Sample size: complete API reference and formulas

```
library(testflow)
```

Purpose

The "Sample size planning" vignette introduces the `sample_size()` API and its main use cases. This vignette is a complete reference: every `design`/`objective`/`method` combination each planning function accepts, paired with the exact formula it evaluates, and a runnable example. Use it as a lookup table when you already know which planning function you need and want to see (or check) precisely what it computes.

Notation used throughout, matching the function documentation:

- α: type I error rate; $1 - \beta$: power.
- $z_{1-\alpha}$, $z_{1-\alpha/2}$: one- and two-sided standard normal quantiles; $z_{1-\beta}$: power quantile.
- $r = n_B/n_A$: allocation ratio (`allocation`), with $n_B = rn_A$.
- Dropout adjustment is always $n_{recruit} = \lceil n_{eval}/(1-d) \rceil$.

Superiority tests use two-sided α; non-inferiority and equivalence use one-sided α (pass `alpha = 0.025` for a one-sided-0.025 design, or keep `alpha = 0.05` for a one-sided-0.05 design). `testflow` does not rescale `alpha` for you.

`sample_size_continuous()`

```
sample_size_continuous(
  design = c("parallel", "paired", "repeated"),
  objective = c("superiority", "noninferiority", "equivalence"),
  delta = NULL, sd = NULL, sd_diff = NULL, expected_difference = 0,
  margin = NULL, alpha = 0.05, power = 0.90, allocation = 1, dropout = 0,
  n_time = 2, correlation = 0.5
)
```

`delta` doubles as the superiority effect size *or* the non-inferiority/ equivalence margin, depending on `objective`; `expected_difference` is the planned/observed difference Δ used only for non-inferiority and equivalence.

`design = "parallel"`

`objective = "superiority"`

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

```
sample_size_continuous(
  design = "parallel", objective = "superiority",
  delta = 5, sd = 10, allocation = 1, alpha = 0.05, power = 0.90
)
#> 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
```

`objective = "noninferiority"` With $d_{NI} = \Delta + \delta$ (`delta` is the margin δ), requiring $d_{NI} > 0$:

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

```
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
```

`objective = "equivalence"` General case, with $d_{EQ} = \delta - |\Delta|$, requiring $d_{EQ} > 0$:

$$n_A \approx \frac{(r+1)\sigma^2(z_{1-\beta} + z_{1-\alpha})^2}{r d_{EQ}^2}$$

```
sample_size_continuous(
  design = "parallel", objective = "equivalence",
  delta = 4, expected_difference = 1, 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 = 3.000.
#>
#> Result
#> Raw n = 190.308
#> Adjusted n = A=191, B=191
#> Adjusted per-group n = A=191, B=191
#>
#> Report
#> Parallel continuous equivalence sample size was calculated with expected_difference = 1.000, margin
```

Special case $\Delta = 0$ (`testflow` switches formulas automatically):

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

```
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
```

`design = "paired"`

Paired designs use the same three forms with the paired-difference standard deviation `sd_diff` in place of σ, and no allocation ratio (a paired analysis is a one-sample test on the differences):

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

```
sample_size_continuous(
  design = "paired", objective = "superiority",
  delta = 5, sd_diff = 10, alpha = 0.05, power = 0.90
)
#> 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

sample_size_continuous(
  design = "paired", objective = "noninferiority",
  delta = 2, expected_difference = 0.5, sd_diff = 10, alpha = 0.025, power = 0.90
)
#> Sample size planning
#>
#> Endpoint: continuous
#> Design: paired
#> Objective: noninferiority
#> Method: paired non-inferiority normal approximation
#>
#> Assumptions
#> * Planning note 1: assumed: Paired observations are assumed.
#> * Planning note 2: assumed: Effective SD = 10.000.
#> * Planning note 3: assumed: Distance to the non-inferiority boundary = 2.500.
#>
#> Result
#> Raw n = 168.119
#> Adjusted n = 169
#> Adjusted total n = 169
#>
#> Report
#> Paired continuous non-inferiority sample size was calculated with expected_difference = 0.500, margi

sample_size_continuous(
  design = "paired", objective = "equivalence",
  delta = 4, expected_difference = 0, sd_diff = 10, alpha = 0.05, power = 0.90
```

```
)
#> Sample size planning
#>
#> Endpoint: continuous
#> Design: paired
#> Objective: equivalence
#> Method: paired equivalence approximation
#>
#> Assumptions
#> * Planning note 1: assumed: Paired observations are assumed.
#> * Planning note 2: assumed: Effective SD = 10.000.
#> * Planning note 3: assumed: Distance to the nearest equivalence boundary = 4.000.
#>
#> Result
#> Raw n = 135.277
#> Adjusted n = 136
#> Adjusted total n = 136
#>
#> Report
#> Paired continuous equivalence sample size was calculated with expected_difference = 0.000, margin =
```

If `sd_diff` is omitted, `sd` is used in its place.

