Package: incubate (via r-universe)

June 8, 2026

Title Parametric Time-to-Event Analysis with Variable Incubation Phases

Version 1.4.1

Date 2026-06-04

Description Fit parametric models for time-to-event data that show an initial 'incubation period', i.e. a variable delay phase where no events occur. The delayed Weibull distribution serves as the foundational data model. For parameter estimation, different flavours of maximum likelihood estimation ('MLE') and the method of maximum product of spacings estimation ('MPSE') are implemented. Bootstrap confidence intervals for parameters and significance tests in a two group setting are provided.

License LGPL (>= 3)

Encoding UTF-8

LazyData true

Imports future (>= 1.21), future.apply (>= 1.6), glue (>= 1.4), MASS, minqa, purrr (>= 0.3), rlang (>= 0.4), stats, survival, tibble

Suggests boot, dplyr, future.callr, ggplot2 (>= 3.3), knitr, numDeriv, patchwork, rmarkdown, spelling, testthat (>= 3.0.0), tidyr, withr

URL https://gitlab.com/imb-dev/incubate/

BugReports https://gitlab.com/imb-dev/incubate/-/work_items

Language en-GB

Config/roxygen2/version 8.0.0

Config/roxygen2/markdown TRUE

Config/testthat/edition 3

Config/Needs/check spelling

Depends R (>= 3.6.0)

Collate 'S3_integration.R' 'aaa.R' 'cpp11.R' 'data.R' 'utils.R' 'delay_estimation.R' 'delay.R' 'delay_test.R' 'incubate-package.R' 'zzz.R'

VignetteBuilder knitr

LinkingTo cpp11

NeedsCompilation yes

Author Matthias Kuhn [aut, cre, cph] (ORCID:
<https://orcid.org/0000-0003-2868-5155>)

Maintainer Matthias Kuhn <matthias.kuhn@tu-dresden.de>

Repository https://cran.r-universe.dev

Date/Publication 2026-06-08 18:40:02 UTC

RemoteUrl https://github.com/cran/incubate

RemoteRef HEAD

RemoteSha 81c9fd39c4e6c52020af9001e9dad6cd9b09fe24

Contents

`as_percent` *Format a number as percentage.*

Description

Internal helper function that is not exported.

Usage

```
as_percent(x, digits = 1)
```

Arguments

`x`	numeric vector to be formatted as percentage
`digits`	requested number of decimal digits of the percentage

Value

number formatted as percentage character

`bsDataStep` *Generate bootstrap distribution of model parameters to a fitted incubate model*

Description

Given a fitted incubate model, this function generates bootstrap samples to estimate the variability of the model parameters. The bootstrap data is used to make bootstrap inference in the second step. It is an internal function, the main entry point is `confint.incubate_fit()`.

Usage

```
bsDataStep(
  object,
  bs_data = c("parametric", "ordinary"),
  R,
  useBoot = FALSE,
  smd_factor = 1
)
```

Arguments

object	an incubate_fit-object
bs_data	character. Which type of bootstrap method to generate data?
R	integer. Number of bootstrapped model coefficient estimates
useBoot	flag. Do you want to use the boot-package? Default value is FALSE.
smd_factor	numeric. smooth delay factor for initial delay that influences the amount of smoothing. 0 means no smoothing at all. Default is 1. Higher values mean more variability in the delay parameter during bootstrap data generation.

Details

The bootstrap data can be generated by two methods:

1. parametric bootstrap, new data are generated from the fitted model and then the model is refitted to these data.
2. ordinary bootstrap, new data are generated by resampling with replacement from the original data and then the model is refitted to these data.

For the initial delay parameter we allow to smooth the delay estimate in parametric bootstrap. The initial delay parameter is special because this parameter determines when observations can start to occur. Bootstrapping from a delay model will only produce data that starts never before the given delay. Smoothing adds variability here. The default value for smoothing is smd_factor=1 which means to add normal noise of 1xstd.dev of the first observation to the first observation. Alternatively, when using the objective function to find a region for delay1 (this branch is turned off in the code, cf. USE_OBJFUN hard-coded), the default 1 was an optimal value in a simulation for log-quantile together with log delay-shift = 5.

Value

bootstrap data, either as matrix or of class boot (depending on the useBoot-flag)

---

buildControl *Build control list for* delay_model() *and* objFunFactory()

Description

Default values are set as default argument values in this constructor.

Usage

```
buildControl(
  verbose = 0,
  profiled,
  pen_shape = FALSE,
  MLEw_weight = "sdist_median",
  MLEw_optim = "min",
  ties = "density"
)
```

Arguments

verbose	numeric. Degree of verbosity. Default value 0 means no extra output.
profiled	logical. Use objective function based on parameters that are profiled out?
pen_shape	logical. Should we penalize high shape parameters?
MLEw_weight	character. How to derive the weights?
MLEw_optim	character. How to find the optimum? Currently only "min" for minimization is supported.
ties	character. How to handle tied observations?

Details

This is an internal function. The function might change without precautionary measures taken.

Value

list. Control settings for fitting routine `delay_model`

---

buildDist *Builds the distribution object*

Description

This object contains all relevant informations about the chosen distribution.

Usage

```
buildDist(distribution)
```

Arguments

distribution	character(1). Which distribution?

Value

distribution object. currently, it is simply a list.

coef.incubate_fit *Coefficients of a delay-model fit.*

Description

Coefficients of a delay-model fit.

Usage

```
## S3 method for class 'incubate_fit'
coef(object, transformed = FALSE, group = NULL, ...)
```

Arguments

object	object that is a incubate_fit
transformed	flag. Do we request the transformed parameters as used within the optimization?
group	character string to request the canonical parameter for one group
...	further arguments, currently not used.

