Package: primarycensored (via r-universe)

October 9, 2024

Title Primary Event Censored Distributions in R and Stan

Version 0.6.0

Description This package provides R functions for working with primary event censored distributions and Stan implementations for use in Bayesian modeling. Primary event censored distributions are useful for modeling delayed reporting scenarios in epidemiology and other fields. It provides support for arbitrary delay distributions, a range of common primary distributions, and allows for truncation and secondary event censoring to be accounted for. In addition, it provides both frequentist and Bayesian methods for fitting primary event censored distributions to data.

License MIT + file LICENSE

URL https://primarycensored.epinowcast.org, https://github.com/epinowcast/primarycensored/

BugReports https://github.com/epinowcast/primarycensored/issues/

Depends R (>= 4.0.0)

Imports pracma

Suggests bookdown, cmdstanr, dplyr, fitdistrplus, knitr, ggplot2, rmarkdown, spelling, testthat (>= 3.1.9), usethis, withr

Additional_repositories https://stan-dev.r-universe.dev

Config/Needs/hexsticker hexSticker, sysfonts, ggplot2

Config/Needs/website r-lib/pkgdown, epinowcast/enwtheme

Config/testthat/edition 3

Encoding UTF-8

Language en-GB

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

VignetteBuilder knitr

Repository https://epinowcast.r-universe.dev

RemoteUrl https://github.com/epinowcast/primarycensored

RemoteRef v0.6.0

RemoteSha 394db8f23b5669433c22149ea0b93ef72bacd4d8

Contents

`check_dprimary` *Check if a function is a valid bounded probability density function (PDF)*

Description

This function tests whether a given function behaves like a valid PDF by checking if it integrates to approximately 1 over the specified range and if it takes the arguments min and max.

Usage

```
check_dprimary(dprimary, pwindow, dprimary_args = list(), tolerance = 0.001)
```

Arguments

dprimary	Function to generate the probability density function (PDF) of primary event times. This function should take a value x and a pwindow parameter, and return a probability density. It should be normalized to integrate to 1 over [0, pwindow]. Defaults to a uniform distribution over [0, pwindow]. Users can provide custom functions or use helper functions like dexpgrowth for an exponential growth distribution. See primary_dists.R for examples.
pwindow	Primary event window
dprimary_args	List of additional arguments to be passed to dprimary. For example, when using dexpgrowth, you would pass list(min = 0, max = pwindow, r = 0.2) to set the minimum, maximum, and rate parameters
tolerance	The tolerance for the integral to be considered close to 1

Value

NULL. The function will stop execution with an error message if dprimary is not a valid PDF.

See Also

Distribution checking functions check_pdist(), check_truncation()

Examples

```
check_dprimary(dunif, pwindow = 1)
```

---

check_pdist *Check if a function is a valid cumulative distribution function (CDF)*

Description

This function tests whether a given function behaves like a valid CDF by checking if it's monotonically increasing and bounded between 0 and 1.

Usage

```
check_pdist(pdist, D, ...)
```

Arguments

pdist	Distribution function (CDF)
D	Maximum delay (truncation point). If finite, the distribution is truncated at D. If set to Inf, no truncation is applied. Defaults to Inf.
...	Additional arguments to be passed to pdist

Value

NULL. The function will stop execution with an error message if pdist is not a valid CDF.

See Also

Distribution checking functions `check_dprimary()`, `check_truncation()`

Examples

```
check_pdist(pnorm, D = 10)
```

---

check_truncation *Check if truncation time is appropriate relative to the maximum delay*

Description

This function checks if the truncation time D is appropriate relative to the maximum delay. If D is much larger than necessary, it suggests considering setting it to `Inf` for better efficiency with minimal accuracy cost.

Usage

```
check_truncation(delays, D, multiplier = 2)
```

Arguments

`delays`	A numeric vector of delay times
`D`	The truncation time
`multiplier`	The multiplier for the maximum delay to compare with D. Default is 2.

Value

Invisible NULL. Prints a message if the condition is met.

See Also

Distribution checking functions `check_dprimary()`, `check_pdist()`

Examples

```
check_truncation(delays = c(1, 2, 3, 4), D = 10, multiplier = 2)
```

`dprimarycensored` *Compute the primary event censored PMF for delays*

Description

This function computes the primary event censored probability mass function (PMF) for a given set of quantiles. It adjusts the PMF of the primary event distribution by accounting for the delay distribution and potential truncation at a maximum delay (D). The function allows for custom primary event distributions and delay distributions.

