Package 'iclogcondist'

July 22, 2025

Type Package

Title Log-Concave Distribution Estimation with Interval-Censored Data

Version 1.0.1

Description We consider the non-parametric maximum likelihood estimation of the underlying distribution function, assuming log-concavity, based on mixed-case interval-censored data. The algorithm implemented is base on Chi Wing Chu, Hok Kan Ling and Chaoyu Yuan (2024, <doi:10.48550/arXiv.2411.19878>).

License GPL-3

Imports Rcpp, flexsurv, ggplot2, icenReg, monotone, fdrtool

LinkingTo Rcpp, RcppArmadillo

RoxygenNote 7.3.2

Encoding UTF-8

Depends R (>= 3.5.0)

NeedsCompilation yes

Author Chi Wing Chu [aut],
Hok Kan Ling [aut],
Chaoyu Yuan [aut, cre]

Maintainer Chaoyu Yuan <chaoyu.yuan@columbia.edu>

Repository CRAN

Date/Publication 2024-12-05 19:00:06 UTC

Contents

case_II_X 2
current_status_X 3
data_prep 3
find_dir_deriv 4
find_qsi 5
get_F_at_x 5
get_F_at_x.iclogcondist 6
iclogcondist_visualization 7

icm_subset_cpp . . . 8
ic_LCMLE . . . 9
ic_LCM_UMLE . . . 10
ic_UMLE . . . 11
initial_values . . . 12
lgnm . . . 13
neg_log_like . . . 14
plot.iclogcondist . . . 15
ptllogis . . . 16
ptlnorm . . . 16
ptweibull . . . 17
qtllogis . . . 18
qtlnorm . . . 19
qtweibull . . . 19
rtllogis . . . 20
rtlnorm . . . 21
rtweibull . . . 22
simulate_ic_data . . . 22
unique_X_weight . . . 24

Index **25**

`case_II_X` *Construct Case II Interval Censoring Data*

Description

This function constructs case II interval-censored data using the provided event times and censoring (survey) times. Each individual's event time is either left-censored, right-censored, or interval-censored based on two survey times: the left and right bounds of the interval.

Usage

```
case_II_X(event_times, survey_times)
```

Arguments

`event_times`	A numeric vector of event times for each individual.
`survey_times`	A numeric matrix with two columns, where each row contains the left and right censoring (survey) times for each individual.

Value

A matrix with two columns, where each row represents an individual's interval-censored data. The first column is the left endpoint, and the second column is the right endpoint. If the event time is before the left survey time, the interval is `(0, left survey time]`. If the event time is after the right survey time, the interval is `(right survey time, Inf)`. If the event time falls between the left and right survey times, the interval is `(left survey time, right survey time]`.

current_status_X — *Construct Case I Interval Censoring Data (Current Status Data)*

Description

This function constructs case I interval-censored data (current status data) using the provided event times and censoring (survey) times. Each individual's event time is either left-censored or right-censored at their survey time, depending on whether the event has occurred by the survey time.

Usage

```
current_status_X(event_times, survey_times)
```

Arguments

`event_times`	A numeric vector of event times for each individual.
`survey_times`	A numeric vector of censoring (survey) times for each individual.

Value

A matrix with two columns, where each row represents an individual's interval-censored data. The first column is the left endpoint, and the second column is the right endpoint. If the event time is before the survey time, the interval is `(0, survey_time]`. If the event time is after the survey time, the interval is `(survey_time, Inf)`.

data_prep — *Prepare Data for Interval-Censored Model*

Description

This function processes interval-censored data and prepares various components needed for model fitting, including unique time points, censoring intervals, and weights.

Usage

```
data_prep(X)
```

Arguments

`X`	A matrix or data frame of interval-censored data where each row contains the lower and upper bounds of the interval for each observation.

Value

A list containing:

tau Unique time points.

m The number of unique time points (excluding infinity if present).

L_Rc Indices of observations where the event is in the intersection of L group and the complement of R group. The L group consists of samples with left intervals time <= min(all right intervals time). The R group consists of samples with infinity right interval time.

