Package ‘escalation’

October 18, 2020

Type Package
Title Modular Approach to Dose Finding Clinical Trials
Version 0.1.4
Date 2020-10-16
Maintainer Kristian Brock <kristian.brock@gmail.com>

Description Methods for working with dose-finding clinical trials. We provide implementations of many dose-finding clinical trial designs, including the continual reassessment method (CRM) by O'Quigley et al. (1990) <doi:10.2307/2531628>, the toxicity probability interval (TPI) design by Ji et al. (2007) <doi:10.1177/1740774507079442>, the modified TPI (mTPI) design by Ji et al. (2010) <doi:10.1177/1740774510382799>, the Bayesian optimal interval design (BOIN) by Liu & Yuan (2015) <doi:10.1111/rssc.12089>, EffTox by Thall & Cook (2004) <doi:10.1111/j.0006-341X.2004.00218.x>; the design of Wages & Tait (2015) <doi:10.1080/10543406.2014.920873>, and the 3+3 described by Korn et al. (1994) <doi:10.1002/sim.4780131802>. All designs are implemented with a common interface. We also offer optional additional classes to tailor the behaviour of all designs, including avoiding skipping doses, stopping after n patients have been treated at the recommended dose, stopping when a toxicity condition is met, or demanding that n patients are treated before stopping is allowed. By daisy-chaining together these classes using the pipe operator from 'magrittr', it is simple to tailor the behaviour of a dose-finding design so it behaves how the trialist wants. Having provided a flexible interface for specifying designs, we then provide functions to run simulations and calculate dose-paths for future cohorts of patients.

License GPL (>= 3)
Encoding UTF-8
LazyData true
Depends magrittr
Imports dplyr, tidyr (>= 1.0), tidyselect, stringr, purrr, tibble, gtools, dfcrm, BOIN, trialr (>= 0.1.5), DiagrammeR, RColorBrewer, viridis, binom
RoxygenNote 7.1.1
R topics documented:

Suggests testthat, knitr, rmarkdown, ggplot2, covr

VignetteBuilder knitr

NeedsCompilation no

Author Kristian Brock [aut, cre] (<https://orcid.org/0000-0002-3921-0166>)

Repository CRAN

Date/Publication 2020-10-18 21:40:06 UTC

R topics documented:

- `escalation-package` ........................................... 4
- `as_tibble.dose_paths` ........................................ 4
- `calculate_probabilities` ..................................... 5
- `cohort` .......................................................... 6
- `cohorts_of_n` ................................................... 6
- `continue` ........................................................ 7
- `crystallised_dose_paths` ..................................... 8
- `demand_n_at_dose` ............................................. 9
- `doint_skip_doses` ............................................. 10
- `doses_given` ................................................... 11
- `dose_admissible` ............................................. 12
- `dose_indices` .................................................. 13
- `dose_paths` .................................................... 13
- `dose_paths_function` ......................................... 14
- `eff` ................................................................. 14
- `eff_at_dose` .................................................... 15
- `eff_limit` ........................................................ 16
- `empiric_eff_rate` ............................................. 17
- `empiric_tox_rate` ............................................ 17
- `enforce_three_plus_three` ................................... 18
- `fit` ................................................................. 19
- `follow_path` .................................................... 19
- `get_boin` ........................................................ 20
- `get_dfcrm` ....................................................... 21
- `get_dose_paths` ............................................... 23
- `get_empiric_crm_skeleton_weights` ......................... 24
- `get_mtpi` ....................................................... 24
- `get_random_selector` ......................................... 26
- `get_three_plus_three` ....................................... 27
- `get_tpi` .......................................................... 28
- `get_trialr_crm` ............................................... 30
- `get_trialr_efftox` ............................................ 32
- `get_trialr_nbg` ............................................... 33
- `get_wages_and_tait` .......................................... 35
- `graph_paths` ................................................... 37
- `is_randomising` ............................................... 38
- `mean_prob_eff` ................................................ 38
mean_prob_tox ............................................................. 39
median_prob_eff ......................................................... 40
median_prob_tox ......................................................... 41
model_frame .............................................................. 41
num_cohort_outcomes .................................................. 42
num_doses ................................................................. 43
num_dose_path_nodes ................................................... 43
num_eff ................................................................. 44
num_patients ........................................................... 45
num_tox ................................................................. 45
n_at_dose ............................................................... 46
n_at_recommended_dose ............................................... 47
parse_phase1_2_outcomes .............................................. 47
parse_phase1_outcomes ................................................ 48
phase1_2_outcomes_to_cohorts ....................................... 50
phase1_outcomes_to_cohorts .......................................... 51
prob_administer .......................................................... 52
prob_eff_quantile ........................................................ 53
prob_recommend ........................................................ 54
prob_tox_exceeds ........................................................ 54
prob_tox_quantile ....................................................... 55
prob_tox_samples ...................................................... 56
recommended_dose ..................................................... 57
selector ................................................................. 58
selector_factory ........................................................ 61
select_boin_mtd ........................................................ 63
select_dose_by_cibp .................................................... 64
simulate_trials .......................................................... 65
simulations .............................................................. 68
simulation_function .................................................... 70
spread_paths ............................................................ 70
stop_at_n ............................................................... 71
stop_when_n_at_dose .................................................... 72
stop_when_too_toxic ..................................................... 73
stop_when_tox_ci_covered ............................................. 75
supports_sampling ...................................................... 77
three_plus_three ....................................................... 77
tox ................................................................. 78
tox_at_dose ............................................................. 79
tox_limit ............................................................... 80
tox_target ............................................................. 81
trial_duration .......................................................... 81
try_rescue_dose ........................................................ 82

Index 84
The `escalation` package.

Description

escalation provides methods for working with dose-finding clinical trials. We provide implementations of many dose-finding clinical trial designs, including the continual reassessment method (CRM) by O’Quigley et al. (1990) <doi:10.2307/2531628>, the toxicity probability interval (TPI) design by Ji et al. (2007) <doi:10.1177/1740774507079442>, the modified TPI (mTPI) design by Ji et al. (2010) <doi:10.1177/1740774510382799>, the Bayesian optimal interval design (BOIN) by Liu & Yuan (2015) <doi:10.1111/rssc.12089>, EffTox by Thall & Cook (2004) <doi:10.1111/j.0006-341X.2004.00218.x>, the design of Wages & Tait (2015) <doi:10.1080/10543406.2014.920873>, and the 3+3 described by Korn et al. (1994) <doi:10.1002/sim.4780131802>. All designs are implemented with a common interface. We also offer optional additional classes to tailor the behaviour of all designs, including avoiding skipping doses, stopping after n patients have been treated at the recommended dose, stopping when a toxicity condition is met, or demanding that n patients are treated before stopping is allowed. By daisy-chaining together these classes using the pipe operator from ‘magrittr’, it is simple to tailor the behaviour of a dose-finding design so it behaves how the trialist wants. Having provided a flexible interface for specifying designs, we then provide functions to run simulations and calculate dose-paths for future cohorts of patients.

as_tibble.dose_paths

Cast dose_paths object to tibble.

Description

Cast `dose_paths` object to `tibble`.

Usage

```r
## S3 method for class 'dose_paths'
as_tibble(x, ...)
```

Arguments

- `x` Object of class `dose_finding_paths`.
- `...` Extra args passed onwards.

Value

Object of class `tibble`
**calculate_probabilities**

*Calculate dose-path probabilities*

**Description**

Crystallise a set of `dose_paths` with probabilities to calculate how likely each path is. Once probabilised in this way, the probabilities of the terminal nodes in this set of paths will sum to 1. This allows users to calculate operating characteristics.

**Usage**

```r
calculate_probabilities(dose_paths, true_prob_tox, true_prob_eff = NULL, ...)
```

**Arguments**

- `dose_paths` Object of type `dose_paths`
- `true_prob_tox` Numeric vector, true probability of toxicity.
- `true_prob_eff` vector of true efficacy probabilities, optionally NULL if efficacy not analysed.
- `...` Extra parameters

**See Also**

- `dose_paths`

**Examples**

```r
# Phase 1 example.
# Calculate dose paths for the first three cohorts in a 3+3 trial of 5 doses:
paths <- get_three_plus_three(num_doses = 5) %>%
  get_dose_paths(cohort_sizes = c(3, 3, 3))

# Set the true probabilities of toxicity
ture_prob_tox <- c(0.12, 0.27, 0.44, 0.53, 0.57)
# And calculate exact operating performance
x <- paths %>% calculate_probabilities(true_prob_tox)
prob_recommend(x)

# Phase 1/2 example.
prob_select = c(0.1, 0.3, 0.5, 0.07, 0.03)
selector_factory <- get_random_selector(prob_select = prob_select,
supports_efficacy = TRUE)
paths <- selector_factory %>% get_dose_paths(cohort_sizes = c(2, 2))
true_prob_eff <- c(0.27, 0.35, 0.41, 0.44, 0.45)
x <- paths %>% calculate_probabilities(true_prob_tox = true_prob_tox,
true_prob_eff = true_prob_eff)
prob_recommend(x)
```
cohort

Cohort numbers of evaluated patients.

Description

Get a vector of integers that reflect the cohorts to which the evaluated patients belong.

Usage

cohort(x, ...)

Arguments

x

Object of type selector.

...

Extra args are passed onwards.

Value

an integer vector

Examples

skeleton <- c(0.05, 0.1, 0.25, 0.4, 0.6)
target <- 0.25
model <- get_dfcrm(skeleton = skeleton, target = target)
fit <- model %>% fit('1NNN 2NTN')
fit %>% cohort()

cohorts_of_n

Sample times between patient arrivals using the exponential distribution.

Description

Sample times between patient arrivals using the exponential distribution.

Usage

cohorts_of_n(n = 3, mean_time_delta = 1)

Arguments

n

integer, sample arrival times for this many patients.

mean_time_delta

the average gap between patient arrival times. I.e. the reciprocal of the rate parameter in an Exponential distribution.
Value
data.frame with column time_delta containing durations of time between patient arrivals.

Examples
cohorts_of_n()
cohorts_of_n(n = 10, mean_time_delta = 5)

Description
Should this dose-finding experiment continue? Or have circumstances prevailed that dictate this
trial should stop? This method is critical to the automatic calculation of statistical operating char-
acteristics and dose-pathways. You add stopping behaviours to designs using calls like stop_at_n
and stop_when_too_toxic.

Usage
continue(x, ...)

Arguments
x Object of type selector.

Value
logical

Examples
skeleton <- c(0.05, 0.1, 0.25, 0.4, 0.6)
target <- 0.25
model1 <- get_dfcrm(skeleton = skeleton, target = target)
fit1 <- model1 %>% fit('1NNN 2NTN')
fit1 %>% continue()

model2 <- get_dfcrm(skeleton = skeleton, target = target) %>%
  stop_at_n(n = 6)
fit2 <- model2 %>% fit('1NNN 2NTN')
fit2 %>% continue()
crystallised_dose_paths

Dose-paths with probabilities attached.

Description

dose_paths reflect all possible paths a dose-finding trial may take. When the probability of those
paths is calculated using an assumed set of true dose-event probabilities, in this package those paths
are said to be crystallised. Once crystallised, operating characteristics can be calculated.

Usage

crystallised_dose_paths(
  dose_paths,
  true_prob_tox,
  true_prob_eff = NULL,
  terminal_nodes
)

Arguments

dose_paths Object of type dose_paths
true_prob_tox vector of toxicity probabilities at doses 1..n
true_prob_eff vector of efficacy probabilities at doses 1..n, optionally NULL if efficacy not evaluated.
terminal_nodes tibble of terminal nodes on the dose-paths

Value

An object of type crystallised_dose_paths

Examples

# Calculate dose paths for the first three cohorts in a 3+3 trial of 5 doses:
paths <- get_three_plus_three(num_doses = 5) %>%
geet_dose_paths(cohort_sizes = c(3, 3, 3))

# Set the true probabilities of toxicity
ttrue_prob_tox <- c(0.12, 0.27, 0.44, 0.53, 0.57)
# Crystallise the paths with the probabilities of toxicity
x <- paths %>% calculate_probabilities(true_prob_tox)
# And then examine, for example, the probabilities of recommending each dose
# at the terminal nodes of these paths:
prob_recommend(x)
**demand_n_at_dose**

**Description**

This method continues a dose-finding trial until there are \( n \) patients at a dose. Once that condition is met, it delegates stopping responsibility to its parent dose selector, whatever that might be. This class is greedy in that it meets its own needs before asking any other selectors in a chain what they want. Thus, different behaviours may be achieved by nesting dose selectors in different orders. See examples.

**Usage**

\[
\text{demand_n_at_dose(parent_selector_factory, n, dose)}
\]

**Arguments**

- **parent_selector_factory**
  - Object of type `selector_factory`.
- **n**
  - Continue at least until there are \( n \) at a dose.
- **dose**
  - 'any' to continue until there are \( n \) at any dose; 'recommended' to continue until there are \( n \) at the recommended dose; or an integer to continue until there are \( n \) at a particular dose-level.

**Value**

An object of type `selector_factory` that can fit a dose-finding model to outcomes.

**Examples**

```r
skeleton <- c(0.05, 0.1, 0.25, 0.4, 0.6)
target <- 0.25

# This model will demand 9 at any dose before it countenances stopping.
model1 <- get_dfcrm(skeleton = skeleton, target = target) %>%
  demand_n_at_dose(n = 9, dose = 'any')

# This model will recommend continuing:
model1 %>% fit(1NNT 1NNN 2TNN 2NNN) %>% continue()
# It tells you to continue because there is no selector considering when you should stop - dfcrm implements no stopping rule by default.

# In contrast, we can add a stopping selector to discern the behaviour of
# demand_n_at_dose. We will demand 9 are seen at the recommended dose before
# stopping is permitted in model3:
model2 <- get_dfcrm(skeleton = skeleton, target = target) %>%
  stop_at_n(n = 12)
model3 <- get_dfcrm(skeleton = skeleton, target = target) %>%
```


stop_at(n = 12) %>%
demand_n_at_dose(n = 9, dose = 'recommended')

# This model advocates stopping because 12 patients are seen in total:
model2 %>% fit('1NNN 1NNN 2TNN 2NNN') %>% continue()
# But this model advocates continuing because 9 patients have not been seen
# at any dose yet:
model3 %>% fit('1NNN 1NNN 2TNN 2NNN') %>% continue()
# This shows how demand_n_at_dose overrides stopping behaviours that come
# before it in the daisychain.

# Once 9 are seen at the recommended dose, the decision to stop is made:
fit <- model3 %>% fit('1NNN 1NNN 2TNN 2NNN 2TTN')
fit %>% continue()
fit %>% recommended_dose()

dont_skip_doses

Prevent skipping of doses.

Description

This method optionally prevents dose selectors from skipping doses when escalating and / or deescalating. The default is that skipping when escalating is prevented but skipping when deescalating is permitted, but both of these behaviours can be altered.