Usage

```
dprimarycensored(
  x,
  pdist,
  pwindow = 1,
  swindow = 1,
  D = Inf,
  dprimary = stats::dunif,
  dprimary_args = list(),
  log = FALSE,
  pdist_name = NULL,
  dprimary_name = NULL,
  ...
)

dpcens(
  x,
  pdist,
  pwindow = 1,
  swindow = 1,
  D = Inf,
  dprimary = stats::dunif,
  dprimary_args = list(),
  log = FALSE,
  pdist_name = NULL,
  dprimary_name = NULL,
  ...
)
```

Arguments

`x`	Vector of quantiles
`pdist`	Distribution function (CDF)
`pwindow`	Primary event window
`swindow`	Secondary event window (default: 1)

`D`	Maximum delay (truncation point). If finite, the distribution is truncated at D. If set to Inf, no truncation is applied. Defaults to Inf.
`dprimary`	Function to generate the probability density function (PDF) of primary event times. This function should take a value `x` and a `pwindow` parameter, and return a probability density. It should be normalized to integrate to 1 over [0, pwindow]. Defaults to a uniform distribution over [0, pwindow]. Users can provide custom functions or use helper functions like `dexpgrowth` for an exponential growth distribution. See `primary_dists.R` for examples.
`dprimary_args`	List of additional arguments to be passed to dprimary. For example, when using `dexpgrowth`, you would pass `list(min = 0, max = pwindow, r = 0.2)` to set the minimum, maximum, and rate parameters
`log`	Logical; if TRUE, probabilities p are given as log(p)
`pdist_name`	A string specifying the name of the delay distribution function. If NULL, the function name is extracted using `.extract_function_name()`. Used to determine if a analytical solution exists for the primary censored distribution. Must be set if `pdist` is passed a pre-assigned variable rather than a function name.
`dprimary_name`	A string specifying the name of the primary event distribution function. If NULL, the function name is extracted using `.extract_function_name()`. Used to determine if a analytical solution exists for the primary censored distribution. Must be set if `dprimary` is passed a pre-assigned variable rather than a function name.
`...`	Additional arguments to be passed to the distribution function

Details

The primary event censored PMF is computed by taking the difference of the primary event censored cumulative distribution function (CDF) at two points, $d + \text{swindow}$ and d. The primary event censored PMF, $f_{\text{cens}}(d)$, is given by:

$$f_{\text{cens}}(d) = F_{\text{cens}}(d + \text{swindow}) - F_{\text{cens}}(d)$$

where F_{cens} is the primary event censored CDF.

The function first computes the CDFs for all unique points (including both d and $d +$ swindow) using `pprimarycensored()`. It then creates a lookup table for these CDFs to efficiently calculate the PMF for each input value. For non-positive delays, the function returns 0.

If a finite maximum delay D is specified, the PMF is normalized to ensure it sums to 1 over the range [0, D]. This normalization can be expressed as:

$$f_{\text{cens,norm}}(d) = \frac{f_{\text{cens}}(d)}{\sum_{i=0}^{D-1} f_{\text{cens}}(i)}$$

where $f_{\text{cens,norm}}(d)$ is the normalized PMF and $f_{\text{cens}}(d)$ is the unnormalized PMF. For the explanation and mathematical details of the CDF, refer to the documentation of `pprimarycensored()`.

Value

Vector of primary event censored PMFs, normalized by D if finite (truncation adjustment)

See Also

Primary event censored distribution functions `pprimarycensored()`, `rprimarycensored()`

Examples

```
# Example: Weibull distribution with uniform primary events
dprimarycensored(c(0.1, 0.5, 1), pweibull, shape = 1.5, scale = 2.0)

# Example: Weibull distribution with exponential growth primary events
dprimarycensored(
  c(0.1, 0.5, 1), pweibull,
  dprimary = dexpgrowth,
  dprimary_args = list(r = 0.2), shape = 1.5, scale = 2.0
)
```

`expgrowth` *Exponential growth distribution functions*

Description

Density, distribution function, and random generation for the exponential growth distribution.