Value

named coefficient vector

confint.incubate_fit *Confidence intervals for parameters of delay model fit*

Description

Bias-corrected bootstrap confidence limits (either based on quantile or normal approximation) are generated. Optionally, there are also variants that use a log-transformation first. Ordinary or parametric bootstrap are supported. At least R=1000 bootstrap replications are recommended. Default are quantile-based confidence intervals that internally use a log transformation.

Usage

```
## S3 method for class 'incubate_fit'
confint(
  object,
  parm,
  level = 0.95,
  R = 199L,
  bs_data,
 bs_infer = c("logquantile", "lognormal", "quantile", "quantile0", "normal", "normal0"),
  useBoot = FALSE,
  ...
)
```

Arguments

object	object of class incubate_fit
parm	character. Which parameters to get confidence interval for?
level	numeric. Which is the requested confidence level for the interval? Default value is 0.95
R	number of bootstrap replications. Used only if not bs_data-object is provided.
bs_data	character or bootstrap data object. If character, it specifies which type of bootstrap is requested and the bootstrap data will be generated accordingly. Data can also be provided here directly. If missing it uses parametric bootstrap.
bs_infer	character. Which type of bootstrap inference is requested to generate the confidence interval?
useBoot	logical. Delegate bootstrap confint calculation to the boot-package?
...	further arguments, currently not used.

Value

A matrix (or vector) with columns giving lower and upper confidence limits for each parameter.

---

delay_fit *Fit optimal parameters according to the objective function (either MPSE or MLE-based)*

Description

The objective function carries the given data in its environment and it is to be minimized. R's standard routine stats::optim does the numerical optimization, using numerical derivatives.

Usage

```
delay_fit(objFun, optim_args = NULL, verbose = 0)
```

Arguments

objFun	objective function to be minimized
optim_args	list of own arguments for optimization. If NULL it uses the default optim arguments associated to the objective function.
verbose	integer that indicates the level of verboseness. Default 0 is quiet.

Details

When the analytical solution is available (as attribute opt to the objective function), it is returned directly.

Value

optimization object including a named parameter vector or NULL in case of errors during optimization

`delay_model` *Fit a delayed Exponential or Weibull model to one or two given sample(s)*

Description

Maximum product of spacings estimation is used by default to fit the parameters. Estimation via naive maximum likelihood (`method = "MLEn"`) is available, too, but MLEn yields severely biased estimates for small samples. MLEc is a corrected version of MLEn due to Cheng and Iles (1987).

Usage

```
delay_model(
  x = stop("Specify observations for first group x=!", call. = FALSE),
  y = NULL,
  distribution = c("exponential", "weibull", "normal"),
  twoPhase = FALSE,
  bind = NULL,
  method = c("MPSE", "MLEn", "MLEw", "MLEc"),
  control = list()
)
```

Arguments

`x`	numeric. observations of 1st group. Can also be a list of data from two groups.
`y`	numeric. observations from 2nd group
`distribution`	Which delayed distribution is assumed? Exponential or Weibull. Can be given as character or as distribution list-object.
`twoPhase`	logical. Allow for two phases?
`bind`	character. parameter names that are bound together in 2-group situation.
`method`	character. Which method to fit the model? 'MPSE' = maximum product of spacings estimation *or* 'MLEn' = naive maximum likelihood estimation *or* 'MLEw' = weighted MLE' *or* MLEc' = corrected MLE
`control`	list. Details that control the optimization. E.g., profiling, penalization.

Details

The parameter `control=` allows to set specifics of the optimization process of the delay model fit. Possible list entries are

- `verbose` level of verboseness. Default 0 is quiet
- `ties` character. Strategy to handle ties for `method = "MPSE"`. Either 'density' (default), 'equispaced' or 'error'.
- `profiled` aim to profile out a parameter

- pen_shape logical. Should high values of shape (for Weibull distribution) be penalized? Default is FALSE.
- MLEw_weight character. Name of method to build weights for MLEw-method.
- MLEw_optim character. Name for strategy to find extremum: either minimization of L2-norm of 1st partial derivatives (as stated in Cousineau, 2009) or using root-finding

Numerical minimization is normally done by stats::optim. If this minimization attempt fails minqa::bobyqa is used as fall-back. For MLEw, we can also use root finding of gradient function instead of minimization of L2-norm of gradient of MLEw-objective function.

Value

incubate_fit the delay-model fit object with criterion to minimize. Or NULL if optimization failed (e.g. too few observations).

References

R. C. H. Cheng, T. C. Iles, Corrected Maximum Likelihood in Non-Regular Problems, Journal of the Royal Statistical Society: Series B (Methodological), Volume 49, Issue 1, September 1987, pp. 95–101, doi:10.1111/j.25176161.1987.tb01428.x

---

DelayedExponential *Delayed Exponential Distribution*

Description

Density, distribution function, quantile function, random generation and restricted mean survival time function for the delayed exponential distribution. There is an initial delay phase (parameter delay1) where no events occur. After that, rate1 applies. Optionally, a second phase is possible where the hazard rate might change (parameters delay2 and rate2).

