

Package ‘tout’

May 8, 2026

Title Optimal Sample Size and Progression Criteria for Three-Outcome Trials

Version 1.0.3

Description Find the optimal decision rules (AKA progression criteria) and sample size for clinical trials with three (stop/pause/go) outcomes. Both binary and continuous endpoints can be accommodated, as can cases where an adjustment is planned following a pause outcome. For more details see Wilson et al. (2024) <[doi:10.1186/s12874-024-02351-x](https://doi.org/10.1186/s12874-024-02351-x)>.

License MIT + file LICENSE

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

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

RoxygenNote 7.3.3

URL <https://github.com/DTWilson/tout>, <https://dtwilson.github.io/tout/>

BugReports <https://github.com/DTWilson/tout/issues>

NeedsCompilation no

Author Duncan Wilson [aut, cre, cph] (ORCID:
<<https://orcid.org/0000-0001-7949-8718>>)

Maintainer Duncan Wilson <d.t.wilson@leeds.ac.uk>

Repository CRAN

Date/Publication 2025-11-07 14:00:02 UTC

Contents

plot.tout	2
print.tout	2
tout_design	3
Index	5

`plot.tout`*Plot sampling distributions of three-outcome designs*

Description

Takes an object of class `tout` and plots sampling distributions under the null and alternative hypotheses, highlighting which portions correspond to stop, pause, and go outcomes.

Usage

```
## S3 method for class 'tout'  
plot(x, ...)
```

Arguments

`x` object of class `tout` as produced by `tout_design()`.
`...` further arguments passed to or from other methods.

Value

no return value, called for side effects.

`print.tout`*Print a tout object*

Description

The default print method for a `tout` object.

Usage

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

Arguments

`x` object of class `tout` as produced by `tout_design()`.
`...` further arguments passed to or from other methods.

Value

no return value, called for side effects.

 tout_design

Find optimal sample size and progression criteria

Description

Given a null and alternative hypothesis, this function finds the lowest sample size such that a design with optimal progression criteria (as determined by the function `opt_pc`) satisfies upper constraints on three operating characteristics.

Usage

```
tout_design(
  rho_0,
  rho_1,
  alpha_nom,
  beta_nom,
  gamma_nom = 1,
  eta_0 = 0.5,
  eta_1 = eta_0,
  tau = c(0, 0),
  max_n = NULL,
  n = NULL,
  x = NULL,
  sigma = NULL
)
```

Arguments

<code>rho_0</code>	null hypothesis.
<code>rho_1</code>	alternative hypothesis.
<code>alpha_nom</code>	nominal upper constraint on alpha.
<code>beta_nom</code>	nominal upper constraint on beta.
<code>gamma_nom</code>	nominal upper constraint on gamma. Defaults to 1.
<code>eta_0</code>	probability of an incorrect decision under the null hypothesis after an intermediate result. Defaults to 0.5.
<code>eta_1</code>	probability of an incorrect decision under the alternative hypothesis after an intermediate result. Defaults to <code>eta_0</code> .
<code>tau</code>	two element vector denoting lower and upper limits of the effect of adjustment.
<code>max_n</code>	optional upper limit to use in search over sample sizes.
<code>n</code>	optional sample size (optimised if left unspecified).
<code>x</code>	optional vector of decision thresholds (optimised if left unspecified).
<code>sigma</code>	standard deviation of outcome. If left unspecified, a binary outcome is assumed.

Value

An object of class `tout`, which is a list containing the following components:

<code>valid</code>	boolean indicating if the nominal constraints are met.
<code>n</code>	sample size.
<code>thresholds</code>	numeric vector of the two decision thresholds.
<code>alpha</code>	attained value of operating characteristic alpha.
<code>beta</code>	attained value of operating characteristic beta.
<code>gamma</code>	attained value of operating characteristic gamma.

Examples

```
rho_0 <- 0.5
rho_1 <- 0.7
alpha_nom <- 0.05
beta_nom <- 0.2

tout_design(rho_0, rho_1, alpha_nom, beta_nom)

# Allowing for adjustment effects:

tout_design(rho_0, rho_1, alpha_nom, beta_nom, tau = c(0.08, 0.12))

# Allowing for different error probabilities following a pause decision

tout_design(rho_0, rho_1, alpha_nom, beta_nom, eta_0 = 0.3)

# Designs for continuous outcomes:

tout_design(rho_0 = 0, rho_1 = 0.4, alpha_nom, beta_nom, sigma = 1)
```

Index

plot.tout, 2
print.tout, 2
tout_design, 3