

Package ‘primarycensored’

September 25, 2026

Title Primary Event Censored Distributions

Version 1.6.0

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)

Imports lifecycle

Suggests bookdown, cmdstanr, dplyr, fitdistrplus, flexsurv, 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

VignetteBuilder knitr

RoxygenNote 7.3.3

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 2026-09-25 12:00:02 UTC

Contents

add_name_attribute	3
check_dprimary	4
check_pdist	5
check_truncation	6
ddiscretehazard	6
ddiscretetest	7
discretehazard_start	8
dprimarycensored	9
expgrowth	12
fitdistdoublecens	14
hazards_to_pmf	18
new_pcens	19
pcd_as_stan_data	21
pcd_cmdstan_model	23
pcd_distributions	24
pcd_dist_name	25
pcd_load_stan_functions	25
pcd_primary_distributions	26
pcd_stan_dist_id	27
pcd_stan_files	28
pcd_stan_functions	28
pcd_stan_function_deps	29
pcd_stan_path	30
pcens_cdf	30
pcens_cdf.default	31
pcens_cdf.pcens_pdiscretehazard	32
pcens_cdf.pcens_pdiscretetest	33
pcens_cdf.pcens_pgamma_dunif	34
pcens_cdf.pcens_pgengamma.orig_dunif	35

pcens_cdf.pcens_pgengamma_dunif	36
pcens_cdf.pcens_plnorm_dunif	37
pcens_cdf.pcens_pweibull_dunif	38
pcens_pmf	39
pcens_pmf.default	40
pcens_quantile	41
pcens_quantile.default	42
pdiscretehazard	43
pdiscretetest	44
pmf_to_hazards	45
pprimarycensored	46
qprimarycensored	49
rdiscretehazard	52
rdiscretetest	53
rprimarycensored	53
update.pcens	56

Index	58
--------------	-----------

add_name_attribute	<i>Helper method for custom distributions</i>
--------------------	---

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. `punif`, `dexp`) via the `name` attribute.

Usage

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

Arguments

func	Function, for example the p- or d- form of a distribution function.
name	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 $[0, \text{pwindow}]$. 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.
pwindow	Primary event window. Use <code>pwindow = 0</code> for an exactly observed primary event, in which case the delay CDF is used directly.
dprimary_args	[Deprecated] Use <code>primary_args</code> instead.
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. With `pwindow = 0` the primary event time is exact and only the arguments of `dprimary` are checked.

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)
```

ddiscretehazard	<i>Hazard-parameterised piecewise-constant PMF</i>
-----------------	--

Description

Returns the probability mass for each value in x . Converts hazards to a PMF via [hazards_to_pmf\(\)](#) then delegates to [ddiscretetestep\(\)](#).

Usage

```
ddiscretehazard(x, boundaries = NULL, hazards)
```

Arguments

<code>x</code>	Numeric vector of values at which to evaluate the PMF.
<code>boundaries</code>	Numeric vector of length $K + 1$ defining the bin edges. Must be strictly increasing. Defaults to $0:K$.
<code>hazards</code>	Numeric vector of length K (or $K - 1$, in which case a trailing 1 is appended) giving the discrete-time conditional hazard for each bin. Values must be in $[0, 1]$.

Details

Outside fitting it is a deterministic wrapper around `ddiscretetestep()`. It earns its keep as a fitting parameterisation in `fitdistdoublecens()` because the random walk on the logit hazard smooths the recovered PMF.

Value

Numeric vector of PMF values, the same length as `x`.

See Also

`ddiscretetestep()`, `hazards_to_pmf()`, `pmf_to_hazards()`, `fitdistdoublecens()`
 Other `pdiscretehazard`: `discretehazard_start()`, `pdiscretehazard()`, `rdiscretehazard()`

Examples

```
hazards <- c(0.3, 0.5, 1)
ddiscretehazard(c(1, 2, 3), boundaries = 0:3, hazards = hazards)
```

<code>ddiscretetestep</code>	<i>Step (piecewise-constant) PMF</i>
------------------------------	--------------------------------------

Description

Returns the probability mass for each value in `x`. Mass `pmf[i]` is located at `boundaries[i+1]` (the right edge of bin i); all other values return 0.

Usage

```
ddiscretetestep(x, boundaries = NULL, pmf)
```

Arguments

<code>x</code>	Numeric vector of values at which to evaluate the PMF.
<code>boundaries</code>	Numeric vector of length $K + 1$ defining the bin edges. Must be strictly increasing. Defaults to $0:K$ (unit-width daily bins) where K is inferred from <code>length(pmf)</code> .
<code>pmf</code>	Numeric vector of length K giving the probability mass for each bin. Must be non-negative and sum to approximately 1; if either condition is violated the function returns a vector of <code>.Machine\$double.eps</code> rather than 0 (a soft simplex penalty that keeps log-density finite inside fitting closures).

Details

Like `pdiscretetestep`, this function applies a soft simplex penalty: if `pmf` contains negative entries or fails to sum to 1 (within 10^{-8}), the function returns near-zero density (`.Machine$double.eps`) rather than erroring. This makes it safe to call from inside `fitdistrplus::fitdist()` closures driven by `fitdistribdoublecens`.

Value

Numeric vector of PMF values, the same length as `x`.

See Also

Other `pdiscretetestep`: `hazards_to_pmf()`, `pdiscretetestep()`, `pmf_to_hazards()`, `rdiscretetestep()`

Examples

```
ddiscretetestep(c(0, 1, 2, 3), boundaries = 0:3, pmf = c(0.2, 0.5, 0.3))
```

`discretehazard_start` *Start values for the logit-hazard parameterisation*

Description

Builds a named list of starting values for `fitdistribdoublecens()` with `distr = "discretehazard"`. The free parameters are the logit intercept `alpha`, the log random-walk (or random-effect) scale `log_sigma`, and the $K - 1$ innovations `eps_1, \dots, eps_{K-1}`. The final bin hazard is pinned to 1 inside the parameterisation so the implied PMF sums to 1.

Usage

```
discretehazard_start(K, alpha = -2, log_sigma = log(1), eps = 0)
```

Arguments

<code>K</code>	Integer, number of bins in the hazard parameterisation.
<code>alpha</code>	Numeric, start value for the logit intercept.
<code>log_sigma</code>	Numeric, start value for the log scale.
<code>eps</code>	Numeric, scalar or length- $K - 1$ vector of start values for the innovations <code>eps_1, \dots, eps_{K-1}</code> .

Value

Named list suitable for the `start` argument of `fitdistribdoublecens()`.

See Also

Other `pdiscretehazard`: `ddiscretehazard()`, `pdiscretehazard()`, `rdiscretehazard()`

Examples

```
discretehazard_start(K = 5)
```

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 = dunif,
  primary_args = NULL,
  pprimary = NULL,
  dprimary_args = NULL,
  log = FALSE,
  ...,
  check = TRUE
)
```

```
dpcens(
  x,
  pdist,
  pwindow = 1,
  swindow = 1,
  L = -Inf,
  D = Inf,
  dprimary = dunif,
  primary_args = NULL,
  pprimary = NULL,
  dprimary_args = NULL,
  log = FALSE,
  ...,
  check = TRUE
)
```

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 add_name_attribute() to yield properly tagged functions if they wish to leverage the analytical solutions.
pwindow	Primary event window. Use <code>pwindow = 0</code> for an exactly observed primary event, in which case the delay CDF is used directly.
swindow	Secondary event window (default: 1). Use <code>swindow = 0</code> for an exactly observed secondary event, in which case a density is returned rather than a probability (see Details).
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> .
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 <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.
primary_args	List of additional arguments to be passed to <code>dprimary</code> (and the matching primary CDF). 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. Replaces the deprecated <code>dprimary_args</code> ; defaults to <code>NULL</code> .
pprimary	Optional CDF for the primary event distribution. May be a function or a character string naming a primary distribution in <code>pcd_primary_distributions</code> . Defaults to <code>NULL</code> , in which case the primary CDF is looked up automatically from the registry using the "name" attribute of <code>dprimary</code> . When both <code>dprimary</code> and <code>pprimary</code> carry a "name" attribute (or are base R functions whose name can be inferred), the two names must agree on everything other than the leading <code>d/p</code> prefix; mismatches such as <code>dunif + pexpgrowth</code> raise an error. Supplying <code>pprimary</code> explicitly is mainly useful when using a custom primary distribution whose CDF is not in the registry.
dprimary_args	[Deprecated] Use <code>primary_args</code> instead.
log	Logical; if <code>TRUE</code> , probabilities <code>p</code> are given as <code>log(p)</code>
...	Additional arguments to be passed to the distribution function
check	Logical; if <code>TRUE</code> (the default) <code>pdist</code> is validated with check_pdist() and <code>dprimary</code> with check_dprimary() . Set to <code>FALSE</code> to skip both when they have already been validated. check_pdist() evaluates <code>pdist</code> at four points drawn with stats::runif() , so skipping it avoids that cost and leaves the random number stream untouched. Must be given by its full name, as it follows ...

