Package 'EstimationTools'

March 10, 2021

Type Package

Title Maximum Likelihood Estimation for Probability Functions from Data Sets

Version 2.1.0

Depends R (>= 3.0.0), DEoptim, survival, stringr, BBmisc

Imports Rdpack, utils, stats, numDeriv, boot, matrixStats, autoimage, graphics

RdMacros Rdpack

Suggests gamlss.dist, knitr, rmarkdown, AdequacyModel, readr, covr, testthat (>= 3.0.0), vdiffr, spelling

VignetteBuilder knitr, utils

Description Routines for parameter estimation for any probability density or mass function implemented in R via maximum likelihood (ML) given a data set. The main routines 'maxlogL' and 'maxlogLreg' are wrapper functions specifically developed for ML estimation. There are included optimization procedures such as 'nlminb' and 'optim' from base package, and 'DEoptim' Mullen (2011) <doi: 10.18637/jss.v040.i06>. Standard errors are estimated with 'numDeriv' Gilbert (2011) <https://CRAN.R-project.org/package=numDeriv> or the option 'Hessian = TRUE' of 'optim' function.

License GPL-3

URL https://jaimemosg.github.io/EstimationTools/, https://github.com/Jaimemosg/EstimationTools

BugReports https://github.com/Jaimemosg/EstimationTools/issues

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

Config/testthat/edition 3

Language en-US

NeedsCompilation no

Author Jaime Mosquera [aut, cre] (<https://orcid.org/0000-0002-1684-4756>),
Freddy Hernandez [ctb] (<https://orcid.org/0000-0001-7459-3329>)

Maintainer Jaime Mosquera <jmosquerag@unal.edu.co>

Repository CRAN

Date/Publication 2021-03-10 18:40:05 UTC

R topics documented:

`bootstrap_maxlogL` *Bootstrap computation of standard error for* `maxlogL` *class objects.*

Description

`bootstrap_maxlogL` computes standard errors of `maxlogL` class objects by non-parametric bootstrap.

Usage

```
bootstrap_maxlogL(object, R = 2000, silent = FALSE, ...)
```

Arguments

object	an object of maxlogL class whose standard errors are going to be computed by bootstrap.
R	numeric. It is the number of resamples performed with the dataset in bootstrap computation. Default value is 2000.
silent	logical. If TRUE, notifications of bootstrap_maxlogL are suppressed.
...	arguments passed to boot used in this routine for estimation of standard errors.

Details

The computation performed by this function may be invoked when Hessian from optim and hessian fail in maxlogL or in maxlogLreg.

However, this function can be run even if Hessian matrix calculation does not fails. In this case, standard errors in the maxlogL class object is replaced.

Value

A modified object of class maxlogL.

Author(s)

Jaime Mosquera Gutiérrez, <jmosquerag@unal.edu.co>

References

Canty A, Ripley BD (2017). *boot: Bootstrap R (S-Plus) Functions*.

See Also

maxlogL, maxlogLreg, boot

Examples

```
library(EstimationTools)

#---------------------------------------------------------------------------
# First example: Comparison between standard error computation via Hessian matrix
# and standard error computation via bootstrap

N <- rbinom(n = 100, size = 10, prob = 0.3)
phat1 <- maxlogL(x = N, dist = 'dbinom', fixed = list(size = 10),
                 link = list(over = "prob", fun = "logit_link"))

## Standard error computation method and results
print(phat1$outputs$StdE_Method)   # Hessian
summary(phat1)

## 'bootstrap_maxlogL' implementation
phat2 <- phat1                   # Copy the first 'maxlogL' object
```

```
bootstrap_maxlogL(phat2, R = 100)

## Standard error computation method and results
print(phat2$outputs$StdE_Method)   # Bootstrap
summary(phat2)


#-------------------------------------------------------------------------------
```

`Fibers` *Tensile strengths*

Description

Tensile strengths (in GPa) of 69 specimens of carbon fiber tested under tension at gauge lengths of 20 mm.

