Package: incubate (via r-universe)

September 15, 2024

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

Version 1.3.0

Date 2024-08-16

Description Fit parametric models for time-to-event data that show an initial 'incubation period', i.e., a variable delay phase where the hazard is zero. The delayed Weibull distribution serves as foundational data model. The specific method of 'MPSE' (maximum product of spacings estimation) and MLE-based methods are used for parameter estimation. Bootstrap confidence intervals for parameters and significance tests in a two group setting are provided.

License LGPL (>= 3)

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

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

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

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

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

RoxygenNote 7.3.2

Config/testthat/edition 3

Depends R (>= 3.5.0)

Collate 'data.R' 'delay_estimation.R' 'delay.R' 'delay_test.R' 'incubate-package.R' 'utils.R'

Repository https://lenz99.r-universe.dev

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

RemoteRef HEAD

RemoteSha 1330b240075acffa41c8c8aff6651bcee256f2f0

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 formated 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 fitted incubate model.*

Description

Bootstrap data are here estimated coefficients from models fitted to bootstrap samples. 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 = 0.25
)
```

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: influence the amount of smoothing. 0 means no smoothing at all. Default is 0.25 (as was optimal in 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)

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 incubate-model fits.*

Description

Bias-corrected bootstrap confidence limits (either quantile-based or normal-approximation based) are generated. Optionally, there are also variants that use a log-transformation first. 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. 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.

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,
  ...
)

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
)

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.
...	further arguments are passed on to the underlying non-delayed function, e.g., lower.tail= to stats::pexp()
p	A numeric vector of probabilities.
lower.tail	logical. Give cumulative probability of lower tail?
log.p	logical. P-value on log-sclae?

n	integer. Number of random observations requested.
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. 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.

Value

Functions pertaining to the delayed exponential distribution:

- dexp_delayed gives the density
- pexp_delayed gives the distribution function
- 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.

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
)

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
)

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-sclae?
p	A numeric vector of probabilities.
n	integer. Number of random observations requested.
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 distribution function
- 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.

---

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. or the analytical solution is returned directly if available.

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.

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 biased estimates. MLEc is a corrected version of MLE due to Cheng.

Usage

```
delay_model(
  x = stop("Specify observations for at least one group x=!", call. = FALSE),
  y = NULL,
  distribution = c("exponential", "weibull"),
  twoPhase = FALSE,
  bind = NULL,
  ties = c("density", "equidist", "random", "error"),
  method = c("MPSE", "MLEn", "MLEw", "MLEc"),
  profiled = method == "MLEw",
  optim_args = NULL,
  verbose = 0
)
```

Arguments

`x`	numeric. observations of 1st group. Can also be a list of data from two groups.
`y`	numeric. observations from 2nd group
`distribution`	character. Which delayed distribution is assumed? Exponential or Weibull.
`twoPhase`	logical. Allow for two phases?
`bind`	character. parameter names that are bind together in 2-group situation.
`ties`	character. How to handle ties.
`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
`profiled`	logical. Profile out scale from log-likelihood if possible.
`optim_args`	list. optimization arguments to use. Use `NULL` to use the data-dependent default values.
`verbose`	integer. level of verboseness. Default 0 is quiet.

Details

Numerical optimization is done by `stats::optim`.

Value

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

`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(-4L, 6L), maxObs = 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
maxObs	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"),
  type = c("cdf", "prob", "density", "random", "param"),
  twoPhase = FALSE,
  twoGroup = FALSE,
  bind = NULL,
  profiled = FALSE,
  transformed = FALSE
)
```

Arguments

distribution	character(1). delay 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

---

minObjFunPORT *Minimize an objective function with PORT routine (nlminb)*

Description

Minimize an objective function with PORT routine (nlminb)

Usage

```
minObjFunPORT(objFun, start, lower = -Inf, upper = +Inf, verbose = 0)
```

Arguments

objFun	objective function
start	numeric vector of parameter values to start optimization
lower	numeric. lower bound for parameters (boxed constraint)
upper	numeric. upper bound for parameters (boxed constraint)
verbose	numeric. Verbosity level.

`objFunFactory` *Factory method for objective function, either according to maximum product of spacings estimation ('MPSE') or according to some flavour of maximum likelihood estimation (e.g., naive ('MLEn') or corrected ('MLEc') or weighted ('MLEw') MLE).*

Description

Given the observed data this factory method produces an objective function which is either the negative of the MPSE-criterion H or the negative log-likelihood for MLE.

Usage

```
objFunFactory(
  x,
  y = NULL,
  distribution = c("exponential", "weibull"),
  twoPhase = FALSE,
  bind = NULL,
  method = c("MPSE", "MLEn", "MLEc", "MLEw"),
  profiled = FALSE,
  ties = c("density", "equidist", "random", "error"),
  verbose = 0
)
```

Arguments

`x`	numeric. observations
`y`	numeric. observations in second group.
`distribution`	character(1). delayed distribution family
`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 bind together (i.e. equated) between both groups
`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.
`profiled`	logical. Should scale parameter be profiled out prior to optimization?
`ties`	character. How to handle ties within data of a group.
`verbose`	integer flag. How much verbosity in output? The higher the more output. Default value is 0 which is no output.

