Package 'rcens'

July 23, 2026

Type Package

Title Generate Sample Censoring

Version 0.2.2

Author Daniel Saavedra [aut, cre, cph] (ORCID:
<https://orcid.org/0000-0002-5084-6843>),
Pedro L Ramos [aut]

Maintainer Daniel Saavedra <dlsaavedra@uc.cl>

Description
Provides functions to generate censored samples of type I, II and III, from any random sample generator. It also supplies the option to create left and right censorship. Along with this, the generation of samples with interval censoring is in the testing phase, with two options of fixed length intervals and random lengths. Additional functions generate complex inspection, delayed-entry, hybrid, progressive, covariate-dependent, frailty-dependent, and competing-risk observation structures while retaining a consistent and verifiable data object.

License MIT + file LICENSE

Encoding UTF-8

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

Config/testthat/edition 3

VignetteBuilder knitr

URL https://github.com/dlsaavedra/rcens

BugReports https://github.com/dlsaavedra/rcens/issues

Config/roxygen2/version 8.0.0

NeedsCompilation no

Repository CRAN

Date/Publication 2026-07-23 05:50:02 UTC

Contents

`rcenscomp` *Generate data under a complex censoring or observation scheme*

Description

'rcenscomp()' is the unified entry point for the complex-censoring extension. It separates latent event generation from the observation operator and dispatches to a specialized 'rcenscomp_*()' function.

Usage

```
rcenscomp(
  scheme,
  x = NULL,
  ...,
  rdistrX = NULL,
  param_X = list(),
  n = NULL,
  keep_latent = TRUE
)
```

Arguments

scheme	Required scheme name. See Details for canonical values.
x	Optional numeric vector of complete latent event times.
...	Named arguments for the specialized scheme function. Unknown arguments are rejected.
rdistrX	Optional random generator whose first argument is the sample size.
param_X	Named or unnamed list of arguments passed to 'rdistrX'.
n	Sample size when 'rdistrX' is used or when the selected scheme generates its processes internally.
keep_latent	If 'TRUE', retain latent quantities needed to audit the observation mechanism.

Details

The supported taxonomy comprises inspection schemes ('interval', 'current_status'), delayed entry, hybrid schemes, progressive schemes, conditionally independent covariate-dependent censoring, informative censoring induced by shared process frailty, and competing risks.

For schemes based on one latent sample, provide exactly one of 'x' and 'rdistrX'. With 'rdistrX', the latent sample is generated by 'do.call(rdistrX, c(list(n), param_X))'. Schemes that generate multiple processes internally ('covariate_censoring', 'informative_frailty', and 'competing_risks') require 'n' and reject 'x' and 'rdistrX'.

Each result records the canonical scheme name and the observed data object. The likelihood appropriate to that observed object is not fitted by this package extension.

Value

An object of class 'rcenscomp' with components 'data', 'latent', 'scheme', 'design', 'diagnostics', and 'call'.

References

Ramos, P. L., Kumar, I., Dutta, S., and Kundu, D. *Sampling with Complex Censored Data: A Practical Guide* (unpublished manuscript).

See Also

- rcenscomp_interval
- rcenscomp_hybrid_type1
- rcenscomp_progressive_type2
- rcenscomp_informative_frailty
- rcenscomp_competing_risks

Examples

```
set.seed(2026)
dat <- rcenscomp(
  scheme = "hybrid_type1",
  rdistrX = rexp,
  param_X = list(rate = 2),
  n = 100,
  T = 1,
  r = 60
)
dat

x <- rexp(20, rate = 2)
rcenscomp("interval", x = x, visits = c(0.2, 0.5, 1))
```

`rcenscomp-methods` *Methods for complex censored samples*

Description

'print()' gives a compact design summary, 'summary()' returns the complete design diagnostics, and 'as.data.frame()' extracts the observed data object.

Usage

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

## S3 method for class 'rcenscomp'
summary(object, ...)

## S3 method for class 'rcenscomp'
as.data.frame(x, row.names = NULL, optional = FALSE, ...)
```

Arguments

`x`	An object of class 'rcenscomp'.
`...`	Additional arguments passed by the corresponding generic.
`object`	An object of class 'rcenscomp'.
`row.names`	Optional row names for the extracted data frame.
`optional`	Logical argument required by [as.data.frame()].

Value

'print()' returns 'x' invisibly. 'summary()' returns an object of class 'summary.rcenscomp'. 'as.data.frame()' returns 'x$data'.

References

Ramos, P. L., Kumar, I., Dutta, S., and Kundu, D. *Sampling with Complex Censored Data: A Practical Guide* (unpublished manuscript).

Examples

```
x <- c(0.2, 0.5, 0.9, 1.3)
z <- rcenscomp_hybrid_type1(x, T = 1, r = 3)
print(z)
summary(z)
as.data.frame(z)
```

`rcenscomp_adaptive_progressive_type2`
Apply adaptive progressive Type-II hybrid censoring

Description

Observes exactly ‘length(R)‘ failures, follows planned removals through failures at or before ‘T‘, and suppresses future intermediate removals after the ideal time is crossed.

Usage

```
rcenscomp_adaptive_progressive_type2(
  x,
  R,
  T,
  ties = "error",
  keep_latent = TRUE
)
```

Arguments

`x`	Positive finite latent event times.
`R`	Non-negative integer planned withdrawal vector satisfying the progressive Type-II size constraint.
`T`	Positive ideal time.
`ties`	Either ‘"error"‘ or ‘"first"‘.
`keep_latent`	If ‘TRUE‘, retain exact latent event times.

Details

Failures at times ‘<= T‘ still use the planned withdrawal. At the first pre-final failure after ‘T‘, adaptation begins and later intermediate withdrawals are zero. All remaining units are withdrawn at the ‘m‘th failure. If the first crossing occurs only at the final failure, there is no future intermediate withdrawal to adapt and the ‘adaptive‘ indicator remains false.

Value

An ‘rcenscomp‘ object with exactly ‘m‘ events, realized ‘R_used‘, an adaptation indicator and, when applicable, its first index and time.

References

Ng, H. K. T., Kundu, D., and Chan, P. S. (2009). Statistical analysis under an adaptive Type-II progressive censoring scheme. *Naval Research Logistics*, 56, 687-698.

See Also

- rcenscomp_progressive_hybrid

Examples

