Sample size planning

```
library(testflow)
```

Purpose

`testflow` provides sample-size planning functions for the trial-design problems that come up most often in practice: continuous, binary, survival, and ordinal endpoints, each under parallel, paired, or repeated-measures designs, and each for superiority, non-inferiority, or equivalence objectives.

Every planning function returns a `sample_size` object with a raw sample size, a dropout-adjusted sample size, the assumptions used, a report-ready sentence, and (where available) a plot. The underlying formulas and their literature references live in the function documentation (`?sample_size_continuous`, `?sample_size_binary`, `?sample_size_survival`, `?sample_size_ordinal`) rather than in the returned object, so this vignette walks through the same formulas alongside runnable examples.

The dispatcher `sample_size()` routes to the endpoint-specific helper based on `endpoint`, `design`, and `objective`:

```
sample_size(
  endpoint = c("continuous", "binary", "survival", "ordinal"),
  design = c("parallel", "paired", "repeated"),
  objective = c("superiority", "noninferiority", "equivalence"),
  ...
)
```

All dropout adjustments use $n_{recruit} = \lceil n_{eval}/(1-d) \rceil$, never $n_{eval}(1+d)$, since the latter under-adjusts except for very small dropout proportions.

Continuous endpoints

`sample_size_continuous()` covers parallel two-arm, paired, and repeated-measures designs.

Parallel superiority

With allocation ratio $r = n_B/n_A$, common variance σ^2, and target difference Δ:

$$n_A = \frac{(r+1)\sigma^2(z_{1-\beta} + z_{1-\alpha/2})^2}{r\Delta^2}$$

```
parallel_ss <- sample_size_continuous(
  design = "parallel",
  objective = "superiority",
  delta = 5,
  sd = 10,
  alpha = 0.05,
  power = 0.90
)
parallel_ss
#> Sample size planning
```

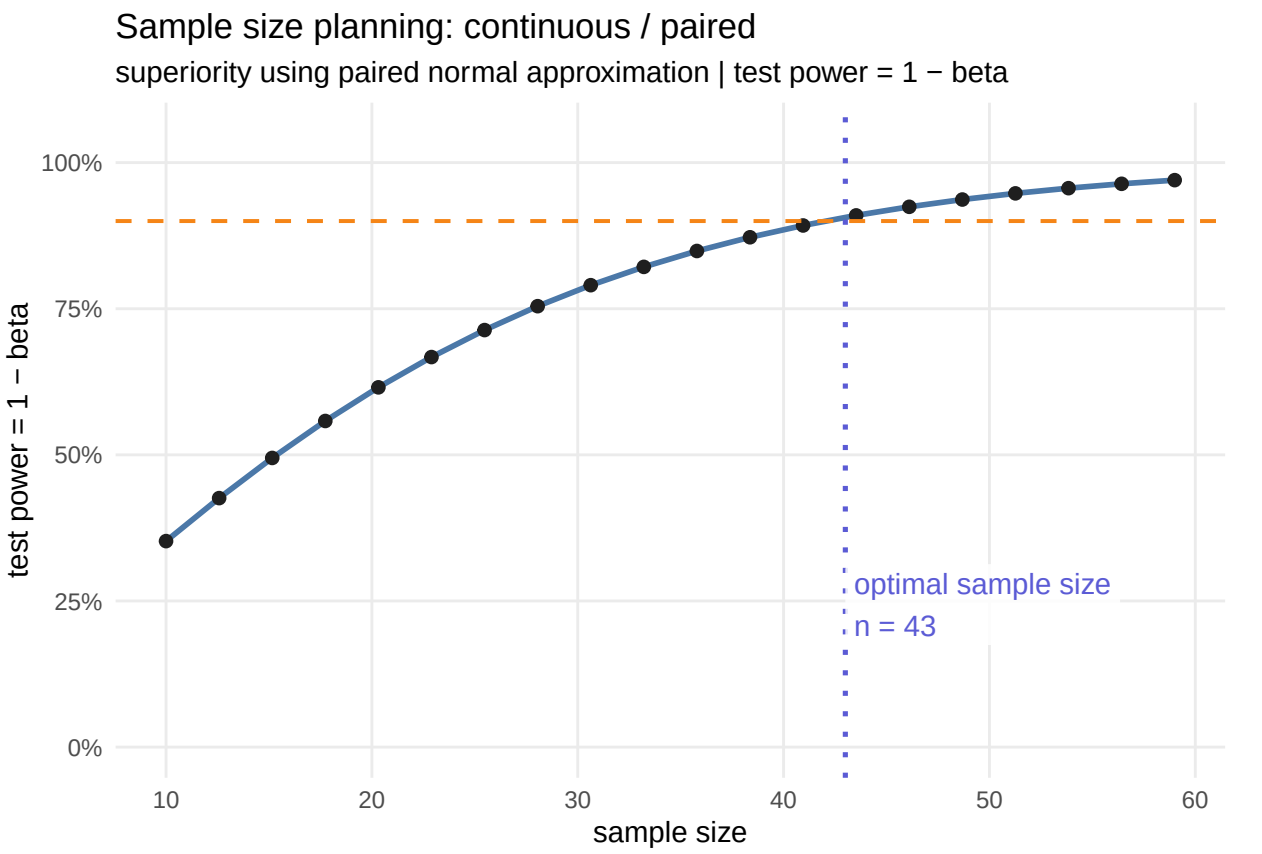
```
#>
#> Endpoint: continuous
#> Design: parallel
#> Objective: superiority
#> Method: parallel normal approximation
#>
#> Assumptions
#> * Planning note 1: assumed: Two independent groups are assumed.
#> * Planning note 2: assumed: Allocation ratio B/A = 1.000.
#>
#> Result
#> Raw n = 84.059
#> Adjusted n = A=85, B=85
#> Adjusted per-group n = A=85, B=85
#>
#> Report
#> Parallel continuous superiority sample size was calculated with delta = 5.000, sd = 10.000, allocati
```

