Package ‘escalation’

March 8, 2020

Type     Package
Title    Modular Approach to Dose Finding Clinical Trials
Version  0.1.1
Date     2020-02-29
Maintainer Kristian Brock <kristian.brock@gmail.com>
Description Methods for working with dose-finding clinical trials. We start by providing a common interface to various dose-finding methodologies like the continual reassessment method (CRM) by O'Quigley et al. (1990) <doi:10.2307/2531628>, the Bayesian optimal interval design (BOIN) by Liu & Yuan (2015) <doi:10.1111/rssc.12089>, and the 3+3 described by Korn et al. (1994) <doi:10.1002/sim.4780131802>. We then add optional embellishments to provide extra desirable behaviour, like avoiding skipping doses, stopping after n patients have been treated at the recommended dose, or demanding that n patients are treated before stopping is allowed. By daisy-chaining together these embellishments using the pipe operator from ‘magrittr’, it is simple to tailor the behaviour of dose-finding designs so that they do what you want. Furthermore, using this flexible interface for creating dose-finding designs, it is simple to run simulations or calculate dose-pathways 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
RoxygenNote 7.0.2
Suggests testthat, knitr, rmarkdown, covr
VignetteBuilder knitr
NeedsCompilation no
Author  Kristian Brock [aut, cre] (<https://orcid.org/0000-0002-3921-0166>)
Repository CRAN
Date/Publication 2020-03-08 17:10:02 UTC
R topics documented:

cohort ................................................................. 3
continue ............................................................. 3
demand_n_at_dose .................................................. 4
dont_skip_doses ..................................................... 5
doses_given ........................................................ 7
dose_indices ........................................................ 7
empiric_tox_rate .................................................. 8
enforce_three_plus_three ....................................... 9
fit ................................................................. 9
follow_path .......................................................... 10
get_boin ........................................................... 11
get_dfcrm ........................................................... 12
get_three_plus_three ............................................ 13
mean_prob_tox .................................................... 14
median_prob_tox ................................................... 15
model_frame ....................................................... 16
num_doses .......................................................... 17
num_patients ....................................................... 17
num_tox ............................................................ 18
n_at_dose ........................................................... 18
parse_phase1_outcomes ........................................... 19
phase1_outcomes_to_cohorts .................................... 20
prob_administer .................................................... 21
prob_tox_exceeds .................................................. 22
prob_tox_quantile ................................................ 23
prob_tox_samples ............................................... 23
recommended_dose ................................................ 24
selector ............................................................ 25
selector_factory .................................................... 27
select_dose_by_cibp ............................................. 29
stop_at_n ........................................................... 30
stop_when_n_at_dose ............................................. 32
stop_when_too_toxic .............................................. 33
stop_when_tox_ci_covered ...................................... 34
supports_sampling ............................................... 36
three_plus_three ................................................. 37
tox ............................................................... 38
tox_at_dose ........................................................ 38
tox_target ........................................................ 39

Index .............................................................. 40
**cohort**

*Cohort numbers of evaluated patients.*

**Description**

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

**Usage**

```r
cohort(selector, ...)
```

**Arguments**

- `selector` 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 %>% cohort()
```

**continue**

*Should this dose-finding experiment continue?*

**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 characteristics and dose-pathways. You add stopping behaviours to designs using calls like `stop_at_n` and `stop_when_too_toxic`.

**Usage**

```r
continue(selector, ...)
```

**Arguments**

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

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()

demand_n_at_dose  Demand there are n patients at a dose before considering stopping.

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

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('1NN 1NNN 2TNN 2NNN') %>% continue()

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

# This model advocates stopping because 12 patients are seen in total:
model2 %>% fit('1NNT 1NNN 2TNN 2NNN') %>% continue()

# But this model advocates continuing because 9 patients have not been seen
# at any dose yet:
model3 %>% fit('1NNT 1NNN 2TNN 2NNN') %>% continue()

# This model advocates stopping because 12 patients are seen in total:
# 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('1NNT 1NNN 2TNN 2NNN 2TTN')
fit %>% continue()
fit %>% recommended_dose()
```

**don't_skip_doses**

*Prevent skipping of doses.*

**Description**

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

**Usage**

don't_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

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

model2 <- get_dfcrm(skeleton = skeleton, 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 = skeleton, target = target) %>%
dont_skip_doses(when_deescalating = TRUE)
fit1 <- model1 %>% fit('1NNN 2N 3TTT')

model2 <- get_dfcrm(skeleton = skeleton, 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(selector, ...)

