

Package: AHTauDesign (via r-universe)

October 18, 2026

Type Package

Title Prospective Trial Planning for Average Hazard Estimands

Version 1.0

Description Provides a simulation-based prospective planning workflow for average hazard estimands in two-arm time-to-event trials. Supports calibration and freezing of design-specific information-support rules, assessment of clinically preferred truncation horizons, conditional selection of clinically acceptable alternative horizons, and reverse design over sample size and additional follow-up. Separates information support from statistical power and supports auditable planning reports.

License GPL-3

Encoding UTF-8

Depends R (>= 4.1.0)

Imports stats, survAH (>= 1.2.0)

Suggests knitr, rmarkdown, testthat (>= 3.0.0), roxygen2

VignetteBuilder knitr

Config/testthat/edition 3

RoxygenNote 7.3.3

NeedsCompilation no

Author Xinyue Zhao [aut, cre], Haitao Pan [aut]

Maintainer Xinyue Zhao <xinyuezhao27@gwu.edu>

Repository <https://xyzhao27.r-universe.dev>

Date/Publication 2026-10-18 11:00:40 UTC

RemoteUrl <https://github.com/cran/AHTauDesign>

RemoteRef HEAD

RemoteSha 21521f3238dc76a7cd471d5b7d6bd1a6d6a76d26

Contents

find_n_req	2
freeze_support_rule	3
planning_report	7
print.ahtau_planning_report	8
simulate_calibration_null	9
simulate_design	12
stage1_audit	13
stage2_evaluate	14
stage2_select	15

Index	16
-------	----

find_n_req	<i>Part II Reverse Design for Average Hazard</i>
------------	--

Description

Determine the required sample size for a fixed AH horizon and additional follow-up duration.

Usage

```
find_n_req(  
  F_followup,  
  tau,  
  thresholds,  
  trial_args = list(),  
  trial_generator,  
  N_lower = 100,  
  N_upper = 600,  
  coarse_step = 20,  
  fine_step = 10,  
  n_coarse,  
  n_fine,  
  n_validate,  
  power_target = 0.8,  
  search_seed,  
  validation_seed,  
  alpha = 0.05  
)
```

Arguments

F_followup	Additional follow-up duration.
tau	Prespecified AH horizon.
thresholds	Frozen Stage 1 support thresholds.

trial_args	Named list of additional generator arguments. Do not include 'N' or 'F': the search supplies both at each evaluated sample size.
trial_generator	Function used to simulate trial data.
N_lower	Whole-number lower sample-size bound (at least 2).
N_upper	Whole-number upper bound, greater than 'N_lower'.
coarse_step	Positive whole-number coarse increment; must be a multiple of 'fine_step'.
fine_step	Positive whole-number fine increment. 'N_upper - N_lower' must be divisible by 'fine_step'.
n_coarse	Number of simulations per coarse evaluation.
n_fine	Number of simulations per fine evaluation.
n_validate	Number of simulations per validation.
power_target	Target overall empirical power.
search_seed	Random seed for coarse and fine search.
validation_seed	Independent validation seed, distinct from 'search_seed'. Different seeds are required but alone do not guarantee independence if the generator resets its own RNG state.
alpha	One-sided significance level.

Details

The procedure uses coarse search, fine localization, and independent validation. Final feasibility requires Stage 1 support and isotonic-stabilized overall power meeting the target.

Value

A list containing the required sample size, search status, coarse and fine results, independent validation results, complete search trace, and settings.

freeze_support_rule	<i>Freeze a Calibrated Stage 1 Support Rule</i>
---------------------	---

Description

Evaluate support-based calibration diagnostics using the existing reliability and candidate-rule functions, validate an explicitly selected rule, and save its thresholds and calibration evidence for subsequent planning.