Usage

```
dexpgrowth(x, min = 0, max = 1, r, log = FALSE)

pexpgrowth(q, min = 0, max = 1, r, lower.tail = TRUE, log.p = FALSE)

rexpgrowth(n, min = 0, max = 1, r)
```

Arguments

`x`, `q`	Vector of quantiles.
`min`	Minimum value of the distribution range. Default is 0.
`max`	Maximum value of the distribution range. Default is 1.
`r`	Rate parameter for the exponential growth.
`log`, `log.p`	Logical; if TRUE, probabilities p are given as log(p).
`lower.tail`	Logical; if TRUE (default), probabilities are P[X x], otherwise, P[X > x].
`n`	Number of observations. If `length(n) > 1`, the length is taken to be the number required.

Details

The exponential growth distribution is defined on the interval [min, max] with rate parameter (r). Its probability density function (PDF) is:

$$f(x) = \frac{r \cdot \exp(r \cdot (x - min))}{\exp(r \cdot max) - \exp(r \cdot min)}$$

The cumulative distribution function (CDF) is:

$$F(x) = \frac{\exp(r \cdot (x - min)) - \exp(r \cdot min)}{\exp(r \cdot max) - \exp(r \cdot min)}$$

For random number generation, we use the inverse transform sampling method:

1. Generate $u \sim \text{Uniform}(0, 1)$
2. Set $F(x) = u$ and solve for x:

$$x = min + \frac{1}{r} \cdot \log(u \cdot (\exp(r \cdot max) - \exp(r \cdot min)) + \exp(r \cdot min))$$

This method works because of the probability integral transform theorem, which states that if X is a continuous random variable with CDF $F(x)$, then $Y = F(X)$ follows a Uniform$(0, 1)$ distribution. Conversely, if U is a Uniform$(0, 1)$ random variable, then $F^{-1}(U)$ has the same distribution as X, where F^{-1} is the inverse of the CDF.

In our case, we generate u from Uniform$(0, 1)$, then solve $F(x) = u$ for x to get a sample from our exponential growth distribution. The formula for x is derived by algebraically solving the equation:

$$u = \frac{\exp(r \cdot (x - min)) - \exp(r \cdot min)}{\exp(r \cdot max) - \exp(r \cdot min)}$$

When r is very close to 0 ($|r| < 1e{-}10$), the distribution approximates a uniform distribution on [min, max], and we use a simpler method to generate samples directly from this uniform distribution.

Value

`dexpgrowth` gives the density, `pexpgrowth` gives the distribution function, and `rexpgrowth` generates random deviates.

The length of the result is determined by `n` for `rexpgrowth`, and is the maximum of the lengths of the numerical arguments for the other functions.

Examples

```
x <- seq(0, 1, by = 0.1)
probs <- dexpgrowth(x, r = 0.2)
cumprobs <- pexpgrowth(x, r = 0.2)
samples <- rexpgrowth(100, r = 0.2)
```

`fitdistdoublecens` *Fit a distribution to doubly censored data*

Description

This function wraps the custom approach for fitting distributions to doubly censored data using fitdistrplus and primarycensored.

Usage

```
fitdistdoublecens(
  censdata,
  distr,
  pwindow = 1,
  D = Inf,
  dprimary = stats::dunif,
  dprimary_name = NULL,
  dprimary_args = list(),
  truncation_check_multiplier = 2,
  ...
)
```

Arguments

`censdata`	A data frame with columns 'left' and 'right' representing the lower and upper bounds of the censored observations. Unlike `fitdistrplus::fitdistcens()` `NA` is not supported for either the upper or lower bounds.
`distr`	A character string naming the distribution to be fitted.
`pwindow`	Primary event window
`D`	Maximum delay (truncation point). If finite, the distribution is truncated at D. If set to Inf, no truncation is applied. Defaults to Inf.
`dprimary`	Function to generate the probability density function (PDF) of primary event times. This function should take a value `x` and a `pwindow` parameter, and return a probability density. It should be normalized to integrate to 1 over [0, pwindow]. Defaults to a uniform distribution over [0, pwindow]. Users can provide custom functions or use helper functions like `dexpgrowth` for an exponential growth distribution. See `primary_dists.R` for examples.
`dprimary_name`	A string specifying the name of the primary event distribution function. If NULL, the function name is extracted using `.extract_function_name()`. Used to determine if a analytical solution exists for the primary censored distribution. Must be set if `dprimary` is passed a pre-assigned variable rather than a function name.
`dprimary_args`	List of additional arguments to be passed to dprimary. For example, when using `dexpgrowth`, you would pass `list(min = 0, max = pwindow, r = 0.2)` to set the minimum, maximum, and rate parameters

`truncation_check_multiplier`
: Numeric multiplier to use for checking if the truncation time D is appropriate relative to the maximum delay. Set to NULL to skip the check. Default is 2.