Lc_R Indices of observations where the event is in the intersection of the complement of L group and R group.

Lc_Rc Indices of observations where the event is in the intersection of the complement of L group and the complement of R group.

ri Indices corresponding to the right bounds of the intervals in `tau`.

li Indices corresponding to the left bounds of the intervals in `tau`.

tau_no_Inf Unique time points excluding infinity.

weight Weights for each unique interval.

X Processed matrix of interval-censored data with unique rows.

`find_dir_deriv` *Compute Directional Derivatives for Active Set Algorithm*

Description

This function computes the directional derivatives for the active set algorithm used in the estimation of the distribution function under log-concavity with interval-censored data. The calculation takes advantage of the specific structure of the basis matrix, making it efficient to compute in `O(n)` time complexity.

Usage

```
find_dir_deriv(diff_tau, first_order)
```

Arguments

`diff_tau`	A numeric vector containing the differences between consecutive time points (tau).
`first_order`	A numeric vector representing the first-order derivatives at each time point.

Value

A numeric vector of length `length(diff_tau) + 1` representing the directional derivatives for the active set algorithm.

`find_qsi` *Compute qsi Matrix*

Description

This function computes the qsi matrix for specified indices and time points.

Usage

```
find_qsi(is, tau_no_Inf)
```

Arguments

`is`	Indices of nodes.
`tau_no_Inf`	Unique time points excluding infinity.

Value

qsi matrix.

`get_F_at_x` *Generic Function to compute F at X*

Description

Computes the value of the function $F(x)$ for a given object of class `iclogcondist`. This is a generic function to compute $F(x)$ for object class `iclogcondist`. For usage details, please refer to function `get_F_at_x.iclogcondist`

Usage

```
get_F_at_x(object, ...)
```

Arguments

`object`	An object for which the method is defined.
`...`	Additional arguments passed to the method.

Value

A numeric vector of values, either $F(x)$ or $log(F(x))$.

Examples

```
# Example usage:
data(lgnm)

# Evaluate for LCMLE object
fit_LCMLE <- ic_LCMLE(lgnm)
get_F_at_x(fit_LCMLE)

# Evaluate for UMLE object
fit_UMLE <- ic_UMLE(lgnm)
x = seq(0.001, 6, length.out = 1000)
get_F_at_x(fit_UMLE, x = x)
```

`get_F_at_x.iclogcondist` *Evaluate F(x) for Objects of Class 'iclogcondist'*

Description

Computes the value of the function $F(x)$ for a given object of class `iclogcondist`.

Usage

```
## S3 method for class 'iclogcondist'
get_F_at_x(object, x = NA, log = FALSE, ...)
```

Arguments

`object`	An object of class `iclogcondist`. Must also belong to one of the subclasses: `"ic_LCMLE"`, `"ic_LCM_UMLE"`, or `"ic_UMLE"`.
`x`	A numeric vector of values at which $F(x)$ is evaluated. If not specified, the `tau_no_Inf` attribute of the `object` object is used.
`log`	Logical; if `TRUE`, returns the result in log-transformed form. Default is `FALSE`.
`...`	Additional arguments (not currently used).

Value

A numeric vector of values, either $F(x)$ or $log(F(x))$.

Examples

```
# Example usage:
data(lgnm)

# Evaluate for LCMLE object
fit_LCMLE <- ic_LCMLE(lgnm)
get_F_at_x(fit_LCMLE)
```

```
# Evaluate for UMLE object
fit_UMLE <- ic_UMLE(lgnm)
x = seq(0.001, 6, length.out = 1000)
get_F_at_x(fit_UMLE, x = x)
```

`iclogcondist_visualization` *Visualize the Estimated Cumulative Distribution Functions*

Description

This function visualizes a user-specified distribution `true_dist` (if available) and the estimated cumulative distribution functions (CDF) $F(t)$ and $logF(t)$ for a given range. The function overlays the estimated functions from a list of fitted models on the same plot, allowing comparison with the user-specified distribution (if provided). In a simulation study, the user-specified distribution can correspond to the true underlying distribution.