```
set.seed(4)
rcenscomp_adaptive_progressive_type2(
  x = c(0.2, 0.6, 1.1, 1.5, 2),
  R = c(1, 0, 1), T = 0.5
)
```

---

rcenscomp_calibrate_progressive_hybrid_time

Calibrate the time cap of a progressive hybrid design

Description

Chooses a time cap ‘T‘ so the expected number of failures in a time-capped progressive hybrid design is close to ‘target_D‘.

Usage

```
rcenscomp_calibrate_progressive_hybrid_time(
  n,
  beta,
  alpha,
  R,
  target_D,
  lower,
  upper,
  B = 1000,
  grid_size = 201,
  seed = 2026
)
```

Arguments

n	Positive integer initial sample size.
beta	Positive Weibull rate-like parameter.
alpha	Positive Weibull shape parameter.
R	Valid progressive withdrawal plan of length ‘m‘ with ‘sum(R) = n - m‘.
target_D	Finite expected failure-count target in ‘[0, m]‘.
lower, upper	Positive finite candidate limits.
B	Positive integer Monte Carlo replication count.
grid_size	Integer candidate-grid size, at least two.
seed	Non-negative integer local seed.

Details

Latent Weibull samples and withdrawal-randomization streams are held common across every candidate ‘T‘. The Weibull parameterization uses shape ‘alpha‘ and rate-like ‘beta‘. This is a design calibration only and does not fit a censored-data model.

Value

A list with selected ‘T‘, achieved expected failures ‘achieved_D‘, complete grid, seed, replication count, and requested target.

See Also

- rcenscomp_calibrate_time
- rcenscomp_progressive_hybrid

Examples

```
rcenscomp_calibrate_progressive_hybrid_time(
  n = 10, beta = 2.5, alpha = 1.5,
  R = c(1, 1, 0, 0, 0, 0, 0, 0),
  target_D = 5, lower = 0.1, upper = 1,
  B = 20, grid_size = 7, seed = 2026
)
```

`rcenscomp_calibrate_time` *Calibrate a time parameter by Monte Carlo with common random numbers*

Description

Selects the grid value whose Monte Carlo design summary is closest to a requested target while reusing latent samples and randomization streams at every grid point.

Usage

```
rcenscomp_calibrate_time(
  generator,
  censor_fun,
  target,
  lower,
  upper,
  B = 1000,
  grid_size = 201,
  seed = 2026,
 metric = c("censoring_proportion", "expected_failures", "probability_few_failures",
    "expected_stopping_time"),
  d_min = NULL
)
```

Arguments

`generator`	Function with no arguments that returns one latent sample.
`censor_fun`	Function of ‘(latent, value)‘ applying the observation mechanism at a candidate time value.
`target`	Finite requested target.
`lower, upper`	Positive finite grid limits with ‘lower < upper‘.
`B`	Positive integer Monte Carlo replication count.
`grid_size`	Integer number of candidate values, at least two.
`seed`	Non-negative integer local calibration seed.
`metric`	One of ‘"censoring_proportion"‘, ‘"expected_failures"‘, ‘"probability_few_failures"‘, or ‘"expected_stopping_time"‘.
`d_min`	Non-negative integer threshold required for ‘"probability_few_failures"‘.

Details

‘generator()‘ must return one latent sample. ‘censor_fun(latent, value)‘ must return either an ‘rcenscomp‘ object or a data frame. Supported metrics are expected censoring proportion, expected failure count, probability of fewer than ‘d_min‘ failures, and expected stopping time. Common random numbers

reduce Monte Carlo noise across grid values. The supplied ‘seed‘ is local: the caller’s ‘.Random.seed‘ is restored exactly, or removed again if it did not previously exist.

The procedure is a finite-grid calibration, not a proof that the target is attainable or that the response is monotone. A warning is emitted when the target lies outside the achieved grid range. Every replicate must produce a finite scalar summary; invalid or missing summaries are rejected explicitly.

Value

A list with selected ‘value‘, ‘achieved‘, complete ‘grid‘, ‘seed‘, ‘B‘, ‘target‘, ‘metric‘, and ‘d_min‘.

References

Ramos, P. L., Kumar, I., Dutta, S., and Kundu, D. *Sampling with Complex Censored Data: A Practical Guide* (unpublished manuscript).

See Also

- rcenscomp_calibrate_progressive_hybrid_time

Examples

```
generator <- function() rexp(20, rate = 2)
censor_fun <- function(x, T) rcenscomp_hybrid_type1(x, T = T, r = 10)
rcenscomp_calibrate_time(
  generator, censor_fun, target = 0.5,
  lower = 0.1, upper = 1.5, B = 20, grid_size = 9,
  seed = 2026, metric = "censoring_proportion"
)
```

---

rcenscomp_competing_risks

Generate independent-latent-time competing risks

Description

Generates one Weibull latent time for each cause, records the earliest cause, and applies optional administrative censoring.

Usage

```
rcenscomp_competing_risks(
  n,
  beta,
  alpha,
  censor_time = Inf,
  keep_latent = TRUE,
  ties = "error"
)
```

Arguments

n	Positive integer sample size.
beta	Positive rate-like parameter vector, one value per cause.
alpha	Positive shape parameter vector of the same length as ‘beta‘.
censor_time	Positive administrative censoring time, either scalar or length ‘n‘; positive infinity is allowed.
keep_latent	If ‘TRUE‘, retain the complete matrix of cause-specific latent times.
ties	Either ‘"error"‘ or ‘"first"‘ for exact cause-time ties.

Details

‘status = 0‘ denotes censoring and ‘status = 1, ..., K‘ denotes the cause of the earliest event. Independent latent cause times are a simulation assumption and are not identifiable from ordinary competing-risk data. Competing events are not ordinary independent censoring for a cumulative incidence estimade. Cause ties are rejected by default; ‘ties = "first"‘ uses the smallest cause index as an explicit computational convention.

Value

An ‘rcenscomp‘ object whose observed data has ‘id‘, ‘time‘, and integer ‘status‘. Diagnostics include counts for censoring and each cause.

References

Prentice, R. L. et al. (1978). The analysis of failure times in the presence of competing risks. *Biometrics*, 34, 541-554.

See Also

- rcenscomp_rweibull_rate

Examples