`...`
: Additional arguments to be passed to `fitdistrplus::fitdist()`.

Details

This function temporarily assigns and then removes functions from the global environment in order to work with fitdistr. Users should be aware of this behaviour, especially if they have existing functions with the same names in their global environment.

Value

An object of class "fitdist" as returned by fitdistrplus::fitdist.

See Also

Modelling wrappers for external fitting packages `pcd_as_stan_data()`, `pcd_cmdstan_model()`

Examples

```
# Example with normal distribution
set.seed(123)
n <- 1000
true_mean <- 5
true_sd <- 2
pwindow <- 2
swindow <- 2
D <- 10
samples <- rprimarycensored(
  n, rnorm,
  mean = true_mean, sd = true_sd,
  pwindow = pwindow, swindow = swindow, D = D
)

delay_data <- data.frame(
  left = samples,
  right = samples + swindow
)

fit_norm <- fitdistdoublecens(
  delay_data,
  distr = "norm",
  start = list(mean = 0, sd = 1),
  D = D, pwindow = pwindow
)

summary(fit_norm)
```

`new_pcens` *S3 class for primary event censored distribution computation*

Description

S3 class for primary event censored distribution computation

Usage

```
new_pcens(
  pdist,
  dprimary,
  dprimary_args,
  pdist_name = NULL,
  dprimary_name = NULL,
  ...
)
```

Arguments

`pdist`	Distribution function (CDF)
`dprimary`	Function to generate the probability density function (PDF) of primary event times. This function should take a value `x` and a `pwindow` parameter, and return a probability density. It should be normalized to integrate to 1 over [0, pwindow]. Defaults to a uniform distribution over [0, pwindow]. Users can provide custom functions or use helper functions like `dexpgrowth` for an exponential growth distribution. See `primary_dists.R` for examples.
`dprimary_args`	List of additional arguments to be passed to dprimary. For example, when using `dexpgrowth`, you would pass `list(min = 0, max = pwindow, r = 0.2)` to set the minimum, maximum, and rate parameters
`pdist_name`	A string specifying the name of the delay distribution function. If NULL, the function name is extracted using `.extract_function_name()`. Used to determine if a analytical solution exists for the primary censored distribution. Must be set if `pdist` is passed a pre-assigned variable rather than a function name.
`dprimary_name`	A string specifying the name of the primary event distribution function. If NULL, the function name is extracted using `.extract_function_name()`. Used to determine if a analytical solution exists for the primary censored distribution. Must be set if `dprimary` is passed a pre-assigned variable rather than a function name.
`...`	Additional arguments to be passed to pdist

Value

An object of class pcens_pdist_name_dprimary_name

See Also

Low level primary event censored distribution objects and methods `pcens_cdf()`, `pcens_cdf.default()`, `pcens_cdf.pcens_pgamma_dunif()`, `pcens_cdf.pcens_plnorm_dunif()`, `pcens_cdf.pcens_pweibull_dunif()`

`pcd_as_stan_data` *Prepare data for primarycensored Stan model*

Description

This function takes in delay data and prepares it for use with the primarycensored Stan model.

Usage

```
pcd_as_stan_data(
  data,
  delay = "delay",
  delay_upper = "delay_upper",
  n = "n",
  pwindow = "pwindow",
  relative_obs_time = "relative_obs_time",
  dist_id,
  primary_id,
  param_bounds,
  primary_param_bounds,
  priors,
  primary_priors,
  compute_log_lik = FALSE,
  use_reduce_sum = FALSE,
  truncation_check_multiplier = 2
)
```

Arguments

`data`	A data frame containing the delay data.
`delay`	Column name for observed delays (default: "delay")
`delay_upper`	Column name for upper bound of delays (default: "delay_upper")
`n`	Column name for count of observations (default: "n")
`pwindow`	Column name for primary window (default: "pwindow")
`relative_obs_time`	Column name for relative observation time (default: "relative_obs_time")
`dist_id`	Integer identifying the delay distribution: 1 = Lognormal, 2 = Gamma, 3 = Weibull, 4 = Exponential, 5 = Generalized Gamma, 6 = Negative Binomial, 7 = Poisson, 8 = Bernoulli, 9 = Beta, 10 = Binomial, 11 = Categorical, 12 = Cauchy, 13 = Chi-square, 14 = Dirichlet, 15 = Gumbel, 16 = Inverse Gamma, 17 = Logistic

`primary_id` Integer identifying the primary distribution: 1 = Uniform, 2 = Exponential growth

`param_bounds` A list with elements `lower` and `upper`, each a numeric vector specifying bounds for the delay distribution parameters.

