdag
Propensity scores and IPTW
G-methods, IV, DiD, RDD; heterogeneity of treatment effect
SIR and SEIR models with deSolve
Pre-registration and statistical analysis plans

16.3 Balanced Incomplete Block Designs

16.3.1 Introduction

Balanced incomplete block designs (BIBDs) allocate treatments to blocks when each block can hold fewer experimental units than the total number of treatments to be compared. The constraint arises constantly in practice — a litter of animals contains fewer pups than treatments, a panel of taste-testers has limited capacity per session, a microscopy slide accommodates only a few specimens, an oven processes a limited number of samples per run. The “balanced” property requires that every pair of treatments appears together in the same number λ of blocks, ensuring that every pairwise treatment contrast is estimated

with equal precision despite the block-level incompleteness. BIBDs are classical constructions in agricultural experimentation, sensory evaluation, and any setting where complete randomised blocks are physically impossible.

16.3.2 Prerequisites

A working understanding of randomised complete block designs, combinatorial design parameters, and basic mixed-effects modelling for the recovery of inter-block information.

16.3.3 Theory

A BIBD is specified by parameters (v, b, r, k, λ):

- v = number of treatments,
- b = number of blocks,
- r = replicates per treatment,
- k = block size (with $k < v$),
- λ = number of blocks in which each treatment pair co-occurs.

These parameters satisfy two necessary identities: $bk = vr$ and $\lambda(v-1) = r(k-1)$. Not every parameter combination corresponds to an existing BIBD; Fisher's inequality requires $b \geq v$, and the Bruck–Ryser–Chowla theorem provides further necessary conditions. The classical analysis fits blocks as either fixed or random effects; treating blocks as random recovers inter-block information and is generally more efficient.

16.3.4 Assumptions

Treatment pairs are exchangeable via the balance property, within-block residuals are exchangeable and approximately Normal with constant variance, and block effects are additive (no block × treatment interaction). The design is implemented as planned — running fewer or more units than specified destroys the balance and requires reanalysis as a partially balanced or unbalanced incomplete block design.

16.3.5 R Implementation

```
library(crossdes)

bibd ← find.BIB(7, 7, 3)
bibd

set.seed(2026)
df ← data.frame(
  block = rep(1:7, each = 3),
  treatment = factor(as.vector(t(bibd)))
)
df$y ← rnorm(nrow(df), mean = c(10, 11, 12, 13, 14, 15, 16)
[as.integer(df$treatment)],
              sd = 0.8)

fit ← aov(y ~ factor(block) + treatment, data = df)
summary(fit)
```

16.3.6 Output & Results

`find.BIB()` constructs a BIBD with the specified parameters when one exists. The fitted ANOVA partitions variability into block, treatment, and residual components, with the treatment F-test computed using the intra-block error. Reporting the proportion of variance absorbed by blocks alongside the treatment effect characterises the gain from incomplete blocking.

16.3.7 Interpretation

A reporting sentence: "The $(v, b, r, k, \lambda) = (7, 7, 3, 3, 1)$ BIBD — equivalent to the Fano plane in projective geometry — tested seven treatments in seven blocks of size three. ANOVA showed a significant treatment effect ($F_{6,8} = 8.2$, $p = 0.002$) with block-level variance absorbed by the block factor (block sum of squares 12.4, residual sum of squares 4.7). All 21 pairwise treatment contrasts were estimated with equal precision because of the balance property." Always report the BIBD parameters explicitly.

16.3.8 Practical Tips

- Use a BIBD whenever block size is smaller than the treatment count and balanced pairwise comparisons are required; the balance property ensures every contrast has the same standard error, which simplifies interpretation and multiple-comparison adjustment.
- Analyse with blocks as **random effects** (`(1 | block)` in `lme4`) to recover inter-block information; the random-effects analysis is generally more efficient than the fixed-effect intra-block analysis, especially when the number of blocks is large.
- Not all parameter combinations (v, k, λ) correspond to an existing BIBD; consult catalogue tables or use `crossdes::find.BIB()` to check existence before finalising the design. Fisher's inequality and the Bruck–Ryser–Chowla theorem give the necessary conditions.
- For large treatment counts where a BIBD is impractical, partially balanced incomplete block designs (PBIBDs) relax the equal-pairwise-co-occurrence requirement at the cost of unequal pairwise precision; they are far easier to find and adequate for most applied work.
- Pre-specify the block allocation in the protocol; changing mid-experiment — adding a unit, dropping a block — destroys the balance property and forces reanalysis as a general unbalanced design.
- Visualise the incidence matrix (treatment × block) before running the experiment; an inadvertent imbalance (a treatment missing from a block, a duplicated treatment within a block) is far easier to spot at the design stage than after data collection.

16.3.9 R Packages Used

`crossdes::find.BIB()` and `crossdes::isGYD()` for BIBD construction and verification; `agricolae::design.bib()` for an alternative interface integrated with the agricultural-DoE toolkit; `lme4::lmer()` for random-block analysis recovering inter-block information; `ibd`

for advanced incomplete-block constructions including PBIBDs; `daewr` for textbook companion datasets.

16.3.10 For Reviewers

Use the shared reviewer checklist, together with this method's assumptions and practical tips.

Method-specific reviewer notes have not yet been developed for this entry. The links above support a structured review; they do not certify the example or a manuscript's application of the method.

16.3.11 See also — labs in this chapter

- Observational designs and STROBE
- Randomised controlled trials
- Adaptive, non-inferiority, equivalence trials
- Bench and translational design
- MCAR, MAR, MNAR
- Multiple imputation with mice
- Linear mixed models with lme4
- GLMMs and GEE
- DAGs with dagitty and ggdag
- Propensity scores and IPTW
- G-methods, IV, DiD, RDD; heterogeneity of treatment effect
- SIR and SEIR models with deSolve
- Pre-registration and statistical analysis plans

16.4 Principles of Blocking

16.4.1 Introduction

Blocking is the experimental-design technique that groups experimental units by a known source of nuisance variation — batch of raw material, day of measurement, operator, animal litter, plot location — and then randomises treatment assignments within each block. The result is a design that systematically removes block-to-block variation from the error term, thereby shrinking the residual variance and substantially improving the power to detect genuine treatment effects. Fisher's classical mantra — "block what you can, randomise what you cannot" — captures the principle: when a nuisance source of variation is known and identifiable, controlling it through design is far more efficient than relying on randomisation alone to average it out.

16.4.2 Prerequisites

A working understanding of randomisation as the basis of comparative inference, the partition of total variability into systematic and residual components in ANOVA, and the distinction between fixed and random effects in mixed-effects models.

16.4.3 Theory

In its simplest form — the randomised complete block design (RCBD) — the model is

$$y_{ij} = \mu + \rho_i + \tau_j + \varepsilon_{ij}, \qquad \varepsilon_{ij} \sim N(0, \sigma^2),$$

with block effect ρ_i and treatment effect τ_j. Within-block variance is typically much smaller than between-block variance, so absorbing the latter into the design removes a large component from the residual error. The relative efficiency of an RCBD versus a CRD is roughly the ratio of the residual variances; gains of an order of magnitude are not unusual when blocking is well chosen.

16.4.4 Assumptions

Blocks represent exchangeable groupings of units, no block-by-treatment interaction is present (or, if it is, the interaction is modelled — typically as a random effect), and within-block residual variance is homogeneous. Violations of additivity invalidate the standard analysis but can be detected with a Tukey one-degree-of-freedom test for non-additivity.

16.4.5 R Implementation

```
set.seed(2026)
block ← factor(rep(1:4, each = 3))
trt ← factor(rep(c("A", "B", "C"), 4))

block_re ← rnorm(4, 0, 2)[as.integer(block)]
trt_eff  ← c(0, 1.5, 2.5)[as.integer(trt)]
y ← 10 + block_re + trt_eff + rnorm(12, 0, 0.5)

df ← data.frame(block, trt, y)

fit_b ← aov(y ~ block + trt, data = df)
fit_nb ← aov(y ~ trt, data = df)

c(with_block_MSE = summary(fit_b)[[1]]["Residuals", "Mean Sq"],
  without_block_MSE = summary(fit_nb)[[1]]["Residuals", "Mean Sq"])
```

16.4.6 Output & Results

The script reports residual mean-square error with and without blocking. When the block-to-block standard deviation is comparable to or larger than the within-block standard deviation, blocking produces dramatic reductions in residual MSE and correspondingly large increases in the F-statistic for the treatment effect. The same point can be visualised by comparing residual diagnostic plots from the two fits.

16.4.7 Interpretation

A reporting sentence: “Blocking on production batch reduced residual mean-square error from 3.8 to 0.24, a 16-fold reduction. The treatment F-ratio increased from $F_{2,9} = 1.1$ ($p = 0.37$) without blocking to $F_{2,6} = 17.3$ ($p = 0.003$) with blocking, demonstrating that

batch was the dominant nuisance source and that blocking restored the detection power lost to batch-level drift." Always report residual MSE before and after blocking when justifying the design choice.

16.4.8 Practical Tips

- Block first on the largest known source of unwanted variation: temporal (day, week, batch), spatial (plot, shelf, freezer), or operator-related (technician, machine, lot). Unblocked sources contribute to the residual; blocked sources are absorbed into the design.
- One block per replicate is the standard RCBD; multiple blocking factors call for Latin square (two factors) or Graeco-Latin square (three factors) layouts that retain orthogonality.
- Incomplete blocks (block size smaller than the number of treatments) require balanced incomplete block designs (BIBD) or partially balanced incomplete blocks (PBIBD); the analysis is more involved but the principle is the same.
- Treat blocks as random effects (`(1|block)` in `lme4`) when blocks are sampled from a larger population of possible blocks, as is typical for batches, animals, or experimental days; this also recovers inter-block information that the fixed-effect analysis discards.
- Do not block on a variable you actually want to study: if the nuisance factor is itself of interest, treat it as a fixed treatment factor in a factorial design instead. Blocking removes inferential interest in the blocked factor.
- Incomplete or unbalanced block sizes are increasingly common in clinical trials (centre-stratified randomisation) and call for mixed-effects models rather than classical fixed-effect ANOVA.

16.4.9 R Packages Used

Base R `aov()` for fixed-effect RCBD analysis; `lme4` and `lmerTest` for random-block mixed-effects models with Satterthwaite or Kenward–Roger denominator df; `agricolae` for `design.rcbd()` , `BIB.test()` , and related agricultural design tools; `emmeans` for marginal means and contrasts under blocking; `crossdes` for systematic construction of incomplete-block layouts.

16.4.10 For Reviewers

Use the shared reviewer checklist, together with this method's assumptions and practical tips.

Method-specific reviewer notes have not yet been developed for this entry. The links above support a structured review; they do not certify the example or a manuscript's application of the method.

16.4.11 See also — labs in this chapter

- Observational designs and STROBE

- Randomised controlled trials
- Adaptive, non-inferiority, equivalence trials
- Bench and translational design
- MCAR, MAR, MNAR
- Multiple imputation with mice
- Linear mixed models with lme4
- GLMMs and GEE
- DAGs with dagitty and ggdag
- Propensity scores and IPTW
- G-methods, IV, DiD, RDD; heterogeneity of treatment effect
- SIR and SEIR models with deSolve
- Pre-registration and statistical analysis plans

16.5 Box-Behnken Designs

16.5.1 Introduction

Box-Behnken designs (BBD), introduced by Box and Behnken in 1960, are rotatable three-level response-surface designs whose distinguishing feature is that they never run the extreme-corner combinations of the factorial cube. Each factor is set at one of three coded levels (-1, 0, or $+1$), but every experimental run leaves at least one factor at the centre level — so no run combines all factors at their most extreme settings simultaneously. This makes the design particularly useful when extreme corner conditions are infeasible (safety limits, equipment constraints, prohibitive cost, or unstable chemistry at extremes), while still allowing estimation of the full second-order response-surface model with main effects, quadratic terms, and two-factor interactions.

16.5.2 Prerequisites

A working understanding of factorial designs, response-surface methodology (RSM), the second-order polynomial model, and the concept of design rotatability — that prediction variance depends only on distance from the design centre, not on direction.

16.5.3 Theory

For k factors a Box-Behnken design places each factor at one of $\{-1, 0, +1\}$, with runs taken only at combinations where two factors are at ± 1 and all remaining factors are at the centre level 0. Replicated centre runs provide pure-error estimation and a lack-of-fit test. For $k = 3$ a BBD requires 12 factorial-style runs plus typically 3 centre points (15 runs total), comparable in size to a central composite design (CCD) with 14–20 runs.

BBDs are exactly rotatable for $k = 4$ and $k = 7$; for other values of k they are nearly rotatable. The absence of axial (star) points means prediction near the cube's corners is extrapolation with elevated variance.

16.5.4 Assumptions

A second-order polynomial response surface is adequate over the design region (no strong curvature beyond quadratic), corner-extreme runs are either infeasible or unnecessary, and the residual error is approximately Normal with constant variance. The pure-error replications at the centre allow these assumptions to be diagnosed.

16.5.5 R Implementation

```
library(rsm)

bbd ← bbd(~ x1 + x2 + x3, n0 = 3, randomize = FALSE)
bbd

set.seed(2026)
bbd$y ← 10 + 2 * bbd$x1 + 1.5 * bbd$x2 - 1 * bbd$x3 -
         0.9 * bbd$x1^2 + 0.5 * bbd$x1 * bbd$x2 +
         rnorm(nrow(bbd), 0, 0.4)

fit ← rsm(y ~ SO(x1, x2, x3), data = bbd)
summary(fit)
```

16.5.6 Output & Results

`bbd()` constructs the layout; the `rsm()` second-order model returns coefficients for all linear, quadratic, and two-factor-interaction terms together with a lack-of-fit test against the pure-error replicates. The canonical analysis includes a stationary-point analysis and an eigenvalue decomposition that classifies the surface as a maximum, minimum, or saddle.

16.5.7 Interpretation

A reporting sentence: “The 15-run Box-Behnken design with 3 centre points estimated all linear, quadratic, and pairwise-interaction effects of the three factors. The fit revealed dominant curvature in x_1 (quadratic coefficient -0.9, $p < 0.001$) with significant linear effects on all three factors and one interaction ($x_1 \times x_2 = 0.5$, $p = 0.03$). The lack-of-fit test was non-significant ($p = 0.42$), supporting the second-order model. Predicted maximum lies at $x_1 = 1$, $x_2 = 1$, $x_3 = -1$.” Always report the lack-of-fit test outcome.

16.5.8 Practical Tips

- Prefer BBD over CCD when corner combinations are infeasible (extreme operating conditions, safety limits, equipment constraints, regulatory bounds, or unstable chemistry at extremes); BBD is the design of choice in such settings.
- BBD has no axial points and so does not extend beyond the cube; prediction near or beyond the cube corners is extrapolation with elevated variance, and the design region should be chosen with this in mind.
- Centre-point replications are essential — they provide pure-error estimation and the lack-of-fit test that diagnoses inadequacy of the second-order model. Three to five

replicated centre points is the typical compromise between cost and inferential value.
- BBD is generally less efficient than CCD in prediction variance over the cube; the choice between them is driven by feasibility of corner runs and the practical cost of extra axial-point experimentation.
- BBD is not defined for $k = 2$; for two-factor RSM, use a central composite design or a face-centred CCD.
- For mixed-precision designs (some factors continuous, others two-level discrete), `rsm` can fit hybrid models, but pre-specified screening followed by RSM follow-up is usually clearer.

16.5.9 R Packages Used

`rsm::bbd()` and `rsm::rsm()` for the canonical Box-Behnken construction and second-order analysis with stationary-point and canonical analysis; `DoE.base` for general orthogonal-array DoE construction; `daewr` for textbook companion examples; `qualityTools` for an alternative RSM toolkit including BBD; `Vdgraph` for variance-dispersion graphs comparing BBD to CCD.

16.5.10 For Reviewers

Use the shared reviewer checklist, together with this method's assumptions and practical tips.

Method-specific reviewer notes have not yet been developed for this entry. The links above support a structured review; they do not certify the example or a manuscript's application of the method.

16.5.11 See also — labs in this chapter

- Observational designs and STROBE
- Randomised controlled trials
- Adaptive, non-inferiority, equivalence trials
- Bench and translational design
- MCAR, MAR, MNAR
- Multiple imputation with mice
- Linear mixed models with lme4
- GLMMs and GEE
- DAGs with dagitty and ggdag
- Propensity scores and IPTW
- G-methods, IV, DiD, RDD; heterogeneity of treatment effect
- SIR and SEIR models with deSolve
- Pre-registration and statistical analysis plans

16.6 Central Composite Designs

16.6.1 Introduction

The central composite design (CCD), introduced by Box and Wilson in 1951, is the workhorse of response-surface methodology and arguably the most influential design construction in industrial statistics. It augments a 2^k factorial with axial (or star) points placed along each factor axis at distance α from the design centre and a number of replicated centre points. Together these three components — factorial, axial, centre — provide enough information to fit a full second-order polynomial response surface in k factors with substantially fewer runs than a 3^k full factorial would require, while still supporting estimation of curvature, lack-of-fit testing, and prediction-variance characterisation across the design region.

16.6.2 Prerequisites

A working understanding of two-level factorial designs, polynomial regression for response-surface modelling, and the concept of design rotatability — that prediction variance depends only on radial distance from the centre.

16.6.3 Theory

A CCD has three components:

- **Factorial**: 2^k runs at the corners of the unit cube (± 1 in each factor).
- **Axial**: $2k$ runs at $(\pm\alpha, 0, \ldots, 0), (0, \pm\alpha, 0, \ldots, 0), \ldots, (0, \ldots, 0, \pm\alpha)$.
- **Centre**: replicated $(0, 0, \ldots, 0)$ runs supplying pure-error estimation.

The choice of α determines the design's properties: rotatable CCDs use $\alpha = \sqrt[4]{2^k}$ (giving constant prediction variance at fixed distance from the centre), face-centred CCDs use $\alpha = 1$ (staying within the original ± 1 cube, useful when factor extremes are constrained), and orthogonal CCDs choose α to make the columns of the second-order design matrix orthogonal.

16.6.4 Assumptions

The response is well modelled by a second-order polynomial within the design region, the residual error is approximately Normal with constant variance, and the centre-point replicates provide a valid pure-error estimate against which lack-of-fit can be tested.

16.6.5 R Implementation

```
library(rsm)

ccd ← ccd(~ x1 + x2, n0 = c(3, 3), alpha = "rotatable",
          randomize = FALSE)
ccd

set.seed(2026)
ccd$y ← 10 + 2 * ccd$x1 + 1.5 * ccd$x2 - 1.2 * ccd$x1^2 -
        0.8 * ccd$x2^2 + 0.6 * ccd$x1 * ccd$x2 +
        rnorm(nrow(ccd), 0, 0.5)

fit ← rsm(y ~ SO(x1, x2), data = ccd)
summary(fit)

steepest(fit, dist = 1.0)
```

16.6.6 Output & Results

`ccd()` constructs the layout, and `rsm()` fits the second-order model with stationary-point analysis, eigenvalue decomposition, and a lack-of-fit test against the pure-error replicates. The `steepest()` function traces the path of steepest ascent, useful when the stationary point lies outside the design region and follow-up experimentation is needed in the optimal direction.

16.6.7 Interpretation

A reporting sentence: "The 14-run rotatable CCD with three centre points identified a concave response surface with stationary point at $(x_1, x_2) = (0.83, 0.69)$, predicted optimum 12.3. Quadratic terms in both factors were significant ($p < 0.01$), confirming curvature; the interaction term was marginal ($p = 0.07$). The lack-of-fit test was non-significant ($p = 0.31$), supporting the second-order model. Path-of-steepest-ascent analysis was unnecessary because the optimum lay inside the design region." Always run a lack-of-fit test and report the canonical-form classification.

16.6.8 Practical Tips

- Rotatable CCDs give equal prediction variance at fixed distance from the centre and are the default RSM choice; the rotatability property is what makes CCDs ideal for sequential exploration of the response surface.
- Face-centred CCDs ($\alpha = 1$) are useful when factor levels cannot exceed the original cube — common when factor ranges are determined by safety or feasibility limits — at the cost of slightly reduced prediction-variance properties.
- Include 3–5 replicated centre points for pure-error degrees of freedom and curvature testing; fewer than 3 leaves the lack-of-fit test underpowered.
- After fitting, use `contour()` and `persp()` plots to visualise the response surface; engineers and scientists overwhelmingly prefer surface plots to coefficient tables when interpreting RSM results.
- Follow up the second-order fit with canonical analysis to classify the stationary point as a maximum, minimum, or saddle (eigenvalue signs of the quadratic-form matrix); a

saddle point usually means the optimum lies outside the design region and additional runs are warranted.
- For more than three factors, consider sequential experimentation: a fractional factorial first to screen for active factors, then a CCD or BBD on the active subset to characterise curvature.

16.6.9 R Packages Used

`rsm::ccd()` and `rsm::rsm()` for canonical CCD construction, second-order fitting, canonical analysis, and steepest-ascent paths; `qualityTools::rsmDesign()` for an alternative interface; `Vdgraph` for variance-dispersion graphs comparing rotatable, face-centred, and orthogonal CCD variants; `daewr` for textbook companion datasets and `DoE.wrapper` for higher-level DoE workflow integration.

16.6.10 For Reviewers

Use the shared reviewer checklist, together with this method's assumptions and practical tips.

Method-specific reviewer notes have not yet been developed for this entry. The links above support a structured review; they do not certify the example or a manuscript's application of the method.

16.6.11 See also — labs in this chapter

- Observational designs and STROBE
- Randomised controlled trials
- Adaptive, non-inferiority, equivalence trials
- Bench and translational design
- MCAR, MAR, MNAR
- Multiple imputation with mice
- Linear mixed models with lme4
- GLMMs and GEE
- DAGs with dagitty and ggdag
- Propensity scores and IPTW
- G-methods, IV, DiD, RDD; heterogeneity of treatment effect
- SIR and SEIR models with deSolve
- Pre-registration and statistical analysis plans

16.7 Completely Randomised Design

16.7.1 Introduction

The completely randomised design (CRD) is the simplest and most flexible of all comparative experimental designs: each of the available experimental units is assigned to one of several treatment groups by an unrestricted random mechanism, with no blocking, stratification, or other restriction on randomisation. CRD is the default starting point in any

introductory experimental-design course and remains the right choice whenever the experimental units are reasonably homogeneous and no nuisance source of variation justifies the additional complexity of a blocked layout. Its analysis is one-way ANOVA, its assumptions are minimal, and its results are easily generalised to whatever target population the units were sampled from.

16.7.2 Prerequisites

A working understanding of randomisation as the foundation of causal inference in experiments, one-way analysis of variance with the F-test, and the multiple-comparisons problem that arises when more than two treatment groups are compared.

16.7.3 Theory

With a treatment levels and r replicates per treatment, $n = a \cdot r$ experimental units are randomly allocated to treatments without any restriction. The standard analysis model is

$$y_{ij} = \mu + \tau_i + \varepsilon_{ij}, \qquad \varepsilon_{ij} \sim N(0, \sigma^2),$$

with τ_i the fixed effect of the i-th treatment and ε_{ij} independent Normal error. The omnibus null hypothesis is $\tau_1 = \cdots = \tau_a$, tested by the one-way ANOVA F-statistic. Post-hoc pairwise comparisons (Tukey HSD, Dunnett against a control, Scheffé for arbitrary contrasts) follow when the omnibus test is significant and pairwise inference is needed.

16.7.4 Assumptions

The experimental units are exchangeable prior to randomisation (homogeneous), the errors are independent within and across treatment groups, the within-group residuals are approximately Normal, and the residual variance is the same in each group. Departures from variance homogeneity can usually be addressed by transformation, weighted ANOVA, or Welch's F-test.

16.7.5 R Implementation

```r
set.seed(2026)
a ← 4; r ← 6
treatment ← factor(rep(LETTERS[1:a], each = r))
y ← rnorm(a * r, mean = rep(c(10, 12, 11, 14), each = r), sd = 2)

df ← data.frame(treatment, y)
fit ← aov(y ~ treatment, data = df)
summary(fit)

TukeyHSD(fit)
```

16.7.6 Output & Results

`aov()` returns the one-way ANOVA partitioning of the total sum of squares into between-group and within-group components, with the omnibus F-test reported as the primary inference. `TukeyHSD()` provides simultaneous 95 % confidence intervals for all pairwise

differences, controlling family-wise error rate at the chosen α, and is the standard follow-up when the omnibus null is rejected.

16.7.7 Interpretation

A reporting sentence: "The one-way ANOVA showed significant treatment differences ($F_{3,20} = 5.9$, $p = 0.005$); Tukey HSD identified treatment D as significantly higher than treatment A (mean difference 3.7, adjusted $p = 0.004$, 95 % CI 1.2 to 6.2). Other pairwise contrasts did not reach the adjusted $\alpha = 0.05$ level. Residual diagnostics (Q-Q plot, Levene's test $p = 0.41$) supported the Normal-equal-variance assumptions." Always pair the omnibus test with adjusted pairwise comparisons and a brief assumption-check statement.

16.7.8 Practical Tips

- CRD is most efficient when experimental units are reasonably homogeneous; if a substantial source of nuisance variation is known in advance, a randomised complete block design will reduce error variance and increase power without changing the inferential framework.
- Balanced designs with equal r across groups are optimal in efficiency and most robust to small departures from variance homogeneity; modest imbalance is acceptable but complicates type-III sum-of-squares interpretation.
- Always check residual diagnostics: a Q-Q plot, residuals-versus-fitted plot, and Levene's or Brown–Forsythe test for variance homogeneity. Mild departures rarely affect the omnibus F-test, but severe departures motivate a transformation or robust alternative.
- For markedly non-Normal responses, log or square-root transformations often suffice; for ordinal or otherwise non-transformable outcomes, use Kruskal–Wallis as a non-parametric omnibus and Dunn's procedure for post-hoc comparisons.
- Document the randomisation procedure (seed, software, date) so that the assignment list can be reconstructed from the protocol — a basic reproducibility requirement that auditors increasingly demand.
- If interest centres on a specific contrast (treatment vs control), Dunnett's procedure is more powerful than Tukey HSD and should be pre-specified in the protocol.