```
set.seed(22)
rcenscomp_competing_risks(
  n = 10,
  beta = c(1.2, 0.8),
  alpha = c(1.4, 1.1),
  censor_time = 1
)
```

`rcenscomp_covariate_censoring` *Generate conditionally independent covariate-dependent censoring*

Description

Generates Weibull PH event and censoring times that share recorded covariates but use independent uniforms.

Usage

```
rcenscomp_covariate_censoring(
  n,
  beta_x,
  alpha_x,
  gamma_x,
  beta_c,
  alpha_c,
  gamma_c,
  z = NULL,
  keep_latent = TRUE
)
```

Arguments

`n`	Positive integer sample size.
`beta_x, alpha_x`	Positive Weibull event rate-like and shape parameters.
`gamma_x`	Finite event PH coefficient vector.
`beta_c, alpha_c`	Positive Weibull censoring rate-like and shape parameters.
`gamma_c`	Finite censoring PH coefficient vector with the same length as ‘gamma_x‘.
`z`	Optional finite numeric covariate matrix with ‘n‘ rows.
`keep_latent`	If ‘TRUE‘, retain ‘x_true‘, ‘c_true‘, and covariates.

Details

Conditional on ‘z‘, the generated event and censoring times are independent. Consequently, covariate-dependent censoring is not automatically informative when the analysis conditions on the relevant covariates. If ‘z = NULL‘, one Bernoulli(0.5) and one standard normal covariate are generated, requiring both coefficient vectors to have length two.

Value

An ‘rcenscomp‘ object whose observed data has ‘id‘, ‘time‘, ‘delta‘, and the covariate columns.

References

Ramos, P. L., Kumar, I., Dutta, S., and Kundu, D. *Sampling with Complex Censored Data: A Practical Guide* (unpublished manuscript).

See Also

- rcenscomp_rweibull_ph
- rcenscomp_informative_frailty

Examples

```
set.seed(20)
rcenscomp_covariate_censoring(
  n = 10,
  beta_x = 2.5, alpha_x = 1.5, gamma_x = c(0.5, -0.3),
  beta_c = 1.2, alpha_c = 1.1, gamma_c = c(-0.2, 0.4)
)
```

---

rcenscomp_current_status

Generate current-status observations

Description

Records only whether each latent event occurred by its single inspection time.

Usage

```
rcenscomp_current_status(x, inspection, keep_latent = TRUE)
```

Arguments

x	Positive finite latent event times.
inspection	Positive finite inspection times with the same length as ‘x‘.
keep_latent	If ‘TRUE‘, retain the exact latent event times.

Details

The observed status is ‘as.integer(x <= inspection)‘. When it equals one, ‘inspection‘ is not the exact failure time; the event is known only to have occurred in ‘(0, inspection]‘. The function assumes the inspection process is supplied externally and does not assert ignorability on its own.

Value

An ‘rcenscomp‘ object whose observed ‘data‘ has ‘id‘, ‘inspection‘, and binary ‘status‘ columns. Diagnostics give the number and proportion of positive current-status records.

References

Groeneboom, P. and Wellner, J. A. (1992). *Information Bounds and Nonparametric Maximum Likelihood Estimation*. Birkhauser.

See Also

- rcenscomp_interval

Examples

```
rcenscomp_current_status(
  x = c(0.2, 0.8, 1.5),
  inspection = c(0.3, 0.5, 2)
)
```

---

rcenscomp_delayed_entry

Apply delayed entry and post-entry right censoring

Description

Excludes units with ‘x <= entry‘, then records right-censored follow-up for the units that survived long enough to enter observation.

Usage

```
rcenscomp_delayed_entry(x, entry, censor, keep_latent = TRUE)
```

Arguments

x	Positive finite latent event times.
entry	Non-negative finite entry times, one per latent event.
censor	Post-entry censoring times, one per latent event. Positive infinity is allowed; every value must exceed its entry time.
keep_latent	If ‘TRUE‘, retain all latent units, including those excluded by truncation.

Details

Units excluded by ‘x <= entry‘ are left truncated, not censored. Their IDs remain absent from the observed table. For included units, ‘time‘ is ‘pmin(x, censor)‘ and ‘delta‘ is ‘as.integer(x <= censor)‘. Valid analysis must condition on survival to entry; this generator does not fit that likelihood. It assumes the supplied entry and post-entry censoring process is appropriate for the intended simulation.

Value

An ‘rcenscomp‘ object whose observed ‘data‘ has ‘id‘, ‘entry‘, ‘time‘, and ‘delta‘. Diagnostics distinguish truncation from post-entry censoring and give total post-entry time at risk.

References

Andersen, P. K., Borgan, O., Gill, R. D., and Keiding, N. (1993). *Statistical Models Based on Counting Processes*. Springer.

See Also

- rcenscomp_current_status

Examples

```
rcenscomp_delayed_entry(
  x = c(0.2, 0.8, 1.5, 2),
  entry = c(0.3, 0.1, 0.5, 1),
  censor = c(1, 1, 1.2, 3)
)
```

---

rcenscomp_generalized_hybrid

Apply generalized hybrid censoring

Description

Uses two failure thresholds and an intermediate planned time to bound the realized number of failures under the continuous-time convention.

Usage

```
rcenscomp_generalized_hybrid(x, T, r1, r2, keep_latent = TRUE)
```

Arguments

x	Positive finite latent event times.
T	Positive planned time threshold.
r1	Integer lower failure target.
r2	Integer upper failure target satisfying ‘r1 < r2 <= length(x)‘.
keep_latent	If ‘TRUE‘, retain exact latent event times.

Details

The stopping time is exactly ‘min(max(T, sort(x)[r1]), sort(x)[r2])‘. With continuous lifetimes, ‘r1 <= D <= r2‘. Ties at the stopping time are not broken silently and produce a warning because they can violate the nominal upper count.

Value

An ‘rcenscomp‘ object with subject-wise right-censored data and design fields ‘Tstar‘, ‘D‘, ‘r1‘, and ‘r2‘.

References

Chandrasekar, B., Childs, A., and Balakrishnan, N. (2004). Exact likelihood inference under generalized hybrid censoring. *Naval Research Logistics*, 51, 994-1004.

See Also

- rcenscomp_hybrid_type1
- rcenscomp_hybrid_type2

Examples