Usage

```
iclogcondist_visualization(X, range = NA, fit_list = list(), true_dist = NA)
```

Arguments

`X`	A dataset or input data used to prepare the plot range if `range` is not specified.
`range`	A numeric vector of length 2 specifying the range of `t` values for plotting. If `NA` the function calculates the range based on the input data `X`.
`fit_list`	A named list of fitted models, where each element is expected to contain an `est` object with estimates for generating the CDF plots. The name of the list should be `"LCMLE"`,`"UMLE"` or `"LCM_UMLE"`
`true_dist`	Optional. A data frame or list containing the user-specified distribution values, with components `x` and `y` representing the values of t and $F(t)$ respectively.

Value

A list containing two ggplot objects: `logF_plot` for $logF(t)$ and `F_plot` for $F(t)$.

Examples

```
# Example usage
data(lgnm)
fit_LCMLE <- ic_LCMLE(lgnm)
fit_UMLE <- ic_UMLE(lgnm)
iclogcondist_visualization(
  X = lgnm,
  range = c(0, 10),
  fit_list = list(
    "UMLE" = fit_UMLE,
    "LCMLE" = fit_LCMLE
```

```
  )
)
```

`icm_subset_cpp` *Iterative Convex Minorant (ICM) Subset Algorithm*

Description

This function implements the ICM algorithm for solving the sub-problem in the active set algorithm. This is a support of the active set algorithm, computing the optimal values `phi_tilde` with reduced number of knots in the sub-problem. It uses backtracking to ensure convergence (Jongbloed, 1998).

Usage

```
icm_subset_cpp(
  phi_tilde_initial,
  is,
  tau_no_Inf,
  L_Rc,
  Lc_R,
  Lc_Rc,
  ri,
  li,
  weight,
  tol = 1e-10,
  max_iter = 500
)
```

Arguments

`phi_tilde_initial`
: A numeric vector representing the initial values of the reduced variables `phi_tilde`.

`is`
: A numeric vector indicating the nodes with unequal left-hand slope and right-hand slope.

`tau_no_Inf`
: A numeric vector containing the unique time points, excluding infinity.

`L_Rc`
: Indices of observations where the event is in the intersection of L group and the complement of R group. The L group consists of samples with left intervals time <= min(all right intervals time). The R group consists of samples with infinity right interval time.

`Lc_R`
: Indices of observations where the event is in the intersection of the complement of L group and R group.

`Lc_Rc`
: Indices of observations where the event is in the intersection of the complement of L group and the complement of R group.

`ri`
: A numeric vector of indices corresponding to the right bounds of the intervals in `tau_no_Inf`.

li	A numeric vector of indices corresponding to the left bounds of the intervals in tau_no_Inf.
weight	A numeric vector representing the weights for each observation.
tol	A numeric value specifying the tolerance for convergence. Default is 1e-10.
max_iter	An integer specifying the maximum number of iterations. Default is 500.

Value

A list containing:

phi_tilde_hat The estimated values of the reduced variable phi_tilde at the end of the ICM iterations.

References

Jongbloed, G.: The iterative convex minorant algorithm for nonparametric estimation. J. Comput. Gr. Stat. 7(3), 310–321 (1998)

---

ic_LCMLE *Compute the log-concave MLE for Interval-censored Data using an Active Set Algorithm*

Description

This function computes the log-concave MLE of the cumulative distribution function for interval-censored data under log-concavity on the underlying distribution function based on an active set algorithm. The active set algorithm adjusts the knots set based on certain directional derivatives.

Usage

```
ic_LCMLE(
  X,
  initial = "LCM",
  print = FALSE,
  max_iter = 500,
  tol_conv = 1e-07,
  tol_conv_like = 1e-10,
  tol_K = 1e-05
)
```

Arguments

X	A matrix with two columns, where each row represents an interval (L, R] for interval-censored data. L and R are the left and right endpoints, with R = Inf indicating right-censoring.
initial	A character string specifying the method of obtaining an initial value ("LCM" or "MLE") for the estimation process. Default is "LCM".

