

Package: marp (via r-universe)

September 21, 2026

Version 0.1.1

Type Package

Title Model-Averaged Renewal Process

Maintainer Veronica Tsou <wantungtsou@gmail.com>

Description To implement a model-averaging approach with different renewal models, with a primary focus on forecasting large earthquakes. Based on six renewal models (i.e., Poisson, Gamma, Log-Logistics, Weibull, Log-Normal and BPT), model-averaged point estimates are calculated using AIC weights. Additionally, both percentile and studentized bootstrapped model-averaged confidence intervals are constructed. In comparison, point and interval estimation from the individual or ``best" model (determined via model selection) can be retrieved.

URL <https://github.com/kanji709/marp>

BugReports <https://github.com/kanji709/marp/issues>

Depends R (>= 2.15)

Imports stats, gtools, statmod, VGAM,

Suggests knitr, rmarkdown, devtools, roxygen2, testthat (>= 3.0.0)

License MIT + file LICENSE

Encoding UTF-8

Config/testthat/edition 3

Config/roxygen2/version 8.0.0

VignetteBuilder knitr

Repository <https://kanji709.r-universe.dev>

Date/Publication 2026-08-22 00:16:30 UTC

RemoteUrl <https://github.com/kanji709/marp>

RemoteRef HEAD

RemoteSha cc24bf6fb582050dbd1b7f2ab9ee266492af870b

Contents

bpt_bstrp	3
bpt_logl	4
bpt_rp	5
coef.marp_model_fit	6
confint.marp_fit	7
dllog	8
gamma_bstrp	8
gamma_logl	10
gamma_rp	11
logLik.marp_model_fit	12
loglogis_bstrp	12
loglogis_logl	14
loglogis_rp	15
lognorm_bstrp	16
lognorm_rp	18
lowerT	19
marp	20
marp_bstrp	21
marp_confint	23
nobs.marp_fit	26
nobs.marp_model_fit	27
percent_confint	27
pllog	29
poisson_bstrp	30
poisson_rp	31
print.marp_confint	32
print.marp_fit	33
print.marp_model_fit	33
print.summary_marp_fit	34
print.summary_marp_model_fit	34
student_confint	35
summary.marp_fit	37
summary.marp_model_fit	38
upperT	38
weibull_bstrp	39
weibull_logl	41
weibull_rp	42

Index	44
--------------	-----------

bpt_bstrp	<i>A function to generate (double) bootstrap samples and fit BPT renewal model</i>
-----------	--

Description

A function to generate (double) bootstrap samples and fit BPT renewal model

Usage

```
bpt_bstrp(n, t, B, BB, m, par_hat, mu_hat, pr_hat, haz_hat, y)
```

Arguments

n	Number of inter-event times generated in each bootstrap sample.
t	Time points at which log-hazards are evaluated.
B	Number of bootstrap samples.
BB	Number of double-bootstrap samples per bootstrap sample.
m	Positive integer controlling repeated random-start optimizations; see [marp()].
par_hat	Length-12 vector containing 'par1' for models 1–6 followed by 'par2' for models 1–6; see [marp()] for model order.
mu_hat	Length-6 vector of model-specific mean estimates.
pr_hat	Length-6 vector of model-specific logit event probabilities.
haz_hat	Matrix of model-specific log-hazards with 'length(t)' rows and six model columns.
y	Time point at which logit event probabilities are evaluated.

Value

A list of bootstrap estimates and variance/T-statistic quantities. Components beginning with 'pr_' use the logit-probability scale and those beginning with 'haz_' use the log-hazard scale. If the supplied BPT fit is unavailable, the same structure is returned with 'NA' values.

mu_star Estimated mean from bootstrapped samples

pr_star Logit event probabilities from bootstrap samples

haz_star Log-hazards from bootstrap samples

mu_var_hat Variance of estimated mean

pr_var_hat Variance of estimated probability

haz_var_hat Variance of estimated hazard rates

mu_var_double Variance of estimated mean of bootstrapped samples (via double-bootstrapping)

pr_var_double Variance of estimated probability of bootstrapped samples (via double-bootstrapping)

haz_var_double Variance of estimated hazard rates of bootstrapped samples (via double-bootstrapping)

mu_Tstar Pivot quantity of the estimated mean

pr_Tstar Pivot quantity of the estimated probability

haz_Tstar Pivot quantity of the estimated hazard rates

Examples

```
## Not run:
# set some parameters
n <- 30 # sample size
t <- seq(100, 200, by = 10) # time intervals
B <- 100 # number of bootstraps
BB <- 100 # number of double-bootstraps
m <- 10 # repeated random-start optimization setting
par_hat <- c(
  3.41361e-03, 2.76268e+00, 2.60370e+00, 3.30802e+02, 5.48822e+00, 2.92945e+02, NA,
  9.43071e-03, 2.47598e+02, 1.80102e+00, 6.50845e-01, 7.18247e-01
)
mu_hat <- c(292.94512, 292.94513, 319.72017, 294.16945, 298.87286, 292.94512)
pr_hat <- c(0.60039, 0.42155, 0.53434, 0.30780, 0.56416, 0.61795)
haz_hat <- matrix(c(
  -5.67999, -5.67999, -5.67999, -5.67999, -5.67999, -5.67999,
  -5.67999, -5.67999, -5.67999, -5.67999, -5.67999, -6.09420,
  -5.99679, -5.91174, -5.83682, -5.77031, -5.71085, -5.65738,
  -5.60904, -5.56512, -5.52504, -5.48833, -6.09902, -5.97017,
  -5.85769, -5.75939, -5.67350, -5.59856, -5.53336, -5.47683,
  -5.42805, -5.38621, -5.35060, -6.17146, -6.09512, -6.02542,
  -5.96131, -5.90194, -5.84668, -5.79498, -5.74642, -5.70064,
  -5.65733, -5.61624, -5.92355, -5.80239, -5.70475, -5.62524,
  -5.55994, -5.50595, -5.46106, -5.42359, -5.39222, -5.36591,
  -5.34383, -5.79111, -5.67660, -5.58924, -5.52166, -5.46879,
  -5.42707, -5.39394, -5.36751, -5.34637, -5.32946, -5.31596
),length(t),6)
y <- 304 # cut-off point for probablity estimation

# generate bootstrapped samples then fit renewal model
res <- marp::bpt_bstrp(n, t, B, BB, m, par_hat, mu_hat, pr_hat, haz_hat, y)

## End(Not run)
```

bpt_logl

*A function to calculate the log-likelihood of BPT model***Description**

A function to calculate the log-likelihood of BPT model

Usage

```
bpt_logl(param, x)
```

Arguments

param Length-2 numeric vector containing log(mean) and log(alpha squared), where alpha is the BPT aperiodicity parameter.

x Numeric vector of positive observations.

Value

returns the value of negative log-likelihood of the BPT model

Examples

```
set.seed(42)
data <- rgamma(30,3,0.01)

# set some parameters
par_hat <- c(292.945125794581, 0.718247184450307) # estimated parameters
param <- c(log(par_hat[1]),log(par_hat[2]^2)) # input parameters for logl function

# calculate log-likelihood
result <- marp::bpt_logl(param, data)

result
```

bpt_rp	<i>A function to fit BPT renewal model</i>
--------	--

Description

A function to fit BPT renewal model

Usage

```
bpt_rp(data, t, m, y)
```

Arguments

data	A numeric vector of positive inter-event times.
t	A numeric vector of time points at which log-hazards are evaluated.
m	A positive integer controlling repeated random-start optimizations; the current implementation seeks 'm - 1' acceptable 'nlm()' fits and caps the number of attempted starts at '20 * m'.
y	A time point at which the logit-transformed cumulative event probability is evaluated.

Value

An object of class ‘marp_model_fit’ retaining the following eight named list components:

par1 Estimated mean parameter (μ) of the BPT model
par2 Estimated aperiodicity parameter (α) of the BPT model
logL Maximized log-likelihood
AIC Akaike information criterion (AIC)
BIC Bayesian information criterion (BIC)
mu_hat Estimated mean inter-event time
pr_hat Logit-transformed cumulative event probability at ‘y’
haz_hat Log-hazard values at ‘t’

If the requested plausible fits cannot be obtained within the existing attempt limit, the function warns and returns the same components with unavailable estimates set to ‘NA’ and information criteria set to ‘Inf’.

Examples

```
set.seed(42)
data <- rgamma(30, 3, 0.01)

# set some parameters
m <- 10 # repeated random-start optimization setting
t <- seq(100, 200, by=10) # time intervals
y <- 304 # cut-off year for estimating probability

# fit BPT renewal model
fit <- marp::bpt_rp(data, t, m, y)
fit
summary(fit)
```

coef.marp_model_fit	<i>Extract parameters from a model-specific fit</i>
---------------------	---

Description

Extract parameters from a model-specific fit

Usage

```
## S3 method for class 'marp_model_fit'
coef(object, ...)
```

Arguments

object	A 'marp_model_fit' object.
...	Additional arguments, currently unused.

