Package 'weibulltools'

July 25, 2018

Type Package

Title Statistical Methods for Life Data Analysis

Version 0.5.4

Description Contains methods for examining bench test or field data using the well-known Weibull Analysis. It includes Monte Carlo simulation for estimating the life span of products that have not failed, taking account of registering and reporting delays as stated in (Verband der Automobilindustrie e.V. (VDA), 2016, <ISSN:0943-9412>). If the products looked upon are vehicles, the covered mileage can be estimated as well.
It also provides 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.
Methods for estimating the parameters of lifetime distributions, like Maximum Likelihood and Median-Rank Regression, (Genschel and Meeker, 2010, <DOI:10.1080/08982112.2010.503447>) 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 are also included.
If desired, the data can automatically be divided into subgroups using segmented regression.
Besides the calculation, methods for interactive visualization of the edited data using *Plotly* are provided as well. These visualizations include the layout of a probability plot for a specified distribution, the graphical technique of probability plotting and the possibility of adding regression lines and confidence bounds to existing plots.

License GPL-2

Imports dplyr, magrittr, plotly, Rcpp, sandwich, segmented, SPREDA, survival

LinkingTo Rcpp

Encoding UTF-8

LazyData true

RoxygenNote 6.0.1

Suggests knitr, rmarkdown

NeedsCompilation yes

Author Hensel Tim-Gunnar [aut, cre]

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

Repository CRAN

Date/Publication 2018-07-25 21:50:03 UTC

R topics documented:

calculate_ranks . . . 3
confint_betabinom . . . 3
confint_fisher . . . 5
delta_method . . . 6
dist_delay_register . . . 7
dist_delay_report . . . 8
dist_mileage . . . 9
johnson_method . . . 10
kaplan_method . . . 11
mcs_delays . . . 12
mcs_delay_register . . . 14
mcs_delay_report . . . 16
mcs_mileage . . . 18
mixmod_regression . . . 20
ml_estimation . . . 21
mr_method . . . 22
nelson_method . . . 23
plot_conf . . . 24
plot_layout . . . 26
plot_mod . . . 27
plot_mod_mix . . . 28
plot_pop . . . 29
plot_prob . . . 30
plot_prob_mix . . . 32
predict_prob . . . 33
predict_quantile . . . 34
rank_regression . . . 35
weibulltools . . . 36

Index **37**

calculate_ranks *Computation of Johnson Ranks*

Description

This function calculates the Johnson ranks which are used to estimate the failure probabilities in case of (multiple) right censored data.

Usage

```
calculate_ranks(f, n_out, n)
```

Arguments

f	a numeric vector indicating the number of failed units for a specific realization of the lifetime characteristic.
n_out	a numeric vector indicating the number of failed and censored units that have a shorter realization of lifetime characteristic as unit *i*.
n	an integer value indicating the sample size.

Value

A numeric vector containing the computed Johnson ranks.

Examples

```
defectives <- c(0, 1, 2, 0, 0, 0, 3, 0, 2, 0)
n_out <- c(0, 2, 4, 8, 9, 11, 12, 16, 20, 22)
n <- 23
johnson_ranks <- calculate_ranks(f = defectives, n_out = n_out, n = n)
```

confint_betabinom *Beta Binomial Confidence Bounds for Quantiles and/or Probabilities*

Description

This non-parametric approach calculates confidence bounds for quantiles and/or failure probabilities using a procedure that is similar to that used in calculating median ranks. The location-scale parameters estimated by rank regression are needed.

Usage

```
confint_betabinom(x, event, loc_sc_params, distribution = c("weibull",
  "lognormal", "loglogistic"), bounds = c("two_sided", "lower", "upper"),
  conf_level = 0.95, direction = c("y", "x"))
```

Arguments

x	a numeric vector which consists of lifetime data. x is used to specify the range of confidence region(s).
event	a vector of binary data (0 or 1) indicating whether unit i is a right censored observation (= 0) or a failure (= 1).
loc_sc_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.
distribution	supposed distribution of the random variable. The value can be "weibull", "lognormal" or "loglogistic". Other distributions have not been implemented yet.
bounds	a character string specifying the interval(s) which has/have to be computed. Must be one of "two_sided" (default), "lower" or "upper".
conf_level	confidence level of the interval. The default value is conf_level = 0.95.
direction	a character string specifying the direction of the computed interval(s). Must be either "y" (failure probabilities) or "x" (quantiles).

Value

A data frame containing the lifetime characteristic, interpolated ranks as a function of probabilities, the probabilities which are used to compute the ranks and computed values for the specified confidence bound(s).

Examples

```
obs   <- seq(10000, 100000, 10000)
state <- c(0, 1, 1, 0, 0, 0, 1, 0, 1, 0)

df_john <- johnson_method(x = obs, event = state)
mrr <- rank_regression(x = df_john$characteristic,
                       y = df_john$prob,
                       event = df_john$status,
                       distribution = "weibull",
                       conf_level = .95)
conf_betabin <- confint_betabinom(x = df_john$characteristic,
                                  event = df_john$status,
                                  loc_sc_params = mrr$loc_sc_coefficients,
                                  distribution = "weibull",
                                  bounds = "two_sided",
                                  conf_level = 0.95,
                                  direction = "y")
```

confint_fisher *Fisher Confidence Bounds for Quantiles and/or Probabilities*

Description

This method computes normal-approximation confidence intervals for quantiles and/or failure probabilities using the delta_method. The required location-scale parameters and variance-covariance matrix need to be estimated by Maximum Likelihood.