16.7.9 R Packages Used

Base R `aov()`, `lm()`, and `TukeyHSD()` cover the canonical CRD analysis; `agricolae` for `LSD.test()` and Duncan-style multiple-comparison tools used in agricultural DoE; `multcomp` for Dunnett, Scheffé, and arbitrary linear contrasts with adjustment; `emmeans` for marginal-mean estimation, plotting, and flexible contrast construction across designs.

16.7.10 For Reviewers

Use the shared reviewer checklist, together with this method's assumptions and practical tips.

Method-specific reviewer notes have not yet been developed for this entry. The links above support a structured review; they do not certify the example or a manuscript's application of the method.

16.7.11 See also — labs in this chapter

- Observational designs and STROBE
- Randomised controlled trials
- Adaptive, non-inferiority, equivalence trials
- Bench and translational design
- MCAR, MAR, MNAR
- Multiple imputation with mice
- Linear mixed models with lme4
- GLMMs and GEE
- DAGs with dagitty and ggdag
- Propensity scores and IPTW
- G-methods, IV, DiD, RDD; heterogeneity of treatment effect
- SIR and SEIR models with deSolve
- Pre-registration and statistical analysis plans

16.8 Constrained Mixture Designs

16.8.1 Introduction

In many real-world mixture experiments, each component carries individual lower and upper bounds — salt must remain at least 5 %, alcohol must not exceed 15 %, an active pharmaceutical ingredient must be between 2 and 8 % — so the feasible blend region is no longer the full simplex but a constrained convex polytope inside it. Standard simplex-lattice and simplex-centroid designs do not exploit this structure and would place runs outside the feasible region. Extreme-vertices designs (McLean and Anderson, 1966) instead enumerate the vertices of the constraint polytope and sample them, augmenting with edge midpoints, face centroids, and an overall centroid as the design budget allows. The result is a design that is simultaneously feasible everywhere and informative about the response across the full constrained region.

16.8.2 Prerequisites

A working understanding of simplex-lattice and simplex-centroid mixture designs, the Scheffé canonical polynomial, and basic convex-geometry concepts (vertices, edges, polytopes).

16.8.3 Theory

Given lower bounds L_i and upper bounds U_i on each of the q components, the feasible region is the convex polytope $\{x : L_i \le x_i \le U_i, \sum_i x_i = 1\}$. Its vertices are computable from the constraint set and form the natural design points; the design is augmented with

edge midpoints, face centroids, and the overall centroid as needed for fitting the chosen Scheffé polynomial order.

The pseudo-component transformation $x_i' = (x_i - L_i)/(1 - \sum_i L_i)$ maps the constrained region back to a standard unit simplex when only lower bounds are active, allowing standard simplex-lattice or simplex-centroid construction in the transformed coordinates. Predictions can then be back-transformed to the original-component scale.

16.8.4 Assumptions

Components are non-negative and sum to one (they are proportions of a fixed total amount), the lower and upper bounds are fixed and known in advance, and the residual error is approximately Normal with constant variance over the feasible region.

16.8.5 R Implementation

```
library(mixexp)

lb ← c(0.2, 0.1, 0.3)
ub ← c(0.5, 0.6, 0.7)

xv ← Xvert(nfac = 3, lc = lb, uc = ub, ndm = 2)
xv

set.seed(2026)
xv$y ← 10 * xv$x1 + 12 * xv$x2 + 8 * xv$x3 +
        6 * xv$x1 * xv$x2 +
        rnorm(nrow(xv), 0, 0.3)

fit ← MixModel(xv, "y",
               mixcomps = c("x1", "x2", "x3"), model = 2)
summary(fit)
```

16.8.6 Output & Results

`Xvert()` returns the vertices of the constrained polytope along with optionally requested edge midpoints and centroids. `MixModel()` fits the Scheffé linear, quadratic, or special-cubic mixture model on the design, with coefficients interpretable as blend effects. Visualising the polytope and overlaying the design points before running the experiment is a recommended sanity check.

16.8.7 Interpretation

A reporting sentence: “The extreme-vertices design identified seven feasible vertices of the constrained simplex with bounds $L = (0.2, 0.1, 0.3)$ and $U = (0.5, 0.6, 0.7)$. The fitted quadratic Scheffé model showed the response was maximised near the vertex $(0.5, 0.2, 0.3)$ with a meaningful pairwise synergy between components 1 and 2 ($\beta_{12} = 5.9$, $p = 0.01$). Linear blend coefficients indicated component 2 was individually most effective.” Always describe the feasible region and report which vertex achieved the optimum.

16.8.8 Practical Tips

- Use extreme-vertices designs whenever component bounds are genuinely binding; interior-only designs (simplex-lattice or centroid) miss optima that lie on the boundary of the feasible region — a common occurrence when constraints reflect formulation requirements.
- D-optimal mixture designs (via `AlgDesign::optMonteCarlo` or `AlgDesign::optFederov`) handle complex constraints elegantly and give the most efficient design when the number of feasible candidate points exceeds the run budget.
- Pseudo-component analysis makes the fit and interpretation more intuitive when only lower bounds are active: fit on the transformed unit simplex, then back-transform predictions to original component proportions for reporting.
- For ratio constraints (e.g., $\text{solvent}_1 : \text{solvent}_2 = 2 : 1$), use a ratio reparameterisation rather than naive component-level bounds; the constraint algebra is cleaner and the design more efficient.
- Always plot the feasible region in 3D or on the simplex before finalising the design; visualising the polytope helps detect over-tight constraints (a near-empty polytope) or accidentally redundant ones, and reveals which vertices are clinically meaningful.
- When the polytope is degenerate (very thin in one direction), the design will be ill-conditioned; relax the offending constraint or accept a higher prediction variance in the squeezed direction.

16.8.9 R Packages Used

`mixexp::Xvert()` for the canonical extreme-vertices design and `MixModel()` for Scheffé-model fitting; `AlgDesign::optFederov()` and `optMonteCarlo()` for D-optimal mixture designs under arbitrary linear constraints; `qualityTools::mixDesign()` for an alternative interface; `daewr` for textbook companion datasets and worked constrained-mixture examples; `MixExpR` for a tidyverse-friendly mixture-design workflow.

16.8.10 For Reviewers

Use the shared reviewer checklist, together with this method's assumptions and practical tips.

Method-specific reviewer notes have not yet been developed for this entry. The links above support a structured review; they do not certify the example or a manuscript's application of the method.

16.8.11 See also — labs in this chapter

- Observational designs and STROBE
- Randomised controlled trials
- Adaptive, non-inferiority, equivalence trials
- Bench and translational design
- MCAR, MAR, MNAR
- Multiple imputation with mice

- Linear mixed models with lme4
- GLMMs and GEE
- DAGs with dagitty and ggdag
- Propensity scores and IPTW
- G-methods, IV, DiD, RDD; heterogeneity of treatment effect
- SIR and SEIR models with deSolve
- Pre-registration and statistical analysis plans

16.9 Crossover Designs

16.9.1 Introduction

Crossover designs apply two or more treatments to each subject in sequence, separated by washout periods, so that every subject acts as their own control. The within-subject comparison gains substantial efficiency over parallel-group designs because between-subject variability — typically the largest component of variance in clinical-pharmacology studies — is removed from the treatment contrast. The simplest and most widely used crossover is the two-period two-treatment AB/BA design, in which half of the subjects receive treatment A in period 1 and B in period 2, and the other half receive the reverse sequence. Crossovers are standard in bioequivalence studies, in early-phase clinical pharmacology, and in many sensory and ergonomic experiments where conditions can be evaluated repeatedly on the same subject.

16.9.2 Prerequisites

A working understanding of repeated-measures analysis, mixed-effects models with subject as a random effect, and the concept of carryover (residual treatment effect from one period influencing the next).

16.9.3 Theory

In an AB/BA crossover, the standard mixed-effects analysis model is

$$y_{ijk} = \mu + \pi_j + \tau_k + s_i + \varepsilon_{ijk},$$

with period effect π_j, treatment effect τ_k, subject random effect $s_i \sim N(0, \sigma_s^2)$, and residual error ε_{ijk}. The treatment effect is estimated as the within-subject difference, eliminating the between-subject variance. **Carryover** — a residual treatment effect from period 1 affecting period 2 — manifests as a sequence-by-period interaction; if present, it biases the simple within-subject treatment estimate and requires either an adequate washout, a more complex Williams design, or a model that includes carryover terms.

16.9.4 Assumptions

Carryover is absent or has been controlled by an adequate washout period (pharmacologically, at least five half-lives of the active compound), the underlying condition is stable across periods (no progressive disease or natural recovery), and

treatment effects are independent of order. Subjects are exchangeable between sequences.

16.9.5 R Implementation

```
library(nlme)

set.seed(2026)
n ← 20
sequence ← sample(c("AB", "BA"), n, replace = TRUE)
subj_re ← rnorm(n, 0, 1)

y_A ← subj_re + rnorm(n, 0, 0.5)
y_B ← subj_re + 0.6 + rnorm(n, 0, 0.5)

df ← data.frame(
  subject = rep(1:n, each = 2),
  period  = rep(1:2, n),
  sequence = rep(sequence, each = 2),
  treatment = unlist(lapply(sequence, function(s) strsplit(s, "")[[1]])),
  y = unlist(lapply(1:n, function(i)
    if (sequence[i] == "AB") c(y_A[i], y_B[i]) else c(y_B[i], y_A[i])))
)

fit ← lme(y ~ treatment + period, random = ~ 1 | subject, data = df)
summary(fit)$tTable
```

16.9.6 Output & Results

The mixed-effects fit returns the treatment effect with within-subject standard error and the period effect as a separate fixed term. The random-intercept structure absorbs between-subject variance, dramatically tightening the treatment estimate compared with a parallel-group design of the same total sample size.

16.9.7 Interpretation

A reporting sentence: “The two-period AB/BA crossover analysis estimated the B–A treatment difference as 0.56 (95 % CI 0.32 to 0.80, $p < 0.001$), achieving roughly three-fold precision relative to a parallel-group design with the same number of subjects. The period effect was not significant ($p = 0.41$), and the sequence × period interaction (a marker of carryover) was also non-significant ($p = 0.62$), supporting the no-carryover assumption. The mandated washout of seven half-lives was sufficient.” Always report period and carryover diagnostics.

16.9.8 Practical Tips

- Adequate washout — at least five half-lives of the active compound, often more for compounds with long-lived metabolites — is the best defence against carryover; designing in conservative washouts is far cheaper than dealing with biased results post hoc.
- Avoid crossover designs for conditions that evolve naturally during the study (progressive disease, healing wounds, post-surgical recovery) or for interventions

whose effects are short-lived and wash out rapidly without further dosing; both violate the stability assumption.
- The 3 × 3 Williams design balances first-order carryover effects across treatment pairs when carryover is plausible and washout cannot be made arbitrarily long; it is the standard alternative to AB/BA in pharmacology.
- Always analyse with a mixed-effects model that includes subject as a random effect; pooled two-sample t-tests ignore the within-subject structure and overstate the true precision dramatically.
- Report the period effect separately from the treatment effect; a strong period effect (e.g., a drift in measurement instrument) can bias the treatment estimate if the design is imbalanced across sequences.
- Pre-specify the carryover diagnostic and what will be done if carryover is detected; reactive analysis decisions after seeing the data inflate type-I error and are increasingly flagged by regulators.

16.9.9 R Packages Used

`nlme::lme()` for mixed-effects analysis with explicit random-effect specification; `lme4::lmer()` and `lmerTest` as alternatives with Satterthwaite or Kenward-Roger denominator df; `Crossover` for canonical crossover-design construction including Williams squares; `crossdes` for systematic generation of balanced crossover layouts; `daewr` for textbook companion datasets including pharmacology examples.

16.9.10 For Reviewers

Use the shared reviewer checklist, together with this method's assumptions and practical tips.

Method-specific reviewer notes have not yet been developed for this entry. The links above support a structured review; they do not certify the example or a manuscript's application of the method.

16.9.11 See also — labs in this chapter

- Observational designs and STROBE
- Randomised controlled trials
- Adaptive, non-inferiority, equivalence trials
- Bench and translational design
- MCAR, MAR, MNAR
- Multiple imputation with mice
- Linear mixed models with lme4
- GLMMs and GEE
- DAGs with dagitty and ggdag
- Propensity scores and IPTW
- G-methods, IV, DiD, RDD; heterogeneity of treatment effect
- SIR and SEIR models with deSolve
- Pre-registration and statistical analysis plans

16.10 Desirability Functions

16.10.1 Introduction

Many experiments produce more than one response that must be optimised jointly: maximise yield while minimising impurity, hit a target dissolution time while keeping particle size within spec, maximise tensile strength while minimising weight. Desirability functions, introduced by Harrington in 1965 and refined by Derringer and Suich in 1980, are the standard tool for combining multiple responses into a single overall optimisation criterion. Each response is converted to a $[0, 1]$ desirability scale via a transformation specific to that response's goal (maximise, minimise, or hit a target), and the individual desirabilities are combined into an overall desirability via the geometric mean. The compound score is then maximised over the factor space, often using the response surfaces fit during the second-order RSM phase.

16.10.2 Prerequisites

A working understanding of response-surface methodology, multi-objective optimisation concepts, and the difference between geometric and arithmetic aggregation of normalised scores.

16.10.3 Theory

For each response y_i, define a desirability function $d_i(y_i) \in [0, 1]$ — larger values are more desirable, irrespective of which direction is "good" on the original scale. The standard forms (Derringer–Suich) are:

- **Maximise**: $d_i = 0$ below a lower limit, $d_i = 1$ above a target value, linearly increasing between.
- **Minimise**: mirror of the maximise form.
- **Target**: $d_i = 0$ at lower and upper limits, $d_i = 1$ at the target, with linear or power transitions.

The overall desirability is $D = \left(\prod_i d_i\right)^{1/n}$, the geometric mean. Crucially, if any single $d_i = 0$ then $D = 0$ — the geometric mean is "non-compensatory", meaning no amount of excellence on one response can rescue an unacceptable value on another.

16.10.4 Assumptions

Each d_i is appropriately parameterised with realistic limits and target values, the geometric-mean aggregation is acceptable to the decision-maker (alternative weighted-power means exist for differential weighting), and the underlying response surfaces are reasonably well estimated.

16.10.5 R Implementation

```r
library(desirability)

set.seed(2026)
df <- data.frame(x1 = runif(100, -1, 1), x2 = runif(100, -1, 1))
df$yield  <- 70 + 10 * df$x1 + 5 * df$x2 + rnorm(100)
df$cost   <- 20 - 2 * df$x1 + 4 * df$x2 + rnorm(100)

d_yield <- dMax(low = 60, high = 85)
d_cost  <- dMin(low = 10, high = 25)
dOverall <- dOverall(d_yield, d_cost)

df$d_overall <- predict(dOverall, with(df,
                                       cbind(yield = yield, cost = cost)))

head(df[order(-df$d_overall), ], 5)
```

16.10.6 Output & Results

The `desirability` package returns per-response and overall desirability for each observation or grid point. Combined with a fitted response-surface model, the workflow is to predict each response over a dense grid of factor settings, compute the overall desirability at each grid point, and identify the factor combination that maximises D. Reporting the achieved desirability for each individual response alongside the overall D communicates both the compromise and its quality.

16.10.7 Interpretation

A reporting sentence: "The best factor combination achieved yield 82 (90 % of the maximum desirability range) and cost 13 (85 % of the minimum desirability range), with overall $D = 0.87$. Pure yield optimisation would have driven yield to 88 but at cost 19 ($D = 0.42$); pure cost optimisation would have minimised cost to 11 but at yield 65 ($D = 0.31$). The desirability-balanced optimum is therefore meaningfully different from either single-objective optimum and is recommended for production." Always report the per-response desirabilities alongside D.

16.10.8 Practical Tips

- Use the geometric mean rather than the arithmetic mean to combine individual desirabilities; the geometric mean penalises low-desirability responses heavily and prevents one response from dominating the aggregate, while the arithmetic mean allows compensation that the decision-maker may not actually accept.
- Set realistic lower and upper limits for each desirability function; unrealistic limits flatten the function (everything inside is fully desirable, everything outside is zero) and the optimisation becomes uninformative.
- Differential weighting is achieved by exponentiating each desirability before averaging: $d_i^{s_i}$ with $s_i > 1$ increases the influence of response i, $s_i < 1$ decreases it. The Derringer–Suich formulation has this exponent as a built-in parameter.
- For fitted response surfaces, predict each response over a fine grid (or use a numerical optimiser) and maximise the overall desirability; the maximum is rarely at a

corner of the design region and benefits from explicit search.
- Desirability functions are a heuristic for combining responses; for high-stakes decisions, complement them with formal multi-objective Pareto analysis to expose the trade-off curve and let the decision-maker choose explicitly.
- Document the desirability-function specifications (limits, targets, exponents) in the protocol; reviewers and downstream users cannot reproduce the optimisation without them, and the choices materially affect the recommended factor settings.

16.10.9 R Packages Used

`desirability` for canonical Derringer–Suich functions (`dMax`, `dMin`, `dTarget`) and combined-desirability prediction (`dOverall`); `qualityTools::desirability()` for an alternative interface integrated with RSM workflow; `mco` for formal multi-objective Pareto-front analysis as a complement to desirability heuristics; `rsm` for fitting the response surfaces that feed into the desirability optimisation.

16.10.10 For Reviewers

Use the shared reviewer checklist, together with this method's assumptions and practical tips.

Method-specific reviewer notes have not yet been developed for this entry. The links above support a structured review; they do not certify the example or a manuscript's application of the method.

16.10.11 See also — labs in this chapter

- Observational designs and STROBE
- Randomised controlled trials
- Adaptive, non-inferiority, equivalence trials
- Bench and translational design
- MCAR, MAR, MNAR
- Multiple imputation with mice
- Linear mixed models with lme4
- GLMMs and GEE
- DAGs with dagitty and ggdag
- Propensity scores and IPTW
- G-methods, IV, DiD, RDD; heterogeneity of treatment effect
- SIR and SEIR models with deSolve
- Pre-registration and statistical analysis plans

16.11 2^k Factorial Designs

16.11.1 Introduction

The 2^k factorial design studies k factors, each at exactly two levels conventionally coded as -1 (low) and $+1$ (high), and runs every one of the 2^k possible combinations of factor levels. Because the design is balanced and orthogonal, every main effect and every interaction up to the k-way is estimable independently of every other, with all estimates having the same standard error. This combination of compactness, orthogonality, and full-information estimation makes the 2^k factorial the workhorse of industrial design of experiments (DoE), the natural starting point for product development and process improvement, and the foundation on which fractional factorials and response-surface designs are built.

16.11.2 Prerequisites

A working understanding of ANOVA with multiple factors, the concept of an interaction effect, and the use of coded factor levels (the orthogonal ± 1 scheme that simplifies computation and interpretation).

16.11.3 Theory

With coded levels ± 1, every factor effect is an orthogonal contrast: the main effect of factor A is the average response at $A = +1$ minus the average response at $A = -1$, and analogously for any interaction (a product of ± 1 columns averaged across runs). Orthogonality means each effect is estimated independently of every other effect, that the effects are uncorrelated, and that the analysis can proceed by simple linear regression with the coded factors and their products as predictors.

For an unreplicated 2^k run, residual degrees of freedom are zero and conventional ANOVA inference is impossible; Daniel's normal-probability plot of effects provides a graphical alternative that flags large effects against the inert ones, which form a Normal-distributed reference.

16.11.4 Assumptions

Two-level factor effects suffice to capture variation in the response (no curvature within the design region requiring a third level), runs are independent, and the residual error is approximately Normal with constant variance. Replicated designs satisfy these assumptions more transparently and allow formal F-tests; unreplicated designs rely on the small number of large effects against many inert effects ("effect sparsity") for inference.

16.11.5 R Implementation

```r
library(FrF2)

fac <- FrF2(nruns = 8, nfactors = 3, replications = 2,
            randomize = TRUE, seed = 2026)
head(fac)

fac$A <- as.numeric(as.character(fac$A))
fac$B <- as.numeric(as.character(fac$B))
fac$C <- as.numeric(as.character(fac$C))

set.seed(2026)
fac$y <- 10 + 2 * fac$A + 1.5 * fac$B + 0.8 * fac$A * fac$B +
         rnorm(nrow(fac), 0, 0.7)

fit <- lm(y ~ A * B * C, data = fac)
summary(fit)
```

16.11.6 Output & Results

`FrF2()` generates a randomised 2^k layout with replication. The fitted regression returns coefficients for all main effects and all interactions; in coded units these coefficients are exactly half the conventional "factor effect" magnitudes. With replication, residual degrees of freedom support standard t-tests and an ANOVA partitioning into main, interaction, and residual sums of squares.

16.11.7 Interpretation

A reporting sentence: "The 2^3 factorial design with two replicates estimated significant main effects of factor A (2.1, $p < 0.001$) and factor B (1.4, $p < 0.001$) and a significant AB interaction (0.8, $p = 0.01$); factor C and all higher-order interactions were not significantly different from zero. Combined with an interaction plot, the AB synergy suggests using both factors at their high levels yields the optimal response." Always pair the regression output with an interaction plot when AB-type interactions are significant.

16.11.8 Practical Tips

- Always randomise the run order to protect against time-trend confounding; running treatments in factor-level order produces designs in which a temporal drift could be mistaken for a factor effect.
- With replication, residual degrees of freedom allow formal t-tests and F-tests; without replication, Daniel's normal-probability plot of effects (or Lenth's pseudo-standard-error procedure) is the standard inferential tool, exploiting effect sparsity.
- Screen large factor counts ($k \geq 5$) first with fractional factorials of resolution III or IV, then follow up with a full factorial on the small subset of active effects identified — the classic two-stage industrial DoE workflow.
- Code factors as ± 1 rather than the natural units; coded coefficients are directly comparable, orthogonal, and interpretable as half-effects, while natural-unit coefficients depend on factor scale and confound location with effect.

- Always report effect size alongside the p-value; a statistically significant tiny effect rarely justifies a process change, while a large but underpowered effect may motivate follow-up experimentation regardless of significance.
- Pair main-effect plots and interaction plots with the regression table; visual diagnostics are far more informative than coefficient lists alone for engineers and scientists who must act on the result.

16.11.9 R Packages Used

`FrF2` for the canonical 2^k design generation, including replication and randomisation, and Daniel's plot via `DanielPlot()`; `DoE.base` for general orthogonal-array construction including 2^k as a special case; base R `lm()` and `aov()` for analysis; `unrepx` for unreplicated-design effect identification (Lenth, half-Normal, Box-Meyer); `daewr` for textbook companion datasets and analysis helpers.

16.11.10 For Reviewers

Use the shared reviewer checklist, together with this method's assumptions and practical tips.

Method-specific reviewer notes have not yet been developed for this entry. The links above support a structured review; they do not certify the example or a manuscript's application of the method.

16.11.11 See also — labs in this chapter

- Observational designs and STROBE
- Randomised controlled trials
- Adaptive, non-inferiority, equivalence trials
- Bench and translational design
- MCAR, MAR, MNAR
- Multiple imputation with mice
- Linear mixed models with lme4
- GLMMs and GEE
- DAGs with dagitty and ggdag
- Propensity scores and IPTW
- G-methods, IV, DiD, RDD; heterogeneity of treatment effect
- SIR and SEIR models with deSolve
- Pre-registration and statistical analysis plans

16.12 3^k Factorial Designs

16.12.1 Introduction

The 3^k factorial design uses three coded levels per factor — low (-1), middle (0), and high ($+1$) — and runs all 3^k combinations. The third level is what distinguishes the 3^k family from the simpler 2^k family: with three levels the design supports estimation of pure quadratic terms in each factor, and therefore the second-order polynomial approximation to a smooth response surface. The 3^k factorial is the simplest full-factorial construction that captures curvature and is the conceptual ancestor of the modern response-surface designs (central composite, Box-Behnken) that economise on the run count by replacing the dense 3^k grid with sparser alternatives carrying the same second-order information.

16.12.2 Prerequisites

Familiarity with 2^k factorial designs, the concept of an interaction effect, and quadratic regression for capturing curvature in a response.

16.12.3 Theory

For k three-level factors a full factorial has 3^k runs. With coded levels $\{-1, 0, +1\}$ the standard analysis model is the second-order polynomial

$$y = \beta_0 + \sum_i \beta_i x_i + \sum_i \beta_{ii} x_i^2 + \sum_{i<j} \beta_{ij} x_i x_j + \varepsilon,$$

with linear, pure-quadratic, and two-factor-interaction terms. The design grows quickly: $3^3 = 27$, $3^4 = 81$, $3^5 = 243$. For screening purposes the 3^k becomes impractical at $k = 4$ or beyond, and fractional 3^k designs or central composite / Box-Behnken layouts are far more economical for the same second-order information.

16.12.4 Assumptions

The response is well approximated by a second-order polynomial within the design region (no strong cubic or higher curvature), residuals are approximately Normal with constant variance, and the three levels span a range over which the polynomial approximation is plausible.

16.12.5 R Implementation

```
library(rsm)

set.seed(2026)
df ← expand.grid(A = c(-1, 0, 1), B = c(-1, 0, 1))
df$y ← 10 + 2 * df$A + 1.5 * df$B - df$A^2 +
        0.8 * df$A * df$B + rnorm(nrow(df), 0, 0.5)

fit ← rsm(y ~ SO(A, B), data = df)
summary(fit)
```

16.12.6 Output & Results

`rsm()` with the second-order specification returns coefficients for linear, pure-quadratic, and two-factor-interaction terms together with a stationary-point analysis (eigenvalue classification of the quadratic-form matrix as maximum, minimum, or saddle) and a lack-of-fit test against centre replicates when present. Reporting all components of the canonical analysis is the standard summary.