Value

A named numeric vector of fitted distribution parameters.

confint.marp_fit	<i>Confidence intervals for a model-averaged renewal-process fit</i>
------------------	--

Description

This delegates to [marp_confint()] without changing its bootstrap calculations. The original data are deliberately not stored in 'object', so they must be supplied explicitly.

Usage

```
## S3 method for class 'marp_fit'
confint(object, parm = NULL, level = 0.95, data, B, BB, ...)
```

Arguments

object	A 'marp_fit' object.
parm	Currently unsupported. It must be 'NULL'.
level	Confidence level. The default is 0.95.
data	The original inter-event times used to fit 'object'.
B	Number of bootstrap samples.
BB	Number of double-bootstrap samples.
...	Additional arguments passed to [marp_confint()].

Value

The existing nested confidence-interval result returned by [marp_confint()], with class 'marp_confint'.

dllog	<i>Density function of Log-Logistics model</i>
-------	--

Description

Density function of Log-Logistics model

Usage

```
dllog(x, shape = 1, scale = 1, log = FALSE)
```

Arguments

x	Numeric vector of positive quantiles.
shape	Positive shape parameter of the log-logistic distribution.
scale	Positive scale parameter of the log-logistic distribution.
log	Logical; if 'TRUE', return log-densities.

Value

A numeric vector of densities, or log-densities when 'log = TRUE'.

Examples

```
x <- as.numeric(c(350., 450., 227., 352., 654.))
# set paramters
shape <- 5
scale <- 3
log <- FALSE
result_1 <- marp::dllog(x, shape, scale, log)

# alternatively, set log == TRUE
log <- TRUE
result_2 <- marp::dllog(x, shape, scale, log)
```

gamma_bstrp	<i>A function to generate (double) bootstrap samples and fit Gamma renewal model</i>
-------------	--

Description

A function to generate (double) bootstrap samples and fit Gamma renewal model

Usage

```
gamma_bstrp(n, t, B, BB, m, par_hat, mu_hat, pr_hat, haz_hat, y)
```

Arguments

n	Number of inter-event times generated in each bootstrap sample.
t	Time points at which log-hazards are evaluated.
B	Number of bootstrap samples.
BB	Number of double-bootstrap samples per bootstrap sample.
m	Positive integer controlling repeated random-start optimizations; see [marp()].
par_hat	Length-12 vector containing 'par1' for models 1–6 followed by 'par2' for models 1–6; Gamma uses shape and rate.
mu_hat	Length-6 vector of model-specific mean estimates.
pr_hat	Length-6 vector of model-specific logit event probabilities.
haz_hat	Matrix of model-specific log-hazards with 'length(t)' rows and six model columns.
y	Time point at which logit event probabilities are evaluated.

Value

A list of bootstrap estimates and variance/T-statistic quantities. Components beginning with 'pr_' use the logit-probability scale and those beginning with 'haz_' use the log-hazard scale.

mu_star Estimated mean from bootstrapped samples

pr_star Logit event probabilities from bootstrap samples

haz_star Log-hazards from bootstrap samples

mu_var_hat Variance of estimated mean

pr_var_hat Variance of logit event probabilities

haz_var_hat Variance of log-hazards

mu_var_double Variance of estimated mean of bootstrapped samples (via double-bootstrapping)

pr_var_double Variance of estimated probability of bootstrapped samples (via double-bootstrapping)

haz_var_double Variance of estimated hazard rates of bootstrapped samples (via double-bootstrapping)

mu_Tstar Pivot quantity of the estimated mean

pr_Tstar Pivot quantity of the estimated probability

haz_Tstar Pivot quantity of the estimated hazard rates

Examples

```
## Not run:
# set some parameters
n <- 30 # sample size
t <- seq(100, 200, by = 10) # time intervals
B <- 100 # number of bootstraps
BB <- 100 # number of double-bootstraps
m <- 10 # repeated random-start optimization setting
par_hat <- c(
  3.41361e-03, 2.76268e+00, 2.60370e+00, 3.30802e+02, 5.48822e+00, 2.92945e+02, NA,
  9.43071e-03, 2.47598e+02, 1.80102e+00, 6.50845e-01, 7.18247e-01
```

```

)
mu_hat <- c(292.94512, 292.94513, 319.72017, 294.16945, 298.87286, 292.94512)
pr_hat <- c(0.60039, 0.42155, 0.53434, 0.30780, 0.56416, 0.61795)
haz_hat <- matrix(c(
  -5.67999, -5.67999, -5.67999, -5.67999, -5.67999, -5.67999,
  -5.67999, -5.67999, -5.67999, -5.67999, -5.67999, -6.09420,
  -5.99679, -5.91174, -5.83682, -5.77031, -5.71085, -5.65738,
  -5.60904, -5.56512, -5.52504, -5.48833, -6.09902, -5.97017,
  -5.85769, -5.75939, -5.67350, -5.59856, -5.53336, -5.47683,
  -5.42805, -5.38621, -5.35060, -6.17146, -6.09512, -6.02542,
  -5.96131, -5.90194, -5.84668, -5.79498, -5.74642, -5.70064,
  -5.65733, -5.61624, -5.92355, -5.80239, -5.70475, -5.62524,
  -5.55994, -5.50595, -5.46106, -5.42359, -5.39222, -5.36591,
  -5.34383, -5.79111, -5.67660, -5.58924, -5.52166, -5.46879,
  -5.42707, -5.39394, -5.36751, -5.34637, -5.32946, -5.31596
),length(t),6)
y <- 304 # cut-off year for estimating probablity

# generate bootstrapped samples then fit renewal model
res <- marp::gamma_bstrp(n, t, B, BB, m, par_hat, mu_hat, pr_hat, haz_hat, y)

## End(Not run)

```

gamma_logl

A function to calculate the log-likelihood of Gamma model

Description

A function to calculate the log-likelihood of Gamma model

Usage

```
gamma_logl(param, x)
```

Arguments

param	Length-2 numeric vector containing log(shape) and log(rate).
x	Numeric vector of positive observations.

Value

returns the value of negative log-likelihood of the Gamma model

Examples

```

set.seed(42)
data <- rgamma(30,3,0.01)

# set some parameters
par_hat <- c(2.7626793657057762, 0.0094307059277139432) # estimated parameters
param <- log(par_hat) # input parameters for logl function

# calculate log-likelihood
result <- marp::gamma_logl(param, data)

result

```

gamma_rp

*A function to fit Gamma renewal model***Description**

A function to fit Gamma renewal model

Usage

```
gamma_rp(data, t, m, y)
```

Arguments

data	A numeric vector of positive inter-event times.
t	A numeric vector of time points at which log-hazards are evaluated.
m	A positive integer controlling repeated random-start optimizations; the current implementation seeks ‘m - 1’ acceptable ‘nlm()’ fits.
y	A time point at which the logit-transformed cumulative event probability is evaluated.

Value

An object of class ‘marp_model_fit’ retaining the following eight named list components:

par1 Estimated shape parameter of the Gamma model

par2 Estimated rate parameter of the Gamma model

logL Maximized log-likelihood

AIC Akaike information criterion (AIC)

BIC Bayesian information criterion (BIC)

mu_hat Estimated mean inter-event time

pr_hat Logit-transformed cumulative event probability at ‘y’

haz_hat Log-hazard values at ‘t’

Examples

```

set.seed(42)
data <- rgamma(100,3,0.01)

# set some parameters
m = 10 # repeated random-start optimization setting
t = seq(100, 200, by=10) # time intervals
y = 304 # cut-off year for estimating probablity

# fit Gamma renewal model
fit <- marp::gamma_rp(data, t, m, y)
fit
summary(fit)

```

`logLik.marp_model_fit` *Extract the log-likelihood from a model-specific fit*

Description

Extract the log-likelihood from a model-specific fit

Usage

```

## S3 method for class 'marp_model_fit'
logLik(object, ...)

```

Arguments

`object` A ‘marp_model_fit’ object.
`...` Additional arguments, currently unused.

Value

An object of class ‘logLik’.