```
rcenscomp_generalized_hybrid(
  c(0.2, 0.5, 0.9, 1.4, 2), T = 0.8, r1 = 2, r2 = 4
)
```

`rcenscomp_hybrid_type1` *Apply Hybrid Type-I censoring*

Description

Stops a complete life test at the earlier of a time limit and the ‘r‘th ordered latent failure.

Usage

```
rcenscomp_hybrid_type1(x, T, r, keep_latent = TRUE)
```

Arguments

`x`	Positive finite latent event times.
`T`	Positive planned time limit.
`r`	Integer failure target between one and ‘length(x)‘.
`keep_latent`	If ‘TRUE‘, retain exact latent event times.

Details

The stopping time is ‘Tstar = min(T, sort(x)[r])‘. Subject-wise observations are ‘time = pmin(x, Tstar)‘ and ‘delta = as.integer(x <= Tstar)‘. The number of failures may be smaller than ‘r‘. Ties at the stopping time generate a warning because they can change the nominal event count; the design is theoretically stated for continuous lifetimes.

Value

An ‘rcenscomp‘ object with subject-wise ‘id‘, ‘time‘, and ‘delta‘, plus ‘Tstar‘ and the realized failure count ‘D‘.

References

Childs, A., Chandrasekar, B., Balakrishnan, N., and Kundu, D. (2003). Exact likelihood inference based on Type-I and Type-II hybrid censored samples. *Annals of the Institute of Statistical Mathematics*, 55, 319-330.

See Also

- rcenscomp_hybrid_type2
- rcenscomp_generalized_hybrid

Examples

```
rcenscomp_hybrid_type1(c(0.2, 0.5, 0.9, 1.4), T = 0.8, r = 3)
```

---

rcenscomp_hybrid_type2

Apply Hybrid Type-II censoring

Description

Stops a complete life test at the later of a time threshold and the ‘r‘th ordered latent failure.

Usage

```
rcenscomp_hybrid_type2(x, T, r, keep_latent = TRUE)
```

Arguments

x	Positive finite latent event times.
T	Positive planned time limit.
r	Integer failure target between one and ‘length(x)‘.
keep_latent	If ‘TRUE‘, retain exact latent event times.

Details

The stopping time is ‘Tstar = max(T, sort(x)[r])‘. Under continuous event times, at least ‘r‘ failures are observed. Ties at the stopping point are reported because they may produce more events than a nominal target.

Value

An ‘rcenscomp‘ object with subject-wise ‘id‘, ‘time‘, and ‘delta‘, plus ‘Tstar‘ and the realized failure count ‘D‘.

References

Balakrishnan, N. and Kundu, D. (2013). Hybrid censoring: models, inferential results and applications. *Computational Statistics & Data Analysis*, 57, 166-209.

See Also

- rcenscomp_hybrid_type1
- rcenscomp_generalized_hybrid

Examples

```
rcenscomp_hybrid_type2(c(0.2, 0.5, 0.9, 1.4), T = 0.8, r = 3)
```

---

rcenscomp_informative_frailty

Generate informative censoring through shared process frailty

---

Description

Generates event and censoring times that share an unobserved individual gamma frailty, with exponent ‘rho‘ in the censoring process.

Usage

```
rcenscomp_informative_frailty(
  n,
  beta_x,
  alpha_x,
  beta_c,
  alpha_c,
  gamma_x,
  gamma_c,
  theta,
  rho,
  z = NULL,
  keep_latent = TRUE
)
```

Arguments

`n`	Positive integer sample size.
`beta_x, alpha_x`	Positive Weibull event parameters.
`beta_c, alpha_c`	Positive Weibull censoring parameters.
`gamma_x, gamma_c`	Finite PH coefficient vectors of equal length.
`theta`	Positive frailty variance.
`rho`	Finite exponent controlling how frailty enters censoring.
`z`	Optional finite numeric covariate matrix with ‘n‘ rows.
`keep_latent`	If ‘TRUE‘, retain ‘x_true‘, ‘c_true‘, ‘frailty‘, and covariates.

Details

Frailty is ‘Gamma(shape = 1 / theta, scale = theta)‘. The event linear predictor adds ‘log(V_i)‘ and the censoring predictor adds ‘rho * log(V_i)‘. The two processes use separate uniforms conditional on frailty and covariates. This is a sensitivity generator, not an inferential correction, and the full joint mechanism is generally not identified by the observed ‘(time, delta, z)‘ table alone. The boundary ‘rho = 0‘ removes frailty from the censoring predictor, making censoring independent of the event time conditional on ‘z‘. At that boundary the censoring mechanism is ignorable, but this does not imply that an ordinary event model correctly handles omitted event frailty.

Value

An ‘rcenscomp‘ object with observed ‘id‘, ‘time‘, ‘delta‘, and covariates, plus separate latent process values when requested.

References

Lagakos, S. W. (1979). General right censoring and its impact on survival analysis. *Biometrics*, 35, 139-156.

See Also

- `rcenscomp_covariate_censoring`
- `rcenscomp_rweibull_ph_frailty`

Examples

```
set.seed(21)
rcenscomp_informative_frailty(
  n = 10,
  beta_x = 2.5, alpha_x = 1.5,
  beta_c = 1.3, alpha_c = 1.1,
  gamma_x = c(0.5, -0.3), gamma_c = c(-0.2, 0.4),
  theta = 0.6, rho = 0.8
)
```

`rcenscomp_interval` *Apply a common visit schedule to latent event times*

Description

Converts exact latent event times into left-, interval-, or right-censored records under a common inspection schedule.

Usage

