Package 'adaptr'

June 17, 2022

Title Adaptive Trial Simulator

Version 1.1.0

Date 2022-06-16

Description Package that simulates adaptive clinical trials using adaptive stopping, adaptive arm dropping, and/or adaptive randomisation. Developed as part of the INCEPT (Intensive Care Platform Trial) project (<https://incept.dk/>), which is primarily supported by a grant from Sygeforsikringen ``danmark'' (<https://www.sygeforsikring.dk/>).

License GPL (>= 3)

Imports stats, parallel, utils

Encoding UTF-8

NeedsCompilation no

URL https://incept.dk/, https://github.com/INCEPTdk/adaptr/, https://inceptdk.github.io/adaptr/

BugReports https://github.com/INCEPTdk/adaptr/issues/

RoxygenNote 7.2.0

Suggests covr, ggplot2, rmarkdown, knitr, testthat, vdiffr

VignetteBuilder knitr

Config/testthat/edition 3

Author Anders Granholm [aut, cre] (<https://orcid.org/0000-0001-5799-7655>), Benjamin Skov Kaas-Hansen [aut] (<https://orcid.org/0000-0003-1023-0371>), Aksel Karl Georg Jensen [ctb] (<https://orcid.org/0000-0002-1459-0465>), Theis Lange [ctb] (<https://orcid.org/0000-0001-6807-8347>)

Maintainer Anders Granholm <andersgran@gmail.com>

Repository CRAN

Date/Publication 2022-06-17 08:20:18 UTC

R topics documented:

`adaptr-package` *adaptr: Adaptive Trial Simulator*

Description

The `adaptr` package simulates adaptive (multi-arm) trials using adaptive stopping, adaptive arm dropping and/or response-adaptive randomisation. The package is developed as part of the INCEPT (Intensive Care Platform Trial) project, funded primarily by a grant from Sygeforsikringen "danmark".

Details

The `adaptr` package contains the following primary functions:

1. `setup_trial()` is the general function that sets up a trial specification. The simpler, special-case functions `setup_trial_binom()` and `setup_trial_norm()` may be used for easier specification of trial designs using binary, binomially distributed or continuous, normally distributed outcomes, respectively, with some limitations in flexibility.
2. The `run_trial()` and `run_trials()` functions are used to conduct single or multiple simulations, respectively, according to a trial specification setup as described in #1.
3. The `extract_results()` and `summary()` functions are used to extract or summarise the results of multiple trial simulations.
4. The `plot_status()` and `plot_history()` functions are used to plot the overall trial/arm statuses for multiple simulated trials or the history of trial metrics over time for single/multiple simulated trials, respectively.

For further information see the function documentation or the **Overview** vignette (`vignette("Overview", package = "adaptr")`) for an example of how the functions work in combination. For further examples and guidance on setting up trial specifications, see setup_trial documentation, the **Basic examples** vignette (`vignette("Basic-examples", package = "adaptr")`) and the **Advanced example** vignette (`vignette("Advanced-example", package = "adaptr")`).

If using the package, please consider citing it using `citation(package = "adaptr")`.

References

Granholm A, Jensen AKG, Lange T, Kaas-Hansen BS (2022). adaptr: an R package for simulating and comparing adaptive clinical trials. Journal of Open Source Software, 7(72), 4284. doi: 10.21105/joss.04284

Website/manual

GitHub repository

See Also

`setup_trial()`, `setup_trial_binom()`, `setup_trial_norm()`, `run_trial()`, `run_trials()`, `extract_results()`, `summary()`, `print()`, `plot_status()` and `plot_history()`.

`extract_results` *Extract simulation results*

Description

This function extracts relevant information from multiple simulations of the same trial specification in a tidy `data.frame` (1 simulation per row). See also the `summary()` function.

Usage

```
extract_results(
  object,
  select_strategy = "control if available",
  select_last_arm = FALSE,
  select_preferences = NULL,
  te_comp = NULL,
  raw_ests = FALSE
)
```

Arguments

`object` `trial_results` object, output from the `run_trials()` function.

`select_strategy`

single character string. For trials not stopped due to superiority (or with only 1 arm remaining, if `select_last_arm` is set to `TRUE` in trial designs with a common `control` arm; see below), this parameter specifies which arm will be considered selected when calculating trial design performance metrics (described below; this corresponds to the consequence of an inconclusive trial, i.e., which arm would then be used in practice).
The following options are available and must be written exactly as below (case sensitive, cannot be abbreviated):

- `"control if available"` (default): selects the **first** `control` arm for trials with a common control arm **if** this arm is active at end-of-trial, otherwise no arm will be selected. For trial designs without a common `control`, no arm will be selected.

- "none": selects no arm in trials not ending with superiority.
- "control": similar to "control if available", but will throw an error for trial designs without a common control arm.
- "final control": selects the **final** control arm regardless of whether the trial was stopped for practical equivalence, futility, or at the maximum sample size; this strategy can only be specified for trial designs with a common control arm.
- "control or best": selects the **first** control arm if still active at end-of-trial, otherwise selects the best remaining arm (defined as the remaining arm with the highest probability of being the best in the final analysis). Only works for trial designs with a common control arm.
- "best": selects the best remaining arm (as described under "control or best").
- "list or best": selects the first remaining arm from a specified list (specified using select_preferences, technically a character vector). If none of these arms are are active at end-of-trial, the best remaining arm will be selected (as described above).
- "list": as specified above, but if no arms on the provided list remain active at end-of-trial, no arm is selected.

select_last_arm

single logical, defaults to FALSE. If TRUE, the only remaining active arm (the last control) will be selected in trials with a common control arm ending with equivalence or futility, before considering the options specified in select_strategy. Must be FALSE for trial designs without a common control arm.

select_preferences

character vector specifying a number of arms used for selection if one of the "list or best" or "list" options are specified for select_strategy. Can only contain valid arms available in the trial.

te_comp

character string, treatment-effect comparator. Can be either NULL (the default) in which case the **first** control arm is used for trial designs with a common control arm, or a single trial arm. Will be used when calculating sq_err_te (the squared error of the treatment effect comparing the selected arm to the comparator arm, as described below).

raw_ests

single logical. If FALSE (default), the posterior estimates (post_ests, see setup_trial() and run_trial()) will be used to calculate sq_err (the squared error of the estimated compared to the specified effect in the selected arm) and sq_err_te (the squared error of the treatment effect comparing the selected arm to the comparator arm, as described for te_comp and below). If TRUE, the raw estimates (raw_ests, see setup_trial() and run_trial()) will be used instead of the posterior estimates.

Value

A data.frame containing the following columns:

- sim: the simulation number (from 1 to the number of simulations).

- final_n: the final sample size in each simulation.
- sum_ys: the sum of the total counts in all arms, e.g., the total number of events in trials with a binary outcome (setup_trial_binom()) or the sum of the arm totals in trials with a continuous outcome (setup_trial_norm()).
- ratio_ys: calculated as sum_ys/final_n.
- final_status: the final trial status for each simulation, either "superiority", "equivalence", "futility", or "max", as described in run_trial().
- superior_arm: the final superior arm in simulations stopped for superiority, will be NA in simulations not stopped for superiority.
- selected_arm: the final selected arm (as described above), will correspond to the superior_arm in simulations stopped for superiority and be NA if no arm is selected. See select_strategy above.
- sq_err: the squared error of the estimate in the selected arm, calculated as (estimated effect - true effect)^2 for the selected arms.
- sq_err_te: the squared error of the treatment effect comparing the selected arm to the comparator arm (as specified in te_comp). Calculated as:
((estimated effect in the selected arm - estimated effect in the comparator arm) - (true effect in the selected arm - true effect in the comparator arm))^2
Will be NA for simulations without a selected arm or with no comparator specified (see te_comp above).

Examples

```
# Setup a trial specification
binom_trial <- setup_trial_binom(arms = c("A", "B", "C", "D"),
                                 control = "A",
                                 true_ys = c(0.20, 0.18, 0.22, 0.24),
                                 data_looks = 1:20 * 100)

# Run 10 simulations with a specified random base seed
res <- run_trials(binom_trial, n_rep = 10, base_seed = 12345)

# Extract results and Select the control arm if available
# in simulations not ending with superiority
extract_results(res, select_strategy = "control")
```

---

find_beta_params *Find beta distribution parameters from thresholds*

Description

Helper function to find a beta distribution with parameters corresponding to the fewest possible patients with events/non-events and a specified event proportion. Used in the **Advanced example** vignette (vignette("Advanced-example", "adaptr")) to derive beta prior distributions for use in *beta-binomial conjugate models*, based on a belief that the true event probability lies within a specified percentile-based interval (defaults to 95%). May similarly be used by users to derive other beta priors.

Usage

```
find_beta_params(
  theta = NULL,
  boundary_target = NULL,
  boundary = "lower",
  interval_width = 0.95,
  n_dec = 0,
  max_n = 10000
)
```

Arguments

theta	single numeric > 0 and < 1, expected true event probability.
boundary_target	single numeric > 0 and < 1, target lower or upper boundary of the interval.
boundary	single character string, either "lower" (default) or "upper", used to select which boundary to use when finding appropriate parameters for the beta distribution.
interval_width	width of the credible interval whose lower/upper boundary should be used (see boundary_target); must be > 0 and < 1; defaults to 0.95.
n_dec	single non-negative integer; the returned parameters are rounded to this number of decimals. Defaults to 0, in which case the parameters will correspond to whole number of patients.
max_n	single integer > 0 (default 10000), the maximum total sum of the parameters, corresponding to the maximum total number of patients that will be considered by the function when finding the optimal parameter values. Corresponds to the maximum number of patients contributing information to a beta prior; more than the default number of patients are unlikely to be used in a beta prior.