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 creates a pcens object with `new_pcens()` and computes the PMF with `pcens_pmf()`. This evaluates the CDF once for all unique points (including both d and $d + \text{swindow}$) and reuses these values to calculate the PMF for each input value.

When the secondary censoring interval extends past the upper truncation point ($d + \text{swindow} > D$) but the lower endpoint satisfies $d < D$, the upper endpoint is internally clipped to D before evaluating the CDF. The likelihood for such an observation is $P(X \in [d, \min(d + \text{swindow}, D)] \mid L \leq X \leq D)$, which equals the usual interval probability when $d + \text{swindow} \leq D$. This avoids erroring when an observation's secondary window straddles the truncation point (relevant for non-parametric delays such as `pdiscretetestep()`).

Observations with $d \geq D$ are rejected with an error: under the truncation $X \leq D$, no event with latent value $d \geq D$ is observable, and accepting such inputs would otherwise yield a 0/0 likelihood.

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()`.

Zero-width windows:

With `pwindow = 0` the primary event time is known exactly and the primary event censored CDF is the delay CDF, so the PMF is $F(d + \text{swindow}) - F(d)$.

With `swindow = 0` the secondary event time is known exactly. The probability of the interval is then zero, so the density of the primary event censored delay at d is returned instead. This is the derivative of F_{cens} at d , the limit of the PMF divided by `swindow` as `swindow` goes to zero. With `pwindow = 0` as well it is the delay density. With a uniform primary event distribution it is $(F(d) - F(d - \text{pwindow}))/\text{pwindow}$. Otherwise the delay density is integrated against the primary event density. The delay density is found from the name of `pdist` (for example `dgamma()` for `pgamma()`) and an error is raised if it cannot be found. Densities are normalised for truncation in the same way as probabilities. `swindow` may be a vector, so densities and probabilities can be mixed in one call.

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,
  primary_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)

# Example: exact primary events, and exact secondary events (a density)
dprimarycensored(1:3, pweibull, pwindow = 0, shape = 1.5, scale = 2.0)
dprimarycensored(
  1:3, pweibull,
  pwindow = 1, swindow = 0, 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

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 $P[X \leq x]$, otherwise, $P[X > x]$.
n	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 $[\min, \max]$ with rate parameter (r). Its probability density function (PDF) is:

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

The cumulative distribution function (CDF) is:

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

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 :

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

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 $\text{Uniform}(0, 1)$ 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:

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

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 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

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`. 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 = dunif,
  primary_args = NULL,
  pprimary = NULL,
  dprimary_args = NULL,
  truncation_check_multiplier = 2,
  prior = NULL,
  hazard_model = c("rw", "re"),
  check = TRUE,
  ...
)
```

Arguments

<code>censdata</code>	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. Use <code>left == right</code> for an exactly observed secondary event.
<code>distr</code>	A character string naming the distribution to be fitted. Special values "discretetestep" and "discretehazard" select the non-parametric step-distribution fitting; see Details.
<code>left</code>	Column name for lower bound of observed values (default: "left").
<code>right</code>	Column name for upper bound of observed values (default: "right").
<code>pwindow</code>	Column name for primary window (default: "pwindow"). Use a primary window of 0 for an exactly observed primary event.

L	Column name for minimum delay (lower truncation point). For any finite L the distribution is left-truncated at L; use $L = -\text{Inf}$ 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, $L = -\text{Inf}$ 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"). Observations whose secondary censoring interval straddles D ($\text{left} < D \leq \text{right}$) are accepted: the upper endpoint is internally clipped to D and the likelihood becomes $P(X \text{ in } [\text{left}, \min(\text{right}, D)] \mid L \leq X \leq D)$. This is a no-op for the standard parametric case where $\text{right} \leq D$. Observations with $\text{left} \geq D$ are rejected because under truncation at D no event with latent value $\geq D$ is observable.
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 $[0, \text{pwindow}]$. Defaults to a uniform distribution over $[0, \text{pwindow}]$. Users can provide custom functions or use helper functions like dexpgrowth 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.
primary_args	List of additional arguments to be passed to dprimary (and the matching primary CDF). 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. Replaces the deprecated dprimary_args; defaults to NULL.
pprimary	Optional CDF for the primary event distribution. May be a function or a character string naming a primary distribution in pcd_primary_distributions . Defaults to NULL, in which case the primary CDF is looked up automatically from the registry using the "name" attribute of dprimary. When both dprimary and pprimary carry a "name" attribute (or are base R functions whose name can be inferred), the two names must agree on everything other than the leading d/p prefix; mismatches such as dunif + pexpgrowth raise an error. Supplying pprimary explicitly is mainly useful when using a custom primary distribution whose CDF is not in the registry.
dprimary_args	[Deprecated] Use primary_args instead.
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.
prior	Optional list of prior settings used by the dist function's fit_penalty attribute (currently only "discretehazard"). Each element is itself a list with mean and sd entries. Defaults are used for any component not supplied. See pdiscretehazard() for the default values.
hazard_model	One of "rw" (default) or "re". Only consulted when <code>distr = "discretehazard"</code> . "rw" selects the random-walk transform $\text{logit}(h_i) = \alpha + \sigma * \text{cumsum}(\text{eps})$; "re" selects the IID logit random-effect transform $\text{logit}(h_i) = \alpha + \sigma * \text{eps}_i$. See Details.

check Logical; if TRUE (the default) `pdist` is validated with `check_pdist()` and `dprimary` with `check_dprimary()`. Neither changes across a fit, so validation runs on the first likelihood evaluation only rather than on every one. Set to FALSE to skip it entirely. For non-parametric distributions, `start` is required and determines the number of bins; pass boundaries here to override the default $0:K$ unit-width bins.

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

Details

How distribution functions are resolved:

The `distr` argument names a distribution. The function looks up the density and CDF functions by prepending `d` and `p` to the name (e.g. `distr = "gamma"` resolves to `dgamma()` and `pgamma()`). Custom distributions can be used as long as the corresponding `d<distr>()` and `p<distr>()` functions are defined.

Parametric distributions are fitted in the parameterisation named by `start`, and the returned estimates and covariance matrix use the same names. For example, `gamma` can be fitted with either `start = list(shape = , rate = )` or `start = list(shape = , scale = )`. Parameters can be held fixed by passing `fix.arg` to `fitdistrplus::fitdist()` through ..., either as a named list or as a function of the delays returning one.

Non-parametric distributions:

Two non-parametric distributions are supported. They share a common fitting machinery: the `dist` function carries a `vector_param` attribute ("`pmf`" for `pdiscreteteststep()/ddiscreteteststep()`, "`hazards`" for `pdiscretehazard()/ddiscretehazard()`) that drives this function to build a closure mapping flat scalar parameters into the underlying vector argument.

- `distr = "discreteteststep"`: free parameters `p1, ..., p_{K-1}` (in $[0, 1]$); the last bin probability is $1 - \text{sum}(p1, \dots, p_{K-1})$. See `pdiscreteteststep()` for parameterisation details and the soft simplex penalty applied when probabilities are infeasible.
- `distr = "discretehazard"`: free parameters `alpha, log_sigma, eps_1, ..., eps_{K-1}`. The hazard form parameterises the same family of step distributions as "`discreteteststep`", but its free parameters drive either a Gaussian random walk on the logit hazard (`hazard_model = "rw"`, the default, $\text{logit}(h_i) = \alpha + \sigma * \text{cumsum}(\text{eps})$) or an IID logit random-effect transform (`hazard_model = "re"`, $\text{logit}(h_i) = \alpha + \sigma * \text{eps}_i$ with $\text{eps}_i \sim N(0, 1)$). The smoothing of the random walk regularises the recovered PMF against overfitting in sparse data and replaces the simplex constraint with an unconstrained optimisation; the random-effect variant models hazards as independent draws around `alpha` rather than a smoothed trajectory. See `pdiscretehazard()` for full parameterisation details and the MAP-equivalent prior penalty applied during fitting; pass `prior` to override the default prior settings.

For non-parametric distributions `K` is implied by `length(start)`: $K = \text{length}(\text{start}) + 1$ for "`discreteteststep`" and $K = \text{length}(\text{start}) - 1$ for "`discretehazard`". `start` is therefore required.

Exact observations:

Rows with `pwindow = 0` have an exactly known primary event time. Rows with `left == right` have an exactly known secondary event time and contribute a density rather than a probability to

the likelihood (see [dprimarycensored\(\)](#)). Rows of different types can be mixed in one fit, as for the exact, single interval censored and doubly interval censored observations of `coarseDataTools::dic.fit()`. Data with the primary event in [EL, ER] and the secondary event in [SL, SR] map to `left = SL - EL`, `right = SR - EL` and `pwindow = ER - EL`.

Value

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

See Also

[pdiscretetest\(\)](#) [pdiscretehazard\(\)](#)

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)

# Example with discretetest (non-parametric PMF) distribution
set.seed(42)
true_pmf <- c(0.1, 0.3, 0.4, 0.15, 0.05)
step_samples <- rprimarycensored(
  500, rdiscretetest,
  boundaries = 0:5, pmf = true_pmf,
```

```

    pwindow = 1, swindow = 1, D = 6
  )
  step_data <- data.frame(
    left = step_samples,
    right = step_samples + 1,
    pwindow = rep(1, 500),
    D = rep(6, 500)
  )
  fit_step <- fitdistdoublecens(
    step_data,
    distr = "discretetestep",
    boundaries = 0:5,
    start = as.list(setNames(rep(0.2, 4), paste0("p", 1:4)))
  )

# Example with discretehazard (logit-hazard random walk) distribution
fit_haz <- fitdistdoublecens(
  step_data,
  distr = "discretehazard",
  boundaries = 0:5,
  start = c(
    list(alpha = -2, log_sigma = log(0.5)),
    as.list(setNames(rep(0, 4), paste0("eps_", 1:4)))
  )
)