`loglogis_bstrp` *A function to generate (double) bootstrap samples and fit Log-Logistic renewal model*

Description

A function to generate (double) bootstrap samples and fit Log-Logistic renewal model

Usage

```

loglogis_bstrp(n, t, B, BB, m, par_hat, mu_hat, pr_hat, haz_hat, y)

```

Arguments

<code>n</code>	Number of inter-event times generated in each bootstrap sample.
<code>t</code>	Time points at which log-hazards are evaluated.
<code>B</code>	Number of bootstrap samples.
<code>BB</code>	Number of double-bootstrap samples per bootstrap sample.
<code>m</code>	Positive integer controlling repeated random-start optimizations; see <code>[marp()]</code> .
<code>par_hat</code>	Length-12 vector containing 'par1' for models 1–6 followed by 'par2' for models 1–6; see <code>[marp()]</code> for model order.
<code>mu_hat</code>	Length-6 vector of model-specific mean estimates.
<code>pr_hat</code>	Length-6 vector of model-specific logit event probabilities.
<code>haz_hat</code>	Matrix of model-specific log-hazards with 'length(t)' rows and six model columns.
<code>y</code>	Time point at which logit event probabilities are evaluated.

Value

A list of bootstrap estimates and variance/T-statistic quantities. Components beginning with 'pr_' use the logit-probability scale and those beginning with 'haz_' use the log-hazard scale.

mu_star Estimated mean from bootstrapped samples

pr_star Logit event probabilities from bootstrap samples

haz_star Log-hazards from bootstrap samples

mu_var_hat Variance of estimated mean

pr_var_hat Variance of estimated probability

haz_var_hat Variance of estimated hazard rates

mu_var_double Variance of estimated mean of bootstrapped samples (via double-bootstrapping)

pr_var_double Variance of estimated probability of bootstrapped samples (via double-bootstrapping)

haz_var_double Variance of estimated hazard rates of bootstrapped samples (via double-bootstrapping)

mu_Tstar Pivot quantity of the estimated mean

pr_Tstar Pivot quantity of the estimated probability

haz_Tstar Pivot quantity of the estimated hazard rates

Examples

```
## Not run:
# set some parameters
n <- 30 # sample size
t <- seq(100, 200, by = 10) # time intervals
B <- 100 # number of bootstraps
BB <- 100 # number of double-bootstraps
m <- 10 # repeated random-start optimization setting
par_hat <- c(
  3.41361e-03, 2.76268e+00, 2.60370e+00, 3.30802e+02, 5.48822e+00, 2.92945e+02, NA,
  9.43071e-03, 2.47598e+02, 1.80102e+00, 6.50845e-01, 7.18247e-01
```

```

)
mu_hat <- c(292.94512, 292.94513, 319.72017, 294.16945, 298.87286, 292.94512)
pr_hat <- c(0.60039, 0.42155, 0.53434, 0.30780, 0.56416, 0.61795)
haz_hat <- matrix(c(
  -5.67999, -5.67999, -5.67999, -5.67999, -5.67999, -5.67999,
  -5.67999, -5.67999, -5.67999, -5.67999, -5.67999, -6.09420,
  -5.99679, -5.91174, -5.83682, -5.77031, -5.71085, -5.65738,
  -5.60904, -5.56512, -5.52504, -5.48833, -6.09902, -5.97017,
  -5.85769, -5.75939, -5.67350, -5.59856, -5.53336, -5.47683,
  -5.42805, -5.38621, -5.35060, -6.17146, -6.09512, -6.02542,
  -5.96131, -5.90194, -5.84668, -5.79498, -5.74642, -5.70064,
  -5.65733, -5.61624, -5.92355, -5.80239, -5.70475, -5.62524,
  -5.55994, -5.50595, -5.46106, -5.42359, -5.39222, -5.36591,
  -5.34383, -5.79111, -5.67660, -5.58924, -5.52166, -5.46879,
  -5.42707, -5.39394, -5.36751, -5.34637, -5.32946, -5.31596
),length(t),6)
y <- 304 # cut-off year for estimating probablity

# generate bootstrapped samples then fit renewal model
res <- marp::loglogis_bstrp(n, t, B, BB, m, par_hat, mu_hat, pr_hat, haz_hat, y)

## End(Not run)

```

loglogis_logl

A function to calculate the log-likelihood of Log-Logistics model

Description

A function to calculate the log-likelihood of Log-Logistics model

Usage

```
loglogis_logl(param, x)
```

Arguments

param	Length-2 numeric vector containing log(scale) and log(shape).
x	Numeric vector of positive observations.

Value

returns the value of negative log-likelihood of the Log-Logistics model

Examples

```

set.seed(42)
data <- rgamma(30,3,0.01)

# set some parameters
par_hat <- c(2.6037079185931518, 247.59811806509711) # estimated parameters
param <- c(log(par_hat[2]),log(par_hat[1])) # input parameters for logl function

# calculate log-likelihood
result <- marp::loglogis_logl(param, data)

result

```

loglogis_rp	<i>A function to fit Log-Logistics renewal model</i>
-------------	--

Description

A function to fit Log-Logistics renewal model

Usage

```
loglogis_rp(data, t, m, y)
```

Arguments

data	A numeric vector of positive inter-event times.
t	A numeric vector of time points at which log-hazards are evaluated.
m	A positive integer controlling repeated random-start optimizations; the current implementation seeks ‘m - 1’ acceptable ‘nlm()’ fits.
y	A time point at which the logit-transformed cumulative event probability is evaluated.

Value

An object of class ‘marp_model_fit’ retaining the following eight named list components:

par1 Estimated shape parameter of the Log-Logistics model

par2 Estimated scale parameter of the Log-Logistics model

logL Maximized log-likelihood

AIC Akaike information criterion (AIC)

BIC Bayesian information criterion (BIC)

mu_hat Estimated mean inter-event time

pr_hat Logit-transformed cumulative event probability at ‘y’

haz_hat Log-hazard values at ‘t’

Examples

```

set.seed(42)
data <- rgamma(100,3,0.01)

# set some parameters
m = 10 # repeated random-start optimization setting
t = seq(100, 200, by=10) # time intervals
y = 304 # cut-off year for estimating probablity

# fit Log-Logistic renewal model
fit <- marp::loglogis_rp(data, t, m, y)
fit
summary(fit)

```

lognorm_bstrp	<i>A function to generate (double) bootstrap samples and fit Log-Normal renewal model</i>
---------------	---

Description

A function to generate (double) bootstrap samples and fit Log-Normal renewal model

Usage

```
lognorm_bstrp(n, t, B, BB, par_hat, mu_hat, pr_hat, haz_hat, y)
```

Arguments

n	Number of inter-event times generated in each bootstrap sample.
t	Time points at which log-hazards are evaluated.
B	Number of bootstrap samples.
BB	Number of double-bootstrap samples per bootstrap sample.
par_hat	Length-12 vector containing ‘par1’ for models 1–6 followed by ‘par2’ for models 1–6; see [marp()] for model order.
mu_hat	Length-6 vector of model-specific mean estimates.
pr_hat	Length-6 vector of model-specific logit event probabilities.
haz_hat	Matrix of model-specific log-hazards, with ‘length(t)’ rows and six model columns.
y	Time point at which logit event probabilities are evaluated.

Value

A list of bootstrap estimates and variance/T-statistic quantities. Components beginning with 'pr_' use the logit-probability scale and those beginning with 'haz_' use the log-hazard scale.

mu_star Estimated mean from bootstrapped samples

pr_star Logit event probabilities from bootstrap samples

haz_star Log-hazards from bootstrap samples

mu_var_hat Variance of estimated mean

pr_var_hat Variance of logit event probabilities

haz_var_hat Variance of log-hazards

mu_var_double Variance of estimated mean of bootstrapped samples (via double-bootstrapping)

pr_var_double Variance of estimated probability of bootstrapped samples (via double-bootstrapping)

haz_var_double Variance of estimated hazard rates of bootstrapped samples (via double-bootstrapping)

