

# Package: powerbrmsINLA (via r-universe)

July 2, 2026

**Title** Bayesian Power Analysis Using 'brms' and 'INLA'

**Version** 1.3.0

**Maintainer** Tony Myers <admyers@aol.com>

**Description** Provides tools for Bayesian power analysis and assurance calculations using the statistical frameworks of 'brms' and 'INLA'. Includes simulation-based approaches, support for multiple decision rules (direction, threshold, ROPE), sequential designs, and visualisation helpers. Methods are based on Kruschke (2014, ISBN:9780124058880) ``Doing Bayesian Data Analysis: A Tutorial with R, JAGS, and Stan'', O'Hagan & Stevens (2001) <doi:10.1177/0272989X0102100307> ``Bayesian Assessment of Sample Size for Clinical Trials of Cost-Effectiveness'', Kruschke (2018) <doi:10.1177/2515245918771304> ``Rejecting or Accepting Parameter Values in Bayesian Estimation'', Rue et al. (2009) <doi:10.1111/j.1467-9868.2008.00700.x> ``Approximate Bayesian inference for latent Gaussian models by using integrated nested Laplace approximations'', and Bürkner (2017) <doi:10.18637/jss.v080.i01> ``brms: An R Package for Bayesian Multilevel Models using Stan''.

**License** MIT + file LICENSE

**Encoding** UTF-8

**RoxygenNote** 7.3.2

**Depends** R (>= 4.1.0)

**Imports** brms (>= 2.19.0), dplyr (>= 1.1.0), ggplot2 (>= 3.4.0), pbapply, rlang (>= 1.1.0), tibble (>= 3.2.0), scales (>= 1.2.0), viridisLite (>= 0.4.0), stats, tools, utils, magrittr (>= 2.0.0)

**Suggests** INLA (>= 22.05.07), testthat (>= 3.0.0), rmarkdown, knitr, MASS, circular, sn

**VignetteBuilder** knitr

**URL** <https://github.com/Tony-Myers/powerbrmsINLA>

**BugReports** <https://github.com/Tony-Myers/powerbrmsINLA/issues>

**Additional\_repositories** <https://inla.r-inla-download.org/R/stable>

**NeedsCompilation** no

**Author** Tony Myers [aut, cre] (ORCID:  
<<https://orcid.org/0000-0003-4516-4829>>)

**Repository** <https://cran.r-universe.dev>

**Date/Publication** 2026-07-02 10:40:02 UTC

**RemoteUrl** <https://github.com/cran/powerbrmsINLA>

**RemoteRef** HEAD

**RemoteSha** 88fcb54eff6ad4f8635ce529fe69e4f33f6d7c0b

## Contents

add_decision_overlay . . . . .	3
assurance_prior_weights . . . . .	4
beta_binom_power . . . . .	5
beta_weights_on_grid . . . . .	5
brms_inla_power . . . . .	6
brms_inla_power_design . . . . .	9
brms_inla_power_parallel . . . . .	10
brms_inla_power_sequential . . . . .	11
brms_inla_power_two_stage . . . . .	13
brms_inla_sequential_trial . . . . .	15
compute_assurance . . . . .	19
decide_sample_size . . . . .	21
hdi_of_icdf . . . . .	24
min_n_beta_binom . . . . .	25
plot_assurance_curve . . . . .	26
plot_assurance_with_robustness . . . . .	27
plot_bf_assurance_curve . . . . .	28
plot_bf_assurance_curve_smooth . . . . .	28
plot_bf_expected_evidence . . . . .	29
plot_bf_heatmap . . . . .	30
plot_decision_assurance_curve . . . . .	31
plot_decision_threshold_contour . . . . .	32
plot_design_prior . . . . .	33
plot_interaction_surface . . . . .	34
plot_power_assurance_overlay . . . . .	35
plot_power_contour . . . . .	36
plot_power_heatmap . . . . .	37
plot_precision_assurance_curve . . . . .	38
plot_precision_fan_chart . . . . .	39
plot_sequential_monitor . . . . .	40
print.brms_inla_power . . . . .	40
print.powerbrmsINLA_assurance . . . . .	41

*add\_decision\_overlay* 3

<code>print.powerbrmsINLA_sample_size</code>	42
<code>print.powerbrmsINLA_seq_design</code>	42
<code>print.powerbrmsINLA_seq_monitor</code>	43
<code>print.powerbrmsINLA_seq_trial</code>	43
<code>sequential_analysis</code>	44
<code>sequential_design</code>	45
<code>validate_inla_vs_brms</code>	47
<code>validate_sd_spec</code>	49

**Index** 50

---

`add_decision_overlay` *Add sample-size decision overlay to a conditional power contour*

---

## Description

Overlays the output of `decide_sample_size()` as a step-line on a contour plot (e.g. from `plot_power_contour()`).

## Usage

```
add_decision_overlay(p, decisions, x_effect = NULL, colour = "red")
```

## Arguments

<code>p</code>	A ggplot object, typically from <code>plot_power_contour()</code> .
<code>decisions</code>	A data.frame or tibble returned by <code>decide_sample_size()</code> , containing at least one effect column and <code>n_recommended</code> .
<code>x_effect</code>	Name of the effect column to use on the x-axis. If NULL, the first column that is not <code>n_recommended</code> or <code>rationale</code> is used.
<code>colour</code>	Colour for the overlay line (default "red").

## Details

If the decisions contain multiple effect-grid columns, the overlay is aggregated over all effect columns except `x_effect` by taking the worst-case (maximum) recommended `n` at each value of `x_effect`.

## Value

A ggplot object.

assurance\_prior\_weights

*Create prior weights over an effect grid for use with compute\_assurance()*

---

## Description

A convenience wrapper that evaluates a parametric prior distribution at each value in effects, normalises the densities to sum to 1, and returns a named numeric vector that can be passed directly to [compute\\_assurance\(\)](#) as prior\_weights.

## Usage

```
assurance_prior_weights(effects, dist = c("normal", "uniform", "beta"), ...)
```

## Arguments

effects	Numeric vector of effect-grid values (the same vector passed to <a href="#">brms_inla_power()</a> as effect_grid).
dist	Character string naming the distribution. One of "normal", "uniform", or "beta".
...	Named distribution parameters forwarded to the internal density calculator. See <a href="#">compute_assurance()</a> @details for the full list per distribution.

## Details

The output format is identical to that of [beta\\_weights\\_on\\_grid\(\)](#), which can also be used directly when a beta prior is appropriate.

## Value

Named numeric vector of normalised weights (sums to 1), with names equal to `as.character(effects)`.

## Examples

```
effects <- c(0.1, 0.3, 0.5, 0.7, 0.9)
# Normal prior centred on 0.5
assurance_prior_weights(effects, dist = "normal", mean = 0.5, sd = 0.2)
# Uniform prior (equal weight)
assurance_prior_weights(effects, dist = "uniform")
```

---

beta_binom_power	<i>Analytic Assurance for Beta-Binomial Designs</i>
------------------	---

---

**Description**

Computes assurance (power) using generating and audience Beta priors for a binomial count via a Beta-Binomial predictive distribution.

**Usage**

```
beta_binom_power(
  n,
  gen_prior_a,
  gen_prior_b,
  aud_prior_a,
  aud_prior_b,
  hdi_mass = 0.95,
  rope = NULL,
  hdi_max_width = NULL
)
```

**Arguments**

n	Sample size (number of trials).
gen_prior_a, gen_prior_b	Generating Beta prior parameters.
aud_prior_a, aud_prior_b	Audience Beta prior parameters.
hdi_mass	HDI mass (e.g., 0.95).
rope	Length-2 numeric vector for ROPE bounds, or NULL for max-width rule.
hdi_max_width	Positive width threshold for the HDI (used if rope=NULL).

**Value**

Assurance value between 0 and 1.

---

beta_weights_on_grid	<i>Beta-Prior Weights Over an Effect Grid</i>
----------------------	---

---

**Description**

Computes prior weights over a grid of true effect values by evaluating a Beta(mode, n) prior. If the grid is not in (0,1), it is rescaled linearly.

**Usage**

```
beta_weights_on_grid(effects, mode, n)
```

**Arguments**

effects	Numeric vector of effect values (grid).
mode	Prior mode in (0,1).
n	Prior concentration (> 2).

**Value**

Normalised numeric weights over the grid (sum to 1).

---

brms_inla_power	<i>Core Bayesian Assurance / Power Simulation (Modern, Multi-Effect Ready)</i>
-----------------	--

---

**Description**

Provides Bayesian power analysis and assurance calculation using INLA (Integrated Nested Laplace Approximation) for efficient computation. Implements simulation-based power analysis for generalized linear mixed models with automatic threading optimization.