Usage

dont_skip_doses(
    parent_selector_factory,
    when_escalating = TRUE,
    when_deescalating = FALSE
)

Arguments

parent_selector_factory
    Object of type selector_factory.

when_escalating
    TRUE to prevent skipping when attempting to escalate.

when_deescalating
    TRUE to prevent skipping when attempting to deescalate.

Value

an object of type selector_factory that can fit a dose-finding model to outcomes.
Examples

```r
dose <- c(0.05, 0.1, 0.25, 0.4, 0.6)
target <- 0.25
model1 <- get_dfcrm(skeleton = dose, target = target) %>%
  dont_skip_doses()
fit1 <- model1 %>% fit('1NNN')

model2 <- get_dfcrm(skeleton = dose, target = target)
fit2 <- model2 %>% fit('1NNN')

# fit1 will not skip doses
fit1 %>% recommended_dose()
# But fit2 will:
fit2 %>% recommended_dose()

# Similar demonstration for de-escalation
model1 <- get_dfcrm(skeleton = dose, target = target) %>%
  dont_skip_doses(when_deescalating = TRUE)
fit1 <- model1 %>% fit('1NNN 2N 3TTT')

model2 <- get_dfcrm(skeleton = dose, target = target)
fit2 <- model2 %>% fit('1NNN 2N 3TTT')

# fit1 will not skip doses
fit1 %>% recommended_dose()
# But fit2 will:
fit2 %>% recommended_dose()
```

doses_given

_Doses given to patients._

**Description**

Get a vector of the dose-levels that have been administered to patients.

**Usage**

doses_given(x, ...)

**Arguments**

- `x` Object of type `selector`.
- `...` Extra args are passed onwards.

**Value**

an integer vector
Examples

```r
elements <- c(0.05, 0.1, 0.25, 0.4, 0.6)  
threshold <- 0.25  
model <- get_dfcrm(skeleton = elements, target = threshold)  
fit <- model %>% fit('1NNN 2NTN')  
fit %>% doses_given()
```

---

dose_admissible  
Is each dose admissible?

Description

Get a vector of logical values reflecting whether each dose is admissible. Admissibility is defined in different ways for different models, and may not be defined at all in some models. For instance, in the TPI method, doses are inadmissible when the posterior probability is high that the toxicity rate exceeds the target value. In contrast, admissibility is not defined in the general CRM model (but it can be added with auxiliary classes). In this latter case, doses are implicitly considered to be admissible, by default.

Usage

```
dose_admissible(x, ...)
```

Arguments

- `x`  
  Object of class `selector`
- `...`  
  arguments passed to other methods

Value

A logical vector

Examples

```r
outcomes <- '1NNN 2TTT'

# TPI example. This method defines admissibility.
fit1 <- get_tpi(num_doses = 5, target = 0.3, k1 = 1, k2 = 1.5,  
exclusion_certainty = 0.95) %>%
  fit(outcomes)  
fit1 %>% dose_admissible()

# Ordinary CRM example with no admissibility function.
skeleton <- c(0.05, 0.1, 0.25, 0.4, 0.6)  
threshold <- 0.25  
fit2 <- get_dfcrm(skeleton = skeleton, target = threshold) %>%
  fit(outcomes)  
fit2 %>% dose_admissible()
```
# Same CRM example with added admissibility function
fit3 <- get_dfcrm(skeleton = skeleton, target = target) %>%
  stop_when_too_toxic(dose = 1, tox_threshold = target, confidence = 0.8) %>%
  fit(outcomes)
fit3 %>% dose_admissible()

dose_indices

Dose indices

Description

Get the integers from 1 to the number of doses under investigation.

Usage

dose_indices(x, ...)

Arguments

x Object of type selector.
...
Extra args are passed onwards.

Value

an integer vector

Examples

skeleton <- c(0.05, 0.1, 0.25, 0.4, 0.6)
target <- 0.25
model <- get_dfcrm(skeleton = skeleton, target = target)
fit <- model %>% fit('1NNN 2NTN')
fit %>% dose_indices()

dose_paths

Dose pathways

Description

A dose-escalation design exists to select doses in response to observed outcomes. The entire space of possible responses can be calculated to show the behaviour of a design in response to all feasible outcomes. The get_dose_paths function performs that task and returns an instance of this object.

Usage

dose_paths()
See Also

selector

Examples

# Calculate dose-paths for the 3+3 design:
paths <- get_three_plus_three(num_doses = 5) %>%
  get_dose_paths(cohort_sizes = c(3, 3))

dose_paths_function

Get function for calculating dose pathways.

Description

This function does not need to be called by users. It is used internally.

Usage

dose_paths_function(selector_factory)

Arguments

selector_factory

Object of type selector_factory.

Value

A function.

eff

Binary efficacy outcomes.

Description

Get a vector of the binary efficacy outcomes for evaluated patients.

Usage

eff(x, ...)

Arguments

x

Object of type selector.

... Extra args are passed onwards.
Value

an integer vector

Examples

\[
\text{prob_select} = c(0.1, 0.3, 0.5, 0.07, 0.03) \\
\text{model} <- \text{get_random_selector}(\text{prob_select} = \text{prob_select}, \\
\phantom{\text{model} <- \text{get_random_selector}(\text{prob_select} = \text{prob_select},} \\
\phantom{\text{model} <- \text{get_random_selector}(\text{prob_select} = \text{prob_select}, \phantom{\text{supports_efficacy = TRUE})}} \\
\text{x} <- \text{model} \%\% \text{fit('1NTN 2EN 5BB')} \\
\text{eff(x)}
\]

---

**eff_at_dose**  
*Number of toxicities seen at each dose.*

Description

Get the number of toxicities seen at each dose under investigation.

Usage

\[
\text{eff_at_dose}(x, \ldots)
\]

Arguments

- **x**: Object of class *selector*
- **\ldots**: arguments passed to other methods

Value

an integer vector

Examples

\[
\text{prob_select} = c(0.1, 0.3, 0.5, 0.07, 0.03) \\
\text{model} <- \text{get_random_selector}(\text{prob_select} = \text{prob_select}, \\
\phantom{\text{model} <- \text{get_random_selector}(\text{prob_select} = \text{prob_select},} \\
\phantom{\text{model} <- \text{get_random_selector}(\text{prob_select} = \text{prob_select}, \phantom{\text{supports_efficacy = TRUE})}} \\
\text{x} <- \text{model} \%\% \text{fit('1NTN 2EN 5BB')} \\
\text{eff_at_dose(x)}
\]
**eff_limit**  
**Efficacy rate limit**

**Description**

Get the minimum permissible efficacy rate, if supported. NULL if not.

**Usage**

`eff_limit(x, ...)`

**Arguments**

- `x` Object of type `selector`.
- `...` Extra args are passed onwards.

**Value**

numeric

**Examples**

```r
efftox_priors <- trialr::efftox_priors  
p <- efftox_priors(alpha_mean = -7.9593, alpha_sd = 3.5487,  
                 beta_mean = 1.5482, beta_sd = 3.5018,  
                 gamma_mean = 0.7367, gamma_sd = 2.5423,  
                 zeta_mean = 3.4181, zeta_sd = 2.4406,  
                 eta_mean = 0, eta_sd = 0.2,  
                 psi_mean = 0, psi_sd = 1)  
real_doses = c(1.0, 2.0, 4.0, 6.6, 10.0)  
model <- get_trialr_efftox(real_doses = real_doses,  
                         efficacy_hurdle = 0.5, toxicity_hurdle = 0.3,  
                         p_e = 0.1, p_t = 0.1,  
                         eff0 = 0.5, tox1 = 0.65,  
                         eff_star = 0.7, tox_star = 0.25,  
                         priors = p, iter = 1000, chains = 1, seed = 2020)  
x <- model %>% fit('1N 2E 3B')  
eff_limit(x)
```
empiric_eff_rate

empiric_eff_rate  
*Observed efficacy rate at each dose.*

**Description**

Get the empirical or observed efficacy rate seen at each dose under investigation. This is simply the number of efficacies divided by the number of patients evaluated.

**Usage**

```r
empiric_eff_rate(x, ...)
```

**Arguments**

- `x` Object of class `selector`
- `...` arguments passed to other methods

**Value**

a numerical vector

**Examples**

```r
prob_select = c(0.1, 0.3, 0.5, 0.07, 0.03)
model <- get_random_selector(prob_select = prob_select,
supports_efficacy = TRUE)
x <- model %>% fit('1NTN 2EN 5BB')
empiric_tox_rate(x)
```

empiric_tox_rate

empiric_tox_rate  
*Observed toxicity rate at each dose.*

**Description**

Get the empirical or observed toxicity rate seen at each dose under investigation. This is simply the number of toxicities divided by the number of patients evaluated.

**Usage**

```r
empiric_tox_rate(x, ...)
```

**Arguments**

- `x` Object of class `selector`
- `...` arguments passed to other methods
enforce_three_plus_three

Enforce that a trial path has followed the 3+3 method.

Description

This function stops with an error if it detects that outcomes describing a trial path have diverged from that advocated by the 3+3 method.

Usage

enforce_three_plus_three(outcomes, allow_deescalate = FALSE)

Arguments

outcomes

Outcomes observed. See parse_phase1_outcomes.

allow_deescalate

TRUE to allow de-escalation, as described by Korn et al. Default is FALSE.

Value

Nothing. Function stops if problem detected.

Examples

## Not run:
enforce_three_plus_three('1NNN 2NTN 2NNN')  # OK
enforce_three_plus_three('1NNN 2NTN 2N')   # OK too, albeit in-progress cohort
enforce_three_plus_three('1NNN 1N')         # Not OK because should have escalated

## End(Not run)
**fit**

*Fit a dose-finding model.*

**Description**

Fit a dose-finding model to some outcomes.

**Usage**

```r
fit(selector_factory, outcomes, ...)
```

**Arguments**

- `selector_factory`: Object of type `selector_factory`.
- `outcomes`: Outcome string. See `parse_phase1_outcomes`.
- `...`: Extra args are passed onwards.

**Value**

Object of generic type `selector`.

**See Also**

`selector`, `selector_factory`

**Examples**

```r
skeleton <- c(0.05, 0.1, 0.25, 0.4, 0.6)
target <- 0.25
model <- get_dfcrm(skeleton = skeleton, target = target)
fit <- model %>% fit('1NN 2NTN')
fit %>% recommended_dose()  # Etc
```

**follow_path**

*Follow a pre-determined dose administration path.*

**Description**

This method creates a dose selector that will follow a pre-specified trial path. Whilst the trial path is matched by realised outcomes, the selector will recommend the next dose in the desired sequence. As soon as the observed outcomes diverge from the desired path, the selector stops giving dose recommendations. This makes it possible, for instance, to specify a fixed escalation plan that should be followed until the first toxicity is seen. This tactic is used by some model-based designs to get rapidly to the doses where the action is. See, for example, the dfcrm package and Cheung (2011).
Usage

follow_path(path)

Arguments

path Follow this outcome path. See parse_phase1_outcomes.

Value

an object of type selector_factory that can fit a dose-finding model to outcomes.

References


Examples

model1 <- follow_path(path = '1NNN 2NNN 3NNN 4NNN')

fit1 <- model1 %>% fit('1NNN 2N')
fit1 %>% recommended_dose()
fit1 %>% continue()
# The model recommends continuing at dose 2 because the observed outcomes
# perfectly match the desired escalation path.

fit2 <- model1 %>% fit('1NNN 2NT')
fit2 %>% recommended_dose()
fit2 %>% continue()
# Uh oh. Toxicity has now been seen, the outcomes diverge from the sought
# path, hence this class recommends no dose now.
# At this point, we can hand over dose selection decisions to another class
# by chaining them together, like:
model2 <- follow_path(path = '1NNN 2NNN 3NNN 4NNN') %>%
  get_dfcrm(skeleton = c(0.05, 0.1, 0.25, 0.4, 0.6), target = 0.25)
fit3 <- model2 %>% fit('1NNN 2NT')
# Now the CRM model is using all of the outcomes to calculate the next dose:
fit3 %>% recommended_dose()
fit3 %>% continue()

get_boin

Get an object to fit the BOIN model using the BOIN package.

Description

Get an object to fit the BOIN model using the BOIN package.

Usage

get_boin(num_doses, target, use_stopping_rule = TRUE, ...)

get_boin
Arguments

- num_doses: Number of doses under investigation.
- target: We seek a dose with this probability of toxicity.
- use_stopping_rule: TRUE to use the toxicity stopping rule described in Yan et al. (2019). FALSE to suppress the authors’ stopping rule, with the assumption being that you will test the necessity to stop early in some other way.

... Extra args are passed to \texttt{get.boundary}.

Value

an object of type \texttt{selector\_factory} that can fit the BOIN model to outcomes.

References


Examples

```r
target <- 0.25
model1 <- get_boin(num_doses = 5, target = target)

outcomes <- '1NNN 2NTN'
model1 %>% fit(outcomes) %>% recommended_dose()
```

get\_dfcrm

\textit{Get an object to fit the CRM model using the dfcrm package.}

Description

This function returns an object that can be used to fit a CRM model using methods provided by the dfcrm package.

Dose selectors are designed to be daisy-chained together to achieve different behaviours. This class is a **resumptive** selector, meaning it carries on when the previous dose selector, where present, has elected not to continue. For example, this allows instances of this class to be preceded by a selector that follows a fixed path in an initial escalation plan, such as that provided by \texttt{follow\_path}.

In this example, when the observed trial outcomes deviate from that initial plan, the selector following the fixed path elects not to continue and responsibility passes to this class. See Examples.

Usage

\texttt{get\_dfcrm(parent\_selector\_factory = NULL, skeleton, target, ...)}
Arguments

parent_selector_factory
optional object of type selector_factory that is in charge of dose selection before this class gets involved. Leave as NULL to just use CRM from the start.
skeleton Dose-toxicity skeleton, a non-decreasing vector of probabilities.
target We seek a dose with this probability of toxicity.
... Extra args are passed to CRM.

Value

an object of type selector_factory that can fit the CRM model to outcomes.

