

Package ‘primarycensored’

June 4, 2026

Title Primary Event Censored Distributions

Version 1.5.0

Description Provides 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 (Charniga et al. (2024) <[doi:10.48550/arXiv.2405.08841](https://doi.org/10.48550/arXiv.2405.08841)>). It also provides support for arbitrary delay distributions, a range of common primary distributions, and allows for truncation and secondary event censoring to be accounted for (Park et al. (2024) <[doi:10.1101/2024.01.12.24301247](https://doi.org/10.1101/2024.01.12.24301247)>). A subset of common distributions also have analytical solutions implemented, allowing for faster computation. In addition, it provides multiple methods for fitting primary event censored distributions to data via optional dependencies.

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)

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

RoxygenNote 7.3.3

VignetteBuilder knitr

NeedsCompilation no

Author Sam Abbott [aut, cre, cph] (ORCID:

<<https://orcid.org/0000-0001-8057-8037>>),

Sam Brand [aut] (ORCID: <<https://orcid.org/0000-0003-0645-5367>>),

Adam Howes [ctb] (ORCID: <<https://orcid.org/0000-0003-2386-4031>>),

James Mba Azam [aut] (ORCID: <<https://orcid.org/0000-0001-5782-7330>>),

Carl Pearson [aut] (ORCID: <<https://orcid.org/0000-0003-0701-7860>>),

Sebastian Funk [aut] (ORCID: <<https://orcid.org/0000-0002-2842-3406>>),

Kelly Charniga [aut] (ORCID: <<https://orcid.org/0000-0002-7648-7041>>)

Maintainer Sam Abbott <contact@samabbott.co.uk>

Repository CRAN

Date/Publication 2026-06-04 10:20:02 UTC

Contents

add_name_attribute	3
check_dprimary	4
check_pdist	5
check_truncation	5
dprimarycensored	6
expgrowth	8
fitdistdoublecens	10
new_pcens	13
pcd_as_stan_data	14
pcd_cmdstan_model	16
pcd_distributions	17
pcd_dist_name	17
pcd_load_stan_functions	18
pcd_primary_distributions	19
pcd_stan_dist_id	20
pcd_stan_files	21
pcd_stan_functions	21
pcd_stan_function_deps	22
pcd_stan_path	23
pcens_cdf	23
pcens_cdf.default	24
pcens_cdf.pcens_pgamma_dunif	25
pcens_cdf.pcens_plnorm_dunif	26
pcens_cdf.pcens_pweibull_dunif	26
pcens_quantile	27
pcens_quantile.default	28
pprimarycensored	29
qprimarycensored	32
rprimarycensored	34

Index

37

add_name_attribute *Helper method for custom distributions*

Description

[pprimarycensored\(\)](#) and related functions can identify which distributions are provided via the `pdist` and `dprimary` arguments when those are base R functions (e.g. `pnif`, `dexp`) via the `name` attribute.

Usage

```
add_name_attribute(func, name)
```

Arguments

<code>func</code>	Function, for example the <code>p</code> - or <code>d</code> - form of a distribution function.
<code>name</code>	Character string, starting with "p" or "d" indicating the underlying distribution.

Details

If you need to use a non-base R implementation, but know the distribution name, you can use this helper function to set it in a way that will be detected by [pprimarycensored\(\)](#) and related functions.

This is useful as it enables the automatic use of analytical solutions for distributions where they exist. You can check which analytical solutions are available using `methods(pcens_cdf)` and check distribution names using [pcd_dist_name\(\)](#).

Value

Function, with a "name" attribute added

See Also

Utility functions for working with distributions [pcd_dist_name\(\)](#), [pcd_distributions](#), [pcd_primary_distributions](#)

Examples

```
dist <- add_name_attribute(pnorm, "hello")
attr(dist, "name")
```

check_dprimary	<i>Check if a function is a valid bounded probability density function (PDF)</i>
----------------	--

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 <code>pwindow</code> 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 <code>dexpgrowth</code> for an exponential growth distribution. See <code>pcd_primary_distributions()</code> for examples. The package can identify base R distributions for potential analytical solutions. For non-base R functions, users can apply <code>add_name_attribute()</code> to yield properly tagged functions if they wish to leverage analytical solutions.
pwindow	Primary event window
dprimary_args	List of additional arguments to be passed to <code>dprimary</code> . For example, when using <code>dexpgrowth</code> , you would pass <code>list(min = 0, max = pwindow, r = 0.2)</code> 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	<i>Check if a function is a valid cumulative distribution function (CDF)</i>
-------------	--