**Usage**

```
brms_inla_power(
  formula,
  family = gaussian(),
  family_control = NULL,
  Ntrials = NULL,
  E = NULL,
  scale = NULL,
  priors = NULL,
  data_generator = NULL,
  effect_name,
  effect_grid = 0.5,
  sample_sizes = c(50, 100, 200, 400),
  nsims = 200,
  power_threshold = 0.8,
  precision_target = NULL,
  prob_threshold = 0.95,
  effect_threshold = 0,
  credible_level = 0.95,
  rope_bounds = NULL,
  error_sd = 1,
  group_sd = 0.5,
```

```

  obs_per_group = 10,
  predictor_means = NULL,
  predictor_sds = NULL,
  seed = 123,
  inla_hyper = NULL,
  compute_bayes_factor = FALSE,
  bf_method = c("sd", "marglik"),
  bf_cutoff = 10,
  inla_num_threads = NULL,
  progress = c("auto", "text", "none"),
  family_args = list()
)

```

### Arguments

formula	Model formula.
family	brms GLM family (e.g., gaussian(), binomial()).
family_control	Optional list for INLA's control.family.
Ntrials	Optional vector for binomial trials.
E	Optional vector for Poisson exposure.
scale	Optional vector scale parameter for INLA families.
priors	Optional brms::prior specification. Supported analysis priors are translated to INLA controls where possible and recorded in settings\$prior_translation.
data_generator	Optional function(n, effect) returning a dataset.
effect_name	Character vector of fixed effect names.
effect_grid	Vector/data.frame of effect values (supports multi-effect). For single effects, use a numeric vector. For multiple effects, use a data.frame with column names matching effect_name.
sample_sizes	Vector of sample sizes.
nsims	Number of simulations per cell.
power_threshold	Decision probability threshold for summary.
precision_target	Optional credible interval width target.
prob_threshold	Posterior probability threshold for decision rules.
effect_threshold	Effect-size threshold.
credible_level	Credible interval level (default 0.95).
rope_bounds	Optional Region of Practical Equivalence bounds (length 2 vector).
error_sd	Residual standard deviation for Gaussian-like families. Accepts either: <ul style="list-style-type: none"> <li>• A positive numeric scalar (default 1), or</li> <li>• A named list specifying a distribution from which a fresh value is drawn at every simulation iteration:</li> </ul>

- `list(dist = "halfnormal", sd = X, location = Y)` — draws  $|Normal(location, sd)|$ ; location defaults to 0.
- `list(dist = "lognormal", meanlog = X, sdlog = Y)`
- `list(dist = "uniform", min = X, max = Y)` — requires  $min \geq 0$ . See [validate\\_sd\\_spec\(\)](#) for validation details.

group_sd	Random-effects standard deviation. Accepts the same scalar or distributional list formats as error_sd.
obs_per_group	Observations per group.
predictor_means	Optional named list of predictor means.
predictor_sds	Optional named list of predictor standard deviations.
seed	Random seed.
inla_hyper	Optional INLA-specific hyperparameters.
compute_bayes_factor	Logical, compute Bayes Factor if TRUE.
bf_method	Character. "sd" = Savage-Dickey at 0 (requires proper Normal prior on the tested coefficient); "marglik" = marginal-likelihood Bayes factor via INLA by comparing full vs reduced model (slower).
bf_cutoff	Numeric Bayes-factor threshold for declaring a "hit" (default 10).
inla_num_threads	Character string specifying INLA threading (e.g., "4:1" for 4 threads). If NULL (default), automatically detects optimal setting: "4:1" for 4+ cores, "2:1" for 2-3 cores, "1:1" otherwise.
progress	One of "auto", "text", or "none" for progress display.
family_args	List of arguments for family-specific data generators.

## Details

### Variance uncertainty (distributional error\_sd / group\_sd):

When error\_sd or group\_sd is supplied as a distributional list, a fresh scalar value is drawn from the specified distribution at the start of **each** simulation iteration. The drawn value is used by the automatic data generator for that iteration and stored in the per-simulation results as sampled\_error\_sd or sampled\_group\_sd. The per-cell summary then reports the mean and standard deviation of the drawn values.

This corresponds to integrating power over variance uncertainty, analogous to the unconditional Bayesian assurance of O'Hagan & Stevens (2001) and the prior-predictive power framing of Chen et al. (2018). It is particularly useful when the residual variance is itself uncertain (e.g., estimated from a small pilot study).

Note: distributional error\_sd / group\_sd specifications only affect the built-in automatic data generator. When a custom data\_generator function is supplied the drawn values are recorded but are **not** injected into the custom function.

### brms prior translation:

The priors argument is a convenience interface for selected brms-style analysis priors. Gaussian fixed-effect and intercept priors are translated directly to INLA control.fixed; student\_t()

fixed-effect priors use the package's existing Normal approximation. For Gaussian models, `prior(exponential(rate), class = "sigma")` and `prior(exponential(rate), class = "sd", group = "...")` are translated to INLA `pc.prec` priors with  $P(\text{sigma} > u) = \alpha$ , where  $u = -\log(\alpha) / \text{rate}$  and  $\alpha = 0.05$ . Direct `family_control` settings take precedence over translated sigma priors. Unsupported brms priors are reported in `settings$prior_translation` rather than silently translated. Data-generation controls such as `error_sd` and `group_sd` remain distinct from INLA analysis priors.

## Value

List with results, summary, diagnostics, and settings.

## Examples

```
## Not run:
# Integrate over uncertainty in the residual SD using a half-normal prior
# centred at 1.0 with spread 0.3
brms_inla_power(
  formula      = y ~ treatment,
  effect_name  = "treatment",
  effect_grid  = 0.5,
  sample_sizes = c(50, 100),
  nsims       = 50,
  error_sd    = list(dist = "halfnormal", sd = 0.3, location = 1.0),
  seed        = 42
)
## End(Not run)
```

---

brms\_inla\_power\_design

*Design-based wrapper for Bayesian power / assurance*

---

## Description

Dispatches to one of the three engines depending on design. This function must accept `...` and pass it on unchanged.

## Usage

```
brms_inla_power_design(design = c("fixed", "two_stage", "sequential"), ...)
```

## Arguments

`design` Character scalar: "fixed", "two\_stage", or "sequential".  
`...` Arguments passed on to the corresponding engine.

**Value**

Whatever the underlying engine returns.

---

```
brms_inla_power_parallel
```

*Parallel wrapper for fixed-design Bayesian power / assurance simulations*

---

**Description**

Parallelises over cells defined by `sample_sizes` x `effect_grid` for the fixed-n engine `brms_inla_power()`.

**Usage**

```
brms_inla_power_parallel(
  design = c("fixed"),
  sample_sizes,
  effect_grid,
  nsims,
  n_cores = max(1L, parallel::detectCores() - 1L),
  seed = 123L,
  progress = c("auto", "text", "none"),
  ...
)
```

**Arguments**

<code>design</code>	Character scalar. Currently only "fixed" is supported.
<code>sample_sizes</code>	Numeric vector of sample sizes (required).
<code>effect_grid</code>	Numeric vector or data frame defining effect scenarios (required).
<code>nsims</code>	Integer number of simulations per cell.
<code>n_cores</code>	Integer number of worker processes. Default is <code>max(1L, parallel::detectCores() - 1L)</code> .
<code>seed</code>	Integer base seed. Each cell uses <code>seed + cell_id</code> .
<code>progress</code>	Logical or character; controls wrapper-level progress bar.
<code>...</code>	Further arguments passed directly to <code>brms_inla_power()</code> , such as <code>formula</code> , <code>family</code> , <code>priors</code> , <code>effect_name</code> , <code>compute_bayes_factor</code> , <code>bf_method</code> , <code>inla_hyper</code> , <code>inla_num_threads</code> , etc.

**Value**

A list with components `summary`, `results`, and `settings`.

---

`brms_inla_power_sequential`*Sequential Bayesian Assurance Simulation Engine (Modern, Multi-Effect Ready)*

---

**Description**

Simulates assurance sequentially in batches, stopping early per cell based on Wilson confidence intervals.

**Usage**

```
brms_inla_power_sequential(  
  formula,  
  family = gaussian(),  
  family_control = NULL,  
  Ntrials = NULL,  
  E = NULL,  
  scale = NULL,  
  priors = NULL,  
  data_generator = NULL,  
  effect_name,  
  effect_grid,  
  sample_sizes,  
  metric = c("direction", "threshold", "rope", "bf"),  
  target = 0.8,  
  prob_threshold = 0.95,  
  effect_threshold = 0,  
  rope_bounds = NULL,  
  credible_level = 0.95,  
  compute_bayes_factor = FALSE,  
  error_sd = 1,  
  group_sd = 0.5,  
  obs_per_group = 10,  
  predictor_means = NULL,  
  predictor_sds = NULL,  
  seed = 123,  
  batch_size = 20,  
  min_sims = 40,  
  max_sims = 600,  
  ci_conf = 0.95,  
  margin = 0.02,  
  inla_num_threads = NULL,  
  family_args = list(),  
  progress = TRUE  
)
```

**Arguments**

formula	brms-style model formula.
family	GLM family (e.g., gaussian(), binomial()).
family_control	Optional list for INLA's control.family.
Ntrials	Optional vector of binomial trial counts (for binomial families).
E	Optional vector of exposures (for Poisson families).
scale	Optional numeric vector for scale parameter in INLA.
priors	brms prior specification object. Supported priors are translated to INLA controls where possible and audited in settings\$prior_translation.
data_generator	Optional function(n, effect) to simulate data.
effect_name	Character vector of fixed effects to assess.
effect_grid	Data frame or vector of effect values.
sample_sizes	Vector of sample sizes.
metric	Character; one of "direction", "threshold", "rope", or "bf" for Bayesian decision metric.
target	Target conditional power for the stopping rule (0-1).
prob_threshold	Posterior probability threshold for decision metrics.
effect_threshold	Effect-size threshold.
rope_bounds	Numeric length-2 vector defining ROPE.
credible_level	Credible interval level for Bayesian inference.
compute_bayes_factor	Logical; TRUE if metric is "bf".
error_sd	Residual standard deviation.
group_sd	Standard deviation of random effects.
obs_per_group	Number of observations per group.
predictor_means	Optional named list of predictor means.
predictor_sds	Optional named list of predictor standard deviations.
seed	Random seed.
batch_size	Number of simulations per sequential look.
min_sims	Minimum simulations before early stopping.
max_sims	Maximum simulations per cell.
ci_conf	Confidence level for Wilson confidence intervals.
margin	Margin around target for early stopping decision.
inla_num_threads	Character string specifying INLA threading (e.g., "4:1"). If NULL (default), automatically detects optimal setting based on CPU cores.
family_args	List of family-specific args passed to data generator.
progress	Logical; if TRUE, show progress messages.

**Details**

Sequential Bayesian Assurance Simulation Engine (Modern, Multi-Effect Ready)

Simulates assurance sequentially in batches, stopping early per cell based on Wilson confidence intervals.

**Value**

A list of class "brms\_inla\_power" with a per-cell summary (including the conditional\_power column, the Monte Carlo estimate at each fixed effect value) and simulation settings. Note this is conditional power at each grid value, not unconditional assurance; pass the result to `compute_assurance()` for the latter.