```

hazards_to_pmf

Convert discrete-time hazards to a PMF

Description

Given a vector of discrete-time hazards $h_1, \dots, h_K$, computes the corresponding PMF:

$$pmf_i = h_i \prod_{j < i} (1 - h_j)$$

Usage

```
hazards_to_pmf(hazards)
```

Arguments

hazards Numeric vector of hazards in $[0, 1]$. The last entry must equal 1 (within 10^{-8}), or a vector of length $K - 1$ may be supplied and the trailing 1 will be appended.

Details

The last hazard must equal 1 (to ensure the PMF sums to 1). If a vector of length $K - 1$ is supplied (all hazards except the final exit hazard), 1 is appended automatically.

Value

Numeric vector of PMF values.

See Also

Other `pdiscretetest`: [ddiscretetest\(\)](#), [pdiscretetest\(\)](#), [pmf_to_hazards\(\)](#), [rdiscretetest\(\)](#)

Examples

```
hazards_to_pmf(c(0.2, 0.3, 1))
```

`new_pcens`

S3 class for primary event censored distribution computation

Description

S3 class for primary event censored distribution computation

Usage

```
new_pcens(
  pdist,
  dprimary,
  primary_args = NULL,
  pprimary = NULL,
  dprimary_args = NULL,
  ...
)
```

Arguments

- | | |
|---------------------------|--|
| <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>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. |
| <code>primary_args</code> | List of additional arguments to be passed to <code>dprimary</code> (and the looked-up <code>pprimary</code>). Replaces the deprecated <code>dprimary_args</code> . |

`pprimary` CDF of the primary event distribution. May be a function or a character string naming a primary distribution in `pcd_primary_distributions`. When `NULL` (the default), it is looked up automatically from the registry using the "name" attribute of `dprimary`. When both `dprimary` and `pprimary` carry a name, the two must agree on everything other than the leading d/p prefix; mismatches such as `dunif + pexpgrowth` raise an error.

`dprimary_args` [Deprecated] Use `primary_args` instead.

`...` Additional arguments to be passed to `pdist`

Value

An object with class hierarchy `c("pcens_{pdist_name}_{dprimary_name}", "pcens_{pdist_name}", "pcens")`. It is a list with the fields:

`pdist` The delay distribution CDF.

`dprimary` The primary event distribution density.

`primary_args` A list of arguments passed to `dprimary` and `pprimary`.

`dprimary_args` A copy of `primary_args`, kept for backward compatibility.

`pprimary` The primary event CDF, or `NULL` if none is available.

`args` A named list of the delay distribution parameters passed through `...`

It can be used with `pcens_cdf()` to compute the primary event censored CDF and with `pcens_pmf()` to compute the PMF. Use `update()` to change its parameters.

See Also

Low level primary event censored distribution objects and methods `pcens_cdf()`, `pcens_cdf.default()`, `pcens_cdf.pcens_pdiscretehazard()`, `pcens_cdf.pcens_pdiscretetestep()`, `pcens_cdf.pcens_pgamma_dunif()`, `pcens_cdf.pcens_pgengamma.orig_dunif()`, `pcens_cdf.pcens_pgengamma_dunif()`, `pcens_cdf.pcens_plnorm_dunif()`, `pcens_cdf.pcens_pweibull_dunif()`, `pcens_pmf()`, `pcens_pmf.default()`, `pcens_quantile()`, `pcens_quantile.default()`, `update.pcens()`

Examples

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

pcd_as_stan_data	<i>Prepare data for primarycensored Stan model</i>
------------------	--