Value

A single-row data.frame with five columns: the two shape parameters of the beta distribution (alpha, beta), rounded according to n_dec, and the actual lower and upper boundaries of the interval and the median (with appropriate names, e.g. p2.5, p50, and p97.5 for a 95% interval), when using those rounded values.

---

plot_history *Plot trial metric history*

Description

Plots the history of relevant metrics over the progress of single or multiple simulations. Simulated trials **only** contribute until the time they are stopped, i.e., if some trials are stopped earlier than others, they will not contribute to the summary statistics at later adaptive looks. Data from individual arms in a trial contribute until the complete trial is stopped.
These history plots require non-sparse results (sparse set to FALSE; see run_trial() and run_trials()) and the ggplot2 package installed.

Usage

```
plot_history(object, x_value = "look", y_value = "prob", line = NULL, ...)

## S3 method for class 'trial_result'
plot_history(object, x_value = "look", y_value = "prob", line = NULL, ...)

## S3 method for class 'trial_results'
plot_history(
  object,
  x_value = "look",
  y_value = "prob",
  line = NULL,
  ribbon = list(width = 0.5, alpha = 0.2),
  ...
)
```

Arguments

object	trial_results object, output from the run_trials() function.
x_value	single character string, determining whether the number of adaptive analysis looks ("look", default) or the total cumulated number of patients allocated ("total n") are plotted on the x-axis.
y_value	single character string, determining which values are plotted on the y-axis. The following options are available: allocation probabilities ("prob", default), the total number of patients allocated to each arm ("n"), the percentage of patients allocated to each arm of the total number of patients randomised ("pct"), the sum of all outcomes in each arm ("sum ys"), the ratio of outcomes ("ratio ys", the sum of outcomes in each arm divided by the number of patients allocated to that arm).
line	list styling the lines as per **ggplot2** conventions (e.g., linetype, size).
...	additional arguments, not used.
ribbon	list, as line but only appropriate for trial_results objects (i.e., when multiple simulations are run). Also allows to specify the width of the interval: must be between 0 and 1, with 0.5 (default) showing the inter-quartile ranges.

Value

A ggplot2 plot object.

See Also

plot_status().

Examples

```
#### Only run examples if ggplot2 is installed ####
if (requireNamespace("ggplot2", quietly = TRUE)){
```

```
# Setup a trial specification
binom_trial <- setup_trial_binom(arms = c("A", "B", "C", "D"),
                                 control = "A",
                                 true_ys = c(0.20, 0.18, 0.22, 0.24),
                                 data_looks = 1:20 * 100)



# Run a single simulation with a fixed random seed
res <- run_trial(binom_trial, seed = 12345)

# Plot total allocations to each arm according to overall total allocations
plot_history(res, x_value = "total n", y_value = "n")



# Run multiple simulation with a fixed random base seed
# Notice that sparse = FALSE is required
res_mult <- run_trials(binom_trial, n_rep = 15, base_seed = 12345, sparse = FALSE)

# Plot allocation probabilities at each look
plot_history(res_mult, x_value = "look", y_value = "prob")

# Other y_value options are available but not shown in these examples

# Do not return/print last plot in documentation
invisible(NULL)
}
```

`plot_status` *Plot statuses*

Description

Plots the statuses over time of multiple simulated trials (overall or for a specific arm). Requires the `ggplot2` package installed.

Usage

```
plot_status(object, x_value = "look", arm = NULL, area = list(alpha = 0.5))

## S3 method for class 'trial_results'
plot_status(object, x_value = "look", arm = NULL, area = list(alpha = 0.5))
```

Arguments

`object`	`trial_results` object, output from the `run_trials()` function.

x_value	single character string, determining whether the number of adaptive analysis looks ("look", default) or the total cumulated number of patients allocated ("total n") are plotted on the x-axis.
arm	single character string or NULL (default); can be set to a valid trial arm. If NULL, the overall trial statuses are plotted, otherwise the statuses for a single, specific trial arm are plotted.
area	list of styling settings for the area as per **ggplot2** conventions (e.g., alpha, size). The default (list(alpha = 0.5)) sets the transparency to 50% so overlain shaded areas are visible.

Value

A ggplot2 plot object.

See Also

plot_history().

Examples

```
#### Only run examples if ggplot2 is installed ####
if (requireNamespace("ggplot2", quietly = TRUE)){

  # Setup a trial specification
  binom_trial <- setup_trial_binom(arms = c("A", "B", "C", "D"),
                                   control = "A",
                                   true_ys = c(0.20, 0.18, 0.22, 0.24),
                                   data_looks = 1:20 * 100)

  # Run multiple simulation with a fixed random base seed
  res_mult <- run_trials(binom_trial, n_rep = 25, base_seed = 12345)

  # Plot trial statuses at each look according to total allocations
  plot_status(res_mult, x_value = "total n")

  # Do not return/print last plot in documentation
  invisible(NULL)
}
```

---

print *Print methods for adaptive trial objects*

Description

Prints contents of the first input x in a human-friendly way, see **Details** for more information.

Usage

```
## S3 method for class 'trial_spec'
print(x, prob_digits = 3, ...)

## S3 method for class 'trial_result'
print(x, prob_digits = 3, ...)

## S3 method for class 'trial_results'
print(
  x,
  select_strategy = "control if available",
  select_last_arm = FALSE,
  select_preferences = NULL,
  te_comp = NULL,
  raw_ests = FALSE,
  restrict = NULL,
  digits = 1,
  ...
)

## S3 method for class 'trial_results_summary'
print(x, digits = 1, ...)
```

Arguments

x	object to print, see Details below.
prob_digits	single integer, the number of digits used when printing probabilities, allocation probabilities and softening powers.
...	additional arguments, not used.
select_strategy	single character string. For trials not stopped due to superiority (or with only 1 arm remaining, if `select_last_arm` is set to `TRUE` in trial designs with a common `control` arm; see below), this parameter specifies which arm will be considered selected when calculating trial design performance metrics (described below; this corresponds to the consequence of an inconclusive trial, i.e., which arm would then be used in practice). The following options are available and must be written exactly as below (case sensitive, cannot be abbreviated): • `"control if available"` (default): selects the **first** `control` arm for trials with a common control arm **if** this arm is active at end-of-trial, otherwise no arm will be selected. For trial designs without a common `control`, no arm will be selected. • `"none"`: selects no arm in trials not ending with superiority. • `"control"`: similar to `"control if available"`, but will throw an error for trial designs without a common `control` arm. • `"final control"`: selects the **final** `control` arm regardless of whether the trial was stopped for practical equivalence, futility, or at the maximum sam-

ple size; this strategy can only be specified for trial designs with a common `control` arm.

- `"control or best"`: selects the **first** `control` arm if still active at end-of-trial, otherwise selects the best remaining arm (defined as the remaining arm with the highest probability of being the best in the final analysis). Only works for trial designs with a common `control` arm.
- `"best"`: selects the best remaining arm (as described under `"control or best"`).
- `"list or best"`: selects the first remaining arm from a specified list (specified using `select_preferences`, technically a character vector). If none of these arms are are active at end-of-trial, the best remaining arm will be selected (as described above).
- `"list"`: as specified above, but if no arms on the provided list remain active at end-of-trial, no arm is selected.

`select_last_arm`
: single logical, defaults to `FALSE`. If `TRUE`, the only remaining active arm (the last `control`) will be selected in trials with a common `control` arm ending with `equivalence` or `futility`, before considering the options specified in `select_strategy`. Must be `FALSE` for trial designs without a common `control` arm.

`select_preferences`
: character vector specifying a number of arms used for selection if one of the `"list or best"` or `"list"` options are specified for `select_strategy`. Can only contain valid `arms` available in the trial.

`te_comp`
: character string, treatment-effect comparator. Can be either `NULL` (the default) in which case the **first** `control` arm is used for trial designs with a common control arm, or a single trial `arm`. Will be used when calculating `sq_err_te` (the squared error of the treatment effect comparing the selected arm to the comparator arm, as described below).

`raw_ests`
: single logical. If `FALSE` (default), the posterior estimates (`post_ests`, see `setup_trial()` and `run_trial()`) will be used to calculate `sq_err` (the squared error of the estimated compared to the specified effect in the selected arm) and `sq_err_te` (the squared error of the treatment effect comparing the selected arm to the comparator arm, as described for `te_comp` and below). If `TRUE`, the raw estimates (`raw_ests`, see `setup_trial()` and `run_trial()`) will be used instead of the posterior estimates.

`restrict`
: single character string or `NULL`. If `NULL` (default), results are summarised for all simulations; if `"superior`, results are summarised for simulations ending with superiority only; if `"selected"`, results are summarised for simulations ending with a selected arm (according to the specified arm selection strategy for simulations not ending with superiority). Some summary measures (e.g., `prob_conclusive`) can only be calculated across all simulations and several are calculated regardless of `restrict` settings, but have substantially different interpretations if restricted.

`digits`
: single integer, number of digits to print for probabilities and some other summary values (with 2 extra digits added for outcome rates).