Usage

```
freeze_support_rule(
  diagnostic,
  rules,
  rule_id,
  rationale,
  calibration_scope,
  provenance,
  simulation_settings,
  type1_tolerance = 0.06,
  coverage_tolerance = 0.94,
  support_tolerance = 0.95
)
```

Arguments

diagnostic	A data frame with one row per calibration setting, uniquely identified by <code>scenario_id</code> and <code>tau</code> . Required support columns are <code>min_mean_p_risk</code> , <code>min_mean_n_risk</code> , <code>p_support</code> , <code>p_support_lower</code> , and <code>p_support_upper</code> . Required inference columns are <code>type1_one_lower</code> , <code>type1_one_upper</code> , <code>coverage_lower</code> , and <code>coverage_upper</code> : these must be Monte Carlo intervals for Type I error and coverage conditional on support-valid trials. Also accepts the manuscript audit column names <code>type1_support_lower</code> , <code>type1_support_upper</code> , <code>coverage_support_lower</code> , and <code>coverage_support_upper</code> without altering their values. Support validity requires numerical validity and a nonempty risk set in each arm. Missing interval bounds retain the existing classifier's boundary-based decisions: an available bound may establish Pass or Fail; otherwise the component remains Indeterminate. Attempted simulation counts may be supplied as <code>total_sim_n</code> (the manuscript field) or <code>n_sim</code> . If either count field or <code>n_support_valid</code> is supplied, at least one attempted-count field and <code>n_support_valid</code> are required and checked against <code>p_support</code> . When both attempted-count fields are supplied, they must agree in every row. All columns and denominators are retained without renaming or replacement. At least two calibration settings are required.
rules	A candidate-rule data frame with at least one row and unique <code>rule_id</code> , <code>p_risk_threshold</code> , <code>p_support_threshold</code> , logical <code>use_absolute_n</code> , and <code>n_risk_threshold</code> columns. Supply family labels as <code>rule_family</code> (the manuscript field) or the compatible alias <code>family</code> . If both are present, their labels must agree in every row. All supplied columns are retained. Single-candidate evaluation preserves a one-column pass matrix. The <code>p_support_threshold</code> field specifies the required support-availability point estimate (<code>p_support</code>). Absolute thresholds must be finite and nonnegative when <code>use_absolute_n</code> is <code>TRUE</code> ; otherwise they may be <code>NA</code> . Family names are retained as labels; <code>use_absolute_n</code> determines the implemented gate.
rule_id	A single, explicitly selected candidate ID. There is no default selection or preference based on candidate order.
rationale	A nonempty explanation of the selection, including any use of interpretability or parsimony among empirically equivalent maximizers.

calibration_scope	A nonempty named list describing the calibration design space, such as population assumptions, sample sizes, allocation, accrual, follow-up, event and censoring models, and horizons. Retained as supplied; the function does not establish validity outside this scope.
provenance	A nonempty named list recording calibration sources and versions. Must include <code>inference_conditioning = "support_valid"</code> , an explicit caller declaration that the supplied Type I error and coverage intervals use support-valid trials. Column names alone cannot independently verify this declaration. Include script/manuscript versions and data identifiers as appropriate. Retained as supplied.
simulation_settings	A nonempty named list recording the simulation settings, such as attempted replicates per setting, random seeds, generator and RNG configuration, testing level, and Monte Carlo interval method and confidence level. Retained as supplied; no simulations are run.
type1_tolerance	Type I error tolerance passed unchanged to <code>classify_calib_reliability()</code> . The manuscript value is 0.06.
coverage_tolerance	Coverage tolerance passed unchanged to <code>classify_calib_reliability()</code> . The manuscript value is 0.94.
support_tolerance	Support-availability tolerance passed unchanged to <code>classify_calib_reliability()</code> . It is applied to the <code>p_support</code> Monte Carlo confidence interval; the manuscript used 0.95.

Details

Reliability is recomputed with `classify_calib_reliability()` using `p_support_lower` and `p_support_upper`, summarized with `count_calib_reliability()`, and evaluated with `evaluate_calib_rules()` using `p_support`. Supplied reliability labels are not trusted. Numerical-only availability (`p_numeric`) is not used. Simulation denominators and interval calculations are unchanged. Each diagnostic row retains equal weight.

A candidate qualifies only when its conservative unsafe-pass rate (UPR) is defined and at most 0.05. This rate is the number of passing Unreliable plus Indeterminate settings divided by all passing settings. Among qualifying candidates, the confirmed adequate-pass rate (APR) must be maximized: passing Reliable settings divided by all Reliable settings. A candidate with no passing settings has undefined UPR and is ineligible; no Reliable settings means APR is undefined and freezing stops.

All qualifying maximizers and full Pass/Not-Pass patterns are preserved. Exact APR equality is used, as every candidate has the same Reliable-setting denominator. Interpretability or parsimony can justify an explicit choice among equivalent maximizing rules, but cannot override UPR or APR criteria. If maximizers have different patterns, freezing stops with their IDs and pattern IDs for review. No statistical tie-breaker is introduced.