`design = "repeated"`

With `n_time = 2` this is identical to `design = "paired"`. With `n_time > 2`, `sd_diff` (or `sd`) is rescaled to an effective standard deviation under a compound-symmetry working correlation before being plugged into the same paired formulas above:

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

where m is `n_time` and ρ is `correlation`. This design-effect step is a `testflow` extension for planning an average treatment-effect test across exchangeably correlated repeated measurements; it is not itself a closed-form textbook formula.

```
sample_size_continuous(
  design = "repeated", n_time = 4, correlation = 0.5,
  objective = "superiority", delta = 5, sd_diff = 10, alpha = 0.05, power = 0.90
)
#> 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.

sample_size_continuous(
  design = "repeated", n_time = 4, correlation = 0.3,
  objective = "noninferiority", delta = 2, expected_difference = 0.5,
  sd_diff = 10, alpha = 0.025, power = 0.90
)
#> Sample size planning
#>
#> Endpoint: continuous
#> Design: repeated (4 time points)
#> Objective: noninferiority
#> Method: repeated-measures non-inferiority normal approximation
#>
#> Assumptions
#> * Planning note 1: assumed: Repeated measurements from the same subject are assumed (4 time points).
#> * Planning note 2: assumed: Effective SD = 6.892.
#> * Planning note 3: assumed: Within-subject correlation = 0.300.
#> * Planning note 4: assumed: Distance to the non-inferiority boundary = 2.500.
#>
#> Result
#> Raw n = 79.856
#> Adjusted n = 80
#> Adjusted total n = 80
#>
#> Report
#> Repeated-measures continuous non-inferiority sample size was calculated with expected_difference = 0

sample_size_continuous(
  design = "repeated", n_time = 4, correlation = 0.3,
  objective = "equivalence", delta = 4, expected_difference = 0,
  sd_diff = 10, alpha = 0.05, power = 0.90
)
#> Sample size planning
#>
#> Endpoint: continuous
#> Design: repeated (4 time points)
#> Objective: equivalence
#> Method: repeated-measures equivalence approximation
#>
#> Assumptions
#> * Planning note 1: assumed: Repeated measurements from the same subject are assumed (4 time points).
#> * Planning note 2: assumed: Effective SD = 6.892.
#> * Planning note 3: assumed: Within-subject correlation = 0.300.
#> * Planning note 4: assumed: Distance to the nearest equivalence boundary = 4.000.
#>
#> Result
#> Raw n = 64.257
#> Adjusted n = 65
```

```
#> Adjusted total n = 65
#>
#> Report
#> Repeated-measures continuous equivalence sample size was calculated with expected_difference = 0.000
```

sample_size_binary()

```
sample_size_binary(
  design = c("parallel", "paired", "repeated"),
  objective = c("superiority", "noninferiority", "equivalence"),
  p1, p2, margin = NULL, method = c("pooled", "anticipated"),
  discordant_or = NULL, discordance_rate = NULL, p10 = NULL, p01 = NULL,
  alpha = 0.05, power = 0.90, allocation = 1, dropout = 0, n_time = 2
)
```

design = "parallel"

objective = "superiority", method = "pooled" With pooled proportion $\bar{p} = (p_A + rp_B)/(1+r)$:

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

```
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
```

objective = "superiority", method = "anticipated"

$$n_A \approx \frac{(z_{1-\alpha/2} + z_{1-\beta})^2 \left[p_A(1-p_A) + p_B(1-p_B)/r\right]}{(p_A - p_B)^2}$$

```
sample_size_binary(
  design = "parallel", objective = "superiority",
  p1 = 0.4, p2 = 0.25, method = "anticipated", alpha = 0.05, power = 0.90
)
#> Sample size planning
#>
#> Endpoint: binary
#> Design: parallel
#> Objective: superiority
#> Method: parallel anticipated-response normal approximation
#>
#> Assumptions
#> * Planning note 1: assumed: Two independent groups are assumed.
#> * Planning note 2: assumed: Risk difference = 0.150.
#>
#> Result
#> Raw n = 199.641
#> Adjusted n = A=200, B=200
#> Adjusted per-group n = A=200, B=200
#>
#> Report
#> Parallel binary superiority sample size was calculated with p1 = 0.400, p2 = 0.250, allocation ratio
```

`objective = "noninferiority"` With $d_{NI} = (p_A - p_B) + \delta$ (`margin` is δ):

$$n_A \approx \frac{(z_{1-\alpha} + z_{1-\beta})^2 \left[p_A(1-p_A) + p_B(1-p_B)/r\right]}{d_{NI}^2}$$

```
sample_size_binary(
  design = "parallel", objective = "noninferiority",
  p1 = 0.5, p2 = 0.45, margin = 0.1, alpha = 0.025, power = 0.90
)
#> Sample size planning
#>
#> Endpoint: binary
#> 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 = 0.150.
#>
#> Result
#> Raw n = 232.331
#> Adjusted n = A=233, B=233
#> Adjusted per-group n = A=233, B=233
#>
#> Report
#> Parallel binary non-inferiority sample size was calculated with p1 = 0.500, p2 = 0.450, margin = 0.1
```

`objective = "equivalence"` General case, with $d_{EQ} = \delta - |p_A - p_B|$:

$$n_A \approx \frac{(z_{1-\alpha} + z_{1-\beta})^2 \left[p_A(1-p_A) + p_B(1-p_B)/r\right]}{d_{EQ}^2}$$

```
sample_size_binary(
  design = "parallel", objective = "equivalence",
  p1 = 0.42, p2 = 0.4, margin = 0.15, 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 = 0.130.
#>
#> Result
#> Raw n = 245.058
#> Adjusted n = A=246, B=246
#> Adjusted per-group n = A=246, B=246
#>
#> Report
#> Parallel binary equivalence sample size was calculated with p1 = 0.420, p2 = 0.400, margin = 0.150;
```

