Package 'weibulltools'

January 12, 2021

Type Package

Title Statistical Methods for Life Data Analysis

Version 2.0.0

Description Provides statistical methods and visualizations that are often used in reliability engineering. Comprises a compact and easily accessible set of methods and visualization tools that make the examination and adjustment as well as the analysis and interpretation of field data (and bench tests) as simple as possible.
Non-parametric estimators like Median Ranks, Kaplan-Meier (Abernethy, 2006, <ISBN:978-0-9653062-3-2>), Johnson (Johnson, 1964, <ISBN:978-0444403223>), and Nelson-Aalen for failure probability estimation within samples that contain failures as well as censored data are included.
The package supports methods like Maximum Likelihood and Rank Regression, (Genschel and Meeker, 2010, <DOI:10.1080/08982112.2010.503447>) for the estimation of multiple parametric lifetime distributions, as well as the computation of confidence intervals of quantiles and probabilities using the delta method related to Fisher's confidence intervals (Meeker and Escobar, 1998, <ISBN:9780471673279>) and the beta-binomial confidence bounds.
If desired, mixture model analysis can be done with segmented regression and the EM algorithm.
Besides the well-known Weibull analysis, the package also contains Monte Carlo methods for the correction and completion of imprecisely recorded or unknown lifetime characteristics.
(Verband der Automobilindustrie e.V. (VDA), 2016, <ISSN:0943-9412>).
Plots are created statically ('ggplot2') or interactively ('plotly') and can be customized with functions of the respective visualization package.
The graphical technique of probability plotting as well as the addition of regression lines and confidence bounds to existing plots are supported.

License GPL-2

URL https://tim-tu.github.io/weibulltools, https://github.com/Tim-TU/weibulltools

BugReports https://github.com/Tim-TU/weibulltools/issues

Imports dplyr, ggplot2, lifecycle, magrittr, plotly, purrr, Rcpp, sandwich, segmented, SPREDA, survival, tibble, tidyr

LinkingTo Rcpp (>= 0.12.18), RcppArmadillo

Depends R (>= 3.5.0)

Language en-US

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

Suggests knitr, rmarkdown, testthat

VignetteBuilder knitr

Config/testthat/edition 3

RdMacros lifecycle

NeedsCompilation yes

Author Tim-Gunnar Hensel [aut, cre], David Barkemeyer [aut]

Maintainer Tim-Gunnar Hensel <tim-gunnar.hensel@tu-berlin.de>

Repository CRAN

Date/Publication 2021-01-12 09:30:02 UTC

R topics documented:

`weibulltools-package` *weibulltools*

Description

Provides statistical methods and visualizations that are often used in reliability engineering. Comprises a compact and easily accessible set of methods and visualization tools that make the examination and adjustment as well as the analysis and interpretation of field data (and bench tests) as simple as possible.

Besides the well-known Weibull analysis, the package supports multiple lifetime distributions and also contains Monte Carlo methods for the correction and completion of imprecisely recorded or unknown lifetime characteristics.

Plots are created statically (`ggplot2`) or interactively (`plotly`) and can be customized with functions of the respective visualization package.

alloy *Fatigue Life for Alloy T7989 Specimens*

Description

A dataset containing the number of cycles of fatigue life for Alloy T7987 specimens.

Usage

```
alloy
```

Format

A tibble with 72 rows and 2 variables:

cycles Number of cycles (in thousands).

status If specimen failed before 300 thousand cycles `1` else `0`.

Source

Meeker, William Q; Escobar, Luis A., Statistical Methods for Reliability Data, New York: Wiley series in probability and statistics (1998, p.131)

confint_betabinom *Beta Binomial Confidence Bounds for Quantiles and Probabilities*

Description

This non-parametric approach computes the beta binomial bounds (BB) for quantiles and failure probabilities using a procedure similar to the calculation of probabilities in terms of (*Median Ranks*).

Usage

```
confint_betabinom(x, ...)

## S3 method for class 'wt_model'
confint_betabinom(
  x,
  b_lives = c(0.01, 0.1, 0.5),
  bounds = c("two_sided", "lower", "upper"),
  conf_level = 0.95,
  direction = c("y", "x"),
  ...
)
```

Arguments

x	Object with class wt_model and one of the classes wt_model_estimation or wt_model_estimation_list returned from rank_regression.
...	Further arguments passed to or from other methods. Currently not used.
b_lives	A numeric vector indicating the probabilities p of the B_p-lives (quantiles) to be considered.
bounds	A character string specifying of which bounds have to be computed. One of "two_sided", "lower" or "upper".
conf_level	Confidence level of the interval.
direction	A character string specifying the direction of the confidence interval. One of "y" (failure probabilities) or "x" (quantiles).

Details

The difference to *Median Ranks*, i.e. finding the probability of rank *j* at a 50% level, is to determine the probability of rank *j* on another level, the specified confidence level.

Value

A tibble with class wt_confint containing the following columns:

- x : An ordered sequence of the lifetime characteristic regarding the failed units, starting at min(x) and ending up at max(x). With b_lives = c(0.01,0.1,0.5) the 1%, 10% and 50% quantiles are additionally included in x, but only if the specified probabilities are in the range of the estimated probabilities.
- rank : Interpolated ranks as a function of probabilities, computed with the converted approximation formula of Benard.
- prob : An ordered sequence of probabilities with specified b_lives included.
- lower_bound : Provided, if bounds is one of "two_sided" or "lower". Lower limit of the confidence region with respect to direction, i.e. quantiles or probabilities.
- upper_bound : Provided, if bounds is one of "two_sided" or "upper". Upper limit of the confidence region with respect to direction, i.e. quantiles or probabilities.
- distribution : Specified distribution (determined when calling rank_regression).
- bounds : Specified bound(s).
- direction : Specified direction.
- cdf_estimation_method : Specified method for the estimation of failure probabilities (determined when calling estimate_cdf).

Examples

```
# Reliability data preparation:
## Data for two-parametric model:
data_2p <- reliability_data(
  shock,
  x = distance,
```

```
  status = status
)

## Data for three-parametric model:
data_3p <- reliability_data(
  alloy,
  x = cycles,
  status = status
)

# Probability estimation:
prob_tbl_2p <- estimate_cdf(
  data_2p,
  methods = "johnson"
)

prob_tbl_3p <- estimate_cdf(
  data_3p,
  methods = "johnson"
)

prob_tbl_mult <- estimate_cdf(
  data_3p,
  methods = c("johnson", "mr")
)

# Model estimation with rank_regression():
rr_2p <- rank_regression(
  prob_tbl_2p,
  distribution = "weibull"
)

rr_3p <- rank_regression(
  prob_tbl_3p,
  distribution = "lognormal3",
  conf_level = 0.90
)

rr_lists <- rank_regression(
  prob_tbl_mult,
  distribution = "loglogistic3",
  conf_level = 0.90
)

# Example 1 - Two-sided 95% confidence interval for probabilities ('y'):
conf_betabin_1 <- confint_betabinom(
  x = rr_2p,
  bounds = "two_sided",
  conf_level = 0.95,
  direction = "y"
)

# Example 2 - One-sided lower/upper 90% confidence interval for quantiles ('x'):
```

```
conf_betabin_2_1 <- confint_betabinom(
  x = rr_2p,
  bounds = "lower",
  conf_level = 0.90,
  direction = "x"
)

conf_betabin_2_2 <- confint_betabinom(
  x = rr_2p,
  bounds = "upper",
  conf_level = 0.90,
  direction = "x"
)

# Example 3 - Two-sided 90% confidence intervals for both directions using
# a three-parametric model:
conf_betabin_3_1 <- confint_betabinom(
  x = rr_3p,
  bounds = "two_sided",
  conf_level = 0.90,
  direction = "y"
)

conf_betabin_3_2 <- confint_betabinom(
  x = rr_3p,
  bounds = "two_sided",
  conf_level = 0.90,
  direction = "x"
)

# Example 4 - Confidence intervals if multiple methods in estimate_cdf, i.e.
# "johnson" and "mr", were specified:

conf_betabin_4 <- confint_betabinom(
  x = rr_lists,
  bounds = "two_sided",
  conf_level = 0.99,
  direction = "y"
)
```

`confint_betabinom.default` *Beta Binomial Confidence Bounds for Quantiles and Probabilities*

Description

This non-parametric approach computes the beta binomial bounds (BB) for quantiles and failure probabilities using a procedure similar to the calculation of probabilities in terms of (*Median Ranks*).

Usage

```
## Default S3 method:
confint_betabinom(
  x,
  status,
  dist_params,
 distribution = c("weibull", "lognormal", "loglogistic", "normal", "logistic", "sev",
    "weibull3", "lognormal3", "loglogistic3"),
  b_lives = c(0.01, 0.1, 0.5),
  bounds = c("two_sided", "lower", "upper"),
  conf_level = 0.95,
  direction = c("y", "x"),
  ...
)
```

Arguments

x	A numeric vector which consists of lifetime data. Lifetime data could be every characteristic influencing the reliability of a product, e.g. operating time (days/months in service), mileage (km, miles), load cycles.
status	A vector of binary data (0 or 1) indicating whether unit i is a right censored observation (= 0) or a failure (= 1).
dist_params	A (named) numeric vector of (log-)location-scale parameters returned from rank_regression.
distribution	Supposed distribution of the random variable. Has to be in line with the specification made in rank_regression.
b_lives	A numeric vector indicating the probabilities p of the B_p-lives (quantiles) to be considered.
bounds	A character string specifying of which bounds have to be computed. One of "two_sided", "lower" or "upper".
conf_level	Confidence level of the interval.
direction	A character string specifying the direction of the confidence interval. One of "y" (failure probabilities) or "x" (quantiles).
...	Further arguments passed to or from other methods. Currently not used.

Details

The difference to *Median Ranks*, i.e. finding the probability of rank j at a 50% level, is to determine the probability of rank j on another level, the specified confidence level.

Value

A tibble with class wt_confint containing the following columns:

- x : An ordered sequence of the lifetime characteristic regarding the failed units, starting at min(x) and ending up at max(x). With b_lives = c(0.01,0.1,0.5) the 1%, 10% and 50% quantiles are additionally included in x, but only if the specified probabilities are in the range of the estimated probabilities.

- rank : Interpolated ranks as a function of probabilities, computed with the converted approximation formula of Benard.
- prob : An ordered sequence of probabilities with specified b_lives included.
- lower_bound : Provided, if bounds is one of "two_sided" or "lower". Lower limit of the confidence region with respect to direction, i.e. quantiles or probabilities.
- upper_bound : Provided, if bounds is one of "two_sided" or "upper". Upper limit of the confidence region with respect to direction, i.e. quantiles or probabilities.
- distribution : Specified distribution.
- bounds : Specified bound(s).
- direction : Specified direction.
- cdf_estimation_method : A character that is always NA_character. Due to the generic visualization functions this column has to be provided.

See Also

confint_betabinom

Examples

```
# Vectors:
obs <- seq(10000, 100000, 10000)
status_1 <- c(0, 1, 1, 0, 0, 0, 1, 0, 1, 0)

cycles <- alloy$cycles
status_2 <- alloy$status

# Probability estimation:
prob_tbl <- estimate_cdf(
  x = obs,
  status = status_1,
  method = "johnson"
)

prob_tbl_2 <- estimate_cdf(
  x = cycles,
  status = status_2,
  method = "johnson"
)

# Model estimation with rank_regression():
rr <- rank_regression(
  x = prob_tbl$x,
  y = prob_tbl$prob,
  status = prob_tbl$status,
  distribution = "weibull",
  conf_level = 0.9
)

rr_2 <- rank_regression(
```

```
  x = prob_tbl_2$x,
  y = prob_tbl_2$prob,
  status = prob_tbl_2$status,
  distribution = "lognormal3"
)

# Example 1 - Two-sided 95% confidence interval for probabilities ('y'):
conf_betabin_1 <- confint_betabinom(
  x = prob_tbl$x,
  status = prob_tbl$status,
  dist_params = rr$coefficients,
  distribution = "weibull",
  bounds = "two_sided",
  conf_level = 0.95,
  direction = "y"
)

# Example 2 - One-sided lower/upper 90% confidence interval for quantiles ('x'):
conf_betabin_2_1 <- confint_betabinom(
  x = prob_tbl$x,
  status = prob_tbl$status,
  dist_params = rr$coefficients,
  distribution = "weibull",
  bounds = "lower",
  conf_level = 0.9,
  direction = "x"
)

conf_betabin_2_2 <- confint_betabinom(
  x = prob_tbl$x,
  status = prob_tbl$status,
  dist_params = rr$coefficients,
  distribution = "weibull",
  bounds = "upper",
  conf_level = 0.9,
  direction = "x"
)

# Example 3 - Two-sided 90% confidence intervals for both directions using
# a three-parametric model:

conf_betabin_3_1 <- confint_betabinom(
  x = prob_tbl_2$x,
  status = prob_tbl_2$status,
  dist_params = rr_2$coefficients,
  distribution = "lognormal3",
  bounds = "two_sided",
  conf_level = 0.9,
  direction = "y"
)

conf_betabin_3_2 <- confint_betabinom(
  x = prob_tbl_2$x,
```

```
  status = prob_tbl_2$status,
  dist_params = rr_2$coefficients,
  distribution = "lognormal3",
  bounds = "two_sided",
  conf_level = 0.9,
  direction = "x"
)
```

`confint_fisher` *Fisher's Confidence Bounds for Quantiles and Probabilities*

Description

This function computes normal-approximation confidence intervals for quantiles and failure probabilities.

Usage

```
confint_fisher(x, ...)

## S3 method for class 'wt_model'
confint_fisher(
  x,
  b_lives = c(0.01, 0.1, 0.5),
  bounds = c("two_sided", "lower", "upper"),
  conf_level = 0.95,
  direction = c("y", "x"),
  ...
)
```

Arguments

`x`	Object with classes `wt_model` **and** `wt_ml_estimation` returned from `ml_estimation`.
`...`	Further arguments passed to or from other methods. Currently not used.
`b_lives`	A numeric vector indicating the probabilities p of the B_p-lives (quantiles) to be considered.
`bounds`	A character string specifying of which bounds have to be computed. One of `"two_sided"`, `"lower"` or `"upper"`.
`conf_level`	Confidence level of the interval.
`direction`	A character string specifying the direction of the confidence interval. One of `"y"` (failure probabilities) or `"x"` (quantiles).

Details

The basis for the calculation of these confidence bounds are the standard errors determined by the delta method and hence the required (log-)location-scale parameters as well as the variance-covariance matrix of these have to be estimated with maximum likelihood.

The bounds on the probability are determined by the *z-procedure*. See 'References' for more information on this approach.

Value

A tibble with class `wt_confint` containing the following columns:

- `x` : An ordered sequence of the lifetime characteristic regarding the failed units, starting at `min(x)` and ending up at `max(x)`. With `b_lives = c(0.01,0.1,0.5)` the 1%, 10% and 50% quantiles are additionally included in `x`, but only if the specified probabilities are in the range of the estimated probabilities.
- `prob` : An ordered sequence of probabilities with specified `b_lives` included.
- `std_err` : Estimated standard errors with respect to `direction`.
- `lower_bound` : Provided, if `bounds` is one of `"two_sided"` or `"lower"`. Lower limit of the confidence region with respect to `direction`, i.e. quantiles or probabilities.
- `upper_bound` : Provided, if `bounds` is one of `"two_sided"` or `"upper"`. Upper limit of the confidence region with respect to `direction`, i.e. quantiles or probabilities.
- `distribution` : Specified distribution (determined when calling `ml_estimation`).
- `bounds` : Specified bound(s).
- `direction` : Specified direction.
- `cdf_estimation_method` : A character that is always `NA_character`. For the generic visualization functions this column has to be provided.

References

Meeker, William Q; Escobar, Luis A., Statistical methods for reliability data, New York: Wiley series in probability and statistics, 1998

Examples

```
# Reliability data preparation:
## Data for two-parametric model:
data_2p <- reliability_data(
  shock,
  x = distance,
  status = status
)

## Data for three-parametric model:
data_3p <- reliability_data(
  alloy,
  x = cycles,
  status = status
```

```
)

# Model estimation with ml_estimation():
ml_2p <- ml_estimation(
  data_2p,
  distribution = "weibull"
)

ml_3p <- ml_estimation(
  data_3p,
  distribution = "lognormal3",
  conf_level = 0.90
)


# Example 1 - Two-sided 95% confidence interval for probabilities ('y'):
conf_fisher_1 <- confint_fisher(
  x = ml_2p,
  bounds = "two_sided",
  conf_level = 0.95,
  direction = "y"
)

# Example 2 - One-sided lower/upper 90% confidence interval for quantiles ('x'):
conf_fisher_2_1 <- confint_fisher(
  x = ml_2p,
  bounds = "lower",
  conf_level = 0.90,
  direction = "x"
)

conf_fisher_2_2 <- confint_fisher(
  x = ml_2p,
  bounds = "upper",
  conf_level = 0.90,
  direction = "x"
)

# Example 3 - Two-sided 90% confidence intervals for both directions using
# a three-parametric model:

conf_fisher_3_1 <- confint_fisher(
  x = ml_3p,
  bounds = "two_sided",
  conf_level = 0.90,
  direction = "y"
)

conf_fisher_3_2 <- confint_fisher(
  x = ml_3p,
  bounds = "two_sided",
  conf_level = 0.90,
  direction = "x"
```

```
)
```

`confint_fisher.default` *Fisher's Confidence Bounds for Quantiles and Probabilities*

Description

This function computes normal-approximation confidence intervals for quantiles and failure probabilities.

Usage

```
## Default S3 method:
confint_fisher(
  x,
  status,
  dist_params,
  dist_varcov,
 distribution = c("weibull", "lognormal", "loglogistic", "normal", "logistic", "sev",
    "weibull3", "lognormal3", "loglogistic3"),
  b_lives = c(0.01, 0.1, 0.5),
  bounds = c("two_sided", "lower", "upper"),
  conf_level = 0.95,
  direction = c("y", "x"),
  ...
)
```

Arguments

`x`	A numeric vector which consists of lifetime data. Lifetime data could be every characteristic influencing the reliability of a product, e.g. operating time (days/months in service), mileage (km, miles), load cycles.
`status`	A vector of binary data (0 or 1) indicating whether unit i is a right censored observation (= 0) or a failure (= 1).
`dist_params`	A (named) numeric vector of (log-)location-scale parameters returned from `ml_estimation`.
`dist_varcov`	A (named) numeric matrix of estimated variances and covariances returned from `ml_estimation`.
`distribution`	Supposed distribution of the random variable. Has to be in line with the specification made in `ml_estimation`.
`b_lives`	A numeric vector indicating the probabilities p of the B_p-lives (quantiles) to be considered.
`bounds`	A character string specifying of which bounds have to be computed. One of `"two_sided"`, `"lower"` or `"upper"`.

conf_level	Confidence level of the interval.
direction	A character string specifying the direction of the confidence interval. One of "y" (failure probabilities) or "x" (quantiles).
...	Further arguments passed to or from other methods. Currently not used.