16.12.7 Interpretation

A reporting sentence: "The 3^2 full factorial estimated linear effects of A (2.0, $p < 0.001$) and B (1.5, $p = 0.002$), a significant quadratic term in A (-0.9, $p = 0.004$), and a mild interaction (0.7, $p = 0.07$). The stationary point of the fitted surface lay inside the design region and was a maximum, consistent with the expected concave behaviour. Predicted optimum was 12.4 at $(A, B) = (0.9, 0.8)$." Always pair the coefficient table with the canonical-analysis classification.

16.12.8 Practical Tips

- For $k \geq 3$, prefer central composite or Box-Behnken designs over full 3^k factorials; both deliver the same second-order information with far fewer runs and are the modern industrial-DoE default for response-surface modelling.
- Replicate the centre point ($x = 0$ for all factors) several times in any three-level design; the centre replicates supply pure-error degrees of freedom and a lack-of-fit test that diagnoses inadequacy of the second-order model.
- Coded levels $\{-1, 0, +1\}$ keep the design balanced and orthogonal, ensuring that linear, quadratic, and interaction effects are estimated independently and on directly comparable scales.
- Quadratic effects matter only when the response shows genuine curvature within the design region; if a 2^k factorial with replicated centre points fails to detect curvature, a 3^k or RSM follow-up is unnecessary.
- A 3^k full factorial is often a precursor to formal response-surface methodology; once the second-order surface is mapped, sequential experimentation along the path of steepest ascent or descent locates the optimum.
- For mixed-level designs (some factors at three levels, others at two) use orthogonal-array layouts via `DoE.base`; full crossed factorials become quickly unwieldy in

mixed-level cases.

16.12.9 R Packages Used

`rsm` for second-order polynomial fitting with stationary-point and canonical analysis; `FrF2.mix` for fractional three-level and mixed-level designs; `DoE.base` for general orthogonal-array construction including 3^k as a special case; `agricolae` for textbook three-level layouts; `Vdgraph` for variance-dispersion comparisons across 3^k, CCD, and BBD alternatives.

16.12.10 For Reviewers

Use the shared reviewer checklist, together with this method's assumptions and practical tips.

Method-specific reviewer notes have not yet been developed for this entry. The links above support a structured review; they do not certify the example or a manuscript's application of the method.

16.12.11 See also — labs in this chapter

- Observational designs and STROBE
- Randomised controlled trials
- Adaptive, non-inferiority, equivalence trials
- Bench and translational design
- MCAR, MAR, MNAR
- Multiple imputation with mice
- Linear mixed models with lme4
- GLMMs and GEE
- DAGs with dagitty and ggdag
- Propensity scores and IPTW
- G-methods, IV, DiD, RDD; heterogeneity of treatment effect
- SIR and SEIR models with deSolve
- Pre-registration and statistical analysis plans

16.13 Fractional Factorial Designs

16.13.1 Introduction

Fractional factorial designs run only a carefully chosen subset of a full 2^k factorial, accepting controlled confounding (also called aliasing) among effects in exchange for dramatic reductions in run count. They are the standard screening tool in industrial DoE: when many candidate factors must be evaluated and only a few are expected to matter, a fractional factorial sifts the active factors from the inert ones at a fraction of the cost of a

full factorial. The trade-off is that some pairs (or larger groups) of effects share their estimating contrast and cannot be disentangled from the data alone, so design selection requires careful attention to which aliasing patterns are scientifically tolerable.

16.13.2 Prerequisites

A working understanding of full 2^k factorial designs, the concept of confounding among effect contrasts, and the Box–Hunter–Hunter sparsity-of-effects principle.

16.13.3 Theory

A 2^{k-p} design uses 2^{k-p} runs — a $1/2^p$-fraction of the corresponding full 2^k. The choice of p generators (defining relations among the factor columns) determines which effects are aliased with which. The design's **resolution** is the length of the shortest word in the defining relation:

- **Resolution III**: main effects are aliased with two-factor interactions; suitable for early screening only, where interactions are assumed negligible.
- **Resolution IV**: main effects are clear of two-factor interactions, but two-factor interactions alias with each other.
- **Resolution V**: main effects and two-factor interactions are all clear of each other; only three-factor and higher interactions are aliased.

Higher resolution generally requires more runs; the design choice is a balance between run economy and inferential cleanliness.

16.13.4 Assumptions

The sparsity-of-effects principle holds: lower-order effects (main, two-factor) dominate higher-order interactions, so aliasing among higher-order effects is acceptable. Each run is independent, residual error is approximately Normal with constant variance, and the chosen resolution is appropriate to the scientific question.

16.13.5 R Implementation

```r
library(FrF2)

frac <- FrF2(nruns = 16, nfactors = 5, resolution = 5, randomize = TRUE)
design.info(frac)$aliased

set.seed(2026)
for (v in LETTERS[1:5]) frac[[v]] <- as.numeric(as.character(frac[[v]]))
frac$y <- 10 + 2 * frac$A + 1.5 * frac$B - 0.8 * frac$C +
          0.6 * frac$A * frac$B + rnorm(16, 0, 0.5)

fit <- lm(y ~ (A + B + C + D + E)^2, data = frac)
summary(fit)
```

16.13.6 Output & Results

`FrF2()` constructs the fractional factorial layout with the requested resolution; `design.info()` documents the alias structure explicitly so users see which effects are confounded with which. The regression fit returns coefficients for all main effects and two-factor interactions (the latter representing the joint estimate of any aliased pair), supporting standard t-tests on each.

16.13.7 Interpretation

A reporting sentence: "The 2^{5-1} resolution-V design used 16 runs — half of the full 32-run factorial — with main effects clear of all two-factor interactions and only the highest-order alias chain $ABCDE = I$. Significant effects were A (2.0, $p < 0.001$), B (1.5, $p = 0.001$), C (-0.8, $p = 0.02$), and the AB interaction (0.6, $p = 0.04$); factors D and E were inactive. The active subset $\{A, B, C\}$ will be followed up in a full factorial with replicated centre points to characterise curvature." Always document the alias structure.

16.13.8 Practical Tips

- Use resolution IV or V for screening whenever enough runs are available; resolution III is acceptable only for very early screening when interactions are confidently negligible and clean main-effect estimates are not essential.
- Resolution III Plackett–Burman designs are more economical (any number of factors that is a multiple of 4) but confound main effects with two-factor interactions; follow them up with a fold-over fraction or a higher-resolution design before drawing causal conclusions.
- Fold-over designs convert a resolution III into a resolution IV by adding a mirror-image fraction with all factor signs reversed; this de-aliases main effects from two-factor interactions at the cost of doubling the run count.
- After the screening step, follow up with a full factorial or response-surface design on the small subset of active factors identified; the classical industrial DoE workflow is "screen $\rightarrow$ characterise $\rightarrow$ optimise" in three sequential stages.
- Include centre points in fractional factorials whenever feasible; they enable a curvature test against the average of the factorial corners and provide pure-error degrees of freedom for lack-of-fit testing.
- Always document the alias chain in the methods section; reviewers will check it, and an undocumented aliasing pattern is a recurring source of misinterpreted DoE results.

16.13.9 R Packages Used

`FrF2` for canonical fractional-factorial construction with resolution control, alias-structure documentation, and Daniel-plot diagnostics; `DoE.base` for general orthogonal-array construction including fractional factorials; `BsMD` for Bayesian screening of fractional designs; `unrepx` for unreplicated-design effect identification (Lenth, half-Normal, Box-Meyer); `daewr` for textbook companion datasets.

16.13.10 For Reviewers

Use the shared reviewer checklist, together with this method's assumptions and practical tips.

Method-specific reviewer notes have not yet been developed for this entry. The links above support a structured review; they do not certify the example or a manuscript's application of the method.

16.13.11 See also — labs in this chapter

- Observational designs and STROBE
- Randomised controlled trials
- Adaptive, non-inferiority, equivalence trials
- Bench and translational design
- MCAR, MAR, MNAR
- Multiple imputation with mice
- Linear mixed models with lme4
- GLMMs and GEE
- DAGs with dagitty and ggdag
- Propensity scores and IPTW
- G-methods, IV, DiD, RDD; heterogeneity of treatment effect
- SIR and SEIR models with deSolve
- Pre-registration and statistical analysis plans

16.14 Graeco-Latin Square Designs

16.14.1 Introduction

A Graeco-Latin square superimposes two orthogonal Latin squares to block simultaneously on three nuisance factors in a single $t \times t$ layout. Each of the t treatments appears exactly once in every row, every column, and every Greek-letter level, so the four classifications — rows, columns, Greek letters, and Latin treatment letters — are mutually orthogonal. The design is extraordinarily compact: t^2 runs accommodate t treatments with three layers of blocking, where a comparable randomised-block design would need t^3 runs to achieve the same blocking. This makes Graeco-Latin squares classical tools in industrial and agricultural experiments where each run is expensive and three nuisance factors (operator, machine, day; or row, column, fertiliser-batch) need to be controlled simultaneously.

16.14.2 Prerequisites

Familiarity with Latin square designs, the orthogonality of factor classifications, and the concept of degrees-of-freedom partitioning in block ANOVA models.

16.14.3 Theory

The standard model is

$$y_{ijkl} = \mu + \rho_i + \gamma_j + \alpha_k + \tau_l + \varepsilon_{ijkl},$$

with row effect ρ_i, column effect γ_j, Greek-letter effect α_k, treatment effect τ_l, and independent Normal error. Orthogonal Latin squares of order t — required to construct the design — exist for all t except $t = 2$ and $t = 6$, the famous exceptions from the work of Tarry and the Bose–Shrikhande–Parker theory of mutually orthogonal Latin squares (Fisher's argument originally excluded $t = 6$).

16.14.4 Assumptions

No interactions exist between any pair of the four classifications (treatment, row, column, Greek), the residual error has homogeneous variance, and the response is approximately Normal. Because residual degrees of freedom are very small for the smallest squares, the design is sensitive to violations of these assumptions.

16.14.5 R Implementation

```
library(agricolae)

set.seed(2026)
trts ← LETTERS[1:4]
greek ← letters[1:4]
gls ← design.graeco(trts, greek, seed = 2026)
print(gls$book)

gls$book$y ← rnorm(16, mean = c(10, 12, 14, 11)[as.integer(factor(gls$book$trts))],
sd = 1.3)
fit ← aov(y ~ row + col + greek + trts, data = gls$book)
summary(fit)
```

16.14.6 Output & Results

`design.graeco()` returns the experimental layout with row, column, Greek, and treatment indicators for each of the t^2 runs. The ANOVA partitions the total sum of squares into row, column, Greek, treatment, and residual components, with the treatment F-test as the primary inference. Reporting all four block effects alongside the treatment effect gives a complete picture of how variability was partitioned.

16.14.7 Interpretation

A reporting sentence: "The Graeco-Latin square blocked simultaneously on row, column, and Greek-letter classifications. The treatment main effect was marginal ($F_{3,3} = 6.4$, $p = 0.08$); residual degrees of freedom were limited ($\mathrm{df}_{\mathrm{res}} = 3$) so the test had low power, and a replicated design or a Latin square with one fewer blocking layer would have given more stable inference. Block sums of squares were modest, indicating the nuisance factors did not dominate the response variability." Always note residual df explicitly.

16.14.8 Practical Tips

- Residual degrees of freedom in a $t \times t$ Graeco-Latin square are $(t-1)(t-3)$, which is dangerously small for $t \leq 5$; for $t = 4$ only 3 residual df remain, severely limiting power.
- Replicate the entire square if residual df are too few; replicated Graeco-Latin squares restore power without losing the orthogonal blocking structure.
- The design is not available for $t = 2$ (trivially) or $t = 6$ (the Bose–Shrikhande–Parker non-existence result); for $t = 6$ a Latin square or a different blocking scheme must be used.
- The analysis assumes additivity of all four classifications; substantial interactions among rows, columns, Greek levels, and treatment invalidate the F-test, and a Tukey one-degree-of-freedom test for non-additivity is a useful diagnostic.
- Graeco-Latin squares are most common in agricultural field trials and industrial process-screening; in clinical-biostatistics they are rare because the assumed lack of interactions is hard to justify.
- Higher-order extensions (hyper-Graeco-Latin squares with four or more orthogonal Latin squares superimposed) exist but consume residual df even more aggressively and are seldom used in practice.

16.14.9 R Packages Used

`agricolae::design.graeco()` for the canonical layout construction and randomisation; `crossdes` for systematic generation of mutually orthogonal Latin squares of higher order; `DoE.base` for general orthogonal-array construction encompassing Graeco-Latin layouts; `lme4` and `lmerTest` for mixed-model analysis when block effects are treated as random.

16.14.10 For Reviewers

Use the shared reviewer checklist, together with this method's assumptions and practical tips.

Method-specific reviewer notes have not yet been developed for this entry. The links above support a structured review; they do not certify the example or a manuscript's application of the method.

16.14.11 See also — labs in this chapter

- Observational designs and STROBE
- Randomised controlled trials
- Adaptive, non-inferiority, equivalence trials
- Bench and translational design
- MCAR, MAR, MNAR
- Multiple imputation with mice
- Linear mixed models with lme4
- GLMMs and GEE
- DAGs with dagitty and ggdag
- Propensity scores and IPTW

- G-methods, IV, DiD, RDD; heterogeneity of treatment effect
- SIR and SEIR models with deSolve
- Pre-registration and statistical analysis plans

16.15 Latin Square Designs

16.15.1 Introduction

A Latin square design simultaneously blocks on two nuisance factors — the rows and columns of a $t \times t$ grid — with t treatments arranged so that every treatment appears exactly once in each row and exactly once in each column. The result is a remarkably economical design: t treatments are replicated t times in only t^2 experimental units while controlling for two layers of nuisance variation. Latin squares are classical workhorses of agricultural experimentation (where rows and columns might be soil-fertility gradients in two perpendicular directions), industrial DoE (where day and operator are blocked), and clinical-pharmacology crossover trials (where subject and period block the treatment sequence).

16.15.2 Prerequisites

A working understanding of completely randomised designs, the principle of blocking, and the partitioning of variability in two-way ANOVA models without replication.

16.15.3 Theory

The standard analysis model is

$$y_{ijk} = \mu + \rho_i + \gamma_j + \tau_k + \varepsilon_{ijk}, \qquad \varepsilon_{ijk} \sim N(0, \sigma^2),$$

with row effect ρ_i, column effect γ_j, treatment effect τ_k, and independent Normal error. The constraint that each treatment appears once per row and once per column defines a Latin square. The design has $(t-1)$ treatment df, $(t-1)$ row df, $(t-1)$ column df, and $(t-1)(t-2)$ residual df, requiring t^2 runs in total.

16.15.4 Assumptions

No interactions exist between any pair of the three classifications (treatment, row, column), residual variance is homogeneous, and the response is approximately Normal. The design uses one observation per cell, so departures from these assumptions cannot easily be diagnosed from replication; residual diagnostics and a Tukey one-degree-of-freedom test for non-additivity are essential checks.

16.15.5 R Implementation

```
library(agricolae)

set.seed(2026)
t ← 4
trts ← LETTERS[1:t]
ls_design ← design.lsd(trts, seed = 2026)
book ← ls_design$book
print(book)

book$y ← rnorm(t^2, mean = c(10, 12, 14, 11)[as.integer(book$trts)], sd = 1.5)
fit ← aov(y ~ row + col + trts, data = book)
summary(fit)
```

16.15.6 Output & Results

`design.lsd()` returns a randomised Latin-square layout with row, column, and treatment indicators. The ANOVA partitions the total sum of squares into row, column, treatment, and residual components, with the treatment F-test as the primary inference. Reporting the proportions of variability absorbed by row and column blocking alongside the treatment effect characterises the gain from the design.

16.15.7 Interpretation

A reporting sentence: "The 4×4 Latin square partitioned the 16 observations' variation into row, column, and treatment components. The treatment F-test was significant ($F_{3,6} = 7.2$, $p = 0.02$), with row blocking absorbing 20 % of total variability and column blocking 14 %; residual variance was correspondingly small. The design assumes no row-treatment or column-treatment interaction, supported by visual inspection of the residuals." Always report blocking variance components and verify the no-interaction assumption.

16.15.8 Practical Tips

- Latin squares have very few residual df for small t (only 6 for $t = 4$); consider replication or a larger square if effects of interest are modest in size.
- The analysis assumes no interactions between treatment and either blocking factor; substantive interactions invalidate the standard F-test, and the no-interaction assumption should always be defended from context (or tested via a Tukey one-df non-additivity check).
- For $t > 7$ a Latin square requires too many units ($t^2 \geq 49$); a balanced incomplete block design or a fractional-factorial blocked layout is usually more practical.
- Graeco-Latin squares extend to a third blocking factor by superimposing two orthogonal Latin squares; they are even more economical but with even fewer residual df and the same additive-effects assumption.
- Always randomly permute rows, columns, and treatment labels of a standard Latin square to obtain a valid realisation; using an unrandomised square confounds systematic patterns with treatment effects.

- In crossover clinical trials, “subject” plays the role of one block and “period” the role of the other; carryover effects, if present, violate the no-interaction assumption and must be modelled separately.

16.15.9 R Packages Used

`agricolae::design.lsd()` for the canonical layout and randomisation; base R `aov()` for the ANOVA analysis; `crossdes` for systematic Latin-square construction including replicated and balanced layouts; `daewr` for textbook examples; `emmeans` for marginal means and treatment contrasts after fitting.

16.15.10 For Reviewers

Use the shared reviewer checklist, together with this method’s assumptions and practical tips.

Method-specific reviewer notes have not yet been developed for this entry. The links above support a structured review; they do not certify the example or a manuscript’s application of the method.

16.15.11 See also — labs in this chapter

- Observational designs and STROBE
- Randomised controlled trials
- Adaptive, non-inferiority, equivalence trials
- Bench and translational design
- MCAR, MAR, MNAR
- Multiple imputation with mice
- Linear mixed models with lme4
- GLMMs and GEE
- DAGs with dagitty and ggdag
- Propensity scores and IPTW
- G-methods, IV, DiD, RDD; heterogeneity of treatment effect
- SIR and SEIR models with deSolve
- Pre-registration and statistical analysis plans

16.16 Simplex Centroid Designs

16.16.1 Introduction

Simplex centroid designs are mixture experiments that include the centroids of every sub-simplex of the component simplex: each pure component (vertex), each binary midpoint (edge centroid), each ternary centroid (face centroid), and on up to the overall centroid of the full simplex. The result is a set of $2^q - 1$ design points for q components that spans all blend orders simultaneously, making the design well suited to detecting higher-order synergies — three-way and four-way blend interactions that simpler simplex-lattice designs cannot estimate. They are widely used in food science, pharmaceutical

formulation, and chemical product development whenever the response depends on combinations of multiple ingredients and pure components alone do not capture the behaviour.

16.16.2 Prerequisites

Familiarity with mixture experiments, the simplex-lattice design, and the Scheffé canonical polynomial form for mixture models.

16.16.3 Theory

For q components a simplex centroid design has $2^q - 1$ points: $\binom{q}{1} = q$ vertices, $\binom{q}{2}$ edge midpoints, $\binom{q}{3}$ face centroids, and so on, up to the overall centroid. The Scheffé canonical polynomial including all blend orders is

$$y = \sum_i \beta_i x_i + \sum_{i<j} \beta_{ij} x_i x_j + \sum_{i<j<k} \beta_{ijk} x_i x_j x_k + \cdots,$$

with no intercept (because $\sum_i x_i = 1$). The simplex centroid design supplies exactly enough information to estimate every blend term up to the q-way interaction, in contrast to simplex-lattice designs that estimate only up to a chosen order $m \le q$.

16.16.4 Assumptions

Components are non-negative and sum to one (so the design lies on the standard simplex), higher-order synergies are potentially meaningful (otherwise a lower-order simplex-lattice would suffice), and residual error is approximately Normal with constant variance across the simplex.

16.16.5 R Implementation

```
library(mixexp)

design ← SCD(3)
design

set.seed(2026)
design$y ← 10 * design$x1 + 12 * design$x2 + 8 * design$x3 +
            6 * design$x1 * design$x2 +
            4 * design$x1 * design$x3 +
            3 * design$x1 * design$x2 * design$x3 +
            rnorm(nrow(design), 0, 0.3)

fit ← MixModel(design, "y",
                mixcomps = c("x1", "x2", "x3"), model = 3)
summary(fit)
```

16.16.6 Output & Results

`SCD()` returns the seven design points for a three-component simplex centroid (three vertices, three edge midpoints, the centroid). Fitting the cubic Scheffé model yields coefficients for the linear blend, the three pairwise blend interactions, and the ternary blend. Reporting the ternary coefficient explicitly is the canonical way to demonstrate that the centroid design has captured higher-order synergy beyond pairwise interaction.

16.16.7 Interpretation

A reporting sentence: "The seven-run simplex centroid design with cubic Scheffé fit detected a meaningful ternary synergy ($\beta_{123} = 3.1$, $p = 0.02$) beyond the pairwise blend interactions ($\beta_{12} = 6.0$, $\beta_{13} = 3.9$). The ternary centroid (1/3, 1/3, 1/3) achieved a fitted response of 11.3, exceeding any binary edge centroid; the optimal blend therefore mixes all three components rather than just pairs." Always report the higher-order coefficients explicitly when justifying a centroid design.

16.16.8 Practical Tips

- Simplex centroid designs require more runs than simplex-lattice designs of the same maximum interaction order; use the centroid form only when higher-order blends are scientifically plausible and worth estimating.
- Plot the fitted response surface on the ternary (three-component) or quaternary simplex; mixture-design intuition is overwhelmingly visual, and contour plots on the simplex reveal where the optimum lies far more clearly than coefficient lists.
- Augment the simplex centroid with axial points (positions partway between the centroid and each vertex) to improve coverage of non-centroid mixtures and stabilise prediction in the interior of the simplex.
- For bounded mixtures with explicit constraints on minimum or maximum component proportions, extreme-vertices designs are the appropriate alternative; the standard centroid form assumes the full unconstrained simplex.
- Mixture-amount designs cross the simplex with a total-amount factor and are appropriate when both formulation and total dose matter; they are common in pharmaceutical formulation and feed-mixture studies.
- The `mixexp` and `MixExpR` packages also support mixture-process designs that combine mixture components with non-mixture process variables (e.g., temperature, pressure); these are increasingly common in chemical-engineering DoE.

16.16.9 R Packages Used

`mixexp::SCD()` and `mixexp::MixModel()` for the canonical simplex centroid design and Scheffé-model fitting; `MixExpR` for an alternative tidyverse-friendly mixture-design ecosystem; `qualityTools::mixDesign()` for related mixture-design construction; `daewr` for textbook companion datasets and worked examples.

16.16.10 For Reviewers

Use the shared reviewer checklist, together with this method's assumptions and practical tips.

Method-specific reviewer notes have not yet been developed for this entry. The links above support a structured review; they do not certify the example or a manuscript's application of the method.

16.16.11 See also — labs in this chapter

- Observational designs and STROBE
- Randomised controlled trials
- Adaptive, non-inferiority, equivalence trials
- Bench and translational design
- MCAR, MAR, MNAR
- Multiple imputation with mice
- Linear mixed models with lme4
- GLMMs and GEE
- DAGs with dagitty and ggdag
- Propensity scores and IPTW
- G-methods, IV, DiD, RDD; heterogeneity of treatment effect
- SIR and SEIR models with deSolve
- Pre-registration and statistical analysis plans

16.17 Simplex Lattice Mixture Designs

16.17.1 Introduction

Mixture experiments study how a response depends on the relative proportions of two or more components in a blend, where the proportions are constrained to sum to one. They arise routinely in food formulation (recipes), paint and coating chemistry, pharmaceutical excipient blending, fuel formulation, and any setting in which the absolute amount is fixed and only the relative composition can vary. Because the simplex constraint $x_1 + \cdots + x_q = 1$ removes one degree of freedom, standard polynomial regression models are not directly applicable, and Scheffé's canonical polynomial forms — which omit the intercept and pure quadratic terms — are the appropriate functional family. Simplex-lattice designs are the foundational mixture-design construction and the natural starting point for any mixture experiment.

16.17.2 Prerequisites

A working understanding of factorial designs, basic simplex geometry, and the Scheffé canonical polynomial form for mixture models.

16.17.3 Theory

A $\{q, m\}$ simplex-lattice design for q components samples all blends in which each component takes a proportion in the set $\{0, 1/m, 2/m, \ldots, 1\}$, restricted to those proportions that sum to one. For $q = 3$ and $m = 2$, the design has six points: three pure components (vertices of the simplex), and three binary midpoints (edge centres at $(1/2, 1/2, 0)$, $(1/2, 0, 1/2)$, $(0, 1/2, 1/2)$). The Scheffé canonical polynomial of order m in canonical form is

$$y = \sum_i \beta_i x_i + \sum_{i<j} \beta_{ij} x_i x_j + \cdots,$$

with no intercept (because the linear terms already absorb a constant) and no pure-quadratic terms (because $\sum x_i = 1$ implies $x_i^2 = x_i - \sum_{j \neq i} x_i x_j$, redistributing the squared terms into linear and interaction blends).

16.17.4 Assumptions

The components sum to one (so the design lies on the standard $(q-1)$-simplex), the response is a smooth function of the component proportions, the residual error is approximately Normal with constant variance, and the chosen Scheffé polynomial order m adequately captures the surface curvature.

16.17.5 R Implementation

```
library(mixexp)

design ← SLD(3, 2)
design

set.seed(2026)
design$y ← 10 * design$x1 + 12 * design$x2 + 8 * design$x3 +
            6 * design$x1 * design$x2 +
            rnorm(nrow(design), 0, 0.3)

fit ← MixModel(design, "y", mixcomps = c("x1", "x2", "x3"), model = 2)
summary(fit)
```

16.17.6 Output & Results

`SLD()` returns the simplex-lattice points, and `MixModel()` fits the Scheffé polynomial of the requested order. The output reports per-component linear blend coefficients and pairwise interaction coefficients; reading these together with a contour plot on the ternary simplex is the standard way to identify the optimal blend.