References


Examples

skeleton <- c(0.05, 0.1, 0.25, 0.4, 0.6)
target <- 0.25
model1 <- get_dfcrm(skeleton = skeleton, target = target)

# By default, dfcrm fits the empiric model:
outcomes <- '1NNN 2NTN'
model1 %>% fit(outcomes) %>% recommended_dose()

# But we can provide extra args to get_dfcrm that are than passed onwards to # the call to dfcrm::crm to override the defaults. For example, if we want # the one-parameter logistic model:
model2 <- get_dfcrm(skeleton = skeleton, target = target, model = 'logistic')
model2 %>% fit(outcomes) %>% recommended_dose()

# dfcrm does not offer a two-parameter logistic model but other classes do.

# We can use an initial dose-escalation plan, a pre-specified path that # should be followed until trial outcomes deviate, at which point the CRM # model takes over. For instance, if we want to use two patients at each of # the first three doses in the absence of toxicity, irrespective the model's # advice, we would run:
model1 <- follow_path('1NNN 2NNN 3NNN') %>%
  get_dfcrm(skeleton = skeleton, target = target)

# If outcomes match the desired path, the path is followed further:
model1 %>% fit('1NNN 2N') %>% recommended_dose()
get_dose_paths

# But when the outcomes diverge:
model1 %>% fit('1NN 2T') %>% recommended_dose()

# Or the pre-specified path comes to an end:
model1 %>% fit('1NN 2NN 3NN') %>% recommended_dose()
# The CRM model takes over.

get_dose_paths  Calculate future dose paths.

Description

A dose-escalation design exists to select doses in response to observed outcomes. The entire space of possible responses can be calculated to show the behaviour of a design in response to all feasible outcomes. This function performs that task.

Usage

get_dose_paths(selector_factory, cohort_sizes, ...)

Arguments

selector_factory
  Object of type selector_factory.
cohort_sizes
  Integer vector representing sizes of
...  Extra args are passed onwards.

Value

Object of type dose_paths.

Examples

# Calculate paths for a 3+3 design for the next two cohorts of three patients
paths <- get_three_plus_three(num_doses = 5) %>%
  get_dose_paths(cohort_sizes = c(3, 3))
get_empiric_crm_skeleton_weights

*Get posterior model weights for several empiric CRM skeletons.*

**Description**

Get posterior model weights for several empiric CRM skeletons, assuming a normal prior on the beta model parameter.

**Usage**

```r
get_empiric_crm_skeleton_weights(
  skeletons,
  events_at_dose,
  n_at_dose,
  prior = rep(1, nrow(skeletons))
)
```

**Arguments**

- `skeletons`: matrix with one skeleton per row, so that the number of columns is the number of doses under investigation.
- `events_at_dose`: integer vector of number of events at doses
- `n_at_dose`: integer vector of number of patients at doses
- `prior`: vector of prior model weights. Length should be same as number of rows in `skeletons`. Default is equal weighting.

**Value**

numerical vector, posterior weights of the skeletons.

**Examples**

```r
# TODO
```

get_mtpi

*Get an object to fit the mTPI dose-finding model.*

**Description**

The modified toxicity probability interval (mTPI) is a dose-escalation design by Ji et al. As the name suggests, it is an adaptation of the TPI design.
get_mtpi

Usage

get_mtpi(
    num_doses,  
target,  
epsilon1,  
epsilon2,  
exclusion_certainty,  
alpha = 1,  
beta = 1,  
...  
)

Arguments

num_doses Number of doses under investigation.
target We seek a dose with this probability of toxicity.
epsilon1 This parameter determines the lower bound of the equivalence interval. See Details.
epsilon2 This parameter determines the upper bound of the equivalence interval. See Details.
exclusion_certainty Numeric, threshold posterior certainty required to exclude a dose for being excessively toxic. The authors discuss values in the range 0.7 - 0.95. Set to a value > 1 to suppress the dose exclusion mechanism. The authors use the Greek letter xi for this parameter.
alpha First shape parameter of the beta prior distribution on the probability of toxicity.
beta Second shape parameter of the beta prior distribution on the probability of toxicity.
... Extra args are passed onwards.

Value

an object of type selector_factory that can fit the TPI model to outcomes.

Details

The design seeks a dose with probability of toxicity \( p_i \) close to a target probability \( p_T \) by iteratively calculating the interval

\[
p_T - \epsilon_1 < p_i < p_T + \epsilon_2
\]

In this model, \( \epsilon_1 \) and \( \epsilon_2 \) are specified constants. \( p_i \) is estimated by a Bayesian beta-binomial conjugate model

\[
p_i|data \sim Beta(\alpha + x_i, \beta + n_i - x_i),
\]

where \( x_i \) is the number of toxicities observed and \( n_i \) is the number of patients treated at dose \( i \), and \( \alpha \) and \( \beta \) are hyperparameters for the beta prior on \( p_i \). A dose is excluded as inadmissible if

\[
P(p_i > p_T|data) > \xi
\]
The trial commences at a starting dose, possibly dose 1. If dose $i$ has just been evaluated in patient(s), dose selection decisions proceed by calculating the unit probability mass of the true toxicity rate at dose $i$ using the partition of the probability space $p_i < p_T - \epsilon_1$, $p_T - \epsilon_1 < p_i < p_T + \epsilon_2$, and $p_i > p_T + \epsilon_2$. The unit probability mass (UPM) of an interval is the posterior probability that the true toxicity rate belongs to the interval divided by the width of the interval. The interval with maximal UPM determines the recommendation for the next patient(s), with the intervals corresponding to decisions to escalate, stay, and de-escalate dose, respectively. Further to this are rules that prevent escalation to an inadmissible dose. In their paper, the authors demonstrate acceptable operating performance using $\alpha = \beta = 1$, $K_1 = 1$, $K_2 = 1.5$ and $\xi = 0.95$. See the publications for full details.

References


Examples

```r
target <- 0.25
model1 <- get_mtpi(num_doses = 5, target = target, epsilon1 = 0.05,
                   epsilon2 = 0.05, exclusion_certainty = 0.95)

outcomes <- 'NNTN 2NTN'
model1 %>% fit(outcomes) %>% recommended_dose()
```

get_random_selector

Get an object to fit a dose-selector that randomly selects doses.

Description

Get an object to fit a dose-selector that randomly selects doses. Whilst this design is unlikely to pass the ethical hurdles when investigating truly experimental treatments, this class is useful for illustrating methods and can be useful for benchmarking.

Usage

```r
get_random_selector(
  parent_selector_factory = NULL,
  prob_select,
  supports_efficacy = FALSE,
  ...
)
```
get_three_plus_three

Description
Get an object to fit the 3+3 model.

Usage
get_three_plus_three(num_doses, allow_deescalate = FALSE, ...)

Arguments
num_doses Number of doses under investigation.
allow_deescalate TRUE to allow de-escalation, as described by Korn et al. Default is FALSE.
... Extra args are not currently used.
get_tpi

Value

an object of type selector_factory that can fit the 3+3 model to outcomes.

References


Examples

model <- get_three_plus_three(num_doses = 5)

fit1 <- model %>% fit('1NNN 2NTN')
fit1 %>% recommended_dose()
fit1 %>% continue()

fit2 <- model %>% fit('1NNN 2NTN 2NNT')
fit2 %>% recommended_dose()
fit2 %>% continue()
**Arguments**

- `num_doses`: Number of doses under investigation.
- `target`: We seek a dose with this probability of toxicity.
- `k1`: The K1 parameter in TPI determines the upper bound of the equivalence interval. See Details.
- `k2`: The K2 parameter in TPI determines the lower bound of the equivalence interval. See Details.
- `exclusion_certainty`: Numeric, threshold posterior certainty required to exclude a dose for being excessively toxic. The authors discuss values in the range 0.7 - 0.95. Set to a value > 1 to suppress the dose exclusion mechanism. The authors use the Greek letter xi for this parameter.
- `alpha`: First shape parameter of the beta prior distribution on the probability of toxicity.
- `beta`: Second shape parameter of the beta prior distribution on the probability of toxicity.
- ... Extra args are passed onwards.

**Value**

An object of type `selector_factory` that can fit the TPI model to outcomes.

**Details**

The design seeks a dose with probability of toxicity $p_i$ close to a target probability $p_T$ by iteratively calculating the interval

$$ p_T - K_2 \sigma_i < p_i < p_T + K_1 \sigma_i $$

In this model, $K_1$ and $K_2$ are specified constants and $\sigma_i$ is the standard deviation of $p_i$ arising from a Bayesian beta-binomial conjugate model

$$ p_i \mid data \sim Beta(\alpha + x_i, \beta + n_i - x_i), $$

where $x_i$ is the number of toxicities observed and $n_i$ is the number of patients treated at dose $i$, and $\alpha$ and $\beta$ are hyperparameters for the beta prior on $p_i$. A dose is excluded as inadmissible if

$$ P(p_i > p_T \mid data) > \xi $$

The trial commences at a starting dose, possibly dose 1. If dose $i$ has just been evaluated in patient(s), dose selection decisions proceed by calculating the posterior probability that the true toxicity rate at dose $i$ belongs to the three partition regions $p_i < p_T - K_2 \sigma_i$, $p_T - K_2 \sigma_i < p_i < p_T + K_1 \sigma_i$, and $p_i > p_T + K_2 \sigma_i$, corresponding to decisions escalate, stay, and de-escalate dose, respectively. Further to this are rules that prevent escalation to an inadmissible dose. In their paper, the authors demonstrate acceptable operating performance using $\alpha = \beta = 0.005$, $K_1 = 1$, $K_2 = 1.5$ and $\xi = 0.95$. See the publications for full details.

**References**


Examples

target <- 0.25
model1 <- get_tpi(num_doses = 5, target = target, k1 = 1, k2 = 1.5,
exclusion_certainty = 0.95)

outcomes <- '1NNN 2NTN'
model1 %>% fit(outcomes) %>% recommended_dose()

get_trialr_crm

Get an object to fit the CRM model using the trialr package.

Description

This function returns an object that can be used to fit a CRM model using methods provided by the
trialr package.

Dose selectors are designed to be daisy-chained together to achieve different behaviours. This class
is a **resumptive** selector, meaning it carries on when the previous dose selector, where present,
has elected not to continue. For example, this allows instances of this class to be preceded by a se-
lector that follows a fixed path in an initial escalation plan, such as that provided by follow_path.
In this example, when the observed trial outcomes deviate from that initial plan, the selector fol-
lowing the fixed path elects not to continue and responsibility passes to this class. See Examples.

Usage

get_trialr_crm(parent_selector_factory = NULL, skeleton, target, model, ...)

Arguments

parent_selector_factory
optional object of type selector_factory that is in charge of dose selection
before this class gets involved. Leave as NULL to just use CRM from the start.

skeleton  Dose-toxicity skeleton, a non-decreasing vector of probabilities.
target     We seek a dose with this probability of toxicity.
model     character string identifying which model form to use. Options include empiric,
          logistic, logistic2. The model form chosen determines which prior hyperparam-
          eters are required. See stan_crm for more details.
          ...
          Extra args are passed to stan_crm.

Value

an object of type selector_factory that can fit the CRM model to outcomes.
References


Examples

```r
skeleton <- c(0.05, 0.1, 0.25, 0.4, 0.6)
target <- 0.25

# The model to use must be specified in trialr:
model1 <- get_trialr_crm(skeleton = skeleton, target = target,
                         model = 'empiric', beta_sd = 1.34)

# Refer to the trialr documentation for more details on model forms.
outcomes <- '1NNN 2NTN'
model1 %>% fit(outcomes) %>% recommended_dose()

# But we can provide extra args to trialr that are than passed onwards to
# the call to trialr::stan_crm to override the defaults.
# For example, if we want the one-parameter logistic model, we run:
model2 <- get_trialr_crm(skeleton = skeleton, target = target,
                         model = 'logistic', a0 = 3,
                         beta_mean = 0, beta_sd = 1)
model2 %>% fit(outcomes) %>% recommended_dose()

# And, if we want the two-parameter logistic model, we run:
model3 <- get_trialr_crm(skeleton = skeleton, target = target,
                         model = 'logistic2',
                         alpha_mean = 0, alpha_sd = 2,
                         beta_mean = 0, beta_sd = 1)
model3 %>% fit(outcomes) %>% recommended_dose()

# We can use an initial dose-escalation plan, a pre-specified path that
# should be followed until trial outcomes deviate, at which point the CRM
# model takes over. For instance, if we want to use two patients at each of
# the first three doses in the absence of toxicity, irrespective the model's
# advice, we would run:
model1 <- follow_path('1NN 2NN 3NN') %>%
       get_trialr_crm(skeleton = skeleton, target = target, model = 'empiric',
                       beta_sd = 1.34)

# If outcomes match the desired path, the path is followed further:
model1 %>% fit('1NN 2N') %>% recommended_dose()

# But when the outcomes diverge:
model1 %>% fit('1NN 2T') %>% recommended_dose()

# Or the pre-specified path comes to an end:
model1 %>% fit('1NN 2NN 3NN') %>% recommended_dose()
# ...the CRM model takes over.
```
get_trialr_efftox

Get an object to fit the EffTox model using the trialr package.

Description

This function returns an object that can be used to fit the EffTox model for phase I/II dose-finding using methods provided by the trialr package.

Usage

get_trialr_efftox(
  parent_selector_factory = NULL,
  real_doses,
  efficacy_hurdle,
  toxicity_hurdle,
  p_e,
  p_t,
  eff0,
  tox1,
  eff_star,
  tox_star,
  priors,
  ...
)

Arguments

parent_selector_factory
  optional object of type selector_factory that is in charge of dose selection before this class gets involved. Leave as NULL to just use EffTox from the start.

real_doses
  A vector of numbers, the doses under investigation. They should be ordered from lowest to highest and be in consistent units. E.g. to conduct a dose-finding trial of doses 10mg, 20mg and 50mg, use c(10, 20, 50).

efficacy_hurdle
  Minimum acceptable efficacy probability. A number between 0 and 1.

toxicity_hurdle
  Maximum acceptable toxicity probability. A number between 0 and 1.

p_e
  Certainty required to infer a dose is acceptable with regards to being probably efficacious; a number between 0 and 1.

p_t
  Certainty required to infer a dose is acceptable with regards to being probably tolerable; a number between 0 and 1.

eff0
  Efficacy probability required when toxicity is impossible; a number between 0 and 1 (see Details).

tox1
  Toxicity probability permitted when efficacy is guaranteed; a number between 0 and 1 (see Details).
get_trialr_nb

**get_trialr_nb**

`get_trialr_nb` Get an object to fit the NBG dose-finding model using the trialr package.

**Description**

This function returns an object that can be used to fit a Neuenschwander, Branson and Gsponer (NBG) model for dose-finding using methods provided by the trialr package.

**Value**

an object of type `selector_factory` that can fit the EffTox model to outcomes.

**References**


**Examples**

```r
efftox_priors <- trialr::efftox_priors
p <- efftox_priors(alpha_mean = -7.9593, alpha_sd = 3.5487,
                   beta_mean = 1.5482, beta_sd = 3.5018,
                   gamma_mean = 0.7367, gamma_sd = 2.5423,
                   zeta_mean = 3.4181, zeta_sd = 2.4406,
                   eta_mean = 0, eta_sd = 0.2,
                   psi_mean = 0, psi_sd = 1)
real_doses = c(1.0, 2.0, 4.0, 6.6, 10.0)
model <- get_trialr_efftox(real_doses = real_doses,
                          efficacy_hurdle = 0.5, toxicity_hurdle = 0.3,
                          p_e = 0.1, p_t = 0.1,
                          eff0 = 0.5, tox1 = 0.65,
                          eff_star = 0.7, tox_star = 0.25,
                          priors = p, iter = 1000, chains = 1, seed = 2020)
```
Usage

get_trialr_nbgnbr

parent_selector_factory = NULL,
real_doses,
d_star,
target,
alpha_mean,
alpha_sd,
beta_mean,
beta_sd,
...)