Paired superiority

Paired designs replace σ with the standard deviation of the paired differences and drop the allocation ratio, since there is a single sample of differences:

$$n_{pairs} = \frac{(z_{1-\alpha/2} + z_{1-\beta})^2 \sigma_{diff}^2}{\Delta^2}$$

```
paired_ss <- sample_size_continuous(
  design = "paired",
  objective = "superiority",
  delta = 5,
  sd_diff = 10,
  alpha = 0.05,
  power = 0.90
)
paired_ss
#> Sample size planning
#>
#> Endpoint: continuous
#> Design: paired
#> Objective: superiority
#> Method: paired normal approximation
#>
#> Assumptions
#> * Planning note 1: assumed: Paired observations are assumed.
#> * Planning note 2: assumed: Effective SD = 10.000.
#>
#> Result
#> Raw n = 42.030
#> Adjusted n = 43
#> Adjusted total n = 43
#>
#> Report
#> Paired continuous superiority sample size was calculated with delta = 5.000, sd_diff = 10.000, alpha
plot(paired_ss, type = "curve")
```



The curve confirms that the planned sample size reaches the target power: the dashed target-power line crosses the power curve at the vertical target-n marker.

Repeated measures (3+ time points)

For `design = "repeated"` with `n_time > 2`, the paired-difference standard deviation is replaced by an effective standard deviation derived from a compound-symmetry working correlation across the repeated measurements:

$$\sigma_{eff} = \sigma_{diff}\sqrt{\frac{1 + (m-1)\rho}{m}}$$

where m is `n_time` and ρ is the within-subject `correlation`. This design-effect approach is standard for planning an average treatment-effect test across repeated, exchangeably correlated measurements; it is a `testflow` extension beyond the base paired/parallel formulas and is not itself drawn from a specific closed-form textbook example.

```
repeated_ss <- sample_size_continuous(
  design = "repeated",
  n_time = 4,
  correlation = 0.5,
  objective = "superiority",
  delta = 5,
  sd_diff = 10,
  alpha = 0.05,
  power = 0.90
```

```
)
repeated_ss
#> Sample size planning
#>
#> Endpoint: continuous
#> Design: repeated (4 time points)
#> Objective: superiority
#> Method: repeated-measures normal approximation
#>
#> Assumptions
#> * Planning note 1: assumed: Repeated measurements from the same subject are assumed (4 time points).
#> * Planning note 2: assumed: Effective SD = 7.906.
#> * Planning note 3: assumed: Within-subject correlation = 0.500.
#>
#> Result
#> Raw n = 26.269
#> Adjusted n = 27
#> Adjusted total n = 27
#>
#> Report
#> Repeated-measures continuous superiority sample size was calculated with delta = 5.000, sd_diff = 7.
```