16.17.7 Interpretation

A reporting sentence: "The $\{3, 2\}$ simplex-lattice mixture design with six runs identified component 2 as the strongest pure component ($\beta_2 = 12.1$) and a positive synergy between components 1 and 2 ($\beta_{12} = 5.9$). The fitted optimum lies on the x_1-x_2 edge near

$(0.5, 0.5, 0)$ with predicted response 12.5, exceeding any pure-component value. A follow-up centroid-augmented design will confirm whether the optimum truly lies on the binary edge or shifts toward the interior." Always interpret pure components alongside blend interactions.

16.17.8 Practical Tips

- Report coefficients only in the Scheffé canonical form; standard polynomial models with intercept and pure quadratic terms misrepresent mixture data because they ignore the simplex constraint.
- Augment simplex-lattice designs with the overall centroid (and optionally axial points partway between centroid and vertices) for better prediction of non-extreme blends; pure simplex-lattice designs over-sample the boundary and under-sample the interior.
- For experiments with binding lower or upper bounds on components, use constrained mixture designs with extreme-vertices construction or pseudo-component reparameterisation; unconstrained simplex designs would place runs in the infeasible region.
- Mixture-process variable designs cross the simplex with one or more factorial factors (e.g., mixing temperature, time, pressure); they are common in chemical-engineering DoE and require a combined Scheffé + standard-polynomial model.
- Contour plots on the ternary simplex (or generalisations to higher dimensions) are the standard visualisation; mixture intuition is overwhelmingly visual, and contour plots reveal optimal blends and constraint-binding edges far better than coefficient tables.
- For more than four or five components, full simplex-lattice designs become impractical; D-optimal mixture designs or fractional simplex constructions are the modern alternative.

16.17.9 R Packages Used

`mixexp::SLD()` and `mixexp::MixModel()` for the canonical simplex-lattice design and Scheffé-model fitting; `mixexp::SCD()` for the simplex-centroid extension; `MixExpR` for a tidyverse-friendly mixture-design ecosystem; `qualityTools::mixDesign()` for an alternative mixture-design toolkit; `AlgDesign` for D-optimal mixture designs under arbitrary constraints.

16.17.10 For Reviewers

Use the shared reviewer checklist, together with this method's assumptions and practical tips.

Method-specific reviewer notes have not yet been developed for this entry. The links above support a structured review; they do not certify the example or a manuscript's application of the method.

16.17.11 See also — labs in this chapter

- Observational designs and STROBE
- Randomised controlled trials
- Adaptive, non-inferiority, equivalence trials
- Bench and translational design
- MCAR, MAR, MNAR
- Multiple imputation with mice
- Linear mixed models with lme4
- GLMMs and GEE
- DAGs with dagitty and ggdag
- Propensity scores and IPTW
- G-methods, IV, DiD, RDD; heterogeneity of treatment effect
- SIR and SEIR models with deSolve
- Pre-registration and statistical analysis plans

16.18 Optimal Designs: D, A, I

16.18.1 Introduction

Computer-generated optimal designs minimise a chosen criterion on the information matrix. When a standard design does not fit constraints (irregular region, mixed-level factors, restricted runs), optimal designs are the primary recourse.

16.18.2 Prerequisites

Linear regression; Fisher information matrix.

16.18.3 Theory

For a design matrix X under the assumed model, the information matrix is $M = X^T X$. Common criteria: - **D-optimal**: maximise $\det(M)$ – minimises the volume of the joint confidence ellipsoid for coefficients. - **A-optimal**: minimise trace(M^{-1}) – minimises the average variance of coefficient estimates. - **I-optimal**: minimise average prediction variance over the design region. - **G-optimal**: minimise the maximum prediction variance.

Computer algorithms (Fedorov exchange) search for the optimum.

16.18.4 Assumptions

Model is correctly specified; $n \geq p$ (parameters); design region is explicit.

16.18.5 R Implementation

```r
library(AlgDesign)

set.seed(2026)
# Candidate set: 3^2 factorial grid
candidates <- expand.grid(x1 = c(-1, 0, 1), x2 = c(-1, 0, 1))

# D-optimal design for a second-order model, selecting 8 runs
d_opt <- optFederov(~ quad(x1, x2),
                    data = candidates,
                    nTrials = 8,
                    criterion = "D",
                    nRepeats = 20)
d_opt$design

# I-optimal
i_opt <- optFederov(~ quad(x1, x2),
                    data = candidates,
                    nTrials = 8,
                    criterion = "I",
                    nRepeats = 20)
i_opt$design
```

16.18.6 Output & Results

Two sets of 8 runs from the 9-point candidate set; D-optimal maximises determinant of $X^T X$; I-optimal minimises average prediction variance.

16.18.7 Interpretation

“D-optimal selected the 8 runs that most tightly constrain the polynomial coefficients; I-optimal picked a different subset that minimises prediction variance across the design region. Choose D for inference on coefficients, I for prediction.”

16.18.8 Practical Tips

- Use D-optimal for parameter-precision goals; I-optimal for prediction-focused studies.
- Generate several random starts (`nRepeats`); optimal-design search is non-convex.
- Verify the final design’s efficiency against standard designs if one is applicable.
- Use optimal designs when standard textbook designs don’t fit constraints.
- Mixed-level optimal designs can handle categorical factors; pass them as factors to `optFederov`.

16.18.9 For Reviewers

Use the shared reviewer checklist, together with this method’s assumptions and practical tips.

Method-specific reviewer notes have not yet been developed for this entry. The links above support a structured review; they do not certify the example or a manuscript’s application of the method.

16.18.10 See also — labs in this chapter

- Observational designs and STROBE
- Randomised controlled trials
- Adaptive, non-inferiority, equivalence trials
- Bench and translational design
- MCAR, MAR, MNAR
- Multiple imputation with mice
- Linear mixed models with lme4
- GLMMs and GEE
- DAGs with dagitty and ggdag
- Propensity scores and IPTW
- G-methods, IV, DiD, RDD; heterogeneity of treatment effect
- SIR and SEIR models with deSolve
- Pre-registration and statistical analysis plans

16.19 Orthogonal Arrays

16.19.1 Introduction

Orthogonal arrays (OAs) are tabulated fractional-factorial designs in which every pair of columns contains every combination of levels an equal number of times. Genichi Taguchi popularised a catalogue of specific OAs — L_4, L_8, L_9, L_{16}, L_{18}, L_{27}, and others — as economical screening designs in his quality-engineering framework, and the labels are now standard shorthand in industrial DoE. The orthogonality property guarantees that main effects of distinct factors are estimated independently and on equal footing, so OAs deliver the cleanest possible main-effect estimates from the smallest possible run count, given the assumed effect-sparsity context.

16.19.2 Prerequisites

Familiarity with fractional factorial designs, the Plackett–Burman family, and the concept of orthogonality among columns of a design matrix.

16.19.3 Theory

An $L_n(m^k)$ array has n rows (runs) and k columns (factors), each column at m equally-replicated levels. Orthogonality means that for any pair of columns each combination of levels appears an equal number of times, ensuring uncorrelated main-effect estimates.

The standard catalogue includes:

- **L4**: 4 runs, up to 3 two-level factors.
- **L8**: 8 runs, up to 7 two-level factors.
- **L9**: 9 runs, up to 4 three-level factors.
- **L16**: 16 runs, up to 15 two-level factors.
- **L18**: 18 runs, one two-level factor plus 7 three-level factors (mixed-level).

- **L27**: 27 runs, up to 13 three-level factors.

16.19.4 Assumptions

Main effects dominate the response (effect-sparsity principle); two-factor and higher-order interactions are either negligible or have been assigned to specific pre-determined columns by reading the array's linear-graph; runs are independent with constant residual variance.

16.19.5 R Implementation

```
library(qualityTools)

l9 ← taguchiDesign("L9_3^4")
summary(l9)

set.seed(2026)
l9$y ← rnorm(9, mean = 10 + 2 * as.numeric(l9$A) -
                     1.5 * as.numeric(l9$C), sd = 0.5)

effectPlot(l9, response = "y")
```

16.19.6 Output & Results

`taguchiDesign()` returns the requested standard array as a `qualityTools` design object; `effectPlot()` renders the main-effects plot showing the average response at each level of each factor. Reading the plot — the factor whose level-mean line shows the steepest slope is most active — is the canonical first interpretation step in a Taguchi-style screening analysis.

16.19.7 Interpretation

A reporting sentence: "The L_9 orthogonal array screened four three-level factors in nine runs. The main-effects plot identified factors A and C as active (level means ranging across ~ 4 units of response), with optimal levels A_3 and C_1. Factors B and D showed nearly flat level-means and were inactive. The active subset will be carried forward into a full 3^2 factorial with replicated centre points to characterise curvature." Always justify "active" assignments by effect magnitude rather than p-values alone in OA screening.

16.19.8 Practical Tips

- Taguchi orthogonal arrays are a subset of the broader fractional-factorial framework; modern DoE practice often prefers `FrF2`, `DoE.base`, and `conf.design` constructions for greater flexibility, explicit alias documentation, and Lenth-style analysis tools.
- Interaction analysis from an OA requires consulting the array's linear-graph (a graphical representation of which column-pair maps to which interaction); without it, interactions are confounded silently with main effects.

- Mixed-level orthogonal arrays such as L_{18} and L_{36} accommodate combinations of two-level and three-level factors and are useful when one or two factors warrant a quadratic check while others are clearly two-level.
- Follow up an OA screening with a higher-resolution design on the small subset of active factors identified; the screen-then-characterise sequence is the canonical industrial-DoE workflow and applies to OAs as much as to standard fractional factorials.
- Custom-designed D-optimal designs from `AlgDesign::optFederov()` often beat off-the-shelf Taguchi arrays in efficiency, especially for unusual factor counts, mixed levels, or constrained design regions; consider them when the standard catalogue does not fit cleanly.
- The Taguchi philosophy distinguishes "control" factors from "noise" factors via inner and outer arrays; this is the basis of robust parameter design and is treated separately in dedicated robust-design tutorials.

16.19.9 R Packages Used

`qualityTools::taguchiDesign()` for standard Taguchi catalogue arrays and `effectPlot()` for main-effects visualisation; `DoE.base::oa.design()` for general orthogonal-array construction beyond the Taguchi catalogue; `FrF2` for fractional factorials including those equivalent to Taguchi arrays; `AlgDesign` for computer-generated D-optimal alternatives; `conf.design` for low-level construction of confounding patterns.

16.19.10 For Reviewers

Use the shared reviewer checklist, together with this method's assumptions and practical tips.

Method-specific reviewer notes have not yet been developed for this entry. The links above support a structured review; they do not certify the example or a manuscript's application of the method.

16.19.11 See also — labs in this chapter

- Observational designs and STROBE
- Randomised controlled trials
- Adaptive, non-inferiority, equivalence trials
- Bench and translational design
- MCAR, MAR, MNAR
- Multiple imputation with mice
- Linear mixed models with lme4
- GLMMs and GEE
- DAGs with dagitty and ggdag
- Propensity scores and IPTW
- G-methods, IV, DiD, RDD; heterogeneity of treatment effect
- SIR and SEIR models with deSolve
- Pre-registration and statistical analysis plans

16.20 Plackett-Burman Designs

16.20.1 Introduction

Plackett–Burman (PB) designs, introduced by Plackett and Burman in 1946, are two-level orthogonal screening designs with n runs (where n is a multiple of 4) that can accommodate up to $n - 1$ factors. They are widely regarded as the most economical screening tool in industrial DoE: when many candidate factors must be sifted to identify the few that drive the response, a 12-run PB design can screen up to 11 factors, a 20-run PB up to 19, and so on. The price for this economy is a resolution-III aliasing structure in which main effects are confounded with two-factor interactions, so PB designs are appropriate strictly for screening — never for confirmatory or interaction-aware inference.

16.20.2 Prerequisites

Familiarity with fractional factorial designs, the concept of resolution, the effect-sparsity principle, and the distinction between regular and non-regular fractional factorial constructions.

16.20.3 Theory

PB designs are built from Hadamard matrices, square matrices of ± 1 entries with mutually orthogonal columns. With n runs the design supplies $n - 1$ orthogonal contrast columns, each of which can carry one factor. For $n = 12$, 11 factors fit; for $n = 20$, 19 factors.

Non-regular PB designs (those whose run count is a multiple of 4 but not a power of 2, like 12, 20, 24) have a famously complex partial-aliasing structure: each main effect is aliased with multiple two-factor interactions at fractional weights, rather than fully confounded with a single one. This complex partial aliasing is a property the experimentalist must understand to interpret active-effect candidates correctly. Regular PB designs (n a power of 2) coincide with standard resolution-III fractional factorials and have cleaner alias chains.

16.20.4 Assumptions

Two-factor and higher-order interactions are negligible compared with main effects (the sparsity-of-effects principle), and follow-up experiments will be conducted on the screened-in subset to characterise interactions properly. PB designs are a tool of two-stage industrial DoE: screen first, characterise second.

16.20.5 R Implementation

```
library(FrF2)

pb ← pb(nruns = 12, nfactors = 8, randomize = TRUE, seed = 2026)
pb

for (v in paste0("A", 1:8)) {
  pb[[v]] ← as.numeric(as.character(pb[[v]]))
}
set.seed(2026)
pb$y ← 10 + 2.5 * pb$A1 + 1.8 * pb$A2 + rnorm(nrow(pb), 0, 0.6)

fit ← lm(y ~ ., data = pb)
summary(fit)
```

16.20.6 Output & Results

`pb()` constructs a randomised Plackett–Burman layout for the requested run count and factor count. The main-effects regression returns coefficients for each factor; combining the regression with a Lenth-style or half-Normal-plot identification of the active subset is the standard analytical approach for unreplicated PB designs.

16.20.7 Interpretation

A reporting sentence: "The 12-run Plackett–Burman design screened eight two-level factors, identifying A1 (effect 2.5, $p < 0.001$) and A2 (effect 1.8, $p < 0.001$) as the principal drivers of the response. The remaining six factors had effect estimates within Lenth's pseudo-standard-error envelope and were dropped from follow-up. A subsequent 2^2 full factorial with replicated centre points characterised curvature and the A1 × A2 interaction in the active subset." Always describe both active and inactive factors and the screening criterion used.

16.20.8 Practical Tips

- Plackett–Burman is a screening design; always follow it up with a higher-resolution fractional factorial, full factorial, or response-surface design on the active factor subset. Drawing causal conclusions directly from a PB result without follow-up is methodologically unsound.
- The 12-run design is the most popular member of the family because it balances economy (one-third the runs of a comparable resolution-IV fractional factorial) with the orthogonality and ease-of-interpretation of regular Hadamard-based construction.
- Complex partial aliasing in non-regular PB designs can occasionally reveal active two-factor interactions indirectly through effect-heredity logic, but this should be regarded as exploratory; formal interaction inference still requires a higher-resolution follow-up design.
- Always randomise run order to protect against temporal confounding; running treatments in design order conflates time-trend effects with factor effects in any unreplicated screening design.

- Report transparently which factors were screened in and out of the follow-up; the pruning decision is itself a substantive analytical choice and reviewers will want to see the criterion used.
- For mixed-level screening, definitive screening designs (DSDs, Jones and Nachtsheim 2011) generalise PB ideas to three-level factors and offer cleaner analysis of curvature alongside main-effect screening.

16.20.9 R Packages Used

`FrF2::pb()` for Plackett–Burman construction with explicit randomisation; `DoE.base::oa.design()` for general Hadamard-matrix-based orthogonal arrays; `daewr` for textbook companion examples and analysis helpers; `unrepx` for Lenth-style identification of active effects in unreplicated PB designs; `dsd` and `daewr::DefScreen()` for definitive screening designs as a modern alternative.

16.20.10 For Reviewers

Use the shared reviewer checklist, together with this method's assumptions and practical tips.

Method-specific reviewer notes have not yet been developed for this entry. The links above support a structured review; they do not certify the example or a manuscript's application of the method.

16.20.11 See also — labs in this chapter

- Observational designs and STROBE
- Randomised controlled trials
- Adaptive, non-inferiority, equivalence trials
- Bench and translational design
- MCAR, MAR, MNAR
- Multiple imputation with mice
- Linear mixed models with lme4
- GLMMs and GEE
- DAGs with dagitty and ggdag
- Propensity scores and IPTW
- G-methods, IV, DiD, RDD; heterogeneity of treatment effect
- SIR and SEIR models with deSolve
- Pre-registration and statistical analysis plans

16.21 Power Analysis for DOE

16.21.1 Introduction

Power analysis for designed experiments determines the run count required to detect specified main effects and interactions with adequate probability. It bridges experimental-design choice and statistical inference: a design is only useful if its run count gives

sensible power for the smallest effect that would be scientifically actionable. Skimping on power means under-powered experiments that fail to detect real effects and waste resources on inconclusive results; over-powering an experiment beyond what is needed wastes resources differently and exposes the design to detecting irrelevantly small effects. A pre-specified power calculation is therefore now standard in any DoE protocol — industrial, agricultural, or clinical — and is increasingly required by regulators and journal reviewers.

16.21.2 Prerequisites

A working understanding of effect sizes (Cohen's d, f, f^2), the structure of ANOVA F-tests, the non-central F-distribution, and the concepts of significance level and statistical power.

16.21.3 Theory

For a 2^k factorial design, the main-effect F-test on a single factor has non-centrality parameter

$$\lambda = \frac{n\Delta^2}{4\sigma^2},$$

where n is the total number of runs, Δ is the effect (difference between high- and low-level means), and σ^2 is the residual variance. Power is computed as $P(F > F_{\alpha,1,\mathrm{df}} \mid \lambda)$ under the non-central F-distribution. Interactions typically have effect sizes 30–50 % of the corresponding main effects, so designs powered adequately for main effects routinely lack power for two-factor interactions and for higher-order terms.

16.21.4 Assumptions

The residual variance is known or pre-specified, the effect size of interest is fixed in advance, the design and analysis model are pre-specified, and the residuals satisfy the standard ANOVA assumptions of independence, Normality, and constant variance.

16.21.5 R Implementation

```
library(pwr)

pwr.t.test(d = 1, n = 8, sig.level = 0.05,
           type = "two.sample", alternative = "two.sided")

set.seed(2026)
n_sim ← 500
reps ← 2
power_sim ← mean(replicate(n_sim, {
  A ← rep(c(-1, 1), each = 4 * reps)
  B ← rep(rep(c(-1, 1), each = 2 * reps), 2)
  C ← rep(c(-1, 1), times = 4 * reps)
  y ← 1 * A + 0.5 * B + rnorm(8 * reps, 0, 1.5)
  pA ← summary(lm(y ~ A + B + C))$coefficients["A", "Pr(>|t|)"]
  pA < 0.05
}))
power_sim
```

16.21.6 Output & Results

The analytical power calculation for a two-sample t-test reports power for a single main effect under classical Cohen's d assumptions; the simulation loop reports empirical power under the actual analysis model and noise structure. The two should agree to within Monte Carlo error when assumptions match, and they often do not — the simulation approach generalises naturally to mixed-effects models, generalised linear models, and non-Normal residuals where no analytical formula exists.

16.21.7 Interpretation

A reporting sentence: "Pre-specified power analysis for the 2^3 factorial with two replicates ($n = 16$ total runs) achieved 82 % power to detect a 1.0 SD main effect at $\alpha = 0.05$, exceeding the 80 % minimum target. Power for a 0.5 SD two-factor interaction was only 28 %, so the experiment was not powered for interaction detection; if interactions are scientifically important, a third replicate ($n = 24$) raises interaction power to roughly 50 %, and a 2^4 design with one replicate would achieve 70 %." Always report power separately for main effects and interactions.

16.21.8 Practical Tips

- Always power the experiment for the smallest effect of scientific interest, not the largest expected effect; under-powering for small effects is the most common DoE error in published industrial work.
- Two-factor interactions typically have effect sizes 30–50 % of main effects; if interactions are part of the inferential question, scale up replication or run count accordingly. Designs powered for main effects routinely have very poor interaction power.
- Replicate the design (rather than expanding the factor space) to increase power for fixed factor structure; centre points add pure-error df without expanding the design region.

- Use simulation rather than analytical formulas when the analysis model deviates from classical ANOVA — mixed-effects models, generalised linear models, non-Normal residuals, and complex covariance structures are all best handled by Monte Carlo as above.
- Pre-specify the power calculation in the experimental protocol with explicit assumed values for σ, the smallest meaningful effect, and the analysis model; post-hoc power calculations after the experiment are statistically invalid and methodologically frowned upon.
- When prior data on σ are unavailable, conduct a pilot experiment first; treating an arbitrary or wishful σ as known invalidates the whole calculation.

16.21.9 R Packages Used

`pwr` for analytical power on t-tests, F-tests, ANOVA, and correlations; `Superpower` for ANOVA power simulation across complex factorial designs; `simr` for power analysis of mixed-effects models via simulation; `WebPower` for web-style power tools and complex designs; `daewr` for textbook DoE power examples.

16.21.10 For Reviewers

Use the shared reviewer checklist, together with this method's assumptions and practical tips.

Method-specific reviewer notes have not yet been developed for this entry. The links above support a structured review; they do not certify the example or a manuscript's application of the method.

16.21.11 See also — labs in this chapter

- Observational designs and STROBE
- Randomised controlled trials
- Adaptive, non-inferiority, equivalence trials
- Bench and translational design
- MCAR, MAR, MNAR
- Multiple imputation with mice
- Linear mixed models with lme4
- GLMMs and GEE
- DAGs with dagitty and ggdag
- Propensity scores and IPTW
- G-methods, IV, DiD, RDD; heterogeneity of treatment effect
- SIR and SEIR models with deSolve
- Pre-registration and statistical analysis plans

16.22 Randomised Complete Block Design

16.22.1 Introduction

A randomised complete block design (RCBD) is one of the most useful experimental designs in applied research. When a known source of variation – day of measurement, animal cage, plate, subject – is likely to influence the outcome, RCBD groups experimental units into homogeneous blocks and randomises treatments *within* each block. The block effect is thereby separated from the treatment effect, and the residual error is the within-block variability, which is typically much smaller than the pooled unblocked error.

16.22.2 Prerequisites

The reader should have met the one-way ANOVA and the principle of randomisation. Familiarity with R's formula syntax is assumed.

16.22.3 Theory

Let y_{ij} be the response of the experimental unit receiving treatment i in block j, for $i = 1, \ldots, t$ treatments and $j = 1, \ldots, b$ blocks. The additive RCBD model is

$$y_{ij} = \mu + \tau_i + \beta_j + \varepsilon_{ij}, \qquad \varepsilon_{ij} \overset{iid}{\sim} \mathcal{N}(0, \sigma^2),$$

where τ_i is the treatment effect (summing to zero) and β_j is the block effect. Each treatment appears exactly once in each block – that is the "complete" in RCBD.

The ANOVA decomposes the total sum of squares into treatment, block, and residual components. The F-test for treatments has $t - 1$ numerator and $(t - 1)(b - 1)$ denominator degrees of freedom. If blocking has captured real variation, the residual mean square is smaller than it would have been in a completely randomised design, and the test has higher power.

The design assumes **no treatment-by-block interaction**. If an interaction exists (a treatment works better in some blocks than others), the additive model is misspecified. The Tukey one-degree-of-freedom test for non-additivity provides a partial check.

16.22.4 Assumptions

1. **Additivity**: no treatment-by-block interaction.
2. **Independence**: residuals are mutually independent.
3. **Homoscedasticity**: residuals have constant variance across treatments and blocks.
4. **Normality**: residuals are approximately normal.

16.22.5 R Implementation

```
library(agricolae)
library(emmeans)

treatments ← LETTERS[1:4]
blocks ← paste0("B", 1:5)

set.seed(2026)
design ← design.rcbd(trt = treatments, r = length(blocks),
                     seed = 2026)
print(design$book)

design$book$y ← with(design$book,
                     5 +
                       as.numeric(factor(block)) * 0.8 +
                       c(0, 1.2, 1.8, 0.4)[as.numeric(factor(treatments))] +
                       rnorm(nrow(design$book), 0, 0.6))

fit ← aov(y ~ block + treatments, data = design$book)
summary(fit)

emm ← emmeans(fit, ~ treatments)
pairs(emm, adjust = "tukey")
```

`design.rcbd()` produces a randomised plan with each treatment appearing once per block. The simulated response includes a block effect and a treatment effect with the lowest for A, highest for C. `aov()` fits the additive RCBD model; `emmeans::pairs()` performs pairwise comparisons with Tukey adjustment.

16.22.6 Output & Results

The ANOVA table:

```
            Df Sum Sq Mean Sq F value  Pr(>F)
block        4 18.120   4.530  12.815 0.00015 ***
treatments   3  9.452   3.151   8.916 0.00142 **
Residuals   12  4.242   0.353
```

The block effect accounts for a substantial fraction of the variability – confirming that blocking was worthwhile – and the treatment effect is significant at $p = 0.0014$. Pairwise comparisons show C significantly higher than A ($p < 0.001$) and B ($p = 0.03$), and D not significantly different from A.

16.22.7 Interpretation

For a manuscript: “Treatments were compared in a randomised complete block design with four treatments and five blocks. An additive ANOVA showed a significant treatment effect, $F(3, 12) = 8.92$, $p = 0.001$, after accounting for block variation (which itself was substantial, $F(4, 12) = 12.82$, $p < 0.001$). Tukey-adjusted pairwise comparisons indicated that treatment C produced significantly higher responses than treatments A and B (see Table X).”

16.22.8 Practical Tips

- Block on variables that you expect to influence the outcome and that you cannot randomise out. Common choices: day of measurement, microtitre plate, cage, field plot, subject.
- Do not include the block as a "predictor of interest" – its role is to remove nuisance variation. Reporting its effect is informative but rarely the headline.
- If you suspect treatment-by-block interaction, switch to a design that models it: a randomised complete block with replicates within blocks, or a factorial design with block as a factor.
- When the number of treatments exceeds what fits in a block (e.g., 8 treatments but only 4 samples per plate), use an incomplete block design (BIBD).
- Always include randomisation within each block – the "R" in RCBD is essential for validity, even though the formula structure looks the same.