Details

The behaviour depends on the class of x:

- trial_spec: prints a trial specification setup by setup_trial(), setup_trial_binom() or setup_trial_norm().
- trial_result: prints the results of a single trial simulated by run_trial(). More details are saved in the trial_result object and thus printed if the sparse argument in run_trial() or run_trials() is set to FALSE; if TRUE, fewer details are printed, but the omitted details are available by printing the trial_spec object created by setup_trial(), setup_trial_binom() or setup_trial_norm().
- trial_results: prints the results of multiple simulations generated using run_trials(). Further documentation on how multiple trials are summarised before printing can be found in the summary() function documentation.
- trial_results_summary: print method for summary of multiple simulations of the same trial specification, generated by using the summary() function on an object generated by run_trials().

Value

Invisibly returns x.

Methods (by class)

- trial_spec: Trial specification
- trial_result: Single trial result
- trial_results: Multiple trial results
- trial_results_summary: Summary of multiple trial results

---

run_trial *Simulate a single trial*

Description

This function conducts a single trial simulation using a trial specification as specified by setup_trial(), setup_trial_binom() or setup_trial_norm(). During simulation, the function randomises "patients", randomly generates outcomes, calculates the probabilities that each arm is the best (and better than the control, if any). This is followed by checking inferiority, superiority, equivalence and/or futility as desired; dropping arms, and re-adjusting allocation probabilities according to the criteria specified in the trial specification. If there is no common control arm, the trial simulation will be stopped at the maximum sample size, when 1 arm is superior to the others, or when all arms are considered equivalent (if equivalence testing is specified).
If a common control arm is specified, all other arms will be compared to that, and if 1 comparison crosses the superiority threshold, that arm will become the new control and the old control will be

considered inferior. If multiple non-control arms cross the superiority threshold in the same analysis, the one with the highest probability of being the overall best will become the new control. Equivalence/futility will also be checked in trial designs with common controls if specified, and equivalent or futile arms will be dropped. The trial simulation will be stopped when only 1 arm is left, when the final arms are all equivalent, or when the maximum sample size has been reached.

Usage

```
run_trial(trial_spec, seed = NULL, sparse = FALSE)
```

Arguments

`trial_spec`	`trial_spec` object, generated and validated by the `setup_trial()`, `setup_trial_binom()` or `setup_trial_norm()` function.
`seed`	single integer or `NULL` (default), if a value is provided, this value will be used as the random seed when running (the global random seed will be restored after the function has run, so it is not affected).
`sparse`	single logical; if `FALSE` (default) everything listed below is included in the returned object. If `TRUE`, only a limited amount of data is included in the returned object. This can be practical when running many simulations and saving the results using the `run_trials()` function (which relies on this function), as the output file will thus be substantially smaller. However, printing of individual trial results will be substantially less detailed for sparse results and non-sparse results are required by `plot_history()`.

Value

A `trial_result` object containing everything listed below if `sparse` (as described above) is `FALSE`. Otherwise only `final_status`, `final_n`, `trial_res`, `seed` and `sparse` are included.

- `final_status`: either `"superiority"`, `"equivalence"`, `"futility"`, or `"max"`.
- `final_n`: the total number of patients randomised.
- `max_n`: the pre-specified maximum sample size.
- `looks`: numeric vector, the total number of patients at each conducted adaptive analysis.
- `planned_looks`: numeric vector, the cumulated number of patients planned to be randomised at each adaptive analysis, even those not conducted if the simulation is stopped before the maximum sample size.
- `start_control`: character, initial common control arm (if specified).
- `final_control`: character, final common control arm (if relevant).
- `control_prob_fixed`: fixed common control arm probabilities (if specified; see `setup_trial()`).
- `inferiority`, `superiority`, `equivalence_prob`, `equivalence_diff`, `equivalence_only_first`, `futility_prob`, `futility_diff`, `futility_only_first`, `highest_is_best`, and `soften_power`: as specified in `setup_trial()`.
- `best_arm`: the best arm(s), as described in `setup_trial()`.

- trial_res: a data.frame containing most of the information specified for each arm in setup_trial() including true_ys (true outcomes as specified in setup_trial()) and for each arm the sum of the outcomes (sum_ys; i.e., the total number of events for binary outcomes or the totals of continuous outcomes) and patients randomised (ns), summary statistics for the raw outcome data (raw_ests, calculated as specified in setup_trial(), defaults to mean values, i.e., event rates for binary outcomes or means for continuous outcomes) and posterior estimates (post_ests, post_errs, lo_cri, and hi_cri, as specified in setup_trial()), final_status of each arm ("inferior", "superior", "equivalence", "futile", "active", or "control" (currently active control arm, including if the current control when stopped for equivalence)), status_look (specifying the cumulated number of patients randomised when an adaptive analysis changed the final_status to "superior", "inferior", "equivalence", or "futile"), status_probs, the probability that each treatment was the best/better than the common control arm (if any)/equivalent to the common control arm (if any and stopped for equivalence; NA if the control arm was stopped due to the last remaining other arm(s) being stopped for equivalence)/futile if stopped for futility at the last analysis it was included in, final_alloc, the final allocation probability for each arm the last time patients were randomised to it, including for arms stopped at the maximum sample size, and probs_best_last, the probabilities of each remaining arm being the overall best in the last conducted analysis (NA for previously dropped arms).
- all_looks: a list of lists containing one list per conducted trial look (adaptive analysis). These lists contain the variables arms, old_status (status before the analysis of the current round was conducted), new_status (as specified above, status after current analysis has been conducted), sum_ys (as described above), ns (as described above), old_alloc (the allocation probability used during this look), probs_best (the probabilities of each arm being the best in the current adaptive analysis), new_alloc (the allocation probabilities after updating these in the current adaptive analysis; NA for all arms when the trial is stopped and no further analyses will be conducted), probs_better_first (if a common control is provided, specifying the probabilities that each arm was better than the control in the first analysis conducted during that look), probs_better (as probs_better_first, but updated if another arm becomes the new control), probs_equivalence_first and probs_equivalence (as for probs_better/probs_better_first, but for equivalence if equivalence is assessed). The last variables are NA if the arm was not active in the applicable adaptive analysis or if they would not be included during the next adaptive analysis.
- allocs: a character vector containing the allocations of all patients in the order of randomization.
- ys: a numeric vector containing the outcomes of all patients in the order of randomization (0 or 1 for binary outcomes).
- seed: the random seed used, if specified.
- description, add_info, cri_width, n_draws, robust: as specified in setup_trial(), setup_trial_binom() or setup_trial_norm().
- sparse: single logical, corresponding to the sparse input.

Examples

```
# Setup a trial specification
binom_trial <- setup_trial_binom(arms = c("A", "B", "C", "D"),
                                 true_ys = c(0.20, 0.18, 0.22, 0.24),
```

```
                                    data_looks = 1:20 * 100)

# Run trial with a specified random seed
res <- run_trial(binom_trial, seed = 12345)

# Print results with 3 decimals
print(res, digits = 3)
```

`run_trials` *Simulate multiple trials*

Description

This function conducts multiple simulations using a trial specification as specified by `setup_trial()`, `setup_trial_binom()` or `setup_trial_norm()`. This function essentially manages random seeds and runs multiple simulation using `run_trial()` - additional details on individual simulations are provided in that function's description. This function allows simulating trials in parallel using multiple cores, automatically saving and re-loading saved objects, and "growing" already saved simulation files (i.e., appending additional simulations to the same file).

Usage

```
run_trials(
  trial_spec,
  n_rep,
  path = NULL,
  overwrite = FALSE,
  grow = FALSE,
  cores = 1,
  base_seed = NULL,
  sparse = TRUE,
  progress = NULL,
  version = NULL,
  compress = TRUE,
  export = NULL,
  export_envir = parent.frame()
)
```

Arguments

`trial_spec`	`trial_spec` object, generated and validated by the `setup_trial()`, `setup_trial_binom()` or `setup_trial_norm()` function.
`n_rep`	single integer; the number of simulations to run.
`path`	single character; if specified (defaults to `NULL`), files will be written to and loaded from this path using the `saveRDS()` / `readRDS()` functions.

overwrite	single logical; defaults to FALSE, in which case previous simulations saved in the same path will be re-loaded (if the same trial specification was used). If TRUE, the previous file is overwritten. If grow is TRUE, this argument must be set to FALSE.
grow	single logical; defaults to FALSE. If TRUE and a valid path to a valid previous file containing less simulations than n_rep, the additional number of simulations will be run (appropriately re-using the same base_seed, if specified) and appended to the same file.
cores	single integer; the number of cores to run the simulations on using the **parallel** library. Defaults to 1; may be increased to run multiple simulations in parallel. parallel::detectCores() may be used to find the number of available cores.
base_seed	single integer; a random seed used as the basis for simulations; each simulation will set the random seed to a value based on this (+ the trial number), without affecting the global random seed after the function has been run.
sparse	single logical, as described in run_trial(); defaults to TRUE when running multiple simulations, in which case only the data necessary to summarise all simulations are saved for each simulation. If FALSE, more detailed data for each simulation is saved, allowing more detailed printing of individual trial results and plotting using plot_history() (plot_status() does not require non-sparse results).
progress	single numeric > 0 and <= 1 or NULL. If NULL (default), no progress is printed to the console. Otherwise, progress messages are printed to the control at intervals proportional to the value specified by progress. **Note:** as printing is not possible from within clusters on multiple cores, the function conducts batches of simulations on multiple cores (if specified), with intermittent printing of statuses. Thus, all cores have to finish running their current assigned batches before the other cores may proceed with the next batch. If there is substantial differences in the simulation speeds across cores, using progress may thus increase total simulation times.
version	passed to saveRDS() when saving simulations, defaults to NULL (as in saveRDS()), which means that the current default version is used. Ignored if simulations are not saved.
compress	passed to saveRDS() when saving simulations, defaults to TRUE (as in saveRDS()), see saveRDS() for other options. Ignored if simulations are not saved.
export	character vector of names of objects to export to each parallel core if cores > 1; passed as the varlist argument to parallel::clusterExport(). Defaults to NULL (no objects exported), ignored if cores == 1. See **Details** below.
export_envir	environment where to look for the objects defined in export if cores > 1 and export is not NULL. Defaults to the environment from where run_trials() is called.