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,
  dist_options = NULL
)
```

Arguments

<code>data</code>	A data frame containing the delay data.
<code>delay</code>	Column name for observed delays (default: "delay")
<code>delay_upper</code>	Column name for upper bound of delays (default: "delay_upper")
<code>n</code>	Column name for count of observations (default: "n")
<code>pwindow</code>	Column name for primary window (default: "pwindow")
<code>start_relative_obs_time</code>	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 $-\text{Inf}$ to indicate no lower truncation. If the column is not present in data, $L = -\text{Inf}$ is assumed for all observations. (default: "start_relative_obs_time")
<code>relative_obs_time</code>	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 <code>start_relative_obs_time</code>) or $+\text{Inf}$ to indicate no upper truncation. (default: "relative_obs_time")

<code>dist_id</code>	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.
<code>primary_id</code>	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.
<code>param_bounds</code>	A list with elements lower and upper, each a numeric vector specifying bounds for the delay distribution parameters.
<code>primary_param_bounds</code>	A list with elements lower and upper, each a numeric vector specifying bounds for the primary distribution parameters.
<code>priors</code>	A list with elements location and scale, each a numeric vector specifying priors for the delay distribution parameters.
<code>primary_priors</code>	A list with elements location and scale, each a numeric vector specifying priors for the primary distribution parameters.
<code>compute_log_lik</code>	Logical; compute log likelihood? (default: FALSE)
<code>use_reduce_sum</code>	Logical; use reduce_sum for performance? (default: FALSE)
<code>truncation_check_multiplier</code>	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.
<code>dist_options</code>	Optional list carrying the shape of a non-parametric delay. When <code>dist_id</code> is one of 26 (step CDF, Dirichlet prior on the PMF), 27 (step CDF, random walk on the logit hazards) or 28 (step CDF, IID logit random effects on the hazards), this list must be supplied and must contain: • K: integer, the number of bins. • boundaries: numeric vector of length K + 1. Priors and param_bounds follow the same channel as the parametric path: pass them through priors and param_bounds. The semantics of priors depend on <code>dist_id</code>: • <code>dist_id = 26</code>: <code>priors\$scale</code> is the length-K Dirichlet concentration vector; <code>priors\$location</code> is unused. • <code>dist_id = 27</code> or <code>28</code>: <code>priors\$location = c(alpha_mean, log_sigma_mean)</code> and <code>priors\$scale = c(alpha_sd, log_sigma_sd)</code>. <code>param_bounds</code> is unused for the non-parametric paths; pass <code>list(lower = numeric(0), upper = numeric(0))</code>. If priors is empty when a non-parametric <code>dist_id</code> is given, sensible defaults are applied (Dirichlet(1, ..., 1) for <code>dist_id = 26</code>; $N(0, 5)$ on alpha and $N(0, 1)$ on log_sigma for 27 and 28).

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	<i>Create a CmdStanModel with primarycensored Stan functions</i>
-------------------	--

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	<i>Supported delay distributions</i>
-------------------	--------------------------------------

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 28 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	<i>Get distribution function cdf or pdf name</i>
---------------	--

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	<i>Load Stan functions as a string</i>
-------------------------	--

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

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	<i>Get distribution stan ID by name</i>
------------------	---

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. Use <code>pwindow = 0</code> for an exactly observed primary event, in which case the delay CDF is used directly.
<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.

Details

When `pwindow = 0` the primary event time is known exactly and the primary event censored CDF is the delay CDF. This case is handled before dispatch, so every method returns `pdist(q)` without integrating over the primary event window.

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_pdiscretehazard()`, `pcens_cdf.pcens_pdiscretetest()`, `pcens_cdf.pcens_pgamma_dunif()`, `pcens_cdf.pcens_pgengamma.orig_dunif()`, `pcens_cdf.pcens_pgengamma_dunif()`, `pcens_cdf.pcens_plnorm_dunif()`, `pcens_cdf.pcens_pweibull_dunif()`, `pcens_pmf()`, `pcens_pmf.default()`, `pcens_quantile()`, `pcens_quantile.default()`, `update.pcens()`

<code>pcens_cdf.default</code>	<i>Default method for computing primary event censored CDF</i>
--------------------------------	--

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

<code>object</code>	A primarycensored object as created by <code>new_pcens()</code> .
<code>q</code>	Vector of quantiles
<code>pwindow</code>	Primary event window. Use <code>pwindow = 0</code> for an exactly observed primary event, in which case the delay CDF is used directly.
<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.

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_pdiscretehazard\(\)](#), [pcens_cdf.pcens_pdiscretetestep\(\)](#), [pcens_cdf.pcens_pgamma_dunif\(\)](#), [pcens_cdf.pcens_pgengamma.orig_dunif\(\)](#), [pcens_cdf.pcens_pgengamma_dunif\(\)](#), [pcens_cdf.pcens_plnorm_dunif\(\)](#), [pcens_cdf.pcens_pweibull_dunif\(\)](#), [pcens_pmf\(\)](#), [pcens_pmf.default\(\)](#), [pcens_quantile\(\)](#), [pcens_quantile.default\(\)](#), [update.pcens\(\)](#)

Examples

```
# Create a primarycensored object with gamma delay and uniform primary
pcens_obj <- new_pcens(
  pdist = pgamma,
  dprimary = dunif,
  primary_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_pdiscretehazard`

Method for hazard CDF delay with general primary event distribution

Description

Computes the analytic primary event censored CDF for a hazard-parameterised piecewise-constant delay distribution. Converts hazards to a PMF via [hazards_to_pmf\(\)](#) then dispatches back through [pcens_cdf\(\)](#) using a freshly constructed step-distribution object. The same primary event distribution and arguments are preserved.

Usage

```
## S3 method for class 'pcens_pdiscretehazard'
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 pwindow = 0 for an exactly observed primary event, in which case the delay CDF is used directly.
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_pdiscretetestep\(\)](#), [pcens_cdf.pcens_pgamma_dunif\(\)](#), [pcens_cdf.pcens_pgengamma.orig_dunif\(\)](#), [pcens_cdf.pcens_pgengamma_dunif\(\)](#), [pcens_cdf.pcens_plnorm_dunif\(\)](#), [pcens_cdf.pcens_pweibull_dunif\(\)](#), [pcens_pmf\(\)](#), [pcens_pmf.default\(\)](#), [pcens_quantile\(\)](#), [pcens_quantile.default\(\)](#), [update_pcens\(\)](#)

pcens_cdf.pcens_pdiscretetestep

Method for step CDF delay with general primary event distribution

Description

Computes the analytic primary event censored CDF for a piecewise-constant (step) delay distribution and an arbitrary primary event distribution whose CDF $F_{primary}$ is available via object\$pprimary.

Usage

```
## S3 method for class 'pcens_pdiscretetestep'
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 pwindow = 0 for an exactly observed primary event, in which case the delay CDF is used directly.
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

The observation CDF is

$$F_{obs}(q) = \int_0^{p_{window}} F_{step}(q - p) dF_{primary}(p)$$

Because F_{step} is piecewise constant, the integral reduces to

$$F_{obs}(q) = \sum_k c_k [F_{primary}(p_k^{end}) - F_{primary}(p_k^{start})]$$

where c_k is the constant value of F_{step} on the k -th sub-interval of the primary event window induced by the step-function knots. The partition is exact for any bin widths, so bins may be wider or narrower than `pwindow`, and for boundaries that start below zero.

Falls back to `pcens_cdf.default` when `use_numeric = TRUE` or when no primary CDF is available on the object.

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_pdiscretehazard()`, `pcens_cdf.pcens_pgamma_dunif()`, `pcens_cdf.pcens_pgengamma.orig_dunif()`, `pcens_cdf.pcens_pgengamma_dunif()`, `pcens_cdf.pcens_plnorm_dunif()`, `pcens_cdf.pcens_pweibull_dunif()`, `pcens_pmf()`, `pcens_pmf.default()`, `pcens_quantile()`, `pcens_quantile.default()`, `update_pcens()`

`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 pwindow = 0 for an exactly observed primary event, in which case the delay CDF is used directly.
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_pdiscretehazard\(\)](#), [pcens_cdf.pcens_pdiscretetestep\(\)](#), [pcens_cdf.pcens_pgengamma.orig_dunif\(\)](#), [pcens_cdf.pcens_pgengamma_dunif\(\)](#), [pcens_cdf.pcens_plnorm_dunif\(\)](#), [pcens_cdf.pcens_pweibull_dunif\(\)](#), [pcens_pmf\(\)](#), [pcens_pmf.default\(\)](#), [pcens_quantile\(\)](#), [pcens_quantile.default\(\)](#), [update_pcens\(\)](#)

pcens_cdf.pcens_pgengamma.orig_dunif

Method for generalised gamma delay with uniform primary

Description

Analytical solution for the generalised gamma distribution in the Stacy parameterisation used by `flexsurv::pgengamma.orig()`, with parameters shape, scale and k . The delay CDF is $F_T(t) = P(k, (t/\theta)^a)$ with P the regularised lower incomplete gamma function, a the shape and θ the scale. The mean is $E[T] = \theta\Gamma(k + 1/a)/\Gamma(k)$ and the partial expectation distribution is $\tilde{F}_T(t) = P(k + 1/a, (t/\theta)^a)$, so the solution generalises the gamma (shape = 1) and Weibull ($k = 1$) cases. See `vignette("analytic-solutions")` for the derivation.

