Package 'DTEAssurance'

October 13, 2026

Type Package

Title Assurance Methods for Clinical Trials with a Delayed Treatment Effect

Version 1.3.0

Description Provides functions for planning clinical trials subject to a delayed treatment effect using assurance-based methods. Includes two 'shiny' applications for interactive exploration, simulation, and visualisation of trial designs and outcomes. The methodology is described in: Salsbury JA, Oakley JE, Julious SA, Hampson LV (2024) ``Assurance methods for designing a clinical trial with a delayed treatment effect'' <doi:10.1002/sim.10136>, Salsbury JA, Oakley JE, Julious SA, Hampson LV (2024) ``Adaptive clinical trial design with delayed treatment effects using elicited prior distributions'' <doi:10.48550/arXiv.2509.07602>.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

RoxygenNote 7.3.3

Imports SHELF, stats, survival, nph, nphRCT, dplyr, rpact, magrittr, rlang, future.apply

Suggests testthat (>= 3.0.0), nleqslv, rjags, coda, shiny, shinyAce, shinyBS, shinyjs, rhandsontable, plotly

Config/testthat/edition 3

Depends R (>= 4.0)

URL https://jamesalsbury.github.io/DTEAssurance/

NeedsCompilation no

Author James Salsbury [aut, cre] (ORCID: <https://orcid.org/0000-0002-2584-3640>)

Maintainer James Salsbury <jsalsbury1@sheffield.ac.uk>

Repository CRAN

Date/Publication 2026-10-13 16:10:24 UTC

Contents

add_recruitment_time . . . 2
apply_design_rule . . . 3
apply_GSD_to_trial . . . 4
assurance_adaptive_shiny_app . . . 6
assurance_shiny_app . . . 6
calc_dte_assurance . . . 7
calc_dte_assurance_adaptive . . . 9
calibrate_matched_futility_boundary . . . 11
calibrate_PP_threshold . . . 13
calibrate_PP_timing . . . 16
cens_data . . . 18
compute_relative_floors . . . 19
INTEREST . . . 20
make_gsd_design . . . 21
MCMC_sample . . . 22
PP_func . . . 22
REVEL . . . 25
run_calibration_grid . . . 26
run_paired_scenario . . . 27
select_kappa_star . . . 28
sim_dte . . . 29
single_paired_rep . . . 31
summarize_grid_by_kappa . . . 33
survival_test . . . 34
update_priors . . . 35
ZODIAC . . . 38

Index **39**

`add_recruitment_time` *Add recruitment time to a survival dataset*

Description

Simulates recruitment timing for each patient in a survival dataset using either a power model or a piecewise constant (PWC) model. The function appends recruitment times and pseudo survival times (time from recruitment to event or censoring).

Usage

```
add_recruitment_time(
  data,
  rec_method,
  rec_period = NULL,
  rec_power = NULL,
  rec_rate = NULL,
```

```
  rec_duration = NULL
)
```

Arguments

data	A dataframe containing survival data with columns: time, status, and group
rec_method	Recruitment method: "power" for power model or "PWC" for piecewise constant model
rec_period	Period length for the power model
rec_power	Power parameter for the power model
rec_rate	Comma-separated string of recruitment rates for the PWC model
rec_duration	Comma-separated string of durations corresponding to each rate in the PWC model

Value

A dataframe with two additional columns:

rec_time Simulated recruitment time for each patient

pseudo_time Time from recruitment to event or censoring

Class: data.frame

Examples

```
set.seed(123)
df <- data.frame(
  time = rexp(20, rate = 0.1),
  status = rbinom(20, 1, 0.8),
  group = rep(c("Control", "Treatment"), each = 10)
)
recruited <- add_recruitment_time(df, rec_method = "power", rec_period = 12, rec_power = 1)
head(recruited)
```

---

apply_design_rule *Apply a design rule to paired replicate output*

Description

Turns the output of run_paired_scenario into the decisions of one design, post hoc. The efficacy look is only checked if the trial was not stopped at the futility look. A missing test statistic never crosses a boundary (as in apply_GSD_to_trial).

Usage

```
apply_design_rule(raw, rule)
```

Arguments

raw	The raw element returned by run_paired_scenario.
rule	A list with type ("D1" to "D5") and the rule's parameters: crit_final (D1); crit_eff, crit_fin (D2-D5); kappa (D3); z_fut (D4, D5); mw_t_star (D5).

Details

D1 Fixed design: success iff Z_fin_LRT > crit_final.

D2 Efficacy look then final: "Stop for efficacy" if Z_eff_LRT > crit_eff, otherwise the final analysis with Z_fin_LRT > crit_fin.

D3 D2 plus PP futility: "Stop for futility" if PP_val < kappa. Stops with an error if PP_val is NA for any replicate that did not fail (e.g. output from compute_PP = FALSE).

D4 D2 plus Z futility: "Stop for futility" if Z_int_LRT < z_fut.

D5 As D4 with the modestly weighted statistics Z_int_MW_t<t>, Z_eff_MW_t<t>, Z_fin_MW_t<t> for t = mw_t_star.

D1, D2, D4 and D5 do not use PP_val, so they also work on output from run_paired_scenario(compute_PP = FALSE).

Value

A data frame with one row per replicate and columns decision ("Stop for efficacy", "Stop for futility", "Successful at final" or "Unsuccessful at final"), success, early_fut, early_eff, sample_size and duration. Rows for failed replicates (non-missing error) are NA.

See Also

run_paired_scenario, summarize_grid_by_kappa

---

apply_GSD_to_trial *Apply a group-sequential design to one simulated trial*

Description

Runs the looks of a group-sequential design on one simulated trial, in order, stopping at the first look whose rule is met: efficacy if Z exceeds the efficacy critical value, futility under the design's futility rule ("Beta": Z below the beta-spending bound; "MatchedZ": Z below GSD_model$futility_boundary_Z; "PP": predictive probability below GSD_model$kappa), and otherwise the final analysis. Each look cuts the trial with cens_data at floor(IF * total_events) events. This is the engine of calc_dte_assurance_adaptive. For simulation studies, use run_paired_scenario and apply_design_rule, which evaluate all designs on the same replicates.

Usage

```
apply_GSD_to_trial(
  n_c,
  n_t,
  trial_data,
  design,
  total_events,
  GSD_model,
  control_model = NULL,
  effect_model = NULL,
  recruitment_model = NULL,
  analysis_model = NULL,
  update_priors_sims = NULL,
  PP_sims = NULL,
  .PP_val = NULL
)
```

Arguments

n_c, n_t	Planned number of patients in the control / treatment arm.
trial_data	A simulated trial with columns time, group, rec_time and pseudo_time.
design	The rpact design, from make_gsd_design.
total_events	Maximum planned number of events.
GSD_model	The design specification (see calc_dte_assurance_adaptive).
control_model, effect_model, recruitment_model	Analysis prior and recruitment model, used only for "PP" futility (recruitment must be method = "power" with power = 1).
analysis_model	Test specification (method, alpha, alternative_hypothesis, and rho, gamma, t_star, s_star as needed); defaults to a one-sided LRT at 0.025.
update_priors_sims, PP_sims	MCMC samples per chain and predictive draws; required for "PP" futility.
.PP_val	Internal, for testing: a precomputed predictive probability to use at the PP futility look instead of computing it.

Value

A list with decision ("Stop for efficacy", "Stop for futility", "Successful at final" or "Unsuccessful at final"), stop_time, sample_size, PP_val, converged and Z_probs.

assurance_adaptive_shiny_app *Launch the 'shiny' adaptive assurance app*

Description

Launches a 'shiny' application to simulate group sequential trials with delayed treatment effects (DTE) using elicited prior distributions. The app allows interactive exploration of trial designs and assurance calculations.

Usage

```
assurance_adaptive_shiny_app()
```

Value

No return value, called for side effects (invisibly returns NULL). The function launches an interactive 'shiny' application.