Usage

```
confint_fisher(x, event, loc_sc_params, loc_sc_varcov,
  distribution = c("weibull", "lognormal", "loglogistic"),
  bounds = c("two_sided", "lower", "upper"), conf_level = 0.95,
  direction = c("y", "x"))
```

Arguments

x	a numeric vector which consists of lifetime data. x is used to specify the range of confidence region(s).
event	a vector of binary data (0 or 1) indicating whether unit i is a right censored observation (= 0) or a failure (= 1).
loc_sc_params	a (named) numeric vector of estimated (by Maximum Likelihood) 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.
loc_sc_varcov	a (named) numeric matrix of estimated (by Maximum Likelihood) location and scale variances and covariances for a specified distribution. The order of elements is important. First entry of the diagonal needs to be the variance of the location parameter Var(μ) and the second element of the diagonal needs to be the variance of the scale parameter Var(σ).
distribution	supposed distribution of the random variable. The value can be "weibull", "lognormal", "loglogistic".
bounds	a character string specifying the interval(s) which has/have to be computed. Must be one of "two_sided" (default), "lower" or "upper".
conf_level	confidence level of the interval. The default value is conf_level = 0.95.
direction	a character string specifying the direction of the computed interval(s). Must be either "y" (failure probabilities) or "x" (quantiles).

Value

A data frame containing the lifetime characteristic, the probabilities, estimated standard errors by the delta method and computed values for the specified confidence bound(s).

Examples

```
obs   <- seq(10000, 100000, 10000)
state <- c(0, 1, 1, 0, 0, 0, 1, 0, 1, 0)
df_john <- johnson_method(x = obs, event = state)
mle <- ml_estimation(x = obs, event = state,
                     distribution = "weibull", conf_level = 0.95)
conf_fish <- confint_fisher(x = df_john$characteristic,
                            event = df_john$status,
                            loc_sc_params = mle$loc_sc_coefficients,
                            loc_sc_varcov = mle$loc_sc_vcov,
                            distribution = "weibull",
                            bounds = "two_sided",
                            conf_level = 0.95,
                            direction = "y")
```

`delta_method` *Delta Method for Parametric Lifetime Distributions*

Description

The delta method estimates the standard error for quantities that can be written as non-linear functions of ML estimators like failure probabilities or quantiles. I.e. the location-scale parameters and variance-covariance matrix of these need to be estimated by Maximum Likelihood.

Usage

```
delta_method(p, loc_sc_params, loc_sc_varcov, distribution = c("weibull",
  "lognormal", "loglogistic"), direction = c("y", "x"))
```

Arguments

`p`	a numeric value of a probability or a quantile. If the standard error of probability is of interest a specific quantile needs to be supplied and vice versa.
`loc_sc_params`	a (named) numeric vector of estimated (by Maximum Likelihood) 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.
`loc_sc_varcov`	a (named) numeric matrix of estimated (by Maximum Likelihood) location and scale variances and covariances for a specified distribution. The order of elements is important. First entry of the diagonal needs to be the variance of the location parameter Var(μ) and the second element of the diagonal needs to be the variance of the scale parameter Var(σ).
`distribution`	supposed distribution of the random variable. The value can be `"weibull"`, `"lognormal"` or `"loglogistic"`. Other distributions have not been implemented yet.
`direction`	a character string specifying the direction of the computed standard errors. Must be either "y" (failure probability) or "x" (quantile). If `p` is a quantile then *direction* needs to be "y" and vice versa.

Value

A numeric value with estimated standard error of failure probability or quantiles.

Examples

```
obs   <- seq(10000, 100000, 10000)
state <- c(0, 1, 1, 0, 0, 0, 1, 0, 1, 0)

mle <- ml_estimation(x = obs, event = state,
                     distribution = "weibull", conf_level = 0.95)
delta_prob <- sapply(obs, delta_method,
                           loc_sc_params = mle$loc_sc_coefficients,
                           loc_sc_varcov = mle$loc_sc_vcov,
                           distribution = "weibull",
                           direction = "y")
```

---

dist_delay_register *Parameter Estimation of the Delay in Registration Distribution*

Description

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 MLE.

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. If no date is available use `NA`.
`date_register`	a vector of class `"character"` or `"Date"`, in the format "yyyy-mm-dd", indicating the date of registration of a unit. If no date is available use `NA`.
`distribution`	supposed distribution of the random variable. The default value is `"lognormal"`. So far no other distribution is implemented.

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

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 MLE.

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. If no date is available use `NA`.
`date_report`	a vector of class `"character"` or `"Date"`, in the format "yyyy-mm-dd", indicating the date of report of a failed unit. If no date is available use `NA`.
`distribution`	supposed distribution of the random variable. The default value is `"lognormal"`. So far no other distribution is implemented.

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 the Mileage Distribution*

Description

This function introduces a random variable of annual mileage using the units in the sample that had a failure and afterwards estimates the parameter(s) of a supposed distribution, using MLE.

Usage

```
dist_mileage(x, event, mileage, distribution = "lognormal")
```

Arguments

`x`	a numeric vector of operating times. If not available use `NA`.
`event`	a vector of binary data (0 or 1) indicating whether unit i is a right censored observation (= 0) or a failure (= 1).
`mileage`	a numeric vector of driven distances. If not available use `NA`.
`distribution`	supposed distribution of the random variable. The default value is `"lognormal"`. So far no other distribution is implemented.