Arguments

parent_selector_factory
    optional object of type selector_factory that is in charge of dose selection before this class gets involved. Leave as NULL to just use this model from the start.
real_doses
    Doses under investigation, a non-decreasing vector of numbers.

Arguments

d_star
    Numeric, reference dose for calculating the covariate log(dose / d_star) when fitting the model. Sometimes (but not always) taken to be the max dose in real_doses.

target
    We seek a dose with this probability of toxicity.

Arguments

alpha_mean
    Prior mean of intercept variable for normal prior. See Details. Also see documentation for trialr package for further details.

Arguments

alpha_sd
    Prior standard deviation of intercept variable for normal prior. See Details. Also see documentation for trialr package for further details.

Arguments

beta_mean
    Prior mean of gradient variable for normal prior. See Details. Also see documentation for trialr package for further details.

Arguments

beta_sd
    Prior standard deviation of slope variable for normal prior. See Details. Also see documentation for trialr package for further details.

Arguments

... Extra args are passed to stan_nbg.

Details

The model form implemented in trialr is:

\[ F(x_i, \alpha, \beta) = \frac{1}{1 + \exp \left( - (\alpha + \exp(\beta) \log(x_i/d_\star)) \right)} \]

with normal priors on alpha and beta.

Dose selectors are designed to be daisy-chained together to achieve different behaviours. This class is a **resumptive** selector, meaning it carries on when the previous dose selector, where present, has elected not to continue. For example, this allows instances of this class to be preceded by a selector that follows a fixed path in an initial escalation plan, such as that provided by follow_path. In this example, when the observed trial outcomes deviate from that initial plan, the selector following the fixed path elects not to continue and responsibility passes to this class. See examples under get_dfcrm.
get_wages_and_tait

**Value**

an object of type `selector_factory` that can fit the NBG model to outcomes.

**References**


**Examples**

```r
real_doses <- c(5, 10, 25, 40, 60)
d_star <- 60
target <- 0.25

model <- get_trialr_nbg(real_doses = real_doses, d_star = d_star,
                        target = target, alpha_mean = 2, alpha_sd = 1,
                        beta_mean = 0.5, beta_sd = 1)
# Refer to the trialr documentation for more details on model & priors.
outcomes <- c('1NNN', '2TNN')
fit <- model %>% fit(outcomes)
fit %>% recommended_dose()
fit %>% mean_prob_tox()
```

---

get_wages_and_tait  
_Get an object to fit Wages & Tait's model for phase I/II dose-finding._

**Description**

This function returns an object that can be used to fit Wages & Taits model for phase I/II dose-finding, i.e. it selects doses according to efficacy and toxicity outcomes. This function delegates prior-to-posterior calculations to the dcrm package.

**Usage**

```r
get_wages_and_tait(
  parent_selector_factory = NULL,
  tox_skeleton,
  eff_skeletons,
  eff_skeleton_weights = rep(1, nrow(eff_skeletons)),
  tox_limit,
  eff_limit,
  num_randomise,
  ...
)
```
get_wages_and_tait

Arguments

**parent_selector_factory**
optional object of type `selector_factory` that is in charge of dose selection before this class gets involved. Leave NULL to just use this model from the start.

**tox_skeleton**
Dose-toxicity skeleton, a non-decreasing vector of probabilities.

**eff_skeletons**
Matrix of dose-efficacy skeletons, with the skeletons in rows. I.e. number of cols is equal to number of doses, and number of rows is equal to number of efficacy skeletons under consideration.

**eff_skeleton_weights**
numerical vector, prior weights to efficacy skeletons. Should have length equal to number of rows in `eff_skeletons`. Default is equal weights.

**tox_limit**
We seek a dose with probability of toxicity no greater than this. Value determines the admissible set. See Wages & Tait (2015).

**eff_limit**
We seek a dose with probability of efficacy no less than this.

**num_randomise**
integer, maximum number of patients to use in the adaptive randomisation phase of the trial.

... Extra args are passed onwards.

Value

an object of type `selector_factory`.

References


Examples

```r
# Example in Wages & Tait (2015)
tox_skeleton = c(0.01, 0.08, 0.15, 0.22, 0.29, 0.36)
eff_skeletons = matrix(nrow=11, ncol=6)
eff_skeletons[1,] <- c(0.60, 0.50, 0.40, 0.30, 0.20, 0.10)
eff_skeletons[2,] <- c(0.50, 0.60, 0.50, 0.40, 0.30, 0.20)
eff_skeletons[3,] <- c(0.40, 0.50, 0.60, 0.50, 0.40, 0.30)
eff_skeletons[4,] <- c(0.30, 0.40, 0.50, 0.60, 0.50, 0.40)
eff_skeletons[5,] <- c(0.20, 0.30, 0.40, 0.50, 0.60, 0.50)
eff_skeletons[6,] <- c(0.10, 0.20, 0.30, 0.40, 0.50, 0.60)
eff_skeletons[7,] <- c(0.20, 0.30, 0.40, 0.50, 0.60, 0.60)
eff_skeletons[8,] <- c(0.30, 0.40, 0.50, 0.60, 0.60, 0.60)
eff_skeletons[9,] <- c(0.40, 0.50, 0.60, 0.60, 0.60, 0.60)
eff_skeletons[10,] <- c(0.50, 0.60, 0.60, 0.60, 0.60, 0.60)
eff_skeletons[11,] <- c(rep(0.60, 6))
eff_skeleton_weights = rep(1, nrow(eff_skeletons))
tox_limit = 0.33
eff_limit = 0.05
model <- get_wages_and_tait(tox_skeleton = tox_skeleton,
```
graph_paths

```
eff_skeletons = eff_skeletons,
tox_limit = tox_limit, eff_limit = eff_limit,
um_randomise = 20)
fit <- model %>% fit(’1NN 2EN 3BE’)
fit %>% recommended_dose()
fit %>% is_randomising()
fit %>% dose_admissible()
fit %>% prob_administer()
```

---

**Description**

Visualise dose-paths as a graph

**Usage**

```
graph_paths(paths, viridis_palette = "viridis", RColorBrewer_palette = NULL)
```

**Arguments**

- **paths**: Object of type `dose_paths`  
- **viridis_palette**: optional name of a colour palette in the viridis package.  
- **RColorBrewer_palette**: optional name of a colour palette in the RColorBrewer package.

**Details**

The viridis package supports palettes: viridis, magma, plasma, inferno, and cividis. The RColorBrewer package supports many palettes. Refer to those packages on CRAN for more details.

**Examples**

```
paths <- get_three_plus_three(num_doses = 5) %>%
  get_dose_paths(cohort_sizes = c(3, 3, 3))
## Not run:
graph_paths(paths)
graph_paths(paths, viridis_palette = 'plasma')
graph_paths(paths, RColorBrewer_palette = 'YlOrRd')
## End(Not run)
```
**is_randomising**  
*Is this selector currently randomly allocating doses?*

**Description**  
Get the percentage of patients evaluated at each dose under investigation.

**Usage**  
```r
is_randomising(x, ...)
```

**Arguments**

- `x`  
  Object of class `selector`

- `...`  
  arguments passed to other methods

**Value**  
A logical value

**Examples**
```r
class(outcomes) <- c('1NNN 2NTN')
fit <- get_random_selector(prob_select = c(0.1, 0.6, 0.3)) %>%
  fit(outcomes)
fit %>% is_randomising()
```

---

**mean_prob_eff**  
*Mean efficacy rate at each dose.*

**Description**  
Get the estimated mean efficacy rate at each dose under investigation. This is a set of modelled statistics. The underlying models estimate efficacy probabilities in different ways. If no model-based estimate of the mean is available, this function will return a vector of NAs.

**Usage**  
```r
mean_prob_eff(x, ...)
```

**Arguments**

- `x`  
  Object of class `selector`

- `...`  
  arguments passed to other methods
mean_prob_tox

Value

a numerical vector

Examples

efftox_priors <- trialr::efftox_priors
p <- efftox_priors(alpha_mean = -7.9593, alpha_sd = 3.5487,
                   beta_mean = 1.5482, beta_sd = 3.5018,
                   gamma_mean = 0.7367, gamma_sd = 2.5423,
                   zeta_mean = 3.4181, zeta_sd = 2.4406,
                   eta_mean = 0, eta_sd = 0.2,
                   psi_mean = 0, psi_sd = 1)
real_doses = c(1.0, 2.0, 4.0, 6.6, 10.0)
model <- get_trialr_efftox(real_doses = real_doses,
                            efficacy_hurdle = 0.5, toxicity_hurdle = 0.3,
                            p_e = 0.1, p_t = 0.1,
                            eff0 = 0.5, tox0 = 0.65,
                            eff_star = 0.7, tox_star = 0.25,
                            priors = p, iter = 1000, chains = 1, seed = 2020)
x <- model %>% fit('1N 2E 3B')
mean_prob_eff(x)

mean_prob_tox

Mean toxicity rate at each dose.

Description

Get the estimated mean toxicity rate at each dose under investigation. This is a set of modelled statistics. The underlying models estimate toxicity probabilities in different ways. If no model-based estimate of the mean is available, this function will return a vector of NAs.

Usage

mean_prob_tox(x, ...)

Arguments

x Object of class selector

... arguments passed to other methods

Value

a numerical vector
Examples

```r
# CRM example
skeleton <- c(0.05, 0.1, 0.25, 0.4, 0.6)
target <- 0.25
outcomes <- '1NNN 2NTN'
fit <- get_dfcrm(skeleton = skeleton, target = target) %>% fit(outcomes)
fit %>% mean_prob_tox()
```

---

**median_prob_eff**  
Median efficacy rate at each dose.

Description

Get the estimated median efficacy rate at each dose under investigation. This is a set of modelled statistics. The underlying models estimate efficacy probabilities in different ways. If no model-based estimate of the median is available, this function will return a vector of NAs.

Usage

```r
median_prob_eff(x, ...)
```

Arguments

- `x` Object of class `selector`
- `...` arguments passed to other methods

Value

a numerical vector

Examples

```r
efftox_priors <- trialr::efftox_priors
p <- efftox_priors(alpha_mean = -7.9593, alpha_sd = 3.5487,
                   beta_mean = 1.5482, beta_sd = 3.5018,
                   gamma_mean = 0.7367, gamma_sd = 2.5423,
                   zeta_mean = 3.4181, zeta_sd = 2.4406,
                   eta_mean = 0, eta_sd = 0.2,
                   psi_mean = 0, psi_sd = 1)
real_doses = c(1.0, 2.0, 4.0, 6.6, 10.0)
model <- get_trialr_efftox(real_doses = real_doses,
                           efficacy_hurdle = 0.5, toxicity_hurdle = 0.3,
                           p_e = 0.1, p_t = 0.1,
                           eff0 = 0.5, tox1 = 0.65,
                           eff_star = 0.7, tox_star = 0.25,
                           priors = p, iter = 1000, chains = 1, seed = 2020)
x <- model %>% fit('1N 2E 3B')
median_prob_eff(x)
```
Description

Get the estimated median toxicity rate at each dose under investigation. This is a set of modelled statistics. The underlying models estimate toxicity probabilities in different ways. If no model-based estimate of the median is available, this function will return a vector of NAs.

Usage

```
median_prob_tox(x, ...)
```

Arguments

- `x` Object of class `selector`
- `...` arguments passed to other methods

Value

a numerical vector

Examples

```
# CRM example
skeleton <- c(0.05, 0.1, 0.25, 0.4, 0.6)
target <- 0.25
outcomes <- c('1NNN', '2NTN')
fit <- get_dfcrm(skeleton = skeleton, target = target) %>% fit(outcomes)
fit %>% median_prob_tox()
```

model_frame

Model data-frame.

Description

Get the model data-frame for a dose-finding analysis, including columns for patient id, cohort id, dose administered, and toxicity outcome. In some scenarios, further columns are provided.

Usage

```
model_frame(x, ...)
```

Arguments

- `x` Object of type `selector`.
- `...` Extra args are passed onwards.
Value

tibble, which acts like a data.frame.

Examples

# In a toxicity-only setting:
skeleton <- c(0.05, 0.1, 0.25, 0.4, 0.6)
target <- 0.25
model <- get_dfcrm(skeleton = skeleton, target = target)
fit <- model %>% fit('1NNN 2NTN')
fit %>% model_frame()

# In an efficacy-toxicity setting
prob_select = c(0.1, 0.3, 0.5, 0.07, 0.03)
model <- get_random_selector(prob_select = prob_select,
supports_efficacy = TRUE)
x <- model %>% fit('1NTN 2EN 5BB', supports_efficacy = TRUE)
fit %>% model_frame()

num_cohort_outcomes

Number of different possible outcomes for a cohort of patients

Description

Number of different possible outcomes for a cohort of patients, each of which will experience one of a number of discrete outcomes. For instance, in a typical phase I dose-finding trial, each patient will experience: no-toxicity (N); or toxicity (T). The number of possible outcomes per patient is two. For a cohort of three patients, the number of cohort outcomes is four: NNN, NNT, NTT, TTT. Consider a more complex example: in a seamless phase I/II trial with efficacy and toxicity outcomes, an individual patient will experience one of four distinct outcomes: efficacy only (E); toxicity only (T); both efficacy and toxicity (B) or neither. How many different outcomes are there for a cohort of three patients? The answer is 20 but it is non-trivial to see why. This convenience function calculates that number using the formula for the number of combinations with replacement,

Usage

num_cohort_outcomes(num_patient_outcomes, cohort_size)

Arguments

num_patient_outcomes
  integer, number of distinct possible outcomes for each single patient
cohort_size
  integer, number of patients in the cohort

Value

integer, number of distinct possible cohort outcomes
**num_doses**

**Number of doses.**

**Examples**

# As described in example, N or T in a cohort of three:
num_cohort_outcomes(num_patient_outcomes = 2, cohort_size = 3)

# Also described in example, E, T, B or N in a cohort of three:
num_cohort_outcomes(num_patient_outcomes = 4, cohort_size = 3)

**Description**

Get the number of doses under investigation in a dose-finding trial.

**Usage**

num_doses(x, ...)

**Arguments**

- **x** Object of type `selector`.
- **...** Extra args are passed onwards.