Details

Exporting objects when using multiple cores

If setup_trial() is used to define a trial specification with custom functions (in the fun_y_gen, fun_draws, and fun_raw_est arguments of setup_trial()) and run_trials() is run with cores

> 1, it is necessary to export additional functions or objects used by these functions and defined by the user outside the function definitions provided. Similarly, functions from external packages loaded using `library()` or `require()` must be exported or called prefixed with the namespace, i.e., `package::function`. The `export` and `export_envir` arguments are used to export objects calling the `parallel::clusterExport()`-function.

Value

A list of a special class `"trial_results"`, which contains the `trial_results` (results from all simulations), `trial_spec` (the trial specification), `n_rep`, `base_seed`, `elapsed_time` (the total simulation run time) and `sparse` (as described above). These results may be extracted using the `extract_results()` function and summarised using the `summary()` or print (`print.trial_results()`) functions; see function documentation for details on additional arguments used to select arms in simulations not ending in superiority and other summary choices.

Examples

```
# Setup a trial specification
binom_trial <- setup_trial_binom(arms = c("A", "B", "C", "D"),
                                 true_ys = c(0.20, 0.18, 0.22, 0.24),
                                 data_looks = 1:20 * 100)

# Run 10 simulations with a specified random base seed
res <- run_trials(binom_trial, n_rep = 10, base_seed = 12345)

# See ?summary and ?print for details on summarising and printing
```

`setup_trial` *Setup a generic trial specification*

Description

Specifies the design of an adaptive trial with any type of outcome and validates all inputs. Use `run_trial()` or `run_trials()` to conduct single/multiple simulations of the specified trial, respectively.
See `setup_trial_binom()` and `setup_trial_norm()` for simplified setup of trial designs common outcome types. For additional trial specification examples, see the the **Basic examples** vignette (`vignette("Basic-examples", package = "adaptr")`) and the **Advanced example** vignette (`vignette("Advanced-example", package = "adaptr")`).

Usage

```
setup_trial(
  arms,
  true_ys,
  fun_y_gen = NULL,
  fun_draws = NULL,
```

```
  start_probs = NULL,
  fixed_probs = NULL,
  min_probs = rep(NA, length(arms)),
  max_probs = rep(NA, length(arms)),
  data_looks = NULL,
  max_n = NULL,
  look_after_every = NULL,
  control = NULL,
  control_prob_fixed = NULL,
  inferiority = 0.01,
  superiority = 0.99,
  equivalence_prob = NULL,
  equivalence_diff = NULL,
  equivalence_only_first = NULL,
  futility_prob = NULL,
  futility_diff = NULL,
  futility_only_first = NULL,
  highest_is_best = FALSE,
  soften_power = 1,
  fun_raw_est = mean,
  cri_width = 0.95,
  n_draws = 5000,
  robust = TRUE,
  description = NULL,
  add_info = NULL
)
```

Arguments

arms	character vector with unique names for the trial arms.
true_ys	numeric vector specifying true outcomes (e.g., event probabilities, mean values, etc.) for all trial arms.
fun_y_gen	function, generates outcomes. See setup_trial() **Details** for information on how to specify this function. **Note:** this function is called once during setup to validate the output structure.
fun_draws	function, generates posterior draws. See setup_trial() **Details** for information on how to specify this function. **Note:** this function is called up to three times during setup to validate the output structure.
start_probs	numeric vector, allocation probabilities for each arm at the beginning of the trial. The default (NULL) is automatically changed to equal randomisation.
fixed_probs	numeric vector, fixed allocation probabilities for each arm - must be either a numeric vector with NA for arms without fixed probabilities and values between 0 and 1 for the other arms or NULL (default), if adaptive randomisation is used for all arms or if one of the special settings ("sqrt-based", "sqrt-based start", "sqrt-based fixed", or "match") is specified for control_prob_fixed (described below).

`min_probs` numeric vector, lower threshold for adaptive allocation probabilities, lower probabilities will be rounded up to these values. Must be `NA` (default for all arms) if no boundary is wanted.

`max_probs` numeric vector, upper threshold for adaptive allocation probabilities, higher probabilities will be rounded down to these values. Must be `NA` (default for all arms) if no boundary is wanted.

`data_looks` vector of increasing integers, specifies when to conduct adaptive analyses (= the total number of patients randomised at each adaptive analysis). The last number in the vector represents the maximum sample size. Instead of specifying `data_looks`, the `max_n` and `look_after_every` arguments can be used in combination (then `data_looks` must be `NULL`, the default).

`max_n` single integer, maximum total sample size (defaults to `NULL`). Must only be specified if `data_looks` is `NULL`. Requires specification of the `look_after_every` argument.

`look_after_every` single integer, specified together with `max_n`. Adaptive analyses will be conducted after every `look_after_every` patients randomised, and at the total sample size as specified by `max_n` (`max_n` does not need to be a multiple of `look_after_every`). If specified, `data_looks` must be `NULL` (as default).

`control` single character string, name of one of the `arms` or `NULL` (default). If specified, this arm will serve as a common control arm, to which all other arms will be compared and the inferiority/superiority/equivalence thresholds (see below) will be for those comparisons. See `setup_trial()` **Details** below for information on behaviour with respect to these comparisons.

`control_prob_fixed` if a common `control` arm is specified, this must be set to either `NULL` (the default), in which case the control arm allocation probability will not be fixed if control arms change (the allocation probability to the first control arm may still be fixed using `fixed_probs`) Otherwise a vector of probabilities of either length `1` or `number of arms - 1` can be provided, or one of the special arguments `"sqrt-based"`, `"sqrt-based start"`, `"sqrt-based fixed"` or `"match"`. See `setup_trial()` **Details** below for details in behaviour.

`inferiority` single numeric (`> 0` and `<1`, default is `0.01`) specifying the inferiority threshold. An arm will be considered inferior and dropped if the probability that it is best (when comparing all arms) or better than the control arm (when a common `control` is used) drops below this threshold.

`superiority` single numeric (`>0` and `<1`, default is `0.99`) specifying the superiority threshold. If the probability that an arm is best (when comparing all arms) or better than the control arm (when a common `control` is used) exceeds this number, said arm will be declared the winner and the trial will be stopped (if no common `control` is used or if the last comparator is dropped in a design with a common control) *or* become the new control and the trial will continue (if a common control is specified).

`equivalence_prob` single numeric (`> 0` and `< 1`) or `NULL` (default, corresponding to no equivalence assessment). If a numeric value is specified, arms will be stopped for equivalence if the probability of either *(a)* equivalence compared to a common `control`

or *(b)* equivalence between all arms remaining (designs without a common control) exceeds this threshold. Requires specification of `equivalence_diff`, `equivalence_only_first`, and a common `control` arm.

`equivalence_diff`
: single numeric value (`> 0`) or `NULL` (default, corresponding to no equivalence assessment). If a numeric value is specified, estimated differences below this threshold will be considered equivalent when assessing equivalence. For designs with a common `control` arm, the differences between each non-control arm and the `control` arm is used, and for trials without a common `control` arm, the difference between the highest and lowest estimated outcome rates are used and the trial is only stopped for equivalence if all remaining arms are thus equivalent.

`equivalence_only_first`
: single logical in trial specifications where `equivalence_prob` and `equivalence_diff` are specified, otherwise `NULL` (default). Must be `NULL` for designs without a common control arm. If a common `control` arm is used, this specifies whether equivalence will only be assessed for the first control (if `TRUE`) or also for subsequent control arms (if `FALSE`) if one arm is superior to the first control and becomes the new control.

`futility_prob`
: single numeric (`> 0` and `< 1`) or `NULL` (default, corresponds to no futility assessment). If a numeric value is specified, `arms` will be stopped for futility when the probability for futility compared to the common `control` exceeds this threshold. Requires a common `control` arm, specification of `futility_diff` and `futility_only_first`.

`futility_diff`
: single numeric value (`> 0`) or `NULL` (default, corresponding to no futility assessment). If a numeric value is specified, estimated differences below this threshold in the *beneficial* direction (as specified in `highest_is_best`) will be considered futile when assessing futility in designs with a common `control` arm. If only 1 arm remains after dropping arms for futility, the trial will be stopped without declaring the last arm superior.

`futility_only_first`
: single logical in trial specifications designs where `futility_prob` and `futility_diff` are specified, otherwise `NULL (default). Must be NULLfor designs without a commoncontrolarm.` if one arm is superior to the first control and becomes the new control.

`highest_is_best`
: single logical, specifies whether larger estimates of the outcome are favourable or not; defaults to `FALSE`, corresponding to, e.g., an undesirable binary outcomes (e.g., mortality) or a continuous outcome where lower numbers are preferred (e.g., hospital length of stay).