mu_Tstar Pivot quantity of the estimated mean

pr_Tstar Pivot quantity of the estimated probability

haz_Tstar Pivot quantity of the estimated hazard rates

Examples

```
## Not run:
# set some parameters
n <- 30 # sample size
t <- seq(100, 200, by = 10) # time intervals
B <- 100 # number of bootstraps
BB <- 100 # number of double-bootstraps
# m <- 10 # number of iterations for MLE optimization
par_hat <- c(
  3.41361e-03, 2.76268e+00, 2.60370e+00, 3.30802e+02, 5.48822e+00, 2.92945e+02, NA,
  9.43071e-03, 2.47598e+02, 1.80102e+00, 6.50845e-01, 7.18247e-01
)
mu_hat <- c(292.94512, 292.94513, 319.72017, 294.16945, 298.87286, 292.94512)
pr_hat <- c(0.60039, 0.42155, 0.53434, 0.30780, 0.56416, 0.61795)
haz_hat <- matrix(c(
  -5.67999, -5.67999, -5.67999, -5.67999, -5.67999, -5.67999,
  -5.67999, -5.67999, -5.67999, -5.67999, -5.67999, -6.09420,
  -5.99679, -5.91174, -5.83682, -5.77031, -5.71085, -5.65738,
  -5.60904, -5.56512, -5.52504, -5.48833, -6.09902, -5.97017,
  -5.85769, -5.75939, -5.67350, -5.59856, -5.53336, -5.47683,
  -5.42805, -5.38621, -5.35060, -6.17146, -6.09512, -6.02542,
  -5.96131, -5.90194, -5.84668, -5.79498, -5.74642, -5.70064,
  -5.65733, -5.61624, -5.92355, -5.80239, -5.70475, -5.62524,
  -5.55994, -5.50595, -5.46106, -5.42359, -5.39222, -5.36591,
  -5.34383, -5.79111, -5.67660, -5.58924, -5.52166, -5.46879,
  -5.42707, -5.39394, -5.36751, -5.34637, -5.32946, -5.31596
),length(t),6)
y <- 304 # cut-off year for estimating probability
```

```
# generate bootstrapped samples then fit renewal model
res <- marp::lognorm_bstrp(n, t, B, BB, par_hat, mu_hat, pr_hat, haz_hat, y)

## End(Not run)
```

lognorm_rp

A function to fit Log-Normal renewal model

Description

A function to fit Log-Normal renewal model

Usage

```
lognorm_rp(data, t, y)
```

Arguments

data	A numeric vector of positive inter-event times.
t	A numeric vector of time points at which log-hazards are evaluated.
y	A time point at which the logit-transformed cumulative event probability is evaluated.

Value

An object of class ‘marp_model_fit’ retaining the following eight named list components:

par1 Estimated mean (on the log scale) of the Log-Normal model

par2 Estimated standard deviation (on the log scale) of the Log-Normal model

logL Maximized log-likelihood

AIC Akaike information criterion (AIC)

BIC Bayesian information criterion (BIC)

mu_hat Estimated mean inter-event time

pr_hat Logit-transformed cumulative event probability at ‘y’

haz_hat Log-hazard values at ‘t’

Examples

```
set.seed(42)
data <- rgamma(100, 3, 0.01)

# set some parameters
t = seq(100, 200, by=10) # time intervals
y = 304 # cut-off year for estimating probability
```

```
# fit Log-Normal renewal model
fit <- marp::lognorm_rp(data, t, y)
fit
summary(fit)
```

lowerT

Evaluate the lower-limit equation for a model-averaged T statistic

Description

Evaluate the lower-limit equation for a model-averaged T statistic

Usage

```
lowerT(low, hat, sigmasq, Tstar, weights, B, alpha)
```

Arguments

low	Candidate lower confidence limit.
hat	Vector of model-specific estimates.
sigmasq	Vector of model-specific variance estimates.
Tstar	Matrix of bootstrap T statistics, with models in rows.
weights	Vector of model weights.
B	Number of bootstrap samples represented in ‘Tstar’.
alpha	Significance level.

Value

A numeric scalar giving the lower-limit root equation value.

Examples

```
# set some parameters
low <- 100 # lower bound
hat <- rep(150, 6) # estimates obtained from each model
sigmasq <- 10 # variance
Tstar <- matrix(rep(100,600),6,100) # T statistics estimated from bootstrap samples
weights <- rep(1/6, 6) # model weights
B <- 100 # number of bootstrapped samples
alpha <- 0.05 # confidence level

# calculate the upper limit of T statistics
res <- marp::lowerT(low, hat, sigmasq, Tstar, weights, B, alpha)

res
```

marp

*A function to apply model-averaged renewal process***Description**

A function to apply model-averaged renewal process

Usage

```
marp(data, t, m, y, which.model = 1)
```

Arguments

<code>data</code>	A numeric vector of positive inter-event times.
<code>t</code>	A numeric vector of time points at which log-hazards are evaluated.
<code>m</code>	A positive integer controlling repeated random-start optimizations for the Gamma, log-logistic, Weibull, and BPT candidate models. The current implementation seeks ‘m - 1’ acceptable ‘nlm()’ fits for each such model.
<code>y</code>	A time point at which the logit-transformed cumulative event probability is evaluated.
<code>which.model</code>	Integer identifying a reference or generating model: 1 = Poisson, 2 = Gamma, 3 = log-logistic, 4 = Weibull, 5 = log-normal, and 6 = Brownian passage time (BPT). This is mainly useful when the generating model is known, such as in simulations.

Value

An object of class ‘marp_fit’: a 19-component named list of model-specific, selected-model, reference-model, and AIC-weighted estimates.

par1 First fitted parameter for each model, in the model order listed under ‘which.model’. Its meaning is model-specific.

par2 Second fitted parameter for each model in the same order; ‘NA’ for the one-parameter Poisson model. Its meaning is model-specific.

logL Maximized log-likelihood for each model.

AIC Akaike information criterion (AIC)

BIC Bayesian information criterion (BIC)

mu_hat Estimated mean inter-event time for each model.

pr_hat Logit-transformed cumulative event probability at ‘y’ for each model.

haz_hat Log-hazard values at ‘t’ for each model.

weights_AIC Model weights calculated based on AIC

model_best Model selected based on the lowest AIC

mu_best Estimated mean obtained from the model with the lowest AIC

pr_best Estimated logit event probability from the model with the lowest AIC.
haz_best Estimated log-hazards from the model with the lowest AIC.
mu_gen Estimated mean obtained from the (true or hypothetical) generating model
pr_gen Estimated logit event probability from the reference model.
haz_gen Estimated log-hazards from the reference model.
mu_aic Estimated mean obtained from model-averaging (using AIC weights)
pr_aic AIC-weighted average of the model-specific logit event probabilities.
haz_aic AIC-weighted averages of the model-specific log-hazards at 't'.

Examples

```
set.seed(42)
data <- rgamma(100,3,0.01)

# set some parameters
m = 10 # repeated random-start optimization setting
t = seq(100, 200, by=10) # time intervals
y = 304 # cut-off year for estimating probability
which.model <- 2 # specify the generating model

# model selection and averaging
result <- marp::marp(data, t, m, y, which.model)
result
summary(result)

# Bootstrap confidence intervals delegate to the existing marp_confint()
# engine. The original data and bootstrap sizes are supplied explicitly.
## Not run:
confint(result, data = data, B = 99, BB = 99)

## End(Not run)
```

marp_bstrp

A function to fit model-averaged renewal process

Description

A function to fit model-averaged renewal process

Usage

```
marp_bstrp(n, t, B, BB, m, par_hat, mu_hat, pr_hat, haz_hat, y)
```

Arguments

<code>n</code>	Number of inter-event times generated in each bootstrap sample.
<code>t</code>	Time points at which log-hazards are evaluated.
<code>B</code>	Number of bootstrap samples.
<code>BB</code>	Number of double-bootstrap samples per bootstrap sample.
<code>m</code>	Positive integer controlling repeated random-start optimizations; see <code>[marp()]</code> .
<code>par_hat</code>	Length-12 vector containing 'par1' for models 1–6 followed by 'par2' for models 1–6; see <code>[marp()]</code> for model order.
<code>mu_hat</code>	Length-6 vector of model-specific mean estimates.
<code>pr_hat</code>	Length-6 vector of model-specific logit event probabilities.
<code>haz_hat</code>	Matrix of model-specific log-hazards with 'length(t)' rows and six model columns.
<code>y</code>	Time point at which logit event probabilities are evaluated.