Details

The basis for the calculation of these confidence bounds are the standard errors determined by the delta method and hence the required (log-)location-scale parameters as well as the variance-covariance matrix of these have to be estimated with maximum likelihood.

The bounds on the probability are determined by the *z-procedure*. See 'References' for more information on this approach.

Value

A tibble with class wt_confint containing the following columns:

- x : An ordered sequence of the lifetime characteristic regarding the failed units, starting at min(x) and ending up at max(x). With b_lives = c(0.01,0.1,0.5) the 1%, 10% and 50% quantiles are additionally included in x, but only if the specified probabilities are in the range of the estimated probabilities.
- prob : An ordered sequence of probabilities with specified b_lives included.
- std_err : Estimated standard errors with respect to direction.
- lower_bound : Provided, if bounds is one of "two_sided" or "lower". Lower limit of the confidence region with respect to direction, i.e. quantiles or probabilities.
- upper_bound : Provided, if bounds is one of "two_sided" or "upper". Upper limit of the confidence region with respect to direction, i.e. quantiles or probabilities.
- distribution : Specified distribution (determined when calling ml_estimation).
- bounds : Specified bound(s).
- direction : Specified direction.
- cdf_estimation_method : A character that is always NA_character. For the generic visualization functions this column has to be provided.

References

Meeker, William Q; Escobar, Luis A., Statistical methods for reliability data, New York: Wiley series in probability and statistics, 1998

See Also

confint_fisher

Examples

```
# Vectors:
obs <- seq(10000, 100000, 10000)
status_1 <- c(0, 1, 1, 0, 0, 0, 1, 0, 1, 0)

cycles <- alloy$cycles
status_2 <- alloy$status


# Model estimation with ml_estimation():
ml <- ml_estimation(
  x = obs,
  status = status_1,
  distribution = "weibull",
  conf_level = 0.90
)

ml_2 <- ml_estimation(
  x = cycles,
  status = status_2,
  distribution = "lognormal3"
)

# Example 1 - Two-sided 95% confidence interval for probabilities ('y'):
conf_fisher_1 <- confint_fisher(
  x = obs,
  status = status_1,
  dist_params = ml$coefficients,
  dist_varcov = ml$varcov,
  distribution = "weibull",
  bounds = "two_sided",
  conf_level = 0.95,
  direction = "y"
)

# Example 2 - One-sided lower/upper 90% confidence interval for quantiles ('x'):
conf_fisher_2_1 <- confint_fisher(
  x = obs,
  status = status_1,
  dist_params = ml$coefficients,
  dist_varcov = ml$varcov,
  distribution = "weibull",
  bounds = "lower",
  conf_level = 0.90,
  direction = "x"
)

conf_fisher_2_2 <- confint_fisher(
  x = obs,
  status = status_1,
  dist_params = ml$coefficients,
  dist_varcov = ml$varcov,
```

```
  distribution = "weibull",
  bounds = "upper",
  conf_level = 0.90,
  direction = "x"
)

# Example 3 - Two-sided 90% confidence intervals for both directions using
# a three-parametric model:

conf_fisher_3_1 <- confint_fisher(
  x = cycles,
  status = status_2,
  dist_params = ml_2$coefficients,
  dist_varcov = ml_2$varcov,
  distribution = "lognormal3",
  bounds = "two_sided",
  conf_level = 0.90,
  direction = "y"
)

conf_fisher_3_2 <- confint_fisher(
  x = cycles,
  status = status_2,
  dist_params = ml_2$coefficients,
  dist_varcov = ml_2$varcov,
  distribution = "lognormal3",
  bounds = "two_sided",
  conf_level = 0.90,
  direction = "x"
)
```

`delta_method` *Delta Method for Parametric Lifetime Distributions*

Description

This function applies the delta method for two- or three-parametric lifetime distributions that belong to the (log-)location-scale family.

Usage

```
delta_method(
  p,
  dist_params,
  dist_varcov,
 distribution = c("weibull", "lognormal", "loglogistic", "normal", "logistic", "sev",
    "weibull3", "lognormal3", "loglogistic3"),
  direction = c("y", "x")
)
```

Arguments

p	A numeric vector of probabilities or quantiles. If the standard errors of quantiles should be determined the corresponding probabilities have to be specified, and if the standard errors of standardized quantiles (z-values) should be computed corresponding quantiles are required.
dist_params	A (named) numeric vector of (log-)location-scale parameters returned from ml_estimation.
dist_varcov	A (named) numeric matrix of estimated variances and covariances returned from ml_estimation.
distribution	Supposed distribution of the random variable. Has to be in line with the specification made in ml_estimation.
direction	A character string specifying for which quantity the standard errors are calculated. One of "y" (if p are quantiles) or "x" (if p are probabilities).

Details

The delta method estimates the standard errors for quantities that can be written as non-linear functions of ML estimators. Hence, (log-)location-scale parameters as well as the variance-covariance matrix of these have to be estimated with maximum likelihood.

The estimated standard errors are needed to calculate Fisher's (normal approximation) confidence intervals. For confidence bounds on the probability, standard errors of the standardized quantiles (direction = "y") have to be computed (*z-procedure*) and for bounds on quantiles, standard errors of quantiles (direction = "x") are required. For more information see confint_fisher.

Value

A numeric vector of estimated standard errors for quantiles or standardized quantiles (*z-values*).

References

Meeker, William Q; Escobar, Luis A., Statistical methods for reliability data, New York: Wiley series in probability and statistics, 1998

Examples

```
# Reliability data preparation:
data <- reliability_data(
  shock,
  x = distance,
  status = status
)

# Parameter estimation using maximum likelihood:
mle <- ml_estimation(
  data,
  distribution = "weibull",
  conf_level = 0.95
)

# Example 1 - Standard errors of standardized quantiles:
```

```
delta_y <- delta_method(
  p = shock$distance,
  dist_params = mle$coefficients,
  dist_varcov = mle$varcov,
  distribution = "weibull",
  direction = "y"
)

# Example 2 - Standard errors of quantiles:
delta_x <- delta_method(
  p = seq(0.01, 0.99, 0.01),
  dist_params = mle$coefficients,
  dist_varcov = mle$varcov,
  distribution = "weibull",
  direction = "x"
)
```

`dist_delay` *Parameter Estimation of a Delay Distribution*

Description

This function models a delay (in days) random variable (e.g. in logistic, registration, report) using a supposed continuous distribution. First, the element-wise differences in days of both vectors `date_1` and `date_2` are calculated and then the parameter(s) of the assumed distribution are estimated with maximum likelihood. See 'Details' for more information.

Usage

```
dist_delay(date_1, date_2, distribution = c("lognormal", "exponential"))
```

Arguments

`date_1`	A vector of class `character` or `Date`, in the format "yyyy-mm-dd", indicating the earlier of the two dates. Use `NA` for missing elements.
`date_2`	A vector of class `character` or `Date`, in the format "yyyy-mm-dd", indicating the later of the two dates. Use `NA` for missing elements.
`distribution`	Supposed distribution of the random variable.

Details

The distribution parameter(s) are determined on the basis of complete cases, i.e. there is no `NA` in one of the related vector elements `c(date_1[i],date_2[i])`. Time differences less than or equal to zero are not considered as well.

Value

A list of class `delay_estimation` which contains:

- `coefficients` : A named vector of estimated parameter(s).
- `delay` : A numeric vector of element-wise computed differences in days.
- `distribution` : Specified distribution.

Examples

```
# Example 1 - Delay in registration:
date_of_production   <- c("2014-07-28", "2014-02-17", "2014-07-14",
                          "2014-06-26", "2014-03-10", "2014-05-14",
                          "2014-05-06", "2014-03-07", "2014-03-09",
                          "2014-04-13", "2014-05-20", "2014-07-07",
                          "2014-01-27", "2014-01-30", "2014-03-17",
                          "2014-02-09", "2014-04-14", "2014-04-20",
                          "2014-03-13", "2014-02-23", "2014-04-03",
                          "2014-01-08", "2014-01-08")
date_of_registration <- c(NA, "2014-03-29", "2014-12-06", "2014-09-09",
                          NA, NA, "2014-06-16", NA, "2014-05-23",
                          "2014-05-09", "2014-05-31", NA, "2014-04-13",
                          NA, NA, "2014-03-12", NA, "2014-06-02",
                          NA, "2014-03-21", "2014-06-19", NA, NA)

params_delay_regist  <- dist_delay(
  date_1 = date_of_production,
  date_2 = date_of_registration,
  distribution = "lognormal"
)

# Example 2 - Delay in report:
date_of_repair <- c(NA, "2014-09-15", "2015-07-04", "2015-04-10", NA,
                    NA, "2015-04-24", NA, "2015-04-25", "2015-04-24",
                    "2015-06-12", NA, "2015-05-04", NA, NA,
                    "2015-05-22", NA, "2015-09-17", NA, "2015-08-15",
                    "2015-11-26", NA, NA)

date_of_report <- c(NA, "2014-10-09", "2015-08-28", "2015-04-15", NA,
                    NA, "2015-05-16", NA, "2015-05-28", "2015-05-15",
                    "2015-07-11", NA, "2015-08-14", NA, NA,
                    "2015-06-05", NA, "2015-10-17", NA, "2015-08-21",
                    "2015-12-02", NA, NA)

params_delay_report  <- dist_delay(
  date_1 = date_of_repair,
  date_2 = date_of_report,
  distribution = "exponential"
)
```

`dist_delay_register` *Parameter Estimation of the Delay in Registration Distribution*

Description

Soft-deprecated

`dist_delay_register()` is no longer under active development, switching to `dist_delay` is recommended.

Usage

```
dist_delay_register(date_prod, date_register, distribution = "lognormal")
```

Arguments

`date_prod`	A vector of class `character` or `Date`, in the format "yyyy-mm-dd", indicating the date of production of a unit. Use `NA` for missing elements.
`date_register`	A vector of class `character` or `Date`, in the format "yyyy-mm-dd", indicating the date of registration of a unit. Use `NA` for missing elements.
`distribution`	Supposed distribution of the random variable. Only `"lognormal"` is implemented.

Details

This function introduces a delay random variable by calculating the time difference between the registration and production date for the sample units and afterwards estimates the parameter(s) of a supposed distribution, using maximum likelihood.

Value

A named vector of estimated parameters for the specified distribution.

Examples

```
date_of_production   <- c("2014-07-28", "2014-02-17", "2014-07-14",
                          "2014-06-26", "2014-03-10", "2014-05-14",
                          "2014-05-06", "2014-03-07", "2014-03-09",
                          "2014-04-13", "2014-05-20", "2014-07-07",
                          "2014-01-27", "2014-01-30", "2014-03-17",
                          "2014-02-09", "2014-04-14", "2014-04-20",
                          "2014-03-13", "2014-02-23", "2014-04-03",
                          "2014-01-08", "2014-01-08")
date_of_registration <- c(NA, "2014-03-29", "2014-12-06", "2014-09-09",
                          NA, NA, "2014-06-16", NA, "2014-05-23",
                          "2014-05-09", "2014-05-31", NA, "2014-04-13",
                          NA, NA, "2014-03-12", NA, "2014-06-02",
                          NA, "2014-03-21", "2014-06-19", NA, NA)
```

```
params_delay_regist  <- dist_delay_register(
  date_prod = date_of_production,
  date_register = date_of_registration,
  distribution = "lognormal"
)
```

`dist_delay_report` *Parameter Estimation of the Delay in Report Distribution*

Description

Soft-deprecated

`dist_delay_report()` is no longer under active development, switching to `dist_delay` is recommended.

Usage

```
dist_delay_report(date_repair, date_report, distribution = "lognormal")
```

Arguments

`date_repair`	a vector of class `character` or `Date`, in the format "yyyy-mm-dd", indicating the date of repair of a failed unit. Use `NA` for missing elements.
`date_report`	a vector of class `character` or `Date`, in the format "yyyy-mm-dd", indicating the date of report of a failed unit. Use `NA` for missing elements.
`distribution`	Supposed distribution of the random variable. Only `"lognormal"` is implemented.

Details

This function introduces a delay random variable by calculating the time difference between the report and repair date for the sample units and afterwards estimates the parameter(s) of a supposed distribution, using maximum likelihood.

Value

A named vector of estimated parameters for the specified distribution.

Examples

```
date_of_repair <- c(NA, "2014-09-15", "2015-07-04", "2015-04-10", NA,
                    NA, "2015-04-24", NA, "2015-04-25", "2015-04-24",
                    "2015-06-12", NA, "2015-05-04", NA, NA,
                    "2015-05-22", NA, "2015-09-17", NA, "2015-08-15",
                    "2015-11-26", NA, NA)

date_of_report <- c(NA, "2014-10-09", "2015-08-28", "2015-04-15", NA,
```

```
                    NA, "2015-05-16", NA, "2015-05-28", "2015-05-15",
                    "2015-07-11", NA, "2015-08-14", NA, NA,
                    "2015-06-05", NA, "2015-10-17", NA, "2015-08-21",
                    "2015-12-02", NA, NA)

params_delay_report  <- dist_delay_report(
  date_repair = date_of_repair,
  date_report = date_of_report,
  distribution = "lognormal"
)
```

`dist_mileage` *Parameter Estimation of an Annual Mileage Distribution*

Description

This function models a mileage random variable on an annual basis with respect to a supposed continuous distribution. First, the distances are calculated for one year (365 days) using a linear relationship between the distance and operating time. Second, the parameter(s) of the assumed distribution are estimated with maximum likelihood. See 'Details' for more information.

Usage

```
dist_mileage(mileage, time, distribution = c("lognormal", "exponential"))
```

Arguments

`mileage`	A numeric vector of distances covered. Use `NA` for missing elements.
`time`	A numeric vector of operating times. Use `NA` for missing elements.
`distribution`	Supposed distribution of the random variable.

Details

The distribution parameter(s) are determined on the basis of complete cases, i.e. there is no `NA` in one of the related vector elements `c(mileage[i],time[i])`. Distances and operating times less than or equal to zero are not considered as well.

Assumption of linear relationship: Imagine a component in a vehicle has endured a distance of 25000 kilometers (km) in 500 days (d), the annual distance of this unit is

$$25000km \cdot (\frac{365d}{500d}) = 18250km$$

Value

A list of class `mileage_estimation` which contains:

- `coefficients` : A named vector of estimated parameter(s).
- `miles_annual` : A numeric vector of element-wise computed annual distances using the linear relationship described in 'Details'.
- `distribution` : Specified distribution.

Examples

```
# Data for examples:
date_of_registration <- c("2014-08-17", "2014-03-29", "2014-12-06",
                          "2014-09-09", "2014-05-14", "2014-07-01",
                          "2014-06-16", "2014-04-03", "2014-05-23",
                          "2014-05-09", "2014-05-31", "2014-08-12",
                          "2014-04-13", "2014-02-15", "2014-07-07",
                          "2014-03-12", "2014-05-27", "2014-06-02",
                          "2014-05-20", "2014-03-21", "2014-06-19",
                          "2014-02-12", "2014-03-27")
date_of_repair       <- c(NA, "2014-09-15", "2015-07-04", "2015-04-10", NA,
                          NA, "2015-04-24", NA, "2015-04-25", "2015-04-24",
                          "2015-06-12", NA, "2015-05-04", NA, NA, "2015-05-22",
                          NA, "2015-09-17", NA, "2015-08-15", "2015-11-26",
                          NA, NA)
date_of_analysis     <- "2015-12-31"

## Assume that mileage is only known for units that have failed (date_of_repair != NA).
mileage              <- c(NA, 15655, 13629, 18292, NA, NA, 33555, NA, 21737,
                          29870, 21068, NA, 122283, NA, NA, 36088, NA, 11153,
                          NA, 122842, 20349, NA, NA)

## time in service is the difference between repair and registration for failed
## items and the difference between date of analysis and date of registration
## for intact units.
time_in_service <- difftime(
  as.Date(date_of_repair, format = "%Y-%m-%d"),
  as.Date(date_of_registration, format = "%Y-%m-%d"),
  units = "days"
)
time_in_service[is.na(time_in_service)] <- difftime(
  as.Date(date_of_analysis, format = "%Y-%m-%d"),
  as.Date(date_of_registration[is.na(time_in_service)], format = "%Y-%m-%d"),
  units = "days"
)
time_in_service <- as.numeric(time_in_service)

# Example 1 - Assuming lognormal annual mileage distribution:
params_mileage_annual <- dist_mileage(
  mileage = mileage,
  time = time_in_service,
  distribution = "lognormal"
```

```
)

# Example 2 - Assuming exponential annual mileage distribution:
params_mileage_annual_2 <- dist_mileage(
  mileage = mileage,
  time = time_in_service,
  distribution = "exponential"
)
```

`estimate_cdf` *Estimation of Failure Probabilities*

Description

This function applies a non-parametric method to estimate the failure probabilities of complete data taking (multiple) right-censored observations into account.

Usage

```
estimate_cdf(x, ...)

## S3 method for class 'wt_reliability_data'
estimate_cdf(
  x,
  methods = c("mr", "johnson", "kaplan", "nelson"),
  options = list(),
  ...
)
```

Arguments

`x`	A tibble returned by reliability_data.
`...`	Further arguments passed to or from other methods. Currently not used.
`methods`	One or multiple methods of `"mr"`, `"johnson"`, `"kaplan"` or `"nelson"` used for the estimation of failure probabilities. See 'Details'.
`options`	A list of named options. See 'Options'.

Details

One or multiple techniques can be used for the `methods` argument:

- `"mr"` : Method *Median Ranks* is used to estimate the failure probabilities of failed units without considering censored items. Tied observations can be handled in three ways (See 'Options'):
  - `"max"` : Highest observed rank is assigned to tied observations.
  - `"min"` : Lowest observed rank is assigned to tied observations.

  - `"average"` : Mean rank is assigned to tied observations.

  Two formulas can be used to determine cumulative failure probabilities *F(t)* (See 'Options'):

  - `"benard"` : Benard's approximation for Median Ranks.
  - `"invbeta"` : Exact Median Ranks using the inverse beta distribution.