The selected rule must include an absolute risk-count requirement. Rules without one (including Family A) remain in the candidate evaluation, but cannot be frozen for the current three-criterion

Stage 1 interface. No absolute threshold is invented. Thresholds are always read from the selected row; the manuscript values 0.10, 10, and 0.95 are not universal defaults.

Supply `frozen$thresholds` as the `thresholds` argument to `stage1_audit()` and `find_n_req()`. Freeze the rule before applying Stage 1 or Part II and keep the thresholds fixed throughout subsequent planning. This function does not run Part II, assess power, or implement a planning report.

Value

An S3 object of class `ahtau_support_rule`, containing:

rule_id, rule_family, family, rationale The explicit selection and explanation. `rule_family` is the selected family label; `family` retains the same label for compatibility with earlier frozen objects.

thresholds A list named `p_risk_min`, `n_risk_min`, and `p_support_min`, compatible with Stage 1 and Part II.

selected_rule, selected_performance The selected candidate row and its evaluated performance.

performance, pass_matrix Unmodified output from the existing evaluator for every candidate, including classification pattern IDs and equivalent rule IDs. Pass-matrix rows follow diagnostic row order.

diagnostic, reliability_counts The classified diagnostics and their overall and component reliability counts.

selection Eligible and maximizing rule IDs, the selected pattern ID, and equivalent rule IDs. Pattern IDs are labels, not ranks.

reliability_criteria Reliability tolerances, the conservative UPR limit, support-column mappings, and inference conditioning.

calibration_scope, provenance, simulation_settings Supplied design space and reproducibility metadata, retained unchanged.

Examples

```
# Small illustrative diagnostics, not a manuscript calibration reproduction.
diagnostic <- data.frame(
  scenario_id = c("design_a", "design_b"), tau = c(3, 4),
  min_mean_p_risk = c(0.25, 0.04), min_mean_n_risk = c(25, 4),
  p_support = c(0.99, 0.80),
  p_support_lower = c(0.97, 0.77), p_support_upper = c(1, 0.83),
  type1_one_lower = c(0.03, 0.07), type1_one_upper = c(0.05, 0.10),
  coverage_lower = c(0.95, 0.88), coverage_upper = c(0.98, 0.92)
)
diagnostic$total_sim_n <- 1000
diagnostic$n_support_valid <- c(990, 800)
rules <- data.frame(
  rule_id = c("relative_010", "relative_015"), rule_family = c("B", "B"),
  p_risk_threshold = c(0.10, 0.15), n_risk_threshold = c(8, 8),
  p_support_threshold = c(0.95, 0.95), use_absolute_n = c(TRUE, TRUE)
)
frozen <- freeze_support_rule(
  diagnostic, rules, rule_id = "relative_015",
```

```

    rationale = "Equivalent maximizing patterns; selected for interpretability.",
    calibration_scope = list(horizons = c(3, 4), design = "illustrative"),
    provenance = list(source = "example", inference_conditioning = "support_valid"),
    simulation_settings = list(source = "constructed diagnostics, no simulation")
  )
  frozen$thresholds
  frozen$selection$maximizing_rule_ids
  # stage1_audit(raw_results, thresholds = frozen$thresholds)
  # find_n_req(..., thresholds = frozen$thresholds)

```

planning_report

Create an Average-Hazard Trial Planning Report

Description

Assemble an auditable planning record from previously computed AHTauDesign results. No simulations, average-hazard estimation, horizon optimization, or sample-size searches are performed by this function.

Usage

```

planning_report(
  trial_spec,
  frozen_rule,
  stage1_result,
  stage2_result = NULL,
  stage2_evaluation = NULL,
  part2_results = NULL,
  simulation_diagnostics = list()
)

```

Arguments

trial_spec	Named list describing the reference trial. Include ‘tau_clin’ (required), and trial-specific design assumptions such as ‘scenario_id’, ‘N’, ‘A’, ‘F’, allocation, survival/censoring model, ‘clinical_window’, and ‘power_target’ (optional). When the Stage 1 audit contains multiple scenarios, ‘scenario_id’ is required.
frozen_rule	An object returned by ‘freeze_support_rule()’. The complete calibration evidence, scope, and provenance are preserved.
stage1_result	Output of ‘stage1_audit()’ or its ‘scenario_summary’.
stage2_result	Optional output of ‘stage2_select()’. Stage 2 is not entered when the clinical horizon passes Stage 1, even if a “Retained” selection result is supplied.
stage2_evaluation	Optional output of ‘stage2_evaluate()’ after a utility-driven alternative has been selected. Its simulations must be independent of the selection simulations; the caller must verify this.

part2_results Optional single output of ‘find_n_req()’, or a named or unnamed list of outputs over follow-up durations or horizons. Each raw and isotonic validation power is reported separately. Different horizons represent different estimands.

simulation_diagnostics Optional named list of additional audit details, such as Stage 1/Stage 2 seeds, models, software versions, and independent evaluation provenance.