Special case $p_A = p_B = p$, which still scales with the allocation ratio `r` through an $(r+1)/r$ factor:

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

```
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;
```

`design = "paired"` / `design = "repeated"` (`n_time = 2`)

Discordant-pairs (McNemar-type) planning, entered either directly via `discordant_or`/`discordance_rate`, or via the discordant-cell probabilities `p10` (A-only) and `p01` (B-only), from which $OR_D = p_{10}/p_{01}$ and $\lambda_D = p_{10} + p_{01}$ are derived:

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

```
sample_size_binary(
  design = "paired", objective = "superiority",
  discordant_or = 2, discordance_rate = 0.3, alpha = 0.05, power = 0.90
)
#> 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 =

sample_size_binary(
  design = "paired", objective = "superiority",
  p10 = 0.2, p01 = 0.1, alpha = 0.05, power = 0.90
)
#> 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 =

sample_size_binary(
  design = "repeated", n_time = 2, objective = "superiority",
  p10 = 0.2, p01 = 0.1, alpha = 0.05, power = 0.90
)
#> Sample size planning
#>
#> Endpoint: binary
#> Design: paired (two time points)
#> 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 =
```

`design = "repeated"` for binary endpoints only supports `n_time = 2` (it is routed to the same discordant-pairs formula as `design = "paired"`); `n_time > 2` is not implemented for binary endpoints.

sample_size_survival()

```
sample_size_survival(
  design = c("parallel"),
  objective = c("superiority", "noninferiority", "equivalence"),
  hr, margin_hr = NULL, lower = NULL, upper = NULL,
  survival_a = NULL, survival_b = NULL,
  alpha = 0.05, power = 0.90, allocation = 1, dropout = 0,
  method = c("exponential", "ph_only"),
  accrual_duration = NULL, follow_up = NULL
)
```

Only `design = "parallel"` with equal allocation is implemented.

objective = "superiority", method = "exponential"

Required total events, assuming an exponential hazard:

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

```
sample_size_survival(
  hr = 0.7, method = "exponential", alpha = 0.05, power = 0.90
)
```

```
#> 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 = 330.378
#> Adjusted n = 331
#> Adjusted total n = 331
#> Required events = 331
#>
#> Report
#> Survival superiority sample size was calculated with hr = 0.700; required total events = 331, estima
```

`objective = "superiority"`, `method = "ph_only"`

Proportional-hazards-only approximation:

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

```
sample_size_survival(
  hr = 0.7, method = "ph_only", alpha = 0.05, power = 0.90
)
#> Sample size planning
#>
#> Endpoint: survival
#> Design: parallel
#> Objective: superiority
#> Method: survival ph_only 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 = 337.405
#> Adjusted n = 338
#> Adjusted total n = 338
#> Required events = 338
#>
#> Report
#> Survival superiority sample size was calculated with hr = 0.700; required total events = 338, estima
```

Converting events to a total sample size

Supplying `survival_a`/`survival_b` (planned survival probabilities at the analysis horizon) converts required events to a total sample size, under equal allocation:

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

which follows from setting expected events $\frac{N}{2}(1 - S_A(t)) + \frac{N}{2}(1 - S_B(t))$ equal to D_{total}. Without `survival_a`/`survival_b`, `n` is the required event count itself.

```
sample_size_survival(
  hr = 0.7, survival_a = 0.8, survival_b = 0.7, alpha = 0.05, power = 0.90
)
#> 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
```

Uniform accrual

Adding `accrual_duration` (R) and `follow_up` replaces the flat $1 - S_g(t)$ event probability with 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; this always requires more subjects than instantaneous accrual for the same event target, and reduces to the flat conversion as $R \to 0$:

```
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
)
#> 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
```

objective = "noninferiority"

On the log-hazard-ratio scale, with $d_{NI} = \log(HR_M) - \log(HR)$ (`margin_hr` is HR_M), requiring $d_{NI} > 0$:

$$D_{total} \approx \frac{4(z_{1-\alpha} + z_{1-\beta})^2}{d_{NI}^2}$$

```
sample_size_survival(
  hr = 0.85, objective = "noninferiority", margin_hr = 1.25,
  survival_a = 0.75, survival_b = 0.75, alpha = 0.025, power = 0.90
)
#> Sample size planning
#>
#> Endpoint: survival
#> Design: parallel
#> Objective: noninferiority
#> Method: survival non-inferiority event approximation
#>
#> Assumptions
#> * Planning note 1: assumed: Two independent groups are assumed.
#> * Planning note 2: assumed: Distance to the non-inferiority boundary = 0.386.
#>
#> Result
#> Raw n = 1130.320
#> Adjusted n = 1,131
#> Adjusted total n = 1,131
#> Required events = 283
#>
#> Report
#> Survival non-inferiority sample size was calculated with hr = 0.850, margin_hr = 1.250; required tot
```

objective = "equivalence"