Usage

```
Fibers
```

Format

A data frame with 69 observations.

`Hazard_Shape` *Summary of* `HazardShape` *objects*

Description

This function displays the estimated hazard shape given a data set.

Usage

```
Hazard_Shape(object)
```

Arguments

`object`	an object of class `HazardShape`, generated with `TTT_hazard_shape`.

Author(s)

Jaime Mosquera Gutiérrez <jmosquerag@unal.edu.co>

Examples

```
#-------------------------------------------------------------------------------
# Example 1: Increasing hazard and its corresponding TTT plot with simulated data
hweibull <- function(x, shape, scale){
  dweibull(x, shape, scale)/pweibull(x, shape, scale, lower.tail = FALSE)
  }
curve(hweibull(x, shape = 2.5, scale = pi), from = 0, to = 42,
               col = "red", ylab = "Hazard function", las = 1, lwd = 2)

y <- rweibull(n = 50, shape = 2.5, scale = pi)
my_initial_guess <- TTT_hazard_shape(formula = y ~ 1)
Hazard_Shape(my_initial_guess)


#-------------------------------------------------------------------------------
```

`interp.options` *Configure various aspects of interpolating function in* `TTT_hazard_shape`

Description

This function allows the user to set the parameters of any of the following interpolating functions which can be used inside `TTT_hazard_shape`.

Usage

```
interp.options(interp.fun = "splinefun", length.out = 10, ...)
```

Arguments

`interp.fun`	character. This argument defines the interpolating function used. Default value is `"splinefun"`. Visit the **Details** section for further information.
`length.out`	numeric. Number of points interpolated. Default value is 10.
`...`	further arguments passed to the interpolating function.

Details

Each interpolating function has its particular arguments. The following interpolating functions are recommended:

- `approxfun`
- `splinefun`
- `spline`

The user can also implement a custom interpolating function.

Author(s)

Jaime Mosquera Gutiérrez <jmosquerag@unal.edu.co>

See Also

approxfun, splinefun, smooth, smooth.spline, loess, TTT_hazard_shape

---

is.maxlogL *Is return of any object of* EstimationTools*?*

Description

Checks if an object is any of the classes implemented in EstimationTools package.

Usage

```
is.maxlogL(x)

is.EmpiricalTTT(x)

is.HazardShape(x)
```

Arguments

x	Any object of EstimationTools.

Author(s)

Jaime Mosquera Gutiérrez <jmosquerag@unal.edu.co>

---

loess.options *Configure various aspects of LOESS in* TTT_hazard_shape

---

Description

This function allows the user to set the parameters of loess function used inside TTT_hazard_shape.

Usage

```
loess.options(span = 2/3, ...)
```

Arguments

span	the parameter which controls the degree of smoothing.
...	further arguments passed to loess function.

Details

Please, visit loess to know further possible arguments. The following arguments are not available for passing to the LOESS estimation:

- dataThe only data handled inside TTT_hazard_shape is the computed empirical TTT.
- subsetThis argument is used in loess to take a subset of data. In this context, it is not necessary.

Author(s)

Jaime Mosquera Gutiérrez <jmosquerag@unal.edu.co>

See Also

loess, TTT_hazard_shape

---

logit_link *Logit link function (for estimation with* maxlogL *object)*

Description

log_link object provides a way to implement logit link function that maxlogL needs to perform estimation. See documentation for maxlogL for further information on parameter estimation and implementation of link objects.

Usage

```
logit_link()
```

Details

logit_link is part of a family of generic functions with no input arguments that defines and returns a list with details of the link function:

1. name: a character string with the name of the link function.
2. g: implementation of the link function as a generic function in R.
3. g_inv: implementation of the inverse link function as a generic function in R.

There is a way to add new mapping functions. The user must specify the details aforesaid.

Value

A list with logit link function, its inverse and its name.