Usage

```
dexp_delayed(
  x,
  delay1 = 0,
  rate1 = 1,
  delay2 = NULL,
  rate2 = NULL,
  delay = delay1,
  rate = rate1,
  log = FALSE
)

pexp_delayed(
  q,
  delay1 = 0,
```

```
  rate1 = 1,
  delay2 = NULL,
  rate2 = NULL,
  delay = delay1,
  rate = rate1,
  lower.tail = TRUE,
  log.p = FALSE,
  grad = FALSE
)

qexp_delayed(
  p,
  delay1 = 0,
  rate1 = 1,
  delay2 = NULL,
  rate2 = NULL,
  delay = delay1,
  rate = rate1,
  lower.tail = TRUE,
  log.p = FALSE
)

rexp_delayed(
  n,
  delay1 = 0,
  rate1 = 1,
  delay2 = NULL,
  rate2 = NULL,
  delay = delay1,
  rate = rate1,
  cens = 0
)

mexp_delayed(
  t = +Inf,
  delay1 = 0,
  rate1 = 1,
  delay2 = NULL,
  rate2 = NULL,
  delay = delay1,
  rate = rate1
)
```

Arguments

x	A numeric vector of values for which to get the density.
delay1	numeric. The first delay, must be non-negative.
rate1	numeric. The event rate, must be non-negative.

delay2	numeric. The second delay, must be non-negative.
rate2	numeric. The second event rate, must be non-negative.
delay	numeric. Alias for first delay.
rate	numeric. Alias for first rate.
log	logical. Return value on log-scale?
q	A numeric vector of quantile values.
lower.tail	logical. Give cumulative probability of lower tail?
log.p	logical. P-value on log-scale?
grad	logical. Should the gradient be calculated at the given quantile values (and not the cumulative probabilities)?
p	A numeric vector of probabilities.
n	integer. Number of random observations requested.
cens	numeric. In [0, 1). Expected proportion of random right-censored observations.
t	A numeric vector of times that restrict the mean survival. Default is +Inf, i.e., the unrestricted mean survival time.

Details

If only a single initial delay phase is there, the numerical arguments other than n are recycled to the length of the result (as with the exponential distribution in stats). With two phases, the arguments are **not** recycled. Only the first element of delays and rates are used as it otherwise becomes ambiguous which delay and rate parameter apply for observations in different phases. Generally, only the first elements of the logical arguments are used.

When cens= is specified greater than 0, the actual number of censored observations is random. On average, it equals the expected number of censored observations as given by cens.

Value

Functions pertaining to the delayed exponential distribution:

- dexp_delayed gives the density
- pexp_delayed gives the vector of cumulative probabilities or the gradient matrix (nbr parameters x quantile times)
- qexp_delayed gives the quantile function
- rexp_delayed generates a pseudo-random sample
- mexp_delayed gives the restricted mean survival time

The length of the result is determined by n for rexp_delayed, and is the maximum of the lengths of the numerical arguments for the other functions, R's recycling rules apply when only single initial delay phase is used.

Vector of cumulative probabilities or gradient matrix (nbr parameter x quantile times)

See Also

stats::Exponential

DelayedWeibull *Delayed Weibull Distribution*

Description

Density, distribution function, quantile function and random generation for the delayed Weibull distribution. Besides the additional parameter `delay`, the other two Weibull-parameters are in principle retained as in R's stats-package:

- `shape`
- `scale` (as inverse of rate)

Usage

```
dweib_delayed(
  x,
  delay1,
  shape1,
  scale1 = 1,
  delay2 = NULL,
  shape2 = NULL,
  scale2 = 1,
  delay = delay1,
  shape = shape1,
  scale = scale1,
  log = FALSE
)

pweib_delayed(
  q,
  delay1,
  shape1,
  scale1 = 1,
  delay2 = NULL,
  shape2 = NULL,
  scale2 = 1,
  delay = delay1,
  shape = shape1,
  scale = scale1,
  lower.tail = TRUE,
  log.p = FALSE,
  grad = FALSE
)

qweib_delayed(
  p,
  delay1,
```

```
  shape1,
  scale1 = 1,
  delay2 = NULL,
  shape2 = NULL,
  scale2 = 1,
  delay = delay1,
  shape = shape1,
  scale = scale1,
  lower.tail = TRUE,
  log.p = FALSE
)

rweib_delayed(
  n,
  delay1,
  shape1,
  scale1 = 1,
  delay2 = NULL,
  shape2 = NULL,
  scale2 = 1,
  delay = delay1,
  shape = shape1,
  scale = scale1,
  cens = 0
)

mweib_delayed(
  t = +Inf,
  delay1,
  shape1,
  scale1 = 1,
  delay2 = NULL,
  shape2 = NULL,
  scale2 = 1,
  delay = delay1,
  shape = shape1,
  scale = scale1
)
```

Arguments

x	A numeric vector of values for which to get the density.
delay1	numeric. The first delay, must be non-negative.
shape1	numeric. First shape parameter, must be positive.
scale1	numeric. First scale parameter (inverse of rate), must be positive.
delay2	numeric. The second delay, must be non-negative.
shape2	numeric. The second shape parameter, must be non-negative.

scale2	numeric. The second scale parameter (inverse of rate), must be positive.
delay	numeric. Alias for first delay.
shape	numeric. Alias for first shape.
scale	numeric. Alias for first scale.
log	logical. Return value on log-scale?
q	A numeric vector of quantile values.
lower.tail	logical. Give cumulative probability of lower tail?
log.p	logical. P-value on log-scale?
grad	logical. Should the gradient be calculated at the given quantile values (and not the cumulative probabilities)?
p	A numeric vector of probabilities.
n	integer. Number of random observations requested.
cens	numeric. Proportion of random right-censored observations. For small values of shape1, on average fewer censorings are achieved. There is still a bug in the CDF for uniform censoring ansatz!
t	A numeric vector of times that restrict the mean survival. Default is +Inf, i.e., the unrestricted mean survival time.

Details

Additional arguments are forwarded via ... to the underlying functions of the exponential distribution in the stats-package.

The numerical arguments other than n are recycled to the length of the result. Only the first elements of the logical arguments are used.