Non-inferiority and equivalence

Non-inferiority uses a one-sided α and the distance to the null boundary $d_{NI} = \Delta + \delta$ (with δ the non-inferiority margin, passed as `delta`, and Δ the expected difference, passed as `expected_difference`):

$$n_{per\ group} = \frac{2\sigma^2(z_{1-\beta} + z_{1-\alpha})^2}{d_{NI}^2}$$

Equivalence uses the distance to the nearer of the two boundaries, $d_{EQ} = \delta - |\Delta|$, with the same one-sided form; when the expected difference is exactly zero, `testflow` switches to the standard closed-form correction that uses $z_{1-\beta/2}$ instead of $z_{1-\beta}$:

```
sample_size_continuous(
  design = "parallel",
  objective = "noninferiority",
  delta = 2,
  expected_difference = 0.5,
  sd = 10,
  alpha = 0.025,
  power = 0.90
)
#> Sample size planning
#>
#> Endpoint: continuous
#> Design: parallel
#> Objective: noninferiority
#> Method: parallel non-inferiority normal approximation
#>
#> Assumptions
#> * Planning note 1: assumed: Two independent groups are assumed.
#> * Planning note 2: assumed: Distance to the non-inferiority boundary = 2.500.
#>
```

```
#> Result
#> Raw n = 336.238
#> Adjusted n = A=337, B=337
#> Adjusted per-group n = A=337, B=337
#>
#> Report
#> Parallel continuous non-inferiority sample size was calculated with expected_difference = 0.500, mar
```

```
sample_size_continuous(
  design = "parallel",
  objective = "equivalence",
  delta = 4,
  expected_difference = 0,
  sd = 10,
  alpha = 0.05,
  power = 0.90
)
#> Sample size planning
#>
#> Endpoint: continuous
#> Design: parallel
#> Objective: equivalence
#> Method: parallel equivalence approximation
#>
#> Assumptions
#> * Planning note 1: assumed: Two independent groups are assumed.
#> * Planning note 2: assumed: Distance to the nearest equivalence boundary = 4.000.
#>
#> Result
#> Raw n = 135.277
#> Adjusted n = A=136, B=136
#> Adjusted per-group n = A=136, B=136
#>
#> Report
#> Parallel continuous equivalence sample size was calculated with expected_difference = 0.000, margin
```

Binary endpoints

`sample_size_binary()` covers parallel risk-difference planning and paired (discordant-pairs) planning.

Parallel superiority

Two variance estimators are available via `method`: `pooled` uses the null (pooled-proportion) variance, and `anticipated` uses the alternative-hypothesis variance.

$$n_A = \frac{\left[z_{1-\alpha/2}\sqrt{(1+1/r)\bar{p}(1-\bar{p})} + z_{1-\beta}\sqrt{p_A(1-p_A)+p_B(1-p_B)/r}\right]^2}{(p_A-p_B)^2}, \qquad \bar{p} = \frac{p_A + r p_B}{1+r}$$

```
sample_size_binary(
  design = "parallel",
  objective = "superiority",
  p1 = 0.4,
  p2 = 0.25,
```

```
  method = "pooled",
  alpha = 0.05,
  power = 0.90
)
#> Sample size planning
#>
#> Endpoint: binary
#> Design: parallel
#> Objective: superiority
#> Method: parallel pooled normal approximation
#>
#> Assumptions
#> * Planning note 1: assumed: Two independent groups are assumed.
#> * Planning note 2: assumed: Risk difference = 0.150.
#>
#> Result
#> Raw n = 202.810
#> Adjusted n = A=203, B=203
#> Adjusted per-group n = A=203, B=203
#>
#> Report
#> Parallel binary superiority sample size was calculated with p1 = 0.400, p2 = 0.250, allocation ratio
```

Paired (discordant-pairs) superiority

Paired binary designs plan on the discordant cells directly. With discordant odds ratio $OR_D = p_{10}/p_{01}$ and discordance rate $\lambda_D = p_{10} + p_{01}$:

$$n_{total} = \left\lceil \frac{(z_{1-\alpha/2} + z_{1-\beta})^2 (OR_D + 1)^2}{(OR_D - 1)^2 \lambda_D} \right\rceil$$

```
paired_binary_ss <- sample_size_binary(
  design = "paired",
  objective = "superiority",
  p10 = 0.2,
  p01 = 0.1,
  alpha = 0.05,
  power = 0.90
)
paired_binary_ss
#> Sample size planning
#>
#> Endpoint: binary
#> Design: paired
#> Objective: superiority
#> Method: discordant pairs approximation
#>
#> Assumptions
#> * Planning note 1: assumed: Matched pairs are assumed.
#> * Planning note 2: assumed: Discordant odds ratio = 2.000.
#> * Planning note 3: assumed: Discordance rate = 0.300.
#>
#> Result
```

```
#> Raw n = 315.223
#> Adjusted n = 316
#> Adjusted total n = 316
#>
#> Report
#> Paired binary superiority sample size was calculated with discordant_or = 2.000, discordance_rate =
```