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 = Inf, ...)
```

Arguments

pdist	Distribution function (CDF). The package can identify base R distributions for potential analytical solutions. For non-base R functions, users can apply add_name_attribute() to yield properly tagged functions if they wish to leverage the analytical solutions.
D	Maximum delay (upper truncation point). If finite, the distribution is truncated at D. If set to Inf, no upper 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	<i>Check if truncation time is appropriate relative to the maximum delay</i>
------------------	--

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) and minimum delay (L). The function allows for custom primary event distributions and delay distributions.

Usage

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

dpcens(
  x,
  pdist,
  pwindow = 1,
```

```

    swindow = 1,
    L = -Inf,
    D = Inf,
    dprimary = stats::dunif,
    dprimary_args = list(),
    log = FALSE,
    ...
)

```

Arguments

x	Vector of quantiles
pdist	Distribution function (CDF). The package can identify base R distributions for potential analytical solutions. For non-base R functions, users can apply <code>add_name_attribute()</code> to yield properly tagged functions if they wish to leverage the analytical solutions.
pwindow	Primary event window
swindow	Secondary event window (default: 1)
L	Minimum delay (lower truncation point). Defaults to <code>-Inf</code> , meaning no left truncation. For any finite value of L the distribution is left-truncated at L.
D	Maximum delay (upper truncation point). If finite, the distribution is truncated at D. If set to <code>Inf</code> , no upper truncation is applied. Defaults to <code>Inf</code> .
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 <code>dexpgrowth</code> for an exponential growth distribution. See <code>pcd_primary_distributions()</code> for examples. The package can identify base R distributions for potential analytical solutions. For non-base R functions, users can apply <code>add_name_attribute()</code> to yield properly tagged functions if they wish to leverage analytical solutions.
dprimary_args	List of additional arguments to be passed to dprimary. For example, when using <code>dexpgrowth</code> , you would pass <code>list(min = 0, max = pwindow, r = 0.2)</code> to set the minimum, maximum, and rate parameters
log	Logical; if TRUE, probabilities p are given as <code>log(p)</code>
...	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 $d + \text{swindow}$) using `pprimarycensored()`. It then creates a lookup table for these CDFs to efficiently calculate the PMF for each input value. For delays less than L , the function returns 0.

The PMF is normalised to ensure it sums to 1 over the range $[L, D]$. This normalization uses:

$$f_{\text{cens, norm}}(d) = \frac{f_{\text{cens}}(d)}{F_{\text{cens}}(D) - F_{\text{cens}}(L)}$$

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

Value

Vector of primary event censored PMFs, normalized over $[L, D]$ if truncation is applied

See Also

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

# Example: Left-truncated distribution (e.g., for generation intervals)
dprimarycensored(1:9, pweibull, L = 1, D = 10, 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)

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

Arguments

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

Details

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

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

The cumulative distribution function (CDF) is:

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

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 :

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

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 $\text{Uniform}(0, 1)$ distribution. Conversely, if U is a $\text{Uniform}(0, 1)$ 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 $\text{Uniform}(0, 1)$, 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:

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

When r is very close to 0 ($|r| < 1e - 10$), the distribution approximates a uniform distribution on $[\text{min}, \text{max}]$, 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 rexprowth generates random deviates.

The length of the result is determined by n for rexprowth, 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 <- rexprowth(100, r = 0.2)
```

fitdistdoublecens	<i>Fit a distribution to doubly censored data</i>
-------------------	---

Description

This function wraps the custom approach for fitting distributions to doubly censored data using fitdistrplus and primarycensored. It handles primary censoring (when the primary event time is not known exactly), secondary censoring (when the secondary event time is interval-censored), and truncation (when events are only observed within a delay range [L, D]).

Usage

```
fitdistdoublecens(
  censdata,
  distr,
  left = "left",
  right = "right",
  pwindow = "pwindow",
  L = "L",
  D = "D",
  dprimary = stats::dunif,
  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 <code>fitdistrplus::fitdistcens()</code> NA is not supported for either the upper or lower bounds.
----------	--

distr	A character string naming the distribution to be fitted. This should be the base name of a distribution with corresponding d (density) and p (cumulative distribution) functions available. For example, use "gamma" (which will use dgamma and pgamma), "lnorm" (for dlnorm and plnorm), "weibull", "norm", etc. Custom distributions can also be used as long as the corresponding d<distr>() and p<distr>() functions are defined and loaded.
left	Column name for lower bound of observed values (default: "left").
right	Column name for upper bound of observed values (default: "right").
pwindow	Column name for primary window (default: "pwindow").
L	Column name for minimum delay (lower truncation point). For any finite L the distribution is left-truncated at L; use L = -Inf for no left truncation. This is useful for modelling generation intervals where day 0 is excluded, particularly when used in renewal models. (default: "L"). If the column is not present in censdata, L = -Inf is assumed.
D	Column name for maximum delay (upper truncation point). If finite, the distribution is truncated at D. If set to Inf, no upper truncation is applied. (default: "D").
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 <code>pcd_primary_distributions()</code> for examples. The package can identify base R distributions for potential analytical solutions. For non-base R functions, users can apply <code>add_name_attribute()</code> to yield properly tagged functions if they wish to leverage analytical solutions.
dprimary_args	List of additional arguments to be passed to dprimary. For example, when using dexpgrowth, you would pass <code>list(min = 0, max = pwindow, r = 0.2)</code> 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 <code>fitdistrplus::fitdist()</code> .