Value

Functions pertaining to the delayed Weibull distribution:

- dweib_delayed gives the density
- pweib_delayed gives the vector of cumulative probabilities or the gradient matrix (nbr parameters x quantile times)
- qweib_delayed gives the quantile function
- rweib_delayed generates a pseudo-random sample
- mweib_delayed gives the restricted mean survival time

The length of the result is determined by n for rweib_delayed, and is the maximum of the lengths of the numerical arguments for the other functions, R's recycling rules apply.

`estimRoundingError` *Estimate rounding error based on given sample of metric values The idea is to check at which level of rounding the sample values do not change.*

Description

Estimate rounding error based on given sample of metric values The idea is to check at which level of rounding the sample values do not change.

Usage

```
estimRoundingError(
  obs,
  roundDigits = seq.int(from = -4L, to = 6L),
  n_obs = 100L
)
```

Arguments

`obs`	numeric. Metric values from a sample to estimate the corresponding rounding error
`roundDigits`	integer. Which level of rounding to test? Negative numbers round to corresponding powers of 10
`n_obs`	integer. How many observations to consider at most? If the provided sample has more observations a sub-sample is used.

Value

estimated rounding error

`getDist` *Get delay distribution function*

Description

Get delay distribution function

Usage

```
getDist(
  distribution = c("exponential", "weibull", "normal"),
  type = c("cdf", "prob", "density", "random", "param"),
  twoPhase = FALSE,
  twoGroup = FALSE,
  bind = NULL,
  profiled = FALSE,
  transformed = FALSE
)
```

Arguments

distribution	character(1). Which distribution?
type	character(1). type of function, cdf: cumulative distribution function, density or random function
twoPhase	logical(1). For type='param', do we model two phases?
twoGroup	logical(1). For type='param', do we have two groups?
bind	character. For type='param', names of parameters that are bind between the two groups.
profiled	logical(1). For type='param', do we request profiling?
transformed	logical(1). For type='param', do we need parameter names transformed (as used inside the optimization function?)

Value

selected distribution function or parameter names

---

getVariance *Get the standard deviation of the fitted delay model fit*

Description

It's similar in notion to variance from package distribution3.

Usage

```
getVariance(object, group = "x", type = c("distribution", "minimum"))
```

Arguments

object	a fitted incubate_fit object
group	the group, "x" or "y"
type	what variance to calculate? Variance of the distribution or variance of the minimum

Details

For two group models you need to specify the group.

Value

predicted variance of distribution or of minimum for the specified group

---

lines.incubate_fit *Add a line fit to a plot of (another)* incubate_fit *object*

Description

Carries over the concept of base-plot `lines` to ggplot. This function returns a ggplot-layer to be added to an existing ggplot-object of an incubate fit. It allows to set aesthetics of the plot manually.

Usage

```
## S3 method for class 'incubate_fit'
lines(x, mapping = NULL, ...)
```

Arguments

`x`	`incubate_fit` object whose model fit is to be added to a ggplot
`mapping`	a `ggplot2::mapping` object. Default to `NULL`.
`...`	further arguments to passed to `geom_function`, outside of the mapping. E.g., `linetype = 'dashed'`

Value

a `geom_function` ggplot-layer

---

logLik.incubate_fit *Extract Log-Likelihood*

Description

The user has the possibility to request different flavours of log-likelihood. By default the flavour matching the fitting method is used.

Usage

```
## S3 method for class 'incubate_fit'
logLik(object, method = NULL, ...)
```

Arguments

object	an incubate_fit object
method	Which flavour of the log-likelihood? By default, it uses the flavour from the model fit
...	further arguments passed on to the object function of the model fit object

Value

Log-likelihood value for the model fit object

---

long2017 *Relapse-free survival of melanoma patients under adjuvant treatment*

Description

Data stem from a double-blind, placebo-controlled phase 3 trial where n=870 patients with completely resected, stage III melanoma with BRAF V600E or V600K mutations were randomly assigned to receive oral Dabrafenib plus Trametinib (combination therapy, 438 patients) or two matched placebo tablets (432 patients) for 12 months.

Usage

```
long2017
```

Format

A data frame with 870 rows and 4 variables:

ID artificially generated patient ID

time Time to relapse-free survival in months

status Status of observation, encoded as 0 for right-censoring vs 1 for RFS-event

trtmt Treatment group: Dabrafenib+Trametinib vs Placebo

Details

Unfortunately, the data set of the clinical trial is not publicly available. Instead, the data were digitized by Sean Devlin based on the published survival plot (see Fig 1A in the original publication of the trial results). Therefore, the data given here do **not** claim to be a 100% faithful representation of the clinical trial. Some deviations are to be expected.

Source

Long GV, Hauschild A, Santinami M, et al. Adjuvant Dabrafenib plus Trametinib in Stage III BRAF-Mutated Melanoma. N Engl J Med. 2017;377(19):1813-1823. doi:10.1056/NEJMoa1708539

Devlin SM and O'Quigley J, The nph2ph-transform: applications to the statistical analysis of completed clinical trials, arXiv:2407.18905, 2024. doi:10.48550/arXiv.2407.18905.

`measles_sailer` *Serial interval times for measles during a long journey on a sailing vessel*

Description

Measles broke out during a sailing ship passage from England to Australia involving six persons on the ship. Serial interval times, i.e. the time of clinical onset between successive cases in a chain of transmission, are provided under the assumption that all transmission pairs represent direct transmissions, excluding co-primary cases and asymptomatic intermediaries. The interval times for the first three cases was not observed completely as they brought measles on board with serial interval times unknown.