Details

The supplied frozen thresholds are used only for a consistency check against Stage 1 decisions; no classification is changed. Information support and overall power remain distinct. A passing clinical horizon is retained, and Stage 2 is reported only following support failure. Missing optional analyses are labeled as not supplied, never simulated.

The returned object retains the original computational results. Use ‘print(report)’ for a readable console summary, ‘saveRDS(report, "report.rds")’ to save all evidence, or ‘writeLines(capture.output(print(report)), "report.txt")’ for a text file.

Value

An object of class ‘ahtau_planning_report’ with the trial specification, frozen rule, supplied analyses, Stage 1/Stage 2 decision, Part II requirement table, simulation diagnostics, and time of creation.

Examples

```
## Not run:
# After frozen <- freeze_support_rule(...) and
# audit <- stage1_audit(..., thresholds = frozen$thresholds):
report <- planning_report(
  trial_spec = list(scenario_id = "reference", N = 120, A = 48,
                    F = 18, tau_clin = 36, power_target = 0.80),
  frozen_rule = frozen, stage1_result = audit
)
print(report)
saveRDS(report, "report.rds")

## End(Not run)
```

```
print.ahtau_planning_report
  Print an Average-Hazard Planning Report
```

Description

Print an Average-Hazard Planning Report

Usage

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

Arguments

x An 'ahtau_planning_report' object.
... Unused.

Value

The report, invisibly.

simulate_calibration_null

Simulate Null Trials for Average-Hazard Support Calibration

Description

Simulate a prespecified set of null trial designs, analyze every candidate horizon on the same generated trial within each replicate, and summarize repeated-sampling support and inferential performance. The trial generator must represent the null hypothesis of equal average hazards (log AH ratio zero) at the candidate horizons; this scientific assumption is supplied by the caller, not checked by the function.

Usage

```
simulate_calibration_null(
  design_space,
  trial_generator,
  tau_grid,
  n_sim,
  seed,
  trial_args = list(),
  alpha = 0.05,
  mc_conf_level = 0.95,
  rules = NULL,
  reliability_criteria = NULL
)
```

Arguments

design_space A nonempty data frame with one row per calibration scenario and a unique character `scenario_id`. All other columns are passed as named arguments to `trial_generator`. Scalar design columns are retained in the replicate results and diagnostic table. List columns may be used for scenario-specific vector parameters and are retained in `design_space`, but not copied into the diagnostic table.

trial_generator	A function returning one observed, two-arm trial as a data frame with numeric arm (0 = control, 1 = treatment), numeric nonnegative observed time X, and 0/1 event indicator status. Both arms must be present. The generator should apply the specified accrual, event-time, and censoring mechanisms under the null. Generator errors stop the run rather than silently reducing simulation denominators.
tau_grid	A nonempty vector of distinct, finite, positive horizons. Every horizon is evaluated on each simulated trial.
n_sim	Positive whole number of attempted trials per scenario .
seed	A required nonnegative whole-number random seed. The caller's RNG kind and .Random.seed are restored on exit, including on error.
trial_args	An optional named list of generator arguments shared by all scenarios. Its argument names must not overlap design_space columns other than scenario_id.
alpha	One-sided test level, passed unchanged to evaluate_tau_grid(). Negative log AH ratios favor treatment.
mc_conf_level	Confidence level of the exact binomial Monte Carlo intervals produced by summarize_calibration_summary . The Wald intervals whose coverage is measured always have 95 percent confidence.
rules	Optional candidate-rule data frame for descriptive comparison, including a p_support_threshold availability gate, using the existing evaluate_calib_rules() interface. Rules are never automatically selected or frozen. At least two design-horizon settings and two candidate rules are required by this simulation interface.
reliability_criteria	A named list supplying type1_tolerance, coverage_tolerance, and support_tolerance when rules is given. These are caller-specified calibration criteria, not universal defaults. support_tolerance is applied to the Monte Carlo interval for support availability (p_support), not numerical-only validity. Supply rules and reliability_criteria together, or neither.