`print`	Logical. If `TRUE`, prints the iterative process details. Default is `FALSE`.
`max_iter`	An integer specifying the maximum number of iterations for the algorithm. Default is 500.
`tol_conv`	A numeric tolerance level for convergence based on the directional derivatives. Default is `1e-7`.
`tol_conv_like`	A numeric tolerance level for convergence based on log-likelihood difference. Default is `1e-10`.
`tol_K`	A numeric tolerance for checking if v^T phi is in K (constraint set) in active constraint set `A`. Default is `1e-5`.

Value

A list with the following components:

`est`	A list containing `tau_no_Inf` (unique finite `tau` values), `phi_hat` (estimate of `log F`), and `F_hat` (estimate of `F`).
`knot_info`	A list with `knot_index` (indices of active knots in `tau_no_Inf`), `tau_on_knot` (tau values at active knots), `F_on_knot` (MLE cumulative distribution function values at active knots), and `phi_on_knot` (logarithmic estimates of `F` at active knots).
`neg_log_likelihood`	Vector of negative log-likelihood values for each iteration of the algorithm.
`dir_derivs`	Vector of directional derivatives for each iteration.
`iter_no`	Integer representing the total number of iterations.
`weight`	Vector of weights corresponding to each interval in the data.
`X`	The original interval-censored data matrix input.

Examples

```
# Example usage:
data(lgnm)
result <- ic_LCMLE(X = lgnm, initial = "LCM", print = TRUE, max_iter = 500)
print(result$est)
```

`ic_LCM_UMLE` *Compute Least Concave Majorant (LCM) of the log of the Unconstrained MLE for Interval-Censored Data*

Description

This function computes the Least Concave Majorant (LCM) of the log of the unconstrained MLE for interval-censored data.

Usage

```
ic_LCM_UMLE(X)
```

Arguments

X	A matrix with two columns, where each row represents an interval (L, R] for interval-censored data. L and R are the left and right endpoints, respectively, with R = Inf indicating right-censoring.

Value

A list containing:

est	A list with tau_no_Inf (finite values of tau), phi_hat (LCM of log of F_hat), and F_hat (exp of phi_hat).
knot_info	A list with knot_index (indices of knots in tau_no_Inf), tau_on_knot (values of tau at knots), F_on_knot (F_hat at knots), and phi_on_knot (phi_hat at knots).
neg_log_likelihood	The negative log-likelihood of the LCM fit.
weight	Vector of weights corresponding to each interval in the data.
X	The original interval-censored data matrix input.

Examples

```
data(lgnm)
result <- ic_LCM_UMLE(X = lgnm)
```

---

ic_UMLE — *Compute Unconstrained Maximum Likelihood Estimate for Interval-Censored Data*

Description

This function computes the unconstrained maximum likelihood estimate (UMLE) for interval-censored data. It utilizes the non-parametric MLE from the ic_np function in the icenReg package as a starting point and prepares key components such as cumulative probabilities, log-transformed values, and knot information.

Usage

```
ic_UMLE(X)
```

Arguments

X	A matrix with two columns, where each row represents an interval (L, R] for interval-censored data. L and R are left and right endpoints, respectively, with R = Inf indicating right-censoring.

Details

The ic_np function from the icenReg package is used to compute the non-parametric MLE for interval-censored data. This provides initial estimates of probabilities (p_hat) and jump points (knot) in the cumulative distribution function. These are then processed to compute the cumulative probabilities (F_hat) and log-transformed values (phi_hat) at unique time points.

Value

A list containing:

- `est` — A list with tau_no_Inf (finite values of tau), phi_hat (log of F_hat values), and F_hat (MLE cumulative distribution function values).
- `knot_info` — A list with knot_index (indices of knots in tau_no_Inf), tau_on_knot (values of tau at knots), F_on_knot (MLE at knots), and phi_on_knot (log of F at knots).
- `neg_log_likelihood` — The negative log-likelihood of the MLE fit.
- `weight` — Vector of weights corresponding to each interval in the data.
- `X` — The original interval-censored data matrix input.

References

Anderson-Bergman, C. (2016) An efficient implementation of the EMICM algorithm for the interval censored NPMLE *Journal of Computational and Graphical Statistics*.