Usage

```
measles_sailer
```

Format

A data frame with 6 rows and 4 variables:

generation Disease generation on sailer. Source cases are generation 0.

symptomOnset Days of first symptoms since the start of the journey

serialInterval Days between successive cases in chain of transmission, from symptom to symptom

status Status indicator for serial interval time: 0 right-censored vs 1 for observed

Details

In 1829, the British sailing vessel HMS America carried 176 prisoners from England to New South Wales, Australia. Alexander Stewart, the ship surgeon, recorded cases of measles during the passage in his medical journal. The journey started on 4 March 1829. A guard, who embarked from Chatham (England), was the first measles case when the ship berthed at Woolwich (England) on 28 March 1829. The measles began affecting children of the guards on 31 March 1829 and spreading to some of the soldiers, later.

In the medical journal, it is not clearly said if clinical onset is defined as fever or rash. The first generation of measles on the sailer comprises three cases for which we assume the minimum serial interval time for measles which is generally estimated to be six days (for both, fever-to-fever and for rash-to-rash).

Source

Records of the Admiralty, Naval Forces, Royal Marines, Coastguard, and related bodies, ADM 101/2/3, `https://discovery.nationalarchives.gov.uk/details/r/C4106406`

References

Paterson BJ, Kirk MD, Cameron AS, et al. Historical data and modern methods reveal insights in measles epidemiology, BMJ Open 2013;3:e002033. doi:10.1136/bmjopen2012002033

Fine PE. The interval between successive cases of an infectious disease. Am J Epidemiol. 2003;158(11):1039-1047. doi:10.1093/aje/kwg251

`minObjFunAlt` *Minimize an objective function with alternative optimizer*

Description

The primary optimization routine is BFGS from `stats::optim`. If this fails for some reason we try an alternative which is implemented here. It can use the derivative-free minimizaiton through `bobyqa` or the PORT-routine `nlminb`.

Usage

```
minObjFunAlt(
  objFun,
  start,
  lower = -Inf,
  upper = +Inf,
  verbose = 0,
  method = c("bobyqa", "nlminb")
)
```

Arguments

`objFun`	function to minimize
`start`	vector of start values for parameters
`lower`	numeric. lower bound for parameters (boxed constraint)
`upper`	numeric. upper bound for parameters (boxed constraint)
`verbose`	numeric. Verbosity level
`method`	Specifies which optimizer to use

Details

This is only a thin wrapper to the chosen alternative optimizer.

Value

optimization object with some common entries like `parOpt`, `valOpt` `convergence`, `methodOpt` and `counts`. Or `NULL` in case of failure.

near *Checks if arguments are numerically close.*

Description

The function is vectorized and R's recycling rules apply.

Usage

```
near(x, y)
```

Arguments

x	numeric first vector
y	numeric second vector

Value

logical vector if arguments from x and y are close

See Also

dplyr::near()

objFunFactory *Factory method for objective function*

Description

Given the observed data this factory method produces an objective function which is either the negative of the MPSE-criterion H or some flavour of the negative log-likelihood for MLE. Implemented variants of MLE-objective functions are naive MLE (`'MLEn'`), corrected MLE (`'MLEc'`) or weighted MLE (`'MLEw'`). In any case, the objective function is to be **minimized**.

Usage

```
objFunFactory(
  x,
  y = NULL,
  dist0,
  method = c("MPSE", "MLEn", "MLEc", "MLEw"),
  twoPhase = FALSE,
  bind = NULL,
  control
)
```

Arguments

x	numeric. observations
y	numeric. observations in second group.
distO	distribution object
method	character(1). Specifies the method for which to build the objective function. Default value is MPSE. MLEn is the naive MLE-method, calculating the likelihood function as the product of density values. MLEc is the modified MLE.
twoPhase	logical flag. Do we allow for two delay phases where event rate may change? Default is FALSE, i.e., a single delay phase.
bind	character. parameter names that are bound together (i.e. equated) between both groups
control	list. Fine-tune parameters for optimization. Needs to be set!

Details

The objective function takes a vector of model parameters as argument. From the observations, negative or infinite values are discarded during pre-processing.

Profiling is implemented for single-phase models for Weibull and Exponential distributions: profiling allows to estimate scale1 parameter (resp. rate1= for exponential) based on delay1 (and shape1 for Weibull). Except for MLEw, the formula is derived from the conventional log-likelihood through equating its partial derivative with respect to scale1 to zer0. This leads to the candidate value for scale1. This approach can be used also for the methods MPSE and MLEc. For MLEw (weighted MLE) we have a weighting factor also in the formula for scale1.

Value

the objective function (e.g., the negative MPSE criterion) for given choice of model parameters or NULL upon errors

---

plot.incubate_fit *Plot a fitted delay-model object of class* incubate_fit

---

Description

The fitted delay-model is plotted together with a Kaplan-Meier survival curve. Optionally, a fit of a second delay-model can be added. When given, we do some preliminary checks that the two models do fit together.

Usage

```
## S3 method for class 'incubate_fit'
plot(x, y, title, subtitle, xlim, ...)
```

Arguments

x	a fitted delay-model
y	optionally, a second fitted delay-model
title	character. Optionally, provide a title to the plot.
subtitle	character. Optionally, provide a subtitle to the plot. By default the coefficients are shown.
xlim	numeric. Optionally, limits for the x-axis (time). If unspecified starts from 0 to last observation.
...	further arguments. Not in use here (it is required for generic plot function)

Details

This function requires the ggplot2-package to be installed.