**Examples**

```
## Not run:
# Sequential design with automatic threading
results <- brms_inla_power_sequential(
  formula = outcome ~ treatment,
  effect_name = "treatment",
  effect_grid = c(0.2, 0.5, 0.8),
  sample_sizes = c(50, 100, 200),
  metric = "direction",
  target = 0.80
)
print(results$summary)

## End(Not run)
```

---

```
brms_inla_power_two_stage
```

*Two-Stage Bayesian Assurance Simulation (Multi-Effect, User-Friendly API)*

---

**Description**

Runs a two-stage Bayesian assurance simulation with formula-based multi-effect grids and adaptive refinement.

**Usage**

```
brms_inla_power_two_stage(
  formula,
  effect_name,
  effect_grid,
  n_range,
  stage1_k_n = 8,
  stage1_nsims = 100,
```

```

stage2_nsims = 400,
refine_metric = c("direction", "threshold", "rope"),
refine_target = 0.8,
prob_threshold = 0.95,
effect_threshold = 0,
obs_per_group = 10,
error_sd = 1,
group_sd = 0.5,
band = 0.06,
expand = 1L,
inla_num_threads = NULL,
...
)

```

### Arguments

formula	Model formula.
effect_name	Character vector of fixed effect names; must match formula terms.
effect_grid	Data frame with columns named by effect_name specifying effect values.
n_range	Numeric length-2 vector specifying sample size range.
stage1_k_n	Number of grid points in stage 1.
stage1_nsims	Number of simulations per cell in stage 1.
stage2_nsims	Number of simulations per cell in stage 2.
refine_metric	Metric used for refinement; one of "direction", "threshold", or "rope".
refine_target	Target assurance for refined cells.
prob_threshold	Posterior probability threshold for decision.
effect_threshold	Effect-size threshold for decision metric.
obs_per_group	Number of observations per group for grouping factors.
error_sd	Residual standard deviation.
group_sd	Standard deviation of random effects.
band	Numeric width of the target refinement band.
expand	Integer; how much to expand the refinement grid around candidates.
inla_num_threads	Character string specifying INLA threading (e.g., "4:1"). If NULL (default), automatically detects optimal setting based on CPU cores.
...	Additional arguments passed to internal functions.

### Value

A list with combined simulation results, summary, and stage parameters.

## Examples

```
## Not run:
# Two-stage design with threading
effect_grid <- expand.grid(
  treatment = c(0.2, 0.5, 0.8),
  covariate = c(0.1, 0.3)
)
results <- brms_inla_power_two_stage(
  formula = outcome ~ treatment + covariate,
  effect_name = c("treatment", "covariate"),
  effect_grid = effect_grid,
  n_range = c(50, 200),
  stage1_nsims = 3,
  stage2_nsims = 3,
  error_sd = 1
)
print(results$summary)

## End(Not run)
```

---

brms\_inla\_sequential\_trial

*Simulate a sequential Bayesian trial with interim stopping rules*

---

## Description

Simulates trials in which data **accumulate** and are analysed with INLA at a prespecified schedule of interim looks. At each look a posterior probability criterion is evaluated and the trial stops for success, futility, or (for the ROPE metric) practical equivalence; otherwise it continues to the maximum sample size. The function returns the design's operating characteristics: the probability of each decision, the distribution of stopping times, the expected sample size, and the average effect estimate at early success stops (a direct measure of optional stopping exaggeration).

This complements `brms_inla_power()` (fixed-n designs) and is **not** the same as `brms_inla_power_sequential()`, which adaptively stops the Monte Carlo simulation itself rather than simulating sequential trials.

Because Bayesian posterior quantities obey the likelihood principle, the *interpretation* of the final posterior is unaffected by the stopping rule; the purpose of this simulation is to quantify the *frequency* properties of the design (false-go rate, expected sample size, estimate exaggeration), which do depend on the stopping rule.

## Usage

```
brms_inla_sequential_trial(
  design = NULL,
  formula = NULL,
  family = gaussian(),
  family_control = NULL,
  priors = NULL,
```

```

data_generator = NULL,
effect_name = NULL,
true_effect = 0.5,
looks = NULL,
nsims = 200,
metric = c("direction", "threshold", "rope"),
alternative = c("greater", "less"),
prob_success = 0.95,
prob_futility = NULL,
effect_threshold = 0,
rope_bounds = NULL,
credible_level = 0.95,
error_sd = 1,
group_sd = 0.5,
obs_per_group = 10,
predictor_means = NULL,
predictor_sds = NULL,
seed = 123,
inla_num_threads = NULL,
family_args = list(),
progress = c("auto", "text", "none")
)

```

### Arguments

design	Optional "powerbrmsINLA_seq_design" object from <a href="#">sequential_design()</a> . When supplied, it overrides formula, family, priors, effect_name, metric, alternative, looks, prob_success, prob_futility, effect_threshold, rope_bounds, and credible_level, so the simulated operating characteristics correspond exactly to the design that will be monitored with <a href="#">sequential_analysis()</a> .
formula	Model formula (brms syntax; random effects supported).
family	Response family, e.g. <code>gaussian()</code> , <code>binomial()</code> .
family_control	Optional INLA control.family list; takes precedence over translated priors, as in <a href="#">brms_inla_power()</a> .
priors	Optional brms prior specification, translated via the built-in prior bridge (see <a href="#">brms_inla_power()</a> ).
data_generator	Optional function( <code>n</code> , <code>effect</code> ) returning a data frame with <code>n</code> rows <b>ordered by accrual time</b> (see Details). Defaults to the package's automatic generator.
effect_name	Single character: the primary (monitored) effect.
true_effect	Either a numeric vector of fixed true-effect scenarios (conditional sequential power per value), or a design-prior list such as <code>list(dist = "normal", mean = 0.5, sd = 0.15)</code> , in which case a fresh true effect is drawn for every simulated trial and the resulting success proportion is the design's <i>sequential assurance</i> .
looks	Increasing integer vector of cumulative sample sizes (total observations, the same units as <code>sample_sizes</code> in <a href="#">brms_inla_power()</a> ) at which interim analyses occur. The last element is the maximum <code>n</code> .

nsims	Number of simulated trials per scenario.
metric	"direction", "threshold", or "rope".
alternative	"greater" or "less"; the prespecified direction for the direction/threshold metrics. Ignored for "rope".
prob_success	Scalar or <code>length(looks)</code> vector of stop-for-success posterior probability thresholds (default 0.95).
prob_futility	Scalar or <code>length(looks)</code> vector of stop-for-futility thresholds, or NULL (default) to disable futility stopping. Ignored with a warning for <code>metric = "rope"</code> .
effect_threshold	Threshold for <code>metric = "threshold"</code> (on the model coefficient scale; use a negative value with <code>alternative = "less"</code> ).
rope_bounds	Length-2 numeric vector for <code>metric = "rope"</code> .
credible_level	Credible level used for the reported interval widths.
error_sd, group_sd	Scalar or distributional list, as in <code>brms_inla_power()</code> . With a distributional specification a fresh value is drawn once per simulated <i>trial</i> (a trial has one true nuisance value; it does not change between looks).
obs_per_group, predictor_means, predictor_sds	Passed to the automatic data generator (see <code>brms_inla_power()</code> ).
seed	Integer seed.
inla_num_threads	INLA threads ("outer:inner"); auto-detected if NULL.
family_args	Named list of family-specific arguments for the automatic generator.
progress	"auto", "text", or "none".

## Details

**Accrual model.** For each simulated trial one dataset of the maximum size is generated, and the analysis at a look of size  $n$  uses its first  $n$  rows, so the data genuinely accumulate across looks. With the automatic data generator, rows are exchangeable given the random effects (grouping factors cycle across rows), which corresponds to all clusters recruiting in parallel while observations accumulate within them. For a different accrual pattern (for example, whole clusters arriving one at a time), supply a custom `data_generator` whose rows are ordered by accrual time.

**Decision rules.** Let  $pr$  be the posterior probability of the prespecified alternative at a look (computed, as elsewhere in the package, from a Normal approximation to the INLA fixed-effect posterior):

- `metric = "direction"`:  $pr = P(\text{effect} > 0 \mid \text{data})$  (or  $< 0$  when `alternative = "less"`).
- `metric = "threshold"`:  $pr = P(\text{effect} > \text{effect\_threshold} \mid \text{data})$  (or  $<$  when `alternative = "less"`).
- `metric = "rope"`: two probabilities are monitored; the trial stops for "success" when  $P(\text{outside ROPE}) \geq \text{prob\_success}$  and for "equivalence" when  $P(\text{inside ROPE}) \geq \text{prob\_success}$ .

For the direction and threshold metrics the trial stops for success when  $pr \geq prob\_success$  and for futility when  $pr \leq prob\_futility$  (futility stopping is disabled when  $prob\_futility = NULL$ ).  $prob\_success$  and  $prob\_futility$  may be vectors of length( $looks$ ) to implement stricter early thresholds (O'Brien-Fleming-like behaviour).

**What to report.** `summary` contains, per scenario: `p_success`, `p_futility`, `p_equivalence`, `p_inconclusive`, `expected_n`, `p_stop_early`, and `mean_est_at_success` next to `mean_true_at_success` (their difference estimates the exaggeration of effect estimates caused by early stopping). Under `true_effect` containing 0, `p_success` is the false-go rate of the design.

## Value

A list of class "powerbrmsINLA\_seq\_trial" with components:

- `results`: one row per simulated trial (scenario, true effect, decision, stopping n and look, posterior estimate at stopping).
- `summary`: operating characteristics per scenario (see Details).
- `look_summary`: per-look stopping distribution per scenario.
- `diagnostics`: failed/warned INLA fits per scenario.
- `settings`: a record of all design parameters.