Details

AH estimates, standard errors, one-sided rejections, and observed risk sets are computed by the existing evaluate_tau_grid() function; it is not modified. For numerically valid results, the additional covered indicator is calculated from the two-sided 95 percent Wald interval $\theta_{\text{hat}} \pm qnorm(0.975) * SE_{\theta}$ checking inclusion of zero.

Numerical validity requires finite θ_{hat} and SE_{θ} , with $SE_{\theta} > 0$. Support validity additionally requires $n_{\text{risk}_C} > 0$ and $n_{\text{risk}_T} > 0$. The diagnostic summary uses **all attempted trials** for arm-specific mean risk sets and support availability; Type I error and Wald coverage are calculated **only among support-valid trials**. Exact binomial Monte Carlo intervals retain those same denominators. A failed AH fit is retained as an attempted analysis with unavailable inference.

Optional reliability and candidate-rule comparisons explicitly use the p_{support} point estimate and its Monte Carlo interval, not p_{numeric} . The output's diagnostic is directly compatible with [freeze_support_rule\(\)](#), which must be called separately with explicit candidate selection and provenance. Neither Stage 1 thresholds nor a frozen support rule are supplied by this simulation function.

Value

A list containing:

raw_results One row per attempted scenario, replicate, and horizon. Includes `ci_lower`, `ci_upper`, `covered`, `numeric_valid`, `tail_support`, and `support_valid` in addition to Stage 1 outputs.

diagnostic The result of `summarize_calibration_support()`, with counts, proportions, support-conditional inferential metrics, and exact Monte Carlo intervals.

design_space, settings The trial design space and simulation specification, including seed, RNG kind, horizon grid, and levels.

classified_diagnostic, reliability_counts, rule_evaluation Only when rules and `reliability_criteria` are provided: the existing reliability classifier, reliability counts, and candidate-rule performance/pass matrix.

See Also

[summarize_calibration_support\(\)](#), [freeze_support_rule\(\)](#)

Examples

```
null_trial <- function(N, A, F, rate) {
  arm <- rep(c(0, 1), each = N / 2)
  entry <- stats::runif(N, 0, A)
  event_time <- stats::rexp(N, rate)
  censor_time <- A + F - entry
  data.frame(arm = arm, X = pmin(event_time, censor_time),
             status = as.numeric(event_time <= censor_time))
}
null_designs <- data.frame(
  scenario_id = c("small", "large"), N = c(40, 80),
  A = 24, F = c(12, 24), rate = 0.03
)
## Not run:
sim <- simulate_calibration_null(
  design_space = null_designs, trial_generator = null_trial,
  tau_grid = c(12, 18), n_sim = 1000, seed = 1234
)
sim$diagnostic
# Supply trial-specific candidate rules and use freeze_support_rule()
# separately, after evaluating candidate-rule performance.

## End(Not run)
```

simulate_design

*Simulate Planning Trials Across Average-Hazard Horizons***Description**

Generate repeated two-arm trials using a caller-supplied trial generator and evaluate all prespecified average-hazard horizons on the same observed trial within each replicate. Return raw results for `stage1_audit()`.

Usage

```
simulate_design(
  trial_generator,
  n_sim,
  tau_grid,
  trial_args = list(),
  scenario_id = "reference",
  seed,
  alpha = 0.05
)
```

Arguments

<code>trial_generator</code>	A function returning one observed trial as a data frame with numeric arm coded 0 (control) or 1 (treatment), observed time X , and event indicator status coded 0/1. Both arms must be represented.
<code>n_sim</code>	Positive whole number of attempted simulation replicates.
<code>tau_grid</code>	Nonempty numeric vector of distinct, finite, positive horizons.
<code>trial_args</code>	Named list of arguments passed to <code>trial_generator</code> on each replicate. If N is supplied, the generated trial must have N rows.
<code>scenario_id</code>	One nonempty character label identifying the design.
<code>seed</code>	Required nonnegative whole-number random seed, no greater than <code>.Machine\$integer.max</code> . The caller's RNG state and kind are restored after the function returns or errors.
<code>alpha</code>	One-sided significance level strictly between 0 and 1.