`primary_param_bounds` A list with elements `lower` and `upper`, each a numeric vector specifying bounds for the primary distribution parameters.

`priors` A list with elements `location` and `scale`, each a numeric vector specifying priors for the delay distribution parameters.

`primary_priors` A list with elements `location` and `scale`, each a numeric vector specifying priors for the primary distribution parameters.

`compute_log_lik` Logical; compute log likelihood? (default: FALSE)

`use_reduce_sum` Logical; use reduce_sum for performance? (default: FALSE)

`truncation_check_multiplier` Numeric multiplier to use for checking if the truncation time D is appropriate relative to the maximum delay for each unique D value. Set to NULL to skip the check. Default is 2.

Value

A list containing the data formatted for use with `pcd_cmdstan_model()`

See Also

Modelling wrappers for external fitting packages `fitdistdoublecens()`, `pcd_cmdstan_model()`

Examples

```
## Not run:
data <- data.frame(
  delay = c(1, 2, 3),
  delay_upper = c(2, 3, 4),
  n = c(10, 20, 15),
  pwindow = c(1, 1, 2),
  relative_obs_time = c(10, 10, 10)
)
stan_data <- pcd_as_stan_data(
  data,
  dist_id = 1,
  primary_id = 1,
  param_bounds = list(lower = c(0, 0), upper = c(10, 10)),
  primary_param_bounds = list(lower = numeric(0), upper = numeric(0)),
  priors = list(location = c(1, 1), scale = c(1, 1)),
  primary_priors = list(location = numeric(0), scale = numeric(0))
)

## End(Not run)
```

`pcd_cmdstan_model` *Create a CmdStanModel with primarycensored Stan functions*

Description

This function creates a CmdStanModel object using the Stan model and functions from primarycensored and optionally includes additional user-specified Stan files.

Usage

```
pcd_cmdstan_model(include_paths = primarycensored::pcd_stan_path(), ...)
```

Arguments

`include_paths`	Character vector of paths to include for Stan compilation. Defaults to the result of `pcd_stan_path()`.
`...`	Additional arguments passed to cmdstanr::cmdstan_model().

Details

The underlying Stan model (`pcens_model.stan`) supports various features:

- Multiple probability distributions for modeling delays
- Primary and secondary censoring
- Truncation
- Optional use of reduce_sum for improved performance (via within chain parallelism).
- Flexible prior specifications
- Optional computation of log-likelihood for model comparison

Value

A CmdStanModel object.

See Also

Modelling wrappers for external fitting packages `fitdistdoublecens()`, `pcd_as_stan_data()`

Examples

```
## Not run:
model <- pcd_cmdstan_model()
fit <- model$sample(data = stan_data)

## End(Not run)
```

`pcd_load_stan_functions` *Load Stan functions as a string*

Description

Load Stan functions as a string

Usage

```
pcd_load_stan_functions(
  functions = NULL,
  stan_path = primarycensored::pcd_stan_path(),
  wrap_in_block = FALSE,
  write_to_file = FALSE,
  output_file = "pcd_functions.stan"
)
```

Arguments

`functions`	Character vector of function names to load. Defaults to all functions.
`stan_path`	Character string, the path to the Stan code. Defaults to the path to the Stan code in the primarycensored package.
`wrap_in_block`	Logical, whether to wrap the functions in a `functions{}` block. Default is FALSE.
`write_to_file`	Logical, whether to write the output to a file. Default is FALSE.
`output_file`	Character string, the path to write the output file if write_to_file is TRUE. Defaults to "pcd_functions.stan".

Value

A character string containing the requested Stan functions

See Also

Tools for working with package Stan functions `pcd_stan_files()`, `pcd_stan_functions()`, `pcd_stan_path()`

pcd_stan_files *Get Stan files containing specified functions*

Description

This function retrieves Stan files from a specified directory, optionally filtering for files that contain specific functions.