With log-hazard-ratio bounds `lower`/`upper` (θ_L, θ_U), and $d_{EQ} = \min(\theta - \theta_L,\ \theta_U - \theta)$ where $\theta = \log(HR)$, requiring $d_{EQ} > 0$:

$$D_{total} \approx \frac{4(z_{1-\alpha} + z_{1-\beta})^2}{d_{EQ}^2}$$

```
sample_size_survival(
  hr = 1.0, objective = "equivalence", lower = 0.8, upper = 1.25,
  survival_a = 0.75, survival_b = 0.75, alpha = 0.05, power = 0.90
```

```
)
#> Sample size planning
#>
#> Endpoint: survival
#> Design: parallel
#> Objective: equivalence
#> Method: survival equivalence event approximation
#>
#> Assumptions
#> * Planning note 1: assumed: Two independent groups are assumed.
#> * Planning note 2: assumed: Distance to the nearest equivalence boundary = 0.223.
#>
#> Result
#> Raw n = 2751.821
#> Adjusted n = 2,752
#> Adjusted total n = 2,752
#> Required events = 688
#>
#> Report
#> Survival equivalence sample size was calculated with hr = 1.000; required total events = 688, estima
```

`sample_size_ordinal()`

```
sample_size_ordinal(
  design = c("parallel"), objective = c("superiority"),
  p_superiority = NULL,
  alpha = 0.05, power = 0.90, dropout = 0
)
```

Only `design = "parallel"` and `objective = "superiority"` are implemented.

Noether's method for two independent groups, with $P = P(A > B) + \frac{1}{2}P(A = B)$ (`p_superiority`, must exceed 0.5):

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

```
sample_size_ordinal(p_superiority = 0.65, alpha = 0.05, power = 0.90)
#> 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 = 77.833
#> Adjusted n = A=78, B=78
#> Adjusted per-group n = A=78, B=78
#> Adjusted total n = 156
```

```
#>
#> Report
#> Ordinal superiority sample size was calculated with p_superiority = 0.650; required per-group size =
```

sample_size_bioequivalence()

```
sample_size_bioequivalence(
  design = c("crossover", "parallel"), gmr = 1,
  cv_within = NULL, cv_between = NULL,
  lower = 0.80, upper = 1.25, alpha = 0.05, power = 0.90,
  allocation = 1, dropout = 0,
  method = c("iterative_tost", "normal_approx")
)
```

Two one-sided tests (TOST) on the log scale, with $\theta_0 = \log(GMR)$, $\theta_L = \log(L)$, $\theta_U = \log(U)$, and $d_{BE} = \min(\theta_0 - \theta_L,\ \theta_U - \theta_0)$ (must be positive: the anticipated GMR must lie strictly inside the bounds).

`method = "iterative_tost"` (default)

Searches for the smallest n achieving the exact TOST power, computed via the *noncentral* t distribution (Phillips 1990) - not a normal approximation, since the estimated standard error itself carries sampling variability:

$$Power(n) = T_{df,\delta_U}(-t_{1-\alpha,df}) - T_{df,\delta_L}(t_{1-\alpha,df})$$

where $T_{df,\delta}$ is the noncentral t CDF, $\delta_U = (\theta_0 - \theta_U)/SE$, $\delta_L = (\theta_0 - \theta_L)/SE$, $SE(n) = \sqrt{2\sigma_w^2/n}$ with $df = n - 2$ (crossover, total n) or $SE(n_A) = \sigma_b\sqrt{(r+1)/(rn_A)}$ with $df = n_A + n_B - 2$ (parallel, per-arm n_A), and coefficients of variation converted via $\sigma = \sqrt{\log(1 + CV^2)}$. Verified in package tests to match `PowerTOST::power.TOST()` (the regulatory-standard, Owen's-Q-based calculation) for balanced designs. An earlier version of this function used a normal approximation here, which under-sized studies by up to ~20% at small n - if you have results computed with `testflow` before this fix, re-run them.

```
sample_size_bioequivalence(
  design = "crossover", gmr = 0.95, cv_within = 0.30,
  alpha = 0.05, power = 0.90
)
#> 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

sample_size_bioequivalence(
  design = "parallel", gmr = 0.95, cv_between = 0.35, allocation = 1.5,
  alpha = 0.05, power = 0.90
)
#> Sample size planning
#>
#> Endpoint: bioequivalence
#> Design: parallel
#> Objective: equivalence
#> Method: parallel bioequivalence (iterative_tost)
#>
#> Assumptions
#> * Planning note 1: assumed: Two independent groups with log-normal between-subject variation are ass
#> * Planning note 2: assumed: Between-subject CV = 0.350.
#> * Planning note 3: assumed: Distance to the nearest bioequivalence boundary (log scale) = 0.172.
#>
#> Result
#> Raw n = 56.658
#> Adjusted n = A=57, B=85
#> Adjusted per-group n = A=57, B=85
#>
#> Report
#> Parallel bioequivalence sample size was calculated with gmr = 0.950, bounds = [0.800, 1.250]; requir
```

`method = "normal_approx"`

Closed-form approximation, switching to $z_{1-\beta/2}$ when $GMR = 1$ exactly (matching the analogous special case in `sample_size_continuous()`):

$$n_{total} \approx \frac{2\sigma_w^2 (z_{1-\beta} + z_{1-\alpha})^2}{d_{BE}^2}$$

```
sample_size_bioequivalence(
  design = "crossover", gmr = 0.95, cv_within = 0.30,
  alpha = 0.05, power = 0.90, method = "normal_approx"
)
#> Sample size planning
#>
#> Endpoint: bioequivalence
#> Design: crossover
#> Objective: equivalence
#> Method: crossover bioequivalence (normal_approx)
#>
#> 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 = 49.980
```

```
#> Adjusted n = 50
#> Adjusted total n = 50
#>
#> Report
#> Crossover bioequivalence sample size was calculated with gmr = 0.950, bounds = [0.800, 1.250]; requi
```

