Package 'precautionary'

September 21, 2020

Type Package

Title Safety Diagnostics for Dose-Escalation Trial Designs

Version 0.1-4

Date 2020-09-20

Maintainer David C. Norris <david@precisionmethods.guru>

Depends magrittr, escalation, data.table

Imports methods, dplyr, rlang, stringr, knitr, kableExtra

Suggests rmarkdown, bookdown, tufte, testthat

Description Enhances various R packages that support the design and simulation of phase 1 dose-escalation trials, adding diagnostics to examine the safety characteristics of these designs in light of expected inter-individual variation in pharmacokinetics and pharmacodynamics. See Norris (2020), ``Retrospective analysis of a fatal dose-finding trial'' <arXiv:2004.12755>.

URL https://precisionmethods.guru/

License MIT + file LICENSE

VignetteBuilder knitr, rmarkdown

Encoding UTF-8

NeedsCompilation no

RoxygenNote 7.1.1

Collate 'enhance.R' 'precautionary-package.R' 'toxicity_generators.R' 'simulate_trials.R'

Author David C. Norris [aut, cre, cph] (<https://orcid.org/0000-0001-9593-6343>)

Repository CRAN

Date/Publication 2020-09-21 07:00:02 UTC

R topics documented:

precautionary-package . . . 2
as.data.table.precautionary . . . 3
cohorts_of_n . . . 4
extend . . . 4
format.safetytab . . . 5
hyper_mtdi_lognormal-class . . . 6
mtdi_lognormal-class . . . 7
phase1_sim . . . 7
plot,mtdi_distribution-method . . . 8
plot,mtdi_generator-method . . . 9
print.hyper . . . 9
print.precautionary . . . 10
safety_kable . . . 10
simulate_trials . . . 11
simulation_function.u_i . . . 13
summary.precautionary . . . 13
test_draw_samples . . . 14

Index **15**

`precautionary-package` *Safety Diagnostics for Dose-Escalation Trial Designs*

Description

Enhances various R packages that support the design and simulation of phase 1 dose-escalation trials, adding diagnostics to examine the safety characteristics of these designs in light of expected inter-individual variation in pharmacokinetics and pharmacodynamics.

Author(s)

David C. Norris (https://orcid.org/0000-0001-9593-6343)

References

1. Norris DC. Dose Titration Algorithm Tuning (DTAT) should supersede 'the' Maximum Tolerated Dose (MTD) in oncology dose-finding trials. *F1000Research*. 2017;6:112. doi: 10.12688/f1000research.10624.3. https://f1000research.com/articles/6-112/v3
2. Norris DC. Costing 'the' MTD. *bioRxiv*. August 2017:150821. doi: 10.1101/150821. https://www.biorxiv.org/content/10.1101/150821v3
3. Norris DC. Precautionary Coherence Unravels Dose Escalation Designs. *bioRxiv*. December 2017:240846. doi: 10.1101/240846. https://www.biorxiv.org/content/10.1101/240846v1
4. Norris DC. One-size-fits-all dosing in oncology wastes money, innovation and lives. *Drug Discov Today*. 2018;23(1):4-6. doi: 10.1016/j.drudis.2017.11.008. https://www.sciencedirect.com/science/article/pii/S1359644617303586

5. Norris DC. Costing 'the' MTD ... in 2-D. *bioRxiv*. July 2018:370817. doi: 10.1101/370817. https://www.biorxiv.org/content/10.1101/370817v1

6. Norris DC. Ethical Review and Methodologic Innovation in Phase 1 Cancer Trials. *JAMA Pediatrics*. 2019;173(6):609 doi: 10.1001/jamapediatrics.2019.0811.

7. Norris DC. Comment on Wages et al., Coherence principles in interval-based dose finding. Pharmaceutical Statistics 2019, DOI: 10.1002/pst.1974. *Pharmaceutical Statistics*. March 2020. doi: 10.1002/pst.2016.