Examples

```
data(lgnm)
result <- ic_UMLE(X = lgnm)
```

---

initial_values — *Initial Values for Estimation under Log-concavity with Interval-Censored Data*

Description

This function obtains initial values for the maximum likelihood estimation under log-concavity with interval-censored data based on the unconstrained maximum likelihood estimate (MLE) or its least concave majorant (LCM). Alternatively, the user can provide a numeric vector of initial values.

Usage

```
initial_values(X, initial = "LCM")
```

Arguments

`X` A matrix or data frame of interval-censored data, where each row contains the lower and upper bounds of the interval for each observation.

`initial` A character string specifying the method for generating initial values. The default is "LCM", which uses the least concave majorant of the log of the unconstrained MLE. Other options are "MLE" for the unconstrained maximum likelihood estimate or a numeric vector provided by the user.

Value

A list containing:

phi_hat The initial values of the `phi` parameter based on the specified method.

phi_hat_MLE Initial values based on the unconstrained MLE.

phi_hat_LCM Initial values based on the least concave majorant.

`lgnm` *LGNM Data: Case II Interval Censoring Example*

Description

This dataset, `lgnm`, provides an example of case II interval censoring data for illustrating the functions in this package. The event time is simulated from a log-normal distribution with parameters mean = 0 and standard deviation = 1. The left censoring time is drawn from a uniform distribution between 0 and 2, and the right censoring time is drawn from a uniform distribution between the left censoring time and 20. Both the left and right censoring times are rounded to four decimal places.

Format

A data frame with 100 observations on the following 2 variables:

left The left censoring time.

right The right censoring time.

Source

Synthetic data generated for illustration purposes.

Examples

```
data(lgnm)
head(lgnm)
```

`neg_log_like` *Compute the Negative Log-Likelihood for the Interval-Censored Model*

Description

This function computes the negative log-likelihood of an interval-censored model based on the specified parameterization.

Usage

```
neg_log_like(x, weight, li, ri, L_Rc, Lc_R, Lc_Rc, type = "", tau_no_Inf)
```

Arguments

x	A numeric vector of parameter estimates (can be in terms of phi, or F).
weight	A numeric vector of weights for the observations.
li	A numeric vector of indices corresponding to the left bounds of the intervals in tau_no_Inf.
ri	A numeric vector of indices corresponding to the right bounds of the intervals in tau_no_Inf.
L_Rc	Indices of observations where the event is in the intersection of L group and the complement of R group. The L group consists of samples with left intervals time <= min(all right intervals time). The R group consists of samples with infinity right interval time.
Lc_R	Indices of observations where the event is in the intersection of the complement of L group and R group.
Lc_Rc	Indices of observations where the event is in the intersection of the complement of L group and the complement of R group.
type	A character string indicating the parameterization of x. Options are "phi" (log of F), or "F".
tau_no_Inf	A numeric vector of unique time points excluding infinity.

Value

The negative log-likelihood value.

plot.iclogcondist *Plot Method for iclogcondist_plot Objects*

Description

This function generates a plot for objects of class `iclogcondist`, which are typically generated by `ic_UMLE`, `ic_LCM_UMLE`, or `ic_LCMLE`. The plot can display either the cumulative distribution function `F(t)` or the log cumulative distribution function `logF(t)`, depending on the setting of the `log` parameter.

Usage

```
## S3 method for class 'iclogcondist'
plot(x, log = FALSE, ...)
```

Arguments

`x`	An object of class `iclogcondist`, typically generated by `ic_UMLE`, `ic_LCM_UMLE`, or `ic_LCMLE`.
`log`	Logical; if `TRUE`, plots the log cumulative distribution function `logF(t)`. If `FALSE`, plots `F(t)`. Default is `FALSE`.
`...`	Additional arguments passed to the plotting function.

Value

An invisible `ggplot` object representing the plot. The plot is also displayed in the current graphics device.