Examples

```
# Launch the interactive Shiny app
assurance_adaptive_shiny_app()
```

assurance_shiny_app *Launch the 'shiny' Assurance app*

Description

Launches a 'shiny' application to calculate assurance for clinical trials where delayed treatment effects (DTE) may be present. The app allows elicitation of prior distributions and calculates assurance metrics.

Usage

```
assurance_shiny_app()
```

Value

No return value, called for side effects (invisibly returns NULL). The function launches an interactive 'shiny' application.

Examples

```
# Launch the interactive Shiny app
assurance_shiny_app()
```

`calc_dte_assurance` *Calculate Assurance for a Trial with a Delayed Treatment Effect*

Description

Simulates operating characteristics for a clinical trial under prior uncertainty about a delayed treatment effect. The function integrates beliefs about control survival, treatment delay, post-delay hazard ratio, recruitment, censoring, and analysis method to estimate assurance and other trial metrics.

Usage

```
calc_dte_assurance(
  n_c,
  n_t,
  control_model,
  effect_model,
  censoring_model,
  recruitment_model,
  analysis_model,
  n_sims = 1000
)
```

Arguments

`n_c`	Vector of control group sample sizes
`n_t`	Vector of treatment group sample sizes
`control_model`	A named list specifying the control arm survival distribution: • `dist`: Distribution type ("Exponential" or "Weibull") • `parameter_mode`: Either "Fixed" or "Distribution" • `fixed_type`: If "Fixed", specify as "Parameters" or "Landmark" • `lambda`, `gamma`: Scale and shape parameters • `t1`, `t2`: Landmark times • `surv_t1`, `surv_t2`: Survival probabilities at landmarks • `t1_Beta_a`, `t1_Beta_b`, `diff_Beta_a`, `diff_Beta_b`: Beta prior parameters
`effect_model`	A named list specifying beliefs about the treatment effect: • `delay_SHELF`, `HR_SHELF`: SHELF objects encoding beliefs • `delay_dist`, `HR_dist`: Distribution types ("hist" by default) • `P_S`: Probability that survival curves separate • `P_DTE`: Probability of delayed separation, conditional on separation
`censoring_model`	A named list specifying the censoring mechanism: • `method`: "Time", "Events", or "IF"

- time, events, IF: Parameters for each method

recruitment_model
: A named list specifying the recruitment process:
  - method: "power" or "PWC"
  - period, power: Parameters for power model
  - rate, duration: Comma-separated strings for PWC model

analysis_model
: A named list specifying the statistical test and decision rule:
  - method: "LRT", "WLRT", or "MW"
  - alpha, alternative_hypothesis: Type I error and hypothesis direction
  - rho, gamma, t_star, s_star: Parameters for WLRT or MW
  - success_threshold_HR: Optional threshold for declaring success

n_sims
: Number of simulations to run (default = 1000)

Value

A named list containing:

assurance Estimated assurance (probability of success under prior uncertainty)

CI 95% confidence interval for assurance

duration Mean trial duration across simulations

sample_size Mean sample size across simulations

diagnostics Additional diagnostics if success_threshold_HR is specified

Class: list

Examples

```
set.seed(123)
control_model <- list(dist = "Exponential", parameter_mode = "Fixed",
fixed_type = "Parameters", lambda = 0.1)
effect_model <- list(delay_SHELF = SHELF::fitdist(c(3, 4, 5),
probs = c(0.25, 0.5, 0.75), lower = 0, upper = 10),
delay_dist = "gamma",
HR_SHELF = SHELF::fitdist(c(0.55, 0.6, 0.7), probs = c(0.25, 0.5, 0.75), lower = 0, upper = 1.5),
HR_dist = "gamma",
P_S = 1, P_DTE = 0)
censoring_model <- list(method = "Time", time = 12)
recruitment_model <- list(method = "power", period = 12, power = 1)
analysis_model <- list(method = "LRT", alpha = 0.025, alternative_hypothesis = "two.sided")
result <- calc_dte_assurance(n_c = 300, n_t = 300,
                                    control_model = control_model,
                                    effect_model = effect_model,
                                    censoring_model = censoring_model,
                                    recruitment_model = recruitment_model,
                                    analysis_model = analysis_model,
                                    n_sims = 10)
str(result)
```

`calc_dte_assurance_adaptive` *Calculates operating characteristics for a Group Sequential Trial with a Delayed Treatment Effect*

Description

Simulates assurance and operating characteristics for a group sequential trial under prior uncertainty about a delayed treatment effect. The function integrates beliefs about control survival, treatment delay, post-delay hazard ratio, recruitment, and group sequential design (GSD) parameters.

Usage

```
calc_dte_assurance_adaptive(
  n_c,
  n_t,
  control_model,
  effect_model,
  recruitment_model,
  GSD_model,
  analysis_model = NULL,
  update_priors_sims = NULL,
  PP_sims = NULL,
  n_sims = 1000
)
```

Arguments

`n_c`	Control group sample size
`n_t`	Treatment group sample size
`control_model`	A named list specifying the control arm survival distribution: • `dist`: Distribution type ("Exponential" or "Weibull") • `parameter_mode`: Either "Fixed" or "Distribution" • `fixed_type`: If "Fixed", specify as "Parameters" or "Landmark" • `lambda`, `gamma`: Scale and shape parameters • `t1`, `t2`: Landmark times • `surv_t1`, `surv_t2`: Survival probabilities at landmarks • `t1_Beta_a`, `t1_Beta_b`, `diff_Beta_a`, `diff_Beta_b`: Beta prior parameters
`effect_model`	A named list specifying beliefs about the treatment effect: • `delay_SHELF`, `HR_SHELF`: SHELF objects encoding beliefs • `delay_dist`, `HR_dist`: Distribution types ("hist" by default) • `P_S`: Probability that survival curves separate • `P_DTE`: Probability of delayed separation, conditional on separation

`recruitment_model`
: A named list specifying the recruitment process:
  - `method`: "power" or "PWC"
  - `period`, `power`: Parameters for power model
  - `rate`, `duration`: Comma-separated strings for PWC model

`GSD_model`
: A named list specifying the group sequential design:
  - `events`: Total number of events
  - `alpha_spending`: User-specified *cumulative* alpha spent by each efficacy look (one value per element of `alpha_IF`; the design uses `rpact`'s `typeOfDesign = "asUser"`). Futility looks are added to the design with zero alpha spent, so they do not change the efficacy critical values.
  - `alpha_IF`: Information Fraction(s) at which we look for efficacy
  - `futility_type`: One of `"none"`, `"Beta"` (pre-specified beta-spending, via `rpact`), `"PP"` (predictive probability futility, D3-style), or `"MatchedZ"` (a fixed, externally-calibrated Z-statistic cutoff, D4/D5-style; the rule is applied as binding, i.e. the simulated trial stops when Z falls below the cutoff; see `calibrate_matched_futility_boundary` for how to obtain `futility_boundary_Z`).
  - `futility_IF`: Information Fraction at which we look for futility (required for `"PP"` and `"MatchedZ"`).
  - `beta_spending`: Cumulative beta spending vector (`"Beta"` only).
  - `kappa`: PP value below which we stop for futility (`"PP"` only).
  - `futility_boundary_Z`: Z-statistic value below which we stop for futility (`"MatchedZ"` only).

`analysis_model`
: A named list specifying the final analysis and decision rule:
  - `method`: e.g. `"LRT"`, `"WLRT"`, or `"MW"`.
  - `alpha`: one-sided type I error level.
  - `alternative_hypothesis`: direction of the alternative (e.g. `"one.sided"`).
  - `rho`, `gamma`, `t_star`, `s_star`: additional parameters for WLRT or MW (if applicable).

`update_priors_sims`
: Number of posterior samples per chain for each interim dataset, passed to `update_priors`. Required when `GSD_model$futility_type == "PP"`; ignored otherwise.