8. Norris DC. Retrospective analysis of a fatal dose-finding trial. arXiv:2004.12755 [stat.ME]. April 2020. https://arxiv.org/abs/2004.12755

`as.data.table.precautionary` *Convert an object of class c('precautionary','simulations') to a data.table*

Description

Convert an object of class c('precautionary','simulations') to a data.table

Usage

```
## S3 method for class 'precautionary'
as.data.table(
  x,
  keep.rownames = FALSE,
  ordinalizer = getOption("ordinalizer"),
  ...
)
```

Arguments

`x`	An object of class c('precautionary','simulations')
`keep.rownames`	Unused; retained for S3 generic/method consistency
`ordinalizer`	If not NULL, this is a function mapping the threshold dose ('MTDi') at which an individual experiences a binary toxicity (as recognized by the dose-escalation design) to a named vector giving dose thresholds for multiple grades of toxicity. The names of this vector will be taken as designations of the toxicity grades.
`...`	Additional parameters passed to the `ordinalizer`

cohorts_of_n *Override cohorts_of_n to include latent toxicity tolerances*

Description

The original function in package **escalation** recognizes that individual trial participants arrive at distinct times. Building upon this acknowledgment of individuality, this override adds an extra line of code to draw as well a latent toxicity tolerance `u_i` for each individual participant.

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 columns `u_i` and `time_delta` containing respectively the uniformly-distributed latent toxicity tolerance and arrival-time increment for each trial participant.

See Also

`phase1_sim()`, which this package also overrides with similarly minute changes in order to incorporate `u_i`.

Examples

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

extend *Extend an existing simulation, using one of several stopping criteria*

Description

A trial simulation carried through a predetermined number of replications may not achieve desired precision as judged by Monte Carlo standard errors (MCSE) of estimated toxicity counts. This method enables simulations of class `c('precautionary','simulations')` to be extended until a given level of precision has been achieved on expected counts of enrollment and DLTs, or (optionally) for a fixed additional number of simulations.

Usage

```
extend(sims, num_sims = NULL, target_mcse = 0.05)
```

Arguments

sims	An existing object of class c('precautionary','simulations')
num_sims	Optionally, a fixed number of additional replications to accumulate
target_mcse	Optionally, an MCSE constraint to be imposed on expected counts of DLTs, non-DLTs, and Total enrollment.

Value

An extended simulation of same class as sims.

Note

The MCSE constraint is imposed during trial *simulation*, at which point only *binary* toxicities are available. Thus, as a practical matter, extend can target MCSEs only for DLTs, non-DLTs and Total enrollment. The subsequent subdivision of these categories during trial *summary* (at which point the ordinalizer comes into play along with its parameters) thus may generate expected counts with MCSEs exceeding target_mcse. In practice, however, this tends to affect the estimated counts only for the *lowest* toxicity grades—those of least concern from a trial-safety perspective.

See Also

See examples under simulate_trials and format.safetytab.

---

format.safetytab *Format a phase 1 trial safety tabulation to show significant digits only*

Description

The essential insight of package precautionary is distilled into the *safety tabulation* which it generates from trial simulations, reporting the expected number of patients who will experience each grade of toxicity. To render this table for easy interpretation, these expectations are simply displayed with a number of significant digits appropriate to their Monte Carlo standard errors (MCSEs).

Usage

```
## S3 method for class 'safetytab'
format(x, ...)
```

Arguments

x	A safety tabulation as found in the safety component of the list returned by summary.precautionary.
...	Unused; included for compatibility with generic signature

Note

The MCSEs of safety tabulations remain available for inspection (see example), but are omitted from standard displays because they may lend themselves to misinterpretation as *confidence bounds* on the number of patients who will experience each toxicity grade *in any given trial*.

Examples

```
mtdi_gen <- hyper_mtdi_lognormal(CV = 1
                                 ,median_mtd = 5
                                 ,median_sdlog = 0.5
                                 ,units="mg/kg")
ordinalizer <- function(MTDi, r0 = 1.5)
  MTDi * r0 ^ c(Gr1=-2, Gr2=-1, Gr3=0, Gr4=1, Gr5=2)
old <- options(dose_levels = c(0.5, 1, 2, 4, 6)
               ,ordinalizer = ordinalizer)
get_boin(num_doses = 5, target = 0.25) %>%
  stop_at_n(n = 24) %>%
  simulate_trials(
    num_sims = 60
  , true_prob_tox = mtdi_gen) -> boin_hsims
safety <- summary(boin_hsims)$safety
safety # The print method invokes 'format.safetytab' ..
# .. but we can also inspect the underlying matrix by indexing:
safety[,]  # indexing strips 'safetytab' class, returning plain matrix
# Note that, by extend()ing the simulation we can increase precision:
if (interactive()) { # may run a bit too long for CRAN servers' taste
  boin_hsims %>% extend(target_mcse = 0.1) -> boin_hsimsX
  summary(boin_hsimsX)$safety
}
options(old)
```