Examples

```
# fit a delay-weibull model to serial interval times from historic measles outbreak data:
fm <- delay_model(survival::Surv(measles_sailer$serialInterval, measles_sailer$status),
                  distribution = "weibull", method = "MLEw")
plot(fm,
     title = "Serial interval times of measles",
     subtitle = "Delay-Weibull model fit using weighted MLE (MLEw)")
```

---

power_diff *Power simulation function for a two-group comparison*

Description

Simulate power for a test of difference between two groups for a given distribution with delay. The effect is specified in terms of the model parameters for both groups. There are two modes of operation:

1. power=NULL: simulate power based on given sample size n (post-hoc power estimation)
2. n=NULL: search iteratively for a suitable sample size n for a given power

Usage

```
power_diff(
  distribution = c("exponential", "weibull"),
  twoPhase = FALSE,
  eff = stop("Provide parameters for both groups that reflect the effect!"),
  param = "delay1",
  test = c("bootstrap", "logrank", "logrank_pp", "LRT"),
  method = c("MPSE", "MLEw", "MLEc", "MLEn"),
  n = NULL,
```

```
  r = 1,
  sig.level = 0.05,
  power = NULL,
  nPowerSim = 1600,
  R = 200,
  nRange = c(5, 250),
  verbose = 0
)
```

Arguments

distribution	character. Which assumed distribution is used for the power calculation. Default is "exponential".
twoPhase	logical(1). Do we model two phases per group? Default is FALSE, i.e. a single delay phase per group.
eff	list of length 2. The two list elements must be numeric vectors that contain the model parameters (as understood by the delay-distribution functions provided by this package) for the two groups.
param	character. Parameter name(s) which are to be tested for difference and for which to simulate the power. Default value is 'delay1'. You can specify multiple parameters, by giving a vector or by concatenating them with a + in a single string. For logrank tests, this argument is ignored.
test	character. Which test to use for this power estimation? Defaults to "bootstrap". Non-parametric logrank test is also possible (either "logrank" or "logrank_pp"). See also test_diff().
method	character. Which fitting method to use in case of a parametric test.
n	integer. Number of observations per group for the power simulation or NULL when n is to be estimated for a given power.
r	numeric. Ratio of both groups sizes, ny / nx. Default value is 1, i.e. balanced group sizes. Must be positive.
sig.level	numeric. Significance level. Default is 0.05.
power	numeric. NULL when power is to be estimated for a given sample size or a desired power is specified (and n is estimated).
nPowerSim	integer. Number of simulation rounds. Default value 1600 yields a standard error of 0.01 for power if the true power is 80%.
R	integer. Number of bootstrap samples for test of difference within each power simulation. It affects the resolution of the P-value for each simulation round. A value of around R=200 gives a resolution of 0.5% which might be enough for power analysis.
nRange	integer. Admissible range for sample size when power is specified and sample size is requested. The routine might not find the optimal sample size when this range is set too wide.
verbose	numeric. How many details are requested? Higher value means more details. 0=off, no details.

Details

The power is estimated by simulating data according to the specified model and testing for differences in the simulated data. The proportion of simulated datasets where the test rejects the null hypothesis at the given significance level is the estimated power. The test can be a parametric bootstrap test or a non-parametric logrank test. For logrank tests, the `param=` argument should not be specified. Specify the effect size (`eff=`) as a list with two elements, each element holds the parameter vector of the distribution of the response variable of one group. The fitting method (`method=`) and the kind of significance test (`test=`) are handed down to `test_diff()`. The more power simulation rounds (parameter `nPowerSim=`) the more densely the space of possible data according to the specified model is sampled.

Note that estimating sample size `n` is computationally intensive. The iterative search uses heuristics to find a sample size `n` within the provided range (`nRange=`), and the estimated sample size might yield a slightly different power level. Hence, check the reported power in the output. The search algorithm comes to better results when the admissible range for sample size (`nRange=`) is chosen sensibly and not too wide. In case the estimated sample size and the achieved power is too high it might pay off to rerun the function with an adapted admissible range for the sample size by giving a narrower range in `nRange=`.

Value

List of results of power simulation. Or `NULL` in case of errors.

See Also

`test_diff()`

Examples

```
# Simulate power for a given sample size:
# test for any difference in a delay-exponential model using logrank test
# the assumed effect is given in terms of model parameters for both groups
power_diff(
  eff = list(ctrl = c(delay1 = 5, rate1 = .09),
             trtm = c(delay1 = 7, rate1 = .12)),
  test = "logrank",
  n = 16, power = NULL, nPowerSim = 300)

## Not run:
# test for difference in delay in a delay-exponential model via a bootstrap test:
# the power is estimated based on nPowersim = 520 simulated datasets
# for real applications, use a higher nPowerSim (e.g. 1600) for more
# precise power estimation and a higher R (e.g. 400) for more precise
# P-value estimation in each simulation round
set.seed(123) # for reproducibility
power_diff(
  eff = list(grA = c(delay1 = 5.2, rate1 = .1),
             grB = c(delay1 = 7, rate1 = .12)),
  param = "delay1",
  test = "bootstrap", method = "MPSE",
  n = 16, power = NULL,
```

```
  nPowerSim = 520, R = 160)

# test for difference in rate in a delay-exponential model via a bootstrap test:
# required sample size is estimated for a given power.
# provide a suitable range for the sample size search via nRange=.
# The search for n takes more time than the previous example, as the function
# iteratively evaluates different sample sizes in order to find the right one.
set.seed(1234) # for reproducibility
power_diff(
  eff = list(grA = c(delay1 = 5, rate1 = .07),
             grB = c(delay1 = 7, rate1 = .18)),
  param = "rate1",
  test = "bootstrap", method = "MPSE",
  n = NULL, power = 0.8,
  nPowerSim = 480, R = 150,
  nRange = c(18, 48))

## End(Not run)
```

`prepResponseVar` *Check and prepare the survival response(s)*

Description

Allowed censoring types are right-, left-, and interval-censoring. If `y0` is not `NULL` this function will return either both numeric, non-Surv or both Surv-objects of the same type.