Details

How distribution functions are resolved:

The `distr` parameter specifies the base name of the distribution. The function automatically looks up the corresponding density (d) and cumulative distribution (p) functions by prepending these prefixes to the distribution name. For example:

- `distr = "gamma"` uses `dgamma()` and `pgamma()`
- `distr = "lnorm"` uses `dlnorm()` and `plnorm()`
- `distr = "weibull"` uses `dweibull()` and `pweibull()`

Any distribution available in base R or loaded packages can be used, as long as the corresponding `d<distr>` and `p<distr>` functions exist and follow standard R distribution function conventions (first argument is x for density, q for CDF).

What this function does internally:

This function creates custom density and CDF functions that account for primary censoring, secondary censoring, and truncation using `dprimarycensored()` and `pprimarycensored()`. These custom functions are then passed to `fitdistrplus::fitdist()` for maximum likelihood estimation.

The function handles varying observation windows across observations, making it suitable for real-world data where truncation times or censoring windows may differ between observations.

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,
  pwindow = rep(pwindow, n),
  D = rep(D, n)
)

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

summary(fit_norm)
```

new_pcens	<i>S3 class for primary event censored distribution computation</i>
-----------	---

Description

S3 class for primary event censored distribution computation

Usage

```
new_pcens(pdist, dprimary, dprimary_args, ...)
```

Arguments

pdist	Distribution function (CDF). The package can identify base R distributions for potential analytical solutions. For non-base R functions, users can apply add_name_attribute() to yield properly tagged functions if they wish to leverage the analytical solutions.
dprimary	Function to generate the probability density function (PDF) of primary event times. This function should take a value <code>x</code> and a <code>pwindow</code> parameter, and return a probability density. It should be normalized to integrate to 1 over $[0, \text{pwindow}]$. Defaults to a uniform distribution over $[0, \text{pwindow}]$. Users can provide custom functions or use helper functions like <code>dexpgrowth</code> for an exponential growth distribution. See pcd_primary_distributions() for examples. The package can identify base R distributions for potential analytical solutions. For non-base R functions, users can apply add_name_attribute() to yield properly tagged functions if they wish to leverage analytical solutions.
dprimary_args	List of additional arguments to be passed to <code>dprimary</code> . For example, when using <code>dexpgrowth</code> , you would pass <code>list(min = 0, max = pwindow, r = 0.2)</code> to set the minimum, maximum, and rate parameters
...	Additional arguments to be passed to <code>pdist</code>

Value

An object of class `pcens_{pdist_name}_{dprimary_name}`. This contains the primary event distribution, the delay distribution, the delay distribution arguments, and any additional arguments. It can be used with the `pcens_cdf()` function to compute the primary event censored cdf.

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\(\)](#), [pcens_quantile\(\)](#), [pcens_quantile.default\(\)](#)

Examples

```
new_pcens(
  pdist = pgamma, dprimary = dunif, dprimary_args = list(min = 0, max = 1),
  shape = 1, scale = 1
)
```

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",
  start_relative_obs_time = "start_relative_obs_time",
  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")
start_relative_obs_time	Column name for start of relative observation time, used as the lower truncation point L. Values may be any finite real number (including negatives) or -Inf to indicate no lower truncation. If the column is not present in data, L = -Inf is assumed for all observations. (default: "start_relative_obs_time")

relative_obs_time	Column name for relative observation time, used as the upper truncation point D. Values may be any finite real number (including negatives, paired with a smaller start_relative_obs_time) or +Inf to indicate no upper truncation. (default: "relative_obs_time")
dist_id	Integer identifying the delay distribution: You can use <code>pcd_stan_dist_id()</code> to get the dist ID for a distribution or look at the <code>pcd_distributions</code> data set.
primary_id	Integer identifying the primary distribution: You can use <code>pcd_stan_dist_id()</code> to get the primary dist ID for a distribution (make sure to select the "primary" type) or look at the <code>pcd_primary_distributions</code> data set.
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

```
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))
)

```

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

```
if (!is.null(cmdstanr::cmdstan_version(error_on_NA = FALSE))) {
  model <- pcd_cmdstan_model(compile = FALSE)
  model
}
```

pcd_distributions *Supported delay distributions*

Description

A dataset containing information about the supported delay distributions in primarycensored. Includes both distributions with base R implementations and those only available in Stan. Distributions beyond these are not supported in the stan code but any user functions can be used in the R code.

Usage