```
rcenscomp_interval(x, visits, keep_latent = TRUE)
```

Arguments

`x`	Positive finite latent event times.
`visits`	Positive, strictly increasing common visit times.
`keep_latent`	If 'TRUE', retain the exact latent times.

Details

For visits '0 < v1 < ... < vK', an event at or before 'v1' is recorded as '(0, v1]'; an event after 'vK' is recorded as '(vK, Inf]'; all other events are assigned to the unique interval '(v[j-1], v[j]]'. This differs from the original [rcensI()] and [rcensIfix()] functions: those functions retain their own censoring-control strategies, while this function applies a fixed inspection operator to every supplied latent lifetime.

Value

An 'rcenscomp' object whose observed 'data' has columns 'id', 'L', 'R', and 'status'. Diagnostics include counts by censoring type and the finite interval widths.

References

Turnbull, B. W. (1976). The empirical distribution function with arbitrarily grouped, censored and truncated data. *JRSS B*, 38, 290-295.

Ramos, P. L., Kumar, I., Dutta, S., and Kundu, D. *Sampling with Complex Censored Data: A Practical Guide* (unpublished manuscript).

See Also

- `rcenscomp_current_status`
- `rcensI`
- `rcensIfix`

Examples

```
rcenscomp_interval(
  x = c(0.1, 0.2, 0.35, 0.6, 1.2),
  visits = c(0.2, 0.5, 1)
)
```

`rcenscomp_progressive_hybrid` *Apply time-capped progressive hybrid censoring*

Description

Follows a progressive failure-time withdrawal plan until the target failure count is reached or the next failure would occur after a time cap.

Usage

```
rcenscomp_progressive_hybrid(x, R, T, ties = "error", keep_latent = TRUE)
```

Arguments

`x`	Positive finite latent event times.
`R`	Non-negative integer progressive plan satisfying ‘sum(R) = length(x) - length(R)‘.
`T`	Positive time cap.
`ties`	Either ‘"error"‘ or ‘"first"‘.
`keep_latent`	If ‘TRUE‘, retain exact latent event times.

Details

‘m‘ is ‘length(R)‘. If the next eligible failure exceeds ‘T‘, all survivors are censored at ‘T‘. If the ‘m‘th failure is reached first, all remaining survivors are censored at that failure time. The returned ‘R_used‘ contains realized withdrawals after failures; ‘Rstar‘ is the terminal censoring count. ‘D = 0‘ is valid and is reported diagnostically rather than treated as an estimation-ready sample.

Value

An ‘rcenscomp‘ object with subject-wise data and design fields ‘D‘, ‘Tstar‘, ‘R_used‘, ‘Rstar‘, and the original ‘R‘.

References

Kundu, D. and Joarder, A. (2006). Analysis of Type-II progressively hybrid censored data. *Computational Statistics & Data Analysis*, 50, 2509-2528.

See Also

- rcenscomp_progressive_type2
- rcenscomp_adaptive_progressive_type2

Examples

```
set.seed(3)
rcenscomp_progressive_hybrid(
  x = c(0.2, 0.5, 0.9, 1.4, 2),
  R = c(1, 0, 1), T = 0.8
)
rcenscomp_progressive_hybrid(
  x = c(1, 2, 3), R = c(0, 0, 0), T = 0.5
)
```

---

rcenscomp_progressive_type1

Apply progressive Type-I censoring

Description

Continuously observes failures while withdrawing survivors at fixed times and censoring every remaining unit at the final scheduled time.

Usage

```
rcenscomp_progressive_type1(
  x,
  times,
  R_plan = integer(0),
  ties = "error",
  keep_latent = TRUE
)
```

Arguments

x	Positive finite latent event times.
times	Positive, strictly increasing withdrawal times; the last is the terminal time.
R_plan	Non-negative integer withdrawals for all intermediate times, of length ‘length(times) - 1‘.
ties	Either ‘"error"‘ or ‘"first"‘. Ties are rejected by default; ‘"first"‘ permits them without changing the simultaneous failure records.
keep_latent	If ‘TRUE‘, retain exact latent event times.

Details

Before each scheduled withdrawal, every still-alive unit with an event time at or before that schedule time is recorded at its exact event time. At an intermediate time, the realized withdrawal is the smaller of the planned count and the eligible risk set. At the final time, all survivors are censored. This is not progressively interval-censored data: failures are monitored continuously.

Value

An ‘rcenscomp‘ object with subject-wise right-censored data and realized withdrawals ‘R_used‘, including the terminal withdrawal.

References

Balakrishnan, N. and Cramer, E. (2014). *The Art of Progressive Censoring*. Birkhauser.

See Also

- rcenscomp_progressive_type2

Examples

```
set.seed(2)
rcenscomp_progressive_type1(
  x = c(0.1, 0.3, 0.7, 1.2, 2),
  times = c(0.25, 0.75, 1.5),
  R_plan = c(1, 0)
)
```

---

rcenscomp_progressive_type2

Apply progressive Type-II censoring

Description

Observes exactly ‘length(R)‘ failures and withdraws the planned numbers of eligible survivors at their corresponding failure times.

Usage

```
rcenscomp_progressive_type2(x, R, ties = "error", keep_latent = TRUE)
```

Arguments

x	Positive finite latent event times.
R	Non-negative integer withdrawal vector.
ties	Either ‘"error"‘ or ‘"first"‘.
keep_latent	If ‘TRUE‘, retain exact latent event times.

Details

At each observed failure, the failed unit is removed before ‘R[j]‘ survivors are sampled uniformly without replacement. Validity requires non-negative integer withdrawals with ‘sum(R) = n - length(R)‘. The default rejects tied latent times. With ‘ties = "first"‘, the smallest original ID is the deterministic failure among a tied risk set; this is a computational convention, not part of the continuous-time design.

Value

An ‘rcenscomp‘ object with subject-wise ‘id‘, ‘time‘, and ‘delta‘, and a compact table containing ‘failure_order‘, ‘failure_id‘, ‘failure_time‘, and ‘R_used‘.

References

Balakrishnan, N. and Aggarwala, R. (2000). *Progressive Censoring: Theory, Methods, and Applications*. Birkhauser.

See Also

- rcenscomp_progressive_type1
- rcenscomp_progressive_hybrid

Examples

```
set.seed(1)
rcenscomp_progressive_type2(
  x = c(0.2, 0.4, 0.8, 1.1, 1.5),
  R = c(1, 0, 1)
)
```

---

rcenscomp_rweibull_censor_ph

Generate covariate-dependent Weibull censoring times

Description

Generates censoring times from their own Weibull proportional-hazards model.

Usage