Usage

```
prepResponseVar(x0, y0 = NULL, simplify = TRUE)
```

Arguments

`x0`	response as numeric or survival::Surv using left, right or interval-coding
`y0`	response as numeric or survival::Surv using left, right or interval-coding
`simplify`	logical. Should the result be as simple as possible? If `FALSE`, result will be in any case survival::Surv objects.

Value

a list of the two responses, either both as survival::Surv or plain numeric (if no censorings and `simplify=TRUE`)

rockette74 *Small data sets from miscellaneous publications*

Description

Most data sets come from publications about parameter estimation in Weibull models. See the references below, in section "Source".

Usage

```
rockette74

fatigue

susquehanna

pollution

graphite
```

Format

An object of class `numeric` of length 4.

An object of class `numeric` of length 10.

An object of class `numeric` of length 20.

An object of class `numeric` of length 20.

An object of class `numeric` of length 41.

Details

The following small data sets are provided as numeric vectors.

`rockette`: Artificial sample of length 4 given by Rockette. The maximum likelihood function has two stationary points, none of them is the global maximum.

`fatigue`: Fatigue times of ten bearings of a specific type in hours.

`susquehanna`: Maximum flood levels (in millions of cubic feet per second) for the Susquehanna River of Harrisburg (Pennsylvania, USA) over 20 4-year periods.

`pollution`: Beach pollution levels in South Wales (measured in number of coliform per 100 ml) on 20 days over a 5-week period.

`graphite`: Breaking stress (in MPa x 10^6) of 41 beam specimens cut from a single graphite H590 block, from a reliability study reported by Margetson & Cooper (1984), cited by Cheng & Stephen (1989)

Source

Different publications related to estimating Weibull data.

References

McCool, J.I., 1974. Inferential techniques for Weibull populations. Technical Report TR 74-0180, Wright Patterson Air Force Base, Ohio.

Rockette, H., 1974. Maximum Likelihood Estimation with the Weibull Model.

Dumonceaux, R. and Antle, C. E., 1973. Discrimination between the lognormal and the Weibull distributions. Technometrics, 15, 923-926.

Steen, P. J. and Stickler, D. J., 1976. A Sewage Pollution Study of Beaches from Cardiff to Ogmore. Report January 1976, Cardiff: Department of Applied Biology, UWIST.

Cheng, R.C.H. and Stephen, M.A., 1989. A Goodness of Fit Test Using Moran's Statistic with Estimated Parameters. Biometrika, 76, 386-392.

---

scalePars *Calculate parameter scaling for optimization routine.*

Description

The scale per parameter corresponds to the step width within the optimization path.

Usage

```
scalePars(parV, lowerB = 0.00001, upperB = 100000)
```

Arguments

parV	named numeric parameter vector for optimization
lowerB	numeric. lower bound for parameter scales
upperB	numeric. upper bound for parameter scales

Value

numeric. vector of parameter scaling

---

stankovic *Survival time of mice with glioma under different treatments*

Description

This data set stems from an animal experiment described in Stankovic (2018). In particular, the data in question is shown in Figure 6J and 6K.

Usage

```
stankovic
```

Format

A data frame with 45 rows and 5 variables:

Figure The figure in the publication where the data is shown

Time Survival in days

Status Right-censor status: 1 means observed event

Group Experimental group identifier

Colour Colour used in the Stankovic publication to mark this group

Details

The data were read directly from the survival plots in the publication with the help of Plot Digitizer, version 2.6.9.

Source

Dudvarski Stankovic N, Bicker F, Keller S, et al. EGFL7 enhances surface expression of integrin a5b1 to promote angiogenesis in malignant brain tumors. EMBO Mol Med. 2018;10(9):e8420. doi:10.15252/emmm.201708420 https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6127886/

---

test_diff *Test the difference for model parameter(s) between two uncorrelated groups*

Description

The test is in fact a model comparison between a null model where the parameters are enforced to be equal and an unconstrained full model. The model parameters can be fit with various methods: MPSE, MLEn, MLEc or MLEw. Parametric bootstrap tests and likelihood ratio tests are supported. As test statistic for the bootstrap test we use twice the difference in best (=lowest) objective function value, i.e. `2 * (val_0 - val_1)`. The factor 2 does not matter but it becomes reminiscent of a likelihood ratio test statistic albeit the objective function is not a negative log-likelihood in all cases (e.g. it is the negative of the maximum product spacing metric for the MPSE-method).

Usage

```
test_diff(
  x,
  y = stop("Provide data for group y!"),
  distribution = c("exponential", "weibull"),
  twoPhase = FALSE,
  type = c("all", "bootstrap", "GOF", "moran", "pearson", "logrank", "LRT"),
  param = "delay1",
  method = c("MPSE", "MLEw", "MLEc", "MLEn"),
  profiled = method != "MPSE",
  ties = c("density", "equispaced", "error"),
```

```
  doLogrank = TRUE,
  R = 400,
  chiSqApprox = FALSE,
  verbose = 0
)
```

Arguments

x	data from reference/control group.
y	data from the treatment group.
distribution	Name of the distribution to use or distribution object.
twoPhase	logical(1). Do we model two phases per group? Default is FALSE, i.e. a single delay phase per group.
type	character. Which type of tests to perform?
param	character. Names of parameters to test difference for. Default value is 'delay1'. You can specify multiple parameters, by providing multiple parameter names or by concatenating them with a + in a single string. Ignored for non-parametric tests.
method	character. Which method to fit the models.
profiled	logical. Use the profiled likelihood?
ties	character. How to handle ties in data vector of a group?
doLogrank	logical. Do also non-parametric logrank tests?
R	numeric(1). Number of bootstrap samples to evaluate the distribution of the test statistic.
chiSqApprox	logical flag. In bootstrap, should we estimate the best degrees of freedom for chi-square to match the distribution of the test statistic under H0?
verbose	numeric. How many details are requested? Higher value means more details. 0=off, no details.