Value

A named vector of estimated parameters for the specified mileage distribution.

Examples

```
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)

op_time <- as.numeric(difftime(as.Date(date_of_repair),
                               as.Date(date_of_registration),
                               units = "days"))
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)
state <- c(0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0)

params_mileage_annual <- dist_mileage(x = op_time, event = state,
                                      mileage = mileage,
                                      distribution = "lognormal")
```

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

Description

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

Usage

```
johnson_method(x, event, id = rep("XXXXXX", length(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.
`event`	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.

Value

A data frame containing id, lifetime characteristic, status of the unit, the adjusted rank and the estimated failure probability. For right censored observations the cells of the rank and probabilty columns are filled with NA values.

Examples

```
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")

df_john <- johnson_method(x = obs, event = state, id = uic)
```

---

kaplan_method *Estimation of Failure Probabilities using Kaplan-Meier*

Description

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 orginial Kaplan-Meier estimator (based on *NIST/SEMATECH e-Handbook of Statistical Methods*, 8.2.1.5.): If the last unit (unit with highest observed lifetime) is a defective unit, the estimator is adjusted in such a way that the survival estimate for this unit is not *zero* and therefore the estimate for the failure probability is not equal to *one*. Otherwise the estimate in this context would be too pessimisitc. Since the failure probability estimation in this function is not based on *Median Ranks*, the Betabinomial confidence intervals cannot be calculated on the basis of Kaplan-Meier failure probabilities.

Usage

```
kaplan_method(x, event, id = rep("XXXXXX", length(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.
event	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.

Value

A data frame containing id, lifetime characteristic, status of the unit and the estimated failure probabilty. For right censored observations the cells of probability column are filled with NA.

References

NIST/SEMATECH e-Handbook of Statistical Methods, *8.2.1.5. Empirical model fitting - distribution free (Kaplan-Meier) approach*, https://www.itl.nist.gov/div898/handbook/apr/section2/apr215.htm, 30/04/2018

Examples

```
# Example 1
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")

df_kap <- kaplan_method(x = obs, event = state, id = uic)

# Example 2
 df <- data.frame(obs = c(10000, 10000, 20000, 20000, 30000,
                          30000, 30000, 30000, 40000, 50000,
                          50000, 60000, 70000, 70000, 70000,
                          70000, 80000, 80000, 80000, 80000,
                          90000, 90000, 100000),
                  state = rep(1, 23))

df_kap2 <- kaplan_method(x = df$obs, event = df$state)
```

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

Description

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.

Usage

```
mcs_delays(date_prod, date_register, date_repair, date_report, x, event,
  distribution = "lognormal", details = FALSE, seed = NULL)
```

Arguments

`date_prod`	a vector of class `"character"` or `"Date"`, in the format "yyyy-mm-dd", indicating the date of production of a unit. If no date is available use `NA`.
`date_register`	a vector of class `"character"` or `"Date"`, in the format "yyyy-mm-dd", indicating the date of registration of a unit. If no date is available use `NA`.
`date_repair`	a vector of class `"character"` or `"Date"`, in the format "yyyy-mm-dd", indicating the date of repair of a failed unit. If no date is available use `NA`.

date_report	a vector of class "character" or "Date", in the format "yyyy-mm-dd", indicating the date of report of a failed unit. If no date is available use NA.
x	a numeric vector of operating times.
event	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. The default value is "lognormal". So far no other distribution is implemented.
details	a logical variable, where the default value is FALSE. 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, estimated distribution parameters and a seed for reproducibility.
seed	if seed = NULL a random seed is used. Otherwise the user can specify an integer for the seed.

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.
- int_seed : Integer seed number for reproducibility.

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))
state <- sample(c(0, 1), size = length(date_of_repair), replace = TRUE)

# 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,
                          x = op_time,
                          event = state,
                          distribution = "lognormal",
                          seed = NULL,
                          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,
                                x = op_time,
                                event = state,
                                distribution = "lognormal",
                                seed = NULL,
                                details = TRUE)
```

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

Description

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).

Usage

```
mcs_delay_register(date_prod, date_register, x, event,
  distribution = "lognormal", seed = NULL, 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. If no date is available use NA.
date_register	a vector of class "character" or "Date", in the format "yyyy-mm-dd", indicating the date of registration of a unit. If no date is available use NA.
x	a numeric vector of operating times.
event	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. The default value is "lognormal". So far no other distribution is implemented.
seed	if seed = NULL a random seed is used. Otherwise the user can specify an integer for the seed.
details	a logical variable, where the default value is FALSE. 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, estimated distribution parameters and a seed for reproducibility.

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.
- int_seed : Integer seed number for reproducibility.