```
pcd_distributions
```

Format

A data.frame with 17 rows and 4 columns:

name Distribution name

pdist R distribution function name (e.g. plnorm), NA if there is no base R implementation

aliases Alternative names/identifiers

stan_id Stan distribution ID used in the stan code

See Also

Utility functions for working with distributions [add_name_attribute\(\)](#), [pcd_dist_name\(\)](#), [pcd_primary_distributions](#)

pcd_dist_name *Get distribution function cdf or pdf name*

Description

Get distribution function cdf or pdf name

Usage

```
pcd_dist_name(name, type = c("delay", "primary"))
```

Arguments

name	String. Distribution name or alias
type	String. "delay" or "primary" corresponding to the type of distribution to use as the look up. If delay then pcd_distributions() is used, if primary then pcd_primary_distributions() is used.

Value

String distribution function name or NA if no base R implementation

See Also

Utility functions for working with distributions [add_name_attribute\(\)](#), [pcd_distributions](#), [pcd_primary_distributions](#)

Examples

```
pcd_dist_name("lnorm")
pcd_dist_name("lognormal")
pcd_dist_name("gamma")
pcd_dist_name("weibull")
pcd_dist_name("exp")
pcd_dist_name("unif", type = "primary")
pcd_dist_name("expgrowth", type = "primary")
```

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",
  dependencies = FALSE
)
```

Arguments

<code>functions</code>	Character vector of function names to load. Defaults to all functions.
<code>stan_path</code>	Character string, the path to the Stan code. Defaults to the path to the Stan code in the primarycensored package.
<code>wrap_in_block</code>	Logical, whether to wrap the functions in a <code>functions{}</code> block. Default is FALSE.
<code>write_to_file</code>	Logical, whether to write the output to a file. Default is FALSE.
<code>output_file</code>	Character string, the path to write the output file if <code>write_to_file</code> is TRUE. Defaults to "pcd_functions.stan".
<code>dependencies</code>	Logical, whether to include all functions that the requested functions depend on. When TRUE, recursively finds and includes all dependencies in the correct order (dependencies before the functions that use them). Default is FALSE.

Value

A character string containing the requested Stan functions

See Also

Tools for working with package Stan functions [pcd_stan_dist_id\(\)](#), [pcd_stan_files\(\)](#), [pcd_stan_function_deps\(\)](#), [pcd_stan_functions\(\)](#), [pcd_stan_path\(\)](#)

`pcd_primary_distributions`

Supported primary event distributions

Description

A dataset containing information about the supported primary event distributions in primarycensored. Distributions beyond these are not supported in the stan code but any user functions can be used in the R code.

Usage

```
pcd_primary_distributions
```

Format

A data.frame with 2 rows and 4 columns:

name Distribution name

dprimary R density function name

aliases Alternative names/identifiers

stan_id Stan distribution ID used in the stan code

See Also

Utility functions for working with distributions [add_name_attribute\(\)](#), [pcd_dist_name\(\)](#), [pcd_distributions](#)

pcd_stan_dist_id *Get distribution stan ID by name*

Description

Get distribution stan ID by name

Usage

```
pcd_stan_dist_id(name, type = c("delay", "primary"))
```

Arguments

name	String. Distribution name or alias
type	String. "delay" or "primary" corresponding to the type of distribution to use as the look up. If delay then pcd_distributions() is used, if primary then pcd_primary_distributions() is used.

Value

Numeric distribution ID

See Also

Tools for working with package Stan functions [pcd_load_stan_functions\(\)](#), [pcd_stan_files\(\)](#), [pcd_stan_function_deps\(\)](#), [pcd_stan_functions\(\)](#), [pcd_stan_path\(\)](#)

Examples

```
pcd_stan_dist_id("lnorm")
pcd_stan_dist_id("lognormal")
pcd_stan_dist_id("gamma")
pcd_stan_dist_id("weibull")
pcd_stan_dist_id("exp")
pcd_stan_dist_id("unif", type = "primary")
```

pcd_stan_files	<i>Get Stan files containing specified functions</i>
----------------	--

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_dist_id\(\)](#), [pcd_stan_function_deps\(\)](#), [pcd_stan_functions\(\)](#), [pcd_stan_path\(\)](#)

pcd_stan_functions	<i>Get Stan function names from Stan files</i>
--------------------	--

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_dist_id\(\)](#), [pcd_stan_files\(\)](#), [pcd_stan_function_deps\(\)](#), [pcd_stan_path\(\)](#)

pcd_stan_function_deps

Get dependencies for a Stan function

Description

Returns all Stan functions that the specified function depends on, in topological order (dependencies before the functions that use them).

Usage

```
pcd_stan_function_deps(
  function_name,
  stan_path = primarycensored::pcd_stan_path()
)
```

Arguments

`function_name` Character string, the name of the Stan function.