Usage

```
## S3 method for class 'pcens_pgengamma.orig_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 pwindow = 0 for an exactly observed primary event, in which case the delay CDF is used directly.

`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_pdiscretehazard()`, `pcens_cdf.pcens_pdiscretetestp()`, `pcens_cdf.pcens_pgamma_dunif()`, `pcens_cdf.pcens_pgengamma_dunif()`, `pcens_cdf.pcens_plnorm_dunif()`, `pcens_cdf.pcens_pweibull_dunif()`, `pcens_pmf()`, `pcens_pmf.default()`, `pcens_quantile()`, `pcens_quantile.default()`, `update.pcens()`

Examples

```
pcens_obj <- new_pcens(
  pdist = flexsurv::pgengamma.orig,
  dprimary = dunif,
  dprimary_args = list(min = 0, max = 1),
  shape = 1.5,
  scale = 2,
  k = 0.8
)
pcens_cdf(pcens_obj, q = c(1, 4, 8), pwindow = 1)
```

```
pcens_cdf.pcens_pgengamma_dunif
```

Method for generalised gamma (Prentice parameterisation) delay with uniform primary

Description

Analytical solution for the generalised gamma distribution in the Prentice parameterisation used by `flexsurv::pgengamma()`, with parameters μ , σ and Q . For $Q > 0$ this is mapped to the Stacy parameterisation of `pcens_cdf.pcens_pgengamma.orig_dunif()` via $\text{shape} = Q / \sigma$, $\text{scale} = \exp(\mu) * Q^{(2 * \sigma / Q)}$ and $k = 1 / Q^2$. For $Q \leq 0$ (the lognormal and reflected cases) the numerical `pcens_cdf.default()` method is used.

Usage

```
## S3 method for class 'pcens_pgengamma_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 pwindow = 0 for an exactly observed primary event, in which case the delay CDF is used directly.
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_pdiscretehazard\(\)](#), [pcens_cdf.pcens_pdiscretetestp\(\)](#), [pcens_cdf.pcens_pgamma_dunif\(\)](#), [pcens_cdf.pcens_pgengamma.orig_dunif\(\)](#), [pcens_cdf.pcens_plnorm_dunif\(\)](#), [pcens_cdf.pcens_pweibull_dunif\(\)](#), [pcens_pmf\(\)](#), [pcens_pmf.default\(\)](#), [pcens_quantile\(\)](#), [pcens_quantile.default\(\)](#), [update_pcens\(\)](#)

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 pwindow = 0 for an exactly observed primary event, in which case the delay CDF is used directly.
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_pdiscretehazard\(\)](#), [pcens_cdf.pcens_pdiscreteteststep\(\)](#), [pcens_cdf.pcens_pgamma_dunif\(\)](#), [pcens_cdf.pcens_pgengamma.orig_dunif\(\)](#), [pcens_cdf.pcens_pgengamma_dunif\(\)](#), [pcens_cdf.pcens_pweibull_dunif\(\)](#), [pcens_pmf\(\)](#), [pcens_pmf.default\(\)](#), [pcens_quantile\(\)](#), [pcens_quantile.default\(\)](#), [update_pcens\(\)](#)

`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

<code>object</code>	A primarycensored object as created by new_pcens() .
<code>q</code>	Vector of quantiles
<code>pwindow</code>	Primary event window. Use <code>pwindow = 0</code> for an exactly observed primary event, in which case the delay CDF is used directly.
<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\(\)](#), [pcens_cdf.default\(\)](#), [pcens_cdf.pcens_pdiscretehazard\(\)](#), [pcens_cdf.pcens_pdiscreteteststep\(\)](#), [pcens_cdf.pcens_pgamma_dunif\(\)](#), [pcens_cdf.pcens_pgengamma.orig_dunif\(\)](#), [pcens_cdf.pcens_pgengamma_dunif\(\)](#), [pcens_cdf.pcens_plnorm_dunif\(\)](#), [pcens_pmf\(\)](#), [pcens_pmf.default\(\)](#), [pcens_quantile\(\)](#), [pcens_quantile.default\(\)](#), [update_pcens\(\)](#)

pcens_pmf

*Compute primary event censored PMF***Description**

Computes the primary event censored PMF for a pcens object as created by [new_pcens\(\)](#). Secondary event windows and truncation are handled as in [dprimarycensored\(\)](#).

Usage

```
pcens_pmf(object, x, pwindow, swindow = 1, L = -Inf, D = Inf, log = FALSE, ...)
```

Arguments

object	A pcens object as created by new_pcens() .
x	Vector of quantiles
pwindow	Primary event window. Use pwindow = 0 for an exactly observed primary event, in which case the delay CDF is used directly.
swindow	Secondary event window (default: 1). Use swindow = 0 for an exactly observed secondary event, in which case a density is returned rather than a probability (see Details).
L	Minimum delay (lower truncation point). Defaults to -Inf, 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 Inf, no upper truncation is applied. Defaults to Inf.
log	Logical; if TRUE, probabilities p are given as log(p)
...	Additional arguments passed to methods.

Value

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

See Also

Low level primary event censored distribution objects and methods [new_pcens\(\)](#), [pcens_cdf\(\)](#), [pcens_cdf.default\(\)](#), [pcens_cdf.pcens_pdiscretehazard\(\)](#), [pcens_cdf.pcens_pdiscretetestep\(\)](#), [pcens_cdf.pcens_pgamma_dunif\(\)](#), [pcens_cdf.pcens_pgengamma.orig_dunif\(\)](#), [pcens_cdf.pcens_pgengamma_dunif\(\)](#), [pcens_cdf.pcens_plnorm_dunif\(\)](#), [pcens_cdf.pcens_pweibull_dunif\(\)](#), [pcens_pmf.default\(\)](#), [pcens_quantile\(\)](#), [pcens_quantile.default\(\)](#), [update.pcens\(\)](#)

pcens_pmf.default	<i>Default method for computing primary event censored PMF</i>
-------------------	--

Description

Computes the PMF by differencing [pcens_cdf\(\)](#) at x and $\min(x + \text{swindow}, D)$, normalised over $[L, D]$. Where $\text{swindow} = 0$ the primary event censored density at x is returned instead. See [dprimarycensored\(\)](#) for the details.

Usage

```
## Default S3 method:
pcens_pmf(object, x, pwindow, swindow = 1, L = -Inf, D = Inf, log = FALSE, ...)
```

Arguments

object	A pcens object as created by new_pcens() .
x	Vector of quantiles
pwindow	Primary event window. Use $\text{pwindow} = 0$ for an exactly observed primary event, in which case the delay CDF is used directly.
swindow	Secondary event window (default: 1). Use $\text{swindow} = 0$ for an exactly observed secondary event, in which case a density is returned rather than a probability (see Details).
L	Minimum delay (lower truncation point). Defaults to $-\text{Inf}$, 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 Inf , no upper truncation is applied. Defaults to Inf .
log	Logical; if TRUE, probabilities p are given as $\log(p)$
...	Additional arguments passed to methods.