16.22.9 For Reviewers

Use the shared reviewer checklist, together with this method's assumptions and practical tips.

Method-specific reviewer notes have not yet been developed for this entry. The links above support a structured review; they do not certify the example or a manuscript's application of the method.

16.22.10 See also — labs in this chapter

- Observational designs and STROBE
- Randomised controlled trials
- Adaptive, non-inferiority, equivalence trials
- Bench and translational design
- MCAR, MAR, MNAR
- Multiple imputation with mice
- Linear mixed models with lme4
- GLMMs and GEE
- DAGs with dagitty and ggdag
- Propensity scores and IPTW
- G-methods, IV, DiD, RDD; heterogeneity of treatment effect
- SIR and SEIR models with deSolve
- Pre-registration and statistical analysis plans

16.23 Randomisation in Design

16.23.1 Introduction

Randomisation is the protective heart of experimental design. It ensures that in expectation, unknown sources of variation are balanced across treatments; it validates the model's assumption of independent errors; and it prevents systematic bias from lurking

variables.

16.23.2 Prerequisites

Experimental design basics.

16.23.3 Theory

Three Fisher principles of DoE: replication, randomisation, blocking. Randomisation: - Ensures **expected balance** of known and unknown covariates. - Justifies the random-error model used in ANOVA. - Validates inference (significance tests are approximate randomisation tests).

Randomisation procedures vary by design: CRD = unrestricted, RCBD = within-block, Latin square = row-column, split-plot = nested.

16.23.4 Assumptions

Randomisation is genuinely random (computer-generated sequence, sealed envelopes); allocation is concealed from the experimenter until assignment.

16.23.5 R Implementation

```r
set.seed(2026)

# Randomise 12 units to 4 treatments for a CRD
treatments ← rep(c("A", "B", "C", "D"), 3)
allocation ← sample(treatments, 12, replace = FALSE)
allocation

# For a RCBD, randomise within each of 3 blocks
blocks ← rep(1:3, each = 4)
alloc_rcbd ← do.call(rbind, lapply(split(1:12, blocks), function(idx) {
  data.frame(unit = idx, treatment = sample(c("A", "B", "C", "D")))
}))
alloc_rcbd
```

16.23.6 Output & Results

A randomised allocation list; randomisation is within block for RCBD to preserve block structure.

16.23.7 Interpretation

“Experimental units were randomly allocated to 4 treatments using a computer-generated sequence (seed 2026); in the RCBD, randomisation was within each block to preserve the local-control benefit.”

16.23.8 Practical Tips

- Document the randomisation seed and procedure; reproducibility is part of responsible reporting.
- Conceal allocation from experimenters until assignment; otherwise bias can creep in.
- Randomise run order as well as treatment allocation to guard against temporal confounding.
- Restricted randomisation (within blocks, strata) is required for blocked designs; unrestricted randomisation can mismatch the analysis.
- Randomisation tests (permutation inference) are a rigorous alternative to parametric tests when distributional assumptions fail.

16.23.9 For Reviewers

Use the shared reviewer checklist, together with this method's assumptions and practical tips.

Method-specific reviewer notes have not yet been developed for this entry. The links above support a structured review; they do not certify the example or a manuscript's application of the method.

16.23.10 See also — labs in this chapter

- Observational designs and STROBE
- Randomised controlled trials
- Adaptive, non-inferiority, equivalence trials
- Bench and translational design
- MCAR, MAR, MNAR
- Multiple imputation with mice
- Linear mixed models with lme4
- GLMMs and GEE
- DAGs with dagitty and ggdag
- Propensity scores and IPTW
- G-methods, IV, DiD, RDD; heterogeneity of treatment effect
- SIR and SEIR models with deSolve
- Pre-registration and statistical analysis plans

16.24 Repeated-Measures Designs

16.24.1 Introduction

Repeated-measures designs measure the same experimental unit at multiple time points or under multiple conditions. They gain statistical power relative to between-subject designs by removing the between-subject component of variance from the within-subject contrasts of interest, but they introduce a within-subject correlation structure that must be modelled correctly. The classical repeated-measures ANOVA requires sphericity and is

restricted to balanced complete designs; the modern alternative is mixed-effects modelling with subject as a random effect, which accommodates unbalanced data, missing time points, and richer covariance structures while remaining valid under broader assumptions.

16.24.2 Prerequisites

A working understanding of one-way and factorial ANOVA, the concept of correlated data, and the essentials of mixed-effects modelling with random intercepts and (optionally) random slopes.

16.24.3 Theory

The standard mixed-effects model for repeated measures is

$$y_{ij} = \mu + \tau_j + s_i + \varepsilon_{ij}, \qquad s_i \sim N(0, \sigma_s^2), \quad \varepsilon_{ij} \sim N(0, \sigma^2),$$

with within-subject factor effect τ_j, subject random intercept s_i, and residual error ε_{ij}. Common covariance structures imposed on the residual vector for one subject include:

- **Compound symmetry**: equal residual variance and equal correlation across all time pairs (strict — implied by a random-intercept model).
- **AR(1)**: autoregressive of order one, with correlation decaying with lag.
- **Unstructured**: fully general $T \times T$ covariance with $T(T+1)/2$ parameters.

For classical repeated-measures ANOVA, sphericity is the relevant assumption, and Greenhouse–Geisser or Huynh–Feldt corrections are applied when Mauchly's test detects a violation.

16.24.4 Assumptions

The within-subject correlation structure is correctly specified, residuals are approximately Normal, and missing data are missing at random (MAR) — a standard assumption that mixed-effects models can accommodate without listwise deletion.

16.24.5 R Implementation

```
library(lme4); library(lmerTest)

set.seed(2026)
n_sub ← 30; n_time ← 4
sub_re ← rnorm(n_sub, 0, 1)

df ← expand.grid(subject = 1:n_sub, time = 1:n_time)
df$y ← 10 + 0.5 * df$time + sub_re[df$subject] +
        rnorm(nrow(df), 0, 0.5)

fit_ri ← lmer(y ~ factor(time) + (1 | subject), data = df)
anova(fit_ri)

fit_rs ← lmer(y ~ time + (time | subject), data = df)
summary(fit_rs)$coefficients
```

16.24.6 Output & Results

`lmer()` with a random intercept gives the within-subject contrast for time, and `anova()` from `lmerTest` provides F-tests with Satterthwaite-adjusted denominator df. Adding a random slope (`(time | subject)`) lets the within-subject trend itself vary across subjects, which is appropriate when there is reason to believe individuals differ in their response trajectories.

16.24.7 Interpretation

A reporting sentence: "Repeated-measures analysis with subject as a random intercept showed a significant time effect ($F_{3,87} = 18.2$, $p < 0.001$). The subject random effect absorbed substantial between-subject variance ($\sigma^2_s = 1.04$, $\sigma^2 = 0.27$, intra-class correlation 0.79), markedly improving precision relative to a between-subject analysis. A random-slope model that allowed subject-specific time trends did not improve fit ($\Delta\text{AIC} = +1.4$), so the random-intercept specification is reported as primary." Always report variance components.

16.24.8 Practical Tips

- Use mixed-effects models (`lmer` from `lme4` , `lme` from `nlme` , or generalised-mixed via `glmmTMB`) rather than classical repeated-measures ANOVA whenever possible; they handle unbalanced data, missing time points, irregular timing, and richer covariance structures naturally.
- For irregular or continuous timing, specify time as a continuous covariate with random slopes; this is far more efficient than treating each time point as a separate factor level when the response is genuinely a smooth function of time.
- If you must use classical repeated-measures ANOVA (e.g., for compatibility with a standard reporting requirement), check sphericity with Mauchly's test and apply Greenhouse–Geisser or Huynh–Feldt correction whenever it is violated.
- Missing time points are handled naturally by mixed-effects models under the MAR assumption; classical repeated-measures ANOVA requires complete cases and discards subjects with any missing time, which is wasteful and can introduce bias.
- Report both the fixed-effect estimates (the substantive scientific output) and the random-effect variance components (which describe how much within-subject correlation is present); the latter is essential for judging whether the design is well chosen.
- For binary or count outcomes measured repeatedly, use generalised mixed-effects models (`glmer` , `glmmTMB`) or generalised estimating equations (GEE via the `geepack` package) — the latter trade efficiency for robust marginal inference.

16.24.9 R Packages Used

`lme4::lmer()` and `lmerTest` for mixed-effects analysis with denominator-df adjustment; `nlme::lme()` for richer covariance-structure specification (AR1, unstructured) on the residuals; `glmmTMB` for generalised and complex-covariance mixed models;

`geepack::geeglm()` for population-averaged GEE analyses; `afex::aov_car()` for classical repeated-measures ANOVA with built-in sphericity correction.

16.24.10 For Reviewers

Use the shared reviewer checklist, together with this method's assumptions and practical tips.

Method-specific reviewer notes have not yet been developed for this entry. The links above support a structured review; they do not certify the example or a manuscript's application of the method.

16.24.11 See also — labs in this chapter

- Observational designs and STROBE
- Randomised controlled trials
- Adaptive, non-inferiority, equivalence trials
- Bench and translational design
- MCAR, MAR, MNAR
- Multiple imputation with mice
- Linear mixed models with lme4
- GLMMs and GEE
- DAGs with dagitty and ggdag
- Propensity scores and IPTW
- G-methods, IV, DiD, RDD; heterogeneity of treatment effect
- SIR and SEIR models with deSolve
- Pre-registration and statistical analysis plans

16.25 Resolution and Aliasing

16.25.1 Introduction

In any fractional factorial design, effects are aliased — two or more distinct effects produce the same contrast in the data and cannot be disentangled from the experimental results alone. The choice of fraction determines exactly which effects are aliased with which, and the design's **resolution** summarises the severity of this aliasing in a single Roman numeral. Higher resolution means lower-order effects are protected from aliasing with each other; lower resolution means smaller designs but messier confounding. Understanding resolution and the alias chain is essential for interpreting any fractional-factorial result, because an unrecognised alias is one of the most common sources of misleading DoE conclusions.

16.25.2 Prerequisites

Familiarity with fractional factorial designs, the algebra of defining relations and word lengths, and the effect-sparsity principle that motivates accepting higher-order aliasing.

16.25.3 Theory

A fractional factorial is defined by one or more **defining relations** of the form $I = ABCD\cdots$, where each word represents a product of factor letters. The defining relation determines the alias structure: multiplying any effect by the defining word (modulo squaring out $A^2 = I$) produces its alias. For example, with $I = ABCD$, we have $A = BCD$, $AB = CD$, and so on. The **resolution** of the design is the length of the shortest word in the defining relation:

- **Resolution III**: the shortest defining word has length 3, so main effects are aliased with two-factor interactions ($A = BC$). Suitable for screening only.
- **Resolution IV**: shortest word length 4; main effects are clear of two-factor interactions, but two-factor interactions alias with each other ($AB = CD$).
- **Resolution V**: shortest word length 5; main effects and all two-factor interactions are mutually clear; only three-factor and higher interactions are aliased with two-factor ones.

16.25.4 Assumptions

The effect-sparsity principle holds — higher-order interactions are negligible compared with lower-order effects, so accepting aliasing among higher-order terms is acceptable. The chosen resolution is appropriate for the scientific question, and the alias chain is documented in the analysis.

16.25.5 R Implementation

```r
library(FrF2)

res3 ← FrF2(nruns = 8, nfactors = 7)
design.info(res3)$aliased

res4 ← FrF2(nruns = 32, nfactors = 7, resolution = 4)
design.info(res4)$aliased

res6 ← FrF2(nruns = 32, nfactors = 6, resolution = 6)
design.info(res6)$aliased
```

16.25.6 Output & Results

`design.info()` reports the complete alias structure of the design — the defining relation, the resolution, and the chain of effects aliased together. Inspecting this output before analysis is essential: it tells the analyst exactly which estimates can be interpreted as pure main or interaction effects and which represent the joint estimate of an aliased pair.

16.25.7 Interpretation

A reporting sentence: "The 2^{7-4}_{III} design used 8 runs to screen seven two-level factors, with main effects aliased with two-factor interactions; the design is appropriate when only main effects are expected to matter and would not be appropriate if interaction effects were of interest. The 2^{7-2}_{IV} design used 32 runs to study the same factor set with main

effects clear of two-factor interactions, at the cost of four-fold more experimentation. The complete alias chain is reported in Supplementary Table S1." Always document resolution and alias chain.

16.25.8 Practical Tips

- Resolution III designs (and Plackett–Burman designs at run counts not a power of 2) are appropriate for early screening when interactions are confidently negligible; never rely on a resolution-III design for inference about interactions or for confirmatory analysis.
- Resolution IV is the standard choice for initial investigation of many factors when the experimentalist wants clean main effects without paying the run-count cost of resolution V.
- Resolution V and above are appropriate for fitting models that include two-factor interactions; many industrial DoE protocols specify resolution V as the minimum for response-surface follow-up designs.
- Fold-over augmentation — adding a mirror-image fraction with all factor signs reversed — upgrades a resolution III design to resolution IV by de-aliasing main effects from two-factor interactions, at the cost of doubling the run count.
- Always document the alias structure explicitly in reports and protocols; reviewers and downstream analysts cannot interpret the results correctly without knowing which effects can be separated from which.
- For mixed-level fractional designs and irregular run counts, software-generated D-optimal designs may achieve cleaner partial aliasing than the standard regular fractions; consider them when the standard catalogue does not match your factor structure.

16.25.9 R Packages Used

`FrF2::FrF2()` and `FrF2::design.info()` for fractional factorial construction with explicit resolution control and alias-structure documentation; `DoE.base` for general orthogonal-array DoE with comprehensive alias reporting; `unrepx` for unreplicated-design effect identification under aliased structures; `BsMD` for Bayesian screening of fractional designs; `daewr` for textbook companion datasets.

16.25.10 For Reviewers

Use the shared reviewer checklist, together with this method's assumptions and practical tips.

Method-specific reviewer notes have not yet been developed for this entry. The links above support a structured review; they do not certify the example or a manuscript's application of the method.

16.25.11 See also — labs in this chapter

- Observational designs and STROBE

- Randomised controlled trials
- Adaptive, non-inferiority, equivalence trials
- Bench and translational design
- MCAR, MAR, MNAR
- Multiple imputation with mice
- Linear mixed models with lme4
- GLMMs and GEE
- DAGs with dagitty and ggdag
- Propensity scores and IPTW
- G-methods, IV, DiD, RDD; heterogeneity of treatment effect
- SIR and SEIR models with deSolve
- Pre-registration and statistical analysis plans

16.26 Response Surface Methodology

16.26.1 Introduction

Response surface methodology (RSM), introduced by Box and Wilson in 1951, is a sequential experimental strategy for finding and characterising the optimum of a process response in the space of multiple controllable factors. RSM combines fractional or full factorial screening, steepest-ascent search, and second-order polynomial fitting near the optimum into a coherent workflow that has become standard practice in chemical engineering, food science, pharmaceutical formulation, manufacturing process improvement, and any setting where a smooth response surface must be navigated efficiently. Its hallmark is the staged, sequential character: each stage of experimentation informs the next, and only the final stage commits resources to the dense second-order designs needed for full characterisation.

16.26.2 Prerequisites

A working understanding of factorial designs, polynomial regression for response-surface modelling, gradient-based optimisation concepts, and the canonical form of a quadratic surface.

16.26.3 Theory

RSM operates in two main phases:

1. **First-order phase**: a fractional or full factorial with replicated centre points fits a linear model in a restricted operating region. The path of steepest ascent (or descent) along the fitted gradient guides further experimentation toward the optimum, with confirmation runs along the path.
2. **Second-order phase**: once experimentation is near the optimum, a central composite or Box-Behnken design fits a full second-order polynomial. Canonical analysis decomposes the quadratic-form matrix into eigenvalues, classifying the stationary point as a maximum, minimum, saddle, or rising/stationary ridge.

The canonical form is what makes RSM more than just curve-fitting: it characterises the geometry of the response surface and reveals when the optimum is genuinely a peak, when it lies on a ridge requiring constrained search, and when further experimentation is needed.

16.26.4 Assumptions

The response is a smooth, well-defined function of the factors over the design region; a second-order polynomial is an adequate local approximation in the neighbourhood of the optimum; residual error is Normal with constant variance; runs are independent.

16.26.5 R Implementation

```
library(rsm)

set.seed(2026)
ccd ← ccd(~ x1 + x2, n0 = c(3, 3), alpha = "rotatable",
          randomize = FALSE)
ccd$y ← 10 + 2 * ccd$x1 + 1.5 * ccd$x2 -
         1.2 * ccd$x1^2 - 0.9 * ccd$x2^2 +
         0.6 * ccd$x1 * ccd$x2 +
         rnorm(nrow(ccd), 0, 0.4)

fit ← rsm(y ~ SO(x1, x2), data = ccd)
summary(fit)

contour(fit, ~ x1 + x2, at = c(0, 0), image = TRUE,
        main = "RSM response surface")

canonical(fit)
```

16.26.6 Output & Results

The `rsm()` summary returns linear, quadratic, and interaction coefficients with significance tests, plus the canonical-analysis classification (maximum, minimum, saddle, or ridge) based on the eigenvalues of the quadratic-form matrix. `contour()` and `persp()` render the surface visually, and `canonical()` reports the stationary point in original and canonical coordinates.

16.26.7 Interpretation

A reporting sentence: “Response-surface methodology with a 14-run rotatable CCD identified a concave surface with stationary point at $(x_1^*, x_2^*) = (0.85, 0.72)$. Canonical analysis confirmed a true maximum (both eigenvalues negative: $\lambda_1 = -1.31$, $\lambda_2 = -0.79$); predicted peak response 12.1, with a 95 % prediction interval of [11.5, 12.7]. Confirmation runs at the predicted optimum yielded 12.0 (mean of 3 replicates), within the interval and supporting the model.” Always validate the predicted optimum with confirmation runs.

16.26.8 Practical Tips

- The first-order steepest-ascent phase can save many runs; don't jump straight to a second-order fit if the operating region is far from the optimum, because a quadratic fit far from the peak yields meaningless coefficients and wasted runs.
- Always inspect contour and 3D surface plots in addition to the coefficient table; numerical summaries can hide ridges, saddles, and other surface features that are obvious visually.
- Canonical analysis reveals the nature of the stationary point through the signs of the eigenvalues: all negative = maximum, all positive = minimum, mixed signs = saddle, near-zero = ridge requiring constrained or directional search.
- When the predicted optimum lies outside the experimental design region, use ridge analysis (constrained optimisation along the path of steepest ascent within the cube) rather than naive extrapolation, which is dangerous and statistically unsupported.
- Follow RSM with confirmation runs at the predicted optimum and inside the prediction interval; falsified predictions reveal model inadequacy and motivate a higher-order fit or a re-positioned design region.
- For multiple responses (a common situation in industrial work — yield, purity, cost), use desirability functions to combine the surfaces into a single optimisation criterion rather than optimising each response independently.

16.26.9 R Packages Used

`rsm` for canonical RSM construction and analysis (CCD, BBD, second-order fits, canonical analysis, steepest-ascent paths, contour and 3D plots); `desirability` for multi-response desirability-function optimisation; `qualityTools` for an alternative RSM toolkit; `Vdgraph` for variance-dispersion graphs comparing RSM design alternatives; `daewr` for textbook companion examples and `DoE.wrapper` for higher-level design-workflow integration.

16.26.10 For Reviewers

Use the shared reviewer checklist, together with this method's assumptions and practical tips.

Method-specific reviewer notes have not yet been developed for this entry. The links above support a structured review; they do not certify the example or a manuscript's application of the method.

16.26.11 See also — labs in this chapter

- Observational designs and STROBE
- Randomised controlled trials
- Adaptive, non-inferiority, equivalence trials
- Bench and translational design
- MCAR, MAR, MNAR
- Multiple imputation with mice
- Linear mixed models with lme4
- GLMMs and GEE

- DAGs with dagitty and ggdag
- Propensity scores and IPTW
- G-methods, IV, DiD, RDD; heterogeneity of treatment effect
- SIR and SEIR models with deSolve
- Pre-registration and statistical analysis plans

16.27 Robust Parameter Design

16.27.1 Introduction

Robust parameter design (Taguchi, 1950s-60s) seeks combinations of controllable design factors that minimise response variability across noise factors (environmental, batch-to-batch, use-conditions). The goal is a product or process that performs consistently regardless of noise.

16.27.2 Prerequisites

Factorial designs; variance decomposition.

16.27.3 Theory

Inner array of control factors crossed with **outer array** of noise factors. At each control setting, compute a summary – mean, standard deviation, or signal-to-noise ratio. Choose the control setting that optimises the summary.

Modern approach: fit a response model with control, noise, and control-x-noise interactions; find control levels that minimise variance while meeting the target mean.

16.27.4 Assumptions

Noise factors represent realistic variation at use time; interactions with control factors are the primary lever for variance reduction.

16.27.5 R Implementation

```
library(FrF2)

set.seed(2026)
# Inner array: 2^3 controllable factors
ctrl ← FrF2(nruns = 8, nfactors = 3, randomize = FALSE)
ctrl$A ← as.numeric(as.character(ctrl$A))
ctrl$B ← as.numeric(as.character(ctrl$B))
ctrl$C ← as.numeric(as.character(ctrl$C))

# Outer array: 2^2 noise factors
noise ← expand.grid(N1 = c(-1, 1), N2 = c(-1, 1))

# Simulate: noise x control interaction
results ← do.call(rbind, lapply(1:nrow(ctrl), function(i) {
  vals ← sapply(1:nrow(noise), function(j) {
    10 + 2 * ctrl$A[i] + 1.5 * ctrl$B[i] +
      0.5 * noise$N1[j] + 0.8 * ctrl$A[i] * noise$N1[j] +
      rnorm(1, 0, 0.3)
  })
  data.frame(run = i, mean = mean(vals), sd = sd(vals),
             snr = -10 * log10(var(vals)))
}))
results

# Choose control level minimising variance
results$A ← ctrl$A
cbind(ctrl, results)[order(results$sd), ]
```

16.27.6 Output & Results

Per-control-run mean, SD, and signal-to-noise ratio; the control combination with the smallest SD is the robust-optimal.

16.27.7 Interpretation

“Robust parameter design identified setting (A = -1, B = +1, C = +1) as minimising response SD across noise factors. Control factor A interacted with noise factor N1, explaining the variance reduction at A = -1.”

16.27.8 Practical Tips

- The inner-outer array approach can be computationally expensive; use a combined model with control-noise interactions when possible.
- Report both mean and variance; robust designs often accept a small mean-response penalty for large variance reduction.
- Consider dual-response surface methods for joint mean-variance optimisation.
- Pay attention to control-noise interactions: that is where robustness lives.
- Combining Taguchi’s philosophy with modern RSM is more flexible than pure Taguchi methods.

16.27.9 For Reviewers

Use the shared reviewer checklist, together with this method's assumptions and practical tips.

Method-specific reviewer notes have not yet been developed for this entry. The links above support a structured review; they do not certify the example or a manuscript's application of the method.

16.27.10 See also — labs in this chapter

- Observational designs and STROBE
- Randomised controlled trials
- Adaptive, non-inferiority, equivalence trials
- Bench and translational design
- MCAR, MAR, MNAR
- Multiple imputation with mice
- Linear mixed models with lme4
- GLMMs and GEE
- DAGs with dagitty and ggdag
- Propensity scores and IPTW
- G-methods, IV, DiD, RDD; heterogeneity of treatment effect
- SIR and SEIR models with deSolve
- Pre-registration and statistical analysis plans

16.28 Signal-to-Noise Ratios

16.28.1 Introduction

Signal-to-noise (SNR) ratios in Taguchi's framework jointly summarise the mean and variance of a response. Three variants handle different quality targets: smaller-is-better, larger-is-better, and nominal-is-best. Maximising SNR selects factor combinations that produce both a desirable mean and low variance.

16.28.2 Prerequisites

Taguchi methods; variance and mean summaries.

16.28.3 Theory

- **Smaller-is-better** (minimisation): $\mathrm{SNR} = -10 \log_{10}\left(\frac{1}{n}\sum y_i^2\right)$.
- **Larger-is-better** (maximisation): $\mathrm{SNR} = -10 \log_{10}\left(\frac{1}{n}\sum 1/y_i^2\right)$.
- **Nominal-is-best** (hit target τ): $\mathrm{SNR} = 10 \log_{10}(\bar{y}^2/s^2)$.

Higher SNR is always better. Bias: SNRs conflate mean and variance; variance reduction can increase SNR even when mean drifts.

16.28.4 Assumptions

Outcomes are non-negative (for smaller/larger-is-better); target is appropriate; replicates exist to estimate variance.

16.28.5 R Implementation

```r
set.seed(2026)

# Smaller-is-better: defects per batch
defects <- matrix(rpois(20 * 3, lambda = 4), 20, 3)
snr_sb <- -10 * log10(rowMeans(defects^2))

# Larger-is-better: process yield
yield <- matrix(rnorm(20 * 3, 80, 5), 20, 3)
snr_lb <- -10 * log10(rowMeans(1 / yield^2))

# Nominal-is-best: target 50
values <- matrix(rnorm(20 * 3, 50, 3), 20, 3)
means <- rowMeans(values); sds <- apply(values, 1, sd)
snr_nom <- 10 * log10(means^2 / sds^2)

head(data.frame(snr_sb, snr_lb, snr_nom))
```

16.28.6 Output & Results

Per-batch SNR values under the three variants; the batch maximising each SNR has the desirable combination of mean and variance.

16.28.7 Interpretation

“Smaller-is-better SNR was -12.5 (batch 3) vs -14.7 (batch 10); batch 3 achieved lower mean and lower variance of defects. The larger-is-better SNR identified a different optimal batch, suggesting SNRs depend on target.”

16.28.8 Practical Tips

- SNR combines mean and variance – convenient but conflates them. Modern practice reports mean and variance separately.
- Dual-response surface methods (Myers-Carter, Vining-Myers) extend SNR thinking by fitting separate mean and variance models.
- For nominal-is-best, ensure $\bar{y}$ hits target before interpreting SNR.
- SNR is sensitive to sample size; fewer replicates inflate or deflate the metric.
- Report SNR in dB (decibels, i.e. using $10 \log_{10}$), to follow Taguchi conventions.