`iterative_tost` is preferred: the two methods agree closely near $GMR = 1$ but can diverge further away from it.

sample_size_precision()

```
sample_size_precision(
  endpoint = c("continuous", "binary"),
  design = c("one_sample", "two_sample", "odds_ratio"),
  width, sd = NULL, p = NULL, p1 = NULL, p2 = NULL,
  alpha = 0.05, allocation = 1, dropout = 0, conservative = FALSE,
  method = c("wald", "wilson", "exact"),
  criterion = c("expected", "worst_case"),
  min_expected_events = 5, max_n = 1e7
)
```

Driven by a target CI half-width `width` rather than power; there is no `power` argument. `design = "odds_ratio"` is binary-only.

```
sample_size_precision(endpoint = "continuous", design = "one_sample", width = 2, sd = 10)
#> 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 = "continuous", design = "two_sample", width = 2, sd = 10, allocation =
#> Sample size planning
#>
#> Endpoint: continuous
#> Design: two_sample
#> Objective: precision
#> Method: two-sample CI half-width
#>
#> Assumptions
#> * Planning note 1: assumed: Target CI half-width for the mean difference = 2.000.
#>
#> Result
```

```
#> Raw n = 160.061
#> Adjusted n = A=161, B=241
#> Adjusted per-group n = A=161, B=241
#>
#> Report
#> Two-sample precision sample size was calculated with sd = 10.000, width = 2.000; required per-group
sample_size_precision(endpoint = "binary", design = "one_sample", width = 0.05, p = 0.3)
#> 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
sample_size_precision(endpoint = "binary", design = "one_sample", width = 0.05, conservative = TRUE)
#> 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, conservative p = 0.5, 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 = 384.146
#> Adjusted n = 385
#> Adjusted total n = 385
#>
#> Report
#> One-proportion precision planning used a 95% normal-approximation (Wald) confidence interval, a cons
#>
#> Diagnostics
#> Anticipated prevalence: 0.500
#> Confidence level: 0.950
#> CI method: wald
#> Precision criterion: expected
#> Complete-case n: 385
#> Adjusted (recruitment) n: 385
#> Expected events (adjusted n): 192.500
#> 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
sample_size_precision(endpoint = "binary", design = "two_sample", width = 0.08, p1 = 0.4, p2 = 0.3, allo
#> 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
sample_size_precision(endpoint = "binary", design = "odds_ratio", width = 0.3, p1 = 0.4, p2 = 0.3, allo
#> Sample size planning
#>
#> Endpoint: binary
#> Design: odds_ratio
#> Objective: precision
#> Method: log-odds-ratio CI half-width
#>
```

```
#> Assumptions
#> * Planning note 1: assumed: Target CI half-width on the log-odds-ratio scale = 0.300.
#>
#> Result
#> Raw n = 279.471
#> Adjusted n = A=280, B=559
#> Adjusted per-group n = A=280, B=559
#>
#> Report
#> Log-odds-ratio precision sample size was calculated with p1 = 0.400, p2 = 0.300, width = 0.300; requ
```

`allocation` is respected for every design here: `two_sample` (continuous), binary `two_sample` (risk-difference half-width, unequal-allocation variance - this used to be equal-allocation only), and `odds_ratio`.

`method`, `criterion`, `min_expected_events`, and `max_n` apply only to `endpoint = "binary"`, `design = "one_sample"`. `method = "wald"` (the default) preserves the original closed-form formula exactly; `"wilson"` and `"exact"` instead search for the minimum integer `n` whose confidence interval has maximum one-sided half-width no greater than `width`, which matters for rare-prevalence planning (e.g. newborn-screening or rare-event screening) where the normal approximation underlying Wald intervals can be unreliable:

```
# Wald (backward-compatible default)
sample_size_precision(endpoint = "binary", design = "one_sample", width = 0.02, p = 0.01)
#> 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. 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 = 95.076
#> Adjusted n = 96
#> Adjusted total n = 96
#>
#> Report
#> One-proportion precision planning used a 95% normal-approximation (Wald) confidence interval, antici
#>
#> Diagnostics
#> Anticipated prevalence: 0.010
#> Confidence level: 0.950
#> CI method: wald
#> Precision criterion: expected
#> Complete-case n: 96
#> Adjusted (recruitment) n: 96
```

```
#> Expected events (adjusted n): 0.960
#> P(zero events): 0.381
#> Achieved maximum half-width: 0.020
#> Note: Expected event or non-event count is below min_expected_events = 5; rare-event asymptotics may

# Wilson score, iterative search - generally a smaller, better-calibrated n
# for rare p than Wald
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

# Exact Clopper-Pearson, requiring the precision target to hold over a
# prevalence-local band of event counts rather than just the anticipated one
x <- 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.
x$diagnostics
#> $anticipated_prevalence
#> [1] 0.001
#>
#> $expected_events_raw
#> [1] 1.406
#>
#> $expected_events
#> [1] 1.406
#>
#> $expected_non_events
#> [1] 1404.594
#>
#> $probability_zero_events
#> [1] 0.2449494
#>
#> $probability_at_least_one_event
#> [1] 0.7550506
#>
#> $requested_half_width
#> [1] 0.005
#>
#> $achieved_lower
#> [1] 1.800681e-05
#>
#> $achieved_upper
#> [1] 0.003956326
#>
#> $achieved_lower_half_width
#> [1] 0.0006932307
#>
#> $achieved_upper_half_width
#> [1] 0.003245088
#>
#> $achieved_maximum_half_width
#> [1] 0.004997661
#>
#> $achieved_total_width
#> [1] 0.003938319
#>
#> $confidence_level
#> [1] 0.95
#>
#> $ci_method
#> [1] "exact"
#>
#> $precision_criterion
#> [1] "worst_case"
#>
#> $complete_case_n
#> [1] 1406
#>
```

```
#> $adjusted_n
#> [1] 1406
#>
#> $dropout
#> [1] 0
#>
#> $approximation_warning
#> [1] "Expected event or non-event count is below min_expected_events = 5; rare-event asymptotics may
```