Value

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

See Also

Low level primary event censored distribution objects and methods [new_pcens\(\)](#), [pcens_cdf\(\)](#), [pcens_cdf.default\(\)](#), [pcens_cdf.pcens_pdiscretehazard\(\)](#), [pcens_cdf.pcens_pdiscretetestep\(\)](#), [pcens_cdf.pcens_pgamma_dunif\(\)](#), [pcens_cdf.pcens_pgengamma.orig_dunif\(\)](#), [pcens_cdf.pcens_pgengamma_dunif\(\)](#), [pcens_cdf.pcens_plnorm_dunif\(\)](#), [pcens_cdf.pcens_pweibull_dunif\(\)](#), [pcens_pmf\(\)](#), [pcens_quantile\(\)](#), [pcens_quantile.default\(\)](#), [update_pcens\(\)](#)

Examples

```
obj <- new_pcens(
  pdist = pgamma, dprimary = dunif,
  primary_args = list(min = 0, max = 1),
  shape = 3, scale = 2
)
pcens_pmf(obj, x = 0:9, pwindow = 1, D = 10)
```

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

<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. Use <code>pwindow = 0</code> for an exactly observed primary event, in which case the delay CDF is used directly.
<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>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>...</code>	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_pdiscretehazard()`, `pcens_cdf.pcens_pdiscretestep()`, `pcens_cdf.pcens_pgamma_dunif()`, `pcens_cdf.pcens_pgengamma.orig_dunif()`, `pcens_cdf.pcens_pgengamma_dunif()`, `pcens_cdf.pcens_plnorm_dunif()`, `pcens_cdf.pcens_pweibull_dunif()`, `pcens_pmf()`, `pcens_pmf.default()`, `pcens_quantile.default()`, `update.pcens()`

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. Use <code>pwindow = 0</code> for an exactly observed primary event, in which case the delay CDF is used directly.
<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_pdiscretehazard\(\)](#), [pcens_cdf.pcens_pdiscretetestep\(\)](#), [pcens_cdf.pcens_pgamma_dunif\(\)](#), [pcens_cdf.pcens_pgengamma.orig_dunif\(\)](#), [pcens_cdf.pcens_pgengamma_dunif\(\)](#), [pcens_cdf.pcens_plnorm_dunif\(\)](#), [pcens_cdf.pcens_pweibull_dunif\(\)](#), [pcens_pmf\(\)](#), [pcens_pmf.default\(\)](#), [pcens_quantile\(\)](#), [update.pcens\(\)](#)

Examples

```
# Create a primarycensored object with gamma delay and uniform primary
pcens_obj <- new_pcens(
  pdist = pgamma,
  dprimary = dunif,
  primary_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)
```

pdiscretehazard

Hazard-parameterised piecewise-constant CDF

Description

Returns the CDF of a discrete distribution specified by its bin-wise discrete-time hazards. Converts hazards to a PMF via [hazards_to_pmf\(\)](#) then delegates to [pdiscretetestep\(\)](#).

Usage

```
pdiscretehazard(q, boundaries = NULL, hazards)
```

Arguments

q	Numeric vector of quantiles.
boundaries	Numeric vector of length $K + 1$ defining the bin edges. Must be strictly increasing. Defaults to $0:K$.
hazards	Numeric vector of length K (or $K - 1$, in which case a trailing 1 is appended) giving the discrete-time conditional hazard for each bin. Values must be in $[0, 1]$.

Details

Outside fitting it is a deterministic wrapper around `pdiscretetestep()`. It earns its keep as a fitting parameterisation in `fitdistdoublecens()` because the random walk on the logit hazard smooths the recovered PMF.

Value

Numeric vector of CDF values, the same length as q.

See Also

`pdiscretetestep()`, `hazards_to_pmf()`, `pmf_to_hazards()`, `fitdistdoublecens()`

Other `pdiscretehazard`: `ddiscretehazard()`, `discretehazard_start()`, `rdiscretehazard()`

Examples

```
hazards <- c(0.3, 0.5, 1)
pdiscretehazard(c(0.5, 1, 2, 3), boundaries = 0:3, hazards = hazards)
```

pdiscretetestep	<i>Step (piecewise-constant) CDF</i>
-----------------	--------------------------------------

Description

Returns the CDF of a discrete distribution whose mass is concentrated at the right boundary of each bin. The CDF is right-continuous and piecewise constant with jumps at `boundaries[2]`, ..., `boundaries[K+1]`.

Usage

```
pdiscretetestep(q, boundaries = NULL, pmf)
```

Arguments

<code>q</code>	Numeric vector of quantiles.
<code>boundaries</code>	Numeric vector of length $K + 1$ defining the bin edges. Must be strictly increasing. Defaults to $0:K$ (unit-width daily bins) where K is inferred from <code>length(pmf)</code> .
<code>pmf</code>	Numeric vector of length K giving the probability mass for each bin. Must be non-negative and sum to approximately 1; if either condition is violated the function returns a vector of zeros (a soft simplex penalty for use inside optimisation).

Details

Below `boundaries[1]` the function returns 0. At `boundaries[i+1]` (the right edge of bin i), F jumps by `pmf[i]`, so $F(\text{boundaries}[i+1]) = \sum_{j=1}^i \text{pmf}_j$. For q in $[\text{boundaries}[i], \text{boundaries}[i+1])$, F equals $\sum_{j=1}^{i-1} \text{pmf}_j$. At or above `boundaries[K+1]` the function returns 1.

Use with `fitdistdoublecens()`:

This function carries the attribute `vector_param = "pmf"` so that `fitdistdoublecens` can drive it from a flat list of scalar parameters $p_1, \dots, p_{K-1}$. The free parameters are the first $K - 1$ bin probabilities; the last is set to $1 - \text{sum}(p_1, \dots, p_{K-1})$. When the implied probabilities violate the simplex (any negative entry, or sum departing from 1 by more than 10^{-8}), the function returns 0 (or near-zero density in `ddiscretetest`) rather than erroring; this drives the optimiser back to the feasible region.

Value

Numeric vector of CDF values, the same length as `q`.

See Also

Other `pdiscretetest`: `ddiscretetest`(), `hazards_to_pmf`(), `pmf_to_hazards`(), `rdiscretetest`()

Examples

```
# Two-bin PMF: mass 0.3 at x=1, mass 0.7 at x=2
pdiscretetest(c(0.5, 1, 1.5, 2), boundaries = 0:2, pmf = c(0.3, 0.7))
```

`pmf_to_hazards`

Convert a PMF to discrete-time hazards

Description

Inverts `hazards_to_pmf()`: given a PMF, computes the discrete-time conditional hazard at each time point.

$$h_i = \text{pmf}_i / (1 - \sum_{j < i} \text{pmf}_j)$$

Usage

```
pmf_to_hazards(pmf)
```

Arguments

pmf Numeric vector of probabilities. Must be non-negative and sum to approximately 1.

Details

The returned vector has the same length as `pmf`, with the last entry equal to 1.

Value

Numeric vector of hazards in $[0, 1]$.

See Also

Other `pdiscretetest`: [ddiscretetest\(\)](#), [hazards_to_pmf\(\)](#), [pdiscretetest\(\)](#), [rdiscretetest\(\)](#)

Examples

```
pmf_to_hazards(c(0.2, 0.3, 0.5))
```

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) 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 = dunif,
  primary_args = NULL,
  pprimary = NULL,
  dprimary_args = NULL,
  ...,
  check = TRUE
```

```

)

ppcens(
  q,
  pdist,
  pwindow = 1,
  L = -Inf,
  D = Inf,
  dprimary = dunif,
  primary_args = NULL,
  pprimary = NULL,
  dprimary_args = NULL,
  ...,
  check = TRUE
)

```

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. Use <code>pwindow = 0</code> for an exactly observed primary event, in which case the delay CDF is used directly.
<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.
<code>primary_args</code>	List of additional arguments to be passed to <code>dprimary</code> (and the matching primary CDF). 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. Replaces the deprecated <code>dprimary_args</code> ; defaults to <code>NULL</code> .
<code>pprimary</code>	Optional CDF for the primary event distribution. May be a function or a character string naming a primary distribution in <code>pcd_primary_distributions</code> . Defaults to <code>NULL</code> , in which case the primary CDF is looked up automatically from the registry using the "name" attribute of <code>dprimary</code> . When both <code>dprimary</code> and <code>pprimary</code> carry a "name" attribute (or are base R functions whose name can

be inferred), the two names must agree on everything other than the leading d/p prefix; mismatches such as `dunif + pexpgrowth` raise an error. Supplying `pprimary` explicitly is mainly useful when using a custom primary distribution whose CDF is not in the registry.

`dprimary_args` [Deprecated] Use `primary_args` instead.
`...` Additional arguments to be passed to `pdist`
`check` Logical; if TRUE (the default) `pdist` is validated with `check_pdist()` and `dprimary` with `check_dprimary()`. Set to FALSE to skip both when they have already been validated. `check_pdist()` evaluates `pdist` at four points drawn with `stats::runif()`, so skipping it avoids that cost and leaves the random number stream untouched. Must be given by its full name, as it follows ...

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,
  primary_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 = dunif,
  primary_args = NULL,
  pprimary = NULL,
  dprimary_args = NULL,
  ...,
  check = TRUE
)

qpcens(
  p,
  pdist,
```

```

pwindow = 1,
L = -Inf,
D = Inf,
dprimary = dunif,
primary_args = NULL,
pprimary = NULL,
dprimary_args = NULL,
...,
check = TRUE
)