`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 function names that the specified function depends on, ordered so that dependencies come before functions that use them. The requested function itself is included as the last element.

See Also

Tools for working with package Stan functions [pcd_load_stan_functions\(\)](#), [pcd_stan_dist_id\(\)](#), [pcd_stan_files\(\)](#), [pcd_stan_functions\(\)](#), [pcd_stan_path\(\)](#)

Examples

```
# See what primarycensored_lpmf depends on
pcd_stan_function_deps("primarycensored_lpmf")

# A function with no dependencies
pcd_stan_function_deps("expgrowth_pdf")
```

pcd_stan_path	<i>Get the path to the Stan code</i>
---------------	--------------------------------------

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_dist_id\(\)](#), [pcd_stan_files\(\)](#), [pcd_stan_function_deps\(\)](#), [pcd_stan_functions\(\)](#)

pcens_cdf	<i>Compute primary event censored CDF</i>
-----------	---

Description

This function dispatches to either analytical solutions (if available) or numerical integration via the default method. To see which combinations have analytical solutions implemented, use `methods(pcens_cdf)`. For example, `pcens_cdf$gamma_unif` indicates an analytical solution exists for gamma delay with uniform primary event distributions.

Usage

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

Arguments

<code>object</code>	A primarycensored object as created by new_pcens() .
<code>q</code>	Vector of quantiles
<code>pwindow</code>	Primary event window
<code>use_numeric</code>	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 computed 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_quantile\(\)](#), [pcens_quantile.default\(\)](#)

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 a 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.

Value

Vector of computed primary event censored CDFs

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_quantile\(\)](#), [pcens_quantile.default\(\)](#)

Examples

```
# Create a primarycensored object with gamma delay and uniform primary
pcens_obj <- new_pcens(
  pdist = pgamma,
  dprimary = dunif,
  dprimary_args = list(min = 0, max = 1),
  shape = 3,
  scale = 2
)

# Compute CDF for a single value
pcens_cdf(pcens_obj, q = 9, pwindow = 1)

# Compute CDF for multiple values
pcens_cdf(pcens_obj, q = c(4, 6, 8), pwindow = 1)
```

```
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.

Value

Vector of computed primary event censored CDFs

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_quantile\(\)](#), [pcens_quantile.default\(\)](#)

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.

Value

Vector of computed primary event censored CDFs

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_quantile\(\)](#), [pcens_quantile.default\(\)](#)

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.

Value

Vector of computed primary event censored CDFs

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\(\)](#), [pcens_quantile\(\)](#), [pcens_quantile.default\(\)](#)

pcens_quantile	<i>Compute primary event censored quantiles</i>
----------------	---

Description

This function inverts the primary event censored CDF to compute quantiles. It uses numerical optimisation via `optim` to find the value `q` such that `pcens_cdf()` is close to the specified probability. Currently, only the default numerical inversion method is implemented. Future analytical solutions may be added.

Usage

```
pcens_quantile(object, p, pwindow, L = -Inf, D = Inf, use_numeric = FALSE, ...)
```

Arguments

object	A primarycensored object as created by new_pcens() .
p	A vector of probabilities at which to compute the quantiles.
pwindow	Primary event window
L	Minimum delay (lower truncation point). Defaults to <code>-Inf</code> , meaning no left truncation. For any finite value of <code>L</code> the distribution is left-truncated at <code>L</code> .
D	Maximum delay (upper truncation point). If finite, the distribution is truncated at <code>D</code> . If set to <code>Inf</code> , no upper truncation is applied. Defaults to <code>Inf</code> .
use_numeric	Logical; if TRUE forces the use of numeric inversion even if an analytical solution is available (not yet implemented).
...	Additional arguments to be passed to <code>pdist</code>

Value

Vector of primary event censored quantiles.

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()`, `pcens_cdf.pcens_pweibull_dunif()`, `pcens_quantile.default()`

pcens_quantile.default

Default method for computing primary event censored quantiles

Description

This method inverts the primary event censored CDF by root-finding via `stats::uniroot()` with `extendInt = "upX"`. The censored CDF is monotone in q , so `stats::uniroot()` extends its starting bracket outward as needed and handles infinite L or D without special casing.

Usage