Examples

```
# Example usage with ic_UMLE, ic_LCM_UMLE, and ic_LCMLE
data(lgnm)
X <- lgnm
fit_UMLE <- ic_UMLE(X)
fit_LCM_UMLE <- ic_LCM_UMLE(X)
fit_LCMLE <- ic_LCMLE(X)
plot(fit_UMLE, log = TRUE)   # Plot logF(t) for UMLE
plot(fit_LCM_UMLE, log = FALSE)  # Plot F(t) for LCM_UMLE
plot(fit_LCMLE, log = FALSE) # Plot F(t) for LCMLE
```

ptllogis — *Cumulative Distribution Function of a Truncated Log-Logistic Distribution*

Description

This function computes the cumulative distribution function (CDF) of a truncated log-logistic distribution at a given point x.

Usage

```
ptllogis(x, shape = 1, scale = 1, upper_bound = Inf)
```

Arguments

x	A numeric vector at which to evaluate the CDF.
shape	A positive numeric value representing the shape parameter of the log-logistic distribution. Default is 1.
scale	A positive numeric value representing the scale parameter of the log-logistic distribution. Default is 1.
upper_bound	A positive numeric value indicating the upper truncation point. Default is Inf (no truncation).

Value

A numeric vector of the CDF values of the truncated log-logistic distribution at x.

Examples

```
# Evaluate the CDF at x = 2 for a truncated log-logistic distribution
ptllogis(2, shape = 2, scale = 1, upper_bound = 5)
```

ptlnorm — *Cumulative Distribution Function of a Truncated Log-Normal Distribution*

Description

This function computes the cumulative distribution function (CDF) of a truncated log-normal distribution at a given point x.

Usage

```
ptlnorm(x, meanlog = 0, sdlog = 1, upper_bound = Inf)
```

Arguments

x	A numeric vector at which to evaluate the CDF.
meanlog	A numeric value representing the mean of the log-normal distribution on the log scale. Default is 0.
sdlog	A positive numeric value representing the standard deviation of the log-normal distribution on the log scale. Default is 1.
upper_bound	A positive numeric value indicating the upper truncation point. Default is Inf (no truncation).

Value

A numeric vector of the CDF values of the truncated log-normal distribution at x.

Examples

```
# Evaluate the CDF at x = 2 for a truncated log-normal distribution
ptlnorm(2, meanlog = 0, sdlog = 1, upper_bound = 5)
```

---

ptweibull — *Cumulative Distribution Function of a Truncated Weibull Distribution*

Description

This function computes the cumulative distribution function (CDF) of a truncated Weibull distribution at a given point x.

Usage

```
ptweibull(x, shape = 1, scale = 1, upper_bound = Inf)
```

Arguments

x	A numeric vector at which to evaluate the CDF.
shape	A positive numeric value representing the shape parameter of the Weibull distribution. Default is 1.
scale	A positive numeric value representing the scale parameter of the Weibull distribution. Default is 1.
upper_bound	A positive numeric value indicating the upper truncation point. Default is Inf (no truncation).

Value

A numeric vector of the CDF values of the truncated Weibull distribution at x.

Examples

```
# Evaluate the CDF at x = 2 for a truncated Weibull distribution
ptweibull(2, shape = 2, scale = 1, upper_bound = 5)
```

---

qtllogis *Quantile Function of a Truncated Log-Logistic Distribution*

Description

This function computes the quantiles of a truncated log-logistic distribution for a given probability vector.

Usage

```
qtllogis(q, shape = 1, scale = 1, upper_bound = Inf)
```

Arguments

q	A numeric vector of probabilities for which to calculate the quantiles.
shape	A positive numeric value representing the shape parameter of the log-logistic distribution. Default is 1.
scale	A positive numeric value representing the scale parameter of the log-logistic distribution. Default is 1.
upper_bound	A positive numeric value indicating the upper truncation point. Default is Inf (no truncation).

Value

A numeric vector of quantiles corresponding to the given probabilities in q.

Examples

```
# Calculate the 0.5 quantile of a truncated log-logistic distribution
qtllogis(0.5, shape = 2, scale = 1, upper_bound = 5)
```

---

qtlnorm *Quantile Function of a Truncated Log-Normal Distribution*

Description

This function computes the quantiles of a truncated log-normal distribution for a given probability vector.