```

Arguments

<code>p</code>	Vector of probabilities between 0 and 1 for which to compute corresponding 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. Use <code>pwindow = 0</code> for an exactly observed primary event, in which case the delay CDF is used directly.
<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.
<code>primary_args</code>	List of additional arguments to be passed to <code>dprimary</code> (and the matching primary CDF). 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. Replaces the deprecated <code>dprimary_args</code> ; defaults to <code>NULL</code> .
<code>pprimary</code>	Optional CDF for the primary event distribution. May be a function or a character string naming a primary distribution in <code>pcd_primary_distributions</code> . Defaults to <code>NULL</code> , in which case the primary CDF is looked up automatically from the registry using the "name" attribute of <code>dprimary</code> . When both <code>dprimary</code> and <code>pprimary</code> carry a "name" attribute (or are base R functions whose name can be inferred), the two names must agree on everything other than the leading d/p prefix; mismatches such as <code>dunif + pexpgrowth</code> raise an error. Supplying <code>pprimary</code> explicitly is mainly useful when using a custom primary distribution whose CDF is not in the registry.

dprimary_args	[Deprecated] Use primary_args instead.
...	Additional arguments to be passed to pdist
check	Logical; if TRUE (the default) pdist is validated with <code>check_pdist()</code> and dprimary with <code>check_dprimary()</code> . Set to FALSE to skip both when they have already been validated. <code>check_pdist()</code> evaluates pdist at four points drawn with <code>stats::runif()</code> , so skipping it avoids that cost and leaves the random number stream untouched. Must be given by its full name, as it follows ...

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,
  primary_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,
  primary_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
)
```

rdiscretehazard	<i>Sample from a hazard-parameterised step distribution</i>
-----------------	---

Description

Draws n independent samples from the discrete distribution defined by hazards. Converts hazards to a PMF via `hazards_to_pmf()` then delegates to `rdiscretetestep()`.

Usage

```
rdiscretehazard(n, boundaries = NULL, hazards)
```

Arguments

<code>n</code>	Integer. Number of samples to draw.
<code>boundaries</code>	Numeric vector of length $K + 1$ defining the bin edges. Must be strictly increasing. Defaults to $0:K$.
<code>hazards</code>	Numeric vector of length K (or $K - 1$, in which case a trailing 1 is appended) giving the discrete-time conditional hazard for each bin. Values must be in $[0, 1]$.

Details

Outside fitting it is a deterministic wrapper around `rdiscretetestep()`. It earns its keep as a fitting parameterisation in `fitdistdoublecens()` because the random walk on the logit hazard smooths the recovered PMF.

Value

Numeric vector of length n .

See Also

`rdiscretetestep()`, `hazards_to_pmf()`, `pmf_to_hazards()`, `fitdistdoublecens()`
 Other `pdiscretehazard`: `ddiscretehazard()`, `discretehazard_start()`, `pdiscretehazard()`

Examples

```
set.seed(42)
rdiscretehazard(10, boundaries = 0:3, hazards = c(0.3, 0.5, 1))
```

rdiscretetestep	<i>Sample from a step distribution</i>
-----------------	--

Description

Draws n independent samples from the discrete distribution with probability mass $\text{pmf}[i]$ at boundaries $[i+1]$.

Usage

```
rdiscretetestep(n, boundaries = NULL, pmf)
```

Arguments

<code>n</code>	Integer. Number of samples to draw.
<code>boundaries</code>	Numeric vector of length $K + 1$ defining the bin edges. Must be strictly increasing. Defaults to $0:K$ (unit-width daily bins) where K is inferred from <code>length(pmf)</code> .
<code>pmf</code>	Numeric vector of length K giving the probability mass for each bin. Must be non-negative and sum to approximately 1; if either condition is violated the function returns a vector of zeros (a soft simplex penalty for use inside optimisation).

Value

Numeric vector of length n .

See Also

Other `pdiscretetestep`: [ddiscretetestep\(\)](#), [hazards_to_pmf\(\)](#), [pdiscretetestep\(\)](#), [pmf_to_hazards\(\)](#)

Examples

```
set.seed(42)
rdiscretetestep(10, boundaries = 0:3, pmf = c(0.2, 0.5, 0.3))
```

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

<code>n</code>	Number of random samples to generate.
<code>rdist</code>	Function to generate random samples from the delay distribution for example <code>stats::rlnorm()</code> for lognormal distribution.
<code>pwindow</code>	Primary event window. Use <code>pwindow = 0</code> for an exactly observed primary event, in which case the delay CDF is used directly.
<code>swindow</code>	Integer specifying the window size for rounding the delay (default is 1). If <code>swindow = 0</code> then no rounding is applied.
<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>rprimary</code>	Function to generate random samples from the primary distribution (default is <code>stats::runif()</code>).
<code>rprimary_args</code>	List of additional arguments to be passed to <code>rprimary</code> .
<code>oversampling_factor</code>	Factor by which to oversample the number of samples to account for truncation (default is 1.2).
<code>...</code>	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 = rexpgrowth, 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)
```

update.pcons

*Update the parameters of a pcons object***Description**

Replaces the delay distribution parameters, and optionally the primary event distribution arguments, of an existing pcons object. The delay and primary distribution functions, the primary event CDF and the class are kept as they are. The updated object therefore dispatches to the same `pcons_cdf()` method without looking up names or rebuilding the class. This is cheaper than calling `new_pcons()` again when one distribution is evaluated for many parameter sets, for example posterior draws.

Usage

```
## S3 method for class 'pcons'
update(object, ..., primary_args = NULL)
```

Arguments

object	A pcons object as created by <code>new_pcons()</code> .
...	Named delay distribution parameters. Each one replaces the entry of the same name in <code>object\$args</code> , or is added if not present. Parameters that are not given keep their current values.
primary_args	Optional named list of primary event distribution arguments. These are merged into <code>object\$primary_args</code> in the same way as ... is merged into <code>object\$args</code> . Defaults to NULL, which leaves the primary event distribution arguments unchanged.

Details

Parameters are merged rather than replaced as a whole, so `update(object, scale = 3)` changes scale and keeps all other parameters. Parameters cannot be removed; use `new_pcons()` for that.

A name in ... that is not already in `object$args` must be an argument of `object$pdist` other than its first, unless `pdist` takes Otherwise an error is raised. Names in `primary_args` are not checked against `dprimary`.

Value

A pcons object with the same class as `object` and updated `args`, `primary_args` and `dprimary_args` fields. See `new_pcons()` for the fields of a pcons object.