- `"johnson"` : The *Johnson* method is used to estimate the failure probabilities of failed units, taking censored units into account. Compared to complete data, correction of probabilities is done by the computation of adjusted ranks.
- `"kaplan"` : The method of *Kaplan* and *Meier* is used to estimate the survival function *S(t)* with respect to (multiple) right censored data. The complement of *S(t)*, i.e. *F(t)*, is returned. In contrast to the original *Kaplan-Meier* estimator, one modification is made (see 'References').

  Note : The *Kaplan-Meier* estimator does not assign ranks to observations, so the beta-binomial confidence intervals *cannot* be calculated using this method.
- `"nelson"` : The *Nelson-Aalen* estimator models the cumulative hazard rate function in case of (multiple) right censored data. Equating the formal definition of the hazard rate with that according to *Nelson-Aalen* results in a formula for the calculation of failure probabilities.

  Note : The *Nelson-Aalen* estimator does not assign ranks to observations, so the beta-binomial confidence intervals *cannot* be calculated using this method.

Value

A tibble containing the following columns:

- `id` : Identification for every unit.
- `x` : Lifetime characteristic.
- `status` : Binary data (0 or 1) indicating whether a unit is a right censored observation (= 0) or a failure (= 1).
- `rank` : The (computed) ranks. Determined for methods `"mr"` and `"johnson"`, filled with `NA` for other methods or if `status = 0`.
- `prob` : Estimated failure probabilities, `NA` if `status = 0`.
- `cdf_estimation_method` : Specified method for the estimation of failure probabilities.

Options

The listed options can only be applied for method `"mr"`:

- `mr_method` : `"benard"` (default) or `"invbeta"`.
- `mr_ties.method` : `"max"` (default), `"min"` or `"average"`.

References

NIST/SEMATECH e-Handbook of Statistical Methods, *8.2.1.5. Empirical model fitting - distribution free (Kaplan-Meier) approach*, NIST SEMATECH, December 3, 2020

Examples

```
# Reliability data:
data <- reliability_data(
  alloy,
  x = cycles,
  status = status
)

# Example 1 - Johnson method:
prob_tbl <- estimate_cdf(
  x = data,
  methods = "johnson"
)

# Example 2 - Multiple methods:
prob_tbl_2 <- estimate_cdf(
  x = data,
  methods = c("johnson", "kaplan", "nelson")
)

# Example 3 - Method 'mr' with options:
prob_tbl_3 <- estimate_cdf(
  x = data,
  methods = "mr",
  options = list(
    mr_method = "invbeta",
    mr_ties.method = "average"
  )
)

# Example 4 - Multiple methods and option for 'mr':
prob_tbl_4 <- estimate_cdf(
  x = data,
  methods = c("mr", "johnson"),
  options = list(
    mr_ties.method = "max"
  )
)
```

`estimate_cdf.default` *Estimation of Failure Probabilities*

Description

This function applies a non-parametric method to estimate the failure probabilities of complete data taking (multiple) right-censored observations into account.

Usage

```
## Default S3 method:
estimate_cdf(
  x,
  status,
  id = NULL,
  method = c("mr", "johnson", "kaplan", "nelson"),
  options = list(),
  ...
)
```

Arguments

x	A numeric vector which consists of lifetime data. Lifetime data could be every characteristic influencing the reliability of a product, e.g. operating time (days/months in service), mileage (km, miles), load cycles.
status	A vector of binary data (0 or 1) indicating whether unit *i* is a right censored observation (= 0) or a failure (= 1).
id	A vector for the identification of every unit. Default is NULL.
method	Method used for the estimation of failure probabilities. See 'Details'.
options	A list of named options. See 'Options'.
...	Further arguments passed to or from other methods. Currently not used.

Details

The following techniques can be used for the method argument:

- "mr" : Method *Median Ranks* is used to estimate the failure probabilities of failed units without considering censored items. Tied observations can be handled in three ways (See 'Options'):
  - "max" : Highest observed rank is assigned to tied observations.
  - "min" : Lowest observed rank is assigned to tied observations.
  - "average" : Mean rank is assigned to tied observations.

  Two formulas can be used to determine cumulative failure probabilities *F(t)* (See 'Options'):
  - "benard" : Benard's approximation for Median Ranks.
  - "invbeta" : Exact Median Ranks using the inverse beta distribution.
- "johnson" : The *Johnson* method is used to estimate the failure probabilities of failed units, taking censored units into account. Compared to complete data, correction of probabilities is done by the computation of adjusted ranks.
- "kaplan" : The method of *Kaplan* and *Meier* is used to estimate the survival function *S(t)* with respect to (multiple) right censored data. The complement of *S(t)*, i.e. *F(t)*, is returned. In contrast to the original *Kaplan-Meier* estimator, one modification is made (see 'References').

  Note : The *Kaplan-Meier* estimator does not assign ranks to observations, so the beta-binomial confidence intervals *cannot* be calculated using this method.

- `"nelson"` : The *Nelson-Aalen* estimator models the cumulative hazard rate function in case of (multiple) right censored data. Equating the formal definition of the hazard rate with that according to *Nelson-Aalen* results in a formula for the calculation of failure probabilities.

  Note : The *Nelson-Aalen* estimator does not assign ranks to observations, so the beta-binomial confidence intervals *cannot* be calculated using this method.

Value

A tibble containing the following columns:

- `id` : Identification for every unit.
- `x` : Lifetime characteristic.
- `status` : Binary data (0 or 1) indicating whether a unit is a right censored observation (= 0) or a failure (= 1).
- `rank` : The (computed) ranks. Determined for methods `"mr"` and `"johnson"`, filled with `NA` for other methods or if `status = 0`.
- `prob` : Estimated failure probabilities, `NA` if `status = 0`.
- `cdf_estimation_method` : Specified method for the estimation of failure probabilities.

Options

The listed options can only be applied for method `"mr"`:

- `mr_method` : `"benard"` (default) or `"invbeta"`.
- `mr_ties.method` : `"max"` (default), `"min"` or `"average"`.

References

NIST/SEMATECH e-Handbook of Statistical Methods, *8.2.1.5. Empirical model fitting - distribution free (Kaplan-Meier) approach*, NIST SEMATECH, December 3, 2020

See Also

`estimate_cdf`

Examples

```
# Vectors:
cycles <- alloy$cycles
status <- alloy$status

# Example 1 - Johnson method:
prob_tbl <- estimate_cdf(
  x = cycles,
  status = status,
  method = "johnson"
)
```

```
# Example 2 - Method 'mr' with options:
prob_tbl_2 <- estimate_cdf(
  x = cycles,
  status = status,
  method = "mr",
  options = list(
    mr_method = "invbeta",
    mr_ties.method = "average"
  )
)
```

---

johnson_method *Estimation of Failure Probabilities using Johnson's Method*

Description

Soft-deprecated

`johnson_method()` is no longer under active development, switching to `estimate_cdf` is recommended.

Usage

```
johnson_method(x, status, id = NULL)
```

Arguments

`x`	A numeric vector which consists of lifetime data. Lifetime data could be every characteristic influencing the reliability of a product, e.g. operating time (days/months in service), mileage (km, miles), load cycles.
`status`	A vector of binary data (0 or 1) indicating whether unit i is a right censored observation (= 0) or a failure (= 1).
`id`	A vector for the identification of every unit. Default is `NULL`.

Details

This non-parametric approach is used to estimate the failure probabilities in terms of uncensored or (multiple) right censored data. Compared to complete data the correction is done by calculating adjusted ranks which takes non-defective units into account.

Value

A tibble containing the following columns:

- `id` : Identification for every unit.
- `x` : Lifetime characteristic.
- `status` : Binary data (0 or 1) indicating whether a unit is a right censored observation (= 0) or a failure (= 1).

- rank : The adjusted ranks.
- prob : Estimated failure probabilities, NA if status = 0.
- cdf_estimation_method : Specified method for the estimation of failure probabilities (always 'johnson').

Examples

```
# Vectors:
obs   <- seq(10000, 100000, 10000)
state <- c(0, 1, 1, 0, 0, 0, 1, 0, 1, 0)
uic   <- c("3435", "1203", "958X", "XX71", "abcd", "tz46",
           "fl29", "AX23", "Uy12", "kl1a")

# Example 1 - Johnson method for intact and failed units:
tbl_john <- johnson_method(
  x = obs,
  status = state,
  id = uic
)

# Example 2 - Johnson's method works also if only defective units are considered:
tbl_john_2 <- johnson_method(
  x = obs,
  status = rep(1, length(obs))
)
```

---

kaplan_method *Estimation of Failure Probabilities using the Kaplan-Meier Estimator*

Description

Soft-deprecated

kaplan_method() is no longer under active development, switching to estimate_cdf is recommended.

Usage

```
kaplan_method(x, status, id = NULL)
```

Arguments

x	A numeric vector which consists of lifetime data. Lifetime data could be every characteristic influencing the reliability of a product, e.g. operating time (days/months in service), mileage (km, miles), load cycles.
status	A vector of binary data (0 or 1) indicating whether unit i is a right censored observation (= 0) or a failure (= 1).
id	A vector for the identification of every unit. Default is NULL.

Details

Whereas the non-parametric Kaplan-Meier estimator is used to estimate the survival function *S(t)* in terms of (multiple) right censored data, the complement is an estimate of the cumulative distribution function *F(t)*. One modification is made in contrast to the original Kaplan-Meier estimator (see 'References').

Note : The *Kaplan-Meier* estimator does not assign ranks to observations, so the beta-binomial confidence intervals *cannot* be calculated using this method.

Value

A tibble containing the following columns:

- `id` : Identification for every unit.
- `x` : Lifetime characteristic.
- `status` : Binary data (0 or 1) indicating whether a unit is a right censored observation (= 0) or a failure (= 1).
- `rank` : Filled with `NA`.
- `prob` : Estimated failure probabilities, `NA` if `status = 0`.
- `cdf_estimation_method` : Specified method for the estimation of failure probabilities (always 'kaplan').

References

NIST/SEMATECH e-Handbook of Statistical Methods, *8.2.1.5. Empirical model fitting - distribution free (Kaplan-Meier) approach*, NIST SEMATECH, December 3, 2020

Examples

```
# Vectors:
obs   <- seq(10000, 100000, 10000)
state <- c(0, 1, 1, 0, 0, 0, 1, 0, 1, 0)
state_2 <- c(0, 1, 1, 0, 0, 0, 1, 0, 0, 1)
uic   <- c("3435", "1203", "958X", "XX71", "abcd", "tz46",
           "fl29", "AX23","Uy12", "kl1a")

# Example 1 - Observation with highest characteristic is an intact unit:
tbl_kap <- kaplan_method(
  x = obs,
  status = state,
  id = uic
)

# Example 2 - Observation with highest characteristic is a defective unit:
tbl_kap_2 <- kaplan_method(
  x = obs,
  status = state_2
)
```

loglik_function *Log-Likelihood Function for Parametric Lifetime Distributions*

Description

This function computes the log-likelihood value with respect to a given set of parameters. For two-parametric models the location and scale parameters are required. If a three-parametric lifetime distribution is needed an additional threshold parameter has to be provided. In terms of *Maximum Likelihood Estimation* this function can be optimized (optim) to estimate the parameters and variance-covariance matrix of the parameters.

Usage

```
loglik_function(
  x,
  status,
  wts = rep(1, length(x)),
  dist_params,
 distribution = c("weibull", "lognormal", "loglogistic", "normal", "logistic", "sev",
    "weibull3", "lognormal3", "loglogistic3")
)
```

Arguments

x	A numeric vector which consists of lifetime data. Lifetime data could be every characteristic influencing the reliability of a product, e.g. operating time (days/months in service), mileage (km, miles), load cycles.
status	A vector of binary data (0 or 1) indicating whether unit i is a right censored observation (= 0) or a failure (= 1).
wts	Optional vector of case weights. The length of wts must be the same as the number of observations in x.
dist_params	A (named) numeric vector of (log-)location-scale parameters in the order of location (μ) and scale (σ). If a three-parametric model is selected, the threshold parameter (γ) has to be the third element.
distribution	Supposed distribution of the random variable.

Value

Returns the log-likelihood value for the data with respect to the parameters given in dist_params.

References

Meeker, William Q; Escobar, Luis A., Statistical methods for reliability data, New York: Wiley series in probability and statistics, 1998

Examples

```
# Vectors:
cycles <- alloy$cycles
status <- alloy$status

# Example 1 - Evaluating Log-Likelihood function of two-parametric weibull:
loglik_weib <- loglik_function(
  x = cycles,
  status = status,
  dist_params = c(5.29, 0.33),
  distribution = "weibull"
)

# Example 2 - Evaluating Log-Likelihood function of three-parametric weibull:
loglik_weib3 <- loglik_function(
  x = cycles,
  status = status,
  dist_params = c(4.54, 0.76, 92.99),
  distribution = "weibull3"
)
```

---

loglik_profiling *Log-Likelihood Profile Function for Parametric Lifetime Distributions with Threshold*

Description

This function evaluates the log-likelihood with respect to a given threshold parameter of a three-parametric lifetime distribution. In terms of *Maximum Likelihood Estimation* this function can be optimized (optim) to estimate the threshold parameter.

Usage

```
loglik_profiling(
  x,
  status,
  thres,
  distribution = c("weibull3", "lognormal3", "loglogistic3")
)
```

Arguments

x	A numeric vector which consists of lifetime data. Lifetime data could be every characteristic influencing the reliability of a product, e.g. operating time (days/months in service), mileage (km, miles), load cycles.
status	A vector of binary data (0 or 1) indicating whether unit i is a right censored observation (= 0) or a failure (= 1).

thres	A numeric value for the threshold parameter.
distribution	Supposed three-parametric distribution of the random variable.

Value

Returns the log-likelihood value for the data with respect to the threshold parameter thres.

References

Meeker, William Q; Escobar, Luis A., Statistical methods for reliability data, New York: Wiley series in probability and statistics, 1998

Examples

```
# Vectors:
cycles <- alloy$cycles
status <- alloy$status

# Determining the optimal loglikelihood value:
## Range of threshold parameter must be smaller than the first failure:
threshold <- seq(
  0,
  min(cycles[status == 1]) - 0.1,
  length.out = 50
)

## loglikelihood value with respect to threshold values:
profile_logL <- loglik_profiling(
  x = cycles,
  status = status,
  thres = threshold,
  distribution = "weibull3"
)

## Threshold value (among the candidates) that maximizes the
## loglikelihood:
threshold[which.max(profile_logL)]

## plot:
plot(
  threshold,
  profile_logL,
  type = "l"
)
abline(
  v = threshold[which.max(profile_logL)],
  h = max(profile_logL),
  col = "red"
)
```

mcs_delay *Adjustment of Operating Times by Delays using a Monte Carlo Approach*

Description

In general, the amount of available information about units in the field is very different. During the warranty period, there are only a few cases with complete data (mainly *failed units*) but lots of cases with incomplete data (usually *censored units*). As a result, the operating time of units with incomplete information is often inaccurate and must be adjusted by delays.

This function reduces the operating times of incomplete observations by simulated delays (in days). A unit is considered as incomplete if the later of the two dates is unknown, i.e. `date_2 = NA`. See 'Details' for some practical examples.

Random delay numbers are drawn from the distribution determined by complete cases (described in 'Details' of `dist_delay`).

Usage

```
mcs_delay(
  date_1,
  date_2,
  time,
  status = NULL,
  id = paste0("ID", seq_len(length(time))),
  distribution = c("lognormal", "exponential")
)
```

Arguments

`date_1`	A vector of class `character` or `Date`, in the format "yyyy-mm-dd", indicating the earlier of the two dates. Use `NA` for missing elements. If more than one delay should be considered it must be a list where the first element contains the earlier dates of the first delay and the second element contains the earlier dates of the second delay, and so forth.(See 'Examples').
`date_2`	A vector of class `character` or `Date`, in the format "yyyy-mm-dd", indicating the later of the two dates. Use `NA` for missing elements. If more than one delay should be considered it must be a list where the first element contains the later dates of the first delay and the second element contains the later dates of the second delay, and so forth. (See 'Examples').
`time`	A numeric vector of operating times. Use `NA` for missing elements.
`status`	Optional argument. If used it has to be a vector of binary data (0 or 1) indicating whether unit *i* is a right censored observation (= 0) or a failure (= 1). The effect of `status` on the return is described in 'Value'.
`id`	A vector for the identification of every unit.

`distribution` Supposed distribution of the delay random variable. If more than one delay is to be considered and different distributions are assumed for each delay, the argument `distribution` must have the same length as list `date_1` (and `date_2`). For example, in the case of two delays with different distributions, one has to specify the argument as `distribution = c("lognormal","exponential")`. Then the lognormal distribution is applied to the first delay and the exponential distribution to the second (See 'Examples').

Details

In field data analysis time-dependent characteristics (e.g. *time in service*) are often imprecisely recorded. These inaccuracies are caused by unconsidered delays.

For a better understanding of the MCS application in the context of field data, two cases are described below.

- **Delay in registration**: It is common that a supplier, which provides parts to the manufacturing industry does not know when the unit, in which its parts are installed, were put in service (due to unknown `date_2`, i.e. registration or sales date). Without taking the described delay into account, the time in service of the failed units would be the difference between the repair date and `date_1` (i.e. the production date) and for intact units the difference between the present date and `date_1`. But the real operating times are (much) shorter, since the stress on the components have not started until the whole systems were put in service. Hence, units with incomplete data (missing `date_2`) must be reduced by the delays.
- **Delay in report**: Authorized repairers often do not immediately notify the manufacturer or OEM of repairs that were made during the warranty period, but instead pass the information about these repairs in collected forms e.g. weekly, monthly or quarterly. The resulting time difference between the reporting (`date_2`) of the repair in the guarantee database and the actual repair date (`date_1`), which is often assumed to be the failure date, is called the reporting delay. For a given date where the analysis is made there could be units which had a failure but are not registered and therefore treated as censored units. In order to take this case into account and according to the principle of equal opportunities, the lifetime of units with no report date (`date_2 = NA`) is reduced by simulated reporting delays.

Value

A list containing the following elements:

- `data` : A tibble with classes `wt_mcs_data` and `wt_reliability_data` if `status` is provided. Since the class `wt_reliability_data` enables the direct usage of `data` inside `estimate_cdf.wt_reliability_data`, the required lifetime characteristic is automatically set to the operating time `time`.