```
## Default S3 method:
pcens_quantile(
  object,
  p,
  pwindow,
  L = -Inf,
  D = Inf,
  use_numeric = FALSE,
  init = 5,
  tol = 1e-08,
  max_iter = 10000,
  ...
)
```

Arguments

<code>object</code>	A primarycensored object as created by <code>new_pcens()</code> .
<code>p</code>	A vector of probabilities at which to compute the quantiles.
<code>pwindow</code>	Primary event window
<code>L</code>	Minimum delay (lower truncation point). Defaults to <code>-Inf</code> , meaning no left truncation. For any finite value of L the distribution is left-truncated at L .
<code>D</code>	Maximum delay (upper truncation point). If finite, the distribution is truncated at D . If set to <code>Inf</code> , no upper truncation is applied. Defaults to <code>Inf</code> .
<code>use_numeric</code>	Logical; if <code>TRUE</code> forces the use of numeric inversion even if an analytical solution is available (not yet implemented).

<code>init</code>	Half-width of the initial search interval used when one or both truncation bounds are infinite. The starting interval is taken as $[-init, init]$ (or $[L, L + 2 * init]$ / $[D - 2 * init, D]$ when only one bound is finite) and then extended outward by <code>stats::uniroot()</code> as needed. Defaults to 5, which brackets the bulk of most commonly used delay distributions.
<code>tol</code>	Numeric tolerance passed to <code>stats::uniroot()</code> .
<code>max_iter</code>	Maximum number of <code>stats::uniroot()</code> iterations.
<code>...</code>	Additional arguments passed to underlying functions.

Value

A numeric vector containing the computed primary event censored quantiles.

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()`, `pcens_cdf.pcens_pweibull_dunif()`, `pcens_quantile()`

Examples

```
# Create a primarycensored object with gamma delay and uniform primary
pcens_obj <- new_pcens(
  pdist = pgamma,
  dprimary = dunif,
  dprimary_args = list(min = 0, max = 1),
  shape = 3,
  scale = 2
)

# Compute quantile for a single probability
pcens_quantile(pcens_obj, p = 0.8, pwindow = 1)

# Compute quantiles for multiple probabilities
pcens_quantile(pcens_obj, p = c(0.25, 0.5, 0.75), pwindow = 1)

# Compute quantiles for multiple probabilities with truncation
pcens_quantile(pcens_obj, p = c(0.25, 0.5, 0.75), pwindow = 1, D = 10)

# Compute quantiles with left truncation
pcens_quantile(pcens_obj, p = c(0.25, 0.5, 0.75), pwindow = 1, L = 1, D = 10)
```

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) and minimum delay (L). The function allows for custom primary event distributions and delay distributions.

Usage

```
pprimarycensored(
  q,
  pdist,
  pwindow = 1,
  L = -Inf,
  D = Inf,
  dprimary = stats::dunif,
  dprimary_args = list(),
  ...
)

ppcens(
  q,
  pdist,
  pwindow = 1,
  L = -Inf,
  D = Inf,
  dprimary = stats::dunif,
  dprimary_args = list(),
  ...
)
```

Arguments

<code>q</code>	Vector of quantiles
<code>pdist</code>	Distribution function (CDF). The package can identify base R distributions for potential analytical solutions. For non-base R functions, users can apply add_name_attribute() to yield properly tagged functions if they wish to leverage the analytical solutions.
<code>pwindow</code>	Primary event window
<code>L</code>	Minimum delay (lower truncation point). Defaults to <code>-Inf</code> , meaning no left truncation. For any finite value of <code>L</code> the distribution is left-truncated at <code>L</code> .
<code>D</code>	Maximum delay (upper truncation point). If finite, the distribution is truncated at <code>D</code> . If set to <code>Inf</code> , no upper truncation is applied. Defaults to <code>Inf</code> .
<code>dprimary</code>	Function to generate the probability density function (PDF) of primary event times. This function should take a value <code>x</code> and a <code>pwindow</code> parameter, and return a probability density. It should be normalized to integrate to 1 over <code>[0, pwindow]</code> . Defaults to a uniform distribution over <code>[0, pwindow]</code> . Users can provide custom functions or use helper functions like <code>dexpgrowth</code> for an exponential growth

distribution. See `pcd_primary_distributions()` for examples. The package can identify base R distributions for potential analytical solutions. For non-base R functions, users can apply `add_name_attribute()` to yield properly tagged functions if they wish to leverage analytical solutions.

`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

... 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 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 truncation is applied (finite D or finite L), the CDF is normalized:

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

where $F_{\text{cens, norm}}(q)$ is the normalized CDF. For values $q \leq L$, the function returns 0; for values $q \geq D$, it returns 1.

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.

See `methods(pcens_cdf)` for which combinations have analytical solutions implemented.

Value

Vector of primary event censored CDFs, normalized over $[L, D]$ if truncation is applied

See Also

`new_pcens()` and `pcens_cdf()`

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

# Example: Left-truncated distribution (e.g., for generation intervals)
pprimarycensored(
  c(1, 2, 3), plnorm,
  L = 1, D = 10,
  meanlog = 0, sdlog = 1
)
```

qprimarycensored	<i>Compute quantiles corresponding to target probabilities for primary event censored delays</i>
------------------	--

Description

This function computes the quantiles (delay values) that correspond to specified probabilities in the primary event censored distribution. For a given probability p , it computes the delay value q such that the cumulative probability up to q equals p in the primary event censored distribution. The distribution accounts for both the delay distribution and the primary event timing distribution.