`PP_sims`
: Number of predictive simulations per interim dataset, passed to `PP_func`. Required when `GSD_model$futility_type == "PP"`; ignored otherwise.

`n_sims`
: Number of simulations to run (default = 1000)

Value

A data frame with one row per simulated trial and the following columns:

Trial Simulation index

Decision Final interim/final decision outcome – one of `"Stop for efficacy"`, `"Stop for futility"`, `"Successful at final"`, or `"Unsuccessful at final"`. This is the single source of truth for trial outcome; use it directly rather than deriving success/failure independently.

StopTime Time at which the trial stopped or completed

SampleSize Total sample size at the time of decision

Success Logical recode of `Decision` for convenience: `TRUE` if `Decision %in% c("Stop for efficacy", "Successful at final")`, `FALSE` otherwise.

Converged For `"PP"` designs, whether the interim MCMC fit converged (see `update_priors`); `NA` for other futility types, which involve no MCMC step.

PP_val For `"PP"` designs, the predictive probability at the futility look; `NA` otherwise (or if the trial stopped for efficacy before the futility look).

P_Z1, P_Z2, P_Z3 For `"PP"` designs, the posterior probabilities of the three latent states at the futility look; `NA` otherwise.

Class: `data.frame`, with attribute `settings`: a list of all arguments, plus the package, R, rjags and JAGS versions, the git commit of the working directory, the hostname and a timestamp.

Examples

```
set.seed(123)
control_model <- list(dist = "Exponential", parameter_mode = "Fixed",
fixed_type = "Parameters", lambda = 0.1)
effect_model <- list(P_S = 1, P_DTE = 0,
HR_SHELF = SHELF::fitdist(c(0.6, 0.65, 0.7), probs = c(0.25, 0.5, 0.75), lower = 0, upper = 2),
HR_dist = "gamma",
delay_SHELF = SHELF::fitdist(c(3, 4, 5), probs = c(0.25, 0.5, 0.75), lower = 0, upper = 10),
delay_dist = "gamma"
)
recruitment_model <- list(method = "power", period = 12, power = 1)
GSD_model <- list(events = 300, alpha_spending = c(0.0125, 0.025),
                  alpha_IF = c(0.75, 1), futility_type = "none")
result <- calc_dte_assurance_adaptive(n_c = 300, n_t = 300,
                         control_model = control_model,
                         effect_model = effect_model,
                         recruitment_model = recruitment_model,
                         GSD_model = GSD_model,
                         n_sims = 10)
str(result)
```

---

calibrate_matched_futility_boundary

Calibrate a fixed Z-statistic futility boundary

Description

Finds the Z-statistic futility boundary at `futility_IF` such that the probability of stopping for futility under the `"null"` scenario equals `target_null_futility_rate` (the corresponding empirical quantile of the simulated null Z distribution), and reports the resulting futility-stopping rate under every supplied scenario. The result is used as `GSD_model$futility_boundary_Z` with `GSD_model$futility_type = "MatchedZ"` in `calc_dte_assurance_adaptive`.

Usage

```
calibrate_matched_futility_boundary(
  n_c,
  n_t,
  recruitment_model,
  futility_IF,
  total_events,
  analysis_model,
  target_null_futility_rate,
  scenarios,
  n_sims = 2000,
  n_cores = 1,
  seed = NULL
)
```

Arguments

n_c, n_t	Number of patients in the control / treatment group.
recruitment_model	Recruitment specification (see add_recruitment_time).
futility_IF	Information fraction of the futility look.
total_events	Maximum planned number of events.
analysis_model	Analysis specification (see survival_test).
target_null_futility_rate	Target probability of stopping for futility under the null scenario.
scenarios	Named list of data-generating scenarios (see single_matched_futility_rep); must include one named "null".
n_sims	Number of simulated trials per scenario (default 2000).
n_cores	Number of cores (uses parallel::mclapply when > 1).
seed	Optional integer seed.

Value

A list with boundary, scenario_futility_rates, raw_Z_by_scenario and settings (all arguments, plus the package, R, rjags and JAGS versions, the git commit of the working directory, the hostname and a timestamp).

Examples

```
scenarios <- list(
  null = list(lambda_c = log(2) / 12, delay_time = 0, post_delay_HR = 1),
  alt  = list(lambda_c = log(2) / 12, delay_time = 3, post_delay_HR = 0.6)
)
cal <- calibrate_matched_futility_boundary(
  n_c = 100, n_t = 100,
  recruitment_model = list(method = "power", period = 12, power = 1),
  futility_IF = 0.5, total_events = 120,
```

```
  analysis_model = list(method = "LRT", alpha = 0.025,
                        alternative_hypothesis = "one.sided"),
  target_null_futility_rate = 0.5,
  scenarios = scenarios, n_sims = 20, seed = 1)
cal$boundary
cal$scenario_futility_rates
```

`calibrate_PP_threshold` *Function to calculate the 'optimal' PP threshold value*

Description

Function to calculate the 'optimal' PP threshold value

Usage

```
calibrate_PP_threshold(
  n_c,
  n_t,
  control_model,
  effect_model,
  recruitment_model,
  IA_model,
  analysis_model,
  data_generating_model,
  future_boundaries = NULL,
  n_df_sims = 100,
  update_priors_sims = 1000,
  PP_sims = 2000,
  n_cores = 1,
  seed = NULL
)
```

Arguments

`n_c`	Number of control patients
`n_t`	Number of treatment patients
`control_model`	A named list specifying the control arm survival distribution (see `update_priors` for details).
`effect_model`	A named list specifying beliefs about the treatment effect (see `update_priors` for details).
`recruitment_model`	A named list specifying the recruitment process (see `add_recruitment_time` for details).

`IA_model` A named list specifying the interim analysis timing:

- `events`: total planned event count (100% information fraction).
- `IF`: the information fraction at which the interim futility look occurs (i.e. where the data are censored to compute PP).

`analysis_model` A named list specifying the analysis method (see `PP_func` for details).

`data_generating_model`
A named list specifying the true data-generating parameters used to simulate trials for calibration:

- `lambda_c`: hazard rate for the control group.
- `gamma_c`: Weibull shape parameter (`NULL` for Exponential).
- `delay_time`: true delay before the treatment effect begins.
- `post_delay_HR`: true post-delay hazard ratio.

`future_boundaries`
Recommended. A list of future, pre-specified, fixed decision points (efficacy look(s) and the final analysis) to check on each posterior-predictive draw, matching the true group-sequential design under which this PP threshold will actually be used – see `PP_func` for the exact structure. If `NULL` (not recommended, kept only for backward compatibility), PP is computed via the legacy single-stage fallback: censoring directly to `IA_model$events` and testing at a flat `analysis_model$alpha`, which does NOT account for any future efficacy boundary the real design may have. A message is emitted if this fallback is used.

`n_df_sims` Number of interim datasets to simulate (default is 100). For calibration decisions near a specific power/Type I error target, consider a substantially larger number and report Monte Carlo standard errors alongside the result.

`update_priors_sims`
Number of posterior samples to generate per interim dataset via `update_priors` (default is 1000).

`PP_sims` Number of predictive simulations used to estimate PP for each interim dataset, passed to `PP_func` (default is 2000).

`n_cores` Number of cores to parallelise over via `parallel::mclapply` (default is 1, i.e. sequential `lapply`; not supported on Windows for `n_cores > 1`, per base R's `mclapply` limitations).

`seed` Optional integer seed. If supplied, each simulated interim dataset `i` is seeded reproducibly as `seed * 10000 + i`. If `NULL` (default), no seed is set internally – set one in the calling script if reproducibility across runs is required.

Value

A list with:

PP_vec A numeric vector of length `n_df_sims`, the estimated PP for each simulated interim dataset.

settings A list of all arguments, plus the package, R, rjags and JAGS versions, the git commit of the working directory, the hostname and a timestamp.