**Value**

integer

**Examples**

skeleton <- c(0.05, 0.1, 0.25, 0.4, 0.6)
target <- 0.25
model <- get_dfcrm(skeleton = skeleton, target = target)
fit <- model %>% fit('1NNN 2NTN')
fit %>% num_doses()

**num_dose_path_nodes**

**Number of nodes in dose-paths analysis**

**Description**

Number of possible nodes in an exhaustive analysis of dose-paths in a dose-finding trial. The number of nodes at depth i is the number of nodes at depth i-1 multiplied by the number of possible cohort outcomes at depth i. For instance, if there were 16 nodes at the previous depth and four possible cohort outcomes at the current depth, there are 64 possible nodes at the current depth. Knowing the number of nodes in a dose-paths analysis helps the analyst decide whether simulation or dose-paths are a better tool for assessing operating characteristics of a dose-finding design.
Usage

num_dose_path_nodes(num_patient_outcomes, cohort_sizes)

Arguments

num_patient_outcomes
  integer, number of distinct possible outcomes for each single patient
cohort_sizes
  integer vector of cohort sizes

Value

integer vector, number of nodes at increasing depths. The total number of nodes is the sum of this vector.

Examples

# In a 3+3 design, there are two possible outcomes for each patient and
# patients are evaluated in cohorts of three. In an analysis of dose-paths in
# the first two cohorts of three, how many nodes are there?
num_dose_path_nodes(num_patient_outcomes = 2, cohort_sizes = rep(3, 2))

# In contrast, using an EffTox design there are four possible outcomes for
# each patient. In a similar analysis of dose-paths in the first two cohorts
# of three, how many nodes are there now?
num_dose_path_nodes(num_patient_outcomes = 4, cohort_sizes = rep(3, 2))

---

num_eff

Total number of efficacies seen.

Description

Get the number of efficacies seen in a dose-finding trial.

Usage

num_eff(x, ...)

Arguments

x
  Object of type selector.
...
  Extra args are passed onwards.

Value

integer
Examples

```r
prob_select = c(0.1, 0.3, 0.5, 0.07, 0.03)
model <- get_random_selector(prob_select = prob_select,
supports_efficacy = TRUE)
x <- model %>% fit('1NTN 2EN 5BB')
num_eff(x)
```

```
num_patients

Number of patients evaluated.
```

Description

Get the number of patients evaluated in a dose-finding trial.

Usage

```r
num_patients(x, ...)
```

Arguments

- `x` Object of type `selector`.
- `...` Extra args are passed onwards.

Value

integer

Examples

```r
skeleton <- c(0.05, 0.1, 0.25, 0.4, 0.6)
target <- 0.25
model <- get_dfcrm(skeleton = skeleton, target = target)
fit <- model %>% fit('1NNN 2NTN')
fit %>% num_patients()
```

```
num_tox

Total number of toxicities seen.
```

Description

Get the number of toxicities seen in a dose-finding trial.

Usage

```r
num_tox(x, ...)
```
Arguments

\( x \)  
Object of type `selector`.

...  
Extra args are passed onwards.

Value

integer

Examples

```
skeleton <- c(0.05, 0.1, 0.25, 0.4, 0.6)
target <- 0.25
model <- get_dfcrm(skeleton = skeleton, target = target)
fit <- model %>% fit('1NNN 2NTN')
fit %>% num_tox()
```

---

**n_at_dose**  
*Number of patients treated at each dose.*

Description

Get the number of patients evaluated at each dose under investigation.

Usage

\( \text{n_at_dose}(x, \ldots) \)

Arguments

\( x \)  
Object of class `selector`

...  
arguments passed to other methods

Value

an integer vector

Examples

```
# CRM example
skeleton <- c(0.05, 0.1, 0.25, 0.4, 0.6)
target <- 0.25
outcomes <- '1NNN 2NTN'
fit <- get_dfcrm(skeleton = skeleton, target = target) %>% fit(outcomes)
fit %>% n_at_dose()
```
**n_at_recommended_dose**  
*Number of patients treated at the recommended dose.*

**Description**  
Get the number of patients evaluated at the recommended dose.

**Usage**  

\[ n_{at\_recommended\_dose}(x, \ldots) \]

**Arguments**  

- **x**  
  Object of class `selector`  
- **...**  
  arguments passed to other methods

**Value**  

an integer

**Examples**  

```r  
# CRM example  
skeleton <- c(0.05, 0.1, 0.25, 0.4, 0.6)  
target <- 0.25  
outcomes <- '1NNN 2NTN'  
fit <- get_dfcrm(skeleton = skeleton, target = target) %>% fit(outcomes)  
fit %>% n_at_recommended_dose()  
```

---

**parse_phase1_2_outcomes**  
*Parse a string of phase I/II dose-finding outcomes to vector notation.*

**Description**  
Parse a string of phase I/II dose-finding outcomes to a binary vector notation necessary for model invocation.  

The outcome string describes the doses given, outcomes observed and groups patients into cohorts. The format of the string is described in Brock et al. (2017). See Examples.  

The letters E, T, N and B are used to represents patients that experienced (E)fficacy only, (T)oxicity only, (B)oth efficacy and toxicity, and (N)either. These letters are concatenated after numerical dose-levels to convey the outcomes of cohorts of patients. For instance, 2ETB represents a cohort of three patients that were treated at dose-level 2, and experienced efficacy, toxicity and both events, respectively. The results of cohorts are separated by spaces. Thus, 2ETB 1NN extends our previous example, where the next cohort of two were treated at dose-level 1 and both patients experienced neither efficacy nor toxicity. See Examples.
parse_phase1_outcomes

Usage

parse_phase1_2_outcomes(outcomes, as_list = TRUE)

Arguments

outcomes character string, conveying doses given and outcomes observed.
as_list TRUE (the default) to return a list; FALSE to return a data.frame

Value

If as_list == TRUE, a list with elements eff, tox, dose and num_patients. If as_list == FALSE, a data.frame with columns eff, tox and dose.

References


Examples

x = parse_phase1_2_outcomes('TNE 2ENN 3TBB')
# Three cohorts of three patients. The first cohort was treated at dose 1 and
# had no toxicity with one efficacy, etc.
x$num_patients # 9
x$dose # c(1, 1, 1, 2, 2, 2, 3, 3, 3)
x$eff # c(0, 0, 1, 1, 0, 0, 1, 1)
sum(x$eff) # 5
x$tox # c(0, 0, 0, 0, 0, 1, 1, 1)
sum(x$tox) # 3

# The same information can be parsed to a data-frame:
y = parse_phase1_2_outcomes('TNE 2ENN 3TBB', as_list = FALSE)
y

parse_phase1_outcomes Parse a string of phase I dose-finding outcomes to vector notation.

Description

Parse a string of phase I dose-finding outcomes to a binary vector notation necessary for model invocation.

The outcome string describes the doses given, outcomes observed and groups patients into cohorts. The format of the string is described in Brock (2019), and that itself is the phase I analogue of the similar idea described in Brock et al. (2017). See Examples.

The letters T and N are used to represents patients that experienced (T)oxicity and (N)o toxicity. These letters are concatenated after numerical dose-levels to convey the outcomes of cohorts of
patients. For instance, 2NNT represents a cohort of three patients that were treated at dose-level 2, one of whom experienced toxicity, and two that did not. The results of cohorts are separated by spaces. Thus, 2NNT 1NN extends our previous example, where the next cohort of two were treated at dose-level 1 and neither experienced toxicity. See examples.

Usage

```r
parse_phase1_outcomes(outcomes, as_list = TRUE)
```

Arguments

- `outcomes` character string, conveying doses given and outcomes observed.
- `as_list` TRUE (the default) to return a list; FALSE to return a data.frame

Value

If `as_list` == TRUE, a list with elements `tox`, `doses` and `num_patients`. If `as_list` == FALSE, a data.frame with columns `tox` and `doses`.

References


Examples

```r
x = parse_phase1_outcomes('1NNN 2NTN 3TTT')
# Three cohorts of three patients. The first cohort was treated at dose 1 and
# none had toxicity. The second cohort was treated at dose 2 and one of the
# three had toxicity. Finally, cohort three was treated at dose 3 and all
# patients had toxicity.
x$num_patients  # 9
dx$doses        # c(1, 1, 1, 2, 2, 2, 3, 3, 3)
dx$tox          # c(0, 0, 0, 0, 1, 0, 1, 1, 1)
sum(x$tox)      # 4

# The same information can be parsed to a data-frame:
y = parse_phase1_outcomes('1NNN 2NTN 3TTT', as_list = FALSE)
y
```

Break a phase I/II outcome string into a list of cohort parts.

Description

Break a phase I/II outcome string into a list of cohort parts.

The outcome string describes the doses given, outcomes observed and the timing of analyses that recommend a dose. The format of the string is described in Brock _et al_. (2017).

The letters E, T, N & B are used to represents patients that experienced (E)fficacy, (T)oxicity, (N)either and (B)oth. These letters are concatenated after numerical dose-levels to convey the outcomes of cohorts of patients. For instance, 2NET represents a cohort of three patients that were treated at dose-level 2, one of whom experienced toxicity only, one that experienced efficacy only, and one that had neither. The results of cohorts are separated by spaces and it is assumed that a dose-finding decision takes place at the end of a cohort. Thus, 2NET 1NN builds on our previous example, where the next cohort of two were treated at dose-level 1 and neither of these patients experienced either event. See examples.

Usage

-phase1_2_outcomes_to_cohorts(outcomes)-

Arguments

outcomes character string representing the doses given, outcomes observed, and timing of analyses. See Description.

Value

a list with a slot for each cohort. Each cohort slot is itself a list, containing elements: * dose, the integer dose delivered to the cohort; * outcomes, a character string representing the E, T N or B outcomes for the patients in this cohort.

References


Examples

x = phase1_2_outcomes_to_cohorts('1NEN 2ENT 3TB')

length(x)

x[[1]]$dose

x[[1]]$outcomes

x[[2]]$dose
**Description**

Break a phase I outcome string into a list of cohort parts.

The outcome string describes the doses given, outcomes observed and the timing of analyses that recommend a dose. The format of the string is described in Brock (2019), and that itself is the phase I analogue of the similar idea described in Brock _et al_. (2017).

The letters T and N are used to represent patients that experienced (T)oxicity and (N)o toxicity. These letters are concatenated after numerical dose-levels to convey the outcomes of cohorts of patients. For instance, 2NNT represents a cohort of three patients that were treated at dose-level 2, one of whom experienced toxicity, and two that did not. The results of cohorts are separated by spaces and it is assumed that a dose-finding decision takes place at the end of a cohort. Thus, 2NNT 1NN builds on our previous example, where the next cohort of two were treated at dose-level 1 and neither of these patients experienced toxicity. See examples.

**Usage**

```r
phase1_outcomes_to_cohorts(outcomes)
```

**Arguments**

- **outcomes** character string representing the doses given, outcomes observed, and timing of analyses. See Description.

**Value**

a list with a slot for each cohort. Each cohort slot is itself a list, containing elements: * dose, the integer dose delivered to the cohort; * outcomes, a character string representing the T or N outcomes for the patients in this cohort.

**References**


Examples

```r
x = phase1_outcomes_to_cohorts('1NNN 2NTN 3TT')
length(x)
x[[1]]$dose
x[[1]]$outcomes
x[[2]]$dose
x[[2]]$outcomes
x[[3]]$dose
x[[3]]$outcomes
```

---

**prob_administer**

*Percentage of patients treated at each dose.*

Description

Get the percentage of patients evaluated at each dose under investigation.

Usage

```r
prob_administer(x, ...)
```

Arguments

- `x` Object of class `selector`
- `...` arguments passed to other methods

Value

a numerical vector

Examples

```r
# CRM example
skeleton <- c(0.05, 0.1, 0.25, 0.4, 0.6)
target <- 0.25
outcomes <- '1NNN 2NTN'
fit <- get_dfcrm(skeleton = skeleton, target = target) %>% fit(outcomes)
fit %>% prob_administer()
```
prob_eff_quantile

Quantile of the efficacy rate at each dose.

Description

Get the estimated quantile of the efficacy rate at each dose under investigation. This is a set of modelled statistics. The underlying models estimate efficacy probabilities in different ways. If no model-based estimate of the median is available, this function will return a vector of NAs.

Usage

prob_eff_quantile(x, p, ...)

Arguments

x Object of class selector
p quantile probability, decimal value between 0 and 1
... arguments passed to other methods

Value

a numerical vector

Examples

eeftox_priors <- trialr::eftox_priors
p <- eftox_priors(alpha_mean = -7.9593, alpha_sd = 3.5487,
                   beta_mean = 1.5482, beta_sd = 3.5018,
                   gamma_mean = 0.7367, gamma_sd = 2.5423,
                   zeta_mean = 3.4181, zeta_sd = 2.4406,
                   eta_mean = 0, eta_sd = 0.2,
                   psi_mean = 0, psi_sd = 1)
real_doses = c(1.0, 2.0, 4.0, 6.6, 10.0)
model <- get_trialr_eftox(real_doses = real_doses,
                           efficacy_hurdle = 0.5, toxicity_hurdle = 0.3,
                           p_e = 0.1, p_t = 0.1,
                           eff0 = 0.5, tox1 = 0.65,
                           eff_star = 0.7, tox_star = 0.25,
                           priors = p, iter = 1000, chains = 1, seed = 2020)
x <- model %>% fit('1N 2E 3B')
prob_tox_quantile(x, p = 0.9)
prob_recommend

**Probability of recommendation**

- **Description**
  Get the probabilities that each of the doses under investigation is recommended.

- **Usage**
  ```r
  prob_recommend(x, ...)
  ```

- **Arguments**
  - `x`: Object of type `simulations`.
  - `...`: arguments passed to other methods

- **Value**
  vector of probabilities

- **Examples**
  ```r
  true_prob_tox <- c(0.12, 0.27, 0.44, 0.53, 0.57)
  sims <- get_three_plus_three(num_doses = 5) %>%
   simulate_trials(num_sims = 50, true_prob_tox = true_prob_tox)
  sims %>% prob_recommend
  ```

prob_tox_exceeds

**Probability that the toxicity rate exceeds some threshold.**

- **Description**
  Get the probability that the toxicity rate at each dose exceeds some threshold.
  Get the probability that the efficacy rate at each dose exceeds some threshold.