Details

Unlike `simulate_calibration_null()`, this function is for planning scenarios under the caller's working treatment-effect model; it does not calculate calibration reliability, freeze support thresholds, or select a horizon.

For each replicate, the generator runs **once** and its observed data are passed to the package-internal `evaluate_tau_grid()` for every horizon. The underlying AH estimator, test direction, and risk-set calculations are unchanged. An AH fit that cannot be estimated remains in the output with missing

inference values; it is not silently excluded. Invalid trial data or generator errors stop the simulation instead of dropping a replicate.

To perform independent Stage 2 evaluation, call `simulate_design()` again with a different seed and freshly generated trials. The generator should not reset the random seed internally. Independence depends on the supplied data-generating process and cannot be guaranteed by different seeds alone.

Value

A data frame containing one row per attempted replicate and horizon. Columns include `scenario_id`, `sim_id`, `tau`, `theta_hat`, `SE_theta`, `n_risk_C`, `n_risk_T`, `p_risk_C`, `p_risk_T`, and `reject`. Pass this data frame directly to `stage1_audit(raw_results, thresholds)`.

See Also

`stage1_audit()`, `simulate_calibration_null()`

Examples

```
example_trial <- function(N, A, F, rate_C, rate_T) {
  arm <- rep(c(0, 1), each = N / 2)
  entry <- stats::runif(N, 0, A)
  event <- stats::rexp(N, rate = ifelse(arm == 0, rate_C, rate_T))
  censor <- A + F - entry
  data.frame(arm = arm, X = pmin(event, censor),
             status = as.numeric(event <= censor))
}
raw <- simulate_design(
  trial_generator = example_trial, n_sim = 2,
  tau_grid = c(12, 24), seed = 123,
  trial_args = list(N = 60, A = 48, F = 12,
                    rate_C = -log(0.60) / 36,
                    rate_T = -log(0.72) / 36)
)
head(raw)
# audit <- stage1_audit(raw, thresholds = frozen$thresholds)
```

stage1_audit

Stage 1 Information Support Audit

Description

Evaluate whether candidate AH horizons satisfy a prespecified information support rule.

Usage

```
stage1_audit(raw_results, thresholds)
```

Arguments

- raw_results** A data frame containing results from all attempted simulation replicates.
- thresholds** A named list containing `p_risk_min`, `n_risk_min`, and `p_support_min`.

Value

A list containing `replicate_results` and `scenario_summary`.

stage2_evaluate	<i>Independent Evaluation of a Stage 2 Alternative</i>
-----------------	--

Description

Compare the clinically preferred horizon and the selected planning alternative using independent evaluation results.

Usage

```
stage2_evaluate(selection_result, eval_summary)
```

Arguments

- selection_result** Result returned by `stage2_select()`.
- eval_summary** A Stage 1 summary data frame obtained from independent evaluation simulations, containing both the clinical and selected horizons.

Details

This function summarizes independently simulated results. It does not generate new simulations.

Value

A list containing:

eval_summary Independent evaluation results for the clinical and selected horizons.

power_difference Overall power at the selected horizon minus overall power at the clinical horizon. The difference may be positive or negative.

stage2_select

*Stage 2 Conditional Horizon Selection***Description**

Retain the clinical horizon when Stage 1 is passed. Otherwise, identify a supported alternative within the prespecified clinically acceptable window using the manuscript's utility and tie-breaking rules.

Usage

```
stage2_select(stage1_summary, tau_clin, clinical_window)
```

Arguments

`stage1_summary` A Stage 1 summary data frame for one trial design scenario.

`tau_clin` Clinically preferred AH horizon.

`clinical_window` Numeric vector of length two specifying the clinically acceptable horizon range.

Value

A list containing `candidate_summary` and `final_decision_summary`.

Index

`find_n_req`, [2](#)
`find_n_req()`, [6](#)
`freeze_support_rule`, [3](#)
`freeze_support_rule()`, [10](#), [11](#)

`planning_report`, [7](#)
`print.ahtau_planning_report`, [8](#)

`simulate_calibration_null`, [9](#)
`simulate_calibration_null()`, [12](#), [13](#)
`simulate_design`, [12](#)
`stage1_audit`, [13](#)
`stage1_audit()`, [6](#), [12](#), [13](#)
`stage2_evaluate`, [14](#)
`stage2_select`, [15](#)
`summarize_calibration_support()`, [10](#), [11](#)