## Examples

```
## Not run:
looks <- c(40, 80, 120, 160, 200)

# Conditional sequential power at fixed true effects (0 = false-go rate)
seq_fixed <- brms_inla_sequential_trial(
  formula      = y ~ treatment,
  effect_name  = "treatment",
  true_effect  = c(0, 0.5),
  looks       = looks,
  nsims       = 200,
  metric      = "threshold",
  effect_threshold = 0.2,
  prob_success = 0.95,
  prob_futility = 0.05,
  seed        = 123
)
print(seq_fixed)

# Sequential assurance: draw the true effect from a design prior
seq_assur <- brms_inla_sequential_trial(
  formula      = y ~ treatment,
  effect_name  = "treatment",
  true_effect  = list(dist = "normal", mean = 0.5, sd = 0.15),
  looks       = looks,
  nsims       = 200,
  metric      = "threshold",
  effect_threshold = 0.2,
  prob_futility = 0.05,
```

```

  seed          = 123
)
print(seq_assur)

## End(Not run)

```

---

compute_assurance	<i>Compute unconditional Bayesian assurance from simulation results</i>
-------------------	---

---

## Description

Computes **unconditional Bayesian assurance** — the probability of a successful trial outcome averaged over prior uncertainty about the true effect size — from the output of [brms\\_inla\\_power\(\)](#) or related engines.

## Usage

```

compute_assurance(
  power_result,
  prior_weights,
  metric = c("direction", "threshold", "rope", "bf"),
  weight_tol = 0.01
)

```

## Arguments

- |               |   |
|---------------|---|
| power_result  | A list returned by <a href="#">brms_inla_power()</a> , <a href="#">brms_inla_power_parallel()</a> , <a href="#">brms_inla_power_two_stage()</a> or <a href="#">brms_inla_power_sequential()</a> ; or a plain data frame with at least columns <code>n</code> and the relevant <code>power_*</code> metric column.                             |
| prior_weights | Either (a) a named numeric vector of weights over effect-size values (must sum to 1 within tolerance 0.01), or (b) a list specifying a distribution — see <b>Details</b> . The output of <a href="#">beta_weights_on_grid()</a> and <a href="#">assurance_prior_weights()</a> is directly compatible.   |
| metric        | Character string selecting the decision metric. Must match a column present in the summary. One of: <ul style="list-style-type: none"> <li>• "direction" → <code>power_direction</code></li> <li>• "threshold" → <code>power_threshold</code></li> <li>• "rope" → <code>power_rope</code></li> <li>• "bf" → <code>bf_hit_10</code></li> </ul> |
| weight_tol    | Numeric tolerance for the weights-sum-to-1 check (default 0.01).  |

## Details

### Assurance vs. conditional Bayesian power:

The simulations run by `brms_inla_power()` are *conditional*: for each point on the effect grid the engine estimates the probability that the chosen decision rule is satisfied. This is Bayesian design power (a function of the unknown true effect).

**Assurance**, in the sense of O’Hagan & Stevens (2001) and O’Hagan, Stevens & Campbell (2005), is the *unconditional* version:

$$A(n) = \int \text{Power}(n, \delta) \pi(\delta) d\delta \approx \sum_j w_j \cdot \text{Power}(n, \delta_j)$$

where  $\pi(\delta)$  is a **design prior** on the effect size and  $w_j$  are the normalised prior weights over the discrete effect grid  $\{\delta_j\}$ . Assurance therefore accounts for the investigator’s genuine uncertainty about the effect, not just a single "assumed" value (Ristl et al., 2019; Kunzmann et al., 2021).

### Variance uncertainty:

If the simulation was run with multiple sampled variance parameters (stored in columns such as `sampled_error_sd` or `sampled_group_sd` in the results), the averaging over those values is already implicit in each per-cell power estimate, so no additional action is required here.

### Supplying prior weights:

Two forms are accepted for `prior_weights`:

- **Named numeric vector** — names must match the effect-grid values used in `power_result` (as produced by `as.character()`, which is the format used by `beta_weights_on_grid()` and `assurance_prior_weights()`). Unnamed vectors are accepted only when their length equals the number of unique effect values, in which case they are applied in ascending order.
- **Distribution list** — a list with at minimum `$dist` naming one of "normal" (mean, sd), "uniform" (min, max), or "beta" (shape1/shape2 **or** mode/n). Weights are computed by evaluating the density at each grid point and normalising. *Supported for single-effect results only.*

For **multi-effect** grids the `prior_weights` argument must be a numeric vector of length equal to the number of unique effect combinations in the summary (sorted lexicographically by effect columns). Use `assurance_prior_weights()` or `beta_weights_on_grid()` to construct compatible weights for single-effect cases.

## Value

A list of class "powerbrmsINLA\_assurance" containing:

`assurance` Data frame with columns `sample_size` and `assurance`.

`metric` The decision metric used.

`power_col` Name of the summary column used for power.

`prior_spec` The `prior_weights` argument as supplied (useful for reproducibility).

`weights` Named numeric vector of the normalised weights actually applied.

`eff_cols` Character vector of effect-grid column names identified in the summary.

## References

- O'Hagan, A., & Stevens, J. W. (2001). Bayesian assessment of sample size for clinical trials of cost-effectiveness. *Medical Decision Making*, 21(3), 219–230. doi:10.1177/0272989X0102100307
- O'Hagan, A., Stevens, J. W., & Campbell, M. J. (2005). Assurance in clinical trial design. *Pharmaceutical Statistics*, 4(3), 187–201. doi:10.1002/pst.175
- Ristl, R., Glimm, E., Stallard, N., & Posch, M. (2019). Optimal design and analysis of two-stage adaptive enrichment trials. *Biometrical Journal*, 61(6), 1461–1481.
- Kunzmann, K., Grayling, M. J., Lee, K. M., Robertson, D. S., Rufibach, K., & Wason, J. M. S. (2021). A review of Bayesian perspectives on sample size derivation for confirmatory trials. *The American Statistician*, 75(4), 424–432. doi:10.1080/00031305.2021.1901782

## Examples

```
# Build a small synthetic power_result without running INLA
syn_summary <- data.frame(
  n          = rep(c(50, 100, 200), each = 3),
  treatment  = rep(c(0.2, 0.5, 0.8), 3),
  power_direction = c(0.40, 0.65, 0.85,
                    0.60, 0.82, 0.95,
                    0.72, 0.90, 0.98),
  stringsAsFactors = FALSE
)
syn_result <- list(
  summary = syn_summary,
  settings = list(effect_name = "treatment")
)

# (a) Uniform weights – assurance is the simple mean of per-cell powers
w_uniform <- c("0.2" = 1/3, "0.5" = 1/3, "0.8" = 1/3)
out <- compute_assurance(syn_result, prior_weights = w_uniform)
print(out)

# (b) Normal design prior centred on a medium-sized effect
out2 <- compute_assurance(
  syn_result,
  prior_weights = list(dist = "normal", mean = 0.5, sd = 0.2)
)
print(out2)

# (c) Using assurance_prior_weights() to build the weight vector explicitly
w_norm <- assurance_prior_weights(c(0.2, 0.5, 0.8), dist = "normal",
                                 mean = 0.5, sd = 0.2)
out3 <- compute_assurance(syn_result, prior_weights = w_norm)
```

**Description**

Returns the smallest per-group sample size that meets user-specified power or assurance targets. Works with `brms_inla_power()` and related engine outputs.

**Usage**

```
decide_sample_size(
  x,
  direction = NULL,
  threshold = NULL,
  rope_in = NULL,
  bf10 = NULL,
  bf_prop_min = 0,
  targets = NULL,
  prior_weights = NULL,
  target = 0.8
)
```

**Arguments**

<code>x</code>	A list with <code>\$summary</code> (engine output) or a plain <code>data.frame</code> summary.
<code>direction</code>	Numeric in $[0, 1]$ : assurance target (assurance mode) or required conditional power (conditional mode) for <code>power_direction</code> . Omit to exclude this metric.
<code>threshold</code>	Numeric in $[0, 1]$ : assurance target (assurance mode) or required conditional power (conditional mode) for <code>power_threshold</code> . Omit to exclude this metric.
<code>rope_in</code>	Numeric in $[0, 1]$ : assurance target (assurance mode) or maximum allowed $\Pr(\text{in ROPE})$ (conditional mode). Omit to exclude this metric.
<code>bf10</code>	Numeric Bayes-factor cutoff (e.g., 10). If provided the function looks for a column <code>bf_hit_&lt;bf10&gt;</code> ; if absent it falls back to per-simulation <code>bf10</code> in <code>x\$results</code> .
<code>bf_prop_min</code>	Numeric in $[0, 1]$ : minimum proportion of simulations that must achieve $\text{BF} \geq \text{bf10}$ (default 0; conditional mode only).
<code>targets</code>	Optional named list alternative to the direct arguments. Ignored when any direct argument is non-NULL.
<code>prior_weights</code>	A design prior for assurance mode. Accepts the same formats as <code>compute_assurance()</code> : a named numeric vector of weights over effect-grid values, or a distribution list such as <code>list(dist = "normal", mean = 0.5, sd = 0.2)</code> . When NULL (default), the function operates in conditional mode.
<code>target</code>	Numeric in $[0, 1]$ : fallback assurance level used only when a metric is requested without a valid numeric threshold (default 0.80). In normal usage the per-metric value (e.g., <code>direction = 0.70</code> ) supersedes this argument.

**Details**

Targets may be supplied as direct arguments or via a `targets` list; direct arguments take precedence.

**Modes of operation:**

**Assurance mode** (recommended): supply `prior_weights` with a design prior over the effect grid. The function calls `compute_assurance()` internally and returns the smallest sample size where *unconditional* Bayesian assurance reaches the per-metric target for each requested metric.

In assurance mode the numeric value passed to `direction`, `threshold`, `rope_in`, or `bf10` is the **assurance target** for that metric. For example, `direction = 0.70` finds the smallest `n` where direction assurance  $\geq 0.70$ , and `threshold = 0.60` finds the smallest `n` where threshold assurance  $\geq 0.60$ . Multiple metrics may be requested simultaneously, each with its own target.

**Conditional mode** (backward compatible): when `prior_weights = NULL`, the function selects the smallest `n` at which the per-cell conditional power meets the specified target for each effect-size value. If the summary contains sampled variance columns (e.g., from `distributional_error_sd`), they are averaged out before selecting `n`.

**Value**

An object of class "powerbrmsINLA\_sample\_size" (which inherits from `data.frame`) with a `print.powerbrmsINLA_sample_size()` method.