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))
state <- 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,
                                  x = op_time,
                                  event = state,
                                  distribution = "lognormal",
                                  seed = NULL,
                                  details = FALSE)

# Example 2 - Detailed list output:
list_detail <- mcs_delay_register(date_prod = date_of_production,
                                  date_register = date_of_registration,
                                  x = op_time,
                                  event = state,
                                  distribution = "lognormal",
                                  seed = NULL,
                                  details = TRUE)
```

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

Description

The delay in report describes the time between the occurence 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`).

Usage

```
mcs_delay_report(date_repair, date_report, x, event,
  distribution = "lognormal", details = FALSE, seed = NULL)
```

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. If no date is available use NA.
date_report	a vector of class "character" or "Date", in the format "yyyy-mm-dd", indicating the date of report of a failed unit. If no date is available use NA.
x	a numeric vector of operating times.
event	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. The default value is "lognormal". So far no other distribution is implemented.
details	a logical variable, where the default value is FALSE. 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, estimated distribution parameters and a seed for reproducibility.
seed	if seed = NULL a random seed is used. Otherwise the user can specify an integer for the seed.

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.
- int_seed : Integer seed number for reproducibility.

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))
state <- 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,
                                x = op_time,
                                event = state,
                                distribution = "lognormal",
                                seed = NULL,
                                details = FALSE)

# Example 2 - Detailed list output:
list_detail <- mcs_delay_report(date_repair = date_of_repair,
                                date_report = date_of_report,
                                x = op_time,
                                event = state,
                                distribution = "lognormal",
                                seed = NULL,
                                details = TRUE)
```

`mcs_mileage` *Estimation of Driving Distances for Censored Observations using a Monte Carlo Approach*

Description

This function simulates driving distances for censored observations under the condition that the operating time of these items is known up to a certain date where analysis is made. Operating times for these units can be estimated with functions like `mcs_delay_register`, `mcs_delay_report` and `mcs_delays`. The failed observations (where the driving distances are known) are used to estimate an annual mileage distribution. If the mileage distribution is fully specified annual random driving distances are drawn from this distribution and afterwards adjusted to the operating times of the censored observations.

Usage

```
mcs_mileage(x, event, mileage, distribution = "lognormal", seed = NULL,
  details = FALSE)
```

Arguments

`x`	a numeric vector of operating times. If not available use `NA`.
`event`	a vector of binary data (0 or 1) indicating whether unit i is a right censored observation (= 0) or a failure (= 1).

mileage	a numeric vector of driven distances. If not available use NA.
distribution	supposed distribution of the random variable. The default value is "lognormal". So far no other distribution is implemented.
seed	if seed = NULL a random seed is used. Otherwise the user can specify an integer for the seed.
details	a logical variable, where the default value is FALSE. If FALSE the output consists of a vector with simulated driving distances for the censored units regarding to their current operating time and the input driving distances for the failed units. If TRUE the output consists of a detailed list, i.e the same vector as described before, simulated annual driving distances, estimated distribution parameters and a seed for reproducibility.

Value

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

- mileage : Simulated driving distances for the censored units and the input driving distances for the failed units.
- mileage_sim_annual : Simulated annual driving distances 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.
- int_seed : Integer seed number for reproducibility.

Examples

```
# Example 1 - Simplified vector output (complete data):
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("2014-10-21", "2014-09-15", "2015-07-04", "2015-04-10",
                    "2015-02-15", "2015-04-14", "2015-04-24", "2015-02-27",
                    "2015-04-25", "2015-04-24", "2015-06-12", "2015-08-26",
                    "2015-05-04", "2015-04-04", "2015-09-06", "2015-05-22",
                    "2015-08-21", "2015-09-17", "2015-09-15", "2015-08-15",
                    "2015-11-26", "2015-08-22", "2015-10-05")

op_time <- as.numeric(difftime(as.Date(date_of_repair),
                               as.Date(date_of_registration),
                               units = "days"))
mileage <- c(5227, 15655, 13629, 18292, 24291, 34455, 33555, 21659, 21737,
             29870, 21068, 22986, 122283, 31592, 49050, 36088, 10918, 11153,
             122437, 122842, 20349, 65656, 40777)
```

```
state <- sample(c(0, 1), size = length(op_time), replace = TRUE)

mileage_corrected <- mcs_mileage(x = op_time, event = state,
                                 mileage = mileage,
                                 distribution = "lognormal", seed = NULL,
                                 details = FALSE)

# Example 2 - Detailed list output (complete data):
list_detail <- mcs_mileage(x = op_time, event = state, mileage = mileage,
                           distribution = "lognormal", seed = NULL,
                           details = TRUE)

# Example 3 - Detailed list output (realistic example):
op_time <- c(65, 170, 210, 213, 277, 287, 312, 330, 337, 350, 377, 379, 386,
             413, 426, 436, 451, 472, 483, 512, 525, 556, 557)
mileage <- c(NA, 15655, 13629, NA, 24291, 34455, NA, 21659, 21737,
             NA, 21068, 22986, NA, 31592, 49050, NA, 10918, 11153,
             NA, 122842, 20349, NA, 40777)
state <- c(0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0,
           1, 1, 0, 1)

list_detail <- mcs_mileage(x = op_time, event = state, mileage = mileage,
                           distribution = "lognormal", seed = NULL,
                           details = TRUE)
```

`mixmod_regression` *Mixture Model Identification using Segmented Regression*

Description

This method uses piecewise linear regression to separate the data in subgroups, if appropriate. Since this happens in an automated fashion the function tends to overestimate the number of breakpoints and therefore returns too many subgroups. This problem is already stated in the documentation of the function segmented.lm, which is part of the *segmented* package. A maximum of three subgroups can be obtained.