Details

High values of this difference speak against the null model (i.e., high val_0 indicates bad fit under the null model0 and/or low values of val_1 indicate a good fit under the more general model1. The test is implemented as a parametric bootstrap test, i.e., we

1. take the given null-model fit as ground truth
2. regenerate data according to this model fit
3. recalculate the test statistic
4. appraise the observed test statistic in light of the generated distribution under H0

Value

list with the results of the test. Element P contains the different P-values, for instance from parametric bootstrap

Examples

```
set.seed(123)
# generate example data
grA <- rweib_delayed(n = 70, delay1 = 5, shape1 = 2, scale1 = 8)
grB <- rweib_delayed(n = 60, delay1 = 7, shape1 = 1.8, scale1 = 6)

# difference in delay parameter is significant at 5% level
test_diff(x = grA, y = grB,
  distribution = "weibull", param = "delay1",
  type = "bootstrap", method = "MPSE", R = 50)

# but the non-parametric logrank test is not significant
# no need to specify parameters
test_diff(x = grA, y = grB,
  type = "logrank")
```

---

test_GOF *Goodness-of-fit (GOF) test statistic (experimental!)*

Description

The GOF-test is performed for a fitted delay-model that was fit using MPSE. There are different GOF-tests implemented:

- **Moran GOF** is based on spacings, like the MPSE-criterion itself.
- **Pearson GOF** uses categories and compares observed to expected frequencies.

Usage

```
test_GOF(
  delayFit,
  method = c("moran", "pearson", "nikulin", "NRR"),
  estimated = TRUE,
  verbose = 0
)
```

Arguments

delayFit	delay_model fit object
method	character(1). which method to use for GOF. Default is 'moran'.
estimated	flag. Moran test: was the parameter estimated?
verbose	integer. Verbosity level. The higher the more verbose debugging output.

Details

Note that the GOF-tests are currently only implemented for models fitted with maximum product of spacings estimation (MPSE). These tests (Moran & Pearson) are still experimental. So, use with caution. Experimental code!

Value

An `htest`-object containing the GOF-test result

`transform.incubate_fit`

Transform observed data to unit interval

Description

The transformation used is the probability integral transform: the cumulative distribution function with the estimated parameters of the model fit takes the data into the 0-1 interval. All available data in the model fit is transformed. Censored observations lead to censored back-transformed observations as well.

Usage

```
## S3 method for class 'incubate_fit'
transform(`_data`, ...)
```

Arguments

`_data`	a fitted model object of class `incubate_fit`
`...`	currently ignored

Value

The transformed data, either a vector (for single group) or a list with entries x and y (in two group scenario)

Note

This S3-method implementation is quite different from its default method that allows for non-standard evaluation on data frames, primarily intended for interactive use. But the name `transform` fits so nicely to the intended purpose that it is re-used for the probability integral transform, here.

update.incubate_fit *Refit an* incubate_fit*-object with specified optimization arguments*

Description

This function is useful when only an optimization argument is to be changed. If more things need to be changed go back to delay_model and start from scratch.

Usage

```
## S3 method for class 'incubate_fit'
update(object, optim_args = NULL, verbose = 0, ...)
```

Arguments

object	incubate_fit-object
optim_args	optimization arguments
verbose	integer flag. Requested verbosity during delay_fit
...	further arguments, currently not used.

Value

The updated fitted object of class incubate_fit or NULL in case of failure.

w1Fint *Internal MLEw-weight W1 function according to sampling distribution Weight W1 for given sample sizes (of one group). For small* nObs *we use direct results from Monte-Carlo simulation. For higher* nObs *we use an approximation (based on Wilson-Hilferty transformation).*

Description

Note that in R there is the numeric routine stats::qgamma which could also be used as in stats::qgamma(p=.5, shape = 10, rate = 10) for n=10.

Usage

```
w1Fint(nObs)
```

Arguments

nObs	numeric. number of observations (vectorized)

Value

numeric. W1-value corrsponding to nObs. Same length as nObs

`w2Fint` *Internal MLEw weight function W2 according to sampling distribution W2 is either taken directly from the MCSS-results (if available) or are taken from the smooth approximation function, otherwise.*

Description

Internal MLEw weight function W2 according to sampling distribution W2 is either taken directly from the MCSS-results (if available) or are taken from the smooth approximation function, otherwise.

Usage

```
w2Fint(nObs)
```

Arguments

`nObs`	numeric. Sample size (vectorized) of one group

Value

W2 (same size as nObs)

`w3FFint` *Internal factory method for W3-function based on sampling distribution*

Description

Generally, the weight W3 depends on the sample size and the shape parameter. The sample size of a group is fixed. Hence, we return a function that returns W3 for provided shape parameter as argument. If the sample size was used during the Monte-Carlo simulation study the coefficients of generalized logistic curve are returned directly. Otherwise the coefficients stem from a natural cubic spline fit based on the MCS.

Usage

```
w3FFint(nObs)
```

Arguments

`nObs`	sample size for which to build the W3-function

Details

We run the cubic spline fit once when the package is loaded. This guarantees that the spline function of the R-version of the current user is used and hopefully with good performance. Function `w3FFint` is run repeatedly by the `objFunFactory()`, once per group.

Value

W3-function for the given sample size. It is a function of shape.

Index