**Assurance mode** columns:

`metric` Requested metric name ("direction", "threshold", "rope\_in", "bf10").

`target` The assurance target supplied.

`n_recommended` Smallest per-group sample size achieving the target, or NA if none in the grid qualifies.

`assurance_achieved` Assurance value at the recommended `n`.

`prior_description` Plain-text description of the design prior.

**Conditional mode** columns:

`<effect column(s)>` One column per effect variable.

`n_recommended` Smallest per-group sample size meeting all targets, or NA.

`cond_power_*` Conditional power at the recommended `n` for each requested metric.

**See Also**

[compute\\_assurance\(\)](#), [assurance\\_prior\\_weights\(\)](#), [beta\\_weights\\_on\\_grid\(\)](#)

**Examples**

```
# Build a small synthetic power_result without running INLA
syn_summary <- data.frame(
  n           = rep(c(50, 100, 200), each = 3),
  treatment   = rep(c(0.2, 0.5, 0.8), 3),
  power_direction = c(0.40, 0.65, 0.85,
                    0.60, 0.82, 0.95,
                    0.72, 0.90, 0.98),
  power_threshold = c(0.30, 0.55, 0.75,
                    0.50, 0.72, 0.88,
                    0.62, 0.80, 0.92),
```

```

  stringsAsFactors = FALSE
)
syn_result <- list(
  summary = syn_summary,
  settings = list(effect_name = "treatment")
)

# --- Assurance mode: each metric value IS the assurance target ---
w <- assurance_prior_weights(c(0.2, 0.5, 0.8), dist = "normal",
                             mean = 0.5, sd = 0.2)
# Find n where direction assurance >= 0.80 AND threshold assurance >= 0.75
rec_assurance <- decide_sample_size(
  syn_result,
  direction    = 0.80,
  threshold    = 0.75,
  prior_weights = w
)
print(rec_assurance)

# --- Conditional mode (backward compatible) ---
rec_cond <- decide_sample_size(syn_result, direction = 0.80)
print(rec_cond)

```

---

hdi\_of\_icdf

*Highest Density Interval from an Inverse CDF*


---

### Description

Computes an HDI of given mass from any distribution for which you have a quantile function (inverse CDF).

### Usage

```
hdi_of_icdf(qfun, width = 0.95, tol = 1e-08, ...)
```

### Arguments

qfun	Quantile function, e.g., qbeta, qnorm, ...
width	Desired HDI mass (e.g., 0.95).
tol	Optimizer tolerance.
...	Additional arguments passed to qfun.

### Value

Named numeric vector with elements l1 and u1.

---

min_n_beta_binom	<i>Minimum n for Target Assurance (Beta-Binomial)</i>
------------------	---

---

**Description**

Minimum n for Target Assurance (Beta-Binomial)

**Usage**

```
min_n_beta_binom(  
  gen_prior_mode,  
  gen_prior_n,  
  desired_power,  
  aud_prior_mode = 0.5,  
  aud_prior_n = 2,  
  hdi_mass = 0.95,  
  rope = NULL,  
  hdi_max_width = NULL,  
  n_start = 20,  
  n_max = 1e+05,  
  verbose = TRUE  
)
```

**Arguments**

gen_prior_mode	Generating prior mode in (0,1).
gen_prior_n	Generating prior concentration ( $\geq 2$ ).
desired_power	Target assurance value in (0,1).
aud_prior_mode	Audience prior mode in (0,1).
aud_prior_n	Audience prior concentration ( $\geq 2$ ).
hdi_mass	HDI mass (e.g., 0.95).
rope	Length-2 numeric vector for ROPE bounds, or NULL for max-width rule.
hdi_max_width	Positive width threshold for the HDI (used if rope=NULL).
n_start	Starting sample size for search.
n_max	Maximum sample size to try.
verbose	If TRUE, prints progress.

**Value**

Smallest n meeting the target assurance.

---

plot\_assurance\_curve *Plot Assurance Curve(s) vs Sample Size*

---

### Description

Plots unconditional Bayesian assurance against sample size for one or more design priors, with an optional horizontal reference line at a target assurance level. Each prior produces one line, making it easy to compare how the choice of design prior changes the required sample size.

### Usage

```
plot_assurance_curve(
  assurance_results,
  labels = NULL,
  target = 0.8,
  title = NULL,
  subtitle = NULL
)
```

### Arguments

assurance_results	A single powerbrmsINLA_assurance object (from <code>compute_assurance()</code> ) or a list of such objects.
labels	Character vector naming each design prior, e.g. <code>c("Sceptical", "Primary", "Optimistic")</code> . Defaults to "Design prior" for a single object or "Prior 1", "Prior 2", ... for multiple objects.
target	Numeric in (0, 1): horizontal reference line added at this assurance level (default 0.80). Set to NULL to suppress the line.
title, subtitle	Optional plot title and subtitle strings.

### Value

A ggplot object.

### Examples

```
syn_summary <- data.frame(
  n          = rep(c(50, 100, 200), each = 3),
  treatment  = rep(c(0.2, 0.5, 0.8), 3),
  power_direction = c(0.40, 0.65, 0.85, 0.60, 0.82, 0.95, 0.72, 0.90, 0.98)
)
pr <- list(summary = syn_summary, settings = list(effect_name = "treatment"))
a1 <- compute_assurance(pr, list(dist = "normal", mean = 0.5, sd = 0.15))
a2 <- compute_assurance(pr, list(dist = "normal", mean = 0.5, sd = 0.30))
plot_assurance_curve(list(a1, a2), labels = c("Informative", "Diffuse"))
```

---

 plot\_assurance\_with\_robustness

*Plot Conditional Power with Robustness Ribbon (Multi-Effect Grid Friendly)*

---

## Description

Compares conditional Bayesian power results from multiple scenarios by showing the range ("ribbon") of values across scenarios for each sample size and effect grid variable.

## Usage

```
plot_assurance_with_robustness(
  power_results_list,
  metric = c("precision", "direction", "threshold", "bf"),
  x_effect = NULL,
  facet_by = NULL,
  precision_target = NULL,
  p_star = 0.95,
  bf_threshold = 10,
  effect_filters = NULL,
  effect_weights = NULL,
  show_individual_scenarios = FALSE,
  title = NULL,
  subtitle = NULL
)
```

## Arguments

power_results_list	Named list of results objects from <code>brms_inla_power</code> or sequential/two-stage variants.
metric	Which conditional power metric to compute: "precision", "direction", "threshold", or "bf".
x_effect	Name of effect grid column for x-axis (default: first detected grid column).
facet_by	Optional effect grid column(s) to facet by.
precision_target	CI width target if metric="precision".
p_star	Posterior probability threshold for "direction"/"threshold".
bf_threshold	BF10 threshold for "bf".
effect_filters	Optional named list for filtering rows (e.g. <code>list(treatment=0)</code> ).
effect_weights	Optional named numeric vector for averaging over grid values.
show_individual_scenarios	Logical; if TRUE, overlay each scenario's curve.
title, subtitle	Optional plot labels.

**Value**

A ggplot object.

---

plot\_bf\_assurance\_curve

*Bayes-factor conditional power curve (user-facing wrapper)*

---

**Description**

This is the main function users should call to visualise conditional Bayesian power for the Bayes factor criterion. It is a thin wrapper around `plot_bf_assurance_curve_smooth()`, so all existing behaviour is preserved.

**Usage**

```
plot_bf_assurance_curve(power_results, bf_threshold = 10, effect_filter = NULL)
```

**Arguments**

`power_results` Output from a `brms_inla_power*` function (or a `data.frame` with columns `n`, `bf10` and any effect columns).

`bf_threshold` Numeric Bayes factor cutoff (default 10).

`effect_filter` Optional named list to filter effect-grid columns, e.g. `list(treatment = 0.3)`.

**Value**

A ggplot object.

---

plot\_bf\_assurance\_curve\_smooth

*Conditional Bayesian power curve for the Bayes factor criterion with Wilson CIs (multi-effect grid friendly)*

---

**Description**

Plots the conditional power — the proportion of simulations in which BF10 meets or exceeds a threshold at each fixed effect size — grouped by sample size and any effect grid variables.

**Usage**

```
plot_bf_assurance_curve_smooth(x, cutoff = 10, effect_filter = NULL)
```

**Arguments**

x	Engine result (list with \$results) or a data.frame with at least columns n, bf10 and any effect columns (e.g., treatment).
cutoff	Numeric Bayes factor threshold for a "hit" (default 10).
effect_filter	Optional named list to filter effects, e.g. list(treatment = 0.6).

**Value**

A ggplot object.

---

plot\_bf\_expected\_evidence

*Plot Expected Evidence (mean log10 BF10, Multi-Effect Grid Friendly)*

---

**Description**

Plots the average log10 BF10 against any effect grid variable, grouped/faceted.

**Usage**

```
plot_bf_expected_evidence(
  power_results,
  x_effect = NULL,
  facet_by = NULL,
  n = NULL,
  agg_fun = mean,
  title = NULL,
  subtitle = NULL
)
```

**Arguments**

power_results	Simulation results from a brms_inla_power* function with compute_bayes_factor = TRUE.
x_effect	Name of effect grid column for x-axis (default: first grid column).
facet_by	Optional grid column(s) to facet by (default: NULL).
n	Optional sample size to filter to (NULL means plot all; else one curve per grid/facet).
agg_fun	Aggregation function if >1 entries per cell (default: mean).
title, subtitle	Optional plot labels.

**Value**

A ggplot object.

---

plot_bf_heatmap	<i>Plot Bayes Factor Heatmap (mean log10 BF10, Multi-Effect Grid Friendly)</i>
-----------------	--

---

## Description

Heatmap of mean log10 BF10 as a function of two effect grid columns (x/y), with optional faceting.