Value

A list combining bootstrap results from all six candidate models. Components beginning with 'pr_' use the logit-probability scale and those beginning with 'haz_' use the log-hazard scale.

mu_star Estimated mean from bootstrapped samples

pr_star A 6 by 'B' matrix of logit event probabilities

haz_star A 'length(t)' by 6 by 'B' array of log-hazards

mu_var_hat Variance of estimated mean

pr_var_hat Variance of model-specific logit event probabilities

haz_var_hat Variance of model-specific log-hazards

mu_var_double Variance of estimated mean of bootstrapped samples (via double-bootstrapping)

pr_var_double Variance of estimated probability of bootstrapped samples (via double-bootstrapping)

haz_var_double Variance of estimated hazard rates of bootstrapped samples (via double-bootstrapping)

mu_Tstar Pivot quantity of the estimated mean

pr_Tstar Pivot quantity of the estimated probability

haz_Tstar Pivot quantity of the estimated hazard rates

Examples

```
## Not run:
# set some parameters
n <- 30 # sample size
t <- seq(100, 200, by = 10) # time intervals
B <- 100 # number of bootstraps
BB <- 100 # number of double-bootstraps
m <- 10 # repeated random-start optimization setting
par_hat <- c(
  3.41361e-03, 2.76268e+00, 2.60370e+00, 3.30802e+02, 5.48822e+00, 2.92945e+02, NA,
  9.43071e-03, 2.47598e+02, 1.80102e+00, 6.50845e-01, 7.18247e-01
```

```

)
mu_hat <- c(292.94512, 292.94513, 319.72017, 294.16945, 298.87286, 292.94512)
pr_hat <- c(0.60039, 0.42155, 0.53434, 0.30780, 0.56416, 0.61795)
haz_hat <- matrix(c(
  -5.67999, -5.67999, -5.67999, -5.67999, -5.67999, -5.67999,
  -5.67999, -5.67999, -5.67999, -5.67999, -5.67999, -6.09420,
  -5.99679, -5.91174, -5.83682, -5.77031, -5.71085, -5.65738,
  -5.60904, -5.56512, -5.52504, -5.48833, -6.09902, -5.97017,
  -5.85769, -5.75939, -5.67350, -5.59856, -5.53336, -5.47683,
  -5.42805, -5.38621, -5.35060, -6.17146, -6.09512, -6.02542,
  -5.96131, -5.90194, -5.84668, -5.79498, -5.74642, -5.70064,
  -5.65733, -5.61624, -5.92355, -5.80239, -5.70475, -5.62524,
  -5.55994, -5.50595, -5.46106, -5.42359, -5.39222, -5.36591,
  -5.34383, -5.79111, -5.67660, -5.58924, -5.52166, -5.46879,
  -5.42707, -5.39394, -5.36751, -5.34637, -5.32946, -5.31596
),length(t),6)
y <- 304 # cut-off year for estimating probablity

# generate bootstrapped samples then fit renewal model
res <- marp::marp_bstrp(n, t, B, BB, m, par_hat, mu_hat, pr_hat, haz_hat, y)

## End(Not run)

```

marp_confint

*A function to apply model-averaged renewal process***Description**

A function to apply model-averaged renewal process

Usage

```
marp_confint(data, m, t, B, BB, alpha, y, which.model)
```

Arguments

data	A numeric vector of positive inter-event times.
m	A positive integer controlling repeated random-start optimizations; see [marp()] for the exact role of this argument.
t	A numeric vector of time points at which log-hazards are evaluated.
B	Number of bootstrap samples.
BB	Number of double-bootstrap samples.
alpha	Significance level; the confidence level is '1 - alpha'.
y	A time point at which the logit-transformed cumulative event probability is evaluated.
which.model	Integer identifying a reference or generating model: 1 = Poisson, 2 = Gamma, 3 = log-logistic, 4 = Weibull, 5 = log-normal, and 6 = BPT.

Value

The returned object has three top-level components: ‘out’, containing the fitted/model-averaged estimates and bootstrap-weighted estimates; ‘percent_CI’, containing percentile intervals; and ‘student_CI’, containing studentized intervals. The object has class ‘marp_confint’; its three existing list components remain directly accessible with ‘\$’. Probability quantities are stored on the logit scale and hazard quantities on the log-hazard scale throughout.

out The 19 fitted/model-averaged components returned by [marp()], followed by ‘mu_bstrp’, ‘pr_bstrp’, and ‘haz_bstrp’.

percent_CI Bootstrap model weights and percentile summaries for reference-model and selected-model means, logit probabilities, and log-hazards.

student_CI Studentized lower and upper limits for reference-model, selected-model, and model-averaged quantities.

par1 First fitted parameter for each model; its meaning is model-specific.

par2 Second fitted parameter for each model; its meaning is model-specific.

logL Maximized log-likelihood

AIC Akaike information criterion (AIC)

BIC Bayesian information criterion (BIC)

mu_hat Estimated mean inter-event times

pr_hat Logit-transformed cumulative event probabilities at ‘y’

haz_hat Log-hazard values at ‘t’

weights_AIC Model weights calculated based on AIC

model_best Model selected based on the lowest AIC

mu_best Estimated mean obtained from the model with the lowest AIC

pr_best Estimated logit event probability from the model with the lowest AIC

haz_best Estimated log-hazards from the model with the lowest AIC

mu_gen Estimated mean obtained from the (true or hypothetical) generating model

pr_gen Estimated logit event probability from the reference model

haz_gen Estimated log-hazards from the reference model

mu_aic Estimated mean obtained from model-averaging (using AIC weights)

pr_aic AIC-weighted average of logit event probabilities

haz_aic AIC-weighted averages of log-hazards

mu_bstrp Estimated mean obtained from model-averaging (using bootstrapped weights)

pr_bstrp Bootstrap-weighted average of logit event probabilities

haz_bstrp Bootstrap-weighted averages of log-hazards

weights_bstp Model weights calculated by bootstrapping, that is, the frequency of each model being selected as the best model is divided by the total number of bootstraps

mu_gen Median of the percentile bootstrap confidence interval of the estimated mean based on the generating model

mu_gen_lower Lower limit of the percentile bootstrap confidence interval of the estimated mean based on the generating model

mu_gen_upper Upper limit of the percentile bootstrap confidence interval of the estimated mean based on the generating model

mu_best Median of the percentile bootstrap confidence interval of the estimated mean based on the best model

mu_best_lower Lower limit of the percentile bootstrap confidence interval of the estimated mean based on the best model

mu_best_upper Upper limit of the percentile bootstrap confidence interval of the estimated mean based on the best model

pr_gen Median reference-model logit event probability

pr_gen_lower Lower limit for the reference-model logit event probability

pr_gen_upper Upper limit for the reference-model logit event probability

pr_best Median best-model logit event probability

pr_best_lower Lower limit for the best-model logit event probability

pr_best_upper Upper limit for the best-model logit event probability

haz_gen Median reference-model log-hazards

haz_gen_lower Lower limits for reference-model log-hazards

haz_gen_upper Upper limits for reference-model log-hazards

haz_best Median best-model log-hazards

haz_best_lower Lower limits for best-model log-hazards

haz_best_upper Upper limits for best-model log-hazards

mu_lower_gen Lower limit of the studentized bootstrap confidence interval of the estimated mean based on the generating model

mu_upper_gen Upper limit of the studentized bootstrap confidence interval of the estimated mean based on the generating model

mu_lower_best Lower limit of the studentized bootstrap confidence interval of the estimated mean based on the best model

mu_upper_best Upper limit of the studentized bootstrap confidence interval of the estimated mean based on the best model

pr_lower_gen Lower studentized limit for the reference-model logit probability

pr_upper_gen Upper studentized limit for the reference-model logit probability

pr_lower_best Lower studentized limit for the best-model logit probability

pr_upper_best Upper studentized limit for the best-model logit probability

haz_lower_gen Lower studentized limits for reference-model log-hazards

haz_upper_gen Upper studentized limits for reference-model log-hazards

haz_lower_best Lower studentized limits for best-model log-hazards

haz_upper_best Upper studentized limits for best-model log-hazards

mu_lower_ma Lower limit of model-averaged studentized bootstrap confidence interval of the estimated mean

mu_upper_ma Upper limit of model-averaged studentized bootstrap confidence interval of the estimated mean

pr_lower_ma Lower model-averaged limit on the logit-probability scale

pr_upper_ma Upper model-averaged limit on the logit-probability scale

haz_lower_ma Lower model-averaged limits on the log-hazard scale

haz_upper_ma Upper model-averaged limits on the log-hazard scale

Examples

```
## Not run:
# generate random data
set.seed(42)
data <- rgamma(30, 3, 0.01)

# set some parameters
m <- 10 # repeated random-start optimization setting
t <- seq(100,200,by=10) # time intervals
alpha <- 0.05 # confidence level
y <- 304 # cut-off year for estimating probability
B <- 100 # number of bootstraps
BB <- 100 # number of double bootstraps
which.model <- 2 # specify the generating model

# Construct confidence intervals. Production choices of B and BB may be
# computationally expensive because BB double bootstraps are generated for
# each of B bootstrap samples.
res <- marp::marp_confint(data,m,t,B,BB,alpha,y,which.model)
res

## End(Not run)
```

nobs.marp_fit	<i>Number of observations in a model-averaged fit</i>
---------------	---

Description

Number of observations in a model-averaged fit

Usage

```
## S3 method for class 'marp_fit'
nobs(object, ...)
```

Arguments

object	A ‘marp_fit’ object.
...	Additional arguments, currently unused.

Value

The number of observations used for fitting.

nobs.marp_model_fit	<i>Number of observations in a model-specific fit</i>
---------------------	---

Description

Number of observations in a model-specific fit

Usage

```
## S3 method for class 'marp_model_fit'
nobs(object, ...)
```

Arguments

object	A 'marp_model_fit' object.
...	Additional arguments, currently unused.