Usage

```
pcd_stan_files(functions = NULL, stan_path = primarycensored::pcd_stan_path())
```

Arguments

functions	Character vector of function names to search for. If NULL, all Stan files are returned.
stan_path	Character string specifying the path to the directory containing Stan files. Defaults to the Stan path of the primarycensored package.

Value

A character vector of file paths to Stan files.

See Also

Tools for working with package Stan functions `pcd_load_stan_functions()`, `pcd_stan_functions()`, `pcd_stan_path()`

pcd_stan_functions *Get Stan function names from Stan files*

Description

This function reads all Stan files in the specified directory and extracts the names of all functions defined in those files.

Usage

```
pcd_stan_functions(stan_path = primarycensored::pcd_stan_path())
```

Arguments

stan_path	Character string specifying the path to the directory containing Stan files. Defaults to the Stan path of the primarycensored package.

Value

A character vector containing unique names of all functions found in the Stan files.

See Also

Tools for working with package Stan functions `pcd_load_stan_functions()`, `pcd_stan_files()`, `pcd_stan_path()`

---

pcd_stan_path *Get the path to the Stan code*

Description

Get the path to the Stan code

Usage

```
pcd_stan_path()
```

Value

A character string with the path to the Stan code

See Also

Tools for working with package Stan functions `pcd_load_stan_functions()`, `pcd_stan_files()`, `pcd_stan_functions()`

---

pcens_cdf *Compute primary event censored CDF*

Description

Compute primary event censored CDF

Usage

```
pcens_cdf(object, q, pwindow, use_numeric = FALSE)
```

Arguments

`object`	A `primarycensored` object as created by `new_pcens()`.
`q`	Vector of quantiles
`pwindow`	Primary event window
`use_numeric`	Logical, if TRUE forces use of numeric integration even for distributions with analytical solutions. This is primarily useful for testing purposes or for settings where the analytical solution breaks down.

Value

Vector of primary event censored CDFs

See Also

Low level primary event censored distribution objects and methods `new_pcens()`, `pcens_cdf.default()`, `pcens_cdf.pcens_pgamma_dunif()`, `pcens_cdf.pcens_plnorm_dunif()`, `pcens_cdf.pcens_pweibull_dunif()`

---

pcens_cdf.default *Default method for computing primary event censored CDF*

Description

This method serves as a fallback for combinations of delay and primary event distributions that don't have specific implementations. It uses the numeric integration method.

Usage

```
## Default S3 method:
pcens_cdf(object, q, pwindow, use_numeric = FALSE)
```

Arguments

`object`	A `primarycensored` object as created by `new_pcens()`.
`q`	Vector of quantiles
`pwindow`	Primary event window
`use_numeric`	Logical, if TRUE forces use of numeric integration even for distributions with analytical solutions. This is primarily useful for testing purposes or for settings where the analytical solution breaks down.

Details

This method implements the numerical integration approach for computing the primary event censored CDF. It uses the same mathematical formulation as described in the details section of `pprimarycensored()`, but applies numerical integration instead of analytical solutions.

See Also

`pprimarycensored()` for the mathematical details of the primary event censored CDF computation.

Low level primary event censored distribution objects and methods `new_pcens()`, `pcens_cdf()`, `pcens_cdf.pcens_pgamma_dunif()`, `pcens_cdf.pcens_plnorm_dunif()`, `pcens_cdf.pcens_pweibull_dunif()`

pcens_cdf.pcens_pgamma_dunif

Method for Gamma delay with uniform primary

Description

Method for Gamma delay with uniform primary

Usage

```
## S3 method for class 'pcens_pgamma_dunif'
pcens_cdf(object, q, pwindow, use_numeric = FALSE)
```

Arguments

`object`	A `primarycensored` object as created by `new_pcens()`.
`q`	Vector of quantiles
`pwindow`	Primary event window
`use_numeric`	Logical, if TRUE forces use of numeric integration even for distributions with analytical solutions. This is primarily useful for testing purposes or for settings where the analytical solution breaks down.