## Usage

```
plot_bf_heatmap(  
  power_results,  
  x_effect = NULL,  
  y_effect = "n",  
  facet_by = NULL,  
  n = NULL,  
  agg_fun = mean,  
  title = NULL,  
  subtitle = NULL  
)
```

## Arguments

power_results	Simulation results from a <code>brms_inla_power*</code> function with <code>compute_bayes_factor = TRUE</code> .
x_effect	Name of effect grid column for x-axis (default: first grid column).
y_effect	Name of effect grid column for y-axis (default: "n").
facet_by	Optional column(s) to facet by.
n	Optional sample size to filter to (NULL means plot all; else show only that n).
agg_fun	Aggregation function (default: mean).
title, subtitle	Optional plot labels.

## Value

ggplot object.

---

```
plot_decision_assurance_curve
      Plot Conditional Power Curve for a Decision Rule (Multi-Effect Grid
      Friendly)
```

---

### Description

Plots the conditional Bayesian power (proportion of simulation runs meeting a posterior probability decision rule) versus an effect grid variable, for a given metric ("direction", "threshold", or "rope") at a fixed decision probability threshold  $p_{\text{star}}$ .

### Usage

```
plot_decision_assurance_curve(
  power_results,
  metric = c("direction", "threshold", "rope"),
  p_star = 0.95,
  x_effect = NULL,
  facet_by = NULL,
  effect_filters = NULL,
  effect_weights = NULL,
  title = NULL,
  subtitle = NULL
)
```

### Arguments

<code>power_results</code>	A list returned by <code>brms_inla_power*</code> .
<code>metric</code>	Decision metric: "direction", "threshold", or "rope".
<code>p_star</code>	Numeric decision threshold in (0,1).
<code>x_effect</code>	Name of effect grid column for x-axis (default: first grid column).
<code>facet_by</code>	Optional effect grid column(s) to facet by.
<code>effect_filters</code>	Optional named list for filtering rows, e.g. <code>list(treatment=0)</code> .
<code>effect_weights</code>	Optional named numeric vector of weights for selected <code>x_effect</code> values.
<code>title, subtitle</code>	Optional plot labels.

### Details

These plots display **conditional Bayesian power** — the probability of meeting the decision criterion at a fixed effect size. For unconditional assurance (averaged over a design prior on effect size), see [plot\\_assurance\\_curve\(\)](#).

### Value

A ggplot object.

---

`plot_decision_threshold_contour`*Plot Decision Threshold Contour (Multi-Effect Grid Friendly)*

---

### Description

Shows conditional Bayesian power as a function of decision threshold  $p^*$  and one effect grid column, optionally faceted.

### Usage

```
plot_decision_threshold_contour(  
  power_results,  
  metric = c("direction", "threshold", "rope"),  
  p_star_grid = seq(0.5, 0.99, by = 0.01),  
  effect_var = NULL,  
  facet_by = NULL,  
  effect_value = NULL,  
  effect_weights = NULL,  
  title = NULL,  
  subtitle = NULL  
)
```

### Arguments

<code>power_results</code>	brms_inla_power list (or two-stage, etc.)
<code>metric</code>	Which metric: "direction", "threshold", "rope"
<code>p_star_grid</code>	Numeric vector of decision thresholds (default: 0.5 to 0.99 by 0.01)
<code>effect_var</code>	Name of effect grid column for y-axis (default: first detected grid column)
<code>facet_by</code>	Optional effect grid column(s) to facet by
<code>effect_value</code>	Optional value(s) to filter for <code>effect_var</code> , or named list for multi-filter
<code>effect_weights</code>	Optional weights for aggregation (named by <code>effect_var</code> values)
<code>title, subtitle</code>	Optional plot labels.

### Value

ggplot2 object.

---

plot\_design\_prior      *Plot Design Prior Density over the Effect Grid*

---

### Description

Visualises one or more design prior distributions over the effect-size range used in a power analysis. Vertical dashed lines mark the grid points at which power was simulated, helping assess prior-grid alignment.

### Usage

```
plot_design_prior(
  prior_spec,
  effect_grid,
  labels = NULL,
  n_points = 300L,
  title = NULL,
  subtitle = NULL
)
```

### Arguments

prior_spec	A single prior specification list (e.g. <code>list(dist = "normal", mean = 0.5, sd = 0.15)</code> ) <b>or</b> a list of such lists. Supported distributions: "normal" (mean, sd), "uniform" (min, max), "beta" (shape1, shape2 or mode, n).
effect_grid	Numeric vector of effect-grid values passed to <code>brms_inla_power()</code> . Vertical dashed lines are drawn at each value.
labels	Character vector naming each prior. Defaults to "Design prior" for a single spec or "Prior 1", "Prior 2", ... for multiple.
n_points	Number of x points at which to evaluate the density curve (default 300).
title, subtitle	Optional plot title and subtitle strings.

### Value

A ggplot object.

### Examples

```
plot_design_prior(
  prior_spec = list(dist = "normal", mean = 0.5, sd = 0.15),
  effect_grid = c(0.2, 0.5, 0.8)
)

# Multiple priors
plot_design_prior(
  prior_spec = list(
    list(dist = "normal", mean = 0.5, sd = 0.10),
```

```

    list(dist = "normal", mean = 0.5, sd = 0.30)
  ),
  effect_grid = c(0.2, 0.5, 0.8),
  labels      = c("Informative", "Diffuse")
)

```

---

plot\_interaction\_surface

*Plot Interaction Conditional Power Surface/Heatmap/Lines (Multi-Effect Grid Friendly)*

---

### Description

Visualizes a metric (e.g., conditional Bayesian power) as a function of two effect grid variables for a fixed sample size or averaged over n. Allows line, heatmap, or contour modes.

### Usage

```

plot_interaction_surface(
  data,
  metric,
  effect1,
  effect2,
  n = NULL,
  line = FALSE,
  facet_by = NULL,
  agg_fun = mean,
  title = NULL,
  subtitle = NULL
)

```

### Arguments

data	Data frame (typically power_results\$summary).
metric	Name of the summary column to plot, e.g. "power_direction", "power_threshold".
effect1	Name of effect grid column for x-axis.
effect2	Name of effect grid column for y-axis or color/facets.
n	Optional sample size to filter to (else averages/plots all n's).
line	Logical; if TRUE, make a lineplot (effect1 on x, one line for each effect2). If FALSE, make a heatmap or contour.
facet_by	Optional grid column(s) to facet by.
agg_fun	Aggregation function if multiple entries per cell (default = mean).
title, subtitle	Optional plot labels.

**Value**

A ggplot object.

---

plot\_power\_assurance\_overlay

*Plot Conditional Power Curves with Assurance Overlay*

---

**Description**

Shows the conditional power curve for each effect-size value in the design grid (thin lines), overlaid with the unconditional Bayesian assurance curve (thick line). This communicates that assurance is the prior-weighted average of the conditional power curves.

**Usage**

```
plot_power_assurance_overlay(
  power_result,
  assurance_result,
  metric = c("direction", "threshold", "rope", "bf"),
  title = NULL,
  subtitle = NULL
)
```

**Arguments**

`power_result` Output from `brms_inla_power()` (or compatible wrappers) containing a `$summary` element.

`assurance_result` A `powerbrmsINLA_assurance` object from `compute_assurance()`.

`metric` Decision metric to plot: "direction" (default), "threshold", "rope", or "bf".

`title, subtitle` Optional plot title and subtitle strings.

**Value**

A ggplot object.

**Examples**

```
syn_summary <- data.frame(
  n          = rep(c(50, 100, 200), each = 3),
  treatment  = rep(c(0.2, 0.5, 0.8), 3),
  power_direction = c(0.40, 0.65, 0.85, 0.60, 0.82, 0.95, 0.72, 0.90, 0.98)
)
pr <- list(summary = syn_summary, settings = list(effect_name = "treatment"))
a <- compute_assurance(pr, list(dist = "normal", mean = 0.5, sd = 0.15))
plot_power_assurance_overlay(pr, a)
```

---

plot_power_contour	<i>Draw a filled contour plot of conditional Bayesian power for a chosen metric, as a function of two effect grid columns and sample size.</i>
--------------------	--

---

### Description

Plots the conditional Bayesian power — the probability of meeting the decision criterion at each fixed effect size and sample size — as a filled contour surface.

### Usage

```
plot_power_contour(
  power_results,
  power_metric = c("direction", "threshold", "rope"),
  x_effect = NULL,
  y_effect = "n",
  facet_by = NULL,
  power_threshold = 0.8,
  show_threshold_line = TRUE,
  title = NULL,
  subtitle = NULL
)
```

### Arguments

power_results	Output from a brms_inla_power function.
power_metric	Which metric to plot: "direction", "threshold", or "rope".
x_effect	Name of effect grid column for x-axis (default = first effect).
y_effect	Name of effect grid column for y-axis (default = "n").
facet_by	Optional effect grid column(s) to facet by.
power_threshold	Optional reference contour line for conditional power (default 0.8).
show_threshold_line	Logical; add a red contour at power_threshold.
title, subtitle	Optional plot labels.

### Details

These plots display **conditional Bayesian power** — the probability of meeting the decision criterion at a fixed effect size. For unconditional assurance (averaged over a design prior on effect size), see [plot\\_assurance\\_curve\(\)](#).

### Value

A ggplot object.

---

plot_power_heatmap	<i>Plot Conditional Bayesian Power Heatmap (Multi-Effect Grid Friendly)</i>
--------------------	---

---

### Description

Displays the conditional Bayesian power — the probability of meeting the decision criterion at each fixed effect size — as a colour-filled heatmap.

### Usage

```
plot_power_heatmap(  
  power_results,  
  power_metric = c("direction", "threshold", "rope"),  
  x_effect = NULL,  
  y_effect = "n",  
  facet_by = NULL,  
  title = NULL,  
  subtitle = NULL  
)
```

### Arguments

`power_results` Output from a `brms_inla_power` function.

`power_metric` Which metric to plot: "direction", "threshold", or "rope".

`x_effect` Name of effect grid column for x-axis (default = first effect).

`y_effect` Name of effect grid column for y-axis (default = "n").

`facet_by` Optional effect grid column(s) to facet by.

`title, subtitle` Optional plot labels.

### Details

Heatmap of conditional Bayesian power for a chosen metric across two selected effect grid variables and sample sizes.