  If `status = NULL` class is `wt_mcs_data`, which is not supported by `estimate_cdf` due to missing `status`.

  The tibble contains the following columns:

  - `date_1` : Earlier dates. If argument `date_1` is a list of length *i, i > 1* (described in **Arguments**) multiple columns with names `date_1.1`, `date_1.2`, ..., `date_1.i` and the corresponding values of the earlier dates are used.
  - `date_2` : Later dates. In the case of a list with length greater than 1, the routine described above is used.

  - time : Adjusted operating times for incomplete observations and input operating times for the complete observations.
  - status (**optional**) :
    * If argument status = NULL column status does not exist.
    * If argument status is provided the column contains the entered binary data (0 or 1).
  - id : Identification of every unit.

- sim_data : A tibble with column sim_delay that holds the simulated delay-specific numbers for incomplete cases and 0 for complete cases. If more than one delay was considered multiple columns sim_delay.1, sim_delay.2, ..., sim_delay.i with corresponding delay-specific random numbers are presented.
- model_estimation : A list containing a named list ("delay_distribution") with output of dist_delay. For multiple delays the list contains as many lists as there are delays, i.e. ("delay_distribution.1", "delay_distribution.2", ..., "delay_distribution.i").

References

Verband der Automobilindustrie e.V. (VDA); Qualitätsmanagement in der Automobilindustrie. Zuverlässigkeitssicherung bei Automobilherstellern und Lieferanten. Zuverlässigkeits-Methoden und -Hilfsmittel.; 4th Edition, 2016, <ISSN:0943-9412>

See Also

estimate_cdf

Examples

```
# Data for examples:
date_of_production   <- c("2014-07-28", "2014-02-17", "2014-07-14",
                          "2014-06-26", "2014-03-10", "2014-05-14",
                          "2014-05-06", "2014-03-07", "2014-03-09",
                          "2014-04-13", "2014-05-20", "2014-07-07",
                          "2014-01-27", "2014-01-30", "2014-03-17",
                          "2014-02-09", "2014-04-14", "2014-04-20",
                          "2014-03-13", "2014-02-23", "2014-04-03",
                          "2014-01-08", "2014-01-08")

date_of_registration <- c(NA, "2014-03-29", "2014-12-06", "2014-09-09",
                          NA, NA, "2014-06-16", NA, "2014-05-23",
                          "2014-05-09", "2014-05-31", NA, "2014-04-13",
                          NA, NA, "2014-03-12", NA, "2014-06-02",
                          NA, "2014-03-21", "2014-06-19", NA, NA)

date_of_repair <-       c(NA, "2014-09-15", "2015-07-04", "2015-04-10", NA,
                          NA, "2015-04-24", NA, "2015-04-25", "2015-04-24",
                          "2015-06-12", NA, "2015-05-04", NA, NA,
                          "2015-05-22", NA, "2015-09-17", NA, "2015-08-15",
                          "2015-11-26", NA, NA)

date_of_report <-       c(NA, "2014-10-09", "2015-08-28", "2015-04-15", NA,
                          NA, "2015-05-16", NA, "2015-05-28", "2015-05-15",
```

```
                          "2015-07-11", NA, "2015-08-14", NA, NA,
                          "2015-06-05", NA, "2015-10-17", NA, "2015-08-21",
                          "2015-12-02", NA, NA)

time_in_service <- rep(1000, length(date_of_production))
status <- c(0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0)

# Example 1 - MCS for delay in registration:
mcs_regist <- mcs_delay(
  date_1 = date_of_production,
  date_2 = date_of_registration,
  time = time_in_service,
  status = status,
  distribution = "lognormal"
)

# Example 2 - MCS for delay in report:
mcs_report <- mcs_delay(
  date_1 = date_of_repair,
  date_2 = date_of_report,
  time = time_in_service,
  status = status,
  distribution = "exponential"
)

# Example 3 - Reproducibility of random numbers:
set.seed(1234)
mcs_report_reproduce <- mcs_delay(
  date_1 = date_of_repair,
  date_2 = date_of_report,
  time = time_in_service,
  status = status,
  distribution = "exponential"
)

# Example 4 - MCS for delays in registration and report with same distribution:
mcs_delays <- mcs_delay(
  date_1 = list(date_of_production, date_of_repair),
  date_2 = list(date_of_registration, date_of_report),
  time = time_in_service,
  status = status,
  distribution = "lognormal"
)

# Example 5 - MCS for delays in registration and report with different distributions:
## Assuming lognormal registration and exponential reporting delays.
mcs_delays_2 <- mcs_delay(
  date_1 = list(date_of_production, date_of_repair),
  date_2 = list(date_of_registration, date_of_report),
  time = time_in_service,
  status = status,
  distribution = c("lognormal", "exponential")
)
```

mcs_delays *Adjustment of Operating Times by Delays using a Monte Carlo Approach*

Description

Soft-deprecated

`mcs_delays()` is no longer under active development, switching to `mcs_delay` is recommended.

Usage

```
mcs_delays(
  date_prod,
  date_register,
  date_repair,
  date_report,
  time,
  status,
  distribution = "lognormal",
  details = FALSE
)
```

Arguments

`date_prod`	A vector of class `character` or `Date`, in the format "yyyy-mm-dd", indicating the date of production of a unit. Use `NA` for missing elements.
`date_register`	A vector of class `character` or `Date`, in the format "yyyy-mm-dd", indicating the date of registration of a unit. Use `NA` for missing elements.
`date_repair`	a vector of class `character` or `Date`, in the format "yyyy-mm-dd", indicating the date of repair of a failed unit. Use `NA` for missing elements.
`date_report`	a vector of class `character` or `Date`, in the format "yyyy-mm-dd", indicating the date of report of a failed unit. Use `NA` for missing elements.
`time`	A numeric vector of operating times.
`status`	A vector of binary data (0 or 1) indicating whether unit i is a right censored observation (= 0) or a failure (= 1).
`distribution`	Supposed distribution of the random variable. Only `"lognormal"` is implemented.
`details`	A logical. If `FALSE` the output consists of a vector with corrected operating times for the censored units and the input operating times for the failed units. If `TRUE` the output consists of a detailed list, i.e the same vector as described before, simulated random numbers and estimated distribution parameters.

Details

This function is a wrapper that combines both, the mcs_delay_register and mcs_delay_report function for adjusting the operation times of censored units.

Value

A numerical vector of corrected operating times for the censored units and the input operating times for the failed units if details = FALSE. If details = TRUE the output is a list which consists of the following elements:

- time : Numerical vector of corrected operating times for the censored observations and input operating times for failed units.
- x_sim_regist : Simulated random numbers of specified distribution with estimated parameters for delay in registration. The length of x_sim_regist is equal to the number of censored observations.
- x_sim_report : Simulated random numbers of specified distribution with estimated parameters for delay in report. The length of x_sim_report is equal to the number of censored observations.
- coefficients_regist : Estimated coefficients of supposed distribution for delay in registration.
- coefficients_report : Estimated coefficients of supposed distribution for delay in report

Examples

```
date_of_production   <- c("2014-07-28", "2014-02-17", "2014-07-14",
                          "2014-06-26", "2014-03-10", "2014-05-14",
                          "2014-05-06", "2014-03-07", "2014-03-09",
                          "2014-04-13", "2014-05-20", "2014-07-07",
                          "2014-01-27", "2014-01-30", "2014-03-17",
                          "2014-02-09", "2014-04-14", "2014-04-20",
                          "2014-03-13", "2014-02-23", "2014-04-03",
                          "2014-01-08", "2014-01-08")
date_of_registration <- c("2014-08-17", "2014-03-29", "2014-12-06",
                          "2014-09-09", "2014-05-14", "2014-07-01",
                          "2014-06-16", "2014-04-03", "2014-05-23",
                          "2014-05-09", "2014-05-31", "2014-08-12",
                          "2014-04-13", "2014-02-15", "2014-07-07",
                          "2014-03-12", "2014-05-27", "2014-06-02",
                          "2014-05-20", "2014-03-21", "2014-06-19",
                          "2014-02-12", "2014-03-27")
date_of_repair <- c(NA, "2014-09-15", "2015-07-04", "2015-04-10", NA,
                    NA, "2015-04-24", NA, "2015-04-25", "2015-04-24",
                     "2015-06-12", NA, "2015-05-04", NA, NA,
                     "2015-05-22", NA, "2015-09-17", NA, "2015-08-15",
                     "2015-11-26", NA, NA)

date_of_report <- c(NA, "2014-10-09", "2015-08-28", "2015-04-15", NA,
                     NA, "2015-05-16", NA, "2015-05-28", "2015-05-15",
                     "2015-07-11", NA, "2015-08-14", NA, NA,
                     "2015-06-05", NA, "2015-10-17", NA, "2015-08-21",
```

```
                    "2015-12-02", NA, NA)

op_time <- rep(1000, length(date_of_repair))
status <- c(0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0)

# Example 1 - Simplified vector output:
x_corrected <- mcs_delays(
  date_prod = date_of_production,
  date_register = date_of_registration,
  date_repair = date_of_repair,
  date_report = date_of_report,
  time = op_time,
  status = status,
  distribution = "lognormal",
  details = FALSE
)

# Example 2 - Detailed list output:
list_detail <- mcs_delays(
  date_prod = date_of_production,
  date_register = date_of_registration,
  date_repair = date_of_repair,
  date_report = date_of_report,
  time = op_time,
  status = status,
  distribution = "lognormal",
  details = TRUE
)
```

`mcs_delay_register` *Adjustment of Operating Times by Delays in Registration using a Monte Carlo Approach*

Description

Soft-deprecated

`mcs_delay_register()` is no longer under active development, switching to `mcs_delay` is recommended.

Usage

```
mcs_delay_register(
  date_prod,
  date_register,
  time,
  status,
  distribution = "lognormal",
  details = FALSE
)
```

Arguments

date_prod	A vector of class character or Date, in the format "yyyy-mm-dd", indicating the date of production of a unit. Use NA for missing elements.
date_register	A vector of class character or Date, in the format "yyyy-mm-dd", indicating the date of registration of a unit. Use NA for missing elements.
time	A numeric vector of operating times.
status	A vector of binary data (0 or 1) indicating whether unit i is a right censored observation (= 0) or a failure (= 1).
distribution	Supposed distribution of the random variable. Only "lognormal" is implemented.
details	A logical. If FALSE the output consists of a vector with corrected operating times for the censored units and the input operating times for the failed units. If TRUE the output consists of a detailed list, i.e the same vector as described before, simulated random numbers and estimated distribution parameters.

Details

In general the amount of information about units in the field, that have not failed yet, are rare. For example it is common that a supplier, who provides parts to the automotive industry does not know when a vehicle was put in service and therefore does not know the exact operating time of the supplied parts. This function uses a Monte Carlo approach for simulating the operating times of (multiple) right censored observations, taking account of registering delays. The simulation is based on the distribution of operating times that were calculated from complete data (see dist_delay_register).

Value

A numeric vector of corrected operating times for the censored units and the input operating times for the failed units if details = FALSE. If details = TRUE the output is a list which consists of the following elements:

- time : Numeric vector of corrected operating times for the censored observations and input operating times for failed units.
- x_sim : Simulated random numbers of specified distribution with estimated parameters. The length of x_sim is equal to the number of censored observations.
- coefficients : Estimated coefficients of supposed distribution.

Examples

```
date_of_production   <- c("2014-07-28", "2014-02-17", "2014-07-14",
                          "2014-06-26", "2014-03-10", "2014-05-14",
                          "2014-05-06", "2014-03-07", "2014-03-09",
                          "2014-04-13", "2014-05-20", "2014-07-07",
                          "2014-01-27", "2014-01-30", "2014-03-17",
                          "2014-02-09", "2014-04-14", "2014-04-20",
                          "2014-03-13", "2014-02-23", "2014-04-03",
                          "2014-01-08", "2014-01-08")
date_of_registration <- c(NA, "2014-03-29", "2014-12-06", "2014-09-09",
```

```
                          NA, NA, "2014-06-16", NA, "2014-05-23",
                          "2014-05-09", "2014-05-31", NA, "2014-04-13",
                          NA, NA, "2014-03-12", NA, "2014-06-02",
                          NA, "2014-03-21", "2014-06-19", NA, NA)

op_time <- rep(1000, length(date_of_production))
status <- c(0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0)

# Example 1 - Simplified vector output:
x_corrected <- mcs_delay_register(
  date_prod = date_of_production,
  date_register = date_of_registration,
  time = op_time,
  status = status,
  distribution = "lognormal",
  details = FALSE
)

# Example 2 - Detailed list output:
list_detail <- mcs_delay_register(
  date_prod = date_of_production,
  date_register = date_of_registration,
  time = op_time,
  status = status,
  distribution = "lognormal",
  details = TRUE
)
```

`mcs_delay_report` *Adjustment of Operating Times by Delays in Report using a Monte Carlo Approach*

Description

Soft-deprecated

`mcs_delay_report()` is no longer under active development, switching to `mcs_delay` is recommended.

Usage

```
mcs_delay_report(
  date_repair,
  date_report,
  time,
  status,
  distribution = "lognormal",
  details = FALSE
)
```

Arguments

date_repair	a vector of class character or Date, in the format "yyyy-mm-dd", indicating the date of repair of a failed unit. Use NA for missing elements.
date_report	a vector of class character or Date, in the format "yyyy-mm-dd", indicating the date of report of a failed unit. Use NA for missing elements.
time	A numeric vector of operating times.
status	A vector of binary data (0 or 1) indicating whether unit i is a right censored observation (= 0) or a failure (= 1).
distribution	Supposed distribution of the random variable. Only "lognormal" is implemented.
details	A logical. If FALSE the output consists of a vector with corrected operating times for the censored units and the input operating times for the failed units. If TRUE the output consists of a detailed list, i.e the same vector as described before, simulated random numbers and estimated distribution parameters.

Details

The delay in report describes the time between the occurrence of a damage and the registration in the warranty database. For a given date where the analysis is made there could be units which had a failure but are not registered in the database and therefore treated as censored units. To overcome this problem this function uses a Monte Carlo approach for simulating the operating times of (multiple) right censored observations, taking account of reporting delays. The simulation is based on the distribution of operating times that were calculated from complete data, i.e. failed items (see dist_delay_report).

Value

A numeric vector of corrected operating times for the censored units and the input operating times for the failed units if details = FALSE. If details = TRUE the output is a list which consists of the following elements:

- time : Numeric vector of corrected operating times for the censored observations and input operating times for failed units.
- x_sim : Simulated random numbers of specified distribution with estimated parameters. The length of x_sim is equal to the number of censored observations.
- coefficients : Estimated coefficients of supposed distribution.

Examples

```
date_of_repair <- c(NA, "2014-09-15", "2015-07-04", "2015-04-10", NA,
                    NA, "2015-04-24", NA, "2015-04-25", "2015-04-24",
                     "2015-06-12", NA, "2015-05-04", NA, NA,
                     "2015-05-22", NA, "2015-09-17", NA, "2015-08-15",
                     "2015-11-26", NA, NA)

date_of_report <- c(NA, "2014-10-09", "2015-08-28", "2015-04-15", NA,
                     NA, "2015-05-16", NA, "2015-05-28", "2015-05-15",
                     "2015-07-11", NA, "2015-08-14", NA, NA,
```

```
                    "2015-06-05", NA, "2015-10-17", NA, "2015-08-21",
                    "2015-12-02", NA, NA)

op_time <- rep(1000, length(date_of_repair))
status <- c(0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0)

# Example 1 - Simplified vector output:
x_corrected <- mcs_delay_report(
  date_repair = date_of_repair,
  date_report = date_of_report,
  time = op_time,
  status = status,
  distribution = "lognormal",
  details = FALSE
)

# Example 2 - Detailed list output:
list_detail <- mcs_delay_report(
  date_repair = date_of_repair,
  date_report = date_of_report,
  time = op_time,
  status = status,
  distribution = "lognormal",
  details = TRUE
)
```

`mcs_mileage` *Simulation of Unknown Covered Distances using a Monte Carlo Approach*

Description

This function simulates distances for units where these are unknown, i.e. `mileage = NA`.

First, random numbers of the annual mileage distribution, estimated by `dist_mileage`, are drawn. Second, the drawn annual distances are converted with respect to the actual operating times (in days) using a linear relationship. See 'Details'.

Usage

```
mcs_mileage(
  mileage,
  time,
  status = NULL,
  id = paste0("ID", seq_len(length(time))),
  distribution = c("lognormal", "exponential")
)
```

Arguments

mileage	A numeric vector of distances covered. Use NA for missing elements.
time	A numeric vector of operating times. Use NA for missing elements.
status	Optional argument. If used it has to be a vector of binary data (0 or 1) indicating whether unit i is a right censored observation (= 0) or a failure (= 1). The effect of status on the return is described in 'Value'.
id	A vector for the identification of every unit.
distribution	Supposed distribution of the random variable.

Details

Assumption of linear relationship: Imagine the distance of the vehicle is unknown. A distance of 3500.25 kilometers (km) was drawn from the annual distribution and the known operating time is 200 days (d). So the resulting distance of this vehicle is

$$3500.25km \cdot (\frac{200d}{365d}) = 1917.945km$$

Value

A list containing the following elements:

- data : A tibble with classes wt_mcs_data and wt_reliability_data if status is provided. Since the class wt_reliability_data enables the direct usage of data inside estimate_cdf.wt_reliability_data, the required lifetime characteristic is automatically set to the distance mileage.

  If status = NULL class is wt_mcs_data, which is not supported by estimate_cdf due to missing status.

  The tibble contains the following columns:

  - mileage : Simulated distances for unknown mileage and input distances for known mileage.
  - time : Input operating times.
  - status (**optional**) :
    - If argument status = NULL column status does not exist.
    - If argument status is provided the column contains the entered binary data (0 or 1).
  - id : Identification of every unit.
- sim_data : A tibble with column sim_mileage that holds the simulated distances for unknown mileage and 0 otherwise.
- model_estimation : A list containing a named list ("mileage_distribution") with output of dist_mileage.