Value

The number of observations used for fitting.

percent_confint	<i>A function to calculate percentile bootstrap confidence interval</i>
-----------------	---

Description

A function to calculate percentile bootstrap confidence interval

Usage

```
percent_confint(data, B, t, m, y, which.model = 1)
```

Arguments

data	A numeric vector of positive inter-event times.
B	Number of nonparametric bootstrap samples.
t	Time points at which log-hazards are evaluated.
m	A positive integer controlling repeated random-start optimizations; see [marp()].
y	Time point at which logit event probabilities are evaluated.
which.model	Integer identifying the reference model using the mapping 1 = Poisson, 2 = Gamma, 3 = log-logistic, 4 = Weibull, 5 = log-normal, and 6 = BPT.

Value

A list of 95 percent percentile-bootstrap summaries. Components beginning with ‘pr_’ are on the logit-probability scale and components beginning with ‘haz_’ are on the log-hazard scale.

weights_bstp Model weights calculated by bootstrapping, that is, the frequency of each model being selected as the best model is divided by the total number of bootstraps

mu_gen Median of the percentile bootstrap confidence interval of the estimated mean based on the generating model

mu_gen_lower Lower limit of the percentile bootstrap confidence interval of the estimated mean based on the generating model

mu_gen_upper Upper limit of the percentile bootstrap confidence interval of the estimated mean based on the generating model

mu_best Median of the percentile bootstrap confidence interval of the estimated mean based on the best model

mu_best_lower Lower limit of the percentile bootstrap confidence interval of the estimated mean based on the best model

mu_best_upper Upper limit of the percentile bootstrap confidence interval of the estimated mean based on the best model

pr_gen Median logit event probability based on the reference model

pr_gen_lower Lower limit for the reference-model logit event probability

pr_gen_upper Upper limit for the reference-model logit event probability

pr_best Median logit event probability based on the best model

pr_best_lower Lower limit for the best-model logit event probability

pr_best_upper Upper limit for the best-model logit event probability

haz_gen Median log-hazards based on the reference model

haz_gen_lower Lower limits for reference-model log-hazards

haz_gen_upper Upper limits for reference-model log-hazards

haz_best Median log-hazards based on the best model

haz_best_lower Lower limits for best-model log-hazards

haz_best_upper Upper limits for best-model log-hazards

Examples

```
## Not run:
# generate random data
set.seed(42)
data <- rgamma(30, 3, 0.01)

# set some parameters
m <- 10 # repeated random-start optimization setting
t <- seq(100,200,by=10) # time intervals
y <- 304 # time point for estimating event probability
B <- 100 # number of bootstraps
BB <- 100 # number of double bootstraps
```

```

which.model <- 2 # specify the generating model

# construct percentile bootstrap confidence intervals
marp::percent_confint(data, B, t, m, y, which.model)

## End(Not run)

```

pllog

Probability function of Log-Logistics model

Description

Probability function of Log-Logistics model

Usage

```
pllog(q, shape = 1, scale = 1, lower.tail = TRUE, log.p = FALSE)
```

Arguments

q	Numeric vector of positive quantiles.
shape	Positive shape parameter of the log-logistic distribution.
scale	Positive scale parameter of the log-logistic distribution.
lower.tail	Logical; if 'TRUE', return lower-tail probabilities, otherwise return upper-tail probabilities.
log.p	Logical; if 'TRUE', return probabilities on the log scale.

Value

A numeric vector of probabilities, or log-probabilities when 'log.p = TRUE'.

Examples

```

q <- c(1, 2, 3, 4)
# set paramters
shape <- 5
scale <- 3
result_1 <- marp::pllog(q, shape, scale)

# alternatively, return log-probabilities
result_2 <- marp::pllog(q, shape, scale, log.p = TRUE)

```

poisson_bstrp	<i>A function to generate (double) bootstrap samples and fit Poisson renewal model</i>
---------------	--

Description

A function to generate (double) bootstrap samples and fit Poisson renewal model

Usage

```
poisson_bstrp(n, t, B, BB, par_hat, mu_hat, pr_hat, haz_hat, y)
```

Arguments

n	Number of inter-event times generated in each bootstrap sample.
t	Time points at which log-hazards are evaluated.
B	Number of bootstrap samples.
BB	Number of double-bootstrap samples per bootstrap sample.
par_hat	Length-12 vector containing ‘par1’ for models 1–6 followed by ‘par2’ for models 1–6; see [marp()] for model order.
mu_hat	Length-6 vector of model-specific mean estimates.
pr_hat	Length-6 vector of model-specific logit event probabilities.
haz_hat	Matrix of model-specific log-hazards, with ‘length(t)’ rows and six columns in the model order documented by [marp()].
y	Time point at which logit event probabilities are evaluated.

Value

A list of bootstrap estimates and variance/T-statistic quantities. Components beginning with ‘pr_’ use the logit-probability scale and those beginning with ‘haz_’ use the log-hazard scale.

mu_star Estimated mean from bootstrapped samples

pr_star Logit event probabilities from bootstrap samples

haz_star Log-hazards from bootstrap samples

mu_var_hat Variance of estimated mean

pr_var_hat Variance of logit event probabilities

haz_var_hat Variance of log-hazards

mu_var_double Variance of estimated mean of bootstrapped samples (via double-bootstrapping)

pr_var_double Variance of estimated probability of bootstrapped samples (via double-bootstrapping)

haz_var_double Variance of estimated hazard rates of bootstrapped samples (via double-bootstrapping)

mu_Tstar Pivot quantity of the estimated mean

pr_Tstar Pivot quantity of the estimated probability

haz_Tstar Pivot quantity of the estimated hazard rates

Examples

```
## Not run:
# set some parameters
n <- 30 # sample size
t <- seq(100, 200, by = 10) # time intervals
B <- 100 # number of bootstraps
BB <- 100 # number of double-bootstraps
# m <- 10 # number of iterations for MLE optimization
par_hat <- c(
  3.41361e-03, 2.76268e+00, 2.60370e+00, 3.30802e+02, 5.48822e+00, 2.92945e+02, NA,
  9.43071e-03, 2.47598e+02, 1.80102e+00, 6.50845e-01, 7.18247e-01
)
mu_hat <- c(292.94512, 292.94513, 319.72017, 294.16945, 298.87286, 292.94512)
pr_hat <- c(0.60039, 0.42155, 0.53434, 0.30780, 0.56416, 0.61795)
haz_hat <- matrix(c(
  -5.67999, -5.67999, -5.67999, -5.67999, -5.67999, -5.67999,
  -5.67999, -5.67999, -5.67999, -5.67999, -5.67999, -6.09420,
  -5.99679, -5.91174, -5.83682, -5.77031, -5.71085, -5.65738,
  -5.60904, -5.56512, -5.52504, -5.48833, -6.09902, -5.97017,
  -5.85769, -5.75939, -5.67350, -5.59856, -5.53336, -5.47683,
  -5.42805, -5.38621, -5.35060, -6.17146, -6.09512, -6.02542,
  -5.96131, -5.90194, -5.84668, -5.79498, -5.74642, -5.70064,
  -5.65733, -5.61624, -5.92355, -5.80239, -5.70475, -5.62524,
  -5.55994, -5.50595, -5.46106, -5.42359, -5.39222, -5.36591,
  -5.34383, -5.79111, -5.67660, -5.58924, -5.52166, -5.46879,
  -5.42707, -5.39394, -5.36751, -5.34637, -5.32946, -5.31596
),length(t),6)
y <- 304 # cut-off year for estimating probablity

# generate bootstrapped samples then fit renewal model
res <- marp::poisson_bstrp(n, t, B, BB, par_hat, mu_hat, pr_hat, haz_hat, y)

## End(Not run)
```

poisson_rp

*A function to fit Poisson renewal model***Description**

A function to fit Poisson renewal model

Usage

```
poisson_rp(data, t, y)
```

Arguments

data A numeric vector of positive inter-event times.

t	A numeric vector of time points at which log-hazards are evaluated.
y	A time point at which the logit-transformed cumulative event probability is evaluated.

Value

An object of class ‘marp_model_fit’ retaining the following eight named list components:

par1 Estimated exponential rate parameter of the Poisson renewal model

par2 N/A, only keep it as a place holder for output formatting purpose

logL Maximized log-likelihood

AIC Akaike information criterion (AIC)

BIC Bayesian information criterion (BIC)

mu_hat Estimated mean inter-event time

pr_hat Logit-transformed cumulative event probability at ‘y’

haz_hat Log-hazard values at ‘t’

Examples

```
set.seed(42)
data <- rgamma(100,3,0.01)

# set some parameters
t = seq(100, 200, by=10) # time intervals
y = 304 # cut-off year for estimating probability

# fit Poisson renewal model
fit <- marp::poisson_rp(data, t, y)
fit
summary(fit)
```

```
print.marp_confint      Print a model-averaged confidence-interval result
```

Description

Print a model-averaged confidence-interval result

Usage

```
## S3 method for class 'marp_confint'
print(x, ...)
```

Arguments

x An object returned by [marp_confint()] or 'confint.marp_fit()'.
 ... Additional arguments, currently unused.