See Also

Low level primary event censored distribution objects and methods `new_pcens()`, `pcens_cdf()`, `pcens_cdf.default()`, `pcens_cdf.pcens_plnorm_dunif()`, `pcens_cdf.pcens_pweibull_dunif()`

pcens_cdf.pcens_plnorm_dunif

Method for Log-Normal delay with uniform primary

Description

Method for Log-Normal delay with uniform primary

Usage

```
## S3 method for class 'pcens_plnorm_dunif'
pcens_cdf(object, q, pwindow, use_numeric = FALSE)
```

Arguments

`object`	A `primarycensored` object as created by `new_pcens()`.
`q`	Vector of quantiles
`pwindow`	Primary event window
`use_numeric`	Logical, if TRUE forces use of numeric integration even for distributions with analytical solutions. This is primarily useful for testing purposes or for settings where the analytical solution breaks down.

See Also

Low level primary event censored distribution objects and methods `new_pcens()`, `pcens_cdf()`, `pcens_cdf.default()`, `pcens_cdf.pcens_pgamma_dunif()`, `pcens_cdf.pcens_pweibull_dunif()`

---

pcens_cdf.pcens_pweibull_dunif

Method for Weibull delay with uniform primary

Description

Method for Weibull delay with uniform primary

Usage

```
## S3 method for class 'pcens_pweibull_dunif'
pcens_cdf(object, q, pwindow, use_numeric = FALSE)
```

Arguments

`object`	A `primarycensored` object as created by `new_pcens()`.
`q`	Vector of quantiles
`pwindow`	Primary event window
`use_numeric`	Logical, if TRUE forces use of numeric integration even for distributions with analytical solutions. This is primarily useful for testing purposes or for settings where the analytical solution breaks down.

See Also

Low level primary event censored distribution objects and methods `new_pcens()`, `pcens_cdf()`, `pcens_cdf.default()`, `pcens_cdf.pcens_pgamma_dunif()`, `pcens_cdf.pcens_plnorm_dunif()`

`pprimarycensored` *Compute the primary event censored CDF for delays*

Description

This function computes the primary event censored cumulative distribution function (CDF) for a given set of quantiles. It adjusts the CDF of the primary event distribution by accounting for the delay distribution and potential truncation at a maximum delay (D). The function allows for custom primary event distributions and delay distributions.

Usage

```
pprimarycensored(
  q,
  pdist,
  pwindow = 1,
  D = Inf,
  dprimary = stats::dunif,
  dprimary_args = list(),
  pdist_name = NULL,
  dprimary_name = NULL,
  ...
)

ppcens(
  q,
  pdist,
  pwindow = 1,
  D = Inf,
  dprimary = stats::dunif,
  dprimary_args = list(),
  pdist_name = NULL,
  dprimary_name = NULL,
  ...
)
```

Arguments

`q`	Vector of quantiles
`pdist`	Distribution function (CDF)
`pwindow`	Primary event window
`D`	Maximum delay (truncation point). If finite, the distribution is truncated at D. If set to Inf, no truncation is applied. Defaults to Inf.
`dprimary`	Function to generate the probability density function (PDF) of primary event times. This function should take a value `x` and a `pwindow` parameter, and return a probability density. It should be normalized to

integrate to 1 over [0, pwindow]. Defaults to a uniform distribution over [0, pwindow]. Users can provide custom functions or use helper functions like `dexpgrowth` for an exponential growth distribution. See `primary_dists.R` for examples.

`dprimary_args` List of additional arguments to be passed to dprimary. For example, when using `dexpgrowth`, you would pass `list(min = 0, max = pwindow, r = 0.2)` to set the minimum, maximum, and rate parameters

`pdist_name` A string specifying the name of the delay distribution function. If NULL, the function name is extracted using `.extract_function_name()`. Used to determine if a analytical solution exists for the primary censored distribution. Must be set if `pdist` is passed a pre-assigned variable rather than a function name.

`dprimary_name` A string specifying the name of the primary event distribution function. If NULL, the function name is extracted using `.extract_function_name()`. Used to determine if a analytical solution exists for the primary censored distribution. Must be set if `dprimary` is passed a pre-assigned variable rather than a function name.

... Additional arguments to be passed to pdist

Details

The primary event censored CDF is computed by integrating the product of the delay distribution function (CDF) and the primary event distribution function (PDF) over the primary event window. The integration is adjusted for truncation if a finite maximum delay (D) is specified.

The primary event censored CDF, $F_{\text{cens}}(q)$, is given by:

$$F_{\text{cens}}(q) = \int_0^{pwindow} F(q-p) \cdot f_{\text{primary}}(p)\, dp$$

where F is the CDF of the delay distribution, f_{primary} is the PDF of the primary event times, and *pwindow* is the primary event window.