See Also

estimate_cdf

Examples

```
# Data for examples:
date_of_registration <- c("2014-08-17", "2014-03-29", "2014-12-06",
                          "2014-09-09", "2014-05-14", "2014-07-01",
                          "2014-06-16", "2014-04-03", "2014-05-23",
                          "2014-05-09", "2014-05-31", "2014-08-12",
                          "2014-04-13", "2014-02-15", "2014-07-07",
                          "2014-03-12", "2014-05-27", "2014-06-02",
                          "2014-05-20", "2014-03-21", "2014-06-19",
                          "2014-02-12", "2014-03-27")
date_of_repair       <- c(NA, "2014-09-15", "2015-07-04", "2015-04-10", NA,
                          NA, "2015-04-24", NA, "2015-04-25", "2015-04-24",
                          "2015-06-12", NA, "2015-05-04", NA, NA, "2015-05-22",
                          NA, "2015-09-17", NA, "2015-08-15", "2015-11-26",
                          NA, NA)
date_of_analysis     <- "2015-12-31"

## Assume that mileage is only known for units that have failed (date_of_repair != NA).
mileage              <- c(NA, 15655, 13629, 18292, NA, NA, 33555, NA, 21737,
                          29870, 21068, NA, 122283, NA, NA, 36088, NA, 11153,
                          NA, 122842, 20349, NA, NA)

## time in service is the difference between repair and registration for failed
## items and the difference between date of analysis and date of registration
## for intact units.
time_in_service <- difftime(
  as.Date(date_of_repair, format = "%Y-%m-%d"),
  as.Date(date_of_registration, format = "%Y-%m-%d"),
  units = "days"
)
time_in_service[is.na(time_in_service)] <- difftime(
  as.Date(date_of_analysis, format = "%Y-%m-%d"),
  as.Date(date_of_registration[is.na(time_in_service)], format = "%Y-%m-%d"),
  units = "days"
)
time_in_service <- as.numeric(time_in_service)

# Example 1 - Reproducibility of drawn random numbers:
set.seed(1234)
mcs_distances <- mcs_mileage(
  mileage = mileage,
  time = time_in_service,
  distribution = "lognormal"
)

# Example 2 - MCS for distances assuming a exponential annual mileage distribution:
mcs_distances_2 <- mcs_mileage(
  mileage = mileage,
  time = time_in_service,
  distribution = "exponential"
)
```

```
status <- ifelse(!is.na(date_of_repair), 1, 0)

# Example 3 - MCS for distances using status:
mcs_distances_3 <- mcs_mileage(
  mileage = mileage,
  time = time_in_service,
  status = status,
  distribution = "lognormal"
)

## Using result of *$data in estimate_cdf()
prob_estimation <- estimate_cdf(
  x = mcs_distances_3$data,
  methods = "kaplan"
)

plot_prob_estimation <- plot_prob(prob_estimation)
```

`mixmod_em` *Weibull Mixture Model Estimation using EM-Algorithm*

Description

This method applies the expectation-maximization (EM) algorithm to estimate the parameters of a univariate Weibull mixture model. See 'Details'.

Usage

```
mixmod_em(x, ...)

## S3 method for class 'wt_reliability_data'
mixmod_em(
  x,
  distribution = "weibull",
  conf_level = 0.95,
  k = 2,
  method = "EM",
  n_iter = 100L,
  conv_limit = 1e-06,
  diff_loglik = 0.01,
  ...
)
```

Arguments

`x`	An object of class `wt_reliability_data` returned from `reliability_data`.
`...`	Further arguments passed to or from other methods. Currently not used.

distribution	"weibull" until further distributions are implemented.
conf_level	Confidence level for the intervals of the Weibull parameters of every component k.
k	Number of mixture components.
method	"EM" until other methods are implemented.
n_iter	Integer defining the maximum number of iterations.
conv_limit	Numeric value defining the convergence limit.
diff_loglik	Numeric value defining the maximum difference between log-likelihood values, which seems permissible.

Details

The EM algorithm is an iterative algorithm for which starting values must be defined. Starting values can be provided for the unknown parameter vector as well as for the posterior probabilities. This implementation employs initial values for the posterior probabilities. These are assigned randomly by using the dirichlet distribution, the conjugate prior of a multinomial distribution (see Mr. Gelissen's blog post listed under *references*).

M-Step : On the basis of the initial posterior probabilities, the parameter vector is estimated with *Newton-Raphson*.

E-Step : The actual estimated parameter vector is used to perform an update of the posterior probabilities.

This procedure is repeated until the complete log-likelihood has converged.

Value

Returns a list with classes wt_model and wt_mixmod_em. The length of the list depends on the number of specified subgroups *k*. The first k lists contain information provided by ml_estimation. The values of logL, aic and bic are the results of a weighted log-likelihood, where the weights are the posterior probabilities determined by the algorithm. The last list summarizes further results of the EM algorithm and is therefore called em_results. It contains the following elements:

- a_priori : A vector with estimated prior probabilities.
- a_posteriori : A matrix with estimated posterior probabilities.
- groups : Numeric vector specifying the group membership of every observation.
- logL : The value of the complete log-likelihood.
- aic : Akaike Information Criterion.
- bic : Bayesian Information Criterion.

References

- Doganaksoy, N.; Hahn, G.; Meeker, W. Q., Reliability Analysis by Failure Mode, Quality Progress, 35(6), 47-52, 2002
- Blog posts by Stefan Gelissen: https://blogs2.datall-analyse.nl/2016/02/18/rcode_mixture_distribution_censored/; last accessed on 8th December 2020

Examples

```
# Reliability data preparation:
## Data for mixture model:
data_mix <- reliability_data(
  voltage,
  x = hours,
  status = status
)

# Example 1 - EM algorithm with k = 2:
mix_mod_em <- mixmod_em(
  x = data_mix,
  conf_level = 0.95,
  k = 2,
  n_iter = 150
)

# Example 2 - Maximum likelihood is applied when k = 1:
mix_mod_em_2 <- mixmod_em(
  x = data_mix,
  conf_level = 0.95,
  k = 1,
  n_iter = 150
)
```

`mixmod_em.default` *Weibull Mixture Model Estimation using EM-Algorithm*

Description

This method applies the expectation-maximization (EM) algorithm to estimate the parameters of a univariate Weibull mixture model. See 'Details'.

Usage

```
## Default S3 method:
mixmod_em(
  x,
  status,
  distribution = "weibull",
  conf_level = 0.95,
  k = 2,
  method = "EM",
  n_iter = 100L,
  conv_limit = 1e-06,
  diff_loglik = 0.01,
  ...
)
```

Arguments

x	A numeric vector which consists of lifetime data. Lifetime data could be every characteristic influencing the reliability of a product, e.g. operating time (days/months in service), mileage (km, miles), load cycles.
status	A vector of binary data (0 or 1) indicating whether unit i is a right censored observation (= 0) or a failure (= 1).
distribution	"weibull" until further distributions are implemented.
conf_level	Confidence level for the intervals of the Weibull parameters of every component k.
k	Number of mixture components.
method	"EM" until other methods are implemented.
n_iter	Integer defining the maximum number of iterations.
conv_limit	Numeric value defining the convergence limit.
diff_loglik	Numeric value defining the maximum difference between log-likelihood values, which seems permissible.
...	Further arguments passed to or from other methods. Currently not used.

Details

The EM algorithm is an iterative algorithm for which starting values must be defined. Starting values can be provided for the unknown parameter vector as well as for the posterior probabilities. This implementation employs initial values for the posterior probabilities. These are assigned randomly by using the dirichlet distribution, the conjugate prior of a multinomial distribution (see Mr. Gelissen's blog post listed under *references*).

M-Step : On the basis of the initial posterior probabilities, the parameter vector is estimated with *Newton-Raphson*.

E-Step : The actual estimated parameter vector is used to perform an update of the posterior probabilities.

This procedure is repeated until the complete log-likelihood has converged.

Value

Returns a list with classes wt_model and wt_mixmod_em. The length of the list depends on the number of specified subgroups k. The first k lists contain information provided by ml_estimation. The values of logL, aic and bic are the results of a weighted log-likelihood, where the weights are the posterior probabilities determined by the algorithm. The last list summarizes further results of the EM algorithm and is therefore called em_results. It contains the following elements:

- a_priori : A vector with estimated prior probabilities.
- a_posteriori : A matrix with estimated posterior probabilities.
- groups : Numeric vector specifying the group membership of every observation.
- logL : The value of the complete log-likelihood.
- aic : Akaike Information Criterion.
- bic : Bayesian Information Criterion.

References

- Doganaksoy, N.; Hahn, G.; Meeker, W. Q., Reliability Analysis by Failure Mode, Quality Progress, 35(6), 47-52, 2002
- Blog posts by Stefan Gelissen: https://blogs2.datall-analyse.nl/2016/02/18/rcode_mixture_distribution_censored/; last accessed on 8th December 2020

See Also

mixmod_em

Examples

```
# Vectors:
hours <- voltage$hours
status <- voltage$status

# Example 1 - EM algorithm with k = 2:
mix_mod_em <- mixmod_em(
  x = hours,
  status = status,
  distribution = "weibull",
  conf_level = 0.95,
  k = 2,
  n_iter = 150
)

#' # Example 2 - Maximum likelihood is applied when k = 1:
mix_mod_em_2 <- mixmod_em(
  x = hours,
  status = status,
  distribution = "weibull",
  conf_level = 0.95,
  k = 1,
  method = "EM",
  n_iter = 150
)
```

---

mixmod_regression *Mixture Model Identification using Segmented Regression*

Description

This function uses piecewise linear regression to divide the data into subgroups. See 'Details'.

Usage

```
mixmod_regression(x, ...)

## S3 method for class 'wt_cdf_estimation'
mixmod_regression(
  x,
  distribution = c("weibull", "lognormal", "loglogistic"),
  conf_level = 0.95,
  k = 2,
  control = segmented::seg.control(),
  ...
)
```

Arguments

x	Object of class wt_cdf_estimation returned from estimate_cdf.
...	Further arguments passed to or from other methods. Currently not used.
distribution	Supposed distribution of the random variable.
conf_level	Confidence level of the interval. If distribution is "weibull" this must be one of 0.9, 0.95 or 0.99.
k	Number of mixture components. If the data should be split in an automated fashion, k must be set to NULL. The argument fix.psi of control is then set to FALSE.
control	Output of the call to seg.control, which is passed to segmented.lm. See 'Examples' for usage.

Details

The segmentation process is based on the lifetime realizations of failed units and their corresponding estimated failure probabilities for which intact items are taken into account. It is performed with the support of segmented.lm.

Segmentation can be done with a specified number of subgroups or in an automated fashion (see argument k). The algorithm tends to overestimate the number of breakpoints when the separation is done automatically (see 'Warning' in segmented.lm).

In the context of reliability analysis it is important that the main types of failures can be identified and analyzed separately. These are

- early failures,
- random failures and
- wear-out failures.

In order to reduce the risk of overestimation as well as being able to consider the main types of failures, a maximum of three subgroups (k = 3) is recommended.

Value

Returns a list with classes `wt_model` and `wt_rank_regression` if no breakpoint was detected. See `rank_regression`.

Returns a list with classes `wt_model` and `wt_mixmod_regression` if at least one breakpoint was determined. The length of the list depends on the number of identified subgroups. Each list element contains the information provided by `rank_regression`. In addition, the returned tibble `data` of each list element only retains information on the failed units and has two more columns:

- `q` : Quantiles of the standard distribution calculated from column `prob`.
- `group` : Membership to the respective segment.

If more than one method was specified in `estimate_cdf`, the resulting output is a list with classes `wt_model` and `wt_mixmod_regression_list` where each list element has class `wt_model` and `wt_mixmod_regression`.

References

Doganaksoy, N.; Hahn, G.; Meeker, W. Q., Reliability Analysis by Failure Mode, Quality Progress, 35(6), 47-52, 2002

Examples

```
# Reliability data preparation:
## Data for mixture model:
data_mix <- reliability_data(
  voltage,
  x = hours,
  status = status
)

## Data for simple unimodal distribution:
data <- reliability_data(
  shock,
  x = distance,
  status = status
)

# Probability estimation with one method:
prob_mix <- estimate_cdf(
  data_mix,
  methods = "johnson"
)

prob <- estimate_cdf(
  data,
  methods = "johnson"
)

# Probability estimation for multiple methods:
prob_mix_mult <- estimate_cdf(
  data_mix,
```

```
  methods = c("johnson", "kaplan", "nelson")
)

# Example 1 - Mixture identification using k = 2 two-parametric Weibull models:
mix_mod_weibull <- mixmod_regression(
  x = prob_mix,
  distribution = "weibull",
  conf_level = 0.99,
  k = 2
)

# Example 2 - Mixture identification using k = 3 two-parametric lognormal models:
mix_mod_lognorm <- mixmod_regression(
  x = prob_mix,
  distribution = "lognormal",
  k = 3
)

# Example 3 - Mixture identification for multiple methods specified in estimate_cdf:
mix_mod_mult <- mixmod_regression(
  x = prob_mix_mult,
  distribution = "loglogistic"
)

# Example 4 - Mixture identification using control argument:
mix_mod_control <- mixmod_regression(
  x = prob_mix,
  distribution = "weibull",
  control = segmented::seg.control(display = TRUE)
)

# Example 5 - Mixture identification performs rank_regression for k = 1:
mod <- mixmod_regression(
  x = prob,
  distribution = "weibull",
  k = 1
)
```

`mixmod_regression.default` *Mixture Model Identification using Segmented Regression*

Description

This function uses piecewise linear regression to divide the data into subgroups. See 'Details'.

Usage

```
## Default S3 method:
```

```
mixmod_regression(
  x,
  y,
  status,
  distribution = c("weibull", "lognormal", "loglogistic"),
  conf_level = 0.95,
  k = 2,
  control = segmented::seg.control(),
  ...
)
```

Arguments

x	A numeric vector which consists of lifetime data. Lifetime data could be every characteristic influencing the reliability of a product, e.g. operating time (days/months in service), mileage (km, miles), load cycles.
y	A numeric vector which consists of estimated failure probabilities regarding the lifetime data in x.
status	A vector of binary data (0 or 1) indicating whether a unit is a right censored observation (= 0) or a failure (= 1).
distribution	Supposed distribution of the random variable.
conf_level	Confidence level of the interval. If distribution is "weibull" this must be one of 0.9, 0.95 or 0.99.
k	Number of mixture components. If the data should be split in an automated fashion, k must be set to NULL. The argument fix.psi of control is then set to FALSE.
control	Output of the call to seg.control, which is passed to segmented.lm. See 'Examples' for usage.
...	Further arguments passed to or from other methods. Currently not used.

Details

The segmentation process is based on the lifetime realizations of failed units and their corresponding estimated failure probabilities for which intact items are taken into account. It is performed with the support of segmented.lm.

Segmentation can be done with a specified number of subgroups or in an automated fashion (see argument k). The algorithm tends to overestimate the number of breakpoints when the separation is done automatically (see 'Warning' in segmented.lm).

In the context of reliability analysis it is important that the main types of failures can be identified and analyzed separately. These are

- early failures,
- random failures and
- wear-out failures.

In order to reduce the risk of overestimation as well as being able to consider the main types of failures, a maximum of three subgroups (k = 3) is recommended.

Value

Returns a list of class `wt_rank_regression` if no breakpoint was detected. See `rank_regression`. The tibble `data` is returned with class `wt_cdf_estimation` and contains the additional dummy columns `method` and `id`. The former is filled with `NA_character`, due to generic visualization functions and the latter is filled with `"XXXXXX"` to point out that unit identification is not possible when using the vector-based approach.

Returns a list of class `wt_mixmod_regression` if at least one breakpoint was determined. The length of the list depends on the number of identified subgroups. Each list contains the information provided by `rank_regression`. The returned tibble `data` of each list element only retains information on the failed units and has modified and additional columns:

- `id` : Modified id, overwritten with `"XXXXXX"` to point out that unit identification is not possible when using the vector-based approach.
- `method` : A character that is always `"_null"`. Due to generic visualization functions column `method` has to be provided.
- `q` : Quantiles of the standard distribution calculated from column `prob`.
- `group` : Membership to the respective segment.

References

Doganaksoy, N.; Hahn, G.; Meeker, W. Q., Reliability Analysis by Failure Mode, Quality Progress, 35(6), 47-52, 2002

See Also

`mixmod_regression`

Examples

```
# Vectors:
## Data for mixture model:
hours <- voltage$hours
status <- voltage$status

## Data for simple unimodal distribution:
distance <- shock$distance
status_2 <- shock$status

# Probability estimation with one method:
prob_mix <- estimate_cdf(
  x = hours,
  status = status,
  method = "johnson"
)

prob <- estimate_cdf(
  x = distance,
  status = status_2,
  method = "johnson"
```

```
)

 # Example 1 - Mixture identification using k = 2 two-parametric Weibull models:
mix_mod_weibull <- mixmod_regression(
   x = prob_mix$x,
   y = prob_mix$prob,
   status = prob_mix$status,
   distribution = "weibull",
   conf_level = 0.99,
   k = 2
)

# Example 2 - Mixture identification using k = 3 two-parametric lognormal models:
mix_mod_lognorm <- mixmod_regression(
   x = prob_mix$x,
   y = prob_mix$prob,
   status = prob_mix$status,
   distribution = "lognormal",
   k = 3
)

# Example 3 - Mixture identification using control argument:
mix_mod_control <- mixmod_regression(
  x = prob_mix$x,
  y = prob_mix$prob,
  status = prob_mix$status,
  distribution = "weibull",
  k = 2,
  control = segmented::seg.control(display = TRUE)
)

# Example 4 - Mixture identification performs rank_regression for k = 1:
mod <- mixmod_regression(
  x = prob$x,
  y = prob$prob,
  status = prob$status,
  distribution = "weibull",
  k = 1
)
```

`ml_estimation` *ML Estimation for Parametric Lifetime Distributions*

Description

This function estimates the parameters of a two- or three-parametric lifetime distribution for complete and (multiple) right censored data. The parameters are determined in the frequently used (log-)location-scale parameterization.

For the Weibull, estimates are additionally transformed such that they are in line with the parameterization provided by the *stats* package (see Weibull).

Usage

```
ml_estimation(x, ...)

## S3 method for class 'wt_reliability_data'
ml_estimation(
  x,
 distribution = c("weibull", "lognormal", "loglogistic", "normal", "logistic", "sev",
    "weibull3", "lognormal3", "loglogistic3"),
  wts = rep(1, nrow(x)),
  conf_level = 0.95,
  ...
)
```

Arguments

x	Object of class wt_reliability_data returned by reliability_data.
...	Further arguments passed to or from other methods. Currently not used.
distribution	Supposed distribution of the random variable.
wts	Optional vector of case weights. The length of wts must be the same as the number of observations in x.
conf_level	Confidence level of the interval.