16.28.9 For Reviewers

Use the shared reviewer checklist, together with this method’s assumptions and practical tips.

Method-specific reviewer notes have not yet been developed for this entry. The links above support a structured review; they do not certify the example or a manuscript's application of the method.

16.28.10 See also — labs in this chapter

- Observational designs and STROBE
- Randomised controlled trials
- Adaptive, non-inferiority, equivalence trials
- Bench and translational design
- MCAR, MAR, MNAR
- Multiple imputation with mice
- Linear mixed models with lme4
- GLMMs and GEE
- DAGs with dagitty and ggdag
- Propensity scores and IPTW
- G-methods, IV, DiD, RDD; heterogeneity of treatment effect
- SIR and SEIR models with deSolve
- Pre-registration and statistical analysis plans

16.29 Split-Plot Designs

16.29.1 Introduction

Split-plot designs arise whenever some experimental factors are hard to change between runs while others are easy to change within a run. Oven temperature, pasture treatment, baking time, batch-level chemical additive, or instrument calibration are typically hard to change — they can be set once per batch but not freely between samples. By contrast, additive level within a batch, fertiliser sub-plot within a paddock, or sample-level annotation can be randomised within the larger group. Restricted randomisation of the hard-to-change factor at the whole-plot level and unrestricted randomisation of the easy-to-change factor within whole plots produces a design with two error strata, each requiring its own appropriate F-test denominator. Recognising the split-plot structure rather than ignoring it is essential to valid inference.

16.29.2 Prerequisites

A working understanding of factorial designs, mixed-effects models with crossed and nested random effects, and the concept of multiple error strata in restricted-randomisation experiments.

16.29.3 Theory

The standard split-plot model is

$$y_{ijk} = \mu + \alpha_i + \delta_{ij} + \beta_k + (\alpha\beta)_{ik} + \varepsilon_{ijk},$$

with whole-plot effect α_i, whole-plot error $\delta_{ij} \sim N(0, \sigma_W^2)$, sub-plot effect β_k, interaction $(\alpha\beta)_{ik}$, and sub-plot error $\varepsilon_{ijk} \sim N(0, \sigma_S^2)$. The whole-plot factor α is tested against the whole-plot error, while the sub-plot factor β and the interaction $\alpha\beta$ are tested against the (typically smaller) sub-plot error. As a result, the sub-plot factor and the interaction usually have more power than the whole-plot factor — a structural asymmetry that should inform which factor is assigned to which stratum.

16.29.4 Assumptions

Whole-plot errors are independent across whole plots; sub-plot errors are independent within and across whole plots; both error components are Normal with constant variance; the random-effects structure is correctly specified in the analysis.

16.29.5 R Implementation

```r
library(lme4); library(lmerTest)

set.seed(2026)
n_rep ← 2; n_wp ← 3; n_sp ← 4
df ← expand.grid(rep = 1:n_rep, temp = factor(1:n_wp),
                 form = factor(1:n_sp))
df$batch ← interaction(df$rep, df$temp)
batch_re ← rnorm(nlevels(df$batch), 0, 0.6)[df$batch]

df$y ← 10 + 0.5 * as.numeric(df$temp) + 0.7 * as.numeric(df$form) +
        batch_re + rnorm(nrow(df), 0, 0.5)

fit ← lmer(y ~ temp * form + (1 | batch), data = df)
anova(fit)
```

16.29.6 Output & Results

`lmer()` with a `(1 | batch)` random effect captures the whole-plot stratum (each batch corresponds to one whole-plot setting of temperature), and the residual captures the sub-plot stratum. `anova()` from `lmerTest` returns F-tests with Satterthwaite or Kenward-Roger denominator df, automatically routing each effect against its appropriate stratum.

16.29.7 Interpretation

A reporting sentence: "The split-plot mixed model showed a significant whole-plot effect of oven temperature ($F_{2,4} = 5.2$, $p = 0.04$, against batch-level error) and a highly significant sub-plot effect of formulation ($F_{3,18} = 14.8$, $p < 0.001$, against residual error). The temperature × formulation interaction was non-significant ($F_{6,18} = 1.1$, $p = 0.40$). Sub-plot effects were tested with substantially more power because the sub-plot error was much smaller than the whole-plot error ($\sigma_S^2 = 0.25$ vs $\sigma_W^2 = 0.36$)." Always identify which stratum each F-test uses.

16.29.8 Practical Tips

- Always match the error term to the effect's level in the design; analysing a split-plot as if it were a CRD treats whole-plot and sub-plot effects with the wrong denominator and inflates or deflates significance for the wrong factor.
- Mixed-effects models (`lmer` from `lme4`) give correct standard errors and F-tests when the random-effect structure is specified correctly; explicit attention to the random-effect formula is the single most common source of split-plot analysis errors.
- Replicate whole plots whenever possible — whole-plot F-tests have very few df otherwise. With three whole-plot levels and a single replicate, the whole-plot test has essentially no power; a second replicate at minimum is recommended.
- Split-plot structure arises naturally in agricultural field experiments (treatments applied to whole strips, sub-treatments within), manufacturing (batch-level process settings, within-batch additives), and instrumentation studies (calibration once per session, sample-level treatments within session).
- Strip-plot designs extend the split-plot idea to two hard-to-change factors applied in perpendicular directions; they have three error strata and require correspondingly more careful random-effects specification.
- Assigning a factor to the whole-plot stratum reduces its detection power; if a factor is scientifically critical, assign it to the sub-plot stratum if possible — even at the cost of some operational inconvenience.

16.29.9 R Packages Used

`lme4::lmer()` and `lmerTest` for mixed-effects analysis with denominator-df adjustment; `agricolae::design.split()` for canonical split-plot layout construction; `daewr` for textbook companion datasets and analysis helpers; `emmeans` for marginal means and contrasts at each stratum; `nlme::lme()` for an alternative interface with explicit fixed/random specification.

16.29.10 For Reviewers

Use the shared reviewer checklist, together with this method's assumptions and practical tips.

Method-specific reviewer notes have not yet been developed for this entry. The links above support a structured review; they do not certify the example or a manuscript's application of the method.

16.29.11 See also — labs in this chapter

- Observational designs and STROBE
- Randomised controlled trials
- Adaptive, non-inferiority, equivalence trials
- Bench and translational design
- MCAR, MAR, MNAR
- Multiple imputation with mice

- Linear mixed models with lme4
- GLMMs and GEE
- DAGs with dagitty and ggdag
- Propensity scores and IPTW
- G-methods, IV, DiD, RDD; heterogeneity of treatment effect
- SIR and SEIR models with deSolve
- Pre-registration and statistical analysis plans

16.30 Method of Steepest Ascent

16.30.1 Introduction

The method of steepest ascent is the workhorse of the first phase of response-surface methodology. After fitting a linear (first-order) model to a small factorial experiment in the current operating region, the gradient vector points in the direction of fastest predicted improvement; subsequent confirmation runs are taken sequentially along this gradient, moving the experimental region toward the neighbourhood of the optimum. The strategy is efficient because it replaces an exhaustive grid search of the factor space with a directed walk informed by the local linear approximation, dramatically reducing the number of runs required to locate the region near a process optimum.

16.30.2 Prerequisites

A working understanding of first-order linear regression, the concept of the gradient vector, and the sequential staging of experimentation in response-surface methodology.

16.30.3 Theory

Fit a first-order linear model

$$\hat{y} = \beta_0 + \sum_i \beta_i x_i$$

in the current operating region using a small two-level factorial with replicated centre points. The steepest-ascent direction is the gradient $\nabla \hat{y} = (\beta_1, \ldots, \beta_k)$, normalised to unit length. Subsequent runs are taken at successive multiples of a chosen step size along this direction; experimentally observed responses guide whether to continue, slow down, or stop. When the response stops improving along the path, a new first-order design is fit at the new operating region and the procedure iterates.

16.30.4 Assumptions

The local first-order linear approximation is adequate in the current region (no strong curvature would invalidate the gradient direction), the response surface is smooth, and the step size along the path is appropriate — too small and the search wastes runs, too large and it overshoots the optimum. Centre-point replicates allow detection of curvature via lack-of-fit, signalling when to switch to a second-order fit.

16.30.5 R Implementation

```
library(rsm)

set.seed(2026)
region1 ← data.frame(x1 = c(-1, 1, -1, 1, 0),
                     x2 = c(-1, -1, 1, 1, 0))
region1$y ← 10 + 2 * region1$x1 + 1.5 * region1$x2 +
            rnorm(nrow(region1), 0, 0.3)

fit1 ← rsm(y ~ FO(x1, x2), data = region1)
summary(fit1)

direction ← steepest(fit1, dist = c(0.5, 1, 2, 3, 5, 7))
direction
```

16.30.6 Output & Results

`rsm()` with the first-order specification returns the gradient coefficients with significance tests and any lack-of-fit information from centre replicates. `steepest()` traces the path of steepest ascent at requested distances from the design centre, producing predicted responses along the path that guide the experimentalist's choice of follow-up runs.

16.30.7 Interpretation

A reporting sentence: "The first-order fit in the initial operating region gave coefficients $\hat{\beta}_1 = 2.0$ and $\hat{\beta}_2 = 1.5$, both highly significant ($p < 0.001$). The steepest-ascent path moved in the direction $(0.80, 0.60)$ with step ratio approximately $2 : 1.5$. Confirmation runs at distances 1, 3, 5, and 7 from the centre showed monotone increase to a plateau at distance 5; a new first-order design was then fit centred there, revealing significant curvature ($p = 0.004$ from centre-point lack-of-fit), and a second-order CCD was used for final characterisation." Always describe the path and the stopping criterion.

16.30.8 Practical Tips

- Take the first few steps along the path experimentally rather than relying purely on numerical extrapolation; the linear approximation is local, and only experimental confirmation reveals when it begins to break down.
- Stop moving along the path when the response plateaus or decreases, then fit a new first-order design at the new operating region and repeat the process; this iterative restart is the key to efficient convergence on the optimum.
- When the gradient coefficients are roughly equal in magnitude, the ascent direction is diagonal in the factor space; when one dominates, the path is primarily along that axis. Inspecting the path on a contour plot makes the direction concrete and informs step-size choice.
- If the first-order fit shows significant curvature (a centre-point lack-of-fit test rejects), the linear approximation is inadequate and steepest ascent should be replaced by a second-order fit (CCD, BBD) for final characterisation.
- Steepest ascent is unconstrained; if factor levels have hard physical or operational constraints, use ridge analysis (constrained optimisation along the direction of

steepest improvement, restricted to the feasible region) instead.
- Pre-specify the step-size schedule and the stopping criterion in the experimental protocol; ad-hoc decisions during a steepest-ascent run are a recurring source of irreproducible RSM results.

16.30.9 R Packages Used

`rsm::FO()` for first-order model specification, `rsm::steepest()` for the canonical path-of-steepest-ascent computation, `rsm::ridge()` for constrained ridge analysis when factor bounds bind; `qualityTools` for an alternative RSM toolkit; `desirability` for multi-response steepest-ascent equivalents under desirability-function optimisation; `daewr` for textbook companion datasets.

16.30.10 For Reviewers

Use the shared reviewer checklist, together with this method's assumptions and practical tips.

Method-specific reviewer notes have not yet been developed for this entry. The links above support a structured review; they do not certify the example or a manuscript's application of the method.

16.30.11 See also — labs in this chapter

- Observational designs and STROBE
- Randomised controlled trials
- Adaptive, non-inferiority, equivalence trials
- Bench and translational design
- MCAR, MAR, MNAR
- Multiple imputation with mice
- Linear mixed models with lme4
- GLMMs and GEE
- DAGs with dagitty and ggdag
- Propensity scores and IPTW
- G-methods, IV, DiD, RDD; heterogeneity of treatment effect
- SIR and SEIR models with deSolve
- Pre-registration and statistical analysis plans

16.31 Strip-Plot Designs

16.31.1 Introduction

Strip-plot designs — also known as split-block designs — apply two hard-to-change factors in perpendicular strips across a block, with the resulting factor combinations realised in the cells of the cross. They arise naturally in agricultural field experiments where one factor (e.g., irrigation level) is applied along rows and another (e.g., fertiliser type) along columns, in industrial manufacturing where two stages of processing each

cover an entire pallet, and in food and pharmaceutical settings where two restrictions on randomisation operate in different directions. The defining feature is three independent error strata — one for the row factor, one for the column factor, and one for their interaction — each requiring its own appropriate F-test denominator.

16.31.2 Prerequisites

A working understanding of split-plot designs, the concept of multiple error strata, and mixed-effects model specification with nested random effects.

16.31.3 Theory

The strip-plot model has three error strata corresponding to the three sources of restricted randomisation:

- **Row factor**: applied to entire rows; tested against row-stratum error (variation among rows within blocks).
- **Column factor**: applied to entire columns; tested against column-stratum error.
- **Row × column interaction**: tested against the cell-level (smallest) error.

Each main effect therefore has lower power than it would in a completely randomised design, because its denominator is built from a coarser stratum, but the design is operationally feasible when full randomisation of the two hard-to-change factors is impractical or prohibitively expensive.

16.31.4 Assumptions

The three error strata are independent, errors are Normal and homoscedastic at each stratum, and the random-effects structure is correctly specified — typically random effects for row-within-block and column-within-block, with the cell-level residual capturing the interaction-level error.

16.31.5 R Implementation

```
library(lme4); library(lmerTest)

set.seed(2026)
n_rep ← 3
rows ← factor(rep(1:3, each = 4, times = n_rep))
cols ← factor(rep(rep(1:4, 3), n_rep))
rep_f ← factor(rep(1:n_rep, each = 12))
df ← data.frame(rep = rep_f, rows, cols)

df$row_batch ← interaction(df$rep, df$rows)
df$col_batch ← interaction(df$rep, df$cols)

row_re ← rnorm(nlevels(df$row_batch), 0, 0.5)[df$row_batch]
col_re ← rnorm(nlevels(df$col_batch), 0, 0.4)[df$col_batch]

df$y ← 10 + 0.6 * as.numeric(df$rows) + 0.4 * as.numeric(df$cols) +
        row_re + col_re + rnorm(nrow(df), 0, 0.3)

fit ← lmer(y ~ rows * cols + (1 | row_batch) + (1 | col_batch),
           data = df)
anova(fit)
```

16.31.6 Output & Results

`lmer()` with two crossed random-effect terms (row-within-replicate and column-within-replicate) and the cell-level residual recovers the three error strata. `anova()` returns the appropriate F-tests for the row and column main effects against their respective stratum errors and for the interaction against the cell-level error. Reporting all three F-tests with their stratum-specific denominator df is the standard summary.

16.31.7 Interpretation

A reporting sentence: “The strip-plot ANOVA showed the row factor was marginally significant ($F_{2,4} = 3.8$, $p = 0.05$ against row-stratum error), the column factor was significant ($F_{3,6} = 6.4$, $p = 0.01$ against column-stratum error), and the row × column interaction was non-significant when tested against the cell-level error ($F_{6,24} = 1.2$, $p = 0.34$). Random-effect variance components for row-within-replicate and column-within-replicate were modest, indicating the strip structure absorbed limited but non-trivial nuisance variation.” Always report each effect against its appropriate stratum.

16.31.8 Practical Tips

- Strip-plot designs require replication across blocks (multiple replicates of the cross structure) for valid inference; a single block leaves no error degrees of freedom for the row or column main effects.
- Strip-plot structure is most natural in agricultural field experiments where two perpendicular operations (e.g., ploughing direction and spraying direction) are applied independently, and in industrial manufacturing where two upstream processes cover whole batches in different ways.

- Correct random-effects specification is critical; an incorrect model treats restricted randomisation as if it were complete, yielding F-tests with the wrong denominator and inflated false-positive rates.
- Use `lmerTest::ranova()` to test the significance of the random-effect variance components themselves; this also helps diagnose whether the strip structure was substantively different from a CRD analysis would have implied.
- In industrial settings, strip-plot structure arises whenever two manufacturing stages each apply to a whole pallet, batch, or run before disaggregating; recognising the structure rather than ignoring it is essential to valid inference.
- For analysis software-checks, fit the same dataset with the wrong (CRD-like) model and the right (strip-plot) model; comparing the two highlights the inferential cost of ignoring the design and is a useful teaching diagnostic.

16.31.9 R Packages Used

`lme4::lmer()` and `lmerTest` for mixed-effects analysis with denominator-df adjustment; `agricolae::design.strip()` for the canonical strip-plot layout construction; `daewr` for textbook-companion strip-plot datasets and analysis helpers; `emmeans` for marginal means and contrasts at each stratum.

16.31.10 For Reviewers

Use the shared reviewer checklist, together with this method's assumptions and practical tips.

Method-specific reviewer notes have not yet been developed for this entry. The links above support a structured review; they do not certify the example or a manuscript's application of the method.

16.31.11 See also — labs in this chapter

- Observational designs and STROBE
- Randomised controlled trials
- Adaptive, non-inferiority, equivalence trials
- Bench and translational design
- MCAR, MAR, MNAR
- Multiple imputation with mice
- Linear mixed models with lme4
- GLMMs and GEE
- DAGs with dagitty and ggdag
- Propensity scores and IPTW
- G-methods, IV, DiD, RDD; heterogeneity of treatment effect
- SIR and SEIR models with deSolve
- Pre-registration and statistical analysis plans

16.32 Taguchi Methods

16.32.1 Introduction

Taguchi methods (1950s-60s) apply orthogonal-array experiments to optimise product and process design for minimum variability. Central ideas: loss functions defining quality as deviation from target, signal-to-noise (SNR) ratios summarising mean and variance jointly, and compact orthogonal arrays for efficient screening.

16.32.2 Prerequisites

Fractional factorial designs; signal-to-noise.

16.32.3 Theory

Quality loss $L(y) = k(y-\tau)^2$ for target τ – quality decreases quadratically with deviation.

SNR: - Smaller-is-better: $\text{SNR} = -10\log_{10}(\sum y_i^2/n)$. - Larger-is-better: $\text{SNR} = -10\log_{10}(\sum 1/y_i^2/n)$. - Nominal-is-best: $\text{SNR} = 10\log_{10}(\bar{y}^2/s^2)$.

Taguchi arrays (L4, L8, L9, L16) are Plackett-Burman-style orthogonal arrays with Japanese quality-engineering conventions.

16.32.4 Assumptions

Main effects dominate (sparsity); variance reduction via control factor choice (robust design).

16.32.5 R Implementation

```
library(qualityTools)

set.seed(2026)
# L8 array (2-level, 7 factors)
ta ← taguchiDesign("L8_2^7")
summary(ta)

# Simulate responses
ta$y ← rnorm(8, mean = 10 + 2 * as.numeric(ta$A) +
                         1.5 * as.numeric(ta$B), sd = 0.5)
effectPlot(ta, response = "y",
           main = "Taguchi main-effects plot (L8)")
```

16.32.6 Output & Results

L8 orthogonal array and main-effects plot showing the average response at each factor's two levels.

16.32.7 Interpretation

“The L8 design screened 7 factors in 8 runs; the main-effects plot identified factors A and B as dominant, with the optimal settings maximising (or minimising) the target response.”

16.32.8 Practical Tips

- Taguchi arrays are essentially Plackett-Burman designs; modern DoE uses statistical fractional factorials with aliasing awareness.
- SNR metrics conflate mean and variance; modern practice separates them via dual-response surface methods.
- Taguchi’s three-stage design (system, parameter, tolerance) is a useful framework; the specific SNR tools are debated.
- Orthogonal arrays are efficient for screening; interactions must be explicitly assigned to specific columns.
- For manufacturing and quality engineering, Taguchi remains a widely-used vocabulary; modern methods supersede the specifics.

16.32.9 For Reviewers

Use the shared reviewer checklist, together with this method’s assumptions and practical tips.

Method-specific reviewer notes have not yet been developed for this entry. The links above support a structured review; they do not certify the example or a manuscript’s application of the method.

16.32.10 See also — labs in this chapter

- Observational designs and STROBE
- Randomised controlled trials
- Adaptive, non-inferiority, equivalence trials
- Bench and translational design
- MCAR, MAR, MNAR
- Multiple imputation with mice
- Linear mixed models with lme4
- GLMMs and GEE
- DAGs with dagitty and ggdag
- Propensity scores and IPTW
- G-methods, IV, DiD, RDD; heterogeneity of treatment effect
- SIR and SEIR models with deSolve
- Pre-registration and statistical analysis plans

16.33 Week 1, Session 3 — Adaptive, non-inferiority, equivalence trials

Inference lab using the five-step template: Hypothesis → Visualise → Assumptions → Conduct → Conclude.

16.33.1 Learning objectives

- Distinguish superiority, non-inferiority, and equivalence trials by their null hypotheses.
- Choose a non-inferiority margin and say what it means clinically.
- Run and plot a non-inferiority test with a confidence interval versus the margin.

16.33.2 Prerequisites

Inferential statistics and confidence intervals.

16.33.3 Background

A superiority trial asks whether a new treatment beats the comparator. A non-inferiority (NI) trial asks whether the new treatment is *not worse than* the comparator by more than a pre-specified margin (often called Δ). An equivalence trial asks whether the new treatment is neither better nor worse than the comparator by more than Δ on either side. These are distinct statistical questions and they require different designs, analyses, and sample sizes.

The NI margin is the hardest thing about an NI trial. It has to be small enough that clinicians and patients would trade it for the benefits of the new treatment (lower cost, fewer side effects, easier administration), but not so small that the trial is infeasible. Regulators will often ask for a margin no larger than a defined fraction of the historical treatment effect of the comparator versus placebo.

Adaptive trials modify some aspect of the design (sample size, arm allocation, the decision rule) during the trial based on interim results, using pre-specified rules. Group sequential designs, for example, stop early for overwhelming efficacy or for futility, while spending Type I error according to a chosen alpha-spending function.

A one-sided 97.5% confidence interval that stays on the correct side of the margin is the operational test in most NI trials. Graphically, this is a forest plot where you check that the interval does not cross the margin line.

16.33.4 Setup

```
library(tidyverse)
set.seed(42)
theme_set(theme_minimal(base_size = 12))
```

16.33.5 1. Hypothesis

H0: the new treatment is worse than the comparator by more than Δ = 3 units on a continuous outcome. H1: the new treatment is not worse by more than Δ.

16.33.6 2. Visualise

```
n ← 150
trial ← tibble(
  arm = rep(c("comparator", "new"), each = n),
  y   = c(rnorm(n, 70, 10), rnorm(n, 69, 10))
)

trial ▷
  ggplot(aes(arm, y, fill = arm)) +
  geom_boxplot(alpha = 0.6, colour = "grey30") +
  labs(x = NULL, y = "Outcome") +
  theme(legend.position = "none")
```

16.33.7 3. Assumptions

Independent observations, roughly normal within-arm residuals, and a pre-specified margin declared before looking at the data. The *direction* of the test matters: we test whether the lower bound of the mean difference (new − comparator) is above −Δ.

16.33.8 4. Conduct

```
fit ← t.test(y ~ arm, data = trial)
fit
```

```
delta ← 3
ci ← fit$conf.int
est ← diff(rev(fit$estimate))  # new - comparator
ni_pass ← ci[1] > -delta
tibble(estimate = est, low = ci[1], high = ci[2], margin = -delta, ni_pass)
```

```
tibble(estimate = est, low = ci[1], high = ci[2]) ▷
  ggplot(aes(x = estimate, y = 1)) +
  geom_point(size = 3) +
  geom_errorbarh(aes(xmin = low, xmax = high), height = 0.1) +
  geom_vline(xintercept = 0, linetype = "dashed", colour = "grey40") +
  geom_vline(xintercept = -delta, colour = "firebrick") +
  labs(x = "Mean difference (new - comparator)", y = NULL) +
  theme(axis.text.y = element_blank())
```

16.33.9 5. Concluding statement

The new treatment had a mean difference of `round(est, 2)` versus comparator (95% CI: `round(ci[1], 2)` to `round(ci[2], 2)`). With a pre-specified non-inferiority margin of Δ = 3, the new treatment `if (ni_pass) "met" else "did not meet"` the non-inferiority criterion because the lower bound `if (ni_pass) "exceeded" else "fell below"` −Δ.

16.33.9.1 A word on adaptive designs

Adaptive designs work when the adaptation rules and the alpha spent at each look are pre-specified. Running an interim analysis with the hope of extending the trial if it looks promising — without a pre- specified rule — inflates Type I error and destroys the trial's inferential warranty.

16.33.10 Common pitfalls

- Picking the NI margin after seeing the data.
- Using a two-sided superiority test and claiming non-inferiority.
- Running an "interim look" without a pre-specified spending rule.
- Treating equivalence and non-inferiority as interchangeable.

16.33.11 Further reading

- Piaggio G et al. (2012), *CONSORT extension for non-inferiority and equivalence trials*.
- Jennison C, Turnbull BW. *Group Sequential Methods with Applications to Clinical Trials*.
- FDA (2016), *Non-Inferiority Clinical Trials to Establish Effectiveness*.

16.33.12 Session info

```
sessionInfo()
```

16.33.13 See also — chapter index

- Methods in this chapter
- Find your method (Chapter 0)

16.34 Week 1, Session 4 — Bench and translational design

Inference lab using the five-step template: Hypothesis → Visualise → Assumptions → Conduct → Conclude.

16.34.1 Learning objectives

- Recognise blocking, factorial, and split-plot designs in a bench setting.
- Simulate a 2 × 2 factorial experiment and test the interaction.
- Spot and correct a pseudoreplication error in a cell-culture example.

16.34.2 Prerequisites

Linear regression and ANOVA; familiarity with the formula interface.