If the maximum delay D is finite, the CDF is normalized by dividing by $F_{\text{cens}}(D)$:

$$F_{\text{cens,norm}}(q) = \frac{F_{\text{cens}}(q)}{F_{\text{cens}}(D)}$$

where $F_{\text{cens,norm}}(q)$ is the normalized CDF.

This function creates a `primarycensored` object using `new_pcens()` and then computes the primary event censored CDF using `pcens_cdf()`. This abstraction allows for automatic use of analytical solutions when available, while seamlessly falling back to numerical integration when necessary.

Note: For analytical detection to work correctly, `pdist` and `dprimary` must be directly passed as distribution functions, not via assignment or `pdist_name` and `dprimary_name` must be used to override the default extraction of the function name.

Value

Vector of primary event censored CDFs, normalized by D if finite (truncation adjustment)

See Also

`new_pcens()` and `pcens_cdf()`

Primary event censored distribution functions `dprimarycensored()`, `rprimarycensored()`

Examples

```
# Example: Lognormal distribution with uniform primary events
pprimarycensored(c(0.1, 0.5, 1), plnorm, meanlog = 0, sdlog = 1)

# Example: Lognormal distribution with exponential growth primary events
pprimarycensored(
  c(0.1, 0.5, 1), plnorm,
  dprimary = dexpgrowth,
  dprimary_args = list(r = 0.2), meanlog = 0, sdlog = 1
)
```

---

rprimarycensored *Generate random samples from a primary event censored distribution*

Description

This function generates random samples from a primary event censored distribution. It adjusts the distribution by accounting for the primary event distribution and potential truncation at a maximum delay (D). The function allows for custom primary event distributions and delay distributions.

Usage

```
rprimarycensored(
  n,
  rdist,
  pwindow = 1,
  swindow = 1,
  D = Inf,
  rprimary = stats::runif,
  rprimary_args = list(),
  oversampling_factor = 1.2,
  ...
)

rpcens(
  n,
  rdist,
  pwindow = 1,
  swindow = 1,
  D = Inf,
```

```
  rprimary = stats::runif,
  rprimary_args = list(),
  oversampling_factor = 1.2,
  ...
)
```

Arguments

`n`	Number of random samples to generate.
`rdist`	Function to generate random samples from the delay distribution for example `stats::rlnorm()` for lognormal distribution.
`pwindow`	Primary event window
`swindow`	Integer specifying the window size for rounding the delay (default is 1). If `swindow = 0` then no rounding is applied.
`D`	Maximum delay (truncation point). If finite, the distribution is truncated at D. If set to Inf, no truncation is applied. Defaults to Inf.
`rprimary`	Function to generate random samples from the primary distribution (default is `stats::runif()`).
`rprimary_args`	List of additional arguments to be passed to rprimary.
`oversampling_factor`	Factor by which to oversample the number of samples to account for truncation (default is 1.2).
`...`	Additional arguments to be passed to the distribution function.

Details

The mathematical formulation for generating random samples from a primary event censored distribution is as follows:

1. Generate primary event times (p) from the specified primary event distribution (f_p) within the primary event window (pwindow):

$$p \sim f_p(x), \quad 0 \leq x \leq pwindow$$

2. Generate delays (d) from the specified delay distribution (f_d) with parameters theta:

$$d \sim f_d(x; \theta)$$

3. Calculate the total delays (t) by adding the primary event times and the delays:

$$t = p + d$$

4. Apply truncation to ensure that the delays are within the specified range [0, D]:

$$t_{truncated} = \{t \mid 0 \leq t < D\}$$

5. Round the truncated delays to the nearest secondary event window (swindow):

$$t_{valid} = \lfloor \frac{t_{truncated}}{swindow} \rfloor \times swindow$$

The function oversamples to account for potential truncation and generates additional samples if needed to reach the desired number of valid samples.

Value

Vector of random samples from the primary event censored distribution censored by the secondary event window.

See Also

Primary event censored distribution functions `dprimarycensored()`, `pprimarycensored()`

Examples

```
# Example: Lognormal distribution with uniform primary events
rprimarycensored(10, rlnorm, meanlog = 0, sdlog = 1)

# Example: Lognormal distribution with exponential growth primary events
rprimarycensored(
  10, rlnorm,
  rprimary = rexpgrowth, rprimary_args = list(r = 0.2),
  meanlog = 0, sdlog = 1
)
```

Index