- **Usage**
  ```r
  prob_tox_exceeds(x, threshold, ...)
  prob_eff_exceeds(x, threshold, ...)
  ```

- **Arguments**
  - `x`: Object of type `selector`
  - `threshold`: Probability that efficacy rate exceeds what?
  - `...`: arguments passed to other methods
Value

numerical vector of probabilities
numerical vector of probabilities

Examples

# CRM example
skeleton <- c(0.05, 0.1, 0.25, 0.4, 0.6)
target <- 0.25
outcomes <- '1NNN 2NTN'
fit <- get_dFCRM(skeleton = skeleton, target = target) %>% fit(outcomes)
# What is probability that tox rate at each dose exceeds target by >= 10%?
fit %>% prob_tox_exceeds(threshold = target + 0.1)
efftox_priors <- trialr::efftox_priors
p <- efftox_priors(alpha_mean = -7.9593, alpha_sd = 3.5487,
  beta_mean = 1.5482, beta_sd = 3.5018,
  gamma_mean = 0.7367, gamma_sd = 2.5423,
  zeta_mean = 3.4181, zeta_sd = 2.4406,
  eta_mean = 0, eta_sd = 0.2,
  psi_mean = 0, psi_sd = 1)
real_doses = c(1.0, 2.0, 4.0, 6.6, 10.0)
model <- get_trialr_efftox(real_doses = real_doses,
  efficacy_hurdle = 0.5, toxicity_hurdle = 0.3,
  e = 0.1, t = 0.1,
  e0 = 0.5, tox1 = 0.65,
  e_star = 0.7, tox_star = 0.25,
  priors = p, iter = 1000, chains = 1, seed = 2020)
x <- model %>% fit('1N 2E 3B')
prob_tox_exceeds(x, threshold = 0.45)

prob_tox_quantile

Quantile of the toxicity rate at each dose.

Description

Get the estimated quantile of the toxicity rate at each dose under investigation. This is a set of modelled statistics. The underlying models estimate toxicity probabilities in different ways. If no model-based estimate of the median is available, this function will return a vector of NAs.

Usage

prob_tox_quantile(x, p, ...)

Arguments

x Object of class selector
p quantile probability, decimal value between 0 and 1
... arguments passed to other methods
Value

a numerical vector

Examples

```r
# CRM example
skeleton <- c(0.05, 0.1, 0.25, 0.4, 0.6)
target <- 0.25
outcomes <- '1NNN 2NTN'
fit <- get_dfcrm(skeleton = skeleton, target = target) %>% fit(outcomes)
fit %>% prob_tox_quantile(p = 0.9)
```

**prob_tox_samples**

Get samples of the probability of toxicity.

Description

Get samples of the probability of toxicity. For instance, a Bayesian approach that supports sampling would be expected to return posterior samples of the probability of toxicity. If this class does not support sampling, this function will raise an error. You can check whether this class supports sampling by calling `supports_sampling`.

Get samples of the probability of efficacy. For instance, a Bayesian approach that supports sampling would be expected to return posterior samples of the probability of toxicity. If this class does not support sampling, this function will raise an error. You can check whether this class supports sampling by calling `supports_sampling`.

Usage

```r
prob_tox_samples(x, tall = FALSE, ...)
prob_eff_samples(x, tall = FALSE, ...)
```

Arguments

- `x` Object of type `selector`
- `tall` logical, if FALSE, a wide data-frame is returned with columns pertaining to the doses and column names the dose indices. If TRUE, a tall data-frame is returned with data for all doses stacked vertically. In this mode, column names will include `dose` and `prob_eff`.
- `...` arguments passed to other methods

Value

data-frame like object

data-frame like object
### Examples

```r
# CRM example
skeleton <- c(0.05, 0.1, 0.25, 0.4, 0.6)
target <- 0.25
outcomes <- '1NNN 2TN'
fit <- get_dfcrm(skeleton = skeleton, target = target) %>% fit(outcomes)
fit %>% prob_tox_samples()
fit %>% prob_tox_samples(tall = TRUE)
efftox_priors <- trialr::efftox_priors
p <- efftox_priors(alpha_mean = -7.9593, alpha_sd = 3.5487,
                   beta_mean = 1.5482, beta_sd = 3.5018,
                   gamma_mean = 0.7367, gamma_sd = 2.5423,
                   zeta_mean = 3.4181, zeta_sd = 2.4406,
                   eta_mean = 0, eta_sd = 0.2,
                   psi_mean = 0, psi_sd = 1)
real_doses = c(1.0, 2.0, 4.0, 6.6, 10.0)
model <- get_trialr_efftox(real_doses = real_doses,
                            efficacy_hurdle = 0.5, toxicity_hurdle = 0.3,
                            p_e = 0.1, p_t = 0.1,
                            eff0 = 0.5, tox1 = 0.65,
                            eff_star = 0.7, tox_star = 0.25,
                            priors = p, iter = 1000, chains = 1, seed = 2020)
x <- model %>% fit('1N 2E 3B')
prob_tox_samples(x, tall = TRUE)
```

---

**recommended_dose**

Recommended dose for next patient or cohort.

### Description

Get the dose recommended for the next patient or cohort in a dose-finding trial.

### Usage

```r
recommended_dose(x, ...)
```

### Arguments

- **x** Object of type `selector`.
- **...** Extra args are passed onwards.

### Value

integer
Examples

```r
skeleton <- c(0.05, 0.1, 0.25, 0.4, 0.6)
target <- 0.25
model <- get_dfcrm(skeleton = skeleton, target = target)
fit <- model %>% fit('1NN 2NTN')
fit %>% recommended_dose()
```

selector

Dose selector.

Description

This is a core class in this package. It encapsulates that an object (e.g. a CRM model, a 3+3 model) is able to recommend doses, keep track of how many patients have been treated at what doses, what toxicity outcomes have been seen, and whether a trial should continue. It offers a consistent interface to many dose-finding methods, including CRM, TPI, mTPI, BOIN, EffTox, 3+3, and more.

Once you have a standardised interface, modularisation offers a powerful way to adorn dose-finding methods with extra desirable behaviour. selector objects can be daisy-chained together using magrittr’s pipe operator. For instance, the CRM fitting method in dfcrm is fantastic because it runs quickly and is simple to call. However, it does not recommend that a trial stops if a dose is too toxic or if n patients have already been treated at the recommended dose. Each of these behaviours can be bolted on via additional selectors. Furthermore, those behaviours and more can be bolted on to any dose selector because of the modular approach implemented in escalation. See Examples.

selector objects are obtained by calling the fit function on a selector_factory object. A selector_factory object is obtained by initially calling a function like get_dfcrm, get_three_plus_three or get_boin. Users may then add desired extra behaviour with subsequent calls to functions like stop_when_n_at_dose or stop_when_too_toxic.

The selector class also supports that an object will be able to perform inferential calculations on the rates of toxicity via functions like mean_prob_tox, median_prob_tox, and prob_tox_exceeds. However, naturally the sophistication of those calculations will vary by model implementation. For example, a full MCMC method will be able to quantify any probability you like by working with posterior samples. In contrast, a method like the crm function in dfcrm that uses the plug-in method to estimate posterior dose-toxicity curves cannot natively estimate the median probability of tox.

Usage

```r
selector()
```

Details

Every selector object implements the following functions:

- tox_target
- num_patients
- cohort
- doses_given
selector

- tox
- num_tox
- model_frame
- num_doses
- dose_indices
- recommended_dose
- continue
- n_at_dose
- n_at_recommended_dose
- is_randomising
- prob_administer
- tox_at_dose
- empiric_tox_rate
- mean_prob_tox
- median_prob_tox
- dose_admissible
- prob_tox_quantile
- prob_tox_exceeds
- supports_sampling
- prob_tox_samples

Some selectors also add:

- tox_limit
- eff_limit
- eff
- num_eff
- eff_at_dose
- empiric_eff_rate
- mean_prob_eff
- median_prob_eff
- prob_eff_quantile
- prob_eff_exceeds
- prob_eff_samples

See Also

selector_factory
Examples

# Start with a simple CRM model
skeleton <- c(0.05, 0.1, 0.25, 0.4, 0.6)
target <- 0.25
model1 <- get_dfcrm(skeleton = skeleton, target = target)

# Add a rule to stop when 9 patients are treated at the recommended dose
model2 <- get_dfcrm(skeleton = skeleton, target = target) %>%
  stop_when_n_at_dose(n = 9, dose = 'recommended')

# Add a rule to stop if toxicity rate at lowest dose likely exceeds target
model3 <- get_dfcrm(skeleton = skeleton, target = target) %>%
  stop_when_n_at_dose(n = 9, dose = 'recommended') %>%
  stop_when_too_toxic(dose = 1, tox_threshold = target, confidence = 0.5)

# We now have three CRM models that differ in their stopping behaviour.
# Let's fit each to some outcomes to see those differences:

outcomes <- '1NNN 2NTT 1NNT'
fit1 <- model1 %>% fit(outcomes)
fit2 <- model2 %>% fit(outcomes)
fit3 <- model3 %>% fit(outcomes)

fit1 %>% recommended_dose()
fit1 %>% continue()

fit2 %>% recommended_dose()
fit2 %>% continue()

fit3 %>% recommended_dose()
fit3 %>% continue()

# Already model3 wants to stop because of excessive toxicity.

# Let's carry on with models 1 and 2 by adding another cohort:

outcomes <- '1NNN 2NTT 1NNT 1NNN'
fit1 <- model1 %>% fit(outcomes)
fit2 <- model2 %>% fit(outcomes)

fit1 %>% recommended_dose()
fit1 %>% continue()

fit2 %>% recommended_dose()
fit2 %>% continue()

# Model1 wants to continue - in fact it will never stop.
# In contrast, model2 has seen 9 at dose 1 so, rather than suggest dose 1
# again, it suggests the trial should stop.

# For contrast, let us consider a BOIN model on the same outcomes
boin_fitter <- get_boin(num_doses = length(skeleton), target = target)
fit4 <- boin_fitter %>% fit(outcomes)
fit4 %>% recommended_dose()
fit4 %>% continue()

# Full selector interface:
fit <- fit2
fit %>% tox_target()
fit %>% num_patients()
fit %>% cohort()
fit %>% doses_given()
fit %>% tox()
fit %>% num_tox()
fit %>% model_frame()
fit %>% num_doses()
fit %>% dose_indices()
fit %>% recommended_dose()
fit %>% continue()
fit %>% n_at_dose()
fit %>% n_at_recommended_dose()
fit %>% is_randomising()
fit %>% prob_administer()
fit %>% tox_at_dose()
fit %>% empiric_tox_rate()
fit %>% mean_prob_tox()
fit %>% median_prob_tox()
fit %>% dose_admissible()
fit %>% prob_tox_quantile(0.9)
fit %>% prob_tox_exceeds(0.5)
fit %>% supports_sampling()
fit %>% prob_tox_samples()

selector_factory

Dose selector factory.

Description
Along with selector, this is the second core class in the escalation package. It exists to do one thing: fit outcomes from dose-finding trials to the models we use to select doses.

A selector_factory object is obtained by initially calling a function like get_dfcrm, get_three_plus_three or get_boin. Users may then add desired extra behaviour with subsequent calls to functions like stop_when_n_at_dose or stop_when_too_toxic. selector objects are obtained by calling the fit function on a selector_factory object. Refer to examples to see how this works.

Usage
selector_factory()

See Also
selector
Examples

# Start with a simple CRM model
skeleton <- c(0.05, 0.1, 0.25, 0.4, 0.6)
target <- 0.25
model1 <- get_dfcrm(skeleton = skeleton, target = target)

# Add a rule to stop when 9 patients are treated at the recommended dose
model2 <- get_dfcrm(skeleton = skeleton, target = target) %>%
  stop_when_n_at_dose(n = 9, dose = 'recommended')

# Add a rule to stop if toxicity rate at lowest dose likely exceeds target
model3 <- get_dfcrm(skeleton = skeleton, target = target) %>%
  stop_when_n_at_dose(n = 9, dose = 'recommended') %>%
  stop_when_too_toxic(dose = 1, tox_threshold = target, confidence = 0.5)

# We now have three CRM models that differ in their stopping behaviour.
# Let’s fit each to some outcomes to see those differences:
outcomes <- '1NNN 2NTT 1NNT'
fit1 <- model1 %>% fit(outcomes)
fit2 <- model2 %>% fit(outcomes)
fit3 <- model3 %>% fit(outcomes)

fit1 %>% recommended_dose()
fit1 %>% continue()

fit2 %>% recommended_dose()
fit2 %>% continue()

fit3 %>% recommended_dose()
fit3 %>% continue()
# Already model3 wants to stop because of excessive toxicity.

# Let’s carry on with models 1 and 2 by adding another cohort:
outcomes <- '1NNN 2NTT 1NNT 1NNN'
fit1 <- model1 %>% fit(outcomes)
fit2 <- model2 %>% fit(outcomes)

fit1 %>% recommended_dose()
fit1 %>% continue()

fit2 %>% recommended_dose()
fit2 %>% continue()
# Model1 wants to continue - in fact it will never stop.
# In contrast, model2 has seen 9 at dose 1 so, rather than suggest dose 1
# again, it suggests the trial should stop.

# For contrast, let us consider a BOIN model on the same outcomes
boin_fitter <- get_boin(num_doses = length(skeleton), target = target)
fit4 <- boin_fitter %>% fit(outcomes)
fit4 %>% recommended_dose()
fit4 %>% continue()
Examples

```r
# This class is intended to make the final dose selection in a BOIN trial:
target <- 0.25
model <- get_boin(num_doses = 5, target = target) %>%
  stop_at_n(n = 12) %>%
  select_boin_mtd()

outcomes <- '1NNN 2NTN 2NNN 3NTT'
model %>% fit(outcomes) %>% recommended_dose()

# However, we can use this method to select dose at every dose decision:
model2 <- get_boin(num_doses = 5, target = target) %>%
  select_boin_mtd(when = 'always')
model2 %>% fit('1NNT') %>% recommended_dose()
model2 %>% fit('1NNN 2NNT') %>% recommended_dose()

# and with any underlying model:
skeleton <- c(0.05, 0.1, 0.25, 0.4, 0.6)
model3 <- get_dfcrm(skeleton = skeleton, target = target) %>%
  select_boin_mtd(when = 'always')
model3 %>% fit('1NNT') %>% recommended_dose()
model3 %>% fit('1NNN 2NNT') %>% recommended_dose()
```

select_dose_by_cibp

Select dose by the CIBP selection criterion.

Description

This method selects dose by the convex infinite bounds penalisation (CIBP) criterion of Mozgunov & Jaki. Their method is mindful of the uncertainty in the estimates of the probability of toxicity and uses an asymmetry parameter to penalise escalation to risky doses.

Usage

`select_dose_by_cibp(parent_selector_factory, a, target = NULL)`

Arguments

- `parent_selector_factory`:
  Object of type `selector_factory`.
- `a`:
  Number between 0 and 2, the asymmetry parameter. See References.
- `target`:
  We seek a dose with this probability of toxicity. If not provided, the value will be sought from the parent dose-selector.

Value

An object of type `selector_factory` that can fit a dose-finding model to outcomes.
References


Examples

skeleton <- c(0.05, 0.1, 0.25, 0.4, 0.6)
target <- 0.33

# Let’s compare escalation behaviour of a CRM model without CIBP criterion:
model1 <- get_dfcrm(skeleton = skeleton, target = target)
# To one with the CIBP criterion:
model2 <- get_dfcrm(skeleton = skeleton, target = target) %>%
  select_dose_by_cibp(a = 0.3)

# Despite one-in-three tox at first dose, regular model is ready to escalate:
model1 %>% fit('1NTN') %>% recommended_dose()
# But the model using CIBP is more risk averse:
model2 %>% fit('1NTN') %>% recommended_dose()

---

simulate_trials

Simulate clinical trials.