```
rcenscomp_rweibull_censor_ph(z, beta_c, alpha_c, gamma_c)
```

Arguments

z	Finite numeric covariate matrix.
beta_c	Positive censoring baseline rate-like parameter.
alpha_c	Positive censoring shape parameter.
gamma_c	Finite censoring regression coefficient vector compatible with the columns of ‘z‘.

Details

Each call consumes a new independent set of uniforms. Independence from an event generator therefore follows when the caller supplies a common ‘z‘ and invokes the two generators sequentially without reusing random draws.

Value

A numeric vector of censoring times.

See Also

- rcenscomp_rweibull_ph
- rcenscomp_covariate_censoring

Examples

```
set.seed(12)
z <- cbind(z1 = c(0, 1, 0), z2 = c(-1, 0, 1))
rcenscomp_rweibull_censor_ph(z, 1.2, 1.1, c(-0.2, 0.4))
```

---

rcenscomp_rweibull_ph *Generate Weibull proportional-hazards lifetimes*

Description

Generates Weibull event times with cumulative hazard ‘beta * t^alpha * exp(z

Usage

```
rcenscomp_rweibull_ph(n, beta, alpha, gamma, z = NULL)
```

Arguments

n	Positive integer sample size.
beta	Positive Weibull baseline rate-like parameter.
alpha	Positive Weibull shape parameter.
gamma	Finite regression coefficient vector.
z	Optional finite numeric covariate matrix with ‘n‘ rows.

Details

The event time is ‘(-log(U) / (beta * exp(z_i^T gamma)))^(1 / alpha)‘. If ‘z = NULL‘, the documented didactic default generates one Bernoulli(0.5) and one standard normal covariate, so ‘gamma‘ must then have length two. A supplied ‘z‘ may have any number of numeric columns compatible with ‘gamma‘. No intercept is added and no seed is set.

Value

A data frame containing ‘id‘, latent event time ‘x‘, and all covariates.

References

Bender, R., Augustin, T., and Blettner, M. (2005). Generating survival times to simulate Cox proportional hazards models. *Statistics in Medicine*, 24, 1713-1723.

See Also

- rcenscomp_rweibull_rate
- rcenscomp_rweibull_censor_ph
- rcenscomp_rweibull_ph_frailty

Examples

```
set.seed(11)
rcenscomp_rweibull_ph(
  5, beta = 2.5, alpha = 1.5, gamma = c(0.5, -0.3)
)
```

---

rcenscomp_rweibull_ph_frailty

Generate clustered Weibull PH lifetimes with shared gamma frailty

---

Description

Generates a latent event-time model in which all units in a cluster share one gamma frailty. This is an event generator, not a censoring mechanism.

Usage

```
rcenscomp_rweibull_ph_frailty(
  cluster_size,
  beta,
  alpha,
  gamma,
  theta,
  z = NULL
)
```

Arguments

cluster_size	Positive integer sizes for each cluster.
beta	Positive Weibull baseline rate-like parameter.
alpha	Positive Weibull shape parameter.
gamma	Finite regression coefficient vector.
theta	Positive gamma frailty variance.
z	Optional finite numeric covariate matrix with ‘sum(cluster_size)‘ rows.

Details

Frailties use ‘shape = 1 / theta‘ and ‘scale = theta‘, giving mean one and variance ‘theta‘. The event linear predictor adds ‘log(V_cluster)‘. If ‘z = NULL‘, the same two-covariate default as [rcenscomp_rweibull_ph()] is used. The marginal event hazard after integrating frailty is not the baseline conditional hazard.

Value

A data frame with ‘id‘, ‘cluster‘, latent ‘x‘, repeated cluster ‘frailty‘, and covariates.

References

Hougaard, P. (2000). *Analysis of Multivariate Survival Data*. Springer.

See Also

- rcenscomp_rweibull_ph
- rcenscomp_informative_frailty

Examples

```
set.seed(13)
rcenscomp_rweibull_ph_frailty(
  cluster_size = c(2, 3), beta = 2.5, alpha = 1.5,
  gamma = c(0.5, -0.3), theta = 0.6
)
```

---

rcenscomp_rweibull_rate

Generate Weibull lifetimes under the manuscript rate parameterization

---

Description

Generates independent Weibull event times with density ‘alpha * beta * x^(alpha - 1) * exp(-beta * x^alpha)‘.

Usage

```
rcenscomp_rweibull_rate(n, beta, alpha)
```

Arguments

n	Positive integer sample size.
beta	Positive Weibull rate-like parameter.
alpha	Positive Weibull shape parameter.

Details

The inverse transform is '(-log(U) / beta)^(1 / alpha)'. Here 'alpha' is shape and 'beta' is rate-like. The equivalent [stats::rweibull()] scale is 'beta^(-1 / alpha)'. A log-scale equivalent is used when the direct calculation has an invalid intermediate result but the lifetime remains representable as a positive finite double. Unrepresentable results raise an error. No seed is set internally.

Value

A numeric vector of positive latent lifetimes.

References

Weibull, W. (1951). A statistical distribution function of wide applicability. *Journal of Applied Mechanics*, 18, 293-297.

See Also

- rweibull
- rcenscomp_rweibull_ph

Examples

```
set.seed(10)
rcenscomp_rweibull_rate(5, beta = 2.5, alpha = 1.5)
```

---

rcensI *Generate interval censoring sample*

Description

Generator of interval censored samples where the length of interval is random, given a generator of samples of the distribution X (rdistrX) with parameters appended by the list param_X. Generator sample of distribution C (censoring) with parameters appended by the list param_C In which, you can control the desired censorship percentage.

Usage

```
rcensI(
  rdistrX,
  rdistrC,
  param_X,
  param_C,
  n = 10000,
  epsilon = 0.5,
  n_mc = 10000,
  theta = 1,
  verbose = FALSE
)
```

Arguments

rdistrX	sample generator of distribution X. First argument number of samples, next arguments in param_X.
rdistrC	sample generator of distribution C. First argument number of samples, next arguments in param_C.
param_X	list with parameters of rdistrX function.
param_C	list with parameters of rdistrC function, one of these parameters should be "lambda", this will be the searched parameter.
n	number of sample to create.
epsilon	Parameter to estimate the number of visit (in [0,1]), shrink it only if the algorithm takes too long
n_mc	number of sample use to estimate the moments of C, greater n_mc more accuracy.
theta	Desired censoring percentage
verbose	if TRUE print a censoring percentage of new created database.

Value

A list with sample data information:

sample_censored	vector of censored sample
sample_uncensored	vector of uncensored sample (original)
censored_indicator	vector of 1 and 0 indicating whether the i-th sample is censored 1:= no censored, 0:= censored
n_censored	number of censored samples

Author(s)

Daniel Saavedra Morales

See Also

rcensT1 for generate censorship sample type I.
rcensT2 for generate censorship sample type II.
rcensT3 for generate censorship sample type III
rcensIfix for generate interval censoring sample with fix length interval

Examples