`soften_power`
: either a single numeric value or a numeric vector of exactly the same length as the maximum number of looks/adaptive analyses. Values must be between `0` and 1 (default); if `< 1`, then re-allocated non-fixed allocation probabilities are all raised to this power to make allocation probabilities less extreme, in turn used to redistribute remaining probability while respecting limits when defined by `min_probs` and/or `max_probs`. If 1, then no *softening* is applied.

`fun_raw_est`
: function that takes a numeric vector and returns a single numeric value, used to calculate a raw summary estimate of the outcomes in each `arm`. Defaults to `mean()`, which is always used in the `setup_trial_binom()` and `setup_trial_norm()`

functions.
Note: the function is called one time per arm during setup to validate the output structure.

`cri_width` single numeric `>= 0` and `< 1`, the width of the percentile-based credible intervals used when summarising individual trial results. Defaults to `0.95`, corresponding to 95% credible intervals.

`n_draws` single integer, the number of draws from the posterior distributions (for each arm) used when running the trial. Defaults to `5000`; can be reduced for a speed gain (at the potential loss of stability of results if too low) or increased for increased precision (takes longer). Values `< 100` are not allowed and values `< 1000` are not recommended and warned against.

`robust` single logical, if `TRUE` (default) the medians and median absolute deviations (scaled to be comparable to the standard deviation for normal distributions; MAD_SD) are used to summarise the posterior distributions; if `FALSE`, the means and standard deviations (SDs) are used instead (slightly faster, but may be less appropriate for posteriors skewed on the natural scale).

`description` optional single character string describing the trial design, will only be used in print functions if not `NULL` (the default).

`add_info` optional single string containing additional information regarding the trial design or specifications, will only be used in print functions if not `NULL` (the default).

Details

How to specify the `fun_y_gen` function

The function must take the following inputs:

- `allocs`: character vector, the trial `arms` that new patients allocated since the last adaptive analysis are randomised to.

The function must return a single numeric vector, corresponding to the outcomes for all patients allocated since the last adaptive analysis, in the same order as `allocs`.
See the **Examples** vignette (`vignette("Examples", "adaptr")`) for an example with further details.

How to specify the `fun_draws` function

The function must take the following inputs:

- `arms`: character vector, the unique trial arms, in the same order as above, but only the **currently active** arms are specified when the function is called.
- `allocs`: a vector of allocations for all patients, corresponding to the trial arms, including patients allocated to **currently inactive** `arms` when called,
- `ys`: a vector of outcomes for all patients in the same order as `allocs`, including outcomes for patients allocated to **currently inactive** `arms` when called.
- `control`: single character, the current `control arm`, will be `NULL` for designs without a common control arm, but required regardless as the argument is supplied by `run_trial()`/`run_trials()`.
- `n_draws`: single integer, the number of posterior draws for each arm.

The function must return a `matrix` (with numeric values) with `arms` columns and `n_draws` rows. The `matrix` must have columns **only for currently active arms** (when called). Each row should contain a single posterior draw for each arm on the original outcome scale: if they are estimated as, e.g., the *log(odds)*, these estimates must be transformed to probabilities and similarly for other measures.
Important: the `matrix` cannot contain `NA`s, even if no patients have been randomised to an arm yet. See the provided example for one way to alleviate this.
See the **Examples** vignette (`vignette("Examples", "adaptr")`) for an example with further details.

Notes

- Different estimation methods and prior distributions may be used; complex functions will lead to slower simulations compared to simpler methods for obtaining posterior draws, including those specified using the `setup_trial_binom()` and `setup_trial_norm()` functions.
- Technically, using log relative effect measures — e.g. log(odds ratio) or log(risk ratios) - or differences compared to a reference arm (e.g., mean differences or absolute risk differences) instead of absolute values in each arm will work to some extent (**be cautious!**):
- Stopping for superiority/inferiority/max sample sizes will work.
- Stopping for equivalence/futility may be used with relative effect measures on the log scale.
- Several summary statistics from `run_trial()` (`sum_ys` and posterior estimates) may be nonsensical if relative effect measures are used (depending on calculation method).
- In the same vein, `extract_results()` (`sum_ys`, `sq_err`, and `sq_err_te`), and `summary()` (`sum_ys_mean/sd/median/q25/q75`, `rmse`, `rmse_te` and `idp`) may be equally nonsensical when calculated on the relative scale.

Using additional custom or functions from loaded packages in the custom functions If the `fun_y_gen`, `fun_draws`, or `fun_raw_est` functions calls other user-specified functions (or uses objects defined by the user outside these functions or the `setup_trial()`-call) or functions from external packages and simulations are conducted on multiple cores, these objects or functions must be exported or prefixed with their namespaces, respectively, as described in `run_trials()`.

More information on arguments

- `control`: if one or more treatment arms are superior to the control arm (i.e., passes the superiority threshold as defined above), this arm will become the new control (if multiple arms are superior, the one with the highest probability of being the overall best will become the new control), the previous control will be dropped for inferiority, and all remaining arms will be immediately compared to the new control in the same adaptive analysis and dropped if inferior (or possibly equivalent/futile, see below) compared to this new control arm. Only applies in trials with a common `control`.
- `control_prob_fixed`: If the length is 1, then this allocation probability will be used for the `control` group (including if a new arm becomes the control and the original control is dropped). If multiple values are specified the first value will be used when all arms are active, the second when one arm has been dropped, and so forth. If 1 or more values are specified, previously set `fixed_probs`, `min_probs` or `max_probs` for new control arms will be ignored. If all allocation probabilities do not sum to 1 (e.g, due to multiple limits) they will be re-scaled to do so.
  Can also be set to one of the special arguments `"sqrt-based"`, `"sqrt-based start"`, `"sqrt-based`

fixed" or "match" (written exactly as one of those, case sensitive). This requires start_probs to be NULL and relevant fixed_probs to be NULL (or NA for the control arm).
If one of the "sqrt-based"/"sqrt-based start"/"sqrt-based fixed" options are used, the function will set *square-root-transformation-based* starting allocation probabilities. These are defined as:
square root of number of non-control arms to 1-ratio for other arms
scaled to sum to 1, which will generally increase power for comparisons against the common control, as discussed in, e.g., *Park et al, 2020* doi: 10.1016/j.jclinepi.2020.04.025.
If "sqrt-based", square-root-transformation-based allocation probabilities will also be used for new controls when arms are dropped. If "sqrt-based start", the control arm will be fixed to this allocation probability at all times (also after arm dropping, with re-scaling as necessary, as specified above). If "sqrt-based fixed" is chosen, square-root-transformation-based allocation probabilities will be used and all allocation probabilities will be fixed throughout the trial (with re-scaling when arms are dropped).
If "match" is specified, the control group allocation will always be *matched* to be similar to the highest non-control arm allocation ratio.

Superiority and inferiority

In trial designs without a common control arm, superiority and inferiority are assessed by comparing all **currently active** groups. This means that if a "final" analysis of a trial without a common control and > 2 arms is conducted including all arms (as will often be done in practice) *after* an adaptive trial have stopped, the final probabilities of the best arm being superior may differ slightly. For example, in a trial with three arms and no common control arm, one arm may be dropped early for inferiority defined as < 1% probability of being the overall best arm. The trial may then continue with the two remaining arms, and stopped when one is declared superior to the other defined as > 99% probability of being the overall best arm. If a final analysis is then conducted including all arms, the final probability of the best arm being overall superior will generally be slightly lower as the probability of the first dropped arm being the best will generally be > 0%, even if very low and below the inferiority threshold.
This is not relevant trial designs *with* a common control, as pairwise assessments of superiority/inferiority compared to the common control will not be influenced similarly by previously dropped arms (and previously dropped arms may be included in the analyses, even if posterior distributions are not returned for those). Similarly, in actual clinical trials, final probabilities may change slightly as the most recently randomised patients will generally not have outcome data available at the final adaptive analysis where the trial is stopped.

Equivalence

Equivalence is assessed **after** both inferiority and superiority have been assessed (and in case of superiority, it will be assessed against the new control arm in designs with a common control, if specified - see above).

Futility

Futility is assessed **after** inferiority, superiority, **and** equivalence have been assessed (and in case of superiority, it will be assessed against the new control arm in designs with a common control, if specified - see above). Arms will thus be dropped for equivalence before futility.