See Also

Low level primary event censored distribution objects and methods `new_pcons()`, `pcons_cdf()`, `pcons_cdf.default()`, `pcons_cdf.pcons_pdiscretehazard()`, `pcons_cdf.pcons_pdiscretetestp()`, `pcons_cdf.pcons_pgamma_dunif()`, `pcons_cdf.pcons_pgengamma.orig_dunif()`, `pcons_cdf.pcons_pgengamma_dunif()`, `pcons_cdf.pcons_plnorm_dunif()`, `pcons_cdf.pcons_pweibull_dunif()`, `pcons_pmf()`, `pcons_pmf.default()`, `pcons_quantile()`, `pcons_quantile.default()`

Examples

```
obj <- new_pcens(  
  pdist = pgamma, dprimary = dunif,  
  primary_args = list(min = 0, max = 1),  
  shape = 1, scale = 1  
)  
obj <- update(obj, shape = 2, scale = 3)  
pcens_cdf(obj, q = c(1, 5, 10), pwindow = 1)  
  
# Update the primary event distribution arguments  
obj <- new_pcens(  
  pdist = pgamma, dprimary = dexpgrowth,  
  primary_args = list(r = 0.2),  
  shape = 2, scale = 3  
)  
obj <- update(obj, primary_args = list(r = 0.5))  
pcens_pmf(obj, x = 0:5, pwindow = 1)
```

Index

- * **check**
 - check_dprimary, [4](#)
 - check_pdist, [5](#)
 - check_truncation, [6](#)
 - * **datasets**
 - pcd_distributions, [24](#)
 - pcd_primary_distributions, [26](#)
 - * **modelhelpers**
 - fitdistdoublecens, [14](#)
 - pcd_as_stan_data, [21](#)
 - pcd_cmdstan_model, [23](#)
 - * **pcens**
 - new_pcens, [19](#)
 - pcens_cdf, [30](#)
 - pcens_cdf.default, [31](#)
 - pcens_cdf.pcens_pdiscretehazard, [32](#)
 - pcens_cdf.pcens_pdiscretetest, [33](#)
 - pcens_cdf.pcens_pgamma_dunif, [34](#)
 - pcens_cdf.pcens_pgengamma.orig_dunif, [35](#)
 - pcens_cdf.pcens_pgengamma_dunif, [36](#)
 - pcens_cdf.pcens_plnorm_dunif, [37](#)
 - pcens_cdf.pcens_pweibull_dunif, [38](#)
 - pcens_pmf, [39](#)
 - pcens_pmf.default, [40](#)
 - pcens_quantile, [41](#)
 - pcens_quantile.default, [42](#)
 - update.pcens, [56](#)
 - * **pdiscretehazard**
 - ddiscretehazard, [6](#)
 - discretehazard_start, [8](#)
 - pdiscretehazard, [43](#)
 - rdiscretehazard, [52](#)
 - * **pdiscretetest**
 - ddiscretetest, [7](#)
 - hazards_to_pmf, [18](#)
 - pdiscretetest, [44](#)
 - pmf_to_hazards, [45](#)
 - rdiscretetest, [53](#)
 - * **primarycensored**
 - dprimarycensored, [9](#)
 - pprimarycensored, [46](#)
 - qprimarycensored, [49](#)
 - rprimarycensored, [53](#)
 - * **primaryeventdistributions**
 - expgrowth, [12](#)
 - * **stantools**
 - pcd_load_stan_functions, [25](#)
 - pcd_stan_dist_id, [27](#)
 - pcd_stan_files, [28](#)
 - pcd_stan_function_deps, [29](#)
 - pcd_stan_functions, [28](#)
 - pcd_stan_path, [30](#)
 - * **utils**
 - add_name_attribute, [3](#)
 - pcd_dist_name, [25](#)
 - pcd_distributions, [24](#)
 - pcd_primary_distributions, [26](#)
- add_name_attribute, [3](#), [24](#), [25](#), [27](#)
add_name_attribute(), [4](#), [5](#), [10](#), [15](#), [19](#), [47](#), [50](#)
- check_dprimary, [4](#), [5](#), [6](#)
check_dprimary(), [10](#), [16](#), [48](#), [51](#)
check_pdist, [5](#), [5](#), [6](#)
check_pdist(), [10](#), [16](#), [48](#), [51](#)
check_truncation, [5](#), [6](#)
- ddiscretehazard, [6](#), [8](#), [44](#), [52](#)
ddiscretehazard(), [16](#)
ddiscretetest, [7](#), [19](#), [45](#), [46](#), [53](#)
ddiscretetest(), [6](#), [7](#), [16](#)
dexpgrowth (expgrowth), [12](#)
discretehazard_start, [7](#), [8](#), [44](#), [52](#)
dpcens (dprimarycensored), [9](#)
dprimarycensored, [9](#), [48](#), [51](#), [55](#)

- dprimarycensored(), 17, 39, 40
- expgrowth, 12
- fitdistdoublecens, 8, 14, 22, 24, 45
- fitdistdoublecens(), 7, 8, 44, 52
- fitdistrplus::fitdist(), 16
- fitdistrplus::fitdistcens(), 14
- hazards_to_pmf, 8, 18, 45, 46, 53
- hazards_to_pmf(), 6, 7, 32, 43–45, 52
- new_pcens, 19, 31–41, 43, 56
- new_pcens(), 11, 30, 31, 33, 35, 37–42, 48, 51, 56
- pcd_as_stan_data, 17, 21, 24
- pcd_cmdstan_model, 17, 22, 23
- pcd_cmdstan_model(), 22
- pcd_dist_name, 4, 24, 25, 27
- pcd_dist_name(), 3
- pcd_distributions, 4, 22, 24, 25, 27
- pcd_distributions(), 25, 27
- pcd_load_stan_functions, 25, 27–30
- pcd_primary_distributions, 4, 22, 24, 25, 26
- pcd_primary_distributions(), 4, 10, 15, 19, 25, 27, 47, 50
- pcd_stan_dist_id, 26, 27, 28–30
- pcd_stan_dist_id(), 22
- pcd_stan_files, 26, 27, 28, 29, 30
- pcd_stan_function_deps, 26–29, 29, 30
- pcd_stan_functions, 26–28, 28, 29, 30
- pcd_stan_path, 26–29, 30
- pcens_cdf, 20, 30, 32–41, 43, 56
- pcens_cdf(), 20, 32, 40, 41, 48, 56
- pcens_cdf.default, 20, 31, 31, 33–41, 43, 56
- pcens_cdf.default(), 36
- pcens_cdf.pcens_pdiscretehazard, 20, 31, 32, 32, 34–41, 43, 56
- pcens_cdf.pcens_pdiscretetest, 20, 31–33, 33, 35–41, 43, 56
- pcens_cdf.pcens_pgamma_dunif, 20, 31–34, 34, 36–41, 43, 56
- pcens_cdf.pcens_pgengamma.orig_dunif, 20, 31–35, 35, 37–41, 43, 56
- pcens_cdf.pcens_pgengamma.orig_dunif(), 36
- pcens_cdf.pcens_pgengamma_dunif, 20, 31–36, 36, 38–41, 43, 56
- pcens_cdf.pcens_plnorm_dunif, 20, 31–37, 37, 38–41, 43, 56
- pcens_cdf.pcens_pweibull_dunif, 20, 31–38, 38, 39–41, 43, 56
- pcens_pmf, 20, 31–38, 39, 40, 41, 43, 56
- pcens_pmf(), 11, 20
- pcens_pmf.default, 20, 31–39, 40, 41, 43, 56
- pcens_quantile, 20, 31–40, 41, 43, 56
- pcens_quantile(), 51
- pcens_quantile.default, 20, 31–41, 42, 56
- pdiscretehazard, 7, 8, 43, 52
- pdiscretehazard(), 15–17
- pdiscretetest, 8, 19, 44, 46, 53
- pdiscretetest(), 11, 16, 17, 43, 44
- pexpgrowth (expgrowth), 12
- pmf_to_hazards, 8, 19, 45, 45, 53
- pmf_to_hazards(), 7, 44, 52
- ppcens (pprimarycensored), 46
- pprimarycensored, 11, 46, 51, 55
- pprimarycensored(), 3, 11, 31, 32
- qpcens (qprimarycensored), 49
- qprimarycensored, 11, 48, 49, 55
- rdiscretehazard, 7, 8, 44, 52
- rdiscretetest, 8, 19, 45, 46, 53
- rdiscretetest(), 52
- rexpgrowth (expgrowth), 12
- rpcens (rprimarycensored), 53
- rprimarycensored, 11, 48, 51, 53
- stats::rlnorm(), 54
- stats::runif(), 10, 48, 51, 54
- stats::uniroot(), 42
- update(), 20
- update.pcens, 20, 31–41, 43, 56