Non-inferiority and equivalence

Both use the same asymmetric variance form as the anticipated-response superiority estimator, evaluated at the non-inferiority or equivalence distance. When `p1 == p2` for equivalence planning, the allocation ratio still enters through an `(r+1)/r` factor, matching the general two-arm special case rather than assuming equal allocation:

```
sample_size_binary(
  design = "parallel",
  objective = "equivalence",
  p1 = 0.3,
  p2 = 0.3,
  margin = 0.15,
  allocation = 2,
  alpha = 0.05,
  power = 0.90
)
#> Sample size planning
#>
#> Endpoint: binary
#> Design: parallel
#> Objective: equivalence
#> Method: parallel equivalence normal approximation
#>
#> Assumptions
#> * Planning note 1: assumed: Two independent groups are assumed.
#> * Planning note 2: assumed: Distance to the nearest equivalence boundary = 0.150.
#>
#> Result
#> Raw n = 151.510
#> Adjusted n = A=152, B=304
#> Adjusted per-group n = A=152, B=304
#>
#> Report
#> Parallel binary equivalence sample size was calculated with p1 = 0.300, p2 = 0.300, margin = 0.150;
```

Survival endpoints

`sample_size_survival()` plans on the number of events required, then converts to a total sample size using the planned survival probabilities.

Required events

For superiority under an assumed exponential hazard:

$$D_{total} = \frac{4(z_{1-\alpha/2} + z_{1-\beta})^2}{[\log(HR)]^2}$$

or, using the proportional-hazards-only approximation (`method = "ph_only"`):

$$D_{total} = \frac{2(HR+1)^2(z_{1-\alpha/2} + z_{1-\beta})^2}{(HR-1)^2}$$

Converting events to a total sample size

Given planned survival probabilities $S_A(t)$ and $S_B(t)$ at the analysis horizon, and equal allocation, the total sample size satisfying an expected D_{total} events is:

$$N_{total} = \frac{2D_{total}}{(1 - S_A(t)) + (1 - S_B(t))}$$

which follows directly from setting expected events $\frac{N}{2}(1 - S_A(t)) + \frac{N}{2}(1 - S_B(t))$ equal to D_{total} under equal-sized arms. If `survival_a`/`survival_b` are omitted, the function reports required events only and leaves `n` as the event count.

```
survival_ss <- sample_size_survival(
  hr = 0.7,
  survival_a = 0.8,
  survival_b = 0.7,
  alpha = 0.05,
  power = 0.90
)
survival_ss
#> Sample size planning
#>
#> Endpoint: survival
#> Design: parallel
#> Objective: superiority
#> Method: survival exponential event approximation
#>
#> Assumptions
#> * Planning note 1: assumed: Two independent groups are assumed.
#> * Planning note 2: assumed: Event probabilities are derived from the planned survival probabilities.
#>
#> Result
#> Raw n = 1321.512
#> Adjusted n = 1,322
#> Adjusted total n = 1,322
#> Required events = 331
#>
#> Report
#> Survival superiority sample size was calculated with hr = 0.700; required total events = 331, estima
```

Non-inferiority and equivalence work on the log-hazard-ratio scale, with distances $d_{NI} = \log(HR_M) - \log(HR)$ and $d_{EQ} = \min(\theta - \theta_L,\ \theta_U - \theta)$ respectively, both used in $D_{total} \approx 4(z_{1-\alpha} + z_{1-\beta})^2/d^2$ and converted to a total sample size the same way.

Uniform accrual

The flat conversion above assumes every subject is followed for the full study duration. When enrollment is staggered, pass `accrual_duration` (enrollment window R) and `follow_up` (additional follow-up after accrual ends) together with `survival_a`/`survival_b`; the event probability for each arm becomes the accrual-averaged

$$\bar{q}_g = 1 - \frac{e^{-\lambda_g (T-R)} - e^{-\lambda_g T}}{\lambda_g R}, \qquad T = R + \text{follow_up},$$

with λ_g implied by $S_g(T)$ under an exponential-survival assumption, in place of the flat $1 - S_g(t)$:

```
accrual_ss <- sample_size_survival(
  hr = 0.7,
  survival_a = 0.8,
  survival_b = 0.7,
  alpha = 0.05,
  power = 0.90,
  accrual_duration = 12,
  follow_up = 24
)
accrual_ss
#> Sample size planning
#>
#> Endpoint: survival
#> Design: parallel
#> Objective: superiority
#> Method: survival exponential event approximation
#>
#> Assumptions
#> * Planning note 1: assumed: Two independent groups are assumed.
#> * Planning note 2: assumed: Event probabilities are derived from the planned survival probabilities.
#> * Planning note 3: assumed: Uniform accrual over 12.000 time units, plus 24.000 additional follow-up
#>
#> Result
#> Raw n = 1550.398
#> Adjusted n = 1,551
#> Adjusted total n = 1,551
#> Required events = 331
#>
#> Report
#> Survival superiority sample size was calculated with hr = 0.700; required total events = 331, estima
```

Staggered accrual always requires *more* subjects than instantaneous accrual for the same number of events (later-enrolled subjects have less follow-up time), and the accrual-adjusted result reduces to the flat conversion as `accrual_duration` shrinks toward 0.