These plots display **conditional Bayesian power** — the probability of meeting the decision criterion at a fixed effect size. For unconditional assurance (averaged over a design prior on effect size), see [plot\\_assurance\\_curve\(\)](#).

### Value

A ggplot object.

---

```
plot_precision_assurance_curve
```

*Plot Precision Conditional Power Curve (Multi-Effect Grid Friendly)*

---

### Description

Plots the conditional power for precision (proportion of runs where CI width  $\leq$  target) vs. a chosen effect grid variable across sample size(s). Supports faceting, effect filtering, and weights.

### Usage

```
plot_precision_assurance_curve(
  power_results,
  precision_target,
  x_effect = NULL,
  facet_by = NULL,
  effect_filters = NULL,
  effect_weights = NULL,
  title = NULL,
  subtitle = NULL
)
```

### Arguments

`power_results` List returned by `brms_inla_power*`.

`precision_target` Numeric; credible interval width threshold for success.

`x_effect` Name of effect grid column for x-axis (default: first grid column).

`facet_by` Optional effect grid column(s) for faceting.

`effect_filters` Optional named list for filtering rows, e.g. `list(treatment=0)`.

`effect_weights` Optional named numeric vector for weights over selected `x_effect` values.

`title, subtitle` Optional plot labels.

### Details

These plots display **conditional Bayesian power** — the probability of achieving the precision criterion at a fixed effect size. For unconditional assurance (averaged over a design prior on effect size), see [plot\\_assurance\\_curve\(\)](#).

### Value

A ggplot object.

---

`plot_precision_fan_chart`*Precision conditional power as a function of sample size*

---

## Description

Plots the proportion of simulations in which the posterior credible interval width is less than or equal to a target, as a function of sample size  $n$ . Optionally colours separate curves by an effect-grid variable.

## Usage

```
plot_precision_fan_chart(  
  power_results,  
  ci_width_target,  
  effect_filter = NULL,  
  colour_by = NULL,  
  title = NULL,  
  subtitle = NULL  
)
```

## Arguments

`power_results` Output from a `brms_inla_power*` function, or a data.frame with at least columns `n` and `ci_width`, plus any effect-grid columns (e.g. `treatment`, `age_effect`).

`ci_width_target` Numeric, target width for the credible interval. Conditional power is defined as  $\Pr(\text{ci\_width} \leq \text{ci\_width\_target})$ .

`effect_filter` Optional named list for filtering effect-grid columns, e.g. `list(treatment = 0.3)`.

`colour_by` Optional name of an effect-grid column to colour separate curves by. If NULL, only `n` is used.

`title, subtitle` Optional plot labels.

## Details

This implementation works directly from the per-simulation results (column `ci_width`) and does not rely on the robustness engine.

These plots display **conditional Bayesian power** — the probability of achieving the precision criterion at a fixed effect size. For unconditional assurance (averaged over a design prior on effect size), see [plot\\_assurance\\_curve\(\)](#).

## Value

A ggplot object.

---

```
plot_sequential_monitor
```

*Plot a sequential analysis trajectory with stopping boundaries*

---

### Description

Visualises a `sequential_analysis()` monitor: either the monitored posterior probability across looks with the prespecified success and futility boundaries, or the effect estimate with its credible interval across looks.

### Usage

```
plot_sequential_monitor(
  monitor,
  type = c("probability", "estimate"),
  title = NULL,
  subtitle = NULL
)
```

### Arguments

`monitor` A "powerbrmsINLA\_seq\_monitor" object.  
`type` "probability" (default) or "estimate".  
`title, subtitle` Optional plot annotations.

### Value

A ggplot object.

---

```
print.brms_inla_power
```

*Print method for brms\_inla\_power result objects*

---

### Description

Displays a concise summary of a simulation result, including key settings, the first rows of the power summary table, and — when present — a one-line INLA diagnostic summary stating the proportion of fits that produced warnings and the number of failed fits.

### Usage

```
## S3 method for class 'brms_inla_power'
print(x, n_rows = 10L, ...)
```

**Arguments**

x	A list of class "brms_inla_power" as returned by <code>brms_inla_power()</code> , <code>brms_inla_power_parallel()</code> or <code>brms_inla_power_two_stage()</code> .
n_rows	Maximum number of summary rows to print (default 10).
...	Unused; present for S3 compatibility.

**Value**

x, invisibly.

---

```
print.powerbrmsINLA_assurance
```

*Print method for powerbrmsINLA\_assurance objects*

---

**Description**

Displays the unconditional Bayesian assurance by sample size together with a plain-language description of the prior used.

**Usage**

```
## S3 method for class 'powerbrmsINLA_assurance'
print(x, digits = 4L, ...)
```

**Arguments**

x	An object of class "powerbrmsINLA_assurance" returned by <code>compute_assurance()</code> .
digits	Number of significant digits for assurance values (default 4).
...	Unused; present for S3 compatibility.

**Value**

x, invisibly.

---

```
print.powerbrmsINLA_sample_size
```

*Print method for powerbrmsINLA\_sample\_size objects*

---

### Description

Formats recommendations from `decide_sample_size()` in a human-readable way. In assurance mode each row is rendered as a plain-English sentence. In conditional mode a formatted table is printed.

### Usage

```
## S3 method for class 'powerbrmsINLA_sample_size'  
print(x, digits = 4L, ...)
```

### Arguments

x	An object of class "powerbrmsINLA_sample_size" returned by <code>decide_sample_size()</code> .
digits	Number of decimal places for power/assurance values (default 4).
...	Unused; present for S3 compatibility.

### Value

x, invisibly.

---

```
print.powerbrmsINLA_seq_design
```

*Print method for sequential design objects*

---

### Description

Print method for sequential design objects

### Usage

```
## S3 method for class 'powerbrmsINLA_seq_design'  
print(x, ...)
```

### Arguments

x	A "powerbrmsINLA_seq_design" object.
...	Unused; present for S3 compatibility.

### Value

x, invisibly.

---

```
print.powerbrmsINLA_seq_monitor
```

*Print method for sequential analysis monitor objects*

---

**Description**

Print method for sequential analysis monitor objects

**Usage**

```
## S3 method for class 'powerbrmsINLA_seq_monitor'  
print(x, digits = 3L, ...)
```

**Arguments**

x	A "powerbrmsINLA_seq_monitor" object.
digits	Digits for printed numbers.
...	Unused; present for S3 compatibility.

**Value**

x, invisibly.

---

```
print.powerbrmsINLA_seq_trial
```

*Print method for sequential trial simulation results*

---

**Description**

Print method for sequential trial simulation results

**Usage**

```
## S3 method for class 'powerbrmsINLA_seq_trial'  
print(x, digits = 3L, ...)
```

**Arguments**

x	A "powerbrmsINLA_seq_trial" object from <a href="#">brms_inla_sequential_trial()</a> .
digits	Number of digits for printed proportions.
...	Unused; present for S3 compatibility.

**Value**

x, invisibly.

---

sequential\_analysis    *Analyse accumulated real data at a sequential interim look*

---

## Description

Fits the prespecified model to the data accumulated so far, computes the monitored posterior probability, applies the prespecified stopping threshold for the current look, and records the result in an auditable monitor object. Call it once per interim look, passing the design on the first call and the returned monitor object on subsequent calls.

The monitor refuses to continue after a stop decision has been recorded unless `allow_after_stop = TRUE`, in which case the continuation is logged as a protocol deviation. Analyses at sample sizes that do not match the planned schedule are permitted but flagged and recorded.

## Usage

```
sequential_analysis(
  x,
  data,
  Ntrials_col = NULL,
  allow_after_stop = FALSE,
  note = NULL,
  inla_num_threads = NULL
)
```

## Arguments

<code>x</code>	A "powerbrmsINLA_seq_design" (first look) or "powerbrmsINLA_seq_monitor" (subsequent looks).
<code>data</code>	Data frame containing all data accumulated so far (not just the new batch). Must contain the variables in the design formula.
<code>Ntrials_col</code>	Optional name of a column with binomial trial counts (binomial/beta-binomial families). Defaults to a column named <code>.Ntrials</code> if present, else 1 (Bernoulli).
<code>allow_after_stop</code>	Logical; permit analysis after a recorded stop decision (logged as a deviation). Default FALSE.
<code>note</code>	Optional character note recorded with this look (e.g. reason for an off-schedule analysis).
<code>inla_num_threads</code>	INLA threads ("outer:inner"); auto-detected if NULL.

## Value

A list of class "powerbrmsINLA\_seq\_monitor" containing the design, its fingerprint, a looks data frame (one row per analysis: timestamp, n, planned n, estimate, credible interval, monitored probability, thresholds, decision, deviations), and the current status ("ongoing" or a stop decision).

**See Also**

[sequential\\_design\(\)](#), [plot\\_sequential\\_monitor\(\)](#), [brms\\_inla\\_sequential\\_trial\(\)](#)

**Examples**

```
## Not run:
design <- sequential_design(
  formula = y ~ treatment, effect_name = "treatment",
  metric = "direction", looks = c(40, 80, 120),
  prob_success = 0.95, prob_futility = 0.05
)

# After the first 40 observations have been collected:
mon <- sequential_analysis(design, data_so_far)
print(mon)

# After 80:
mon <- sequential_analysis(mon, data_so_far)
plot_sequential_monitor(mon)

## End(Not run)
```

---

sequential\_design

*Prespecify a sequential Bayesian analysis design*

---

**Description**

Creates a frozen specification of a sequential Bayesian analysis *before* data collection: the model, the analysis priors, the monitored effect, the decision metric, the stopping thresholds, and the planned look schedule. The returned object is used both to simulate the design's operating characteristics ([brms\\_inla\\_sequential\\_trial\(\)](#)) and to monitor the real study as data accumulate ([sequential\\_analysis\(\)](#)).

The object carries an MD5 fingerprint of all decision-relevant fields. Quoting this fingerprint in a preregistration or protocol allows readers and reviewers to verify that the stopping rules were not altered after data collection began: any change to the model, priors, metric, thresholds, or look schedule produces a different fingerprint.