Details

ml_estimation calls Lifedata.MLE, which is implemented in *SPREDA*, to obtain the estimates. Normal approximation confidence intervals for the parameters are computed as well.

Value

Returns a list with classes wt_model, wt_ml_estimation and wt_model_estimation containing the following elements:

- coefficients : A named vector of estimated coefficients (parameters of the assumed distribution). **Note**: The parameters are given in location-scale-parameterization.
- confint : Confidence intervals for parameters.
- varcov : Estimated variance-covariance matrix for the parameters.
- shape_scale_coefficients : Only included if distribution is "weibull" or "weibull3" (parameterization used in stats::Weibull).
- shape_scale_confint : Only included if distribution is "weibull" or "weibull3". Confidence intervals for scale η and shape β (and threshold γ) if distribution is "weibull3".
- logL : The log-likelihood value.
- aic : Akaike Information Criterion.
- bic : Bayesian Information Criterion.
- data : A tibble with class wt_reliability_data returned from reliability_data.
- distribution : Specified distribution.

References

Meeker, William Q; Escobar, Luis A., Statistical methods for reliability data, New York: Wiley series in probability and statistics, 1998

Examples

```
# Reliability data preparation:
## Data for two-parametric model:
data_2p <- reliability_data(
  shock,
  x = distance,
  status = status
)

## Data for three-parametric model:
data_3p <- reliability_data(
  alloy,
  x = cycles,
  status = status
)

# Example 1 - Fitting a two-parametric weibull distribution:
ml_2p <- ml_estimation(
  data_2p,
  distribution = "weibull"
)

# Example 2 - Fitting a three-parametric lognormal distribution:
ml_3p <- ml_estimation(
  data_3p,
  distribution = "lognormal3",
  conf_level = 0.99
)
```

---

ml_estimation.default *ML Estimation for Parametric Lifetime Distributions*

Description

This function estimates the parameters of a two- or three-parametric lifetime distribution for complete and (multiple) right censored data. The parameters are determined in the frequently used (log-)location-scale parameterization.

For the Weibull, estimates are additionally transformed such that they are in line with the parameterization provided by the *stats* package (see Weibull).

Usage

```
## Default S3 method:
ml_estimation(
  x,
  status,
 distribution = c("weibull", "lognormal", "loglogistic", "normal", "logistic", "sev",
    "weibull3", "lognormal3", "loglogistic3"),
  wts = rep(1, length(x)),
  conf_level = 0.95,
  ...
)
```

Arguments

x	A numeric vector which consists of lifetime data. Lifetime data could be every characteristic influencing the reliability of a product, e.g. operating time (days/months in service), mileage (km, miles), load cycles.
status	A vector of binary data (0 or 1) indicating whether unit i is a right censored observation (= 0) or a failure (= 1).
distribution	Supposed distribution of the random variable.
wts	Optional vector of case weights. The length of wts must be the same as the number of observations in x.
conf_level	Confidence level of the interval.
...	Further arguments passed to or from other methods. Currently not used.

Details

ml_estimation calls Lifedata.MLE, which is implemented in *SPREDA*, to obtain the estimates. Normal approximation confidence intervals for the parameters are computed as well.

Value

Returns a list with classes wt_model, wt_ml_estimation and wt_model_estimation containing the following elements:

- coefficients : A named vector of estimated coefficients (parameters of the assumed distribution). **Note**: The parameters are given in location-scale-parameterization.
- confint : Confidence intervals for parameters.
- varcov : Estimated variance-covariance matrix for the parameters.
- shape_scale_coefficients : Only included if distribution is "weibull" or "weibull3" (parameterization used in stats::Weibull).
- shape_scale_confint : Only included if distribution is "weibull" or "weibull3". Confidence intervals for scale η and shape β (and threshold γ) if distribution is "weibull3".
- logL : The log-likelihood value.
- aic : Akaike Information Criterion.

- `bic` : Bayesian Information Criterion.
- `data` : A tibble with class `wt_reliability_data` returned from `reliability_data`.
- `distribution` : Specified distribution.

References

Meeker, William Q; Escobar, Luis A., Statistical methods for reliability data, New York: Wiley series in probability and statistics, 1998

See Also

`ml_estimation`

Examples

```
# Vectors:
obs <- seq(10000, 100000, 10000)
status_1 <- c(0, 1, 1, 0, 0, 0, 1, 0, 1, 0)

cycles <- alloy$cycles
status_2 <- alloy$status

# Example 1 - Fitting a two-parametric weibull distribution:
ml <- ml_estimation(
  x = obs,
  status = status_1,
  distribution = "weibull",
  conf_level = 0.90
)

# Example 2 - Fitting a three-parametric lognormal distribution:
ml_2 <- ml_estimation(
  x = cycles,
  status = status_2,
  distribution = "lognormal3"
)
```

---

mr_method *Estimation of Failure Probabilities using Median Ranks*

Description

Soft-deprecated

`mr_method()` is no longer under active development, switching to `estimate_cdf` is recommended.

Usage

```
mr_method(
  x,
  status = rep(1, length(x)),
  id = NULL,
  method = c("benard", "invbeta"),
  ties.method = c("max", "min", "average")
)
```

Arguments

x	A numeric vector which consists of lifetime data. Lifetime data could be every characteristic influencing the reliability of a product, e.g. operating time (days/months in service), mileage (km, miles), load cycles.
status	A vector of ones indicating that every unit *i* has failed.
id	A vector for the identification of every unit. Default is NULL.
method	Method for the estimation of the cdf. Can be "benard" (default) or "invbeta".
ties.method	A character string specifying how ties are treated, default is "max".

Details

This non-parametric approach (*Median Ranks*) is used to estimate the failure probabilities in terms of complete data. Two methods are available to estimate the cumulative distribution function *F(t)*:

- "benard" : Benard's approximation for Median Ranks.
- "invbeta" : Exact Median Ranks using the inverse beta distribution.

Value

A tibble with failed units containing the following columns:

- id : Identification for every unit.
- x : Lifetime characteristic.
- status : Status of failed units (always 1).
- rank : The assigned ranks.
- prob : Estimated failure probabilities.
- method : Specified method for the estimation of failure probabilities (always 'mr').

Examples

```
# Vectors:
obs   <- seq(10000, 100000, 10000)
state <- rep(1, length(obs))
uic   <- c("3435", "1203", "958X", "XX71", "abcd", "tz46",
           "fl29", "AX23", "Uy12", "kl1a")

# Example 1 - Benard's approximation:
```

```
tbl_mr <- mr_method(
  x = obs,
  status = state,
  id = uic,
  method = "benard"
)

# Example 2 - Inverse beta distribution:
tbl_mr_invbeta <- mr_method(
  x = obs,
  status = state,
  method = "invbeta"
)
```

`nelson_method` *Estimation of Failure Probabilities using the Nelson-Aalen Estimator*

Description

Soft-deprecated

`nelson_method()` is no longer under active development, switching to `estimate_cdf` is recommended.

Usage

```
nelson_method(x, status, id = NULL)
```

Arguments

`x`	A numeric vector which consists of lifetime data. Lifetime data could be every characteristic influencing the reliability of a product, e.g. operating time (days/months in service), mileage (km, miles), load cycles.
`status`	A vector of binary data (0 or 1) indicating whether unit *i* is a right censored observation (= 0) or a failure (= 1).
`id`	A vector for the identification of every unit. Default is `NULL`.

Details

This non-parametric approach estimates the cumulative hazard rate in terms of (multiple) right censored data. By equating the definition of the hazard rate with the hazard rate according to Nelson-Aalen one can calculate the failure probabilities.

Note : The *Nelson-Aalen* estimator does not assign ranks to observations, so the beta-binomial confidence intervals *cannot* be calculated using this method.

Value

A tibble containing the following columns:

- `id` : Identification for every unit.
- `x` : Lifetime characteristic.
- `status` : Binary data (0 or 1) indicating whether a unit is a right censored observation (= 0) or a failure (= 1).
- `rank` : Filled with `NA`.
- `prob` : Estimated failure probabilities, `NA` if `status` = `0`.
- `cdf_estimation_method` : Specified method for the estimation of failure probabilities (always 'nelson').

Examples

```
# Vectors:
obs   <- seq(10000, 100000, 10000)
state <- c(0, 1, 1, 0, 0, 0, 1, 0, 1, 0)
uic   <- c("3435", "1203", "958X", "XX71", "abcd", "tz46",
           "f129", "AX23","Uy12", "kl1a")

# Example - Nelson-Aalen estimator applied to intact and failed units:
tbl_nel <- nelson_method(
  x = obs,
  status = state,
  id = uic
)
```

`plot_conf` *Add Confidence Region(s) for Quantiles and Probabilities*

Description

This function is used to add estimated confidence region(s) to an existing probability plot. Since confidence regions are related to the estimated regression line, the latter is provided as well.

Usage

```
plot_conf(p_obj, x, ...)

## S3 method for class 'wt_confint'
plot_conf(
  p_obj,
  x,
  title_trace_mod = "Fit",
  title_trace_conf = "Confidence Limit",
  ...
)
```

Arguments

p_obj	A plot object returned from plot_prob.
x	Confidence interval as returned by confint_betabinom or confint_fisher.
...	Further arguments passed to or from other methods. Currently not used.
title_trace_mod	A character string which is assigned to the mod trace in the legend.
title_trace_conf	A character string which is assigned to the conf trace in the legend.

Value

Returns a plot object containing the probability plot with plotting positions, the estimated regression line and the estimated confidence region(s).

References

Meeker, William Q; Escobar, Luis A., Statistical methods for reliability data, New York: Wiley series in probability and statistics, 1998

Examples

```
# Reliability data:
data <- reliability_data(data = alloy, x = cycles, status = status)

# Probability estimation:
prob_tbl <- estimate_cdf(data, methods = "johnson")

# Example 1 - Probability Plot, Regression Line and Confidence Bounds for Three-Parameter-Weibull:
rr <- rank_regression(prob_tbl, distribution = "weibull3")

conf_betabin <- confint_betabinom(rr)

plot_weibull <- plot_prob(prob_tbl, distribution = "weibull")

plot_conf_beta <- plot_conf(
  p_obj = plot_weibull,
  x = conf_betabin
)

# Example 2 - Probability Plot, Regression Line and Confidence Bounds for Three-Parameter-Lognormal:
rr_ln <- rank_regression(
  prob_tbl,
  distribution = "lognormal3",
  conf_level = 0.9
)

conf_betabin_ln <- confint_betabinom(
  rr_ln,
  bounds = "two_sided",
  conf_level = 0.9,
```

```
  direction = "y"
)

plot_lognormal <- plot_prob(prob_tbl, distribution = "lognormal")

plot_conf_beta_ln <- plot_conf(
  p_obj = plot_lognormal,
  x = conf_betabin_ln
)

# Example 3 - Probability Plot, Regression Line and Confidence Bounds for MLE
ml <- ml_estimation(data, distribution = "weibull")

conf_fisher <- confint_fisher(ml)

plot_weibull <- plot_prob(prob_tbl, distribution = "weibull")

plot_conf_fisher_weibull <- plot_conf(
  p_obj = plot_weibull,
  x = conf_fisher
)
```

`plot_conf.default` *Add Confidence Region(s) for Quantiles and Probabilities*

Description

This function is used to add estimated confidence region(s) to an existing probability plot which also includes the estimated regression line.

Usage

```
## Default S3 method:
plot_conf(
  p_obj,
  x,
  y,
 distribution = c("weibull", "lognormal", "loglogistic", "normal", "logistic", "sev",
    "weibull3", "lognormal3", "loglogistic3"),
  direction = c("y", "x"),
  title_trace = "Confidence Limit",
  ...
)
```

Arguments

`p_obj`	A plot object returned from `plot_mod`.

x	A list containing the x-coordinates of the confidence region(s). The list can be of length 1 or 2. For more information see **Details**.
y	A list containing the y-coordinates of the Confidence Region(s). The list can be of length 1 or 2. For more information see **Details**.
distribution	Supposed distribution of the random variable.
direction	A character string specifying the direction of the plotted interval(s). Must be either "y" (failure probabilities) or "x" (quantiles).
title_trace	A character string which is assigned to the trace shown in the legend.
...	Further arguments passed to or from other methods. Currently not used.

Details

It is important that the length of the vectors provided as lists in x and y match with the length of the vectors x and y in the function plot_mod. For this reason the following procedure is recommended:

- Calculate confidence intervals with the function confint_betabinom or confint_fisher and store it in a data.frame. For instance call it df.
- Inside plot_mod use the output df$x for x and df$prob for y of the function(s) named before.
- In **Examples** the described approach is shown with code.

Value

Returns a plot object containing the probability plot with plotting positions, the estimated regression line and the estimated confidence region(s).

References

Meeker, William Q; Escobar, Luis A., Statistical methods for reliability data, New York: Wiley series in probability and statistics, 1998

See Also

plot_conf

Examples

```
# Vectors:
cycles   <- alloy$cycles
status <- alloy$status

prob_tbl <- estimate_cdf(x = cycles, status = status, method = "johnson")

# Example 1 - Probability Plot, Regression Line and Confidence Bounds for Three-Parameter-Weibull:
rr <- rank_regression(
  x = prob_tbl$x,
  y = prob_tbl$prob,
  status = prob_tbl$status,
  distribution = "weibull3"
)
```

```
conf_betabin <- confint_betabinom(
  x = prob_tbl$x,
  status = prob_tbl$status,
  dist_params = rr$coefficients,
  distribution = "weibull3"
)

plot_weibull <- plot_prob(
  x = prob_tbl$x,
  y = prob_tbl$prob,
  status = prob_tbl$status,
  id = prob_tbl$id,
  distribution = "weibull"
)

plot_reg_weibull <- plot_mod(
  p_obj = plot_weibull,
  x = conf_betabin$x,
  y = conf_betabin$prob,
  dist_params = rr$coefficients,
  distribution = "weibull3"
)

plot_conf_beta <- plot_conf(
  p_obj = plot_reg_weibull,
  x = list(conf_betabin$x),
  y = list(conf_betabin$lower_bound, conf_betabin$upper_bound),
  direction = "y",
  distribution = "weibull3"
)

# Example 2 - Probability Plot, Regression Line and Confidence Bounds for Three-Parameter-Lognormal:
rr_ln <- rank_regression(
  x = prob_tbl$x,
  y = prob_tbl$prob,
  status = prob_tbl$status,
  distribution = "lognormal3"
)

conf_betabin_ln <- confint_betabinom(
  x = prob_tbl$x,
  status = prob_tbl$status,
  dist_params = rr_ln$coefficients,
  distribution = "lognormal3"
)

plot_lognormal <- plot_prob(
  x = prob_tbl$x,
  y = prob_tbl$prob,
  status = prob_tbl$status,
  id = prob_tbl$id,
  distribution = "lognormal"
```

```
)

plot_reg_lognormal <- plot_mod(
  p_obj = plot_lognormal,
  x = conf_betabin_ln$x,
  y = conf_betabin_ln$prob,
  dist_params = rr_ln$coefficients,
  distribution = "lognormal3"
)

plot_conf_beta_ln <- plot_conf(
  p_obj = plot_reg_lognormal,
  x = list(conf_betabin_ln$x),
  y = list(conf_betabin_ln$lower_bound, conf_betabin_ln$upper_bound),
  direction = "y",
  distribution = "lognormal3"
)
```

`plot_mod` *Add Estimated Population Line(s) to a Probability Plot*

Description

This function adds one or multiple estimated regression lines to an existing probability plot (`plot_prob`). Depending on the output of the functions `rank_regression`, `ml_estimation`, `mixmod_regression` or `mixmod_em` one or multiple lines are plotted.

Usage

```
plot_mod(p_obj, x, ...)

## S3 method for class 'wt_model'
plot_mod(p_obj, x, title_trace = "Fit", ...)
```

Arguments

`p_obj`	A plot object returned from `plot_prob`.
`x`	An object of class `wt_model`.
`...`	Further arguments passed to or from other methods. Currently not used.
`title_trace`	A character string which is assigned to the trace shown in the legend.

Details

The name of the legend entry is a combination of the `title_trace` and the number of determined subgroups from `mixmod_regression` or `mixmod_em`. If `title_trace = "Line"` and the data could be split in two groups, the legend entries would be "Line: 1" and "Line: 2".

Value

Returns a plot object containing the probability plot with plotting positions and the estimated regression line(s).

References

Meeker, William Q; Escobar, Luis A., Statistical methods for reliability data, New York: Wiley series in probability and statistics, 1998

Examples

```
# Reliability data:
data <- reliability_data(data = alloy, x = cycles, status = status)

# Probability estimation:
prob_tbl <- estimate_cdf(data, methods = c("johnson", "kaplan"))


## Rank Regression
# Example 1 - Probability Plot and Regression Line Three-Parameter-Weibull:
plot_weibull <- plot_prob(prob_tbl, distribution = "weibull")
rr_weibull <- rank_regression(prob_tbl, distribution = "weibull3")

plot_reg_weibull <- plot_mod(p_obj = plot_weibull, x = rr_weibull)

# Example 2 - Probability Plot and Regression Line Three-Parameter-Lognormal:
plot_lognormal <- plot_prob(prob_tbl, distribution = "lognormal")
rr_lognormal <- rank_regression(prob_tbl, distribution = "lognormal3")

plot_reg_lognormal <- plot_mod(p_obj = plot_lognormal, x = rr_lognormal)


## ML Estimation
# Example 3 - Probability Plot and Regression Line Two-Parameter-Weibull:
plot_weibull <- plot_prob(prob_tbl, distribution = "weibull")
ml_weibull_2 <- ml_estimation(data, distribution = "weibull")

plot_reg_weibull_2 <- plot_mod(p_obj = plot_weibull, ml_weibull_2)


## Mixture Identification
# Reliability data:
data_mix <- reliability_data(voltage, x = hours, status = status)

# Probability estimation:
prob_mix <- estimate_cdf(
  data_mix,
  methods = c("johnson", "kaplan", "nelson")
)

# Example 4 - Probability Plot and Regression Line Mixmod Regression:
mix_mod_rr <- mixmod_regression(prob_mix, distribution = "weibull")
```

```
plot_weibull <- plot_prob(mix_mod_rr)

plot_reg_mix_mod_rr <- plot_mod(p_obj = plot_weibull, x = mix_mod_rr)

# Example 5 - Probability Plot and Regression Line Mixmod EM:
mix_mod_em <- mixmod_em(data_mix)
plot_weibull <- plot_prob(mix_mod_em)

plot_reg_mix_mod_em <- plot_mod(p_obj = plot_weibull, x = mix_mod_em)
```

`plot_mod.default` *Add Estimated Population Line to a Probability Plot*

Description

This function adds an estimated regression lines to an existing probability plot (`plot_prob`).