**Arguments**

selector  
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 %>% doses_given()
```

**dose_indices**

**Dose indices**

**Description**

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

**Usage**

dose_indices(selector, ...)

**Arguments**

selector  
Object of type selector.

...  
Extra args are passed onwards.

**Value**

an integer vector
empiric_tox_rate

Examples

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

tabular_representation

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>empiric_tox_rate</td>
<td>Observed toxicity rate at each dose.</td>
</tr>
</tbody>
</table>

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

Arguments

selector  
Object of class `selector`

...  
arguments passed to other methods

Value

a numerical vector

Examples

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

```r
## 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

fit(selector_factory, outcomes, ...)

follow_path

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

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 %>% recommended_dose() # Etc

follow_pathFollow 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.
get_boin

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. This class recommends no dose now.

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, ...)

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 select.mtd.

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


Examples

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

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

get_dfcrm

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 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

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 tocrm.

Value

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

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

# But we can provide extra args to get_dfcrm that are then 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(c('1NN 2NN 3NN')) %>%
  get_dfcrm(skeleton = skeleton, target = target)

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

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

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

---

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

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()

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(selector, ...)

Arguments

selector 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 %>% mean_prob_tox()

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(selector, ...)

Arguments

selector 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 %>% 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

```r
model_frame(selector, ...)
```

Arguments

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

Value

`tibble`, which acts like a `data.frame`.

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 %>% model_frame()
```
**num_doses**

Number of doses.

**Description**

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

**Usage**

```
num_doses(selector, ...)
```

**Arguments**

- `selector` 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_patients**

Number of patients evaluated.

**Description**

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

**Usage**

```
num_patients(selector, ...)
```

**Arguments**

- `selector` 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 2TN')
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(selector, ...)
```

**Arguments**

- **selector**: 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 2TN')
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**

```r
n_at_dose(selector, ...)
```
### Arguments

- **selector**
  - 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 %>% n_at_dose()
```

### Description

Parse a string of phase I dose-finding outcomes to 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, `2NTT` 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, `2NTT 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`. These elements are congruent with those of the same name in `crm_params`, for example. If `as_list == FALSE`, a data.frame with columns `tox` and `doses`.
phase1_outcomes_to_cohorts

References


Examples

x = parse_phase1_outcomes('1NNN 2NTN 3TTT')
# Three cohorts of three patients. The first cohort was treated at dose 1 and
# non 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. See:
x$num_patients # 9
x$doses # c(1, 1, 1, 2, 2, 2, 3, 3, 3)
x$tox # c(0, 0, 0, 1, 0, 1, 1)
sum(x$tox) # 4

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

phase1_outcomes_to_cohorts

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

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 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, 2NTT 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, 2NTT 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

phase1_outcomes_to_cohorts(outcomes)
prob_administer

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 2NNT 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(selector, ...)
```

Arguments

selector 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 <- '1NNN 2NTN'
fit <- get_dfcrm(skeleton = skeleton, target = target) %>% fit(outcomes)
fit %>% prob_administer()

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.

Usage

prob_tox_exceeds(selector, threshold, ...)

Arguments

selector Object of type selector
threshold Probability that toxicity rate exceeds what?
... arguments passed to other methods

Value

umerical 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)
**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**

```r
prob_tox_quantile(selector, p, ...)  
```

**Arguments**

- `selector`: 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 <- c('NNN', 'NTN')  
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`.

**Usage**

```r
prob_tox_samples(selector, tall = FALSE, ...)
```
Arguments

selector   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_tox.

...        arguments passed to other methods

Value

data-frame like object

Examples

# 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)

---

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

recommended_dose(selector, ...)

Arguments

selector   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 2TN')
fit %>% recommended_dose()
**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 dose-finding methods from several packages, including dfcrm and BOIN. bcrm and trialr will be added.

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**

`selector()`

**Details**

Every selector object implements the following functions:

- `tox_target`
- `num_patients`
- `cohort`
- `doses_given`
- `tox`
- `num_tox`
- `model_frame`
- `num_doses`
- `recommended_dose`
- `continue`
- `n_at_dose`
- `dose_indices`
- `prob_administer`
- `tox_at_dose`
- `empiric_tox_rate`
- `mean_prob_tox`
- `median_prob_tox`
- `prob_tox_quantile`
- `prob_tox_exceeds`

**See Also**

`selector_factory`

**Examples**

```r
# 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 <- c('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 <- c('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 %>% prob_administer()
fit %>% tox_at_dose()
fit %>% empiric_tox_rate()
fit %>% mean_prob_tox()
fit %>% median_prob_tox()
fit %>% prob_tox_quantile(0.9)
fit %>% prob_tox_exceeds(0.5)
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:

```r
outcomes <- c('NNN', '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.

```r
# 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()
```