Usage

```
mixmod_regression(x, y, event, distribution = c("weibull", "lognormal",
  "loglogistic"), 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`.

event	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. The value can be "weibull", "lognormal" or "loglogistic". Other distributions have not been implemented yet.
conf_level	confidence level of the interval. The default value is conf_level = 0.95.

Value

Returns a list where the length of the list depends on the number of identified subgroups. Each list has the same information as provided by rank_regression. Additionally each list has an element that specifies the range regarding the lifetime data for every subgroup.

Examples

```
hours <- c(2, 28, 67, 119, 179, 236, 282, 317, 348, 387, 3, 31, 69, 135,
           191, 241, 284, 318, 348, 392, 5, 31, 76, 144, 203, 257, 286,
           320, 350, 412, 8, 52, 78, 157, 211, 261, 298, 327, 360, 446,
           13, 53, 104, 160, 221, 264, 303, 328, 369, 21, 64, 113, 168,
           226, 278, 314, 328, 377)
state <- c(1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1,
           1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0,
           1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1,
           0, 1, 1, 1, 1, 1, 1)
john <- johnson_method(x = hours, event = state)

mix_mod <- mixmod_regression(x = john$characteristic,
                             y = john$prob,
                             event = john$status,
                             distribution = "weibull")
```

---

ml_estimation *ML Estimation for Two-Parameter Lifetime Distributions*

Description

This method estimates the parameters and calculates normal approximation confidence intervals for a two-parameter lifetime distribution in the frequently used location-scale parametrization. ml_estimation uses the Lifedata.MLE function that is defined in the *SPREDA* package. For the Weibull the estimates are transformed such that they are in line with the parametrization provided by the *stats* package like pweibull. The method is applicable for complete and (multiple) right censored data.

Usage

```
ml_estimation(x, event, distribution = c("weibull", "lognormal",
  "loglogistic"), conf_level = 0.95, details = TRUE)
```

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.
event	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. The value can be "weibull", "lognormal" or "loglogistic". Other distributions have not been implemented yet.
conf_level	confidence level of the interval. The default value is conf_level = 0.95.
details	a logical variable, where the default value is TRUE. If FALSE the output consists of a list that only contains the estimated parameters. If TRUE the output is a detailed list with many more information. See below (**Value**).

Value

Returns a list with the following components (depending on details argument):

- coefficients : Provided, if distribution is "weibull". η is the estimated scale and β the estimated shape parameter.
- confint : Provided, if distribution is "weibull". Confidence interval for η and β.
- loc_sc_coefficients : Estimated location-scale parameters.
- loc_sc_confint : Confidence interval for location-scale parameters.
- loc_sc_vcov : Estimated Variance-Covariance matrix of the used location-scale distribution.
- logL : The log-likelihood value.

Examples

```
obs   <- seq(10000, 100000, 10000)
state <- c(0, 1, 1, 0, 0, 0, 1, 0, 1, 0)

mle <- ml_estimation(x = obs, event = state,
                     distribution = "weibull", conf_level = 0.90)
```

---

mr_method *Estimation of Failure Probabilities using Median Ranks*

Description

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

Usage

```
mr_method(x, event = rep(1, length(x)), id = rep("XXXXXX", length(x)),
  method = "benard")
```

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.
event	a vector of ones indicating that every unit *i* has failed.
id	a character vector for the identification of every unit.
method	method for the estimation of the cdf. Can be "benard" (default) or "invbeta".

Value

A data frame containing id, lifetime characteristic, status of the unit, the rank and the estimated failure probability.

Examples

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

df_mr <- mr_method(x = obs, event = state, id = uic,
                   method = "benard")

# Example 2
df_mr_invbeta <- mr_method(x = obs, event = state, id = uic,
                           method = "invbeta")
```

---

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

Description

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. Since the failure probability estimation in this function is not based on *Median Ranks*, the Betabinomial confidence intervals cannot be calculated on the basis of Nelson-Aalen failure probabilities.

Usage

```
nelson_method(x, event, id = rep("XXXXXX", length(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.
event	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.

Value

A data frame containing id, lifetime characteristic, status of the unit and the estimated failure probabilty. For right censored observations the cells of probability column are filled with NA.

Examples

```
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")

df_nel <- nelson_method(x = obs, event = state, id = uic)
```

---

plot_conf *Add Confidence Region(s) for Quantiles or 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

```
plot_conf(p_obj, x, y, direction = c("y", "x"), distribution = c("weibull",
  "lognormal", "loglogistic"), title_trace = "Confidence Limit")
```

Arguments

p_obj	a plotly object provided by function 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**.
direction	a character string specifying the direction of the plotted interval(s). Must be either "y" (failure probabilities) or "x" (quantiles).

distribution	supposed distribution of the random variable. The value can be "weibull", "lognormal" or "loglogistic". Other distributions have not been implemented yet.
title_trace	a character string which is assigned to the trace shown in the legend.

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$characteristic for x and df$prob for y of the function(s) named before.
- In **Examples** the described approach is shown with code.

Value

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

Examples

```
obs   <- seq(10000, 100000, 10000)
state <- c(0, 1, 1, 0, 0, 0, 1, 0, 1, 0)
id <- LETTERS[1:length(obs)]

df_john <- johnson_method(x = obs, event = state, id = id)
mrr <- rank_regression(x = df_john$characteristic,
                       y = df_john$prob,
                       event = df_john$status,
                       distribution = "weibull",
                       conf_level = .95)
conf_betabin <- confint_betabinom(x = df_john$characteristic,
                                  event = df_john$status,
                                  loc_sc_params = mrr$loc_sc_coefficients,
                                  distribution = "weibull",
                                  bounds = "two_sided",
                                  conf_level = 0.95,
                                  direction = "y")

plot_weibull <- plot_prob(x = df_john$characteristic,
                          y = df_john$prob,
                          event = df_john$status,
                          id = df_john$id,
                          distribution = "weibull",
                          title_main = "Weibull Analysis",
                          title_x = "Mileage in miles",
                          title_y = "Probability of Failure",
                          title_trace = "Failed Items")
plot_reg_weibull <- plot_mod(p_obj = plot_weibull,
```