Usage

```
## Default S3 method:
plot_mod(
  p_obj,
  x,
  dist_params,
 distribution = c("weibull", "lognormal", "loglogistic", "normal", "logistic", "sev",
    "weibull3", "lognormal3", "loglogistic3"),
  title_trace = "Fit",
  ...
)
```

Arguments

`p_obj`	A plot object returned from `plot_prob`.
`x`	A numeric vector containing the x-coordinates of the respective regression line.
`dist_params`	A (named) numeric vector of estimated location and scale parameters for a specified distribution. The order of elements is important. First entry needs to be the location parameter μ and the second element needs to be the scale parameter σ. If a three-parametric model is used the third element is the threshold parameter γ.
`distribution`	Supposed distribution of the random variable.
`title_trace`	A character string which is assigned to the trace shown in the legend.
`...`	Further arguments passed to or from other methods. Currently not used.

Value

Returns a plot object containing the probability plot with plotting positions and the estimated regression line(s).

References

Meeker, William Q; Escobar, Luis A., Statistical methods for reliability data, New York: Wiley series in probability and statistics, 1998

See Also

plot_mod

Examples

```
# Vectors:
cycles <- alloy$cycles
status <- alloy$status

# Probability estimation
prob_tbl <- estimate_cdf(x = cycles, status = status, method = "johnson")

# Example 1: Probability Plot and Regression Line Three-Parameter-Weibull:
plot_weibull <- plot_prob(
  x = prob_tbl$x,
  y = prob_tbl$prob,
  status = prob_tbl$status,
  id = prob_tbl$id,
  distribution = "weibull"
)

rr <- rank_regression(
  x = prob_tbl$x,
  y = prob_tbl$prob,
  status = prob_tbl$status,
  distribution = "weibull3"
)

plot_reg_weibull <- plot_mod(
  p_obj = plot_weibull,
  x = prob_tbl$x,
  dist_params = rr$coefficients,
  distribution = "weibull3"
)



# Example 2: Probability Plot and Regression Line Three-Parameter-Lognormal:
plot_lognormal <- plot_prob(
  x = prob_tbl$x,
  y = prob_tbl$prob,
  status = prob_tbl$status,
  id = prob_tbl$id,
  distribution = "lognormal"
)

rr_ln <- rank_regression(
```

```
  x = prob_tbl$x,
  y = prob_tbl$prob,
  status = prob_tbl$status,
  distribution = "lognormal3"
)

plot_reg_lognormal <- plot_mod(
  p_obj = plot_lognormal,
  x = prob_tbl$x,
  dist_params = rr_ln$coefficients,
  distribution = "lognormal3"
)

## Mixture Identification
# Vectors:
hours <- voltage$hours
status <- voltage$status

# Probability estimation:
prob_mix <- estimate_cdf(
  x = hours,
  status = status,
  method = "johnson"
)
```

`plot_mod_mix` *Add Estimated Population Lines of a Separated Mixture Model to a Probability Plot*

Description

Soft-deprecated

`plot_mod_mix()` is no longer under active development, switching to `plot_mod` is recommended.

Usage

```
plot_mod_mix(
  p_obj,
  x,
  status,
  mix_output,
  distribution = c("weibull", "lognormal", "loglogistic"),
  title_trace = "Fit",
  ...
)
```

Arguments

p_obj	A plot object provided by function plot_prob_mix.
x	A numeric vector containing the x-coordinates of the respective regression line.
status	A vector of binary data (0 or 1) indicating whether unit i is a right censored observation (= 0) or a failure (= 1).
mix_output	A list provided by mixmod_regression or mixmod_em, which consists of elements necessary to visualize the regression lines.
distribution	Supposed distribution of the random variable.
title_trace	A character string which is assigned to the trace shown in the legend.
...	Further arguments passed to or from other methods. Currently not used.

Details

This function adds one or multiple estimated regression lines to an existing probability plot (plot_prob). Depending on the output of the function mixmod_regression or mixmod_em one or multiple lines are plotted.

The name of the legend entry is a combination of the title_trace and the number of determined subgroups. If title_trace = "Line" and the data has been split in two groups, the legend entries would be "Line: 1" and "Line: 2".

Value

Returns a plot object containing the probability plot with plotting positions and estimated regression line(s).

References

Doganaksoy, N.; Hahn, G.; Meeker, W. Q., Reliability Analysis by Failure Mode, Quality Progress, 35(6), 47-52, 2002

Examples

```
# Vectors:
hours <- voltage$hours
status <- voltage$status

# Example 1 - Using result of mixmod_em in mix_output:
mix_mod_em <- mixmod_em(
  x = hours,
  status = status,
  distribution = "weibull",
  conf_level = 0.95,
  k = 2,
  method = "EM",
  n_iter = 150
)

plot_weibull_em <- plot_prob_mix(
```

```
  x = hours,
  status = status,
  id = id,
  distribution = "weibull",
  mix_output = mix_mod_em
)

plot_weibull_emlines <- plot_mod_mix(
  p_obj = plot_weibull_em,
  x = hours,
  status = status,
  mix_output = mix_mod_em,
  distribution = "weibull"
)

# Example 2 - Using result of mixmod_regression in mix_output:
john <- johnson_method(x = hours, status = status)
mix_mod_reg <- mixmod_regression(
  x = john$x,
  y = john$prob,
  status = john$status,
  distribution = "weibull"
)

plot_weibull_reg <- plot_prob_mix(
  x = john$x,
  status = john$status,
  id = john$id,
  distribution = "weibull",
  mix_output = mix_mod_reg,
)

plot_weibull_reglines <- plot_mod_mix(
  p_obj = plot_weibull_reg,
  x = john$x,
  status = john$status,
  mix_output = mix_mod_reg,
  distribution = "weibull"
)
```

`plot_pop` *Add Population Line(s) to an Existing Grid*

Description

This function adds one or multiple linearized CDFs to an existing plot grid.

Usage

```
plot_pop(
  p_obj = NULL,
  x,
  dist_params_tbl,
 distribution = c("weibull", "lognormal", "loglogistic", "normal", "logistic", "sev"),
  tol = 1e-06,
  title_trace = "Population",
  plot_method = c("plotly", "ggplot2")
)
```

Arguments

p_obj	A plot object to which the population lines are added or NULL. If NULL the population lines are plotted in an empty grid.
x	A numeric vector of length two or greater used for the x coordinates of the population line. If length(x) == 2 a sequence of length 200 between x[1] and x[2] is created. This sequence is equidistant with respect to the scale of the x axis. If length(x) > 2 the elements of x are the x coordinates of the population line.
dist_params_tbl	A tibble. See 'Details'.
distribution	Supposed distribution of the random variable. In the context of this function "weibull", "lognormal" and "loglogistic" stand for the two- **and** the three-parametric version of the respective distribution. The distinction is made via dist_params_tbl.
tol	The failure probability is restricted to the interval $[tol, 1-tol]$. The default value is in accordance with the decimal places shown in the hover for plot_method = "plotly".
title_trace	A character string which is assigned to the trace shown in the legend.
plot_method	Plot package, which produces the visual output. Only used with p_obj = NULL, otherwise p_obj is used to determine the plot method.

Details

dist_params_tbl is a data.frame with two or three columns. For location-scale distributions the first column contains the location parameter and the second column contains the scale parameter. For three-parametric distributions the third column contains the threshold parameter.

If only one population line should be displayed, a numeric vector of length two or three is also supported (c(loc,sc) or c(loc,sc,thres)).

Value

A plot object which contains the linearized CDF(s).

Examples

```
x <- rweibull(n = 100, shape = 1, scale = 20000)

# Example 1 - Two-parametric straight line:
pop_weibull <- plot_pop(
  p_obj = NULL,
  x = range(x),
  dist_params_tbl = c(log(20000), 1),
  distribution = "weibull"
)

# Example 2 - Three-parametric curved line:
x2 <- rweibull(n = 100, shape = 1, scale = 20000) + 5000

pop_weibull_2 <- plot_pop(
  p_obj = NULL,
  x = x2,
  dist_params_tbl = c(log(20000 - 5000), 1, 5000),
  distribution = "weibull"
)

# Example 3 - Multiple lines:
pop_weibull_3 <- plot_pop(
  p_obj = NULL,
  x = x,
  dist_params_tbl = data.frame(
    p_1 = c(log(20000), log(20000), log(20000)),
    p_2 = c(1, 1.5, 2)
    ),
  distribution = "weibull",
  plot_method = "ggplot2"
)

# Example 4 - Compare two- and three-parametric distributions:
pop_weibull_4 <- plot_pop(
  p_obj = NULL,
  x = x,
  dist_params_tbl = data.frame(
    param_1 = c(log(20000), log(20000)),
    param_2 = c(1, 1),
    param_3 = c(NA, 2)
  ),
  distribution = "weibull"
)
```

---

plot_prob *Probability Plotting Method for Univariate Lifetime Distributions*

Description

This function is used to apply the graphical technique of probability plotting.

Usage

```
plot_prob(x, ...)

## S3 method for class 'wt_model'
plot_prob(
  x,
 distribution = c("weibull", "lognormal", "loglogistic", "normal", "logistic", "sev"),
  title_main = "Probability Plot",
  title_x = "Characteristic",
  title_y = "Unreliability",
  title_trace = "Sample",
  plot_method = c("plotly", "ggplot2"),
  ...
)

## S3 method for class 'wt_cdf_estimation'
plot_prob(
  x,
 distribution = c("weibull", "lognormal", "loglogistic", "normal", "logistic", "sev"),
  title_main = "Probability Plot",
  title_x = "Characteristic",
  title_y = "Unreliability",
  title_trace = "Sample",
  plot_method = c("plotly", "ggplot2"),
  ...
)
```

Arguments

x	An object of class `wt_cdf_estimation` or `wt_model`.
...	Further arguments passed to or from other methods. Currently not used.
distribution	Supposed distribution of the random variable.
title_main	A character string which is assigned to the main title of the plot.
title_x	A character string which is assigned to the title of the x axis.
title_y	A character string which is assigned to the title of the y axis.
title_trace	A character string which is assigned to the trace shown in the legend.
plot_method	Package, which is used for generating the plot output.

Details

If x was split by `mixmod_em`, `estimate_cdf` with method `"johnson"` is applied to subgroup-specific data. The calculated plotting positions are shaped according to the determined split in `mixmod_em`.

In mixmod_regression a maximum of three subgroups can be determined and thus being plotted. The intention of this function is to give the user a hint for the existence of a mixture model. An in-depth analysis should be done afterwards.

The marker label for x and y are determined by the first word provided in the argument title_x and title_y respectively, i.e. if title_x = "Mileage in km" the x label of the marker is "Mileage".

The name of the legend entry is a combination of the title_trace and the number of determined subgroups (if any). If title_trace = "Group" and the data has been split in two groups, the legend entries are "Group: 1" and "Group: 2".

Value

Returns a plot object containing the probability plot.

References

Meeker, William Q; Escobar, Luis A., Statistical methods for reliability data, New York: Wiley series in probability and statistics, 1998

Examples

```
# Reliability data:
data <- reliability_data(
  alloy,
  x = cycles,
  status = status
)

# Probability estimation:
prob_tbl <- estimate_cdf(
  data,
  methods = c("johnson", "kaplan")
)

# Example 1 - Probability Plot Weibull:
plot_weibull <- plot_prob(prob_tbl)

# Example 2 - Probability Plot Lognormal:
plot_lognormal <- plot_prob(
  x = prob_tbl,
  distribution = "lognormal"
)

## Mixture identification
# Reliability data:
data_mix <- reliability_data(
  voltage,
  x = hours,
  status = status
)

prob_mix <- estimate_cdf(
```

```
  data_mix,
  methods = c("johnson", "kaplan")
)

# Example 3 - Mixture identification using mixmod_regression:
mix_mod_rr <- mixmod_regression(prob_mix)

plot_mix_mod_rr <- plot_prob(x = mix_mod_rr)

# Example 4 - Mixture identification using mixmod_em:
mix_mod_em <- mixmod_em(data_mix)

plot_mix_mod_em <- plot_prob(x = mix_mod_em)
```

`plot_prob.default` *Probability Plotting Method for Univariate Lifetime Distributions*

Description

This function is used to apply the graphical technique of probability plotting.

Usage

```
## Default S3 method:
plot_prob(
  x,
  y,
  status,
  id = rep("XXXXXX", length(x)),
 distribution = c("weibull", "lognormal", "loglogistic", "normal", "logistic", "sev"),
  title_main = "Probability Plot",
  title_x = "Characteristic",
  title_y = "Unreliability",
  title_trace = "Sample",
  plot_method = c("plotly", "ggplot2"),
  ...
)
```

Arguments

`x`	A numeric vector which consists of lifetime data. Lifetime data could be every characteristic influencing the reliability of a product, e.g. operating time (days/months in service), mileage (km, miles), load cycles.
`y`	A numeric vector which consists of estimated failure probabilities regarding the lifetime data in `x`.
`status`	A vector of binary data (0 or 1) indicating whether unit i is a right censored observation (= 0) or a failure (= 1).

id	A character vector for the identification of every unit.
distribution	Supposed distribution of the random variable.
title_main	A character string which is assigned to the main title of the plot.
title_x	A character string which is assigned to the title of the x axis.
title_y	A character string which is assigned to the title of the y axis.
title_trace	A character string which is assigned to the trace shown in the legend.
plot_method	Package, which is used for generating the plot output.
...	Further arguments passed to or from other methods. Currently not used.

Details

If x was split by mixmod_em, estimate_cdf with method "johnson" is applied to subgroup-specific data. The calculated plotting positions are shaped according to the determined split in mixmod_em.

In mixmod_regression a maximum of three subgroups can be determined and thus being plotted. The intention of this function is to give the user a hint for the existence of a mixture model. An in-depth analysis should be done afterwards.

The marker label for x and y are determined by the first word provided in the argument title_x and title_y respectively, i.e. if title_x = "Mileage in km" the x label of the marker is "Mileage".

The name of the legend entry is a combination of the title_trace and the number of determined subgroups (if any). If title_trace = "Group" and the data has been split in two groups, the legend entries are "Group: 1" and "Group: 2".

Value

Returns a plot object containing the probability plot.

References

Meeker, William Q; Escobar, Luis A., Statistical methods for reliability data, New York: Wiley series in probability and statistics, 1998

See Also

plot_prob

Examples

```
# Vectors:
cycles   <- alloy$cycles
status <- alloy$status

# Probability estimation:
prob_tbl <- estimate_cdf(
  x = cycles,
  status = status,
  method = "johnson"
)
```

```
# Example 1: Probability Plot Weibull:
plot_weibull <- plot_prob(
  x = prob_tbl$x,
  y = prob_tbl$prob,
  status = prob_tbl$status,
  id = prob_tbl$id
)

# Example 2: Probability Plot Lognormal:
plot_lognormal <- plot_prob(
  x = prob_tbl$x,
  y = prob_tbl$prob,
  status = prob_tbl$status,
  id = prob_tbl$id,
  distribution = "lognormal"
)
```

`plot_prob_mix` *Probability Plot for Separated Mixture Models*

Description

Soft-deprecated

`plot_prob_mix()` is no longer under active development, switching to `plot_prob` is recommended.

Usage

```
plot_prob_mix(
  x,
  status,
  id = rep("XXXXXX", length(x)),
  distribution = c("weibull", "lognormal", "loglogistic"),
  mix_output,
  title_main = "Probability Plot",
  title_x = "Characteristic",
  title_y = "Unreliability",
  title_trace = "Sample",
  plot_method = c("plotly", "ggplot2"),
  ...
)
```

Arguments

x	A numeric vector which consists of lifetime data. Lifetime data could be every characteristic influencing the reliability of a product, e.g. operating time (days/months in service), mileage (km, miles), load cycles.

status	A vector of binary data (0 or 1) indicating whether unit *i* is a right censored observation (= 0) or a failure (= 1).
id	A character vector for the identification of every unit.
distribution	Supposed distribution of the random variable.
mix_output	A list provided by mixmod_regression or mixmod_em, which consists of values necessary to visualize the subgroups.The default value of mix_output is NULL.
title_main	A character string which is assigned to the main title of the plot.
title_x	A character string which is assigned to the title of the x axis.
title_y	A character string which is assigned to the title of the y axis.
title_trace	A character string which is assigned to the trace shown in the legend.
plot_method	Package, which is used for generating the plot output.
...	Further arguments passed to or from other methods. Currently not used.

Details

This function is used to apply the graphical technique of probability plotting to univariate mixture models that have been separated with functions mixmod_regression or mixmod_em.

If data has been split by mixmod_em the function johnson_method is applied to subgroup-specific data. The calculated plotting positions are shaped regarding the obtained split of the used splitting function.

In mixmod_regression a maximum of three subgroups can be determined and thus being plotted. The intention of this function is to give the user a hint for the existence of a mixture model. An in-depth analysis should be done afterwards.

The marker label for x and y are determined by the first word provided in the argument title_x respective title_y, i.e. if title_x = "Mileage in km" the x label of the marker is "Mileage".

The name of the legend entry is a combination of the title_trace and the number of determined subgroups. If title_trace = "Group" and the data could be split in two groups, the legend entries would be "Group 1" and "Group 2".

References

Doganaksoy, N.; Hahn, G.; Meeker, W. Q., Reliability Analysis by Failure Mode, Quality Progress, 35(6), 47-52, 2002

See Also

plot_prob

Examples

```
# Vectors
hours <- voltage$hours
status <- voltage$status

# Example 1 - Using result of mixmod_em:
mix_mod_em <- mixmod_em(
```

```
  x = hours,
  status = status
)

plot_weibull_em <- plot_prob_mix(
  x = hours,
  status = status,
  distribution = "weibull",
  mix_output = mix_mod_em
)

# Example 2 - Using result of mixmod_regression:
john <- estimate_cdf(
  x = hours,
  status = status,
  method = "johnson"
)

mix_mod_reg <- mixmod_regression(
  x = john$x,
  y = john$prob,
  status = john$status,
  distribution = "weibull"
)

plot_weibull_reg <- plot_prob_mix(
  x = hours,
  status = status,
  distribution = "weibull",
  mix_output = mix_mod_reg
)
```

`predict_prob` *Prediction of Failure Probabilities for Parametric Lifetime Distributions*

Description

This function predicts the (failure) probabilities of two- or three-parametric lifetime distributions that belong to the (log-)location-scale family.