Usage

```
qprimarycensored(
  p,
  pdist,
  pwindow = 1,
  L = -Inf,
  D = Inf,
  dprimary = stats::dunif,
  dprimary_args = list(),
  ...
)

qpcens(
  p,
  pdist,
  pwindow = 1,
  L = -Inf,
  D = Inf,
  dprimary = stats::dunif,
  dprimary_args = list(),
  ...
)
```

Arguments

p	Vector of probabilities between 0 and 1 for which to compute corresponding quantiles
pdist	Distribution function (CDF). The package can identify base R distributions for potential analytical solutions. For non-base R functions, users can apply <code>add_name_attribute()</code> to yield properly tagged functions if they wish to leverage the analytical solutions.
pwindow	Primary event window
L	Minimum delay (lower truncation point). Defaults to <code>-Inf</code> , meaning no left truncation. For any finite value of L the distribution is left-truncated at L.
D	Maximum delay (upper truncation point). If finite, the distribution is truncated at D. If set to <code>Inf</code> , no upper truncation is applied. Defaults to <code>Inf</code> .
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 <code>dexpgrowth</code> for an exponential growth distribution. See <code>pcd_primary_distributions()</code> for examples. The package can identify base R distributions for potential analytical solutions. For non-base R functions, users can apply <code>add_name_attribute()</code> to yield properly tagged functions if they wish to leverage analytical solutions.
dprimary_args	List of additional arguments to be passed to dprimary. For example, when using <code>dexpgrowth</code> , you would pass <code>list(min = 0, max = pwindow, r = 0.2)</code> to set the minimum, maximum, and rate parameters
...	Additional arguments to be passed to pdist

Details

For each probability, the function finds the delay value where that proportion of events have occurred by that time in the primary event censored distribution. This is done by inverting the cumulative distribution function.

The function creates a `primarycensored` object using `new_pcens()` and then computes the quantiles using `pcens_quantile()`. This approach allows for analytical solutions when available, falling back to numerical methods when necessary.

For example, if $p = 0.5$, the function returns the median delay (truncated over [L, D] if specified) where 50% of censored events occur by this time and 50% occur after.

See `methods(pcens_quantile)` for which combinations have analytical solutions implemented.

Value

Vector of delay values (quantiles) corresponding to the input probabilities

See Also

`new_pcens()` and `pcens_quantile()`

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

Examples

```

# Compute delays where 25%, 50%, and 75% of events occur by (quartiles)
# Using lognormal delays with uniform primary events
qprimarycensored(c(0.25, 0.5, 0.75), plnorm, meanlog = 0, sdlog = 1)

# Same quartiles but with exponential growth in primary events
qprimarycensored(
  c(0.25, 0.5, 0.75), plnorm,
  dprimary = dexpgrowth,
  dprimary_args = list(r = 0.2), meanlog = 0, sdlog = 1
)

# Same quartiles but with truncation at 10
qprimarycensored(
  c(0.25, 0.5, 0.75), plnorm,
  dprimary = dexpgrowth,
  dprimary_args = list(r = 0.2), meanlog = 0, sdlog = 1, D = 10
)

# Left-truncated distribution (e.g., for generation intervals)
qprimarycensored(
  c(0.25, 0.5, 0.75), plnorm,
  L = 1, D = 10, meanlog = 0, sdlog = 1
)

```

rprimarycensored	<i>Generate random samples from a primary event censored distribution</i>
------------------	---

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) and minimum delay (L). The function allows for custom primary event distributions and delay distributions.

Usage

```

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

```

```

)

rpcens(
  n,
  rdist,
  pwindow = 1,
  swindow = 1,
  L = -Inf,
  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 <code>stats::rlnorm()</code> for lognormal distribution.
pwindow	Primary event window
swindow	Integer specifying the window size for rounding the delay (default is 1). If <code>swindow = 0</code> then no rounding is applied.
L	Minimum delay (lower truncation point). Defaults to <code>-Inf</code> , meaning no left truncation. For any finite value of L the distribution is left-truncated at L.
D	Maximum delay (upper truncation point). If finite, the distribution is truncated at D. If set to <code>Inf</code> , no upper truncation is applied. Defaults to <code>Inf</code> .
rprimary	Function to generate random samples from the primary distribution (default is <code>stats::runif()</code>).
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) with parameters ϕ , defined between 0 and the primary event window (`pwindow`):

$$p \sim f_p(\phi), \quad p \in [0, pwindow]$$

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

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

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

$$t = p + d$$

4. Apply upper truncation to remove delays $\geq D$:

$$t_{upper} = \{t \mid t < D\}$$

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

$$t_{rounded} = \lfloor \frac{t_{upper}}{swindow} \rfloor \times swindow$$

6. Apply lower truncation on the rounded values to ensure observed delays are $\geq L$:

$$t_{valid} = \{t_{rounded} \mid t_{rounded} \geq L\}$$

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\(\)](#), [qprimarycensored\(\)](#)

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 = rexprowth, rprimary_args = list(r = 0.2),
  meanlog = 0, sdlog = 1
)

# Example: Left-truncated distribution (e.g., for generation intervals)
rprimarycensored(10, rlnorm, L = 1, D = 10, meanlog = 0, sdlog = 1)
```