Usage

```
qtlnorm(q, meanlog = 0, sdlog = 1, upper_bound = Inf)
```

Arguments

q	A numeric vector of probabilities for which to calculate the quantiles.
meanlog	A numeric value representing the mean of the log-normal distribution on the log scale. Default is 0.
sdlog	A positive numeric value representing the standard deviation of the log-normal distribution on the log scale. Default is 1.
upper_bound	A positive numeric value indicating the upper truncation point. Default is Inf (no truncation).

Value

A numeric vector of quantiles corresponding to the given probabilities in q.

Examples

```
# Calculate the 0.5 quantile of a truncated log-normal distribution
qtlnorm(0.5, meanlog = 0, sdlog = 1, upper_bound = 5)
```

---

qtweibull *Quantile Function of a Truncated Weibull Distribution*

Description

This function computes the quantiles of a truncated Weibull distribution for a given probability vector.

Usage

```
qtweibull(q, shape = 1, scale = 1, upper_bound = Inf)
```

Arguments

`q`	A numeric vector of probabilities for which to calculate the quantiles.
`shape`	A positive numeric value representing the shape parameter of the Weibull distribution. Default is `1`.
`scale`	A positive numeric value representing the scale parameter of the Weibull distribution. Default is `1`.
`upper_bound`	A positive numeric value indicating the upper truncation point. Default is `Inf` (no truncation).

Value

A numeric vector of quantiles corresponding to the given probabilities in `q`.

Examples

```
# Calculate the 0.5 quantile of a truncated Weibull distribution
qtweibull(0.5, shape = 2, scale = 1, upper_bound = 5)
```

---

rtllogis *Simulate from a Truncated Log-Logistic Distribution*

Description

This function generates random samples from a truncated log-logistic distribution using an acceptance-rejection method.

Usage

```
rtllogis(n, shape = 1, scale = 1, upper_bound = Inf)
```

Arguments

`n`	An integer specifying the number of random samples to generate.
`shape`	A positive numeric value representing the shape parameter of the log-logistic distribution. Default is `1`.
`scale`	A positive numeric value representing the scale parameter of the log-logistic distribution. Default is `1`.
`upper_bound`	A positive numeric value indicating the upper truncation point. Default is `Inf` (no truncation).

Value

A numeric vector of `n` random samples from the truncated log-logistic distribution.

Examples

```
# Generate 10 random samples from a truncated log-logistic distribution
rtllogis(10, shape = 2, scale = 1, upper_bound = 5)
```

---

rtlnorm *Simulate from a Truncated Log-Normal Distribution*

Description

This function generates random samples from a truncated log-normal distribution using an acceptance-rejection method.

Usage

```
rtlnorm(n, meanlog = 0, sdlog = 1, upper_bound = Inf)
```

Arguments

`n`	An integer specifying the number of random samples to generate.
`meanlog`	A numeric value representing the mean of the log-normal distribution on the log scale. Default is `0`.
`sdlog`	A positive numeric value representing the standard deviation of the log-normal distribution on the log scale. Default is `1`.
`upper_bound`	A positive numeric value indicating the upper truncation point. Default is `Inf` (no truncation).

Value

A numeric vector of `n` random samples from the truncated log-normal distribution.

Examples

```
# Generate 10 random samples from a truncated log-normal distribution
rtlnorm(10, meanlog = 0, sdlog = 1, upper_bound = 5)
```

rtweibull *Simulate from a Truncated Weibull Distribution*

Description

This function generates random samples from a truncated Weibull distribution using inverse transform sampling. When shape = 1, it reduces to a truncated exponential distribution.

Usage

```
rtweibull(n, shape = 1, scale = 1, upper_bound = Inf)
```

Arguments

n	An integer specifying the number of random samples to generate.
shape	A positive numeric value representing the shape parameter of the Weibull distribution. Default is 1.
scale	A positive numeric value representing the scale parameter of the Weibull distribution. Default is 1.
upper_bound	A positive numeric value indicating the upper truncation point. Default is Inf (no truncation).