16.34.3 Background

Bench and translational biology lives on small, carefully structured experiments whose designs are often misread as simple one-way comparisons. Blocking groups observations that share a nuisance variable (a batch, a day, a plate) so that comparisons are made within blocks. Factorial designs manipulate two or more factors at once and estimate main effects and interactions from a single experiment. Split-plot designs randomise two factors at different scales — a hard-to-change factor (oven temperature) at the whole- plot level, an easier one (position on the tray) at the sub-plot level — and the analysis must respect that structure.

Pseudoreplication is what happens when the experimental unit and the observational unit are confused. If one animal's blood is pipetted into ten wells, the ten wells are not ten independent samples; they are ten technical replicates of one biological replicate. Analysing them as independent inflates the apparent sample size, shrinks the standard error, and produces spuriously small *p*-values. The fix is to aggregate within the experimental unit or use a mixed model that knows about the nested structure.

A useful question to ask before every analysis: *what is the unit I could have independently randomised?* That is the unit of analysis. If an intervention was applied to a cage but measurements were taken on each mouse, the cage is the unit.

16.34.4 Setup

```
library(tidyverse)
set.seed(42)
theme_set(theme_minimal(base_size = 12))
```

16.34.5 1. Hypothesis

A drug and a diet each affect a continuous biomarker, and their combination may have a non-additive effect (interaction).

16.34.6 2. Visualise

```
n_per_cell ← 10
fact ← expand_grid(
  drug = c("no", "yes"),
  diet = c("standard", "modified"),
  rep  = seq_len(n_per_cell)
) ▷
  mutate(
    y = 10 +
      (drug == "yes")  * 1.5 +
      (diet == "modified") * 1.0 +
      (drug == "yes" & diet == "modified") * 2.0 +
      rnorm(n(), 0, 1.5)
  )

fact ▷
  ggplot(aes(diet, y, fill = drug)) +
  geom_boxplot(alpha = 0.6, colour = "grey30") +
  labs(x = NULL, y = "Biomarker", fill = "Drug")
```

16.34.7 3. Assumptions

Independent observations *within each cell* (every replicate is a separate biological unit), approximately normal residuals, and equal variance across cells. For the pseudoreplication demonstration we deliberately break the independence assumption.

```
fact ▷
  group_by(drug, diet) ▷
  summarise(mean = mean(y), sd = sd(y), n = n(), .groups = "drop")
```

16.34.8 4. Conduct

```
fit ← lm(y ~ drug * diet, data = fact)
anova(fit)
```

Now a pseudoreplication error. Suppose each biological replicate was measured three times (technical replicates) and we naively treated the 3 × rows as independent.

```
pseudo ← fact ▷
  slice(rep(seq_len(n()), each = 3)) ▷
  mutate(tech_noise = rnorm(n(), 0, 0.3),
         y = y + tech_noise)

fit_bad  ← lm(y ~ drug * diet, data = pseudo)
fit_good ← lm(y ~ drug * diet, data = fact)  # aggregated (original)

bind_rows(
  bad  = broom::tidy(fit_bad)  ▷ filter(term == "drugyes:dietmodified"),
  good = broom::tidy(fit_good) ▷ filter(term == "drugyes:dietmodified"),
  .id = "analysis"
)
```

The point estimate barely moves; the standard error in the bad analysis is smaller because the apparent *n* is three times larger.

16.34.9 5. Concluding statement

> In a simulated 2 × 2 factorial experiment (10 biological replicates per cell), there was evidence of an interaction between drug and diet on the biomarker (*F* and *p* from the ANOVA table above). Naive analysis of technical replicates as independent observations shrinks the standard error of the interaction estimate without changing the point estimate — a textbook pseudoreplication artefact.

16.34.10 Common pitfalls

- Counting wells, sections, or reads as independent samples.
- Forgetting to include the block in the model when the design was blocked.
- Running a one-way ANOVA on a factorial design and losing the interaction.
- Split-plot designs analysed with a single-error-term model.

16.34.11 Further reading

- Montgomery DC. *Design and Analysis of Experiments*.
- Lazic SE (2010), *The problem of pseudoreplication in neuroscientific studies*.
- Festing MFW (2006), *Design and statistical methods in studies using animal models*.

16.34.12 Session info

```
sessionInfo()
```

16.34.13 See also — chapter index

- Methods in this chapter
- Find your method (Chapter 0)

16.35 Week 3, Session 3 — DAGs with dagitty and ggdag

Workflow lab using the variant template: Goal → Approach → Execution → Check → Report.

16.35.1 Learning objectives

- Express a causal scenario as a directed acyclic graph (DAG).
- Identify confounders, mediators, and colliders from a DAG.
- Use `dagitty::adjustmentSets()` to find sufficient adjustment sets.

16.35.2 Prerequisites

Conceptual knowledge of confounding.

16.35.3 Background

A DAG is a set of variables joined by arrows that encode assumed causal directions. The graph distinguishes three kinds of third variable on a path between exposure and outcome: a *confounder* is a common cause; a *mediator* lies on the causal path from exposure to outcome; a *collider* is a common effect of two variables on the path. The practical upshot is that adjusting for a confounder reduces bias, adjusting for a mediator removes part of the effect you are trying to estimate, and adjusting for a collider *creates* bias where none existed.

`dagitty` and `ggdag` let you declare a DAG in text, visualise it, and then query it for adjustment sets — the minimal variable sets that block all back-door paths from exposure to outcome without opening new ones. The right adjustment set depends on the question. For the total effect of X on Y, you want to close all back-door paths but leave mediators alone; for the direct effect, you also block mediators.

M-bias is the classic example of collider adjustment: conditioning on a variable that is a common effect of an unmeasured cause of X and an unmeasured cause of Y opens a path and biases the estimate. This is why “adjust for everything” is bad advice.

16.35.4 Setup

```
library(tidyverse)
library(dagitty)
library(ggdag)
set.seed(42)
theme_set(theme_minimal(base_size = 12))
```

16.35.5 1. Goal

Write a confounder, mediator, and collider scenario as DAGs; ask `dagitty` which variables to adjust for.

16.35.6 2. Approach

```
dag1 ← dagitty('dag {
  X → Y
  C → X
  C → Y
  X [exposure]
  Y [outcome]
}')

dag2 ← dagitty('dag {
  X → M → Y
  X → Y
  X [exposure]
  Y [outcome]
}')

dag3 ← dagitty('dag {
  X → Z
  Y → Z
  X → Y
  X [exposure]
  Y [outcome]
}')

ggdag(dag1) + theme_dag()
ggdag(dag2) + theme_dag()
ggdag(dag3) + theme_dag()
```

16.35.7 3. Execution

```
adjustmentSets(dag1, exposure = "X", outcome = "Y",
               effect = "total")
```

```
# Total vs direct effect when a mediator exists
adjustmentSets(dag2, exposure = "X", outcome = "Y",
               effect = "total")
adjustmentSets(dag2, exposure = "X", outcome = "Y",
               effect = "direct")
```

```
# Collider: do not adjust for Z
adjustmentSets(dag3, exposure = "X", outcome = "Y")
```

16.35.8 4. Check

Simulate dag1 and verify that adjusting for C recovers the true effect while failing to adjust biases it.

```
n  ← 2000
c_ ← rnorm(n)
x  ← 0.6 * c_ + rnorm(n)
y  ← 0.4 * x + 0.7 * c_ + rnorm(n)
df ← tibble(c_, x, y)

coef(lm(y ~ x,         data = df))["x"]
coef(lm(y ~ x + c_,    data = df))["x"]
```

The unadjusted coefficient is inflated; the adjusted one is close to 0.4 as simulated.

16.35.9 5. Report

> A directed acyclic graph is a compact, testable statement of causal assumptions. Dagitty identified the confounder C as the required adjustment set in the first DAG; the collider Z in the third DAG is explicitly *not* in any adjustment set, and conditioning on it would introduce selection bias.

16.35.10 Common pitfalls

- Listing "every variable we measured" as the adjustment set.
- Adjusting for a mediator and calling the result a total effect.
- Using automated variable selection on observational data without a DAG.
- Drawing the DAG after the analysis.

16.35.11 Further reading

- Textor J et al. (2016), *Robust causal inference using directed acyclic graphs: the R package 'dagitty'*.
- Hernán MA, Robins JM. *Causal Inference: What If*.
- Greenland S, Pearl J, Robins JM (1999), *Causal diagrams for epidemiologic research*.

16.35.12 Session info

```
sessionInfo()
```

16.35.13 See also — chapter index

- Methods in this chapter
- Find your method (Chapter 0)

16.36 Week 3, Session 5 — G-methods, IV, DiD, RDD; heterogeneity of treatment effect

Workflow lab: Goal → Approach → Execution → Check → Report.

16.36.1 Learning objectives

- Use inverse-probability weighting as a g-method for time-varying confounding.
- Recognise the data-generating situations that justify an instrumental-variable (IV), difference-in-differences (DiD), or regression-discontinuity (RDD) design.
- Estimate and interpret heterogeneity of treatment effect (HTE) with a subgroup-agnostic method.

16.36.2 Prerequisites

Propensity-score and IPTW basics (W3 S4).

16.36.3 Background

Traditional regression adjustment fails for time-varying confounders that are affected by prior treatment — adjusting for them blocks part of the treatment effect, failing to adjust leaves residual confounding. Robins's g-methods solve this by reweighting or standardising in a way that respects the temporal order. IV methods exploit a variable that affects treatment but not outcome directly; DiD compares changes over time between groups sharing a counterfactual trend; RDD exploits a sharp assignment rule around a threshold.

Heterogeneity of treatment effect is the shift from "is the average effect non-zero?" to "for whom is it largest?" Model-based HTE uses causal forests, meta-learners (S-, T-, X-learner), or Bayesian hierarchical models to estimate conditional average treatment effects (CATEs) while controlling for multiple testing.

16.36.4 Setup

```
library(tidyverse)
library(broom)
set.seed(42)
theme_set(theme_minimal(base_size = 12))
```

16.36.5 1. Goal

Simulate a time-varying-confounding scenario, estimate the causal effect naively and then with IPTW (a marginal structural model), and finish with a simple HTE estimate.

16.36.6 2. Approach

```
n ← 2000
df ← tibble(
  l0 = rnorm(n),
  a0 = rbinom(n, 1, plogis(0.2 * l0)),
  l1 = rnorm(n, mean = 0.5 * a0 + 0.3 * l0),
  a1 = rbinom(n, 1, plogis(0.3 * l1 + 0.2 * a0)),
  y  = 1.0 * a0 + 0.8 * a1 + 0.5 * l0 + 0.5 * l1 + rnorm(n)
)
```

The true effect of each treatment is positive and additive. `l1` is a time-varying confounder because it is affected by `a0` and predicts both `a1` and `y`.

16.36.7 3. Execution

Naive regression adjusting for `l1` blocks part of the `a0` effect.

```
naive ← lm(y ~ a0 + a1 + l0 + l1, data = df)
tidy(naive, conf.int = TRUE) ▷
  filter(term %in% c("a0", "a1"))
```

IPTW with stabilised weights:

```
num0 ← glm(a0 ~ 1, data = df, family = binomial)
den0 ← glm(a0 ~ l0, data = df, family = binomial)
num1 ← glm(a1 ~ a0, data = df, family = binomial)
den1 ← glm(a1 ~ a0 + l0 + l1, data = df, family = binomial)

w0 ← ifelse(df$a0 == 1, fitted(num0), 1 - fitted(num0)) /
      ifelse(df$a0 == 1, fitted(den0), 1 - fitted(den0))
w1 ← ifelse(df$a1 == 1, fitted(num1), 1 - fitted(num1)) /
      ifelse(df$a1 == 1, fitted(den1), 1 - fitted(den1))
df$sw ← w0 * w1

msm ← lm(y ~ a0 + a1, data = df, weights = sw)
tidy(msm, conf.int = TRUE) ▷
  filter(term %in% c("a0", "a1"))
```

16.36.8 4. Check

Balance and weight sanity:

```
summary(df$sw)
```

Stabilised weights should centre on 1 with no extreme tail. If they do not, a term in the treatment model is likely misspecified.

16.36.9 5. Report

> In a simulated cohort with time-varying confounding (n = 2000), naive regression underestimated the effect of initial treatment. An inverse-probability-weighted marginal structural model recovered effects close to the true data-generating values. Instrumental variable, difference-in-differences, and regression-discontinuity designs exploit alternative identification strategies, each with its own assumptions. Heterogeneity of treatment effect can be explored with causal forests or meta-learners.

16.36.9.1 Brief sketches

- **IV:** regress `y` on an instrument `z` to recover `β = cov(y,z)/cov(a,z)`.
- **DiD:** fit `lm(y ~ time*group)` on repeated measures.
- **RDD:** fit local linear regression on each side of the assignment cutoff.

16.36.10 Common pitfalls

- Using ordinary regression adjustment for time-varying confounders.
- Trimming extreme weights silently, hiding model misspecification.
- Reporting subgroup p-values without multiplicity control as HTE evidence.

16.36.11 Further reading

- Hernán MA, Robins JM. *Causal Inference: What If.*
- Athey S, Imbens GW. (2019). *Machine Learning Methods That Economists Should Know About.*

16.36.12 Session info

```
sessionInfo()
```

16.36.13 See also — chapter index

- Methods in this chapter
- Find your method (Chapter 0)

16.37 Week 2, Session 4 — GLMMs and GEE

Inference lab using the five-step template: Hypothesis → Visualise → Assumptions → Conduct → Conclude.

16.37.1 Learning objectives

- Fit a binary GLMM for clustered data with `lme4::glmer` and with `glmmTMB`.
- Fit a GEE with `geepack::geeglm` and compare coefficient interpretations.
- Say why GLMM and GEE estimate different things (subject-specific vs population-average).

16.37.2 Prerequisites

Linear mixed models and logistic regression.

16.37.3 Background

For binary outcomes in clustered data, the two standard approaches are generalised linear mixed models (GLMM) and generalised estimating equations (GEE). A GLMM is a random-effects model on the link scale; its fixed-effect coefficients are conditional on the random effect and describe subject-specific effects. A GEE fits a marginal model using a working correlation structure; its coefficients describe population-averaged effects. For logistic models, the two will not in general agree even when both are correctly specified.

`lme4::glmer` is the classic random-effects fitter; `glmmTMB` handles larger and more complex random structures including correlated random effects and zero-inflated distributions. `geepack` fits GEEs with several working correlation choices; the standard errors are robust to misspecification of the correlation structure but the point estimate can shift.

The choice between GLMM and GEE is not a statistical technicality; it is a choice of estimand. If you care about what happens inside a person over time, you want a GLMM. If you care about a population- average effect for policy or for a marginal treatment effect, you want a GEE (or a marginalised GLMM).

16.37.4 Setup

```
library(tidyverse)
library(lme4)
library(glmmTMB)
library(geepack)
set.seed(42)
theme_set(theme_minimal(base_size = 12))
```

16.37.5 1. Hypothesis

Treatment reduces the probability of a repeated binary event in a clustered study (clusters = clinics). The true log-odds fixed effect is −0.5.

16.37.6 2. Visualise

```
J  ← 30
nj ← 20
dat ← tibble(
  clinic = factor(rep(seq_len(J), each = nj)),
  trt    = rbinom(J * nj, 1, 0.5),
  u      = rep(rnorm(J, 0, 0.8), each = nj)
) ▷
  mutate(lp = -0.5 * trt + u,
         y  = rbinom(n(), 1, plogis(lp)))

dat ▷
  group_by(clinic, trt) ▷
  summarise(p = mean(y), .groups = "drop") ▷
  ggplot(aes(factor(trt), p)) +
  geom_boxplot(alpha = 0.6) +
  labs(x = "Treatment", y = "Event probability per clinic")
```

16.37.7 3. Assumptions

GLMM: normal random intercepts for clinic, correct link. GEE: correct mean structure; standard errors robust to working correlation.

16.37.8 4. Conduct

```
fit_glmer ← glmer(y ~ trt + (1 | clinic),
                  data = dat, family = binomial)
fit_tmb   ← glmmTMB(y ~ trt + (1 | clinic),
                    data = dat, family = binomial)
fit_gee   ← geeglm(y ~ trt, id = clinic, data = dat,
                   family = binomial, corstr = "exchangeable")

summary(fit_glmer)$coefficients
summary(fit_tmb)$coefficients$cond
summary(fit_gee)$coefficients
```

```
bind_rows(
  glmer   = broom.mixed::tidy(fit_glmer, effects = "fixed"),
  glmmTMB = broom.mixed::tidy(fit_tmb,   effects = "fixed"),
  gee     = broom::tidy(fit_gee),
  .id = "model"
) ▷
  filter(term == "trt") ▷
  select(model, estimate, std.error)
```

16.37.9 5. Concluding statement

> The GLMM gave a subject-specific treatment log-odds of about `round(fixef(fit_glmer)["trt"], 2)`; the GEE gave a population- averaged log-odds of about `round(coef(fit_gee)["trt"], 2)`. The GEE estimate is typically attenuated toward zero relative to the GLMM; both are correct answers to different questions.

16.37.10 Common pitfalls

- Comparing GLMM and GEE coefficients as if they estimate the same thing.
- Using a working independence structure in GEE when clusters are large and treatment is heterogeneous within them.
- Ignoring overdispersion in a count GLMM by assuming Poisson when negative binomial fits better.
- Fitting a GLMM with too few clusters (< 10) and trusting the variance component.

16.37.11 Further reading

- Liang KY, Zeger SL (1986), *Longitudinal data analysis using generalized linear models*.
- Zeger SL, Liang KY, Albert PS (1988), *Models for longitudinal data: a generalized estimating equation approach*.
- Brooks ME et al. (2017), *glmmTMB balances speed and flexibility*.

16.37.12 Session info

```
sessionInfo()
```

16.37.13 See also — chapter index

- Methods in this chapter
- Find your method (Chapter 0)

16.38 Week 2, Session 3 — Linear mixed models with lme4

Inference lab using the five-step template: Hypothesis → Visualise → Assumptions → Conduct → Conclude.

16.38.1 Learning objectives

- Fit random-intercept and random-slope linear mixed models to longitudinal data.
- Test whether adding a random slope improves fit (likelihood ratio test with ML).
- Interpret REML estimates of variance components.

16.38.2 Prerequisites

Linear regression; familiarity with `lm()`.

16.38.3 Background

Longitudinal data have two sources of variation: differences between people and differences over time within a person. A random-intercept model lets each person have their own average level; a random-slope model lets each person also have their own rate of change. When the slopes genuinely vary, omitting the random slope underestimates the standard error of the fixed-effect slope and produces *p*-values that are too small.

REML (restricted maximum likelihood) is the default because it gives unbiased estimates of the variance components by integrating out the fixed effects. It is the right choice for reporting variance components. For comparing models that differ in fixed effects, you must refit with ML (likelihood ratio tests on REML fits are invalid); for comparing models that differ only in random effects, REML is fine.

A common misstep is to interpret the variance of the random intercept as the "within-subject variance". It is the between- subject variance of the intercept. The within-subject variance is the residual variance, σ^2, once the random effects are accounted for.

16.38.4 Setup

```
library(tidyverse)
library(lme4)
library(lmerTest)
set.seed(42)
theme_set(theme_minimal(base_size = 12))
```

16.38.5 1. Hypothesis

In the `sleepstudy` data, reaction time increases with days of sleep deprivation, and the *rate* of increase varies by subject.

16.38.6 2. Visualise

```
data(sleepstudy, package = "lme4")

sleepstudy |>
  ggplot(aes(Days, Reaction, group = Subject)) +
  geom_line(alpha = 0.3) +
  geom_smooth(aes(group = 1), method = "lm", se = FALSE,
              colour = "firebrick") +
  labs(x = "Days of deprivation", y = "Reaction time (ms)")
```

16.38.7 3. Assumptions

Linear relationship between Days and Reaction for each subject; normally distributed random intercepts and slopes (if included); residuals independent within and across subjects after random effects account for clustering; approximately constant residual variance.

16.38.8 4. Conduct

```
ri <- lmer(Reaction ~ Days + (1 | Subject),
           data = sleepstudy, REML = TRUE)
rs <- lmer(Reaction ~ Days + (Days | Subject),
           data = sleepstudy, REML = TRUE)
summary(rs)
```

```
# LRT on REML OK here because only random-effects structure differs,
# but we use REFIT = TRUE in anova() to get correct comparison.
anova(ri, rs, refit = FALSE)
```

```
# Fit with ML if the fixed effects changed
fm_ml <- lmer(Reaction ~ Days + (Days | Subject),
              data = sleepstudy, REML = FALSE)
fm_ml
```

```
sleepstudy |>
  mutate(fit_rs = predict(rs)) |>
  ggplot(aes(Days, Reaction, group = Subject)) +
  geom_point(alpha = 0.4) +
  geom_line(aes(y = fit_rs), colour = "steelblue") +
  facet_wrap(~ Subject, ncol = 6) +
  theme(strip.text = element_text(size = 7))
```

16.38.9 5. Concluding statement

> In the sleepstudy data, reaction time increased by about `round(fixef(rs)["Days"], 1)` ms per day of deprivation, with substantial subject-to-subject variation in slope (SD ≈ `round(sqrt(VarCorr(rs)$Subject[2, 2]), 1)` ms/day). Adding a random slope significantly improved fit over a random-intercept model (likelihood-ratio test $p < 0.001$).

16.38.10 Common pitfalls

- Testing random-effects with REML-based LRTs without refitting.
- Ignoring the random slope when slopes clearly vary.
- Treating the `Intercept` variance as the within-subject error.
- Using *p*-values from the Wald *z* on small samples.

16.38.11 Further reading

- Bates D et al. (2015), *Fitting linear mixed-effects models using lme4*.
- Pinheiro JC, Bates DM. *Mixed-Effects Models in S and S-PLUS*.
- Gelman A, Hill J. *Data Analysis Using Regression and Multilevel/Hierarchical Models*.

16.38.12 Session info

```
sessionInfo()
```

16.38.13 See also — chapter index

- Methods in this chapter
- Find your method (Chapter 0)

16.39 Week 2, Session 1 — MCAR, MAR, MNAR

Inference lab using the five-step template: Hypothesis → Visualise → Assumptions → Conduct → Conclude.

16.39.1 Learning objectives

- Define MCAR, MAR, and MNAR and give an example of each.
- Simulate the three mechanisms and measure the bias in a complete-case analysis.
- Decide when a missingness assumption is defensible from the data alone and when it is not.

16.39.2 Prerequisites

Regression basics.

16.39.3 Background

The language of missingness is Rubin's taxonomy. MCAR (missing completely at random) means the probability of being missing does not depend on anything — observed or unobserved. MAR (missing at random) means the probability depends only on observed variables. MNAR (missing not at random) means the probability depends on the missing value itself, even after conditioning on everything observed. Each is a property of the process, not the data, and you can rarely decide between MAR and MNAR from the data alone.

Complete-case analysis drops any row with a missing value. It is unbiased under MCAR, usually biased under MAR, and always suspect under MNAR. Multiple imputation handles MAR correctly if the imputation model is compatible with the analysis model, but cannot rescue MNAR without explicit modelling of the missingness mechanism.

Auxiliary variables — things that correlate with both the outcome and the missingness but are not in the analysis model — are the currency of MAR. If you have them, put them in the imputation model. If you do not, your MAR claim is on thinner ice than it appears.

16.39.4 Setup

```
library(tidyverse)
set.seed(42)
theme_set(theme_minimal(base_size = 12))
```

16.39.5 1. Hypothesis

We estimate the mean of Y from each of three datasets with the same true mean, where Y has been made missing by MCAR, MAR, and MNAR processes.

16.39.6 2. Visualise

```
n  ← 1000
df ← tibble(
  x = rnorm(n, 50, 10),
  y = 2 + 0.5 * x + rnorm(n, 0, 5)
)

mcar ← df ▷ mutate(y_obs = if_else(runif(n) < 0.30, NA_real_, y),
                   mech  = "MCAR")
mar  ← df ▷ mutate(p = plogis(-2 + 0.05 * (x - 50)),
                   y_obs = if_else(runif(n) < p, NA_real_, y),
                   mech  = "MAR")
mnar ← df ▷ mutate(p = plogis(-2 + 0.05 * (y - mean(y))),
                   y_obs = if_else(runif(n) < p, NA_real_, y),
                   mech  = "MNAR")

all ← bind_rows(
  mcar ▷ select(mech, x, y, y_obs),
  mar  ▷ select(mech, x, y, y_obs),
  mnar ▷ select(mech, x, y, y_obs)
)

all ▷
  mutate(missing = is.na(y_obs)) ▷
  ggplot(aes(x, y, colour = missing)) +
  geom_point(alpha = 0.5) +
  facet_wrap(~ mech) +
  labs(colour = "Y missing?")
```

16.39.7 3. Assumptions

Complete-case analysis is the bluntest possible method. It is justified if missingness is MCAR. We will compute the mean of Y on observed rows for each mechanism and compare with the true mean.

16.39.8 4. Conduct

```
results ← all ▷
  group_by(mech) ▷
  summarise(
    truth = mean(y),
    cc    = mean(y_obs, na.rm = TRUE),
    bias  = cc - truth,
    pct_missing = mean(is.na(y_obs)) * 100
  )
results
```

```
results ▷
  ggplot(aes(mech, bias)) +
  geom_col(alpha = 0.7, fill = "steelblue") +
  geom_hline(yintercept = 0, linetype = "dashed") +
  labs(x = NULL, y = "Bias of complete-case mean")
```

16.39.9 5. Concluding statement

> Complete-case estimation of the mean of Y was unbiased under MCAR (bias ≈ `round(results$bias[results$mech == "MCAR"], 2)`) and meaningfully biased under MAR (`round(results$bias[results$mech == "MAR"], 2)`) and MNAR (`round(results$bias[results$mech == "MNAR"], 2)`), despite the same true mean and similar missingness proportions.

16.39.10 Common pitfalls

- Using Little's test as evidence *for* MCAR.
- Imputing with a model that omits the analysis outcome.
- Reporting a sample size inflated by a complete-case filter that hides 40% dropout.
- Using "the missingness was not significantly related to X" as proof of MAR.