Index

- * **check**
 - check_dprimary, 4
 - check_pdist, 5
 - check_truncation, 5
- * **datasets**
 - pcd_distributions, 17
 - pcd_primary_distributions, 19
- * **modelhelpers**
 - fitdistdoublecens, 10
 - pcd_as_stan_data, 14
 - pcd_cmdstan_model, 16
- * **pcens**
 - new_pcens, 13
 - pcens_cdf, 23
 - pcens_cdf.default, 24
 - pcens_cdf.pcens_pgamma_dunif, 25
 - pcens_cdf.pcens_plnorm_dunif, 26
 - pcens_cdf.pcens_pweibull_dunif, 26
 - pcens_quantile, 27
 - pcens_quantile.default, 28
- * **primarycensored**
 - dprimarycensored, 6
 - pprimarycensored, 29
 - qprimarycensored, 32
 - rprimarycensored, 34
- * **primaryeventdistributions**
 - expgrowth, 8
- * **stantools**
 - pcd_load_stan_functions, 18
 - pcd_stan_dist_id, 20
 - pcd_stan_files, 21
 - pcd_stan_function_deps, 22
 - pcd_stan_functions, 21
 - pcd_stan_path, 23
- * **utils**
 - add_name_attribute, 3
 - pcd_dist_name, 17
 - pcd_distributions, 17
 - pcd_primary_distributions, 19
 - add_name_attribute, 3, 17, 18, 20
 - add_name_attribute(), 4, 5, 7, 11, 13, 30, 31, 33
 - check_dprimary, 4, 5, 6
 - check_pdist, 4, 5, 6
 - check_truncation, 4, 5, 5
 - dexpgrowth (expgrowth), 8
 - dpcens (dprimarycensored), 6
 - dprimarycensored, 6, 31, 33, 36
 - dprimarycensored(), 12
 - expgrowth, 8
 - fitdistdoublecens, 10, 15, 16
 - fitdistrplus::fitdist(), 11, 12
 - fitdistrplus::fitdistcens(), 10
 - new_pcens, 13, 24–29
 - new_pcens(), 23–28, 31, 33
 - pcd_as_stan_data, 12, 14, 16
 - pcd_cmdstan_model, 12, 15, 16
 - pcd_cmdstan_model(), 15
 - pcd_dist_name, 3, 17, 17, 20
 - pcd_dist_name(), 3
 - pcd_distributions, 3, 15, 17, 18, 20
 - pcd_distributions(), 18, 20
 - pcd_load_stan_functions, 18, 20–23
 - pcd_primary_distributions, 3, 15, 17, 18, 19
 - pcd_primary_distributions(), 4, 7, 11, 13, 18, 20, 31, 33
 - pcd_stan_dist_id, 19, 20, 21–23
 - pcd_stan_dist_id(), 15
 - pcd_stan_files, 19, 20, 21, 22, 23
 - pcd_stan_function_deps, 19–22, 22, 23
 - pcd_stan_functions, 19–21, 21, 22, 23
 - pcd_stan_path, 19–22, 23
 - pcens_cdf, 13, 23, 24–29

`pcens_cdf()`, 27, 31
`pcens_cdf.default`, 13, 24, 24, 25–29
`pcens_cdf.pcens_pgamma_dunif`, 13, 24, 25, 26–29
`pcens_cdf.pcens_plnorm_dunif`, 13, 24, 25, 26, 27–29
`pcens_cdf.pcens_pweibull_dunif`, 13, 24–26, 26, 28, 29
`pcens_quantile`, 13, 24–27, 27, 29
`pcens_quantile()`, 33
`pcens_quantile.default`, 13, 24–28, 28
`pexpgrowth (expgrowth)`, 8
`ppcens (pprimarycensored)`, 29
`pprimarycensored`, 8, 29, 33, 36
`pprimarycensored()`, 3, 8, 12, 24

`qpcens (qprimarycensored)`, 32
`qprimarycensored`, 8, 31, 32, 36

`rexprowth (expgrowth)`, 8
`rpcens (rprimarycensored)`, 34
`rprimarycensored`, 8, 31, 33, 34

`stats::rlnorm()`, 35
`stats::runif()`, 35
`stats::uniroot()`, 28, 29