```
#Example Exponential - Uniform

Data_I = rcensI(rdistrX = rexp, rdistrC = runif,
               param_X = list("rate" = 2),
              param_C = list("min" = 0, "max" = 1),
             n = 1e02, theta = .9)

## Example with plot in examples_plot/Example_rcensI_plot.R
```

`rcensIfix` *Generate interval censoring sample (Fix)*

Description

Generator of interval censored samples where the length of interval is fixed, given a generator of samples of the distribution X (rdistrX) with parameters appended by the list param_X. In which, you can control the desired censorship percentage.

Usage

```
rcensIfix(
  rdistrX,
  param_X,
  interval_length,
  n = 10000,
  theta = 1,
  verbose = FALSE
)
```

Arguments

`rdistrX`	sample generator of distribution X. First argument number of samples, next arguments in param_X.
`param_X`	list with parameters of rdistrX function.
`interval_length`	length of interval
`n`	number of sample to create.
`theta`	Desired censoring percentage
`verbose`	if TRUE print a censoring percentage of new created database.

Value

A list with sample data information:

`sample_censored`	vector of censored sample
`sample_uncensored`	vector of uncensored sample (original)
`censored_indicator`	vector of 1 and 0 indicating whether the i-th sample is censored 1:= no censored, 0:= censored
`n_censored`	number of censored samples

Author(s)

Daniel Saavedra Morales

See Also

`rcensT1` for generate censorship sample type I.
`rcensT2` for generate censorship sample type II.
`rcensT3` for generate censorship sample type III
`rcensI` for generate interval censoring sample with random length interval

Examples

```
#Example Exponential - Fix Interval

Data_Ifix = rcensIfix(rdistrX = rexp, interval_length = 2,
                  param_X = list("rate" = .5),
                  n = 1e02, theta = .9)

## Example with plot in examples_plot/Example_rcensIfix_plot.R
```

`rcensT1` *Generate Censoring Sample, Type I*

Description

Generator of censored samples type I with right or left censoring, given a generator of samples of the distribution X (rdistrX) with parameters appended by the list param_X. In which, you can control the censorship time or the desired censorship percentage.

Usage

```
rcensT1(
  rdistrX,
  param_X,
  qdistrX = NULL,
  n = 10000,
  t_censored = -1,
  theta = 0.5,
  verbose = FALSE,
  right = TRUE
)
```

Arguments

rdistrX	sample generator of distribution X. First argument number of samples, next arguments in param_X.
param_X	list with parameters of rdistrX function.
qdistrX	quantile function of X.
n	number of sample to create.
t_censored	time level censored.
theta	Desired censoring percentage.
verbose	if TRUE print a censoring percentage of new created database.
right	if TRUE create right-censored data, else create left-censored

Value

A list with sample data information:

sample_censored	vector of censored sample
sample_uncensored	vector of uncensored sample (original)
censored_indicator	vector of 1 and 0 indicating whether the i-th sample is censored 1:= no censored, 0:= censored
censored_time	vector of censorship time
n_censored	number of censored samples

Author(s)

Daniel Saavedra Morales

See Also

rcensT2 for generate censorship sample type II.
rcensT3 for generate censorship sample type III
rcensI for generate interval censoring sample with random length interval
rcensIfix for generate interval censoring sample with fix length interval

Examples

```
## Example Exponential
## time censored

Data_T1 = rcensT1(rdistrX = rexp, param_X = list("rate" = 2),
                  n = 1e02, t_censored = 1)

## time censored estimate with desired censoring percentage.

Data_T1 = rcensT1(rdistrX = rexp, param_X = list("rate" = 2),
                  qdistrX = qexp, n = 1e02, theta = .8)

## Example with plot in examples_plot/Example_rcensT1_plot.R
```

`rcensT2` *Generate Censoring Sample, Type II*

Description

Generator of censored samples type II with right or left censoring, given a generator of samples of the distribution X (rdistrX) with parameters appended by the list param_X. In which, you can control the number of censored sample or the desired censorship percentage.

Usage

```
rcensT2(
  rdistrX,
  param_X,
  n = 10000,
  m_censored = -1,
  theta = 0.5,
  verbose = FALSE,
  right = TRUE
)
```

Arguments

`rdistrX`	sample generator of distribution X. \ First argument number of samples, next arguments in param_X.
`param_X`	list with parameters of rdistrX function.
`n`	number of sample to create.
`m_censored`	number of sample censored. m_censored < n. \ If m_censored <= n, m_censored is estimate with the desired censoring percentage.
`theta`	Desired censoring percentage.
`verbose`	if TRUE print a censoring percentage of new created database.
`right`	if TRUE create right-censored data, else create left-censored

Value

A list with sample data information:

`sample_censored`	vector of censored sample
`sample_uncensored`	vector of uncensored sample (original)
`censored_indicator`	vector of 1 and 0 indicating whether the i-th sample is censored 1:= no censored, 0:= censored
`censored_time`	vector of censorship time
`n_censored`	number of censored samples

Author(s)

Daniel Saavedra Morales

See Also

`rcensT1` for generate censorship sample type I.
`rcensT3` for generate censorship sample type III
`rcensI` for generate interval censoring sample with random length interval
`rcensIfix` for generate interval censoring sample with fix length interval

Examples

```
##Example Exponential

## Number of sample censored
Data_T2 = rcensT2(rdistrX = rexp, param_X = list("rate" = 2), n = 1e02, m_censored = 9)

## Number of censored sample estimate with desired censoring percentage.
Data_T2 = rcensT2(rdistrX = rexp, param_X = list("rate" = 2), n = 1e02, theta = .8)