Description

This function takes a selector_factory, such as that returned by get_dfcrm, get_boin or get_three_plus_three, and conducts many notional clinical trials. We conduct simulations to learn about the operating characteristics of adaptive trial designs.

Usage

simulate_trials(
  selector_factory,
  num_sims,
  true_prob_tox,
  true_prob_eff = NULL,
  ...
)

Arguments

selector_factory
  Object of type selector_factory.
num_sims
  integer, number of trial iterations to simulate.
ture_prob_tox
  numeric vector of true but unknown toxicity probabilities
true_prob_eff
  numeric vector of true but unknown efficacy probabilities. NULL if efficacy not analysed.
...
  Extra args are passed onwards.
Details

By default, dose decisions in simulated trials are made after each cohort of 3 patients. This can be changed by providing a function that simulates the arrival of new patients. The new patients will be added to the existing patients and the model will be fit to the set of all patients. The function that simulates patient arrivals should take as a single parameter a data-frame with one row for each existing patient and columns including cohort, patient, dose, tox, time (and possibly also eff and weight, if a phase I/II or time-to-event method is used). The provision of data on the existing patients allows the patient sampling function to be adaptive. The function should return a data-frame with a row for each new patient and a column for time_delta, the time between the arrival of this patient and the previous, as in cohorts_of_n. See Examples.

This method can simulate the culmination of trials that are partly completed. We just have to specify the outcomes already observed via the previous_outcomes parameter. Each simulated trial will commence from those outcomes seen thus far. See Examples.

We can specify the immediate next dose by specifying next_dose. If omitted, the next dose is calculated by invoking the model on the outcomes seen thus far.

Designs must eventually choose to stop the trial. However, some selectors like those derived from get_dfcrm offer no default stopping method. You may need to append stopping behaviour to your selector via something like stop_at_n or stop_when_n_at_dose, etc. To safeguard against simulating runaway trials that never end, the function will halt a simulated trial after 30 invocations of the dose-selection decision. To breach this limit, specify i_like_big_trials = TRUE in the function call. However, when you forego the safety net, the onus is on you to write selectors that will eventually stop the trial! See Examples.

The model is fit to the prevailing data at each dose selection point. By default, only the final model fit for each simulated trial is retained. This is done to conserve memory. With a high number of simulated trials, storing many model fits per trial may cause the executing machine to run out of memory. However, you can force this method to retain all model fits by specifying return_all_fits = TRUE. See Examples.

Value

Object of type simulations.

See Also

simulations
selector_factory
get_dfcrm
get_boin
get_three_plus_three
cohorts_of_n

Examples

t_true_prob_tox <- c(0.12, 0.27, 0.44, 0.53, 0.57)

# Regular usage examples, for 3+3:
sims <- get_three_plus_three(num_doses = 5) %>%
  simulate_trials(num_sims = 10, true_prob_tox = true_prob_tox)
# and continual reassessment method:
skeleton <- c(0.05, 0.1, 0.25, 0.4, 0.6)
target <- 0.25
sims <- get_dfcrm(skeleton = skeleton, target = target) %>%
  stop_at_n(n = 12) %>%
  simulate_trials(num_sims = 10, true_prob_tox = true_prob_tox)

# Lots of useful information is contained in the returned object:
sims %>% num_patients()
sims %>% num_doses()
sims %>% dose_indices()
sims %>% n_at_dose()
sims %>% n_at_recommended_dose()
sims %>% tox_at_dose()
sims %>% num_tox()
sims %>% recommended_dose()
sims %>% prob_administer()
sims %>% prob_recommend()
sims %>% trial_duration()

# By default, dose decisions are made after each cohort of 3 patients. See
# Details. To override, specify an alternative function via the
# sample_patient_arrivals parameter. E.g. to use cohorts of 2, we run:
sample_patient_arrivals_func <- function(current_data) cohorts_of_n(n = 2)
sims <- get_dfcrm(skeleton = skeleton, target = target) %>%
  stop_at_n(n = 12) %>%
  simulate_trials(num_sims = 10, true_prob_tox = true_prob_tox,
  sample_patient_arrivals = sample_patient_arrivals_func)

# To simulate the culmination of trials that are partly completed, specify
# the outcomes already observed via the previous_outcomes parameter:
sims <- get_dfcrm(skeleton = skeleton, target = target) %>%
  stop_at_n(n = 12) %>%
  simulate_trials(num_sims = 10, true_prob_tox = true_prob_tox,previous_outcomes = '1NTN')

# Outcomes can be described by above outcome string method or data-frame:
previous_outcomes <- data.frame(
  patient = 1:3,
  cohort = c(1, 1, 1),
  tox = c(0, 1, 0),
  dose = c(1, 1, 1)
)
sims <- get_dfcrm(skeleton = skeleton, target = target) %>%
  stop_at_n(n = 12) %>%
  simulate_trials(num_sims = 10, true_prob_tox = true_prob_tox,previous_outcomes = previous_outcomes)

# We can specify the immediate next dose:
sims <- get_dfcrm(skeleton = skeleton, target = target) %>%
  stop_at_n(n = 12) %>%
  simulate_trials(num_sims = 10, true_prob_tox = true_prob_tox,
next_dose = 5)

# By default, the method will stop simulated trials after 30 dose selections.
# To suppress this, specify i_like_big_trials = TRUE. However, please take
# care to specify selectors that will eventually stop!
sims <- get_dfcrm(skeleton = skeleton, target = target) %>%
  stop_at_n(n = 99) %>%
simulate_trials(num_sims = 1, true_prob_tox = true_prob_tox,
    i_like_big_trials = TRUE)

# By default, only the final model fit is retained for each simulated trial.
# To retain all interim model fits, specify return_all_fits = TRUE.
sims <- get_three_plus_three(num_doses = 5) %>%
simulate_trials(num_sims = 10, true_prob_tox = true_prob_tox,
    return_all_fits = TRUE)

# Verify that there are now many analyses per trial with:
sapply(sims$fits, length)

---

**simulations**

*Simulated trials.*

**Description**

This class encapsulates that many notional or virtual trials can be simulated. Each recommends a
dose (or doses), keeps track of how many patients have been treated at what doses, what toxicity
outcomes have been seen, and whether a trial advocates continuing, etc. We run simulations to learn
about the operating characteristics of a trial design.

Computationally, the simulations class supports much of the same interface as selector, and a
little more. Thus, many of the same generic functions are supported - see Examples. However,
compared to selectors, the returned objects reflect that there are many trials instead of one, e.g.
num_patients(sims), returns as an integer vector the number of patients used in the simulated
trials.

**Usage**

```r
simulations(fits, true_prob_tox, true_prob_eff = NULL, ...)
```

**Arguments**

- `fits` Simulated model fits, arranged as list of lists.
- `true_prob_tox` vector of true toxicity probabilities
- `true_prob_eff` vector of true efficacy probabilities, optionally NULL if efficacy not analysed.
- `...` Extra args
Details

The simulations object implements the following functions:

- `num_patients`
- `num_doses`
- `dose_indices`
- `n_at_dose`
- `tox_at_dose`
- `num_tox`
- `recommended_dose`
- `prob_administer`
- `prob_recommend`
- `trial_duration`

Value

list with slots: `fits` containing model fits; and `true_prob_tox`, containing the assumed true probability of toxicity.

See Also

`selector`

`simulate_trials`

Examples

# Simulate performance of the 3+3 design:
true_prob_tox <- c(0.12, 0.27, 0.44, 0.53, 0.57)
sims <- get_three_plus_three(num_doses = 5) %>%
  simulate_trials(num_sims = 10, true_prob_tox = true_prob_tox)
# The returned object has type 'simulations'. The supported interface is:
sims %>% num_patients()
sims %>% num_doses()
sims %>% dose_indices()
sims %>% n_at_dose()
sims %>% tox_at_dose()
sims %>% num_tox()
sims %>% recommended_dose()
sims %>% prob_administer()
sims %>% prob_recommend()
sims %>% trial_duration()

# Access the list of model fits for the ith simulated trial using:
i <- 1
sims$fits[[i]]
# and the jth model fit for the ith simulated trial using:
j <- 1
sims$fits[[i]][[j]]
```r
sims$fits[[i]][[j]]
# and so on.
```

---

**simulation_function**  
*Get function for simulating trials.*

**Description**

This function does not need to be called by users. It is used internally.

**Usage**

```r
simulation_function(selector_factory)
```

**Arguments**

- `selector_factory`: Object of type `selector_factory`.

**Value**

A function.

---

**spread_paths**  
*Spread the information in dose_finding_paths object to a wide data.frame format.*

**Description**

Spread the information in dose_finding_paths object to a wide data.frame format.

**Usage**

```r
spread_paths(df = NULL, dose_finding_paths = NULL, max_depth = NULL)
```

**Arguments**

- `df`: Optional data.frame like that returned by `as_tibble(dose_finding_paths)`. Columns `.depth`, `.node`, `.parent` are required. All other columns are spread with a suffix reflecting depth.
- `dose_finding_paths`: Optional instance of dose_finding_paths. Required if `df` is null.
- `max_depth`: integer, maximum depth of paths to traverse.

**Value**

A data.frame
stop_at_n

Examples

```r
## Not run:
# Calculate paths for the first two cohorts of three patients a CRM trial
skeleton <- c(0.05, 0.1, 0.25, 0.4, 0.6)
target <- 0.25
cohort_sizes <- c(3, 3)
paths <- get_dfcrm(skeleton = skeleton, target = target) %>%
  get_dose_paths(cohort_sizes = cohort_sizes)

## End(Not run)
```

Description

This function adds a restriction to stop a trial when \( n \) patients have been evaluated. It does this by adding together the number of patients treated at all doses and stopping when that total exceeds \( n \).

Dose selectors are designed to be daisy-chained together to achieve different behaviours. This class is a **greedy** selector, meaning that it prioritises its own behaviour over the behaviour of other selectors in the chain. That is, it will advocate stopping when the condition has been met, even if the selectors further up the chain would advocate to keep going. In can be interpreted as an overriding selector. This allows the decision to stop to be executed as soon as it is warranted. Be aware though, that there are other selectors that can be placed after this class that will override the stopping behaviour. See Examples.

Usage

```r
stop_at_n(parent_selector_factory, n)
```

Arguments

- `parent_selector_factory`
  - Object of type `selector_factory`.
- `n`
  - Stop when there are this many patients.

Value

An object of type `selector_factory` that can fit a dose-finding model to outcomes.

Examples

```r
skeleton <- c(0.05, 0.1, 0.25, 0.4, 0.6)
target <- 0.25

# Create CRM model that will stop when 15 patients are evaluated:
model1 <- get_dfcrm(skeleton = skeleton, target = target) %>%
  stop_at_n(n = 15)
```
stop_when_n_at_dose

Description

This method stops a dose-finding trial when there are n patients at a dose. It can stop when the rule is triggered at the recommended dose, at a particular dose, or at any dose.
Usage

\texttt{stop\_when\_n\_at\_dose(parent\_selector\_factory, n, dose)}

Arguments

\begin{itemize}
\item \texttt{parent\_selector\_factory} \hspace{1em} Object of type \texttt{selector\_factory}.
\item \texttt{n} \hspace{1em} Stop when there are \textit{n} at a dose.
\item \texttt{dose} \hspace{1em} 'any' to stop when there are \textit{n} at any dose; 'recommended' to stop when there are \textit{n} at the recommended dose; or an integer to stop when there are \textit{n} at a particular dose-level.
\end{itemize}

Value

an object of type \texttt{selector\_factory} that can fit a dose-finding model to outcomes.

Examples

```
skeleton <- c(0.05, 0.1, 0.25, 0.4, 0.6)
target <- 0.25

# This model will stop when 12 are seen at any dose:
model1 <- get_dfcrm(skeleton = skeleton, target = target) %>%
  stop\_when\_n\_at\_dose(n = 12, dose = 'any')

# This model fit will not stop:
model1 %>% fit('1NNN 2NTN 2TNN 2NNN') %>% continue()
# But this model fit will stop:
model1 %>% fit('1NNN 2NTN 2TNN 2NNN 2NTT') %>% continue()

# This model will stop when 12 are seen at the recommended dose:
model2 <- get_dfcrm(skeleton = skeleton, target = target) %>%
  stop\_when\_n\_at\_dose(n = 12, dose = 'recommended')

# This model fit will not stop:
fit2 <- model2 %>% fit('1NNN 2NTN 2TNN 2NNN')
fit2 %>% recommended\_dose()
fit2 %>% continue()
# But this model fit will stop:
fit3 <- model2 %>% fit('1NNN 2NTN 2TNN 2NNN 2NTT')
fit3 %>% recommended\_dose()
fit3 %>% continue()
```

stop\_when\_too\_toxic \hspace{1em} \textit{Stop when a dose is too toxic.}
**Description**

This method stops a dose-finding trial when sufficient probabilistic confidence is reached that the rate of toxicity at a dose exceeds some threshold. In other words, it stops when it is likely that a dose is too toxic. It can stop when the rule is triggered at the recommended dose, at a particular dose, or at any dose. See Details.

**Usage**

```
stop_when_too_toxic(parent_selector_factory, dose, tox_threshold, confidence)
```

**Arguments**

- `parent_selector_factory`: Object of type `selector_factory`.
- `dose`: 'any' to stop when any dose is too toxic; 'recommended' to stop when the recommended dose is too toxic; or an integer to stop when a particular dose-level is too toxic.
- `tox_threshold`: We are interested in toxicity probabilities greater than this threshold.
- `confidence`: Stop when there is this much total probability mass supporting that the toxicity rate exceeds the threshold.

**Details**

The method for calculating probability mass for toxicity rates will ultimately be determined by the dose-finding model used and the attendant inferential mechanism. For instance, the `crm` function in the dfcrm package calculates the posterior expected mean and variance of the slope parameter in a CRM model. It does not use MCMC to draw samples from the posterior distribution. Thus, to perform inference on the posterior probability of toxicity, this package assumes the dfcrm slope parameter follows a normal distribution with the mean and variance calculated by dfcrm. In contrast, the `stan_crm` function in the trialr package needs no such assumption because it samples from the posterior parameter distribution and uses those samples to infer on the posterior probability of toxicity at each dose, dependent on the chosen model for the dose-toxicity curve.

**Value**

an object of type `selector_factory` that can fit a dose-finding model to outcomes.