Value

A trial_spec object used to run simulations by run_trial() or run_trials(). The output is essentially a list containing the input values (some combined in a data.frame called trial_arms),

but its class signals that these inputs have been validated and inappropriate combinations and settings have been ruled out. Also contains `best_arm` holding the arm(s) with the best value(s) in `true_ys`. Use `str()` to peruse the actual content of the returned object

Examples

```
# Setup a custom trial specification with right-skewed, log-normally
# distributed continuous outcomes (higher values are worse)

# Define the function that will generate the outcomes in each arm
# Notice: contents should match arms/true_ys in the setup_trial() call below
get_ys_lognorm <- function(allocs) {
  y <- numeric(length(allocs))
  # arms (names and order) and values (except for exponentiation) should match
  # those used in setup_trial (below)
  means <- c("Control" = 2.2, "Experimental A" = 2.1, "Experimental B" = 2.3)
  for (arm in names(means)) {
    ii <- which(allocs == arm)
    y[ii] <- rlnorm(length(ii), means[arm], 1.5)
  }
  y
}

# Define the function that will generate posterior draws
# In this example, the function uses no priors (corresponding to improper
# flat priors) and calculates results on the log-scale, before exponentiating
# back to the natural scale, which is required for assessments of
# equivalence, futility and general interpretation
get_draws_lognorm <- function(arms, allocs, ys, control, n_draws) {
  draws <- list()
  logys <- log(ys)
  for (arm in arms){
    ii <- which(allocs == arm)
    n <- length(ii)
    if (n > 1) {
      # Necessary to avoid errors if too few patients randomised to this arm
    draws[[arm]] <- exp(rnorm(n_draws, mean = mean(logys[ii]), sd = sd(logys[ii])/sqrt(n - 1)))
    } else {
      # Too few patients randomised to this arm - extreme uncertainty
    draws[[arm]] <- exp(rnorm(n_draws, mean = mean(logys), sd = 1000 * (max(logys) - min(logys))))
    }
  }
  do.call(cbind, draws)
}

# The actual trial specification is then defined
lognorm_trial <- setup_trial(
  # arms should match those above
  arms = c("Control", "Experimental A", "Experimental B"),
  # true_ys should match those above
  true_ys = exp(c(2.2, 2.1, 2.3)),
  fun_y_gen = get_ys_lognorm, # as specified above
  fun_draws = get_draws_lognorm, # as specified above
```

```
  max_n = 5000,
  look_after_every = 200,
  control = "Control",
  # Qquare-root-based, fixed control group allocation ratio
  # and response-adaptive randomisation for other arms
  control_prob_fixed = "sqrt-based",
  # Equivalence assessment
  equivalence_prob = 0.9,
  equivalence_diff = 0.5,
  equivalence_only_first = TRUE,
  highest_is_best = FALSE,
  # Summarise raw results by taking the mean on the
  # log scale and back-transforming
  fun_raw_est = function(x) exp(mean(log(x))) ,
  # Summarise posteriors using medians with MAD-SDs,
  # as distributions will not be normal on the actual scale
  robust = TRUE,
  # Description/additional info used when printing
  description = "continuous, log-normally distributed outcome",
  add_info = "SD on the log scale for all arms: 1.5"
)

# Print trial specification with 3 digits for all probabilities
print(lognorm_trial, prob_digits = 3)
```

`setup_trial_binom` *Setup a trial specification using a binary, binomially distributed outcome*

Description

Specifies the design of an adaptive trial with a binary, binomially distributed outcome and validates all inputs. Uses *beta-binomial* conjugate models with `beta(1, 1)` prior distributions, corresponding to a uniform prior (or the addition of 2 patients, 1 with an event and 1 without) to the trial. Use `run_trial()` or `run_trials()` to conduct single/multiple simulations of the specified trial, respectively.

Note: `add_info` as specified in `setup_trial()` is set to `NULL` for trial specifications setup by this function.

Further details: please see `setup_trial()`. See `setup_trial_norm()` for simplified setup of trials with normally distributed continuous outcomes.

For additional trial specification examples, see the the **Basic examples** vignette (`vignette("Basic-examples", package = "adaptr")`) and the **Advanced example** vignette (`vignette("Advanced-example", package = "adaptr")`).

Usage

```
setup_trial_binom(
  arms,
```

```
  true_ys,
  start_probs = NULL,
  fixed_probs = NULL,
  min_probs = rep(NA, length(arms)),
  max_probs = rep(NA, length(arms)),
  data_looks = NULL,
  max_n = NULL,
  look_after_every = NULL,
  control = NULL,
  control_prob_fixed = NULL,
  inferiority = 0.01,
  superiority = 0.99,
  equivalence_prob = NULL,
  equivalence_diff = NULL,
  equivalence_only_first = NULL,
  futility_prob = NULL,
  futility_diff = NULL,
  futility_only_first = NULL,
  highest_is_best = FALSE,
  soften_power = 1,
  cri_width = 0.95,
  n_draws = 5000,
  robust = TRUE,
  description = "generic binomially distributed outcome trial"
)
```

Arguments

arms	character vector with unique names for the trial arms.
true_ys	numeric vector, true probabilities (between 0 and 1) of outcomes in all trial arms.
start_probs	numeric vector, allocation probabilities for each arm at the beginning of the trial. The default (NULL) is automatically changed to equal randomisation.
fixed_probs	numeric vector, fixed allocation probabilities for each arm - must be either a numeric vector with NA for arms without fixed probabilities and values between 0 and 1 for the other arms or NULL (default), if adaptive randomisation is used for all arms or if one of the special settings ("sqrt-based", "sqrt-based start", "sqrt-based fixed", or "match") is specified for control_prob_fixed (described below).
min_probs	numeric vector, lower threshold for adaptive allocation probabilities, lower probabilities will be rounded up to these values. Must be NA (default for all arms) if no boundary is wanted.
max_probs	numeric vector, upper threshold for adaptive allocation probabilities, higher probabilities will be rounded down to these values. Must be NA (default for all arms) if no boundary is wanted.
data_looks	vector of increasing integers, specifies when to conduct adaptive analyses (= the total number of patients randomised at each adaptive analysis). The last

number in the vector represents the maximum sample size. Instead of specifying `data_looks`, the `max_n` and `look_after_every` arguments can be used in combination (then `data_looks` must be `NULL`, the default).

`max_n` single integer, maximum total sample size (defaults to `NULL`). Must only be specified if `data_looks` is `NULL`. Requires specification of the `look_after_every` argument.

`look_after_every`
single integer, specified together with `max_n`. Adaptive analyses will be conducted after every `look_after_every` patients randomised, and at the total sample size as specified by `max_n` (`max_n` does not need to be a multiple of `look_after_every`). If specified, `data_looks` must be `NULL` (as default).

`control` single character string, name of one of the `arms` or `NULL` (default). If specified, this arm will serve as a common control arm, to which all other arms will be compared and the inferiority/superiority/equivalence thresholds (see below) will be for those comparisons. See `setup_trial()` **Details** below for information on behaviour with respect to these comparisons.

`control_prob_fixed`
if a common `control` arm is specified, this must be set to either `NULL` (the default), in which case the control arm allocation probability will not be fixed if control arms change (the allocation probability to the first control arm may still be fixed using `fixed_probs`) Otherwise a vector of probabilities of either length `1` or `number of arms - 1` can be provided, or one of the special arguments `"sqrt-based"`, `"sqrt-based start"`, `"sqrt-based fixed"` or `"match"`. See `setup_trial()` **Details** below for details in behaviour.

`inferiority` single numeric (`> 0` and `<1`, default is `0.01`) specifying the inferiority threshold. An arm will be considered inferior and dropped if the probability that it is best (when comparing all arms) or better than the control arm (when a common `control` is used) drops below this threshold.

`superiority` single numeric (`>0` and `<1`, default is `0.99`) specifying the superiority threshold. If the probability that an arm is best (when comparing all arms) or better than the control arm (when a common `control` is used) exceeds this number, said arm will be declared the winner and the trial will be stopped (if no common `control` is used or if the last comparator is dropped in a design with a common control) *or* become the new control and the trial will continue (if a common control is specified).

`equivalence_prob`
single numeric (`> 0` and `< 1`) or `NULL` (default, corresponding to no equivalence assessment). If a numeric value is specified, arms will be stopped for equivalence if the probability of either *(a)* equivalence compared to a common `control` or *(b)* equivalence between all arms remaining (designs without a common control) exceeds this threshold. Requires specification of `equivalence_diff`, `equivalence_only_first`, and a common `control` arm.

`equivalence_diff`
single numeric value (`> 0`) or `NULL` (default, corresponding to no equivalence assessment). If a numeric value is specified, estimated differences below this threshold will be considered equivalent when assessing equivalence. For designs with a common `control` arm, the differences between each non-control arm and

the control arm is used, and for trials without a common control arm, the difference between the highest and lowest estimated outcome rates are used and the trial is only stopped for equivalence if all remaining arms are thus equivalent.

equivalence_only_first
: single logical in trial specifications where equivalence_prob and equivalence_diff are specified, otherwise NULL (default). Must be NULL for designs without a common control arm. If a common control arm is used, this specifies whether equivalence will only be assessed for the first control (if TRUE) or also for subsequent control arms (if FALSE) if one arm is superior to the first control and becomes the new control.

futility_prob
: single numeric (> 0 and < 1) or NULL (default, corresponds to no futility assessment). If a numeric value is specified, arms will be stopped for futility when the probability for futility compared to the common control exceeds this threshold. Requires a common control arm, specification of futility_diff and futility_only_first.

futility_diff
: single numeric value (> 0) or NULL (default, corresponding to no futility assessment). If a numeric value is specified, estimated differences below this threshold in the *beneficial* direction (as specified in highest_is_best) will be considered futile when assessing futility in designs with a common control arm. If only 1 arm remains after dropping arms for futility, the trial will be stopped without declaring the last arm superior.

futility_only_first
: single logical in trial specifications designs where futility_prob and futility_diff are specified, otherwise NULL (default). Must be NULLfor designs without a commoncontrolarm. if one arm is superior to the first control and becomes the new control.

highest_is_best
: single logical, specifies whether larger estimates of the outcome are favourable or not; defaults to FALSE, corresponding to, e.g., an undesirable binary outcomes (e.g., mortality) or a continuous outcome where lower numbers are preferred (e.g., hospital length of stay).

soften_power
: either a single numeric value or a numeric vector of exactly the same length as the maximum number of looks/adaptive analyses. Values must be between 0 and 1 (default); if < 1, then re-allocated non-fixed allocation probabilities are all raised to this power to make allocation probabilities less extreme, in turn used to redistribute remaining probability while respecting limits when defined by min_probs and/or max_probs. If 1, then no *softening* is applied.

cri_width
: single numeric >= 0 and < 1, the width of the percentile-based credible intervals used when summarising individual trial results. Defaults to 0.95, corresponding to 95% credible intervals.

n_draws
: single integer, the number of draws from the posterior distributions (for each arm) used when running the trial. Defaults to 5000; can be reduced for a speed gain (at the potential loss of stability of results if too low) or increased for increased precision (takes longer). Values < 100 are not allowed and values < 1000 are not recommended and warned against.

robust
: single logical, if TRUE (default) the medians and median absolute deviations (scaled to be comparable to the standard deviation for normal distributions;

MAD_SD) are used to summarise the posterior distributions; if FALSE, the means and standard deviations (SDs) are used instead (slightly faster, but may be less appropriate for posteriors skewed on the natural scale).

description — character string, default is "generic binomially distributed outcome trial". See arguments of setup_trial().