Value

The input object, invisibly.

print.marp_fit	<i>Print a model-averaged renewal-process fit</i>
----------------	---

Description

Print a model-averaged renewal-process fit

Usage

```
## S3 method for class 'marp_fit'
print(x, ...)
```

Arguments

x An object returned by [marp()].
 ... Additional arguments, currently unused.

Value

The input object, invisibly.

print.marp_model_fit	<i>Print a model-specific renewal-process fit</i>
----------------------	---

Description

Print a model-specific renewal-process fit

Usage

```
## S3 method for class 'marp_model_fit'
print(x, ...)
```

Arguments

x An object returned by one of the model-specific fitting functions.
 ... Additional arguments, currently unused.

Value

The input object, invisibly.

```
print.summary_marp_fit
```

Print a summary of a model-averaged renewal-process fit

Description

Print a summary of a model-averaged renewal-process fit

Usage

```
## S3 method for class 'summary_marp_fit'  
print(x, ...)
```

Arguments

x	A 'summary_marp_fit' object.
...	Additional arguments, currently unused.

Value

The input object, invisibly.

```
print.summary_marp_model_fit
```

Print a summary of a model-specific renewal-process fit

Description

Print a summary of a model-specific renewal-process fit

Usage

```
## S3 method for class 'summary_marp_model_fit'  
print(x, ...)
```

Arguments

x	A 'summary_marp_model_fit' object.
...	Additional arguments, currently unused.

Value

The input object, invisibly.

student_confint	<i>A function to calculate Studentized bootstrap confidence interval</i>
-----------------	--

Description

A function to calculate Studentized bootstrap confidence interval

Usage

```
student_confint(
  n,
  B,
  t,
  m,
  BB,
  par_hat,
  mu_hat,
  pr_hat,
  haz_hat,
  weights,
  alpha,
  y,
  best.model,
  which.model = 1
)
```

Arguments

n	Number of inter-event times generated in each bootstrap sample.
B	Number of bootstrap samples.
t	Time points at which log-hazards are evaluated.
m	Positive integer controlling repeated random-start optimizations; see [marp()].
BB	Number of double-bootstrap samples per bootstrap sample.
par_hat	Length-12 vector containing ‘par1’ for models 1–6 followed by ‘par2’ for models 1–6.
mu_hat	Length-6 vector of model-specific mean estimates.
pr_hat	Length-6 vector of model-specific logit event probabilities.
haz_hat	Matrix of model-specific log-hazards with ‘length(t)’ rows and six model columns.
weights	Length-6 vector of model weights.
alpha	Significance level; the confidence level is ‘1 - alpha’.
y	Time point at which logit event probabilities are evaluated.
best.model	Integer identifying the lowest-AIC model.
which.model	Integer identifying the reference model using the mapping 1 = Poisson, 2 = Gamma, 3 = log-logistic, 4 = Weibull, 5 = log-normal, and 6 = BPT.

Value

A list of studentized-bootstrap confidence limits. Components beginning with ‘pr_’ are on the logit-probability scale and components beginning with ‘haz_’ are on the log-hazard scale.

mu_lower_gen Lower limit of the studentized bootstrap confidence interval of the estimated mean based on the generating model

mu_upper_gen Upper limit of the studentized bootstrap confidence interval of the estimated mean based on the generating model

mu_lower_best Lower limit of the studentized bootstrap confidence interval of the estimated mean based on the best model

mu_upper_best Upper limit of the studentized bootstrap confidence interval of the estimated mean based on the best model

pr_lower_gen Lower limit for the reference-model logit event probability

pr_upper_gen Upper limit for the reference-model logit event probability

pr_lower_best Lower limit for the best-model logit event probability

pr_upper_best Upper limit for the best-model logit event probability

haz_lower_gen Lower limits for reference-model log-hazards

haz_upper_gen Upper limits for reference-model log-hazards

haz_lower_best Lower limits for best-model log-hazards

haz_upper_best Upper limits for best-model log-hazards

mu_lower_ma Lower limit of model-averaged studentized bootstrap confidence interval of the estimated mean

mu_upper_ma Upper limit of model-averaged studentized bootstrap confidence interval of the estimated mean

pr_lower_ma Lower model-averaged limit on the logit-probability scale

pr_upper_ma Upper model-averaged limit on the logit-probability scale

haz_lower_ma Lower model-averaged limits on the log-hazard scale

haz_upper_ma Upper model-averaged limits on the log-hazard scale

Examples

```
## Not run:
# generate random data
set.seed(42)
data <- rgamma(30, 3, 0.01)

# set some parameters
n <- 30 # sample size
m <- 10 # repeated random-start optimization setting
t <- seq(100,200,by=10) # time intervals
y <- 304 # cut-off year for estimating probability
B <- 100 # number of bootstraps
BB <- 100 # number of double bootstraps
par_hat <- c(
```

```

3.41361e-03, 2.76268e+00, 2.60370e+00, 3.30802e+02, 5.48822e+00, 2.92945e+02, NA,
9.43071e-03, 2.47598e+02, 1.80102e+00, 6.50845e-01, 7.18247e-01)
mu_hat <- c(292.94512, 292.94513, 319.72017, 294.16945, 298.87286, 292.94512)
pr_hat <- c(0.60039, 0.42155, 0.53434, 0.30780, 0.56416, 0.61795)
haz_hat <- matrix(c(
-5.67999, -5.67999, -5.67999, -5.67999, -5.67999, -5.67999,
-5.67999, -5.67999, -5.67999, -5.67999, -5.67999, -6.09420,
-5.99679, -5.91174, -5.83682, -5.77031, -5.71085, -5.65738,
-5.60904, -5.56512, -5.52504, -5.48833, -6.09902, -5.97017,
-5.85769, -5.75939, -5.67350, -5.59856, -5.53336, -5.47683,
-5.42805, -5.38621, -5.35060, -6.17146, -6.09512, -6.02542,
-5.96131, -5.90194, -5.84668, -5.79498, -5.74642, -5.70064,
-5.65733, -5.61624, -5.92355, -5.80239, -5.70475, -5.62524,
-5.55994, -5.50595, -5.46106, -5.42359, -5.39222, -5.36591,
-5.34383, -5.79111, -5.67660, -5.58924, -5.52166, -5.46879,
-5.42707, -5.39394, -5.36751, -5.34637, -5.32946, -5.31596
),length(t),6)
weights <- c(0.00000, 0.21000, 0.02000, 0.55000, 0.00000, 0.22000) # model weights
alpha <- 0.05 # confidence level
y <- 304 # cut-off year for estimating probability
best.model <- 2
which.model <- 2 # specify the generating model#'

# construct Studentized bootstrap confidence interval
marp::student_confint(
  n,B,t,m,BB,par_hat,mu_hat,pr_hat,haz_hat,weights,alpha,y,best.model,which.model
)

## End(Not run)

```

summary.marp_fit

*Summarise a model-averaged renewal-process fit***Description**

Summarise a model-averaged renewal-process fit

Usage

```
## S3 method for class 'marp_fit'
summary(object, ...)
```

Arguments

object	An object returned by [marp()].
...	Additional arguments, currently unused.

Value

A structured object of class ‘summary_marp_fit’.

```
summary.marp_model_fit
```

Summarise a model-specific renewal-process fit

Description

Summarise a model-specific renewal-process fit

Usage

```
## S3 method for class 'marp_model_fit'
summary(object, ...)
```

Arguments

object	An object returned by one of the model-specific fitting functions.
...	Additional arguments, currently unused.

Value

A structured object of class ‘summary_marp_model_fit’.

```
upperT
```

Evaluate the upper-limit equation for a model-averaged T statistic

Description

Evaluate the upper-limit equation for a model-averaged T statistic

Usage

```
upperT(up, hat, sigmasq, Tstar, weights, B, alpha)
```

Arguments

up	Candidate upper confidence limit.
hat	Vector of model-specific estimates.
sigmasq	Vector of model-specific variance estimates.
Tstar	Matrix of bootstrap T statistics, with models in rows.
weights	Vector of model weights.
B	Number of bootstrap samples represented in ‘Tstar’.
alpha	Significance level.

Value

A numeric scalar giving the upper-limit root equation value.