```
                              x = conf_betabin$characteristic,
                              y = conf_betabin$prob,
                              loc_sc_params = mrr$loc_sc_coefficients,
                              distribution = "weibull",
                              title_trace = "Estimated Weibull CDF")
plot_conf_beta <- plot_conf(p_obj = plot_reg_weibull,
                            x = list(conf_betabin$characteristic),
                            y = list(conf_betabin$lower_bound,
                                     conf_betabin$upper_bound),
                            direction = "y",
                            distribution = "weibull",
                            title_trace = "Confidence Region")
```

`plot_layout` *Layout of the Probability Plot*

Description

This function is used to create the layout of a probability plot.

Usage

```
plot_layout(x, distribution = c("weibull", "lognormal", "loglogistic"),
  title_main = "Probability Plot", title_x = "Characteristic",
  title_y = "Unreliability")
```

Arguments

`x`	a numeric vector which consists of lifetime data. `x` is used to specify the grid of the plot.
`distribution`	supposed distribution of the random variable. Can be `"weibull"`, `"lognormal"` or `"loglogistic"`. Other distributions have not been implemented yet.
`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.

Value

Returns a plotly object which contains the layout that is used for probability plotting.

Examples

```
x_layout <- seq(1e-5, 1e+07, length.out = 10)
grid_weibull <- plot_layout(x = x_layout,
                            distribution = "weibull",
                            title_main = "Weibull Analysis",
                            title_x = "Time to Failure",
                            title_y = "Failure Probability in %")
```

plot_mod *Adding an Estimated Population Line to a Probability Plot*

Description

This function adds a regression line to an existing probability plot using a model estimated by rank_regression or ml_estimation.

Usage

```
plot_mod(p_obj, x, y = NULL, loc_sc_params, distribution = c("weibull",
  "lognormal", "loglogistic"), title_trace = "Fit")
```

Arguments

p_obj	a plotly object provided by function plot_prob.
x	a numeric vector containing the x-coordinates of the regression line.
y	a numeric vector containing the y-coordinates of the regression line. The default value of y is NULL. If y is set NULL the y-coordinates with respect to x are calculated by function predict_prob using estimated coefficients in loc_sc_params. If confidence interval(s) should be added to the plot y should not be set to NULL. For more information see **Details** in plot_conf.
loc_sc_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.
distribution	supposed distribution of the random variable. The value can be "weibull", "lognormal" or "loglogistic". Other distributions have not been implemented yet.
title_trace	a character string whis is assigned to the trace shown in the legend.

Value

Returns a plotly object containing the probability plot with plotting positions and the estimated regression line.

Examples

```
obs   <- seq(10000, 100000, 10000)
state <- c(0, 1, 1, 0, 0, 0, 1, 0, 1, 0)
id <- LETTERS[1:length(obs)]

df_john <- johnson_method(x = obs, event = state, id = id)
mrr <- rank_regression(x = df_john$characteristic,
                       y = df_john$prob,
                       event = df_john$status,
                       distribution = "weibull",
                       conf_level = .90)
```

```
plot_weibull <- plot_prob(x = df_john$characteristic,
                          y = df_john$prob,
                          event = df_john$status,
                          id = df_john$id,
                          distribution = "weibull",
                          title_main = "Weibull Analysis",
                          title_x = "Mileage in miles",
                          title_y = "Probability of Failure in %",
                          title_trace = "Failed Items")

plot_reg_weibull <- plot_mod(p_obj = plot_weibull, x = obs,
                             loc_sc_params = mrr$loc_sc_coefficients,
                             distribution = "weibull",
                             title_trace = "Estimated Weibull CDF")
```

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

Description

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

Usage

```
plot_mod_mix(p_obj, x, y = NULL, reg_output, distribution = c("weibull",
  "lognormal", "loglogistic"), title_trace = "Fit")
```

Arguments

`p_obj`	a plotly object provided by function `plot_prob_mix`.
`x`	a numeric vector containing the x-coordinates of the regression line.
`y`	a numeric vector containing the y-coordinates of the regression line. The default value of y is `NULL`. If y is set `NULL` the y-coordinates with respect to `x` are calculated by function `predict_prob` using estimated coefficients in `reg_output`.
`reg_output`	a list provided by `mixmod_regression` which consists of elements necessary to visualize the regression lines.
`distribution`	supposed distribution of the random variable. The value can be `"weibull"`, `"lognormal"` or `"loglogistic"`. Other distributions have not been implemented yet.
`title_trace`	a character string whis is assigned to the trace shown in the legend.

Value

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

Examples

```
hours <- c(2, 28, 67, 119, 179, 236, 282, 317, 348, 387, 3, 31, 69, 135,
          191, 241, 284, 318, 348, 392, 5, 31, 76, 144, 203, 257, 286,
          320, 350, 412, 8, 52, 78, 157, 211, 261, 298, 327, 360, 446,
          13, 53, 104, 160, 221, 264, 303, 328, 369, 21, 64, 113, 168,
          226, 278, 314, 328, 377)
state <- c(1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1,
          1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0,
          1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1,
          0, 1, 1, 1, 1, 1, 1)
john <- johnson_method(x = hours, event = state)