---

**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**

```r
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()

---

stop_at_n

Stop when there are n patients in total.

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

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)

# With 12 patients, this trial should not stop:
fit1 <- model1 %>% fit('1NNN 2NTN 2TNN 2NNN')
fit1 %>% recommended_dose()
fit1 %>% continue()

# With 15 patients, this trial should stop:
fit2 <- model1 %>% fit('1NNN 2NTN 2TNN 2NNN 2NTT')
fit2 %>% recommended_dose()
fit2 %>% continue()

# The stopping behaviour can be overruled by the order of selectors.
# In model2, demanding 9 at recommended dose will trump stopping at 12:
model2 <- get_dfcrm(skeleton = skeleton, target = target) %>%
  stop_at_n(n = 12) %>%
  demand_n_at_dose(dose = 'recommended', n = 9)

# In model3, stopping at 12 will trump demanding 9 at recommended dose:
model3 <- get_dfcrm(skeleton = skeleton, target = target) %>%
  demand_n_at_dose(dose = 'recommended', n = 9) %>%
  stop_at_n(n = 12)

# This model will continue because 9 have not been seen at recommended dose.
fit3 <- model2 %>% fit('1NNN 2NNN 2NNN 3NNN')
fit3 %>% recommended_dose()
fit3 %>% continue()

# This model will stop because 12 have been seen.
fit4 <- model3 %>% fit('1NNN 2NNN 2NNN 3NNN')
fit4 %>% recommended_dose()
fit4 %>% continue()

# With enough observations though, both models will advise stopping because
# both conditions have been met:
fit5 <- model2 %>% fit('1NNN 2NNN 2NNN 5NNN 5NNN 5NNN')
fit5 %>% recommended_dose()
fit5 %>% continue()

fit6 <- model3 %>% fit('1NNN 2NNN 2NNN 5NNN 5NNN 5NNN')
fit6 %>% recommended_dose()
fit6 %>% continue()
```
stop_when_n_at_dose  

Stop when there are n patients at a 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**

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

**Arguments**

- `parent_selector_factory`: Object of type `selector_factory`.
- `n`: Stop when there are n at a dose.
- `dose`: 'any' to stop when there are n at any dose; 'recommended' to stop when there are n at the recommended dose; or an integer to stop when 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 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()
```
stop_when_too_toxic

# But this model fit will stop:
fit3 <- model2 %>% fit('1NNN 2NTN 2TNN 2NNN 2NNT')
fit3 %>% recommended_dose()
fit3 %>% continue()

stop_when_too_toxic  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

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 3NNNT 3NNNN 3TNT 2NNN'
fit1 <- model1 %>% fit(outcomes)
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

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

Arguments

parent_selector_factory
  Object of type selector_factory.
stop_when_tox_ci_covered

Dose

'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

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()
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(selector, ...)

Arguments

selector 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 <- '1NNN 2NTN'
fit <- get_dfcrm(skeleton = skeleton, target = target) %>% fit(outcomes)
fit %>% supports_sampling()
three_plus_three

**Description**

Fit the 3+3 model to some outcomes.

**Usage**

```r
two_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**

Lists containing recommended_dose and a logical value continue saying whether the trial should continue.

**References**


**Examples**

```r
two_plus_three(’2NNN 3NNT’, num_doses = ?)
```
tox

**Description**

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

**Usage**

```r
tox(selector, ...)
```

**Arguments**

- `selector`: 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()
```

---

tox_at_dose

**Description**

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

**Usage**

```r
tox_at_dose(selector, ...)
```

**Arguments**

- `selector`: 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 %>% tox_at_dose()

```

Description

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

Usage

tox_target(selector, ...)

Arguments

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

Value

numeric

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 %>% tox_target()
```
Index

cohort, 3, 25  
continue, 3, 26  
crm, 12, 25, 33, 35

demand_n_at_dose, 4  
dont_skip_doses, 5  
dose_indices, 7, 26  
doses_given, 7, 25

demand_n_at_dose, 4  
fit, 9, 25, 28  
follow_path, 10, 12

get_boin, 11, 25, 28  
get_dfcrm, 12, 25, 28  
get_three_plus_three, 13, 25, 28

mean_prob_tox, 14, 25, 26  
median_prob_tox, 15, 25, 26  
model_frame, 16, 25

n_at_dose, 18, 26  
um_doses, 17, 25  
um_patients, 17, 25  
num_tox, 18, 25

parse_phase1_outcomes, 9, 10, 19, 37  
phase1_outcomes_to_cohorts, 20  
prob_administer, 21, 26  
prob_tox_exceeds, 22, 25, 26  
prob_tox_quantile, 23, 26  
prob_tox_samples, 23

recommenced_dose, 24, 26

select.mtd, 11  
select_dose_by_cibp, 29  
selector, 3, 7, 8, 10, 15–19, 21–24, 25, 28, 36, 38, 39

selector_factory, 4, 6, 10–12, 14, 25, 26, 27, 28–35  
stop_at_n, 3, 30  
stop_when_n_at_dose, 25, 28, 32  
stop_when_too_toxic, 3, 25, 28, 33  
stop_when_tox_ci_covered, 34  
supports_sampling, 23, 36

three_plus_three, 37  
tibble, 16  
tox, 25, 38  
tox_at_dose, 26, 38  
tox_target, 25, 39