Author(s)

Jaime Mosquera Gutiérrez, <jmosquerag@unal.edu.co>

See Also

maxlogL

Other link functions: NegInv_link(), log_link()

Examples

```
# Estimation of proportion in binomial distribution with 'logit' function
# 10 trials, probability of success equals to 30%)
N <- rbinom(n = 100, size = 10, prob = 0.3)
phat <- maxlogL(x = N, dist = 'dbinom', fixed = list(size=10),
                link = list(over = "prob", fun = "logit_link"))
summary(phat)

# Link function name
fun <- logit_link()$name
print(fun)

# Link function
g <- logit_link()$g
curve(g(x), from = 0, to = 1)

# Inverse link function
ginv <- logit_link()$g_inv
curve(ginv(x), from = -10, to = 10)
```

---

log_link *Logarithmic link function (for estimation with* maxlogL *object)*

Description

log_link object provides a way to implement logarithmic link function that maxlogL needs to perform estimation. See documentation for maxlogL for further information on parameter estimation and implementation of link objects.

Usage

```
log_link()
```

Details

log_link is part of a family of generic functions with no input arguments that defines and returns a list with details of the link function:

1. name: a character string with the name of the link function.
2. g: implementation of the link function as a generic function in R.
3. g_inv: implementation of the inverse link function as a generic function in R.

There is a way to add new mapping functions. The user must specify the details aforesaid.

Value

A list with logit link function, its inverse and its name.

See Also

maxlogL

Other link functions: NegInv_link(), logit_link()

Examples

```
# One parameters of normal distribution mapped with logarithmic function
x <- rnorm(n = 10000, mean = 50, sd = 4)
theta_2 <- maxlogL( x = x, link = list(over = "sd",
                                       fun = "log_link") )
summary(theta_2)

# Link function name
fun <- log_link()$name
print(fun)

# Link function
g <- log_link()$g
curve(g(x), from = 0, to = 1)

# Inverse link function
ginv <- log_link()$g_inv
curve(ginv(x), from = -5, to = 5)
```

---

maxlogL *Maximum Likelihood Estimation for parametric distributions*

Description

Function to compute maximum likelihood estimators (MLE) of any distribution implemented in R.

Usage

```
maxlogL(
  x,
  dist = "dnorm",
  fixed = NULL,
  link = NULL,
  start = NULL,
  lower = NULL,
  upper = NULL,
  optimizer = "nlminb",
  control = NULL,
```

```
  silent = FALSE,
  ...
)
```

Arguments

x	a vector with data to be fitted. This argument must be a matrix with hierarchical distributions.
dist	a length-one character vector with the name of density/mass function of interest. The default value is 'dnorm', to compute maximum likelihood estimators of normal distribution.
fixed	a list with fixed/known parameters of distribution of interest. Fixed parameters must be passed with its name.
link	a list with names of parameters to be linked, and names of the link function object. For names of parameters, please visit documentation of density/mass function. There are three link functions available: log_link, logit_link and NegInv_link.
start	a numeric vector with initial values for the parameters to be estimated.
lower	a numeric vector with lower bounds, with the same length of argument start (for box-constrained optimization).
upper	a numeric vector with upper bounds, with the same length of argument start (for box-constrained optimization).
optimizer	a length-one character vector with the name of optimization routine. nlminb, optim and DEoptim are available; nlminb is the default routine.
control	control parameters of the optimization routine. Please, visit documentation of selected optimizer for further information.
silent	logical. If TRUE, warnings of maxlogL are suppressed.
...	further arguments to be supplied to the optimizer.

Details

maxlogL computes the likelihood function corresponding to the distribution specified in argument dist and maximizes it through optim, nlminb or DEoptim. maxlogL generates an S3 object of class maxlogL.

Noncentrality parameters must be named as ncp in the distribution.

Value

A list with class "maxlogL" containing the following lists:

fit	A list with output information about estimation.
inputs	A list with all input arguments.
outputs	A list with some output additional information: • Number of parameters. • Sample size • Standard error computation method.

Note

The following generic functions can be used with a maxlogL object: summary,print,AIC,BIC,logLik.