16.39.11 Further reading

- Little RJA, Rubin DB. *Statistical Analysis with Missing Data*.
- van Buuren S (2018), *Flexible Imputation of Missing Data*.
- Sterne JA et al. (2009), *Multiple imputation for missing data in epidemiological and clinical research*.

16.39.12 Session info

```
sessionInfo()
```

16.39.13 See also — chapter index

- Methods in this chapter
- Find your method (Chapter 0)

16.40 Week 2, Session 2 — Multiple imputation with mice

Inference lab using the five-step template: Hypothesis → Visualise → Assumptions → Conduct → Conclude.

16.40.1 Learning objectives

- Run multiple imputation by chained equations with `mice`.
- Diagnose convergence with trace and density plots.
- Pool regression coefficients across imputations using Rubin's rules.

16.40.2 Prerequisites

The MCAR/MAR/MNAR lab.

16.40.3 Background

Multiple imputation (MI) replaces each missing value with several plausible values drawn from a predictive distribution, producing *m* completed datasets. Each completed dataset is analysed separately; the results are pooled using Rubin's rules so that the standard errors reflect both the within-imputation uncertainty (ordinary sampling variance) and the between-imputation uncertainty (the variance of the imputations themselves).

`mice` implements MI by chained equations: for each incomplete variable, it fits a conditional model given the others and draws imputations from the posterior predictive distribution of that model. The procedure iterates until the imputations stabilise. The standard diagnostics are trace plots (should mix and not drift) and density plots comparing observed and imputed values (should overlap but need not match exactly).

The inclusion of the outcome in the imputation model is not optional. Omitting it biases the imputations toward the null and attenuates the estimated effect in the pooled analysis.

16.40.4 Setup

```
library(tidyverse)
library(mice)
set.seed(42)
theme_set(theme_minimal(base_size = 12))
```

16.40.5 1. Hypothesis

In the `mice::nhanes` teaching dataset, the cholesterol outcome (`chl`) is related to age and BMI (`bmi`). We will estimate that relationship under MI.

16.40.6 2. Visualise

```
data(nhanes, package = "mice")
md.pattern(nhanes, plot = FALSE)
```

```
nhanes ▷
  mutate(missing_chl = is.na(chl)) ▷
  ggplot(aes(bmi, chl, colour = missing_chl)) +
  geom_point(size = 2) +
  labs(x = "BMI", y = "Cholesterol", colour = "chl missing?")
```

16.40.7 3. Assumptions

MAR conditional on the variables in the imputation model; a compatible imputation model (predictive mean matching by default); enough imputations (here *m* = 10) to stabilise the pooled standard errors.

16.40.8 4. Conduct

```
imp ← mice(nhanes, m = 10, method = "pmm",
           printFlag = FALSE, seed = 42)
imp
```

```
plot(imp)
```

```
densityplot(imp)
```

```
fit   ← with(imp, lm(chl ~ age + bmi))
pooled ← pool(fit)
summary(pooled, conf.int = TRUE)
```

16.40.9 5. Concluding statement

> Multiple imputation of the `nhanes` dataset (m = 10, predictive mean matching) gave a pooled coefficient for BMI on cholesterol of `round(summary(pooled)$estimate[summary(pooled)$term == "bmi"], 2)` (see pooled table above), with standard errors reflecting both within- and between-imputation uncertainty through Rubin's rules.

16.40.10 Common pitfalls

- Imputing the outcome separately from the covariates with incompatible models.
- Using m = 5 when the fraction of missing information is high.
- Pooling means or SDs by hand instead of `pool()`.
- Dropping the outcome from the imputation model because it "feels like cheating".

16.40.11 Further reading

- van Buuren S, Groothuis-Oudshoorn K (2011), *mice: Multivariate Imputation by Chained Equations in R*.
- van Buuren S (2018), *Flexible Imputation of Missing Data*, ch. 4–6.
- White IR, Royston P, Wood AM (2011), *Multiple imputation using chained equations: issues and guidance for practice*.

16.40.12 Session info

```
sessionInfo()
```

16.40.13 See also — chapter index

- Methods in this chapter
- Find your method (Chapter 0)

16.41 Week 1, Session 1 — Observational designs and STROBE

Workflow lab using the variant template: Goal → Approach → Execution → Check → Report.

16.41.1 Learning objectives

- Distinguish cohort, case-control, and cross-sectional designs and say when each is the right tool.
- Simulate a source population and draw both a cohort sample and a nested case-control sample from it.
- Map a study report to the STROBE checklist.

16.41.2 Prerequisites

Inferential statistics and regression. Familiarity with tibbles and `glm()`.

16.41.3 Background

Observational designs differ primarily in how participants enter the study and when the outcome is measured relative to the exposure. In a cohort we sample on exposure and wait; in a case-control we sample on the outcome and look back; in a cross-sectional we sample once and measure everything together. The choice is driven by how common the outcome is, how long it takes to develop, and how easy the exposure is to measure retrospectively.

STROBE (Strengthening the Reporting of Observational Studies in Epidemiology) is the reporting checklist that keeps observational papers honest. It has 22 items covering title and abstract, introduction, methods, results, and discussion, with specific branches for the three main designs. You do not need to memorise it; you need to know that reviewers will check it.

A nested case-control sampled from a defined cohort behaves statistically like the full cohort for odds ratios that approximate risk ratios when the outcome is rare. This is the trick behind most large pharmacoepidemiology studies: a full-cohort analysis is wasteful when measurement on every unexposed person is expensive, so analysts sample controls from the risk set instead.

16.41.4 Setup

```
library(tidyverse)
set.seed(42)
theme_set(theme_minimal(base_size = 12))
```

16.41.5 1. Goal

Simulate a source population, carve a cohort study and a nested case-control study out of it, and compare the exposure–outcome association recovered by each.

16.41.6 2. Approach

We generate a population of 5000 people with a binary exposure and a binary outcome whose probability depends on exposure and a confounder (age). We then (a) analyse the full cohort and (b) draw a case-control sample with four controls per case.

```
n_pop ← 5000
pop ← tibble(
  id       = seq_len(n_pop),
  age      = rnorm(n_pop, 55, 10),
  exposure = rbinom(n_pop, 1, 0.30)
)
# outcome depends on exposure (log OR ~ 0.8) and age
lp ← -4 + 0.8 * pop$exposure + 0.04 * (pop$age - 55)
pop ← pop ▷
  mutate(prob = plogis(lp),
         outcome = rbinom(n_pop, 1, prob))
```

```
pop ▷
  count(exposure, outcome) ▷
  mutate(exposure = factor(exposure, labels = c("unexposed", "exposed")),
         outcome  = factor(outcome,  labels = c("no", "yes"))) ▷
  ggplot(aes(exposure, n, fill = outcome)) +
  geom_col(position = "dodge", alpha = 0.8) +
  labs(x = NULL, y = "Count", fill = "Outcome")
```

16.41.7 3. Execution

```
# Cohort analysis on full population
fit_cohort ← glm(outcome ~ exposure + age, data = pop, family = binomial)
coef_cohort ← broom::tidy(fit_cohort, conf.int = TRUE, exponentiate = TRUE)
coef_cohort
```

```
# Nested case-control: all cases, 4 controls per case
cases    ← pop ▷ filter(outcome == 1)
controls ← pop ▷
  filter(outcome == 0) ▷
  slice_sample(n = 4 * nrow(cases))
ncc ← bind_rows(cases, controls)
fit_ncc ← glm(outcome ~ exposure + age, data = ncc, family = binomial)
coef_ncc ← broom::tidy(fit_ncc, conf.int = TRUE, exponentiate = TRUE)
coef_ncc
```

16.41.8 4. Check

The exposure odds ratio from the nested case-control should be close to that from the cohort, because the outcome is uncommon. Precision is worse in the case-control sample but the point estimate is not biased by the sampling scheme.

```
bind_rows(
  cohort = coef_cohort |> filter(term == "exposure"),
  ncc    = coef_ncc    |> filter(term == "exposure"),
  .id = "design"
) |>
  select(design, estimate, conf.low, conf.high)
```

16.41.9 5. Report

> In a simulated source population of 5000 adults, exposure was associated with the outcome with an adjusted odds ratio of `round(coef_cohort$estimate[2], 2)` (95% CI: `round(coef_cohort$conf.low[2], 2)` to `round(coef_cohort$conf.high[2], 2)`) in a full-cohort analysis, and `round(coef_ncc$estimate[2], 2)` (95% CI: `round(coef_ncc$conf.low[2], 2)` to `round(coef_ncc$conf.high[2], 2)`) in a 1:4 nested case-control sample.

16.41.9.1 STROBE in one paragraph

Report the sampling frame, the eligibility criteria, the exposure and outcome definitions, how confounders were chosen and handled, whatever you did about missing data, and a flow diagram of who was included, excluded, and lost. If you can do those six things clearly, you have answered most of the STROBE items for free.

16.41.10 Common pitfalls

- Treating a cross-sectional association as evidence of a causal effect when temporality is unknown.
- Reporting an odds ratio as a risk ratio when the outcome is common.
- Drawing controls from a different population than the cases.
- Skipping a flow diagram in an observational paper.

16.41.11 Further reading

- von Elm E et al. (2007), *The STROBE Statement*.
- Rothman KJ, Greenland S, Lash TL. *Modern Epidemiology*, ch. 6–8.
- Pearce N (2016), *Analysis of matched case-control studies*, BMJ.

16.41.12 Session info

```
sessionInfo()
```

16.41.13 See also — chapter index

- Methods in this chapter
- Find your method (Chapter 0)

16.42 Week 4, Session 5 — Pre-registration and statistical analysis plans

Workflow lab: Goal → Approach → Execution → Check → Report.

16.42.1 Learning objectives

- Distinguish a pre-registration from a statistical analysis plan.
- Draft the core sections of each for a realistic study.
- Recognise the forms of post-hoc flexibility that pre-registration is designed to prevent.

16.42.2 Prerequisites

Earlier design and causal-inference material.

16.42.3 Background

A pre-registration is a timestamped public record of a study's hypotheses, design, and primary analysis. A statistical analysis plan (SAP) is a more detailed, often confidential companion document that specifies exactly how the data will be handled — variable definitions, handling of missing data, subgroup analyses, sensitivity analyses, and reporting conventions. Between them, these two documents make the difference between "we planned this" and "we planned this, we can prove it, and here is the file to show when it was signed."

Flexibility during analysis — *researcher degrees of freedom* — is not fraud. It is usually well-meaning curiosity. But aggregated across a field it produces a literature of spurious findings, and for any one study it produces an analysis that will not replicate. A pre-registration does not forbid curiosity. It separates *confirmatory* analyses (specified in advance) from *exploratory* ones, and requires each to be labelled when reported.

16.42.4 Setup

```
library(tidyverse)
set.seed(42)
theme_set(theme_minimal(base_size = 12))
```

16.42.5 1. Goal

Produce a minimal-but-complete pre-registration template for a fictional two-arm randomised trial, plus a SAP skeleton.

16.42.6 2. Approach

A pre-registration has four mandatory elements:

1. **Question and hypotheses** — exact form of H_0 and H_1 for each primary outcome.

2. **Design** — eligibility, allocation, blinding, intervention, follow-up.
3. **Primary analysis** — the single specification whose *p*-value will be quoted as the headline result.
4. **Sample size justification** — the calculation behind the target N.

A SAP adds:

5. **Secondary analyses** — ordered and pre-specified.
6. **Handling of missing data** — mechanism assumed, method used.
7. **Subgroup analyses** — pre-specified, limited in number.
8. **Sensitivity analyses** — especially for primary outcomes.

16.42.7 3. Execution — template

```
# Pre-registration — Trial X

Protocol version 1.0 | Date 2026-04-18 | PI [NAME]

## 1. Research question
Primary: Does [intervention] reduce [outcome] relative to [control]
in [population] over [time frame]?

## 2. Hypotheses
H0: difference in [outcome] = 0.
H1: difference in [outcome] ≠ 0. Two-sided, alpha = 0.05.

## 3. Design
Two-arm, parallel-group, double-blind, placebo-controlled trial.
1:1 allocation, block randomisation (block size 4), stratified by site.

## 4. Primary analysis
ITT. Linear regression of outcome at follow-up on arm, adjusted for
baseline value and stratification factors. Primary estimand:
adjusted mean difference with 95% CI.

## 5. Sample size
Target n = 200 per arm; power = 0.80 for d = 0.3, alpha = 0.05.

## 6. Data handling
- Missing data: multiple imputation under MAR (m = 20).
- Outliers: retained in primary analysis.
- Adherence: per-protocol analysis as sensitivity.
```

16.42.8 4. Check

Three questions to ask of any pre-registration before posting:

1. Could a reader reconstruct the primary analysis from the text without contacting you?
2. Are exploratory analyses labelled as such?
3. Is the target N justified by a calculation a reader can redo?

16.42.9 5. Report

> A pre-registration for Trial X was deposited on the OSF on [date] (DOI [doi]). The primary analysis, statistical model, and sample-size justification are specified therein. Any deviation from the pre-registered plan is reported in the Deviations section of the manuscript with its rationale.

16.42.10 Common pitfalls

- Treating pre-registration as a checkbox without a concrete analysis plan.
- Pre-registering two primary outcomes and quoting the winner.
- Failing to report deviations from plan.

16.42.11 Further reading

- Nosek BA et al. (2018). *The preregistration revolution.*
- AMRC / NIHR Statistical Analysis Plan templates.

16.42.12 Session info

```
sessionInfo()
```

16.42.13 See also — chapter index

- Methods in this chapter
- Find your method (Chapter 0)

16.43 Week 3, Session 4 — Propensity scores and IPTW

Inference lab using the five-step template: Hypothesis → Visualise → Assumptions → Conduct → Conclude.

16.43.1 Learning objectives

- Estimate a propensity score and match nearest-neighbour with `MatchIt`.
- Construct inverse-probability-of-treatment weights (IPTW) and fit a weighted outcome model.
- Diagnose covariate balance with `cobalt::love.plot`.

16.43.2 Prerequisites

DAGs and logistic regression.

16.43.3 Background

The propensity score is the probability of being treated given covariates. Under the assumptions of conditional exchangeability and positivity, conditioning on the propensity score yields an unbiased estimate of the average treatment effect. Matching and inverse-probability-of-treatment weighting are two ways to do that conditioning.

Matching pairs each treated unit with one (or more) untreated units with similar propensity scores, then analyses the matched sample. IPTW gives each unit a weight of 1/e(X) if treated and 1/(1 − e(X)) if untreated, creating a pseudo-population where treatment is independent of X. Both estimators require the propensity model to be correctly specified; neither handles unmeasured confounding.

Balance diagnostics assess whether the matched or weighted sample has covariate distributions that are similar across treatment groups. The standardised mean difference (SMD) is the workhorse metric; anything below 0.1 is usually considered balanced.

Large weights (extreme propensities near 0 or 1) are a sign of poor positivity. Stabilised weights and truncation can help, but the underlying problem — a region of covariate space where one treatment is never seen — cannot be fixed statistically.

16.43.4 Setup

```
library(tidyverse)
library(MatchIt)
library(cobalt)
set.seed(42)
theme_set(theme_minimal(base_size = 12))
```

16.43.5 1. Hypothesis

Treatment reduces the outcome by a known amount. We will estimate the effect with naive regression, matching, and IPTW, and compare with the simulated truth.

16.43.6 2. Visualise

```
n ← 1000
dat ← tibble(
  age  = rnorm(n, 60, 10),
  sev  = rnorm(n, 0,   1),
  sex  = rbinom(n, 1, 0.5)
) ▷
  mutate(ps = plogis(-1 + 0.04 * (age - 60) + 0.8 * sev + 0.3 * sex),
         trt = rbinom(n, 1, ps),
         y   = 2 - 1.5 * trt + 0.05 * (age - 60) +
                 0.8 * sev + 0.3 * sex + rnorm(n, 0, 1))

ggplot(dat, aes(ps, fill = factor(trt))) +
  geom_density(alpha = 0.5) +
  labs(x = "Propensity score", y = "Density", fill = "Treated?")
```

16.43.7 3. Assumptions

No unmeasured confounding (conditional exchangeability); positivity (everyone has a non-zero chance of each treatment); correct specification of the propensity model.

16.43.8 4. Conduct

```
# Naive regression
fit_naive ← lm(y ~ trt + age + sev + sex, data = dat)
coef(fit_naive)["trt"]
```

```
# Nearest-neighbour matching
m ← matchit(trt ~ age + sev + sex, data = dat,
            method = "nearest", ratio = 1)
m
matched ← match.data(m)

fit_match ← lm(y ~ trt, data = matched,
               weights = matched$weights)
coef(fit_match)["trt"]
```

```
# IPTW
dat ← dat ▷
  mutate(w = if_else(trt == 1, 1 / ps, 1 / (1 - ps)))
fit_iptw ← lm(y ~ trt, data = dat, weights = w)
coef(fit_iptw)["trt"]
```

```
love.plot(m, thresholds = c(m = 0.1), abs = TRUE)
```

16.43.9 5. Concluding statement

> The simulated treatment effect was −1.5. The naive regression gave `round(coef(fit_naive)["trt"], 2)`; nearest-neighbour matching gave `round(coef(fit_match)["trt"], 2)`; IPTW gave `round(coef(fit_iptw)["trt"], 2)`. Balance plots showed that matching reduced all SMDs below 0.1.

16.43.10 Common pitfalls

- Matching with replacement without adjusting inference for the repeated use of controls.
- Reporting only the *p*-value for the treatment effect in the weighted model without a sandwich or bootstrap standard error.
- Ignoring extreme weights.
- Picking the matching method that gives the effect you wanted.

16.43.11 Further reading

- Austin PC (2011), *An introduction to propensity score methods for reducing the effects of confounding in observational studies*.
- Stuart EA (2010), *Matching methods for causal inference*.

- Hernán MA, Robins JM. *Causal Inference: What If*, ch. 12.

16.43.12 Session info

```
sessionInfo()
```

16.43.13 See also — chapter index

- Methods in this chapter
- Find your method (Chapter 0)

16.44 Week 1, Session 2 — Randomised controlled trials

Inference lab using the five-step template: Hypothesis → Visualise → Assumptions → Conduct → Conclude.

16.44.1 Learning objectives

- Contrast parallel-group, crossover, cluster, and factorial RCT designs.
- Simulate a parallel-group RCT with a realistic allocation procedure.
- Interpret intention-to-treat (ITT) and per-protocol (PP) analyses and say why they can differ.

16.44.2 Prerequisites

Inferential statistics; `t.test()`, `lm()`, and basic tidyverse.

16.44.3 Background

The randomised controlled trial (RCT) is the reference design for causal inference because randomisation balances both measured and unmeasured confounders in expectation. Parallel-group RCTs assign each participant to one arm for the duration of the study. Crossover trials give each participant every treatment in a randomised order and rely on a washout period. Cluster trials randomise groups (clinics, villages) rather than individuals. Factorial trials vary more than one treatment at once and can answer more than one question per patient.

Every RCT has two analyses that will often disagree. Intention-to- treat analyses everyone in the arm they were randomised to, regardless of what they actually received. Per-protocol analyses only those who received the assigned treatment as planned. ITT is conservative for superiority and preserves the randomisation; PP is informative for efficacy in compliant patients but can be biased. Report both, and say which is primary in the protocol.

Allocation concealment — the process that keeps the next assignment unknown until the patient has been enrolled — is not the same as blinding. You can have one without the other, and you need both for the trial to protect itself from selection bias and differential

outcome ascertainment.

16.44.4 Setup

```
library(tidyverse)
set.seed(42)
theme_set(theme_minimal(base_size = 12))
```

16.44.5 1. Hypothesis

Treatment reduces the outcome (a continuous symptom score) relative to placebo. We will also see what happens when 15% of participants cross over from treatment to placebo after randomisation.

16.44.6 2. Visualise

```
n ← 200
trial ← tibble(
  id    = seq_len(n),
  arm   = sample(rep(c("placebo", "treatment"), each = n / 2)),
  # 15% of treatment arm never take the drug (will be placebo in reality)
  crossed = if_else(arm == "treatment" & runif(n) < 0.15, TRUE, FALSE),
  received = if_else(crossed, "placebo", arm),
  y = rnorm(n, mean = 50, sd = 8) +
      if_else(received == "treatment", -5, 0)
)

trial ▷
  ggplot(aes(arm, y, fill = arm)) +
  geom_boxplot(alpha = 0.6, colour = "grey30") +
  labs(x = NULL, y = "Symptom score") +
  theme(legend.position = "none")
```

16.44.7 3. Assumptions

A two-sample *t*-test on the ITT population assumes independent observations within arms and roughly normal residuals within each arm. We do not assume equal variances; `t.test()` uses Welch by default. The more important assumption is that randomisation worked — that the allocation is independent of every baseline covariate.

```
trial ▷
  group_by(arm) ▷
  summarise(mean = mean(y), sd = sd(y), n = n())
```

16.44.8 4. Conduct

```
itt ← t.test(y ~ arm, data = trial)
pp  ← t.test(y ~ received, data = trial)
itt
pp
```

The ITT estimate is closer to zero than the PP estimate because the crossed-over patients pull the treatment arm mean toward placebo.

16.44.9 5. Concluding statement

> In a simulated parallel-group RCT (*n* = 200), the ITT analysis showed a mean difference of `round(diff(itt$estimate) * -1, 1)` (95% CI: `round(-itt$conf.int[2], 1)` to `round(-itt$conf.int[1], 1)`) on the symptom score; a per-protocol analysis gave a larger estimated benefit of `round(diff(pp$estimate) * -1, 1)`, illustrating the usual direction of disagreement when non-adherence is informative.

16.44.10 Common pitfalls

- Calling a non-random allocation “randomised” because a coin was flipped after enrolment.
- Reporting only PP and calling it primary.
- Ignoring stratification variables in the analysis when they were used at randomisation.
- Cluster trials analysed as if individuals were independent.

16.44.11 Further reading

- Schulz KF, Altman DG, Moher D (2010), *CONSORT 2010 Statement*.
- Pocock SJ. *Clinical Trials: A Practical Approach*.
- Hernán MA, Hernández-Díaz S (2012), *Beyond the intention-to-treat in comparative effectiveness research*.

16.44.12 Session info

```
sessionInfo()
```

16.44.13 See also — chapter index

- Methods in this chapter
- Find your method (Chapter 0)

16.45 Week 4, Session 4 — SIR and SEIR models with deSolve

Workflow lab: Goal → Approach → Execution → Check → Report.

16.45.1 Learning objectives

- Write a compartmental SIR / SEIR model as a system of ODEs.
- Solve it with `deSolve::ode` and plot the trajectories.

- Compute the basic reproduction number R_0 from model parameters and illustrate its effect.

16.45.2 Prerequisites

Basic calculus intuition and comfort with differential-equation notation.

16.45.3 Background

Compartmental models partition a population into states — *Susceptible*, *Exposed*, *Infectious*, *Recovered* — and describe flows between them with ordinary differential equations. The SIR model has three compartments and two parameters (transmission rate β, recovery rate γ); the SEIR model adds an exposed-but-not-yet-infectious class and a progression rate σ. These models are caricatures, but they capture the shape of an epidemic — exponential rise, peak, decline — surprisingly well, and they make the role of $R_0 = \beta/\gamma$ visible.

The real power of compartmental models in practice is not prediction but counterfactual reasoning: *what if* we had vaccinated half the population on day zero, or *what if* the infectious period were a day shorter? Tuning β and γ to reflect interventions is the bread and butter of public-health modelling.

16.45.4 Setup

```
library(tidyverse)
library(deSolve)
set.seed(42)
theme_set(theme_minimal(base_size = 12))
```

16.45.5 1. Goal

Simulate an SIR outbreak in a population of 1 million, for R_0 values 1.5, 2.5, and 4.

16.45.6 2. Approach

```
sir ← function(time, state, params) {
  with(as.list(c(state, params)), {
    N  ← S + I + R
    dS ← -beta * S * I / N
    dI ←  beta * S * I / N - gamma * I
    dR ←  gamma * I
    list(c(dS, dI, dR))
  })
}
```

16.45.7 3. Execution

```
run_sir ← function(R0, gamma = 1/7, days = 180, N = 1e6, I0 = 10) {
  params ← c(beta = R0 * gamma, gamma = gamma)
  state  ← c(S = N - I0, I = I0, R = 0)
  times  ← seq(0, days, by = 1)
  out ← as.data.frame(ode(state, times, sir, params))
  out$R0 ← R0
  out
}

trajectories ← bind_rows(
  run_sir(1.5), run_sir(2.5), run_sir(4.0)
) ▷
  pivot_longer(c(S, I, R), names_to = "compartment", values_to = "n")
```

```
ggplot(trajectories, aes(time, n, colour = compartment)) +
  geom_line(linewidth = 0.8) +
  facet_wrap(~ R0, labeller = labeller(R0 = \(x) paste0("R0 = ", x))) +
  labs(x = "Day", y = "People")
```

16.45.8 4. Check

Peak prevalence and attack rate should rise with R_0.

```
trajectories ▷
  filter(compartment == "I") ▷
  group_by(R0) ▷
  summarise(peak_I = max(n), peak_day = time[which.max(n)]) ▷
  arrange(R0)
```

16.45.9 5. Report

> An SIR compartmental model was simulated for a population of one million with a 7-day mean infectious period across three basic reproduction numbers (R_0 = 1.5, 2.5, 4). Peak prevalence and the final attack rate increased sharply with R_0, illustrating how small changes in transmission translate into large changes in epidemic size. Calibrated analogues of this model underpin much real-world outbreak forecasting.

16.45.10 Common pitfalls

- Interpreting SIR trajectories as forecasts rather than scenarios.
- Forgetting that a constant β across time ignores interventions.
- Comparing peak prevalence across parameter settings without holding N and I_0 constant.

16.45.11 Further reading

- Keeling MJ, Rohani P. *Modeling Infectious Diseases in Humans and Animals.*
- Soetaert K et al. *Solving Differential Equations in R.*

16.45.12 Session info

```
sessionInfo()
```

16.45.13 See also — chapter index

- Methods in this chapter
- Find your method (Chapter 0)