Deprecated

`calibrate_PP_threshold()` is deprecated and will be removed in a future release. Use `run_paired_scenario` with `summarize_grid_by_kappa` and `select_kappa_star`.

Examples

```
set.seed(123)
control_model <- list(dist = "Exponential", parameter_mode = "Distribution",
                      t1 = 12, t1_Beta_a = 20, t1_Beta_b = 32)

effect_model <- list(delay_SHELF = SHELF::fitdist(c(5.5, 6, 6.5),
                    probs = c(0.25, 0.5, 0.75), lower = 0, upper = 12),
                    delay_dist = "gamma",
                    HR_SHELF = SHELF::fitdist(c(0.5, 0.6, 0.7),
                    probs = c(0.25, 0.5, 0.75), lower = 0, upper = 1),
                    HR_dist = "gamma",
                    P_S = 1, P_DTE = 0)

recruitment_model <- list(method = "power", period = 12, power = 1)

IA_model <- list(events = 40, IF = 0.5)

analysis_model <- list(method = "LRT", alpha = 0.025,
                       alternative_hypothesis = "one.sided")

data_generating_model <- list(lambda_c = log(2)/12, gamma_c = NULL,
                              delay_time = 3, post_delay_HR = 0.75)

# A single future efficacy look at 30 events (Z > 2.24), then final at 40
# events (Z > 2.00) -- match this to the true design's own boundaries.
future_boundaries <- list(
  list(events = 30, crit = 2.24),
  list(events = 40, crit = 2.00)
)

# Deliberately small settings so the example runs quickly. For a real
# analysis use the defaults (update_priors_sims = 1000, PP_sims = 2000)
# and a larger n_df_sims.
threshold <- calibrate_PP_threshold(n_c = 25, n_t = 25,
                    control_model = control_model,
                    effect_model = effect_model,
                    recruitment_model = recruitment_model,
                    IA_model = IA_model,
                    analysis_model = analysis_model,
                    data_generating_model = data_generating_model,
                    future_boundaries = future_boundaries,
                    update_priors_sims = 100,
                    PP_sims = 100,
                    n_df_sims = 2)
```

`calibrate_PP_timing` *Function to calculate the 'optimal' information fraction to calculate PP*

Description

Function to calculate the 'optimal' information fraction to calculate PP

Usage

```
calibrate_PP_timing(
  n_c,
  n_t,
  control_model,
  effect_model,
  recruitment_model,
  IA_model,
  analysis_model,
  future_boundaries = NULL,
  update_priors_sims = 1000,
  PP_sims = 2000,
  n_sims = 100,
  n_cores = 1,
  seed = NULL
)
```

Arguments

`n_c`	Number of control patients
`n_t`	Number of treatment patients
`control_model`	A named list specifying the control arm survival distribution (see `update_priors` for details).
`effect_model`	A named list specifying beliefs about the treatment effect (see `update_priors` for details).
`recruitment_model`	A named list specifying the recruitment process (see `add_recruitment_time` for details).
`IA_model`	A named list specifying the censoring mechanism for the future data: • `events`: Number of events which is 100% information fraction • `IF`: A vector of candidate information fractions to evaluate. **Every candidate must genuinely precede** `future_boundaries` (see below) – e.g. if the true design has an efficacy look at IF = 0.75, do not include candidates at or beyond 0.75 in this sweep; a futility look scheduled at or after the design's own efficacy look is not a well-posed question about that design.

`analysis_model` A named list specifying the final analysis and decision rule (see `PP_func` for details).

`future_boundaries`
Recommended. The chain of fixed future decision points (efficacy look(s) and final analysis) of the true group-sequential design under consideration – see `PP_func`. The same chain is used for every candidate in `IA_model$IF` (each candidate is checked to ensure it genuinely precedes every boundary in the chain; see `single_calibration_rep()`). If `NULL` (not recommended), falls back to the legacy single-stage calculation, and a message is emitted.

`update_priors_sims`
Number of posterior samples per interim dataset (default 1000).

`PP_sims` Number of predictive simulations per interim dataset (default 2000).

`n_sims` Number of interim datasets to simulate per candidate timing (default is 100).

`n_cores` Number of cores to parallelise over via `parallel::mclapply` (default is 1, i.e. sequential `lapply`).

`seed` Optional integer seed; if supplied, replicate `i` under candidate timing `k` is seeded as `seed * 10000 + i` (note: replicates are re-seeded identically across candidate timings, so the same underlying trial trajectories are reused – i.e. paired comparison across timings – unless you deliberately want independent draws per timing, in which case pass a different `seed` per call).

Value

A list with:

outcome_list A list, one element per candidate information fraction, each containing `PP_values` (a vector of estimated PP, one per simulated interim dataset) and `cens_time` (the corresponding calendar times).

settings A list of all arguments, plus the package, R, rjags and JAGS versions, the git commit of the working directory, the hostname and a timestamp.

Examples

```
set.seed(123)
control_model <- list(dist = "Exponential", parameter_mode = "Distribution",
                      t1 = 12, t1_Beta_a = 20, t1_Beta_b = 32)

effect_model <- list(delay_SHELF = SHELF::fitdist(c(5.5, 6, 6.5),
                    probs = c(0.25, 0.5, 0.75), lower = 0, upper = 12),
                    delay_dist = "gamma",
                    HR_SHELF = SHELF::fitdist(c(0.5, 0.6, 0.7),
                    probs = c(0.25, 0.5, 0.75), lower = 0, upper = 1),
                    HR_dist = "gamma",
                    P_S = 1, P_DTE = 0)

recruitment_model <- list(method = "power", period = 12, power = 1)

# Sweep candidates from 0.2 to 0.7 only -- all genuinely precede the
# design's own efficacy look at IF = 0.75.
```

```
# Two candidate timings, both before the design's efficacy look at IF = 0.75.
IA_model <- list(events = 40, IF = c(0.4, 0.6))

analysis_model <- list(method = "LRT", alpha = 0.025,
                       alternative_hypothesis = "one.sided")

future_boundaries <- list(
  list(events = 30, crit = 2.24),   # efficacy look at IF = 0.75
  list(events = 40, crit = 2.00)    # final analysis
)

# Deliberately small settings so the example runs quickly. For a real
# analysis use the defaults (update_priors_sims = 1000, PP_sims = 2000)
# and a larger n_sims.
timing <- calibrate_PP_timing(n_c = 25, n_t = 25,
                      control_model = control_model,
                      effect_model = effect_model,
                      recruitment_model = recruitment_model,
                      IA_model = IA_model,
                      analysis_model = analysis_model,
                      future_boundaries = future_boundaries,
                      update_priors_sims = 100,
                      PP_sims = 100,
                      n_sims = 2)
```

`cens_data` *Censor a survival dataset*

Description

Applies administrative censoring to a survival dataset using one of three methods: fixed time, fixed number of events, or fixed information fraction. The input data must contain columns for pseudo survival time, recruitment time, and observed time.

Usage

```
cens_data(
  data,
  cens_method = "Time",
  cens_time = NULL,
  cens_IF = NULL,
  cens_events = NULL
)
```

Arguments

`data`	A dataframe containing uncensored survival data with columns: `pseudo_time`, `rec_time`, and `time`

cens_method	Censoring method: "Time" (default), "Events", or "IF"
cens_time	Time point for censoring (required if cens_method = "Time")
cens_IF	Information fraction for censoring (required if cens_method = "IF")
cens_events	Number of events for censoring (required if cens_method = "Events")

Details

This is the single routine used throughout the package to cut a trial at an analysis. The cut is made at calendar time t, the calendar time of the k-th event ("Events"); patients recruited before t are included, and a patient has an event if pseudo_time <= t. Cutting at k events therefore leaves exactly k events in the data (assuming no tied event times). cens_events is floored with a small tolerance, so a non-integer such as 840 * 0.3 is treated as 252.