Ordinal endpoints

`sample_size_ordinal()` implements Noether's method for a Mann-Whitney/Wilcoxon-type comparison of two independent groups. With $P = P(A > B) + \frac{1}{2}P(A = B)$ and equal allocation:

$$n_{per\ group} = \frac{(z_{1-\alpha/2} + z_{1-\beta})^2}{6(P - 0.5)^2}$$

```
ordinal_ss <- sample_size_ordinal(
  p_superiority = 0.6,
  alpha = 0.05,
  power = 0.90
)
ordinal_ss
```

```
#> Sample size planning
#>
#> Endpoint: ordinal
#> Design: parallel
#> Objective: superiority
#> Method: Noether superiority approximation
#>
#> Assumptions
#> * Planning note 1: assumed: Two independent ordinal groups are assumed.
#>
#> Result
#> Raw n = 175.124
#> Adjusted n = A=176, B=176
#> Adjusted per-group n = A=176, B=176
#> Adjusted total n = 352
#>
#> Report
#> Ordinal superiority sample size was calculated with p_superiority = 0.600; required per-group size =
```

Bioequivalence

`sample_size_bioequivalence()` plans a two one-sided test (TOST) bioequivalence study on the log scale, for a `crossover` or `parallel` design. With anticipated geometric mean ratio GMR, $\theta_0 = \log(GMR)$, bounds $\theta_L = \log(L)$, $\theta_U = \log(U)$ (usually 0.80 and 1.25), and $d_{BE} = \min(\theta_0 - \theta_L,\ \theta_U - \theta_0)$:

```
be_ss <- sample_size_bioequivalence(
  design = "crossover",
  gmr = 0.95,
  cv_within = 0.30,
  alpha = 0.05,
  power = 0.90
)
be_ss
#> Sample size planning
#>
#> Endpoint: bioequivalence
#> Design: crossover
#> Objective: equivalence
#> Method: crossover bioequivalence (iterative_tost)
#>
#> Assumptions
#> * Planning note 1: assumed: A two-period two-treatment crossover with log-normal within-subject vari
#> * Planning note 2: assumed: Within-subject CV = 0.300.
#> * Planning note 3: assumed: Distance to the nearest bioequivalence boundary (log scale) = 0.172.
#>
#> Result
#> Raw n = 51.624
#> Adjusted n = 52
#> Adjusted total n = 52
#>
#> Report
#> Crossover bioequivalence sample size was calculated with gmr = 0.950, bounds = [0.800, 1.250]; requi
```

The default `method = "iterative_tost"` searches for the smallest sample size achieving the *exact* TOST

power, computed via the noncentral t distribution (verified against `PowerTOST::power.TOST()`, the regulatory-standard calculation, in package tests) rather than a normal approximation, since the estimated standard error carries its own sampling variability. See `?sample_size_bioequivalence` for the full formula. This is more accurate than the closed-form approximation $n_{total} \approx 2\sigma_w^2(z_{1-\beta} + z_{1-\alpha})^2/d_{BE}^2$ available via `method = "normal_approx"`; the two agree closely near $GMR = 1$ but can diverge further away from it. `design = "parallel"` uses `cv_between` in place of `cv_within` and respects the `allocation` ratio.