Value

A trial_spec object used to run simulations by run_trial() or run_trials(). The output is essentially a list containing the input values (some combined in a data.frame called trial_arms), but its class signals that these inputs have been validated and inappropriate combinations and settings have been ruled out. Also contains best_arm holding the arm(s) with the best value(s) in true_ys. Use str() to peruse the actual content of the returned object

Examples

```
# Setup a trial specification a binary, binomially distributed, undesirable outcome
binom_trial <- setup_trial_binom(
  arms = c("Arm A", "Arm B", "Arm C"),
  true_ys = c(0.25, 0.20, 0.30),
  # Minimum allocation of 15% in all arms
  min_probs = rep(0.15, 3),
  data_looks = seq(from = 300, to = 2000, by = 100),
  # Stop for equivalence if > 90% probability of
  # differences < 5 percentage points
  equivalence_prob = 0.9,
  equivalence_diff = 0.05,
  soften_power = 0.5 # Limit extreme allocation ratios
)

# Print using 3 digits for probabilities
print(binom_trial, prob_digits = 3)
```

---

setup_trial_norm — *Setup a trial specification using a continuous, normally distributed outcome*

Description

Specifies the design of an adaptive trial with a continuous, normally distributed outcome and validates all inputs. Uses normally distributed posterior distributions for the mean values in each trial arm; technically, no priors are used (as using *normal-normal* conjugate prior models with extremely wide or uniform priors gives similar results for these simple, unadjusted estimates). Technically, this thus corresponds to using improper, flat priors, although not explicitly specified as such. Use run_trial() or run_trials() to conduct single/multiple simulations of the specified trial, respectively.

Note: add_info as specified in setup_trial() is set to the arms and standard deviations used for

trials specified using this function.
Further details: please see `setup_trial()`. See `setup_trial_binom()` for simplified setup of trials with binomially distributed binary outcomes.
For additional trial specification examples, see the the **Basic examples** vignette (`vignette("Basic-examples", package = "adaptr")`) and the **Advanced example** vignette (`vignette("Advanced-example", package = "adaptr")`).

Usage

```
setup_trial_norm(
  arms,
  true_ys,
  sds,
  start_probs = NULL,
  fixed_probs = NULL,
  min_probs = rep(NA, length(arms)),
  max_probs = rep(NA, length(arms)),
  data_looks = NULL,
  max_n = NULL,
  look_after_every = NULL,
  control = NULL,
  control_prob_fixed = NULL,
  inferiority = 0.01,
  superiority = 0.99,
  equivalence_prob = NULL,
  equivalence_diff = NULL,
  equivalence_only_first = NULL,
  futility_prob = NULL,
  futility_diff = NULL,
  futility_only_first = NULL,
  highest_is_best = FALSE,
  soften_power = 1,
  cri_width = 0.95,
  n_draws = 5000,
  robust = FALSE,
  description = "generic normally distributed outcome trial"
)
```

Arguments

`arms`	character vector with unique names for the trial arms.
`true_ys`	numeric vector, simulated means of the outcome in all trial `arms`.
`sds`	numeric vector, true standard deviations (must be > 0) of the outcome in all trial `arms`.
`start_probs`	numeric vector, allocation probabilities for each arm at the beginning of the trial. The default (`NULL`) is automatically changed to equal randomisation.
`fixed_probs`	numeric vector, fixed allocation probabilities for each arm - must be either a numeric vector with `NA` for arms without fixed probabilities and values between `0`

and 1 for the other arms or NULL (default), if adaptive randomisation is used for all arms or if one of the special settings ("sqrt-based", "sqrt-based start", "sqrt-based fixed", or "match") is specified for control_prob_fixed (described below).

min_probs — numeric vector, lower threshold for adaptive allocation probabilities, lower probabilities will be rounded up to these values. Must be NA (default for all arms) if no boundary is wanted.

max_probs — numeric vector, upper threshold for adaptive allocation probabilities, higher probabilities will be rounded down to these values. Must be NA (default for all arms) if no boundary is wanted.

data_looks — vector of increasing integers, specifies when to conduct adaptive analyses (= the total number of patients randomised at each adaptive analysis). The last number in the vector represents the maximum sample size. Instead of specifying data_looks, the max_n and look_after_every arguments can be used in combination (then data_looks must be NULL, the default).

max_n — single integer, maximum total sample size (defaults to NULL). Must only be specified if data_looks is NULL. Requires specification of the look_after_every argument.

look_after_every — single integer, specified together with max_n. Adaptive analyses will be conducted after every look_after_every patients randomised, and at the total sample size as specified by max_n (max_n does not need to be a multiple of look_after_every). If specified, data_looks must be NULL (as default).

control — single character string, name of one of the arms or NULL (default). If specified, this arm will serve as a common control arm, to which all other arms will be compared and the inferiority/superiority/equivalence thresholds (see below) will be for those comparisons. See setup_trial() **Details** below for information on behaviour with respect to these comparisons.

control_prob_fixed — if a common control arm is specified, this must be set to either NULL (the default), in which case the control arm allocation probability will not be fixed if control arms change (the allocation probability to the first control arm may still be fixed using fixed_probs) Otherwise a vector of probabilities of either length 1 or number of arms - 1 can be provided, or one of the special arguments "sqrt-based", "sqrt-based start", "sqrt-based fixed" or "match". See setup_trial() **Details** below for details in behaviour.

inferiority — single numeric (> 0 and <1, default is 0.01) specifying the inferiority threshold. An arm will be considered inferior and dropped if the probability that it is best (when comparing all arms) or better than the control arm (when a common control is used) drops below this threshold.

superiority — single numeric (>0 and <1, default is 0.99) specifying the superiority threshold. If the probability that an arm is best (when comparing all arms) or better than the control arm (when a common control is used) exceeds this number, said arm will be declared the winner and the trial will be stopped (if no common control is used or if the last comparator is dropped in a design with a common control) *or* become the new control and the trial will continue (if a common control is specified).

equivalence_prob
: single numeric (> 0 and < 1) or NULL (default, corresponding to no equivalence assessment). If a numeric value is specified, arms will be stopped for equivalence if the probability of either *(a)* equivalence compared to a common control or *(b)* equivalence between all arms remaining (designs without a common control) exceeds this threshold. Requires specification of equivalence_diff, equivalence_only_first, and a common control arm.

equivalence_diff
: single numeric value (> 0) or NULL (default, corresponding to no equivalence assessment). If a numeric value is specified, estimated differences below this threshold will be considered equivalent when assessing equivalence. For designs with a common control arm, the differences between each non-control arm and the control arm is used, and for trials without a common control arm, the difference between the highest and lowest estimated outcome rates are used and the trial is only stopped for equivalence if all remaining arms are thus equivalent.

equivalence_only_first
: single logical in trial specifications where equivalence_prob and equivalence_diff are specified, otherwise NULL (default). Must be NULL for designs without a common control arm. If a common control arm is used, this specifies whether equivalence will only be assessed for the first control (if TRUE) or also for subsequent control arms (if FALSE) if one arm is superior to the first control and becomes the new control.

futility_prob
: single numeric (> 0 and < 1) or NULL (default, corresponds to no futility assessment). If a numeric value is specified, arms will be stopped for futility when the probability for futility compared to the common control exceeds this threshold. Requires a common control arm, specification of futility_diff and futility_only_first.

futility_diff
: single numeric value (> 0) or NULL (default, corresponding to no futility assessment). If a numeric value is specified, estimated differences below this threshold in the *beneficial* direction (as specified in highest_is_best) will be considered futile when assessing futility in designs with a common control arm. If only 1 arm remains after dropping arms for futility, the trial will be stopped without declaring the last arm superior.

futility_only_first
: single logical in trial specifications designs where futility_prob and futility_diff are specified, otherwise NULL (default). Must be NULLfor designs without a commoncontrolarm. if one arm is superior to the first control and becomes the new control.

highest_is_best
: single logical, specifies whether larger estimates of the outcome are favourable or not; defaults to FALSE, corresponding to, e.g., an undesirable binary outcomes (e.g., mortality) or a continuous outcome where lower numbers are preferred (e.g., hospital length of stay).

soften_power
: either a single numeric value or a numeric vector of exactly the same length as the maximum number of looks/adaptive analyses. Values must be between 0 and 1 (default); if < 1, then re-allocated non-fixed allocation probabilities are all raised to this power to make allocation probabilities less extreme, in turn used to redistribute remaining probability while respecting limits when defined by min_probs and/or max_probs. If 1, then no *softening* is applied.

cri_width	single numeric >= 0 and < 1, the width of the percentile-based credible intervals used when summarising individual trial results. Defaults to 0.95, corresponding to 95% credible intervals.
n_draws	single integer, the number of draws from the posterior distributions (for each arm) used when running the trial. Defaults to 5000; can be reduced for a speed gain (at the potential loss of stability of results if too low) or increased for increased precision (takes longer). Values < 100 are not allowed and values < 1000 are not recommended and warned against.
robust	single logical, if TRUE (default) the medians and median absolute deviations (scaled to be comparable to the standard deviation for normal distributions; MAD_SD) are used to summarise the posterior distributions; if FALSE, the means and standard deviations (SDs) are used instead (slightly faster, but may be less appropriate for posteriors skewed on the natural scale).
description	character string, default is "generic normally distributed outcome trial". See arguments of setup_trial().