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

plot_weibull_mix <- plot_prob_mix(x = john$characteristic,
                                  y = john$prob,
                                  event = john$status,
                                  id = john$id,
                                  distribution = "weibull",
                                  reg_output = mix_mod,
                                  title_main = "Mixture Weibull Analysis",
                                  title_x = "Time in Hours",
                                  title_y = "Probability of Failure",
                                  title_trace = "classification")
plot_weibull_reg_mix <- plot_mod_mix(p_obj = plot_weibull_mix, x = hours,
                             reg_output = mix_mod,
                             distribution = "weibull",
                             title_trace = "model")
```

`plot_pop` *Add Population Line to an Existing Grid*

Description

This function adds a linearized CDF to an existing plotly grid.

Usage

```
plot_pop(p_obj, x, params, distribution = c("weibull", "lognormal",
  "loglogistic"), color = I("#FF0000"), title_trace = "Population")
```

Arguments

`p_obj`	a plotly object, which at least includes the layout provided by `plot_layout`.
`x`	a numeric vector containing the x-coordinates of the population line.

params	a (named) numeric vector, where the first entry is the location parameter μ and the second entry is the scale parameter σ of a lognormal or loglogistic distribution. If the distribution value is "weibull" the first entry must be the scale parameter η and the second entry must be the shape parameter β. Parametrization is the same used in rweibull.
distribution	supposed distribution of the random variable. The value can be "weibull", "lognormal" or "loglogistic". Other distributions have not been implemented yet.
color	the color of the population line should be added as follows: For hexadecimal codes: color = I("#3C8DBC") and for a color specified with a string: color = I("blue").
title_trace	a character string whis is assigned to the trace shown in the legend.

Value

A plotly object which contains the supposed linearized population CDF. Failure probabilities must be strictly below 1 and for this very reason

Examples

```
x <- rweibull(n = 100, shape = 1, scale = 20000)
grid_weibull <- plot_layout(x = x,
                            distribution = "weibull",
                            title_main = "Weibull Analysis",
                            title_x = "Time to Failure",
                            title_y = "Failure Probability")
pop_weibull <- plot_pop(p_obj = grid_weibull,
                        x = x, params = c(20000, 1),
                        distribution = "weibull", color = I("green"),
                        title_trace = "Population")
```

---

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, y, event, id = rep("XXXXXX", length(x)),
  distribution = c("weibull", "lognormal", "loglogistic"),
  title_main = "Probability Plot", title_x = "Characteristic",
  title_y = "Unreliability", title_trace = "Sample")
```

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.
event	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. Can be "weibull", "lognormal" or "loglogistic". Other distributions have not been implemented yet.
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 whis is assigned to the trace shown in the legend.

Details

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

The marker label for y is determined by the string provided in the argument title_y, i.e. if title_y = "Probability in percent" the y label of the marker is "Probability".

Value

Returns a plotly object containing the layout of the probability plot provided by plot_layout and the plotting positions.

Examples

```
obs   <- seq(10000, 100000, 10000)
state <- c(0, 1, 1, 0, 0, 0, 1, 0, 1, 0)
id <- LETTERS[1:length(obs)]

df_john <- johnson_method(x = obs, event = state, id = id)

plot_weibull <- plot_prob(x = df_john$characteristic,
                          y = df_john$prob,
                          event = df_john$status,
                          id = df_john$id,
                          distribution = "weibull",
                          title_main = "Weibull Analysis",
                          title_x = "Mileage in miles",
                          title_y = "Probability of Failure in %",
                          title_trace = "Failed Items")
```

plot_prob_mix *Probability Plot for Separated Mixture Models*

Description

This function is used to apply the graphical technique of probability plotting to univariate mixture models that where separated with the function mixmod_regression. A maximum of three subgroups can be 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.

Usage

```
plot_prob_mix(x, y, event, id = rep("XXXXXX", length(x)),
  distribution = c("weibull", "lognormal", "loglogistic"),
  reg_output = NULL, title_main = "Probability Plot",
  title_x = "Characteristic", title_y = "Unreliability",
  title_trace = "Sample")
```

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.
event	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. Can be "weibull", "lognormal" or "loglogistic". Other distributions have not been implemented yet.
reg_output	a list provided by mixmod_regression which consists of values necessary to visualize the segments calculated by mixmod_regression.The default value of reg_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 whis is assigned to the trace shown in the legend.

Details

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

The marker label for y is determined by the string provided in the argument title_y, i.e. if title_y = "Probability in percent" the y label of the marker is "Probability".

Value

Returns a plotly object containing the layout of the probability plot provided by plot_layout and the plotting positions.

Examples

```
hours <- c(2, 28, 67, 119, 179, 236, 282, 317, 348, 387, 3, 31, 69, 135,
          191, 241, 284, 318, 348, 392, 5, 31, 76, 144, 203, 257, 286,
          320, 350, 412, 8, 52, 78, 157, 211, 261, 298, 327, 360, 446,
          13, 53, 104, 160, 221, 264, 303, 328, 369, 21, 64, 113, 168,
          226, 278, 314, 328, 377)
state <- c(1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1,
          1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 0,
          1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1,
          0, 1, 1, 1, 1, 1, 1)
john <- johnson_method(x = hours, event = state)

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

plot_weibull_mix <- plot_prob_mix(x = john$characteristic,
                                  y = john$prob,
                                  event = john$status,
                                  id = john$id,
                                  distribution = "weibull",
                                  reg_output = mix_mod,
                                  title_main = "Mixture Weibull Analysis",
                                  title_x = "Time in Hours",
                                  title_y = "Probability of Failure",
                                  title_trace = "classification")
```

---

predict_prob *Estimation of Failure Probabilities for Parametric Lifetime Distributions*

Description

This function estimates the failure probabilities for a given set of estimated location-scale parameters and specified quantiles.