Value

A list containing:

data Censored dataframe with updated status and filtered rows

cens_events Number of events used for censoring (if applicable)

cens_time Time point used for censoring

sample_size Number of patients remaining after censoring

Examples

```
set.seed(123)
df <- data.frame(
  pseudo_time = rexp(20, rate = 0.1),
  rec_time = runif(20, 0, 12),
  time = rexp(20, rate = 0.1)
)
censored <- cens_data(df, cens_method = "Time", cens_time = 10)
str(censored)
```

---

compute_relative_floors

Relative power floors for kappa selection

Description

For each alternative scenario, the D2 power (the power with no futility stopping, i.e. at kappa = 0) minus margin. Pass the result as power_floor to select_kappa_star.

Usage

```
compute_relative_floors(raw_by_scenario, alt_scenarios, margin = 0.05)
```

Arguments

raw_by_scenario	A named list of raw replicate data frames (each with a continuation_success column).
alt_scenarios	Character vector of alternative scenario names.
margin	Allowed loss of power relative to D2 (default 0.05).

Value

A named numeric vector with one floor per alternative scenario.

See Also

select_kappa_star

INTEREST *INTEREST data set*

Description

A reconstructed survival data set for the INTEREST clinical trial

Usage

```
INTEREST
```

Format

A data frame with 710 rows and 2 variables:

Survival time Survival Time (in months)

Status Event indicator (0=Alive, 1=Dead)

Source

Reconstructed survival data set from the following publication: https://www.sciencedirect.com/science/article/pii/S01406736

make_gsd_design *Build the group-sequential design for a GSD_model*

Description

Builds the rpact group-sequential design used by calc_dte_assurance_adaptive, apply_GSD_to_trial and single_paired_rep. GSD_model$alpha_spending is the user-specified *cumulative* alpha spent at each efficacy look GSD_model$alpha_IF (rpact's typeOfDesign = "asUser"). For "PP" and "MatchedZ" futility, the futility look (futility_IF) is added to the information-rate grid with zero alpha spent, so its critical value is Inf and the efficacy critical values are exactly those of the efficacy-only design. For "Beta" futility, GSD_model$beta_spending is the cumulative beta spent at each futility_IF (typeBetaSpending = "bsUser"). Information fractions are rounded to 6 dp before they are compared.

Usage

```
make_gsd_design(GSD_model)

make_rpact_design_from_GSD_model(GSD_model)
```

Arguments

GSD_model	A named list with alpha_IF, alpha_spending, futility_type ("none", "Beta", "PP" or "MatchedZ") and, for futility designs, futility_IF (and beta_spending for "Beta"). See calc_dte_assurance_adaptive.

Details

make_rpact_design_from_GSD_model() is the previous name, kept as an alias for one release.

Value

A list with design (the rpact design object; design$informationRates and design$criticalValues give the looks and their Z critical values), IF_all, alpha_spending_full and beta_spending_full.

Examples

```
# Efficacy look at 75% information and final analysis, with a PP futility
# look at 50% information (zero alpha spent there)
gsd <- make_gsd_design(list(alpha_IF = c(0.75, 1),
                            alpha_spending = c(0.0125, 0.025),
                            futility_type = "PP", futility_IF = 0.5))
gsd$design$informationRates
gsd$design$criticalValues
```

MCMC_sample *MCMC_sample*

Description

An MCMC sample for the example given in Salsbury et al (2024)

Usage

```
MCMC_sample
```

Format

A data frame with 100000 rows and 1 variables:

x Sample from the MAP prior

Source

A MCMC sample for the control group for the example given in https://onlinelibrary.wiley.com/doi/full/10.1002/sim.10136. Three historical data seta are used to generate a Meta-Analytic-Predictive Prior distribution

PP_func *Calculate the predictive probability (PP) given interim data and posterior samples*

Description

Calculate the predictive probability (PP) given interim data and posterior samples

Usage

```
PP_func(
  data,
  posterior_df,
  control_distribution = "Exponential",
  n_c_planned,
  n_t_planned,
  rec_time_planned,
  df_cens_time,
  analysis_model,
  censoring_model = NULL,
  future_boundaries = NULL,
  n_sims
)
```

Arguments

data	A data frame containing interim survival data, censored at df_cens_time, with columns: • time Final observed/event time at the interim (on the analysis time scale). • group Treatment group indicator (e.g. "Control", "Treatment"). • rec_time Recruitment (calendar) time. • pseudo_time time + rec_time (calendar time at event/censoring). • status Event indicator at the interim (1 = event, 0 = censored). • survival_time Observed follow-up time from randomisation to event/censoring at the interim.
posterior_df	A data frame of posterior samples with columns: lambda_c, delay_time and HR, corresponding to the control hazard, the delay (changepoint) time and the post-delay hazard ratio, respectively, plus gamma_c (the Weibull shape) when control_distribution = "Weibull".
control_distribution	Distributional form assumed for the control arm: either "Exponential" (default) or "Weibull".
n_c_planned	Planned maximum number of patients in the control group.
n_t_planned	Planned maximum number of patients in the treatment group.
rec_time_planned	Planned maximum recruitment calendar time for the full trial.
df_cens_time	Calendar time at which df has been censored (interim analysis time).
analysis_model	A named list specifying the analysis method and decision rule: • method: e.g. "LRT", "WLRT", or "MW". • alpha: one-sided type I error level (only used in the legacy fallback path, see censoring_model below). • alternative_hypothesis: direction of the alternative (e.g. "one.sided"). • rho, gamma, t_star, s_star: additional parameters for WLRT or MW (if applicable).
censoring_model	**Legacy fallback, deprecated** (see the Deprecated section). Used only if future_boundaries is NULL. A named list specifying a single censoring mechanism for the future data (method: one of "Time", "Events", or "IF"; plus the corresponding time/events/IF parameter), with success determined by analysis_model$alpha rather than a design-specific critical value.
future_boundaries	**Recommended.** A list of future, pre-specified, fixed decision points to evaluate on each posterior-predictive draw, in ascending chronological order, matching the trial's own group-sequential design. Each element is a list with: • events: the cumulative event count at that analysis (the final element should be the maximum planned event count). • crit: the pre-specified critical value for the test statistic Z at that analysis (e.g. from design$criticalValues of an rpact group-sequential design).

On each predictive draw, boundaries are checked in order: if Z crosses a boundary, the draw is recorded as successful and evaluation stops (mirroring the group-sequential design's own early-stopping logic); if the final boundary in the chain is reached without crossing, the draw is recorded as unsuccessful. This evaluates the trial-success event: crossing any future efficacy boundary, or rejecting H_0 at the final analysis.

Scope: future_boundaries must contain only fixed (non-Bayesian) decision points. A design with more than one future Bayesian-predictive-probability futility look is not supported here, as it would require nested posterior-predictive simulation at each look; see the package/manuscript Limitations.

n_sims — Number of posterior-predictive draws (required, no default).

Details

Model. The control arm is Weibull with rate lambda_c and shape gamma_c; an exponential control arm (control_distribution = "Exponential") is handled as the Weibull case with gamma_c = 1, using the same code path. The treatment arm follows the control hazard up to delay_time and the control hazard multiplied by HR afterwards. Each predictive draw samples one row of posterior_df, simulates event times for patients not yet recruited, and simulates residual event times for patients censored at the interim, conditional on their follow-up so far.

Recruitment assumption. Recruitment times for patients not yet recruited at the interim are drawn from Uniform(df_cens_time, rec_time_planned). This is correct for uniform recruitment (the "power" recruitment model with power = 1) only; the simulation functions that call PP_func() stop if another recruitment model is supplied.

Value

A list with a single element PP_df, a data frame with columns success (0/1, whether this predictive draw ultimately rejects H_0, accounting for any future fixed efficacy boundaries) and Z_val (the test statistic at the analysis where the draw was decided). The predictive probability is mean(PP_df$success).