Precision-based sample size

`sample_size_precision()` sizes a study to a target confidence-interval half-width w instead of power against an effect size — there is no `power` argument because there is no alternative hypothesis being tested.

```
sample_size_precision(
  endpoint = "continuous",
  design = "one_sample",
  width = 2,
  sd = 10,
  alpha = 0.05
)
#> Sample size planning
#>
#> Endpoint: continuous
#> Design: one_sample
#> Objective: precision
#> Method: one-sample CI half-width
#>
#> Assumptions
#> * Planning note 1: assumed: Target CI half-width = 2.000.
#>
#> Result
#> Raw n = 96.036
#> Adjusted n = 97
#> Adjusted total n = 97
#>
#> Report
#> One-sample precision sample size was calculated with sd = 10.000, width = 2.000; required n = 97.

sample_size_precision(
  endpoint = "binary",
  design = "two_sample",
  width = 0.08,
  p1 = 0.4,
  p2 = 0.3,
  alpha = 0.05
)
#> Sample size planning
#>
#> Endpoint: binary
#> Design: two_sample
#> Objective: precision
#> Method: two-proportion CI half-width (unequal-allocation risk-difference variance)
#>
#> Assumptions
#> * Planning note 1: assumed: Target CI half-width for the risk difference = 0.080.
```

```
#> * Planning note 2: assumed: Allocation ratio n_B / n_A = 1.000; Var(p1_hat - p2_hat) = p1(1-p1)/n_A
#>
#> Result
#> Raw n = 270.103
#> Adjusted n = A=271, B=271
#> Adjusted per-group n = A=271, B=271
#> Adjusted total n = 542
#>
#> Report
#> Two-proportion precision sample size was calculated with p1 = 0.400, p2 = 0.300, width = 0.080, allo
```

`design` is `one_sample`, `two_sample`, or (binary only) `odds_ratio` (precision on the log-odds-ratio scale); `endpoint = "binary"`, `design = "one_sample"` also accepts `conservative = TRUE` to plan against the worst-case $p = 0.5$ instead of an anticipated `p` (only with the default `method = "wald"`).

Rare-prevalence planning: Wilson and exact confidence intervals

For `endpoint = "binary"`, `design = "one_sample"`, `method` chooses the confidence-interval used for planning: `"wald"` (default) is the original closed-form normal approximation, unchanged; `"wilson"` and `"exact"` (Clopper-Pearson) instead run a deterministic search for the minimum integer `n` whose confidence interval has maximum one-sided half-width no greater than `width`. This matters for rare-prevalence screening studies (e.g. newborn screening, rare adverse events), where the Wald normal approximation can be unreliable and where a naive doubling of the closed-form `n` is not guaranteed to hit the target precision.

```
# Common prevalence: Wald is adequate and remains the default
sample_size_precision(endpoint = "binary", design = "one_sample", width = 0.05, p = 0.30)
#> Warning: Wald normal-approximation intervals can be unreliable for small or
#> extreme proportions; consider method = "wilson" or method = "exact".
#> Sample size planning
#>
#> Endpoint: binary
#> Design: one_sample
#> Objective: precision
#> Method: one-proportion CI half-width (wald, expected criterion)
#>
#> Assumptions
#> * Planning note 1: assumed: Confidence interval method: normal-approximation (Wald).
#> * Planning note 2: assumed: if (identical(method, "wilson")) "The interval is evaluated at the antic
#> * Planning note 3: assumed: "Half-width" is the maximum one-sided distance from the reference propor
#> * Planning note 4: assumed: Wald planning uses a closed-form normal approximation and can be unrelia
#> * Planning note 5: assumed: Precision-based planning targets confidence-interval width, not the prob
#>
#> Result
#> Raw n = 322.683
#> Adjusted n = 323
#> Adjusted total n = 323
#>
#> Report
#> One-proportion precision planning used a 95% normal-approximation (Wald) confidence interval, antici
#>
#> Diagnostics
#> Anticipated prevalence: 0.300
#> Confidence level: 0.950
#> CI method: wald
```

```
#> Precision criterion: expected
#> Complete-case n: 323
#> Adjusted (recruitment) n: 323
#> Expected events (adjusted n): 96.900
#> P(zero events): 0.000
#> Achieved maximum half-width: 0.050
#> Note: Wald normal-approximation intervals can be unreliable for small or extreme proportions; consid

# Rare prevalence: Wilson score confidence interval
sample_size_precision(endpoint = "binary", design = "one_sample", width = 0.02, p = 0.01, method = "wils
#> Warning: Expected event or non-event count is below min_expected_events = 5;
#> rare-event asymptotics may be unreliable regardless of the confidence-interval
#> method.
#> Sample size planning
#>
#> Endpoint: binary
#> Design: one_sample
#> Objective: precision
#> Method: one-proportion CI half-width (wilson, expected criterion)
#>
#> Assumptions
#> * Planning note 1: assumed: Confidence interval method: Wilson score.
#> * Planning note 2: assumed: if (identical(method, "wilson")) "The interval is evaluated at the antic
#> * Planning note 3: assumed: "Half-width" is the maximum one-sided distance from the reference propor
#> * Planning note 4: assumed: Wilson/exact planning uses a deterministic integer numerical search for
#> * Planning note 5: assumed: Precision-based planning targets confidence-interval width, not the prob
#>
#> Result
#> Raw n = 280.000
#> Adjusted n = 280
#> Adjusted total n = 280
#>
#> Report
#> One-proportion precision planning used a 95% Wilson score confidence interval, anticipated prevalenc
#>
#> Diagnostics
#> Anticipated prevalence: 0.010
#> Confidence level: 0.950
#> CI method: wilson
#> Precision criterion: expected
#> Complete-case n: 280
#> Adjusted (recruitment) n: 280
#> Expected events (adjusted n): 2.800
#> P(zero events): 0.060
#> Achieved maximum half-width: 0.020
#> Note: Expected event or non-event count is below min_expected_events = 5; rare-event asymptotics may

# Rare prevalence: exact Clopper-Pearson, with the precision requirement
# extended over a prevalence-local band of event counts (criterion =
# "worst_case") rather than just the single anticipated count
sample_size_precision(
  endpoint = "binary", design = "one_sample", width = 0.005,
  p = 0.001, method = "exact", criterion = "worst_case"
```

```
)
#> Warning: Expected event or non-event count is below min_expected_events = 5;
#> rare-event asymptotics may be unreliable regardless of the confidence-interval
#> method.
#> Sample size planning
#>
#> Endpoint: binary
#> Design: one_sample
#> Objective: precision
#> Method: one-proportion CI half-width (exact, worst_case criterion)
#>
#> Assumptions
#> * Planning note 1: assumed: Confidence interval method: Clopper-Pearson exact.
#> * Planning note 2: assumed: Event counts are evaluated over a prevalence-local band of plausible cou
#> * Planning note 3: assumed: "Half-width" is the maximum one-sided distance from the reference propor
#> * Planning note 4: assumed: Wilson/exact planning uses a deterministic integer numerical search for
#> * Planning note 5: assumed: Clopper-Pearson exact planning guarantees at-least-nominal coverage at t
#> * Planning note 6: assumed: Precision-based planning targets confidence-interval width, not the prob
#>
#> Result
#> Raw n = 1406.000
#> Adjusted n = 1,406
#> Adjusted total n = 1,406
#>
#> Report
#> One-proportion precision planning used a 95% Clopper-Pearson exact confidence interval, anticipated
#>
#> Diagnostics
#> Anticipated prevalence: 0.001
#> Confidence level: 0.950
#> CI method: exact
#> Precision criterion: worst_case
#> Complete-case n: 1,406
#> Adjusted (recruitment) n: 1,406
#> Expected events (adjusted n): 1.406
#> P(zero events): 0.245
#> Achieved maximum half-width: 0.005
#> Note: Expected event or non-event count is below min_expected_events = 5; rare-event asymptotics may
```