Usage

```
predict_prob(
  q,
  dist_params,
 distribution = c("weibull", "lognormal", "loglogistic", "normal", "logistic", "sev",
    "weibull3", "lognormal3", "loglogistic3")
)
```

Arguments

q	A numeric vector of quantiles.
dist_params	A (named) numeric vector of (log-)location-scale parameters in the order of location (μ) and scale (σ). If a three-parametric model is selected, the threshold parameter (γ) has to be the third element.
distribution	Supposed distribution of the random variable.

Details

For a given set of parameters and specified quantiles the (failure) probabilities of the chosen model are determined.

Value

A vector with predicted (failure) probabilities.

Examples

```
# Example 1 - Predicted probabilities for a two-parameter weibull distribution:
probs_weib2 <- predict_prob(
  q = c(15, 48, 124),
  dist_params = c(5, 0.5),
  distribution = "weibull"
)

# Example 2 - Predicted quantiles for a three-parameter weibull distribution:
probs_weib3 <- predict_prob(
  q = c(25, 58, 134),
  dist_params = c(5, 0.5, 10),
  distribution = "weibull3"
)
```

---

predict_quantile *Prediction of Quantiles for Parametric Lifetime Distributions*

Description

This function predicts the quantiles of two- or three-parametric lifetime distributions that belong to the (log-)location-scale family.

Usage

```
predict_quantile(
  p,
  dist_params,
 distribution = c("weibull", "lognormal", "loglogistic", "normal", "logistic", "sev",
    "weibull3", "lognormal3", "loglogistic3")
)
```

Arguments

p	A numeric vector of probabilities.
dist_params	A (named) numeric vector of (log-)location-scale parameters in the order of location (μ) and scale (σ). If a three-parametric model is selected, the threshold parameter (γ) has to be the third element.
distribution	Supposed distribution of the random variable.

Details

For a given set of parameters and specified probabilities the quantiles of the chosen model are determined.

Value

A vector with predicted quantiles.

Examples

```
# Example 1 - Predicted quantiles for a two-parameter weibull distribution:
quants_weib2 <- predict_quantile(
  p = c(0.01, 0.1, 0.5),
  dist_params = c(5, 0.5),
  distribution = "weibull"
)

# Example 2 - Predicted quantiles for a three-parameter weibull distribution:
quants_weib3 <- predict_quantile(
  p = c(0.01, 0.1, 0.5),
  dist_params = c(5, 0.5, 10),
  distribution = "weibull3"
)
```

---

rank_regression *Rank Regression for Parametric Lifetime Distributions*

Description

This function fits an **x on y** regression to a linearized two- or three-parameter lifetime distribution for complete and (multiple) right censored data. The parameters are determined in the frequently used (log-)location-scale parameterization.

For the Weibull, estimates are additionally transformed such that they are in line with the parameterization provided by the *stats* package (see Weibull).

Usage

```
rank_regression(x, ...)

## S3 method for class 'wt_cdf_estimation'
rank_regression(
  x,
 distribution = c("weibull", "lognormal", "loglogistic", "normal", "logistic", "sev",
    "weibull3", "lognormal3", "loglogistic3"),
  conf_level = 0.95,
  ...
)
```

Arguments

x	Object of class wt_cdf_estimation returned from estimate_cdf.
...	Further arguments passed to or from other methods. Currently not used.
distribution	Supposed distribution of the random variable.
conf_level	Confidence level of the interval. If distribution is "weibull" this must be one of 0.9, 0.95 or 0.99.

Details

If distribution is "weibull" or "weibull3", the approximated confidence intervals for the parameters can only be estimated on the following confidence levels (see 'References' *(Mock, 1995)*):

- conf_level = 0.90,
- conf_level = 0.95,
- conf_level = 0.99.

If the distribution is not the Weibull, the confidence intervals of the parameters are computed on the basis of a heteroscedasticity-consistent covariance matrix. Here it should be said that there is no statistical foundation to determine the standard errors of the parameters using *Least Squares* in context of *Rank Regression*. For an accepted statistical method use maximum likelihood.

Value

Returns a list with classes wt_model, wt_rank_regression and wt_model_estimation containing the following elements:

- coefficients : A named vector of estimated coefficients (parameters of the assumed distribution). **Note**: The parameters are given in location-scale-parameterization.
- confint : Confidence intervals for parameters. If distribution is "lognormal3" or "loglogistic3" no confidence interval for the threshold parameter is computed.
- varcov : Provided, if distribution is not "weibull" or "weibull3". Estimated heteroscedasticity-consistent variance-covariance matrix for the (log-)location-scale parameters.
- shape_scale_coefficients : Only included if distribution is "weibull" or "weibull3" (parameterization used in stats::Weibull).

- shape_scale_confint : Only included if distribution is "weibull" or "weibull3". Approximated confidence intervals for scale η and shape β (and threshold γ) if distribution is "weibull3".
- r_squared : Coefficient of determination.
- data : A tibble with class wt_cdf_estimation returned from estimate_cdf.
- distribution : Specified distribution.

If more than one method was specified in estimate_cdf, the resulting output is a list with class wt_model_estimation_list. In this case each list element has classes wt_rank_regression and wt_model_estimation and the items listed above, are included.

References

- Mock, R., Methoden zur Datenhandhabung in Zuverlässigkeitsanalysen, vdf Hochschulverlag AG an der ETH Zürich, 1995
- Meeker, William Q; Escobar, Luis A., Statistical methods for reliability data, New York: Wiley series in probability and statistics, 1998

Examples

```
# Reliability data preparation:
## Data for two-parametric model:
data_2p <- reliability_data(
  shock,
  x = distance,
  status = status
)

## Data for three-parametric model:
data_3p <- reliability_data(
  alloy,
  x = cycles,
  status = status
)

# Probability estimation:
prob_tbl_2p <- estimate_cdf(
  data_2p,
  methods = "johnson"
)

prob_tbl_3p <- estimate_cdf(
  data_3p,
  methods = "johnson"
)

prob_tbl_mult <- estimate_cdf(
  data_3p,
  methods = c("johnson", "kaplan")
)
```

```
# Example 1 - Fitting a two-parametric weibull distribution:
rr_2p <- rank_regression(
  x = prob_tbl_2p,
  distribution = "weibull"
)

# Example 2 - Fitting a three-parametric lognormal distribution:
rr_3p <- rank_regression(
  x = prob_tbl_3p,
  distribution = "lognormal3",
  conf_level = 0.99
)

# Example 3 - Fitting a three-parametric loglogistic distribution if multiple
# methods in estimate_cdf were specified:
rr_lists <- rank_regression(
  x = prob_tbl_mult,
  distribution = "loglogistic3",
  conf_level = 0.90
)
```

`rank_regression.default` *Rank Regression for Parametric Lifetime Distributions*

Description

This function fits an **x on y** regression to a linearized two- or three-parameter lifetime distribution for complete and (multiple) right censored data. The parameters are determined in the frequently used (log-)location-scale parameterization.

For the Weibull, estimates are additionally transformed such that they are in line with the parameterization provided by the *stats* package (see Weibull).

Usage

```
## Default S3 method:
rank_regression(
  x,
  y,
  status,
 distribution = c("weibull", "lognormal", "loglogistic", "normal", "logistic", "sev",
    "weibull3", "lognormal3", "loglogistic3"),
  conf_level = 0.95,
  ...
)
```

Arguments

x	A numeric vector which consists of lifetime data. Lifetime data could be every characteristic influencing the reliability of a product, e.g. operating time (days/months in service), mileage (km, miles), load cycles.
y	A numeric vector which consists of estimated failure probabilities regarding the lifetime data in x.
status	A vector of binary data (0 or 1) indicating whether a unit is a right censored observation (= 0) or a failure (= 1).
distribution	Supposed distribution of the random variable.
conf_level	Confidence level of the interval. If distribution is "weibull" this must be one of 0.9, 0.95 or 0.99.
...	Further arguments passed to or from other methods. Currently not used.

Details

If distribution is "weibull" or "weibull3", the approximated confidence intervals for the parameters can only be estimated on the following confidence levels (see 'References' *(Mock, 1995)*):

- conf_level = 0.90,
- conf_level = 0.95,
- conf_level = 0.99.

If the distribution is not the Weibull, the confidence intervals of the parameters are computed on the basis of a heteroscedasticity-consistent covariance matrix. Here it should be said that there is no statistical foundation to determine the standard errors of the parameters using *Least Squares* in context of *Rank Regression*. For an accepted statistical method use maximum likelihood.

Value

Returns a list with classes wt_model, wt_rank_regression and wt_model_estimation containing the following elements:

- coefficients : A named vector of estimated coefficients (parameters of the assumed distribution). **Note**: The parameters are given in location-scale-parameterization.
- confint : Confidence intervals for parameters. If distribution is "lognormal3" or "loglogistic3" no confidence interval for the threshold parameter is computed.
- varcov : Provided, if distribution is not "weibull" or "weibull3". Estimated heteroscedasticity-consistent variance-covariance matrix for the (log-)location-scale parameters.
- shape_scale_coefficients : Only included if distribution is "weibull" or "weibull3" (parameterization used in stats::Weibull).
- shape_scale_confint : Only included if distribution is "weibull" or "weibull3". Approximated confidence intervals for scale η and shape β (and threshold γ) if distribution is "weibull3".
- r_squared : Coefficient of determination.
- data : A tibble with columns x, status and prob.
- distribution : Specified distribution.

References

- Mock, R., Methoden zur Datenhandhabung in Zuverlässigkeitsanalysen, vdf Hochschulverlag AG an der ETH Zürich, 1995
- Meeker, William Q; Escobar, Luis A., Statistical methods for reliability data, New York: Wiley series in probability and statistics, 1998

See Also

rank_regression

Examples

```
# Vectors:
obs <- seq(10000, 100000, 10000)
status_1 <- c(0, 1, 1, 0, 0, 0, 1, 0, 1, 0)

cycles <- alloy$cycles
status_2 <- alloy$status

# Example 1 - Fitting a two-parametric weibull distribution:
tbl_john <- estimate_cdf(
  x = obs,
  status = status_1,
  method = "johnson"
)

rr <- rank_regression(
  x = tbl_john$x,
  y = tbl_john$prob,
  status = tbl_john$status,
  distribution = "weibull",
  conf_level = 0.90
)

# Example 2 - Fitting a three-parametric lognormal distribution:
tbl_kaplan <- estimate_cdf(
  x = cycles,
  status = status_2,
  method = "kaplan"
)

rr_2 <- rank_regression(
  x = tbl_kaplan$x,
  y = tbl_kaplan$prob,
  status = tbl_kaplan$status,
  distribution = "lognormal3"
)
```

`reliability_data` *Reliability Data*

Description

Create consistent reliability data based on an existing `data.frame` (preferred) or on multiple equal length vectors.

Usage

```
reliability_data(data = NULL, x, status, id = NULL, .keep_all = FALSE)
```

Arguments

`data`	Either `NULL` or a `data.frame`. If data is `NULL`, `x`, `status` and `id` must be vectors containing the data. Otherwise `x`, `status` and `id` can be either column names or column positions.
`x`	Lifetime data, that means any characteristic influencing the reliability of a product, e.g. operating time (days/months in service), mileage (km, miles), load cycles.
`status`	Binary data (0 or 1) indicating whether a unit is a right censored observation (= 0) or a failure (= 1).
`id`	Identification of every unit.
`.keep_all`	If `TRUE` keep remaining variables in `data`.

Value

A tibble with class `wt_reliability_data` containing the following columns (if `.keep_all = FALSE`):

- `x` : Lifetime characteristic.
- `status` : Binary data (0 or 1) indicating whether a unit is a right censored observation (= 0) or a failure (= 1).
- `id` : Identification for every unit.

If `.keep_all = TRUE`, the remaining columns of `data` are also preserved.

Examples

```
# Example 1 -  Based on an existing data.frame/tibble and column names:
data <- reliability_data(
  data = shock,
  x = distance,
  status = status
)

# Example 2 - Based on an existing data.frame/tibble and column positions:
data_2 <- reliability_data(
```

```
  data = shock,
  x = 1,
  status = 3
)

# Example 3 - Keep all variables of the tibble/data.frame entered to argument data:
data_3 <- reliability_data(
  data = shock,
  x = distance,
  status = status,
  .keep_all = TRUE
)

# Example 4 - Based on vectors:
cycles   <- c(300, 300, 300, 300, 300, 291, 274, 271, 269, 257, 256, 227, 226,
              224, 213, 211, 205, 203, 197, 196, 190, 189, 188, 187, 184, 180,
              180, 177, 176, 173, 172, 171, 170, 170, 169, 168, 168, 162, 159,
              159, 159, 159, 152, 152, 149, 149, 144, 143, 141, 141, 140, 139,
              139, 136, 135, 133, 131, 129, 123, 121, 121, 118, 117, 117, 114,
              112, 108, 104, 99, 99, 96, 94)
state <- c(rep(0, 5), rep(1, 67))
id <- "XXXXXX"

data_4 <- reliability_data(
  x = cycles,
  status = state,
  id = id
)
```

`r_squared_profiling` *R-Squared-Profile Function for Parametric Lifetime Distributions with Threshold*

Description

This function evaluates the coefficient of determination with respect to a given threshold parameter of a three-parametric lifetime distribution. In terms of *Rank Regression* this function can be optimized (`optim`) to estimate the threshold parameter.

Usage

```
r_squared_profiling(x, ...)

## S3 method for class 'wt_cdf_estimation'
r_squared_profiling(
  x,
  thres,
  distribution = c("weibull3", "lognormal3", "loglogistic3"),
  ...
)
```

Arguments

x	Object of class wt_cdf_estimation returned from estimate_cdf.
...	Further arguments passed to or from other methods. Currently not used.
thres	A numeric value for the threshold parameter.
distribution	Supposed three-parametric distribution of the random variable.

Value

Returns the coefficient of determination with respect to the threshold parameter thres.

References

Meeker, William Q; Escobar, Luis A., Statistical methods for reliability data, New York: Wiley series in probability and statistics, 1998

Examples

```
# Data:
data <- reliability_data(
  alloy,
  x = cycles,
  status = status
)

# Probability estimation:
prob_tbl <- estimate_cdf(
  data,
  methods = "johnson"
)

# Determining the optimal coefficient of determination:
## Range of threshold parameter must be smaller than the first failure:
threshold <- seq(
  0,
  min(
    dplyr::pull(
      dplyr::filter(
        prob_tbl,
        status == 1,
        x == min(x)
      ),
      x
    ) - 0.1
  ),
  length.out = 100
)

## Coefficient of determination with respect to threshold values:
profile_r2 <- r_squared_profiling(
  x = dplyr::filter(
```

```
    prob_tbl,
    status == 1
  ),
  thres = threshold,
  distribution = "weibull3"
)

## Threshold value (among the candidates) that maximizes the coefficient of determination:
threshold[which.max(profile_r2)]

## plot:
plot(
  threshold,
  profile_r2,
  type = "l"
)
abline(
  v = threshold[which.max(profile_r2)],
  h = max(profile_r2),
  col = "red"
)
```

`r_squared_profiling.default` *R-Squared-Profile Function for Parametric Lifetime Distributions with Threshold*

Description

This function evaluates the coefficient of determination with respect to a given threshold parameter of a three-parametric lifetime distribution. In terms of *Rank Regression* this function can be optimized (`optim`) to estimate the threshold parameter.

Usage

```
## Default S3 method:
r_squared_profiling(
  x,
  y,
  thres,
  distribution = c("weibull3", "lognormal3", "loglogistic3"),
  ...
)
```

Arguments

`x`	A numeric vector which consists of lifetime data. Lifetime data could be every characteristic influencing the reliability of a product, e.g. operating time (days/months in service), mileage (km, miles), load cycles.

y	A numeric vector which consists of estimated failure probabilities regarding the lifetime data in x.
thres	A numeric value for the threshold parameter.
distribution	Supposed three-parametric distribution of the random variable.
...	Further arguments passed to or from other methods. Currently not used.

Value

Returns the coefficient of determination with respect to the threshold parameter `thres`.

References

Meeker, William Q; Escobar, Luis A., Statistical methods for reliability data, New York: Wiley series in probability and statistics, 1998

See Also

r_squared_profiling

Examples

```
# Vectors:
cycles <- alloy$cycles
status <- alloy$status

# Probability estimation:
prob_tbl <- estimate_cdf(
  x = cycles,
  status = status,
  method = "johnson"
)

# Determining the optimal coefficient of determination:
## Range of threshold parameter must be smaller than the first failure:
threshold <- seq(
  0,
  min(cycles[status == 1]) - 0.1,
  length.out = 100
)

## Coefficient of determination with respect to threshold values:
profile_r2 <- r_squared_profiling(
  x = prob_tbl$x[prob_tbl$status == 1],
  y = prob_tbl$prob[prob_tbl$status == 1],
  thres = threshold,
  distribution = "weibull3"
)

## Threshold value (among the candidates) that maximizes the
## coefficient of determination:
threshold[which.max(profile_r2)]
```

```
## plot:
plot(
  threshold,
  profile_r2,
  type = "l"
)
abline(
  v = threshold[which.max(profile_r2)],
  h = max(profile_r2),
  col = "red"
)
```

---

shock *Distance to Failure for Vehicle Shock Absorbers*

Description

Distance to failure for 38 vehicle shock absorbers.

Usage

```
shock
```

Format

A tibble with 38 rows and 3 variables:

distance Observed distance.

failure_mode One of two failure modes (`mode_1` and `mode_2`) or `censored` if no failure occurred.

status If `failure_mode` is either `mode_1` or `mode_2` this is `1` else `0`.

Source

Meeker, William Q; Escobar, Luis A., Statistical Methods for Reliability Data, New York: Wiley series in probability and statistics (1998, p.630)

`voltage` *High Voltage Stress Test for the Dielectric Insulation of Generator armature bars*

Description

A sample of 58 segments of bars were subjected to a high voltage stress test. Two failure modes occurred, Mode D (degradation failure) and Mode E (early failure).

Usage

```
voltage
```

Format

A tibble with 58 rows and 3 variables:

hours Observed hours.

failure_mode One of two failure modes (`D` and `E`) or `censored` if no failure occurred.

status If `failure_mode` is either `D` or `E` this is `1` else `0`.

Source

Doganaksoy, N.; Hahn, G.; Meeker, W. Q., Reliability Analysis by Failure Mode, Quality Progress, 35(6), 47-52, 2002

Index