`x$diagnostics` (populated only for `endpoint = "binary"`, `design = "one_sample"`) reports rare-event planning quantities that a CI-width target alone does not guarantee: `expected_events`, `probability_zero_events`, `probability_at_least_one_event`, and the achieved precision (`achieved_lower`/`achieved_up` at the `complete_case_n` before dropout inflation, separately from `adjusted_n` after it. A narrow planned interval does not, by itself, ensure that any events will be observed - see `?sample_size_precision` for details.

sample_size_cluster_adjust()

```
sample_size_cluster_adjust(n_ind, m, rho, cv_m = NULL)
```

Inflates an individually randomized sample size `n_ind` by the design effect $DE = 1 + (m-1)\rho$ for equal cluster size m and intracluster correlation ρ, or $DE \approx 1 + ((1 + CV_m^2)m - 1)\rho$ when cluster sizes vary (`cv_m`, their coefficient of variation):

```
sample_size_cluster_adjust(100, m = 20, rho = 0.02)
#> [1] 138
sample_size_cluster_adjust(100, m = 20, rho = 0.02, cv_m = 0.3)
#> [1] 142
```

Like `sample_size_adjust_dropout()`, this returns a plain integer, not a `sample_size` object — it is a post-processing adjustment, not a full planning calculation.

sample_size() dispatcher

`sample_size()` routes to the four functions above by `endpoint`:

```
sample_size(
  endpoint = "binary", design = "parallel", objective = "noninferiority",
  p1 = 0.5, p2 = 0.45, margin = 0.1, alpha = 0.025, power = 0.90
)
#> Sample size planning
#>
#> Endpoint: binary
#> 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 = 0.150.
#>
#> Result
#> Raw n = 232.331
#> Adjusted n = A=233, B=233
#> Adjusted per-group n = A=233, B=233
```

```
#>
#> Report
#> Parallel binary non-inferiority sample size was calculated with p1 = 0.500, p2 = 0.450, margin = 0.1
```

`sample_size_adjust_dropout()`

Exposes the dropout adjustment directly, independent of the planning functions above:

$$n_{recruit} = \left\lceil \frac{n_{eval}}{1 - d} \right\rceil$$

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

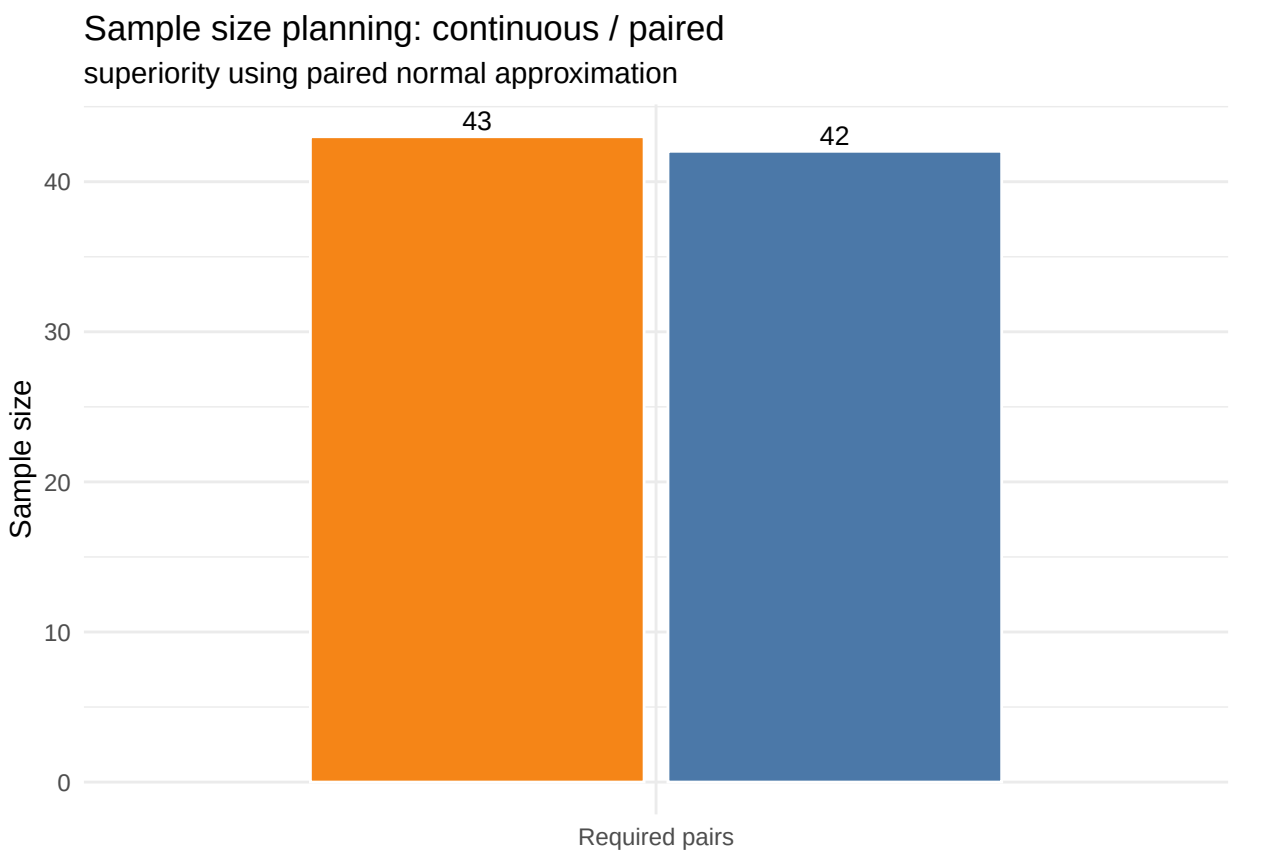
Methods on `sample_size` objects

Every call above returns a `sample_size` object with the following methods:

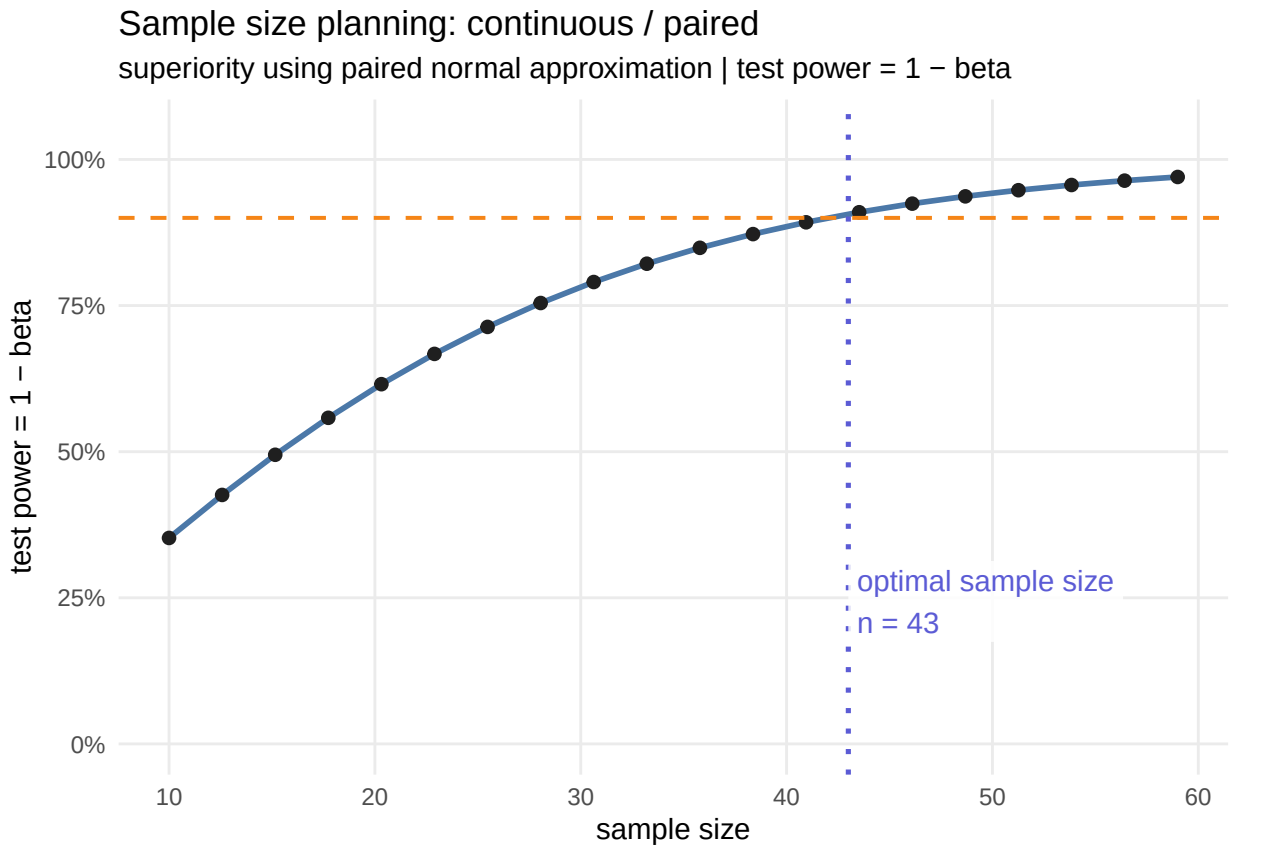
```
x <- sample_size_continuous(
  design = "paired", objective = "superiority",
  delta = 5, sd_diff = 10, alpha = 0.05, power = 0.90
)

x                       # print.sample_size(): formatted console report
#> 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
summary(x)              # summary.sample_size(): compact summary list
#> sample size summary
#>
#> Endpoint: continuous
#> Design: paired
#> Objective: superiority
#> Method: paired normal approximation
#> Raw n: 42.030
#> Adjusted n: 43
#> Adjusted total n: 43
#>
#> Report
```