Usage

```
predict_prob(q, loc_sc_params, distribution = c("weibull", "lognormal",
  "loglogistic"))
```

Arguments

q	a numeric vector which consists of lifetime data.
loc_sc_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.
distribution	supposed distribution of the random variable. The value can be "weibull", "lognormal" or "loglogistic". Other distributions have not been implemented yet.

Value

A vector containing the estimated failure probabilities for a given set of quantiles and estimated parameters.

Examples

```
probs <- predict_prob(q = c(15, 48, 124), loc_sc_params = c(5, 0.5),
                          distribution = "weibull")
```

---

predict_quantile *Estimation of Quantiles for Parametric Lifetime Distributions*

Description

This function estimates the quantiles for a given set of estimated location-scale parameters and specified failure probabilities.

Usage

```
predict_quantile(p, loc_sc_params, distribution = c("weibull", "lognormal",
  "loglogistic"))
```

Arguments

p	a numeric vector which consists of failure probabilities regarding the lifetime data.
loc_sc_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.
distribution	supposed distribution of the random variable. The value can be "weibull", "lognormal" or "loglogistic". Other distributions have not been implemented yet.

Value

A vector containing the estimated quantiles for a given set of failure probabilities and estimated parameters.

Examples

```
quants <- predict_quantile(p = c(0.01, 0.1, 0.5), loc_sc_params = c(5, 0.5),
                           distribution = "weibull")
```

`rank_regression` *Rank Regression for Two-Parameter Lifetime Distributions*

Description

This method fits an **x on y** regression to the linearized two-parameter cdf and is applicable for complete and (multiple) right censored data. The parameters are estimated in the frequently used location-scale parametrization. For the Weibull, estimates are transformed such that they are in line with the parametrization provided by the *stats* package like `pweibull`.

Usage

```
rank_regression(x, y, event, distribution = c("weibull", "lognormal",
  "loglogistic"), conf_level = 0.95, details = TRUE)
```

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`.
`event`	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. The value can be `"weibull"`, `"lognormal"` or `"loglogistic"`. Other distributions have not been implemented yet.
`conf_level`	confidence level of the interval. The default value is `conf_level = 0.95`.
`details`	a logical variable, where the default value is `TRUE`. If `FALSE` the output consists of a list that only contains the estimated parameters. If `TRUE` the output is a detailed list with many more information. See below (**Value**).

Details

When using this method, the approximated confidence intervals for the Weibull parameters (based on p. 51 of Ralf Mock) can only be estimated for the following confidence levels:

- `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 calculated using a heteroscedasticity-consistent covariance matrix. Here it should be said that there is no statistical foundation to calculate the standard errors for the parameters using *Least Squares* in context of *Median Rank Regression*. For an accepted statistical method use MLE (`ml_estimation`).

Value

Returns a list with the following components (depending on `details` argument):

- `coefficients` : Provided, if `distribution` is `"weibull"`. η is the estimated scale and β the estimated shape parameter.
- `confint` : Provided, if `distribution` is `"weibull"`. Confidence interval for η and β.
- `loc_sc_coefficients` : Estimated location-scale parameters.
- `loc_sc_confint` : Confidence interval for location-scale parameters.
- `loc_sc_vcov` : Provided, if `distribution` is not `"weibull"`. Estimated heteroscedasticity-consistent Variance-Covariance matrix of the used location-scale distribution.
- `r_squared` : Coefficient of determination.

References

Mock, R., Methoden zur Datenhandhabung in Zuverlässigkeitsanalysen, vdf Hochschulverlag AG an der ETH Zürich, 1995

Examples

```
obs   <- seq(10000, 100000, 10000)
state <- c(0, 1, 1, 0, 0, 0, 1, 0, 1, 0)

df_john <- johnson_method(x = obs, event = state)
mrr <- rank_regression(x = df_john$characteristic,
                       y = df_john$prob,
                       event = df_john$status,
                       distribution = "weibull",
                       conf_level = .90)
```

`weibulltools` *weibulltools*

Description

The weibulltools package contains methods for examining bench test or field data using the well-known weibull analysis.

Index

calculate_ranks, 3
confint_betabinom, 3, *25*
confint_fisher, 5, *25*

delta_method, *5*, 6
dist_delay_register, 7, *15*
dist_delay_report, 8, *16*
dist_mileage, 9

johnson_method, 10

kaplan_method, 11

Lifedata.MLE, *21*

mcs_delay_register, *12*, 14, *18*
mcs_delay_report, *12*, 16, *18*
mcs_delays, 12, *18*
mcs_mileage, 18
mixmod_regression, 20, *28*, *32*
ml_estimation, 21, *27*, *36*
mr_method, 22

nelson_method, 23

plot_conf, 24, *27*
plot_layout, 26, *29*, *31*, *33*
plot_mod, *24, 25*, 27
plot_mod_mix, 28
plot_pop, 29
plot_prob, *27*, 30
plot_prob_mix, *28*, 32
predict_prob, 33
predict_quantile, 34
pweibull, *21*, *35*

rank_regression, *21*, *27*, 35
rweibull, *30*

segmented.lm, *20*

weibulltools, 36
weibulltools-package *(*weibulltools*)*, 36