Value

A numeric vector of n random samples from the truncated Weibull distribution.

Examples

```
# Generate 10 random samples from a truncated Weibull distribution
rtweibull(10, shape = 2, scale = 1, upper_bound = 5)
```

simulate_ic_data *Simulate Interval-Censored Data*

Description

This function generates interval-censored data, where the event times are generated from one of the following distributions: Weibull, log-normal and log-logistic. It supports both case 1 and case 2 interval censoring.

Usage

```
simulate_ic_data(
  n,
  dist,
  para1,
  para2,
  upper_bound = Inf,
  C1_upper = 1,
  case = 2,
  rounding = FALSE,
  round_digit = 4
)
```

Arguments

`n`	An integer specifying the number of observations to generate.
`dist`	A character string indicating the distribution to use for event times. Options are `"lognormal"`, `"weibull"`, or `"loglogistic"`.
`para1`	A numeric value representing the first parameter of the distribution: • `"lognormal"`: Mean of the log-normal distribution (meanlog). • `"weibull"` and `"loglogistic"`: Shape parameter.
`para2`	A numeric value representing the second parameter of the distribution: • `"lognormal"`: Standard deviation of the log-normal distribution (sdlog). • `"weibull"` and `"loglogistic"`: Scale parameter.
`upper_bound`	A numeric value specifying the upper bound for event times, corresponding to a truncated distribution. Default is `Inf`.
`C1_upper`	A numeric value specifying the upper limit for the first censoring time `C1`. Default is 1.
`case`	An integer specifying the censoring case to simulate: • `1`: Current status (case 1 interval censoring) • `2`: Case 2 Interval censoring
`rounding`	A logical value. If `TRUE`, generated times are rounded to a specified number of decimal places. Default is `FALSE`.
`round_digit`	An integer specifying the number of digits for rounding when `rounding = TRUE`. Default is 4.

Details

- **Censoring Times**:
  - In `case = 1` (current status), one censoring time is generated, where it follows `U(0, C1_upper)`.
  - In `case = 2` (case 2 interval censoring), two censoring times are generated:
    - `C1`: sampled from `U(0, C1_upper)`.
    - `C2`: sampled from `U(C1, min(upper_bound, 20))`.

- **Distributions**:
  - **Weibull**: Parameterized by shape (para1) and scale (para2).
  - **Log-logistic**: Parameterized by shape (para1) and scale (para2).
  - **Log-normal**: Parameterized by mean (para1) and standard deviation (para2).

Value

A matrix of interval-censored data where each row represents an interval (L, R] containing the unobserved event time.

Examples

```
# Simulate data with a truncated Weibull distribution and case II interval censoring
simulate_ic_data(n = 100, dist = "weibull", para1 = 2, para2 = 1, upper_bound = 5, case = 2)
```

---

unique_X_weight *Find Unique Rows in a Matrix and Their Weights*

Description

This function finds the unique rows of a given matrix and calculates the frequency (weight) of each unique row. It returns both the unique rows and the weights (the number of occurrences of each row).

Usage

```
unique_X_weight(X)
```

Arguments

X	A matrix. The matrix whose unique rows are to be found.

Value

A list containing two components:

unique_X A matrix of the unique rows from the input matrix.

weight An integer vector containing the frequency (weight) of each unique row.

Index

∗ **datasets**
lgnm, 13

case_II_X, 2
current_status_X, 3

data_prep, 3

find_dir_deriv, 4
find_qsi, 5

get_F_at_x, 5
get_F_at_x.iclogcondist, 6

ic_LCM_UMLE, 10
ic_LCMLE, 9
ic_UMLE, 11
iclogcondist_visualization, 7
icm_subset_cpp, 8
initial_values, 12

lgnm, 13

neg_log_like, 14

plot.iclogcondist, 15
ptllogis, 16
ptlnorm, 16
ptweibull, 17

qtllogis, 18
qtlnorm, 19
qtweibull, 19

rtllogis, 20
rtlnorm, 21
rtweibull, 22

simulate_ic_data, 22

unique_X_weight, 24