---

hyper_mtdi_lognormal-class

Hyperprior for lognormal MTDi distributions

Description

This hyperprior generates lognormal MTDi distributions with their coefficients of variation being drawn from a Rayleigh distribution with mode parameter set to

$$\sigma := CV.$$

Because the standard deviation of this distribution is

$$sd = \sigma\sqrt{(2 - \pi/2)} \approx 0.655\sigma,$$

this conveniently links our uncertainty about `CV` to its *mode*, and indeed to other measures of centrality that are proportional to this:

$$mean = \sigma\sqrt{\pi/2} \approx 1.25\sigma$$

$$median = \sigma\sqrt{2log(2)} \approx 1.18\sigma.$$

The *medians* of the lognormal distributions generated are themselves drawn from a lognormal distribution with `meanlog = log(median_mtd)` and `sdlog = median_sdlog`. Thus, parameter `median_sdlog` represents a proportional uncertainty in `median_mtd`.

Slots

`CV` Coefficient of variation

`median_mtd` Median MTDi

`median_sdlog` Proportional uncertainty in median MTDi

---

mtdi_lognormal-class *A lognormal MTDi distribution*

Description

A lognormal MTDi distribution

Slots

`dist` A list with `cdf`, `quantile` and `name` components intended to provide that portion of the interface of `distr6::Lognormal`.

---

phase1_sim *Override* `escalation:::phase1_sim` *to incorporate latent toxicity tolerances*

Description

Override `escalation:::phase1_sim` to incorporate latent toxicity tolerances

Usage

```
phase1_sim(
  selector_factory,
  true_prob_tox,
  sample_patient_arrivals = function(df) cohorts_of_n(n = 3, mean_time_delta = 1),
  previous_outcomes = "",
  next_dose = NULL,
  i_like_big_trials = FALSE,
  return_all_fits = FALSE
)
```

Arguments

selector_factory	An selector_factory object
true_prob_tox	A vector of toxicity probabilities for the doses defined in selector_factory
sample_patient_arrivals	A function implementing an arrivals process for trial enrollment
previous_outcomes	This may or may not apply in applications of package precautionary
next_dose	Undocumented
i_like_big_trials	I didn't choose this parameter name
return_all_fits	Don't do this

---

plot,mtdi_distribution-method *Visualize an* mtdi_distribution *object*

Description

Visualize an mtdi_distribution object

Usage

```
## S4 method for signature 'mtdi_distribution'
plot(x, y = NULL, ...)
```

Arguments

x	An mtdi_distribution object
y	Included for compatibility with generic signature
...	Additional arguments passed onward to plot

Examples

```
if (interactive()) {
mtdi_dist <- mtdi_lognormal(CV = 2
                            ,median = 5
                            ,units = "mg/kg")
# Setting pre-specified dose levels via options() causes
# toxicity probabilities to be annotated on the plot.
old <- options(dose_levels = c(0.5, 1, 2, 4, 6))
plot(mtdi_dist, col = "red")
options(old)
}
```

plot,mtdi_generator-method *Visualize n samples from an* mtdi_generator *object*

Description

Visualize n samples from an mtdi_generator object

Usage

```
## S4 method for signature 'mtdi_generator'
plot(x, y = NULL, n = 20, col = "gray", ...)
```

Arguments

x	An mtdi_generator object
y	Included for compatibility with generic signature
n	Number of samples to draw from hyperprior for visualization
col	Color of lines used to depict samples
...	Additional arguments passed onward to plot