Details

Because the posteriors used in this type of trial (with a generic, continuous, normally distributed outcome) are by definition normally distributed, FALSE is used as the default value for the robust argument.

Value

A trial_spec object used to run simulations by run_trial() or run_trials(). The output is essentially a list containing the input values (some combined in a data.frame called trial_arms), but its class signals that these inputs have been validated and inappropriate combinations and settings have been ruled out. Also contains best_arm holding the arm(s) with the best value(s) in true_ys. Use str() to peruse the actual content of the returned object

Examples

```
# Setup a trial specification using a continuous, normally distributed, desirable outcome
norm_trial <- setup_trial_norm(
  arms = c("Control", "New A", "New B", "New C"),
  true_ys = c(15, 20, 14, 13),
  sds = c(2, 2.5, 1.9, 1.8), # SDs in each arm
  max_n = 500,
  look_after_every = 50,
  control = "Control", # Common control arm
  # Square-root-based, fixed control group allocation ratios
  control_prob_fixed = "sqrt-based fixed",
  # Desirable outcome
  highest_is_best = TRUE,
  soften_power = 0.5 # Limit extreme allocation ratios
)

# Print using 3 digits for probabilities
print(norm_trial, prob_digits = 3)
```

summary *Summary of simulated trial results*

Description

Summarises simulation results from the run_trials() function. Uses extract_results(), which may be used directly to extract key trial results without summarising.

Usage

```
## S3 method for class 'trial_results'
summary(
  object,
  select_strategy = "control if available",
  select_last_arm = FALSE,
  select_preferences = NULL,
  te_comp = NULL,
  raw_ests = FALSE,
  restrict = NULL,
  ...
)
```

Arguments

object trial_results object, output from the run_trials() function.

select_strategy

single character string. For trials not stopped due to superiority (or with only 1 arm remaining, if select_last_arm is set to TRUE in trial designs with a common control arm; see below), this parameter specifies which arm will be considered selected when calculating trial design performance metrics (described below; this corresponds to the consequence of an inconclusive trial, i.e., which arm would then be used in practice).
The following options are available and must be written exactly as below (case sensitive, cannot be abbreviated):

- "control if available" (default): selects the **first** control arm for trials with a common control arm **if** this arm is active at end-of-trial, otherwise no arm will be selected. For trial designs without a common control, no arm will be selected.
- "none": selects no arm in trials not ending with superiority.
- "control": similar to "control if available", but will throw an error for trial designs without a common control arm.
- "final control": selects the **final** control arm regardless of whether the trial was stopped for practical equivalence, futility, or at the maximum sample size; this strategy can only be specified for trial designs with a common control arm.

- "control or best": selects the **first** control arm if still active at end-of-trial, otherwise selects the best remaining arm (defined as the remaining arm with the highest probability of being the best in the final analysis). Only works for trial designs with a common control arm.
- "best": selects the best remaining arm (as described under "control or best").
- "list or best": selects the first remaining arm from a specified list (specified using select_preferences, technically a character vector). If none of these arms are are active at end-of-trial, the best remaining arm will be selected (as described above).
- "list": as specified above, but if no arms on the provided list remain active at end-of-trial, no arm is selected.

select_last_arm
: single logical, defaults to FALSE. If TRUE, the only remaining active arm (the last control) will be selected in trials with a common control arm ending with equivalence or futility, before considering the options specified in select_strategy. Must be FALSE for trial designs without a common control arm.

select_preferences
: character vector specifying a number of arms used for selection if one of the "list or best" or "list" options are specified for select_strategy. Can only contain valid arms available in the trial.

te_comp
: character string, treatment-effect comparator. Can be either NULL (the default) in which case the **first** control arm is used for trial designs with a common control arm, or a single trial arm. Will be used when calculating sq_err_te (the squared error of the treatment effect comparing the selected arm to the comparator arm, as described below).

raw_ests
: single logical. If FALSE (default), the posterior estimates (post_ests, see setup_trial() and run_trial()) will be used to calculate sq_err (the squared error of the estimated compared to the specified effect in the selected arm) and sq_err_te (the squared error of the treatment effect comparing the selected arm to the comparator arm, as described for te_comp and below). If TRUE, the raw estimates (raw_ests, see setup_trial() and run_trial()) will be used instead of the posterior estimates.

restrict
: single character string or NULL. If NULL (default), results are summarised for all simulations; if "superior, results are summarised for simulations ending with superiority only; if "selected", results are summarised for simulations ending with a selected arm (according to the specified arm selection strategy for simulations not ending with superiority). Some summary measures (e.g., prob_conclusive) can only be calculated across all simulations and several are calculated regardless of restrict settings, but have substantially different interpretations if restricted.

...
: additional arguments, not used.

Details

The ideal design percentage (IDP) returned (described below) is based on *Viele et al, 2020* doi: 10.1177/1740774519877836 and has been adapted to work for trials with both desirable/undesirable out-

comes and non-binary outcomes. Briefly, the expected outcome is calculated as the sum of the true outcomes in each arm multiplied by the corresponding selection probabilities (ignoring simulations with no selected arm). The IDP is then calculated as:

- For desirable outcomes:
  `100 * (expected outcome - lowest true outcome) / (highest true outcome - lowest true outcome)`
- For undesirable outcomes:
  `100 - IDP calculated for desirable outcomes`

Value

A `"trial_results_summary"` object containing the following:

- `n_rep`: the number of simulations.
- `n_summarised`: the number of simulations summarised.
- `highest_is_best`: as specified in `setup_trial()`.
- `elapsed_time`: the total simulation time.
- `size_mean`, `size_sd`, `size_median`, `size_p25`, `size_p75`: the mean, standard deviation, median as well as 25- and 75-percentiles of the sample sizes of the summarised trial simulations.
- `sum_ys_mean`, `sum_ys_sd`, `sum_ys_median`, `sum_ys_p25`, `sum_ys_p75`: the mean, standard deviation, median as well as 25- and 75-percentiles of the total `sum_ys` (e.g., the total number of events in trials with a binary outcome, or the sums of continuous values for all patients across all arms in trials with a continuous outcome) across all arms in the summarised trial simulations.
- `ratio_ys_mean`, `ratio_ys_sd`, `ratio_ys_median`, `ratio_ys_p25`, `ratio_ys_p75`: the mean, standard deviation, median as well as 25- and 75-percentiles of the final `ratio_ys` (`sum_ys/final_n`) across all arms in the summarised trial simulations.
- `prob_conclusive`: the proportion of conclusive trial simulations (simulations not stopped at the maximum sample size without a superiority, equivalence or futility decision).
- `prob_superior`, `prob_equivalence`, `prob_futility`, `prob_max`: the proportion (0-1) of trial simulations stopped for superiority, equivalence, futility or inconclusive at the maximum allowed sample size, respectively.
- `prob_select_*`: the selection probabilities for each arm and for no selection, according to the specified selection strategy. Contains one element per `arm`, named as `prob_select_arm_<arm name>` and `prob_select_none` for the probability of selecting no arm.
- `rmse`, `rmse_te`: the root mean squared error of the estimates for the selected arm and for the treatment effect, as described further in `extract_results()`.
- `idp`: the ideal design percentage (IDP; 0-100%), see **Details**.
- `select_strategy`, `select_last_arm`, `select_preferences`, `te_comp`, `raw_ests`,`restrict`: as specified above.
- `control`: the control arm specified by `setup_trial()`, `setup_trial_binom()` or `setup_trial_norm()`; `NULL` if no control.
- `equivalence_assessed`, `futility_assessed`: single logicals, specifies whether the trial design specification includes assessments of equivalence and/or futility.
- `base_seed`: as specified in `run_trials()`.
- `cri_width`, `n_draws`, `robust`, `description`, `add_info`: as specified in `setup_trial()`, `setup_trial_binom()` or `setup_trial_norm()`.

Examples

```
# Setup a trial specification
binom_trial <- setup_trial_binom(arms = c("A", "B", "C", "D"),
                                 control = "A",
                                 true_ys = c(0.20, 0.18, 0.22, 0.24),
                                 data_looks = 1:20 * 100)

# Run 10 simulations with a specified random base seed
res <- run_trials(binom_trial, n_rep = 10, base_seed = 12345)

# Summarise simulations - select the control arm if available in trials not
# ending with a superiority decision
res_sum <- summary(res, select_strategy = "control")

# Print summary
print(res_sum, digits = 1)
```

Index