## Example with plot in examples_plot/Example_rcensT2_plot.R
```

`rcensT3` *Generate Censoring Sample,Type III (Random)*

Description

Generator of censored samples type III with right or left censoring, given a generator of samples of the distribution X (rdistrX) with parameters appended by the list param_X. Also accumulate function of distribution and generator sample of distribution C (censoring) with parameters appended by the list param_C. In which, you can control the desired censorship percentage.

Usage

```
rcensT3(
  rdistrX,
  pdistrC,
  rdistrC,
  param_X,
  param_C,
  n = 10000,
  theta = 0.5,
  n_mc = 10000,
  lambda_tol = c(1e-06, 10000),
  verbose = FALSE,
  right = TRUE
)
```

Arguments

rdistrX	sample generator of distribution X. First argument number of samples, next arguments in param_X.
pdistrC	function distribution of C. First argument probabilities, next arguments in param_C.
rdistrC	sample generator of distribution C. First argument number of samples, next arguments in param_C.
param_X	list with parameters of rdistrX function.
param_C	list with parameters of rdistrC function, one of these parameters should be "lambda", this wil be the searched parameter.
n	number of sample to create.
theta	Desired censoring percentage
n_mc	number of sample use in Monte Carlo integration, greater n_mc more accuracy.
lambda_tol	lowest and uppest value where live the search parameter lambda.
verbose	if TRUE print a censoring percentage of new created database.
right	if TRUE create right-censored data, else create left-censored

Value

A list with sample data information:

lambda	searched censoring distribution parameter.
sample_censored	vector of censored sample.
sample_uncensored	vector of uncensored sample (original).
censored_indicator	vector of 1 and 0 indicating whether the i-th sample is censored. 1:= no censored, 0:= censored
censored_time	vector of censorship time.
n_censored	number of censored samples.

Author(s)

Daniel Saavedra Morales

See Also

rcensT1 for generate censorship sample type I.
rcensT2 for generate censorship sample type II
rcensI for generate interval censoring sample with random length interval
rcensIfix for generate interval censoring sample with fix length interval

Examples

```
#Example Exponential - Uniform


Data_T3 = rcensT3(rdistrX = rexp, pdistrC = punif, rdistrC = runif,
             param_X = list("rate" = 2),
            param_C = list("min" = 0, "max" = "lambda"),
           n = 1e02, theta = .9, right = TRUE)


Data_T3 = rcensT3(rdistrX = rexp, pdistrC = punif, rdistrC = runif,
                param_X = list("rate" = 2),
                param_C = list("min" = 0, "max" = "lambda"),
                n = 1e02, theta = .1, right = FALSE)


## Example with plot in examples_plot/Example_rcensT3_plot.R
```

---

rcuref *Title Generate Sample with Cure Fraction*

Description

Generator Sample with Cure Fraction, given a generator of samples of the distribution X (rdistrX) with parameters appended by the list param_X. Also the proportion of cure desired p.

Usage

```
rcuref(rdistrX, param_X, n = 10000, p = 0.5)
```

Arguments

rdistrX	sample generator of distribution X. First argument number of samples, next arguments in param_X.
param_X	list with parameters of rdistrX function.
n	number of sample to create.
p	cure fraction

Value

A list with sample data information:

data_cf	vector of cure fraction sample.
cure_list	vector of 1 and 0 indicating whether the i-th sample is cured. 1:= cure , 0:= no cure
cure_fraction	cure fraction used to create de sample.

Author(s)

Daniel Saavedra Morales

Examples

```
#Example Exponential Cure Fraction p = 0.5

Data = rcuref(rdistrX = rexp, param_X = list("rate" = 1),
              n = 1000, p = 0.5)
```

---

rcurefT3 *Title Generate Sample with Cure Fraction and Random Censoring*

Description

Generator Sample with Cure Fraction, Random Censoring. Given a generator of samples of the distribution X (rdistrX) with parameters appended by the list param_X. Also accumulate function of distribution and generator sample of distribution C (censoring) with parameters appended by the list param_C In which, you can control the desired censorship percentage.
Note: cure fraction (p) must be less than desired censorship percentage.

Usage

```
rcurefT3(
  rdistrX,
  pdistrC,
  rdistrC,
  param_X,
  param_C,
  p = 0.1,
  n = 10000,
  theta = 0.5,
  n_mc = 10000,
  lambda_tol = c(1e-06, 10000),
  verbose = FALSE,
  right = TRUE
)
```

Arguments

rdistrX	sample generator of distribution X. First argument number of samples, next arguments in param_X.
pdistrC	function distribution of C. First argument probabilities, next arguments in param_C.
rdistrC	sample generator of distribution C. First argument number of samples, next arguments in param_C.
param_X	list with parameters of rdistrX function.
param_C	list with parameters of rdistrC function, one of these parameters should be "lambda", this will be the searched parameter.
p	cure fraction
n	number of sample to create.
theta	Desired censoring percentage
n_mc	number of sample use in Monte Carlo integration, greater n_mc more accuracy.
lambda_tol	lowest and uppest value where live the search parameter lambda.
verbose	if TRUE print a censoring percentage of new created database.
right	if TRUE create right-censored data, else create left-censored

Value

A list with sample data information:

lambda	searched censoring distribution parameter.
sample_censored	vector of censored sample.
sample_uncensored	vector of uncensored sample (original).
censored_indicator	vector of 1 and 0 indicating whether the i-th sample is censored. 1:= no censored, 0:= censored

censored_time	vector of censorship time.
n_censored	number of censored samples.
cure_list	vector of 1 and 0 indicating whether the i-th sample is cured. 1:= cure , 0:= no cure
cure_fraction	cure fraction used to create de sample.

Author(s)

Daniel Saavedra Morales

See Also

rcuref Generate Sample with Cure Fraction.

Examples

```
#Example Exponential - Uniform


Data_T3 = rcurefT3(rdistrX = rexp, pdistrC = punif, rdistrC = runif,
                   param_X = list("rate" = 2),
                   param_C = list("min" = 0, "max" = "lambda"),
                   n = 1e02, theta = .9, p = 0.2)



## Example with plot in examples_plot/Example_rcurefT3_plot.R
```

Index