Examples

```
if (interactive()) {
mtdi_gen <- hyper_mtdi_lognormal(CV = 1
                                 ,median_mtd = 5
                                 ,median_sdlog = 0.5
                                 ,units="mg/kg")
plot(mtdi_gen, n=100, col=adjustcolor("red", alpha=0.5))
}
```

print.hyper *Specialize print method defined for class* simulations

Description

Specialize print method defined for class simulations

Usage

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

Arguments

x	An object of class c("hyper","precautionary","simulations")
...	Additional arguments; ignored

print.precautionary *Specialize print method for objects of class* simulations

Description

Specialize print method for objects of class simulations

Usage

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

Arguments

x	An object of class c("precautionary","simulations")
...	Additional arguments; ignored

safety_kable *Output a* kable *for a simulation summary of class* safetytab

Description

Output a kable for a simulation summary of class safetytab

Usage

```
safety_kable(safetytab, ...)
```

Arguments

safetytab	An object of S3 class safetytab
...	Additional parameters passed to knitr::kable

simulate_trials *Simulate trials defined via package* R*hrefhttps://CRAN.R-project.org/package=escalation***escalation**

Description

An S4 generic method providing a more abstract interface to trial simulation than the simulate_trials function of package escalation. This abstraction is needed to support simulations in which escalation's simple vectors of 'true toxicity probabilities' are replaced by precautionary's more realistic toxicity-distribution generators.

Usage

```
simulate_trials(selector_factory, num_sims, true_prob_tox, ...)

## S4 method for signature 'selector_factory,numeric,hyper_mtdi_distribution'
simulate_trials(selector_factory, num_sims, true_prob_tox, ...)

## S4 method for signature 'selector_factory,numeric,mtdi_distribution'
simulate_trials(selector_factory, num_sims, true_prob_tox, ...)
```

Arguments

selector_factory	An object of S3 class selector_factory
num_sims	Number of simulations to run
true_prob_tox	A generator of toxicity distributions
...	Passed to subroutines

Details

If invoked interactively with num_sims > 10, then a txtProgressBar is displayed in the console. The condition on num_sims has the useful side effect of allowing this function to be invoked iteratively by extend (with num_sims = 10) without the nuisance of nested progress bars.

Examples

```
old <- options(dose_levels = c(0.5, 1, 2, 4, 6, 8))
mtdi_gen <- hyper_mtdi_lognormal(CV = 1
                                 , median_mtd = 6, median_sdlog = 0.5
                                 , units="mg/kg")
num_sims <- ifelse(interactive()
, 300
, 15 # avoid taxing CRAN servers
)
hsims <- get_three_plus_three(num_doses = 6) %>%
  simulate_trials(
```

```
    num_sims = num_sims
  , true_prob_tox = mtdi_gen)
summary(hsims, ordinalizer=NULL) # vanilla summary with binary toxicity
summary(hsims, ordinalizer = function(dose, r0 = sqrt(2))
  c(Gr1=dose/r0^2, Gr2=dose/r0, Gr3=dose, Gr4=dose*r0, Gr5=dose*r0^2)
)
hsims <- hsims %>% extend(num_sims = num_sims)
summary(hsims, ordinalizer = function(dose, r0 = sqrt(2))
  c(Gr1=dose/r0^2, Gr2=dose/r0, Gr3=dose, Gr4=dose*r0, Gr5=dose*r0^2)
)$safety
# Set a CRM skeleton from the average probs in above simulation
num_sims <- ifelse(interactive()
, 16
,  4  # avoid taxing CRAN servers
)
get_dfcrm(skeleton = hsims$avg_prob_tox
         ,target = 0.25
         ) %>% stop_at_n(n = 24) %>%
  simulate_trials(
    num_sims = num_sims
  , true_prob_tox = mtdi_gen
  ) -> crm_hsims
summary(crm_hsims
, ordinalizer = function(MTDi, r0 = sqrt(2))
    MTDi * r0^c(Gr1=-2, Gr2=-1, Gr3=0, Gr4=1, Gr5=2)
)
options(old)
old <- options(dose_levels = c(2, 6, 20, 60, 180, 400))
mtdi_dist <- mtdi_lognormal(CV = 0.5
                           ,median = 140
                           ,units = "ng/kg/week")
num_sims <- ifelse(interactive()
, 100
,  10  # avoid taxing CRAN servers
)
sims <- get_three_plus_three(num_doses = 6) %>%
  simulate_trials(
    num_sims = num_sims
  , true_prob_tox = mtdi_dist)
# Now set a proper ordinalizer via options():
options(ordinalizer = function(dose, r0) {
  c(Gr1=dose/r0^2, Gr2=dose/r0, Gr3=dose, Gr4=dose*r0, Gr5=dose*r0^2)
})
summary(sims, r0=2)
# Set a CRM skeleton from the average probs in above simulation
get_dfcrm(skeleton = sims$true_prob_tox
         ,target = 0.25
         ) %>% stop_at_n(n = 24) %>%
  simulate_trials(
    num_sims = 20
  , true_prob_tox = mtdi_dist
  ) -> crm_sims
summary(crm_sims
```

```
, ordinalizer = function(MTDi, r0 = sqrt(2))
    MTDi * r0^c(Gr1=-2, Gr2=-1, Gr3=0, Gr4=1, Gr5=2)
)
if (interactive()) { # don't overtax CRAN servers
crm_sims <- crm_sims %>% extend(target_mcse = 0.1)
summary(crm_sims
, ordinalizer = function(MTDi, r0 = sqrt(2))
    MTDi * r0^c(Gr1=-2, Gr2=-1, Gr3=0, Gr4=1, Gr5=2)
)$safety
}
options(old)
```