Deprecated

The exponential control arm (control_distribution = "Exponential") and the legacy single-stage censoring_model fallback are deprecated and will be removed in a future release. Use control_distribution = "Weibull" (an exponential control arm is the Weibull case with gamma_c = 1) and future_boundaries.

Examples

```
set.seed(123)
n <- 30
cens_time <- 15

time <- runif(n, 0, 12)
rec_time <- runif(n, 0, 12)

df <- data.frame(
  time = time,
```

```
  group = c(rep("Control", n/2), rep("Treatment", n/2)),
  rec_time = rec_time
)

df$pseudo_time <- df$time + df$rec_time
df$status <- df$pseudo_time < cens_time
df$survival_time <- ifelse(df$status == TRUE, df$time, cens_time - df$rec_time)

posterior_df <- data.frame(HR = rnorm(20, mean = 0.75, sd = 0.05),
                           delay_time = rep(0, 20),
                           lambda_c = rnorm(20, log(2)/9, sd = 0.01))

analysis_model <- list(method = "LRT", alpha = 0.025,
                       alternative_hypothesis = "one.sided")

# Recommended usage: pass the design's own future boundaries, e.g. a single
# efficacy look at 20 events (Z > 2.24) followed by a final analysis at
# 28 events (Z > 2.00):
future_boundaries <- list(
  list(events = 20, crit = 2.24),
  list(events = 28, crit = 2.00)
)

PP_outcome <- PP_func(df, posterior_df,
            control_distribution = "Exponential",
            n_c_planned = n/2, n_t_planned = n/2,
            rec_time_planned = 12, df_cens_time = 15,
            analysis_model = analysis_model,
            future_boundaries = future_boundaries,
            n_sims = 10)
```

REVEL *REVEL data set*

Description

A reconstructed survival data set for the REVEL clinical trial

Usage

```
REVEL
```

Format

A data frame with 625 rows and 2 variables:

Survival time Survival Time (in months)

Status Event indicator (0=Alive, 1=Dead)

Source

Reconstructed survival data set from the following publication: https://www.thelancet.com/journals/lancet/article/PIIS0140-6736(14)60845-X/fulltext

---

`run_calibration_grid` *Simulate PP values and true trial outcomes for PP-threshold calibration*

Description

For a single fixed data-generating scenario, simulates `n_sims` trials, computes the predictive probability (PP) at the futility look, and records what actually happens to each trial if it is allowed to continue through the remaining fixed decision points in `future_boundaries`. Because the PP value does not depend on the futility threshold κ, the output can be summarised over a whole grid of thresholds afterwards via `summarize_grid_by_kappa` without re-simulating.

Usage

```
run_calibration_grid(
  n_c,
  n_t,
  control_model,
  effect_model,
  recruitment_model,
  data_generating_model,
  futility_IF,
  total_events,
  future_boundaries,
  analysis_model,
  update_priors_sims = 1000,
  PP_sims = 2000,
  n_sims = 1000,
  n_cores = 1,
  seed = NULL
)
```

Arguments

`n_c`, `n_t`	Planned number of patients in the control / treatment group.
`control_model`, `effect_model`	Prior specification used for the interim posterior update (see `update_priors`). `control_model$parameter_mode` must be `"Distribution"`.
`recruitment_model`	Recruitment specification (see `add_recruitment_time`): `method`, `period`, `power`, `rate`, `duration`.

`data_generating_model`
: A named list giving the true scenario used to simulate data: `lambda_c`, `delay_time`, `post_delay_HR`, and optionally `gamma_c` (Weibull control arm if supplied, exponential otherwise).

`futility_IF`
: Information fraction of the PP futility look.

`total_events`
: Maximum planned number of events.

`future_boundaries`
: List of future fixed decision points, each a list with `events` and `crit` (see `PP_func`).

`analysis_model`
: Analysis specification (see `survival_test`).

`update_priors_sims`
: Number of posterior samples per interim dataset.

`PP_sims`
: Number of posterior-predictive simulations per interim dataset.

`n_sims`
: Number of simulated trials.

`n_cores`
: Number of cores (uses `parallel::mclapply` when > 1).

`seed`
: Optional integer seed; replicate `i` uses `seed * 10000 + i`.

Value

A list with elements

`raw`
: A data frame with one row per simulated trial and columns `PP_val`, `t_interim`, `sample_size_interim`, `continuation_success`, `continuation_stop_time`, `continuation_sample_size`, `converged`, `P_Z1`, `P_Z2`, `P_Z3`.

`settings`
: All arguments, plus the package, R, rjags and JAGS versions, the git commit of the working directory, the hostname and a timestamp.

See Also

`summarize_grid_by_kappa`, `select_kappa_star`

`run_paired_scenario` *Run paired replicates for one scenario, with checkpointing*

Description

Runs `single_paired_rep` for `i = 1, ..., n_sims`, in chunks of `chunk_size` replicates (in parallel via `parallel::mclapply` when `n_cores > 1`). After each chunk, all rows so far are written to `checkpoint_file`; if that file exists at the start, the run resumes from the replicates already completed.

Usage

```
run_paired_scenario(
  n_sims,
  seed,
  ...,
  compute_PP = TRUE,
  n_cores = 1,
  checkpoint_file = NULL,
  chunk_size = 100
)
```

Arguments

n_sims	Number of replicates.
seed	Base seed (see single_paired_rep).
...	Further arguments passed to single_paired_rep.
compute_PP	Passed to single_paired_rep: if FALSE, the PP columns are NA and update_priors_sims and PP_sims are not needed. A checkpoint is only resumed with the same compute_PP. Default TRUE.
n_cores	Number of cores (default 1).
checkpoint_file	Optional path of an RDS checkpoint file.
chunk_size	Number of replicates per chunk (default 100).

Value

A list with raw (one row per replicate, ordered by rep_id) and settings (all arguments, the package, R, rjags and JAGS versions, the git commit of the working directory, the hostname, a timestamp and n_failed). The number of failed replicates is reported with a message, and a warning is given if more than 1\

See Also

single_paired_rep, summarize_grid_by_kappa

---

select_kappa_star *Select the PP futility threshold minimising null expected sample size*

Description

Among candidate thresholds for which the one-sided 95\ confidence bound on power (power_LCB from summarize_grid_by_kappa) is at least the floor under every alternative scenario, selects the one with the smallest expected sample size under the null scenario. A threshold with an NA lower bound in any alternative scenario is infeasible. Ties in null expected sample size (within 1e-8) are broken in favour of the *larger* kappa.

Usage

```
select_kappa_star(
  summary_by_scenario,
  power_floor,
  null_scenario,
  alt_scenarios
)
```

Arguments

summary_by_scenario	A named list of data frames, one per scenario, each as returned by summarize_grid_by_kappa over the same kappa_grid.
power_floor	Minimum acceptable lower confidence bound on power: a scalar applied to every alternative scenario, or a named numeric vector with one entry per alternative scenario (e.g. from compute_relative_floors).
null_scenario	Name of the null scenario in summary_by_scenario.
alt_scenarios	Character vector of alternative scenario names.

Value

A list with kappa_star (the selected threshold, or NA with a warning if none is feasible) and feasibility_table (kappa, null_ESS, feasible and, per alternative scenario, power_<s>, power_LCB_<s> and floor_<s>).

See Also

summarize_grid_by_kappa, compute_relative_floors

---

sim_dte *Simulates survival times for a delayed treatment effect (DTE) scenario, where the treatment group experiences a delayed onset of benefit. Control and treatment groups are generated under exponential or Weibull distributions.*

Description

Simulates survival times for a delayed treatment effect (DTE) scenario, where the treatment group experiences a delayed onset of benefit. Control and treatment groups are generated under exponential or Weibull distributions.