`criterion = "expected"` (the default for `wilson`/`exact`) checks precision at the single anticipated event count `round(n * p)`; `criterion = "worst_case"` additionally requires the target to hold across nearby plausible event counts, which is more conservative and typically requires a somewhat larger `n`.

A confidence-interval width target is not the same planning objective as "observe at least one case": a narrow interval can still be planned around a study that, by chance, observes zero events. `sample_size_precision()` reports this explicitly via `x$diagnostics` for one-sample binary precision planning:

```
x <- sample_size_precision(
  endpoint = "binary", design = "one_sample", width = 0.02,
  p = 0.05, method = "wilson", dropout = 0.10
)
x$diagnostics$complete_case_n       # n required by the precision criterion alone
#> [1] 626
x$diagnostics$adjusted_n            # complete_case_n inflated for dropout
#> [1] 696
```

```r
x$diagnostics$expected_events       # anticipated events at adjusted_n
#> [1] 34.8
x$diagnostics$probability_zero_events
#> [1] 3.130612e-16
x$diagnostics$achieved_maximum_half_width
#> [1] 0.01998522
```

`complete_case_n` is the statistical answer to "how many complete observations achieve the requested precision"; `adjusted_n` is the recruitment target after inflating for `dropout`. Achieved-precision diagnostics (`achieved_lower`, `achieved_upper`, `achieved_maximum_half_width`) are evaluated at `complete_case_n`, since dropout does not change the precision of the completed-case analysis. `expected_events` and `probability_zero_events` are evaluated at `adjusted_n`, the actual number of participants recruited. When either the expected event count or expected non-event count falls below `min_expected_events` (default `5`), or when `method = "wald"` is used, `x$diagnostics$approximation_warning` explains why the reported precision may be optimistic, and the same message is raised as a single `warning()`.