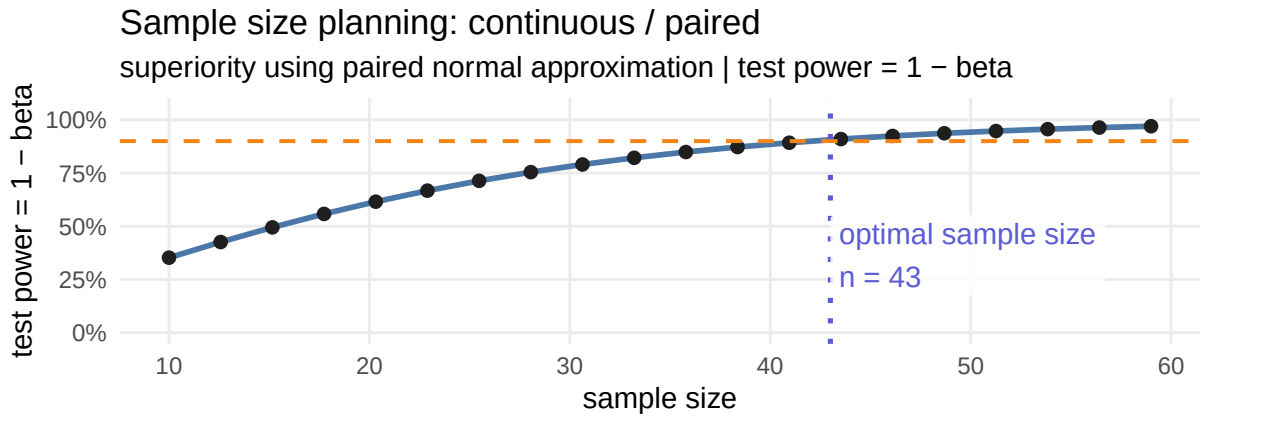
```
#> Paired continuous superiority sample size was calculated with delta = 5.000, sd_diff = 10.000, alpha
report(x)               # report.sample_size(): report-ready sentence
#> [1] "Paired continuous superiority sample size was calculated with delta = 5.000, sd_diff = 10.000,
as_tibble(x)            # as_tibble.sample_size(): one-row tidy summary
#> # 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>
plot(x, type = "summary") # raw vs. dropout-adjusted n bar chart
```

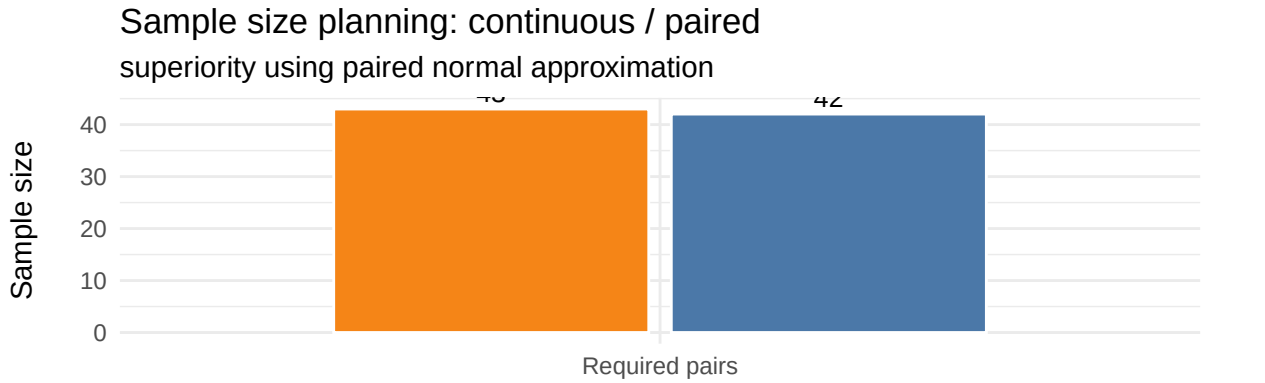



```
plot(x, type = "curve")   # power curve, when curve data is available
```




```
plot(x, type = "both")     # both plots stacked (requires the patchwork package)
```





`curve_data` (and therefore `plot(x, type = "curve"/"both")`) is currently only populated for `sample_size_continuous(design = "paired"/"repeated", objective = "superiority")`; other endpoint/design/objective combinations return a `NULL` curve and fall back to the summary bar chart.

None of these objects carry a `formula` or `reference` field. The formula for each specific call is documented on its function's help page (`?sample_size_continuous`, `?sample_size_binary`, `?sample_size_survival`, `?sample_size_ordinal`, `?sample_size_bioequivalence`, `?sample_size_precision`, `?sample_size_cluster_adjust`) and reproduced above.

References

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.