Details

The objective function takes a vector of model parameters as argument.

From the observations, negative or infinite values are discarded during pre-processing. In any case, the objective function is to be **minimized**.

Value

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

---

power_diff *Power simulation function for a two-group comparison of the delay parameter.*

Description

There are two ways of operation:

1. `power=NULL` Given sample size `n` it simulates the power.
2. `n=NULL` Given a power an iterative search is started to find a suitable `n` within a specified range.

Usage

```
power_diff(
  distribution = c("exponential", "weibull"),
  twoPhase = FALSE,
  param = "delay1",
  test = c("bootstrap", "pearson", "moran", "logrank", "logrank_pp", "LR"),
  eff = stop("Provide parameters for both groups that reflect the effect!"),
  n = NULL,
  r = 1,
  sig.level = 0.05,
  power = NULL,
  nPowerSim = 1600,
  R = 201,
  nRange = c(5, 50),
  verbose = 0
)
```

Arguments

`distribution`	character. Which assumed distribution is used for the power calculation.
`twoPhase`	logical(1). Do we model two phases per group? Default is `FALSE`, i.e. a single delay phase per group.
`param`	character. Parameter name(s) which are to be tested for difference and for which to simulate the power. Default value is `'delay1'`.

test	character. Which test to use for this power estimation?
eff	list. The two list elements contain the model parameters (as understood by the delay-distribution functions provided by this package) for the two groups.
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 in parameter 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 pre-specified and sample size is requested.
verbose	numeric. How many details are requested? Higher value means more details. 0=off, no details.

Details

In any case, the distribution, the parameters that are tested for, the type of test and the effect size (eff=) need to be specified. The more power simulation rounds (parameter nPowerSim=) the more densely the space of data according to the specified model is sampled.

Note that this second modus (when n is estimated) is computationally quite heavy. The iterative search for n uses some heuristics and the estimated sample size might actually give a different power-level. It is important to check the stated power in the output. The search algorithm comes to results closer to the power aimed at when the admissible range for sample size (nRange=) is chosen sensibly. 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.

Value

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

---

publication_examples *Small data sets from different publications*

Description

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

Usage

```
publication_examples

fatigue

susquehanna

pollution
```

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.

Details

These 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.

Source

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.

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

vector of parameter scaling

stankovic *Survival 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

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 delay model parameter(s) between two uncorrelated groups, based on maximum product of spacings estimation (MPSE).*

Description

It is in fact a model comparison between a null model where the parameters are enforced to be equal and an unconstrained full model. As test statistic we use twice the difference in best (=lowest) objective function value, i.e. 2 * (`val_0` - `val_1`). This is reminiscent of a likelihood ratio test statistic albeit the objective function is not a negative log-likelihood but the negative of the maximum product spacing metric.

Usage

```
test_diff(
  x,
  y = stop("Provide data for group y!"),
  distribution = c("exponential", "weibull"),
  twoPhase = FALSE,
  method = c("MPSE", "MLEn", "MLEw", "MLEc"),
  profiled = method == "MLEw",
  ties = c("density", "equidist", "random", "error"),
  param = "delay1",
  type = c("all", "bootstrap", "GOF", "moran", "pearson", "logrank", "LR"),
  doLogrank = TRUE,
  R = 400,
  chiSqApprox = FALSE,
  verbose = 0
)
```

Arguments

`x`	data from reference/control group.
`y`	data from the treatment group.
`distribution`	character(1). Name of the parametric delay distribution to use.
`twoPhase`	logical(1). Do we model two phases per group? Default is `FALSE`, i.e. a single delay phase per group.
`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?
param	character. Names of parameters to test difference for. Default value is 'delay1'.
type	character. Which type of tests to perform?
doLogrank	logical. In any case do log-rank based tests?
R	numeric(1). Number of bootstrap samples to evaluate the distribution of the test statistic.
chiSqApprox	logical flag. In bootstrap, should we calculate the approximate degrees of freedom for 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 0-model and 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 given null-model fit as ground truth
2. regenerate data according to this model.
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

---

test_GOF *Goodness-of-fit (GOF) test statistic.*

Description

The GOF-test is performed for a fitted delay-model. 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"))
```

Arguments

delayFit	delay_model fit
method	character(1). which method to use for GOF. Default is 'moran'.

Value

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

`transform.incubate_fit` *Transform observed data to unit interval*

Description

The transformation is the probability integral transform. It uses the cumulative distribution function with the estimated parameters of the model fit. All available data in the model fit is transformed.

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. If more things need to be changed go back to* `delay_model` *and start from scratch.*

Description

Refit an `incubate_fit`-object with specified optimization arguments. 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

Index