**Examples**

```r
skeleton <- c(0.05, 0.1, 0.25, 0.4, 0.6)
target <- 0.25

# We compare a CRM model without a toxicity stopping rule to one with it:
model1 <- get_dfcrm(skeleton = skeleton, target = target)
model2 <- get_dfcrm(skeleton = skeleton, target = target) %>%
  stop_when_too_toxic(dose = 'any', tox_threshold = 0.5, confidence = 0.7)

outcomes <- '1NNN 2NNN 3NNT 3NNN 3TNT 2NNN'
fit1 <- model1 %>% fit(outcomes)
```
```r
fit2 <- model2 %>% fit(outcomes)

# Naturally the first does not advocate stopping:
fit1 %>% recommended_dose()
fit1 %>% continue()

# However, after the material toxicity at dose 3, the rule is fired:
fit2 %>% recommended_dose()
fit2 %>% continue()
# To verify the requirement to stop, let's calculate the probability that the
# toxicity rate exceeds 50%
fit2 %>% prob_tox_exceeds(0.5)
```

---

**stop_when_tox_ci_covered**

*Stop when uncertainty interval of prob tox is covered.*

**Description**

This method stops a dose-finding trial when the symmetric uncertainty interval for the probability of toxicity falls within a range. This allows trials to be stopped when sufficient precision on the probability of toxicity has been achieved. See Details.

**Usage**

```r
stop_when_tox_ci_covered(
  parent_selector_factory, 
  dose, 
  lower, 
  upper, 
  width = 0.9 
)
```

**Arguments**

- `parent_selector_factory`
  - Object of type `selector_factory`.
- `dose`
  - 'any' to stop when the interval for any dose is covered; 'recommended' to stop when the interval for the recommended dose is covered; or an integer to stop when the interval for a particular dose-level is covered.
- `lower`
  - Stop when lower interval bound exceeds this value
- `upper`
  - Stop when upper interval bound is less than this value
- `width`
  - Width of the uncertainty interval. Default is 0.9, i.e. a range from the 5th to the 95th percentiles.
Details

The method for calculating probability mass for toxicity rates will ultimately be determined by the
dose-finding model used and the attendant inferential mechanism. For instance, the `crm` function
in the `dfcrm` package calculates the posterior expected mean and variance of the slope parameter
in a CRM model. It does not use MCMC to draw samples from the posterior distribution. Thus,
to perform inference on the posterior probability of toxicity, this package assumes the `dfcrm` slope
parameter follows a normal distribution with the mean and variance calculated by `dfcrm`. In contrast,
the `stan_crm` function in the `trialr` package needs no such assumption because it samples from
the posterior parameter distribution and uses those samples to infer on the posterior probability of
toxicity at each dose, dependent on the chosen model for the dose-toxicity curve.

Value

an object of type `selector_factory` that can fit a dose-finding model to outcomes.

Examples

```r
skeleton <- c(0.05, 0.1, 0.25, 0.4, 0.6)
target <- 0.25

# We compare a CRM model without this stopping rule:
model1 <- get_dfcrm(skeleton = skeleton, target = target)
# To two with it, the first demanding a relatively tight CI:
model2 <- get_dfcrm(skeleton = skeleton, target = target) %>%
  stop_when_tox_ci_covered(dose = 'recommended', lower = 0.15, upper = 0.35)
# and the second demanding a relatively loose CI:
model3 <- get_dfcrm(skeleton = skeleton, target = target) %>%
  stop_when_tox_ci_covered(dose = 'recommended', lower = 0.05, upper = 0.45)

outcomes <- '1NNN 2NNN 3NNT 3NNN 3TNT 2NNN'
fit1 <- model1 %>% fit(outcomes)
fit2 <- model2 %>% fit(outcomes)
fit3 <- model3 %>% fit(outcomes)

# Naturally the first does not advocate stopping:
fit1 %>% recommended_dose()
fit1 %>% continue()

# The second does not advocate stopping either:
fit2 %>% recommended_dose()
fit2 %>% continue()

# This is because the CI is too wide:
fit2 %>% prob_tox_quantile(p = 0.05)
fit2 %>% prob_tox_quantile(p = 0.95)

# However, the third design advocates stopping because the CI at the
# recommended dose is covered:
fit3 %>% recommended_dose()
fit3 %>% continue()

# To verify the veracity, inspect the quantiles:
fit3 %>% prob_tox_quantile(p = 0.05)
```
supports_sampling

Does this selector support sampling of outcomes?

Description

Learn whether this selector supports sampling of outcomes. For instance, is it possible to get posterior samples of the probability of toxicity at each dose? If true, prob_tox_samples will return a data-frame of samples.

Usage

supports_sampling(x, ...)

Arguments

x  Object of type selector

... arguments passed to other methods

Value

logical

Examples

# CRM example
skeleton <- c(0.05, 0.1, 0.25, 0.4, 0.6)
target <- 0.25
outcomes <- c('1NNN', '2NTN')
fit <- get_dfcrm(skeleton = skeleton, target = target) %>% fit(outcomes)
fit %>% supports_sampling()

three_plus_three

Fit the 3+3 model to some outcomes.

Description

Fit the 3+3 model to some outcomes.

Usage

three_plus_three(
  outcomes,
  num_doses,
  allow_deescalate = FALSE,
  strict_mode = TRUE
)
Arguments

- **outcomes**: Outcomes observed. See `parse_phase1_outcomes`.
- **num_doses**: Number of doses under investigation.
- **allow_deescalate**: TRUE to allow de-escalation, as described by Korn et al. Default is FALSE.
- **strict_mode**: TRUE to raise errors if it is detected that the 3+3 algorithm has not been followed.

Value

A list containing recommended_dose and a logical value indicating whether the trial should continue.

References


Examples

```r
three_plus_three('2NNN 3NN', num_doses = 7)
```

---

**tox**

*Binary toxicity outcomes.*

Description

Get a vector of the binary toxicity outcomes for evaluated patients.

Usage

```r
tox(x, ...)
```

Arguments

- **x**: Object of type `selector`.
- **...**: Extra args are passed onwards.

Value

An integer vector
Examples

```r
skeleton <- c(0.05, 0.1, 0.25, 0.4, 0.6)
target <- 0.25
model <- get_dfcrm(skeleton = skeleton, target = target)
fit <- model %>% fit('1NNN 2NTN')
fit %>% tox()
```

Description

Get the number of toxicities seen at each dose under investigation.

Usage

```r
tox_at_dose(x, ...)
```

Arguments

- `x` Object of class `selector`
- `...` arguments passed to other methods

Value

an integer vector

Examples

```r
# CRM example
skeleton <- c(0.05, 0.1, 0.25, 0.4, 0.6)
target <- 0.25
outcomes <- '1NNN 2NTN'
fit <- get_dfcrm(skeleton = skeleton, target = target) %>% fit(outcomes)
fit %>% tox_at_dose()
```
### tox_limit

**Toxicity rate limit**

#### Description

Get the maximum permissible toxicity rate, if supported. NULL if not.

#### Usage

```r
tox_limit(x, ...)
```

#### Arguments

- `x` Object of type `selector`.
- `...` Extra args are passed onwards.

#### Value

numeric

#### Examples

```r
efftox_priors <- trialr::efftox_priors
p <- efftox_priors(alpha_mean = -7.9593, alpha_sd = 3.5487, 
  beta_mean = 1.5482, beta_sd = 3.5018, 
  gamma_mean = 0.7367, gamma_sd = 2.5423, 
  zeta_mean = 3.4181, zeta_sd = 2.4406, 
  eta_mean = 0, eta_sd = 0.2, 
  psi_mean = 0, psi_sd = 1)
real_doses = c(1.0, 2.0, 4.0, 6.6, 10.0)
model <- get_trialr_efftox(real_doses = real_doses, 
  efficacy_hurdle = 0.5, toxicity_hurdle = 0.3, 
  p_e = 0.1, p_t = 0.1, 
  eff0 = 0.5, tox1 = 0.65, 
  eff_star = 0.7, tox_star = 0.25, 
  priors = p, iter = 1000, chains = 1, seed = 2020)
x <- model %>% fit('1N 2E 3B')
tox_limit(x)
```
### tox_target

**Target toxicity rate**

**Description**

Get the target toxicity rate, if supported. NULL if not.

**Usage**

```r
 tox_target(x, ...)
```

**Arguments**

- `x` Object of type `selector`.
- `...` Extra args are passed onwards.

**Value**

numeric

**Examples**

```r
 skeleton <- c(0.05, 0.1, 0.25, 0.4, 0.6)
 target <- 0.25
 model <- get_dfcrm(skeleton = skeleton, target = target)
 fit <- model %>% fit('1NNN 2NTN')
 fit %>% tox_target()
```

---

### trial_duration

**Duration of trials.**

**Description**

Get the length of time that trials take to recruit all patients.

**Usage**

```r
 trial_duration(x, ...)
```

**Arguments**

- `x` Object of type `simulations`.
- `...` arguments passed to other methods

**Value**

vector of numerical times
Examples

```r
true_prob_tox <- c(0.12, 0.27, 0.44, 0.53, 0.57)
sims <- get_three_plus_three(num_doses = 5) %>%
  simulate_trials(num_sims = 50, true_prob_tox = true_prob_tox)
sims %>% trial_duration
```

Description

This method continues a dose-finding trial until a safety dose has been given to \( n \) patients. Once that condition is met, it delegates dose selecting and stopping responsibility to its parent dose selector, whatever that might be. This class is greedy in that it meets its own needs before asking any other selectors higher in the chain what they want. Thus, different behaviours may be achieved by nesting dose selectors in different orders. See examples.

Usage

```r
try_rescue_dose(parent_selector_factory, n, dose)
```

Arguments

- `parent_selector_factory`:
  - Object of type `selector_factory`.
- `n`:
  - Continue at least until there are \( n \) at a dose.
- `dose`:
  - an integer to identify the sought rescue dose-level.

Value

an object of type `selector_factory` that can fit a dose-finding model to outcomes.

Examples

```r
skeleton <- c(0.05, 0.1, 0.25, 0.4, 0.6)
target <- 0.25

# This model will demand the lowest dose is tried in at least two patients
# before the trial is stopped for excess toxicity
model1 <- get_dfcrm(skeleton = skeleton, target = target) %>%
  stop_when_too_toxic(dose = 1, tox_threshold = 0.35, confidence = 0.8) %>%
  try_rescue_dose(dose = 1, n = 2)

# In contrast, this model will stop for excess toxicity without trying dose 1
model2 <- get_dfcrm(skeleton = skeleton, target = target) %>%
  stop_when_too_toxic(dose = 1, tox_threshold = 0.35, confidence = 0.8)

# For non-toxic outcomes, both designs will continue at sensible doses:
```
try_rescue_dose

```r
fit1 <- model1 %>% fit('2NNN')
fit1 %>% recommended_dose()
fit1 %>% continue()

fit2 <- model2 %>% fit('2NNN')
fit2 %>% recommended_dose()
fit2 %>% continue()

# For toxic outcomes, the design 1 will use dose 1 before stopping is allowed
fit1 <- model1 %>% fit('2TTT')
fit1 %>% recommended_dose()
fit1 %>% continue()

# For toxic outcomes, however, design 2 will stop despite dose 1 being
# untested:
fit2 <- model2 %>% fit('2TTT')
fit2 %>% recommended_dose()
fit2 %>% continue()

# After dose 1 is given the requisite number of times, dose recommendation
# and stopping revert to being determined by the underlying dose selector:
fit1 <- model1 %>% fit('2TTT 1T')
fit1 %>% recommended_dose()
fit1 %>% continue()

fit1 <- model1 %>% fit('2TTT 1TT')
fit1 %>% recommended_dose()
fit1 %>% continue()
```
Index

as_tibble.dose_paths, 4

calculate_probabilities, 5
cohort, 6, 58
cohorts_of_n, 6, 66
continue, 7, 59
crm, 22, 58, 74, 76

crystallised_dose_paths, 8
demand_n_at_dose, 9
dont_skip_doses, 10
dose_admissible, 12, 59
dose_indices, 13, 59, 69
dose_paths, 4, 5, 8, 13, 23, 37
dose_paths_function, 14
doses_given, 11, 58

eff, 14, 59
eff_at_dose, 15, 59
eff_limit, 16, 59

empiric_eff_rate, 17, 59
empiric_tox_rate, 17, 59

enforce_three_plus_three, 18
escalation (escalation-package), 4
escalation-package, 4

fit, 19, 58, 61
follow_path, 19, 21, 30, 34

get_boundary, 21
get_boin, 20, 58, 61, 63, 65, 66
get_dfcrm, 21, 34, 58, 61, 63, 66
get_dose_paths, 13, 23
get_empiric.crm_skeleton_weights, 24
get_mtpi, 24
get_random_selector, 26
get_three_plus_three, 27, 58, 61, 65, 66
get_tpi, 28
get_trialr.crm, 30
get_trialr_efftox, 32
get_trialr_nbg, 33

get_wages_and_tait, 35

graph_paths, 37

is_randomising, 38, 59

mean_prob_eff, 38, 59
mean_prob_tox, 39, 58, 59
median_prob_eff, 40, 59
median_prob_tox, 41, 58, 59
model_frame, 41, 59

n_at_dose, 46, 59, 69
n_at_recommended_dose, 47, 59
num_cohort_outcomes, 42
num_dose_path_nodes, 43
num_doses, 43, 59, 69
num_eff, 44, 59
num_patients, 45, 58, 69
num_tox, 45, 59, 69

parse_phase1_2_outcomes, 47
parse_phase1_outcomes, 18–20, 48, 78
phase1_2_outcomes_to_cohorts, 50
phase1_outcomes_to_cohorts, 51
prob_administer, 52, 59, 69
prob_eff_exceeds, 59
prob_eff_exceeds (prob_tox_exceeds), 54
prob_eff_quantile, 53, 59

prob_eff_samples, 59
prob_eff_samples (prob_tox_samples), 56
prob_recommend, 54, 69
prob_tox_exceeds, 54, 58, 59
prob_tox_quantile, 55, 59
prob_tox_samples, 56, 59

recommended_dose, 57, 59, 69

select.mtd, 63
select_boin_mtd, 63
select_dose_by_cibp, 64
INDEX

selector, 6, 7, 11–17, 19, 38–41, 43–47, 52–57, 58, 61, 68, 69, 77–81
selector_factory, 9, 10, 14, 19–23, 25, 27–30, 32–36, 58, 59, 61, 61, 63–66, 70, 71, 73–76, 82
simulate_trials, 65, 69
simulation_function, 70
simulations, 54, 66, 68, 81
spread_paths, 70
stan_crm, 30
stan_efftox, 33
stan_nbg, 34
stop_at_n, 7, 66, 71
stop_when_n_at_dose, 58, 61, 66, 72
stop_when_too_toxic, 7, 58, 61, 73
stop_when_tox_ci_covered, 75
supports_sampling, 56, 59, 77
three_plus_three, 77
tibble, 4, 42
tox, 59, 78
tox_at_dose, 59, 69, 79
tox_limit, 59, 80
tox_target, 58, 81
trial_duration, 69, 81
trialr, 33
try_rescue_dose, 82