Usage

```
sim_dte(
  n_c,
  n_t,
  lambda_c,
  delay_time,
  post_delay_HR,
  dist = "Exponential",
  gamma_c = NULL
)
```

Arguments

n_c	The number of patients in the control group
n_t	The number of patients in the treatment group
lambda_c	The baseline hazard rate for the control group
delay_time	The length of delay before treatment effect begins
post_delay_HR	The hazard ratio after the delay period
dist	The distribution for the control group; must be one of "Exponential" (default) or "Weibull"
gamma_c	The shape parameter for the Weibull distribution (only used if dist = "Weibull")

Value

A data frame with two columns:

time	Simulated survival times
group	Group assignment: "Control" or "Treatment"

Class: data.frame

Examples

```
set.seed(123)
sim_data <- sim_dte(n_c = 10, n_t = 10, lambda_c = 0.1,
                    delay_time = 6, post_delay_HR = 0.6)
head(sim_data)
```

`single_paired_rep` *Simulate one paired replicate for post hoc design evaluation*

Description

Simulates a single trial and records everything needed to evaluate a family of designs afterwards, without baking any design into the run: interim statistics and the predictive probability (PP) at the futility look, and the test statistics at the efficacy look and the final analysis of the *real* trial (which is never stopped here).

Usage

```
single_paired_rep(
  i,
  seed,
  n_c,
  n_t,
  total_events,
  futility_IF,
  efficacy_IF = 0.75,
  design,
  control_model,
  effect_model,
  recruitment_model,
  truth = NULL,
  force_state = NULL,
  analysis_model_LRT,
  mw_t_stars = c(2, 6),
  update_priors_sims,
  PP_sims,
  compute_PP = TRUE
)
```

Arguments

`i`	Replicate index.
`seed`	Base seed. Replicate `i` uses `set.seed(seed * 1e5 + i)`, which must be below `.Machine$integer.max`; the JAGS seed is drawn from that stream, so the replicate is fully reproducible.
`n_c`, `n_t`	Planned number of patients in the control / treatment arm.
`total_events`	Maximum planned number of events.
`futility_IF`	Information fraction of the futility (PP) look.
`efficacy_IF`	Information fraction of the interim efficacy look (default 0.75).
`design`	An `rpact` group-sequential design containing the efficacy look at `efficacy_IF` and the final analysis (and optionally the futility look, with zero alpha spent).

control_model, effect_model
: The *analysis* prior, used for the interim posterior update (see update_priors). control_model$parameter_mode must be "Distribution".

recruitment_model
: Recruitment specification. Must be method = "power" with power = 1 (see PP_func).

truth
: Either a list with the true lambda_c, delay_time, post_delay_HR and optionally gamma_c (exponential control arm if NULL) and state, or NULL, in which case the truth is drawn from control_model and effect_model via simulate_trial_with_recruitment(), optionally with the latent state fixed by force_state.

force_state
: NULL, or 1 (no separation), 2 (immediate separation) or 3 (delayed separation). Only used when truth = NULL.

analysis_model_LRT
: Analysis specification for the log-rank test (method = "LRT", alpha, alternative_hypothesis), used for the PP calculation and the *_LRT statistics.

mw_t_stars
: Values of t^* for the modestly weighted log-rank statistics (default c(2, 6)).

update_priors_sims
: Number of posterior samples per chain (required when compute_PP = TRUE).

PP_sims
: Number of posterior-predictive draws (required when compute_PP = TRUE).

compute_PP
: If FALSE, skip the interim posterior update and the PP calculation: PP_val, converged, rhat_max, n_nonfinite_rhat, P_Z1, P_Z2 and P_Z3 are NA, and all other columns are identical to those from compute_PP = TRUE with the same seed. Such output supports rules D1, D2, D4 and D5 but not D3. Default TRUE.

Details

All designs are then summaries of the returned columns. With critical values c_{eff}, c_{fin} from design and a final-only critical value c_1:

- D1 (fixed design): success if Z_fin_LRT > c_1.
- D2 (efficacy look only): success if Z_eff_LRT > c_eff, otherwise Z_fin_LRT > c_fin. This is continuation_success.
- D3 (D2 plus PP futility): stop for futility if PP_val < kappa; see summarize_grid_by_kappa.
- D4, D5 (D2 plus Z futility): stop for futility if Z_int_LRT (or Z_int_MW_t*) is below a cutoff.

Value

A one-row data frame with columns: rep_id, seed_used, state, true_lambda_c, true_gamma_c (1 for an exponential control arm), true_delay, true_HR; interim t_int, n_int, frac_recruited_int (enrolled / planned), fu_gt3_int, fu_gt6_int (fraction of enrolled treated patients with follow-up t_int - rec_time above 3 and 6), Z_int_LRT, Z_int_MW_t<t>; PP_val, converged, rhat_max, n_nonfinite_rhat, P_Z1, P_Z2, P_Z3; t_eff, n_eff, Z_eff_LRT, Z_eff_MW_t<t>, t_fin, n_fin, Z_fin_LRT, Z_fin_MW_t<t>; the D2 outcome continuation_success, continuation_stop_time, continuation_sample_size; aliases t_interim and sample_size_interim (for summarize_grid_by_kappa); and error (NA, or the error message if the replicate failed, in which case all other outcome columns are NA).

See Also

run_paired_scenario

---

summarize_grid_by_kappa *Summarise calibration output over a grid of PP futility thresholds*

Description

Applies each candidate PP futility threshold κ to raw replicate output (from run_paired_scenario or run_calibration_grid): a trial stops for futility if its interim PP is below κ, otherwise it takes its continuation (D2) outcome. At kappa = 0 no trial stops, so the result is the D2 operating characteristics.

Usage

```
summarize_grid_by_kappa(
  raw,
  kappa_grid,
  lcb_level = 0.95,
  exclude_nonconverged = FALSE
)
```

Arguments

raw	A data frame with columns PP_val, continuation_success, continuation_sample_size, continuation_stop_time, sample_size_interim, t_interim and (if exclude_nonconverged = TRUE) converged.
kappa_grid	Numeric vector of candidate PP thresholds.
lcb_level	Confidence level of the one-sided exact (Clopper-Pearson) lower bound on power (default 0.95).
exclude_nonconverged	If TRUE, drop replicates whose interim MCMC did not converge (converged FALSE or NA). Default FALSE.

Value

A data frame with one row per κ and columns kappa, n (replicates used), n_NA_dropped (replicates dropped because PP_val was NA, with a warning), P_early_fut, power_or_typeI, power_SE, power_LCB (one-sided exact lower bound at level lcb_level, 95\ size), ESS_SE and duration (expected trial duration).

See Also

select_kappa_star, compute_relative_floors

`survival_test` *Calculate statistical significance on a survival dataset*

Description

Performs a survival analysis using the standard log-rank test (LRT), a weighted log-rank test in the Fleming-Harrington family (WLRT), or the modestly-weighted log-rank test of Magirr and Burman (MW), and determines whether the result is statistically significant at the specified alpha level and alternative hypothesis.

Usage

```
survival_test(
  data,
  analysis_method = "LRT",
  alternative = "one.sided",
  alpha,
  rho = 0,
  gamma = 0,
  t_star = NULL,
  s_star = NULL,
  return_HR = TRUE
)
```

Arguments

`data`	A dataframe containing survival data. Must include columns for survival time, event status, and treatment group.
`analysis_method`	Method of analysis: `"LRT"` (default) for standard log-rank test, `"WLRT"` for a Fleming-Harrington-family weighted log-rank test, or `"MW"` for the modestly-weighted log-rank test.
`alternative`	String specifying the alternative hypothesis: `"one.sided"` (default) or `"two.sided"`.
`alpha`	Type I error level for the significance test (required, no default). Pass `NULL` if only `Z` is needed; `Signif` is then `NA`.
`rho`	Rho parameter for the Fleming-Harrington weighted log-rank test.
`gamma`	Gamma parameter for the Fleming-Harrington weighted log-rank test.
`t_star`	Parameter t^* used in the modestly weighted test.
`s_star`	Parameter s^* used in the modestly weighted test.
`return_HR`	If `TRUE` (default), fit a Cox model and return the estimated hazard ratio as `observed_HR`. If `FALSE`, the Cox fit is skipped (it does not affect `Signif` or `Z`) and `observed_HR` is `NA`. The group-sequential and predictive probability code uses `FALSE`.