Unequal allocation for two proportions

Binary `design = "two_sample"` precision planning now implements unequal allocation directly, rather than silently defaulting to the equal-allocation formula whenever `allocation != 1`:

```r
sample_size_precision(
  endpoint = "binary", design = "two_sample", width = 0.08,
  p1 = 0.4, p2 = 0.3, allocation = 2
)
#> Sample size planning
#>
#> Endpoint: binary
#> Design: two_sample
#> Objective: precision
#> Method: two-proportion CI half-width (unequal-allocation risk-difference variance)
#>
#> Assumptions
#> * Planning note 1: assumed: Target CI half-width for the risk difference = 0.080.
#> * Planning note 2: assumed: Allocation ratio n_B / n_A = 2.000; Var(p1_hat - p2_hat) = p1(1-p1)/n_A
#>
#> Result
#> Raw n = 207.079
#> Adjusted n = A=208, B=415
#> Adjusted per-group n = A=208, B=415
#> Adjusted total n = 623
#>
#> Report
#> Two-proportion precision sample size was calculated with p1 = 0.400, p2 = 0.300, width = 0.080, allo
```

With $r = n_B/n_A$ and $\mathrm{Var}(\hat{p}_1 - \hat{p}_2) = p_1(1-p_1)/n_A + p_2(1-p_2)/n_B$:

$$n_A = \frac{z^2_{1-\alpha/2}}{w^2}\left[p_1(1-p_1) + \frac{p_2(1-p_2)}{r}\right], \qquad n_B = r\, n_A$$

which reduces exactly to the previous equal-allocation formula when $r = 1$.

Cluster-randomized adjustment

`sample_size_cluster_adjust()` inflates an individually randomized sample size for cluster randomization by the design effect $DE = 1 + (m-1)\rho$ (or, for unevenly sized clusters with coefficient of variation CV_m,

$DE \approx 1 + ((1 + CV_m^2)m - 1)\rho$):

```
parallel_ss <- sample_size_continuous(
  design = "parallel", objective = "superiority",
  delta = 5, sd = 10, alpha = 0.05, power = 0.90
)
sample_size_cluster_adjust(unname(parallel_ss$n_adjusted["A"]), m = 20, rho = 0.02)
#> [1] 118
```

Unlike the other `sample_size_*()` functions, this one returns a plain integer rather than a `sample_size` object, since it is a post-processing adjustment applied to an already-computed sample size rather than a full planning calculation in its own right — the same shape as `sample_size_adjust_dropout()`.

Dropout adjustment and reporting

`sample_size_adjust_dropout()` exposes the dropout adjustment directly for sample sizes computed outside `testflow`:

```
sample_size_adjust_dropout(100, dropout = 0.15)
#> [1] 118
```

Every result carries a report-ready sentence and converts to a tidy one-row summary:

```
report(paired_ss)
#> [1] "Paired continuous superiority sample size was calculated with delta = 5.000, sd_diff = 10.000,
as_tibble(paired_ss)
#> # A tibble: 1 x 10
#>   endpoint   design objective   method          n n_adjusted n_per_group n_total
#>   <chr>      <chr>  <chr>       <chr>       <dbl> <chr>      <chr>       <chr>
#> 1 continuous paired superiority paired nor~  42.0 43         <NA>        43
#> # i 2 more variables: required_events <chr>, report <chr>
```

Summary of supported settings

Endpoint	Supported design(s)	Supported objective(s)	Formula family
Continuous	parallel, paired, repeated (2+ time points)	superiority, non-inferiority, equivalence	normal-approximation formulas for the mean difference, plus a compound-symmetry repeated-measures extension
Binary	parallel, paired, repeated (2 time points)	superiority, non-inferiority, equivalence	risk-difference planning and discordant-pair planning
Survival	parallel	superiority, non-inferiority, equivalence	event-based proportional-hazards planning, optional uniform-accrual adjustment
Ordinal	parallel	superiority	Noether approximation
Bioequivalence	crossover, parallel	equivalence (TOST)	iterative TOST or normal approximation, log scale
Precision	one-sample, two-sample, log-OR	CI half-width (no power/objective)	continuous and binary CI-width formulas

`sample_size_cluster_adjust()` and `sample_size_adjust_dropout()` are post-processing helpers that adjust an already-computed sample size rather than planning one from scratch.

Reference

Julious SA. *Sample Sizes for Clinical Trials*. Chapman & Hall/CRC; 2010.

Phillips KF. Power of the two one-sided tests procedure in bioequivalence. *Journal of Pharmacokinetics and Biopharmaceutics*. 1990;18(2):137-144.

Donner A, Klar N. *Design and Analysis of Cluster Randomization Trials in Health Research*. Arnold; 2000.