Author(s)

Jaime Mosquera Gutiérrez, <jmosquerag@unal.edu.co>

References

Nelder JA, Mead R (1965). "A Simplex Method for Function Minimization." *The Computer Journal*, **7**(4), 308–313. ISSN 0010-4620, doi: 10.1093/comjnl/7.4.308, https://academic.oup.com/comjnl/article-lookup/doi/10.1093/comjnl/7.4.308.

Fox PA, Hall AP, Schryer NL (1978). "The PORT Mathematical Subroutine Library." *ACM Transactions on Mathematical Software*, **4**(2), 104–126. ISSN 00983500, doi: 10.1145/355780.355783, https://dl.acm.org/doi/10.1145/355780.355783.

Nash JC (1979). *Compact Numerical Methods for Computers. Linear Algebra and Function Minimisation*, 2nd Editio edition. Adam Hilger, Bristol.

Dennis JE, Gay DM, Walsh RE (1981). "An Adaptive Nonlinear Least-Squares Algorithm." *ACM Transactions on Mathematical Software*, **7**(3), 348–368. ISSN 00983500, doi: 10.1145/355958.355965, https://dl.acm.org/doi/10.1145/355958.355965.

See Also

summary.maxlogL, optim, nlminb, DEoptim, DEoptim.control, maxlogLreg, bootstrap_maxlogL

Other maxlogL: maxlogLreg()

Examples

```
library(EstimationTools)

#---------------------------------------------------------------------------
# Example 1: estimation with one fixed parameter
x <- rnorm(n = 10000, mean = 160, sd = 6)
theta_1 <- maxlogL(x = x, dist = 'dnorm', control = list(trace = 1),
                 link = list(over = "sd", fun = "log_link"),
                 fixed = list(mean = 160))
summary(theta_1)


#---------------------------------------------------------------------------
# Example 2: both parameters of normal distribution mapped with logarithmic
# function
theta_2 <- maxlogL(x = x, dist = "dnorm",
                   link = list(over = c("mean","sd"),
                               fun = c("log_link","log_link")))
summary(theta_2)


#---------------------------------------------------------------------------
```

```
# Example 3: parameter estimation in ZIP distribution
if (!require('gamlss.dist')) install.packages('gamlss.dist')
library(gamlss.dist)
z <- rZIP(n=1000, mu=6, sigma=0.08)
theta_3  <- maxlogL(x = z, dist='dZIP', start = c(0, 0), lower = c(-Inf, -Inf),
                    upper = c(Inf, Inf), optimizer = 'optim',
                    link = list(over=c("mu", "sigma"),
                    fun = c("log_link", "logit_link")))
summary(theta_3)


#------------------------------------------------------------------------------
# Example 4: parameter estimation with fixed noncentrality parameter.
y_2 <- rbeta(n = 1000, shape1 = 2, shape2 = 3)
theta_41 <- maxlogL(x = y_2, dist = "dbeta",
                     link = list(over = c("shape1", "shape2"),
                     fun = c("log_link","log_link")))
summary(theta_41)

# It is also possible define 'ncp' as fixed parameter
theta_42 <- maxlogL(x = y_2, dist = "dbeta", fixed = list(ncp = 0),
                     link = list(over = c("shape1", "shape2"),
                     fun = c("log_link","log_link")) )
summary(theta_42)


#------------------------------------------------------------------------------
```

`maxlogLreg` *Maximum Likelihood Estimation for parametric linear regression models*

Description

Function to compute maximum likelihood estimators (MLE) of regression parameters of any distribution implemented in R with covariates (linear predictors).