Value

A list containing:

Signif Logical: whether the test is significant at level `alpha`. `FALSE` if `Z` is `NA`; `NA` if `alpha` is `NULL`.

observed_HR Estimated hazard ratio from a Cox proportional hazards model, or `NA` if `return_HR = FALSE`.

Z Signed test statistic. For all three methods, positive values favour the arm coded as "Treatment" (the second factor level of `group`), i.e. positive Z corresponds to a hazard ratio below 1.

Examples

```
set.seed(123)
df <- data.frame(
  survival_time = rexp(40, rate = 0.1),
  status = rbinom(40, 1, 0.8),
  group = rep(c("Control", "Treatment"), each = 20)
)
result <- survival_test(df, analysis_method = "LRT", alpha = 0.05)
str(result)
```

---

update_priors *Update prior distributions using interim survival data*

Description

This function updates elicited priors (defined through SHELF objects and parametric prior distributions) using interim survival data under a delayed-effect, piecewise-exponential model for the treatment arm and an exponential or Weibull model for the control arm.

Usage

```
update_priors(
  data,
  control_model,
  effect_model,
  n.chains = 2,
  n_burnin = 500,
  n_samples = 1000,
  rhat_threshold = 1.1,
  jags_seed = NULL
)
```

Arguments

data	A data frame containing interim survival data with columns: • survival_time Observed time from randomisation to event/censoring. • status Event indicator (1 = event, 0 = censored). • group Group identifier (e.g., "Control", "Treatment").
control_model	A named list specifying the control arm survival distribution: • dist: Distribution type ("Exponential" or "Weibull") • parameter_mode: Either "Fixed" or "Distribution" • fixed_type: If "Fixed", specify as "Parameters" or "Landmark" • lambda, gamma: Scale and shape parameters • t1, t2: Landmark times • surv_t1, surv_t2: Survival probabilities at landmarks • t1_Beta_a, t1_Beta_b, diff_Beta_a, diff_Beta_b: Beta prior parameters
effect_model	A named list specifying beliefs about the treatment effect: • delay_SHELF, HR_SHELF: SHELF objects encoding beliefs • delay_dist, HR_dist: Distribution types ("hist" by default) • P_S: Probability that survival curves separate • P_DTE: Probability of delayed separation, conditional on separation
n.chains	Number of MCMC chains to run (default is 2)
n_burnin	Number of burn-in samples for the MCMC chain(s) (default is 500)
n_samples	Number of posterior samples retained **per chain** after burn-in (default is 1000). The returned data frame has n.chains * n_samples rows (2,000 with the defaults).
rhat_threshold	Convergence threshold on the Gelman-Rubin potential scale reduction factor (default 1.1). Used only to compute the "converged" attribute on the return value; does not affect sampling.
jags_seed	Optional integer seed for JAGS. If supplied, chain ch is initialised with the "base::Wichmann-Hill" RNG and seed jags_seed + ch, so repeated calls with the same jags_seed return identical draws. If NULL (default), JAGS seeds its chains itself and the draws are not reproducible with set.seed().

Value

A data frame containing Monte Carlo samples from the updated (posterior) distribution of the model parameters, with columns lambda_c, delay_time, HR, gamma_c (Weibull only), and Z (the latent scenario indicator: 1 = no separation, 2 = immediate separation, 3 = delayed separation). The following attributes are attached for diagnostic purposes:

rhat A named numeric vector of per-parameter Gelman-Rubin point estimates (accessed via attr(posterior_df, "rhat")).

converged TRUE if every *finite* R-hat is below rhat_threshold, FALSE if not, or NA if there are no finite R-hat values (or the diagnostic could not be computed). Non-finite values (NaN arises for parameters that are constant across draws, e.g. delay_time when Z != 3 throughout) are not treated as failures; they are counted in n_nonfinite_rhat.

`rhat_max` The largest finite R-hat (`NA` if none).

`n_nonfinite_rhat` The number of non-finite R-hat values.

`n_retained_total` The number of retained draws over all chains, `n.chains * n_samples`.

`Z_probs` A named numeric vector `c(P_Z1=, P_Z2=, P_Z3=)`, the posterior probability of each latent state, i.e. `table(posterior_df$Z) / nrow(posterior_df)` (accessed via `attr(posterior_df, "Z_probs")`).

Priors for `lambda_c`, `T`, and `HR` are constructed from elicited distributions using the SHELF framework, then updated through sampling-based posterior inference.

Deprecated

The exponential control arm (`control_model$dist = "Exponential"`) is deprecated and will be removed in a future release; use `"Weibull"`.

Examples

```
set.seed(123)
interim_data = data.frame(survival_time = runif(10, min = 0, max = 10),
status = rbinom(10, size = 1, prob = 0.5),
group = c(rep("Control", 5), rep("Treatment", 5)))
control_model = list(dist = "Exponential",
                     parameter_mode = "Distribution",
                     t1 = 12,
                     t1_Beta_a = 20,
                     t1_Beta_b = 32)

effect_model = list(delay_SHELF = SHELF::fitdist(c(5.5, 6, 6.5),
                    probs = c(0.25, 0.5, 0.75), lower = 0, upper = 12),
                    delay_dist = "gamma",
                    HR_SHELF = SHELF::fitdist(c(0.5, 0.6, 0.7),
                    probs = c(0.25, 0.5, 0.75), lower = 0, upper = 1),
                    HR_dist = "gamma",
                    P_S = 1,
                    P_DTE = 0)

posterior_df <- update_priors(
  data = interim_data,
  control_model = control_model,
  effect_model = effect_model,
  n_samples = 10)

# Diagnostics, e.g.:
attr(posterior_df, "rhat")
attr(posterior_df, "converged")
attr(posterior_df, "Z_probs")
```

ZODIAC *ZODIAC data set*

Description

A reconstructed survival data set for the ZODIAC clinical trial

Usage

```
ZODIAC
```

Format

A data frame with 697 rows and 2 variables:

Survival time Survival Time (in months)

Status Event indicator (0=Alive, 1=Dead)

Source

Reconstructed survival data set from the following publication: https://www.sciencedirect.com/science/article/abs/pii/S14702

Index

∗ **datasets**
    INTEREST, 20
    MCMC_sample, 22
    REVEL, 25
    ZODIAC, 38

add_recruitment_time, 2, *12, 13*, *16*, *26*
apply_design_rule, 3, *4*
apply_GSD_to_trial, *3*, 4, *21*
assurance_adaptive_shiny_app, 6
assurance_shiny_app, 6

calc_dte_assurance, 7
calc_dte_assurance_adaptive, *4, 5*, 9, *11*, *21*
calibrate_matched_futility_boundary, *10*, 11
calibrate_PP_threshold, 13
calibrate_PP_timing, 16
cens_data, *4*, 18
compute_relative_floors, 19, *29*, *33*

INTEREST, 20

make_gsd_design, *5*, 21
make_rpact_design_from_GSD_model *(*make_gsd_design*)*, 21
MCMC_sample, 22

PP_func, *10*, *14*, *17*, 22, *27*, *32*

REVEL, 25
run_calibration_grid, 26, *33*
run_paired_scenario, *3, 4*, *15*, 27, *33*

select_kappa_star, *15*, *19, 20*, *27*, 28, *33*
sim_dte, 29
single_matched_futility_rep, *12*
single_paired_rep, *21*, *27, 28*, 31
summarize_grid_by_kappa, *4*, *15*, *26–29*, *32*, *33*
survival_test, *12*, *27*, 34

update_priors, *10, 11*, *13, 14*, *16*, *26*, *32*, 35

ZODIAC, 38