---

simulation_function.u_i *Get a function that simulates dose-escalation trials using latent* u_i

---

Description

Overrides simulation_function.tox_selector_factory to return a phase1_sim() that employs latent u_i in place of the version native to package **escalation**, which merely invokes rbinom.

Usage

```
## S3 method for class 'u_i'
simulation_function(selector_factory)
```

Arguments

selector_factory	Presently, this must be a tox_selector_factory; no equivalent for simulation_function.derived_do is yet implemented.

Details

This function is exported for the purpose of effecting this override, and is not meant to be invoked directly by the user.

---

summary.precautionary *Specialize a method defined in package 'escalation' for class 'simulations'*

Description

Simulations produced by package precautionary incorporate a 'u_i' latent toxicity tolerance that characterizes the toxic dose-response of each simulated individual trial participant. In conjunction with an 'ordinalizer' function, these extra data enable questions to be asked about trial safety, in terms of the probabilities of high-grade toxicities. This function specializes the escalation:::summary.simulations method accordingly.

Usage

```
## S3 method for class 'precautionary'
summary(object, ordinalizer = getOption("ordinalizer"), ...)
```

Arguments

object	An object of class c('precautionary','simulations')
ordinalizer	An ordinalizer function
...	Additional parameters passed to the ordinalizer

`test_draw_samples` *Test performance of* `draw_samples` *function*

Description

Test performance of `draw_samples` function

Usage

```
test_draw_samples(n = 500)
```

Arguments

n	Number of samples to draw

Index

as.data.table.precautionary, 3

cohorts_of_n, *4*, 4

extend, 4, *11*

format.safetytab, *5*, 5

hyper_mtdi_lognormal
    *(*hyper_mtdi_lognormal-class*)*, 6
hyper_mtdi_lognormal-class, 6

knitr::kable, *10*

mtdi_lognormal *(*mtdi_lognormal-class*)*, 7
mtdi_lognormal-class, 7

phase1_sim, 7
phase1_sim(), *4*, *13*
plot,mtdi_distribution-method, 8
plot,mtdi_generator-method, 9
precautionary, *5*
precautionary *(*precautionary-package*)*, 2
precautionary-package, 2
print.hyper, 9
print.precautionary, 10

safety_kable, 10
selector_factory, *8*, *11*
simulate_trials, *5*, *11*, 11
simulate_trials,selector_factory,numeric,hyper_mtdi_distribution-method
    *(*simulate_trials*)*, 11
simulate_trials,selector_factory,numeric,mtdi_distribution-method
    *(*simulate_trials*)*, 11
simulation_function.tox_selector_factory,
    13
simulation_function.u_i, 13
simulations, *9, 10*
summary.precautionary, *5*, 13

test_draw_samples, 14