Examples

```
# set some parameters
up <- 100 # upper bound
hat <- rep(150, 6) # estimates obtained from each model
sigmasq <- 10 # variance
Tstar <- matrix(rep(100,600),6,100) # T statistics estimated from bootstrap samples
weights <- rep(1/6, 6) # model weights
B <- 100 # number of bootstrapped samples
alpha <- 0.05 # confidence level

# calculate the upper limit of T statistics
res <- marp::upperT(up, hat, sigmasq, Tstar, weights, B, alpha)

res
```

weibull_bstrp	<i>A function to generate (double) bootstrap samples and fit Weibull renewal model</i>
---------------	--

Description

A function to generate (double) bootstrap samples and fit Weibull renewal model

Usage

```
weibull_bstrp(n, t, B, BB, m, par_hat, mu_hat, pr_hat, haz_hat, y)
```

Arguments

n	Number of inter-event times generated in each bootstrap sample.
t	Time points at which log-hazards are evaluated.
B	Number of bootstrap samples.
BB	Number of double-bootstrap samples per bootstrap sample.
m	Positive integer controlling repeated random-start optimizations; see [marp()].
par_hat	Length-12 vector containing ‘par1’ for models 1–6 followed by ‘par2’ for models 1–6; see [marp()] for model order.
mu_hat	Length-6 vector of model-specific mean estimates.
pr_hat	Length-6 vector of model-specific logit event probabilities.
haz_hat	Matrix of model-specific log-hazards with ‘length(t)’ rows and six model columns.
y	Time point at which logit event probabilities are evaluated.

Value

A list of bootstrap estimates and variance/T-statistic quantities. Components beginning with 'pr_' use the logit-probability scale and those beginning with 'haz_' use the log-hazard scale.

mu_star Estimated mean from bootstrapped samples

pr_star Logit event probabilities from bootstrap samples

haz_star Log-hazards from bootstrap samples

mu_var_hat Variance of estimated mean

pr_var_hat Variance of estimated probability

haz_var_hat Variance of estimated hazard rates

mu_var_double Variance of estimated mean of bootstrapped samples (via double-bootstrapping)

pr_var_double Variance of estimated probability of bootstrapped samples (via double-bootstrapping)

haz_var_double Variance of estimated hazard rates of bootstrapped samples (via double-bootstrapping)

mu_Tstar Pivot quantity of the estimated mean

pr_Tstar Pivot quantity of the estimated probability

haz_Tstar Pivot quantity of the estimated hazard rates

Examples

```
## Not run:
# set some parameters
n <- 30 # sample size
t <- seq(100, 200, by = 10) # time intervals
B <- 100 # number of bootstraps
BB <- 100 # number of double-bootstraps
m <- 10 # repeated random-start optimization setting
par_hat <- c(
  3.4136086430979953e-03, 2.7626793657057762e+00, 2.6037039674870583e+00, 3.3080162440951688e+02,
  5.4882183788378658e+00, 2.9294512422957860e+02, NA, 9.4307059277139432e-03,
  2.4759796859031687e+02, 1.8010183507666513e+00, 6.5084541680686814e-01, 7.1824719073918109e-01
)
mu_hat <- c(
  292.94512187913182, 292.94512912200048, 319.72017228620746, 294.16945213908519,
  298.87285747700128, 292.94512422957860
)
pr_hat <- c(
  0.60038574701819891, 0.42154974433034809, 0.53433568234281148, 0.30779792692414687,
  0.56416103510057725, 0.61794524610544410
)
haz_hat <- matrix(c(
  -5.6799852941338829, -5.6799852941338829, -5.6799852941338829, -5.6799852941338829,
  -5.6799852941338829, -5.6799852941338829, -5.6799852941338829, -5.6799852941338829,
  -5.6799852941338829, -5.6799852941338829, -5.6799852941338829, -6.0942031084732298,
  -5.9967873794574516, -5.9117418563554684, -5.8368230853439300, -5.7703089176306639,
  -5.7108525626839901, -5.6573839062669986, -5.6090408956082456, -5.5651206740587922,
  -5.5250440506799734, -5.4883291920475745, -6.0990192429336094, -5.9701664705134210,
  -5.8576899644670348, -5.7593884711134971, -5.6734972529860741, -5.5985621349393231,
```

```

-5.5333565788683616, -5.4768259914915305, -5.4280496904694857, -5.3862145095364315,
-5.3505961502861927, -6.1714638710963881, -6.0951186680582552, -6.0254209583640863,
-5.9613052806725335, -5.9019434350392981, -5.8466788789061646, -5.7949823391436279,
-5.7464209045603756, -5.7006359661738628, -5.6573271297614109, -5.6162402596857071,
-5.9235521978533958, -5.8023896004395645, -5.7047473880293342, -5.6252373537796752,
-5.5599409055534252, -5.5059486025117375, -5.4610610586440487, -5.4235891601883868,
-5.3922173604047572, -5.3659081375131672, -5.3438339586221275, -5.7911126719889303,
-5.6765973314326752, -5.5892417143301261, -5.5216608261560411, -5.4687921205249133,
-5.4270729562323066, -5.3939387902533049, -5.3675067327627373, -5.3463701567645607,
-5.3294619641245422, -5.3159614865560094
),length(t),6)
y <- 304 # cut-off year for estimating probability

# generate bootstrapped samples then fit renewal model
res <- marp::weibull_bstrp(n, t, B, BB, m, par_hat, mu_hat, pr_hat, haz_hat, y)

## End(Not run)

```

weibull_logl

A function to calculate the log-likelihood of Weibull model

Description

A function to calculate the log-likelihood of Weibull model

Usage

```
weibull_logl(param, x)
```

Arguments

param	Length-2 numeric vector containing log(scale) and log(shape).
x	Numeric vector of positive observations.

Value

returns the value of negative log-likelihood of the Weibull model

Examples

```

set.seed(42)
data <- rgamma(30,3,0.01)

# set some parameters
par_hat <- c(330.801103808081, 1.80101338777944) # estimated parameters
param <- log(par_hat) # input parameters for logl function

# calculate log-likelihood

```

```
result <- marp::weibull_logl(param, data)

result
```

weibull_rp

A function to fit Weibull renewal model

Description

A function to fit Weibull renewal model

Usage

```
weibull_rp(data, t, m, y)
```

Arguments

data	A numeric vector of positive inter-event times.
t	A numeric vector of time points at which log-hazards are evaluated.
m	A positive integer controlling repeated random-start optimizations; the current implementation seeks ‘m - 1’ acceptable ‘nlm()’ fits.
y	A time point at which the logit-transformed cumulative event probability is evaluated.

Value

An object of class ‘marp_model_fit’ retaining the following eight named list components:

par1 Estimated scale parameter of the Weibull model

par2 Estimated shape parameter of the Weibull model

logL Maximized log-likelihood

AIC Akaike information criterion (AIC)

BIC Bayesian information criterion (BIC)

mu_hat Estimated mean inter-event time

pr_hat Logit-transformed cumulative event probability at ‘y’

haz_hat Log-hazard values at ‘t’

Examples

```
set.seed(42)
data <- rgamma(100,3,0.01)

# set some parameters
m = 10 # repeated random-start optimization setting
t = seq(100, 200, by=10) # time intervals
y = 304 # cut-off year for estimating probablity

# fit Weibull renewal model
fit <- marp::weibull_rp(data, t, m, y)
fit
summary(fit)
```

Index

bpt_bstrp, [3](#)
bpt_logl, [4](#)
bpt_rp, [5](#)

coef.marp_model_fit, [6](#)
confint.marp_fit, [7](#)

dllog, [8](#)

gamma_bstrp, [8](#)
gamma_logl, [10](#)
gamma_rp, [11](#)

logLik.marp_model_fit, [12](#)
loglogis_bstrp, [12](#)
loglogis_logl, [14](#)
loglogis_rp, [15](#)
lognorm_bstrp, [16](#)
lognorm_rp, [18](#)
lowerT, [19](#)

marp, [20](#)
marp_bstrp, [21](#)
marp_confint, [23](#)

nobs.marp_fit, [26](#)
nobs.marp_model_fit, [27](#)

percent_confint, [27](#)
pllog, [29](#)
poisson_bstrp, [30](#)
poisson_rp, [31](#)
print.marp_confint, [32](#)
print.marp_fit, [33](#)
print.marp_model_fit, [33](#)
print.summary_marp_fit, [34](#)
print.summary_marp_model_fit, [34](#)

student_confint, [35](#)
summary.marp_fit, [37](#)
summary.marp_model_fit, [38](#)

upperT, [38](#)

weibull_bstrp, [39](#)
weibull_logl, [41](#)
weibull_rp, [42](#)