**Usage**

```
sequential_design(
  formula,
  family = gaussian(),
  priors = NULL,
  effect_name,
  metric = c("direction", "threshold", "rope"),
  alternative = c("greater", "less"),
  looks,
```

```

    prob_success = 0.95,
    prob_futility = NULL,
    effect_threshold = 0,
    rope_bounds = NULL,
    credible_level = 0.95,
    label = NULL
  )

```

### Arguments

formula	Model formula (brms syntax; random effects supported).
family	Response family, e.g. <code>gaussian()</code> , <code>binomial()</code> .
priors	Optional brms prior specification (translated to INLA via the package's prior bridge; see <code>brms_inla_power()</code> ).
effect_name	Single character: the monitored fixed effect.
metric	"direction", "threshold", or "rope".
alternative	"greater" or "less": the prespecified direction for the direction/threshold metrics. Ignored for "rope".
looks	Strictly increasing vector of cumulative sample sizes (total observations) at which interim analyses are planned. The final element is the maximum sample size.
prob_success	Scalar or <code>length(looks)</code> vector of stop-for-success posterior probability thresholds in (0.5, 1). Per-look vectors allow stricter early thresholds (O'Brien-Fleming-like behaviour).
prob_futility	Scalar or <code>length(looks)</code> vector of stop-for-futility thresholds in (0, 0.5), or NULL (default) for no futility stopping. Ignored with a warning for <code>metric = "rope"</code> .
effect_threshold	Threshold for <code>metric = "threshold"</code> (model coefficient scale).
rope_bounds	Length-2 increasing numeric vector for <code>metric = "rope"</code> .
credible_level	Credible level for reported intervals (default 0.95).
label	Optional character label for the study (used in printing).

### Details

Because Bayesian posterior quantities obey the likelihood principle, the interpretation of the posterior at the final look does not depend on the stopping rule. The *frequency* properties of the design (false-go rate, expected sample size, exaggeration of estimates at early stops) do. The recommended workflow is therefore: (1) create the design; (2) estimate its operating characteristics by simulation with `brms_inla_sequential_trial()`; (3) preregister the design, including its fingerprint; (4) monitor the study with `sequential_analysis()`; and (5) report the final posterior alongside the simulated operating characteristics.

### Value

A list of class "powerbrmsINLA\_seq\_design" whose fingerprint element contains the MD5 hash of the decision-relevant fields.

**See Also**

[sequential\\_analysis\(\)](#), [brms\\_inla\\_sequential\\_trial\(\)](#), [plot\\_sequential\\_monitor\(\)](#)

**Examples**

```
design <- sequential_design(
  formula      = strength ~ group,
  effect_name  = "group",
  metric       = "threshold",
  effect_threshold = 0.2,
  looks       = c(40, 80, 120, 160, 200),
  prob_success = 0.95,
  prob_futility = 0.05,
  label       = "Periodisation RCT"
)
print(design)
```

---

validate\_inla\_vs\_brms *Spot-check INLA posterior estimates against brms/Stan*

---

**Description**

Generates a small number of synthetic datasets using the same internal data-generator as [brms\\_inla\\_power\(\)](#), fits each dataset with both INLA (as the package does) and `brms::brm()` (via Stan), and compares the posterior mean, posterior SD, and 95 % credible-interval bounds for the primary effect parameter.

This is a **qualitative spot-check**, not a formal equivalence test. Differences are expected due to the Laplace approximation used by INLA versus full MCMC in Stan, and will vary across models and priors.

**Usage**

```
validate_inla_vs_brms(
  formula,
  family = gaussian(),
  priors = NULL,
  data_generator = NULL,
  effect_name,
  effect_value = 0.5,
  sample_size = 100L,
  n_check = 5L,
  brms_iter = 2000L,
  brms_chains = 4L,
  tolerance = NULL,
  seed = 42L,
  error_sd = 1,
```

```

  group_sd = 0.5,
  obs_per_group = 10,
  predictor_means = NULL,
  predictor_sds = NULL,
  family_args = list(),
  inla_num_threads = NULL
)

```

## Arguments

formula	Model formula (same as <code>brms_inla_power()</code> ).
family	GLM family object (default <code>gaussian()</code> ).
priors	A <code>brms::prior()</code> specification or NULL for defaults. Supported priors are translated to INLA controls where possible and audited in the returned settings.
data_generator	Optional function <code>f(n, effect)</code> returning a data frame. If NULL, the automatic generator is used.
effect_name	Character vector of fixed-effect names (same as <code>brms_inla_power()</code> ).
effect_value	Numeric value (or named vector) of the true effect size used when generating data for the comparison. Default 0.5.
sample_size	Integer sample size for each comparison dataset. Default 100.
n_check	Number of independent datasets to compare. Default 5.
brms_iter	Total MCMC iterations per chain passed to <code>brms::brm()</code> . Default 2000.
brms_chains	Number of MCMC chains. Default 4.
tolerance	Numeric threshold: the flag is set to TRUE when the maximum absolute difference in posterior means exceeds tolerance. If NULL (default), $0.1 * \text{mean}(\text{INLA posterior SD})$ is used.
seed	Integer random seed. Default 42.
error_sd, group_sd, obs_per_group, predictor_means, predictor_sds, family_args	Passed to the automatic data generator when <code>data_generator = NULL</code> (see <code>brms_inla_power()</code> for details).
inla_num_threads	INLA threading string (e.g. "4:1"). Auto-detected when NULL.

## Value

A list with components:

comparisons	Data frame with one row per dataset and columns <code>sim</code> , <code>inla_ok</code> , <code>brms_ok</code> , <code>inla_mean</code> , <code>brms_mean</code> , <code>diff_mean</code> , <code>inla_sd</code> , <code>brms_sd</code> , <code>inla_ci_lower</code> , <code>inla_ci_upper</code> , <code>brms_ci_lower</code> , <code>brms_ci_upper</code> .
flag	TRUE if any <code> diff_mean </code> exceeds tolerance.
max_abs_diff	Maximum observed <code> diff_mean </code> .
tolerance	The tolerance value actually used.
settings	List of key settings for reproducibility.

**Performance warning**

This function runs `n_check` full brms/Stan fits. Even with `brms_chains = 2` and `brms_iter = 1000`, this can take several minutes for non-trivial models. Use `n_check = 3` and `brms_chains = 2` for quick sanity checks.

---

validate_sd_spec	<i>Validate an SD Specification for error_sd or group_sd</i>
------------------	--

---

**Description**

Checks whether the input is a valid positive numeric scalar or one of the supported distributional list specifications. Called automatically by `brms_inla_power()` before the simulation loop; can also be called directly for interactive validation.

**Usage**

```
validate_sd_spec(x, arg_name = "x")
```

**Arguments**

<code>x</code>	A positive numeric scalar <b>or</b> a named list with element <code>dist</code> .
<code>arg_name</code>	Character string used in error messages (default "x").

**Details**

Supported distributional formats:

- `list(dist = "halfnormal", sd = X, location = Y)` — draws  $|Normal(location, sd)|$ ; `location` defaults to 0.
- `list(dist = "lognormal", meanlog = X, sdlog = Y)` — draws from a log-normal distribution.
- `list(dist = "uniform", min = X, max = Y)` — draws from `Uniform(min, max)`; requires `min >= 0`.

**Value**

`x`, invisibly. Called for its side effects (stopping on invalid input).

# Index

add\_decision\_overlay, 3  
assurance\_prior\_weights, 4  
assurance\_prior\_weights(), 19, 20, 23  
  
beta\_binom\_power, 5  
beta\_weights\_on\_grid, 5  
beta\_weights\_on\_grid(), 4, 19, 20, 23  
brms\_inla\_power, 6  
brms\_inla\_power(), 4, 15–17, 19, 20, 22, 33, 35, 41, 46–49  
brms\_inla\_power\_design, 9  
brms\_inla\_power\_parallel, 10  
brms\_inla\_power\_parallel(), 19, 41  
brms\_inla\_power\_sequential, 11  
brms\_inla\_power\_sequential(), 15, 19  
brms\_inla\_power\_two\_stage, 13  
brms\_inla\_power\_two\_stage(), 19, 41  
brms\_inla\_sequential\_trial, 15  
brms\_inla\_sequential\_trial(), 43, 45–47  
  
compute\_assurance, 19  
compute\_assurance(), 4, 13, 22, 23, 26, 35, 41  
  
decide\_sample\_size, 21  
decide\_sample\_size(), 42  
  
hdi\_of\_icdf, 24  
  
min\_n\_beta\_binom, 25  
  
plot\_assurance\_curve, 26  
plot\_assurance\_curve(), 31, 36–39  
plot\_assurance\_with\_robustness, 27  
plot\_bf\_assurance\_curve, 28  
plot\_bf\_assurance\_curve\_smooth, 28  
plot\_bf\_expected\_evidence, 29  
plot\_bf\_heatmap, 30  
plot\_decision\_assurance\_curve, 31  
plot\_decision\_threshold\_contour, 32  
plot\_design\_prior, 33  
  
plot\_interaction\_surface, 34  
plot\_power\_assurance\_overlay, 35  
plot\_power\_contour, 36  
plot\_power\_heatmap, 37  
plot\_precision\_assurance\_curve, 38  
plot\_precision\_fan\_chart, 39  
plot\_sequential\_monitor, 40  
plot\_sequential\_monitor(), 45, 47  
print.brms\_inla\_power, 40  
print.powerbrmsINLA\_assurance, 41  
print.powerbrmsINLA\_sample\_size, 42  
print.powerbrmsINLA\_sample\_size(), 23  
print.powerbrmsINLA\_seq\_design, 42  
print.powerbrmsINLA\_seq\_monitor, 43  
print.powerbrmsINLA\_seq\_trial, 43  
  
sequential\_analysis, 44  
sequential\_analysis(), 16, 40, 45–47  
sequential\_design, 45  
sequential\_design(), 16, 45  
  
validate\_inla\_vs\_brms, 47  
validate\_sd\_spec, 49  
validate\_sd\_spec(), 8