Usage

```
maxlogLreg(
  formulas,
  y_dist,
  data = NULL,
  subset = NULL,
  fixed = NULL,
  link = NULL,
  start = NULL,
  lower = NULL,
```

```
  upper = NULL,
  optimizer = "nlminb",
  control = NULL,
  silent = FALSE,
  ...
)
```

Arguments

formulas	a list of formula objects. Each element must have an ~, with the terms on the right separated by + operators. The response variable on the left side is optional. Linear predictor of each parameter must be specified with the name of the parameter followed by the suffix '.fo'. See the examples below for further illustration.
y_dist	a formula object that specifies the distribution of the response variable. On the left side of ~ must be the response, and in the right side must be the name o de probability density/mass function. See the section **Details** and the examples below for further illustration.
data	an optional data frame containing the variables in the model. If data is not specified, the variables are taken from the environment from which maxlogLreg is called.
subset	an optional vector specifying a subset of observations to be used in the fitting process.
fixed	a list with fixed/known parameters of distribution of interest. Fixed parameters must be passed with its name and its value (known).
link	a list with names of parameters to be linked, and names of the link function object. For names of parameters, please visit documentation of density/mass function. There are three link functions available: log_link, logit_link and NegInv_link. Take into account: the order used in argument over corresponds to the order in argument link.
start	a numeric vector with initial values for the parameters to be estimated. Zero is the default value.
lower	a numeric vector with lower bounds, with the same lenght of argument start (for box-constrained optimization). -Inf is the default value.
upper	a numeric vector with upper bounds, with the same lenght of argument start (for box-constrained optimization). Inf is the default value.
optimizer	a lenght-one character vector with the name of optimization routine. nlminb, optim and DEoptim are available; nlminb is the default routine.
control	control parameters of the optimization routine. Please, visit documentation of selected optimizer for further information.
silent	logical. If TRUE, warnings of maxlogL are suppressed.
...	Further arguments to be supplied to the optimization routine.

Details

maxlogLreg calculates computationally the log-likelihood (log L) function corresponding to the distribution specified in argument y_dist with linear predictors specified in argument formulas. Then, it maximizes the log L through optim, nlminb or DEoptim. maxlogLreg generates an S3 object of class maxlogL.

Noncentrality parameters must be named as ncp in the distribution.

Value

A list with class maxlogL containing the following lists:

fit	A list with output information about estimation and method used.
inputs	A list with all input arguments.
outputs	A list with additional information:

- Number of parameters.
- Sample size
- Standard error computation method.
- Number of regression parameters.

Note

The following generic functions can be used with a maxlogL object: summary,print,logLik,AIC.

Author(s)

Jaime Mosquera Gutiérrez, <jmosquerag@unal.edu.co>

References

Nelder JA, Mead R (1965). "A Simplex Method for Function Minimization." *The Computer Journal*, **7**(4), 308–313. ISSN 0010-4620, doi: 10.1093/comjnl/7.4.308, https://academic.oup.com/comjnl/article-lookup/doi/10.1093/comjnl/7.4.308.

Fox PA, Hall AP, Schryer NL (1978). "The PORT Mathematical Subroutine Library." *ACM Transactions on Mathematical Software*, **4**(2), 104–126. ISSN 00983500, doi: 10.1145/355780.355783, https://dl.acm.org/doi/10.1145/355780.355783.

Nash JC (1979). *Compact Numerical Methods for Computers. Linear Algebra and Function Minimisation*, 2nd Editio edition. Adam Hilger, Bristol.

Dennis JE, Gay DM, Walsh RE (1981). "An Adaptive Nonlinear Least-Squares Algorithm." *ACM Transactions on Mathematical Software*, **7**(3), 348–368. ISSN 00983500, doi: 10.1145/355958.355965, https://dl.acm.org/doi/10.1145/355958.355965.

See Also

summary.maxlogL, optim, nlminb, DEoptim, DEoptim.control, maxlogL, bootstrap_maxlogL

Other maxlogL: maxlogL()

Examples

```
library(EstimationTools)

#---------------------------------------------------------------------------
# Example 1: Estimation in simulated normal distribution
n <- 1000
x <- runif(n = n, -5, 6)
y <- rnorm(n = n, mean = -2 + 3 * x, sd = exp(1 + 0.3* x))
norm_data <- data.frame(y = y, x = x)

# It does not matter the order of distribution parameters
formulas <- list(sd.fo = ~ x, mean.fo = ~ x)

norm_mod <- maxlogLreg(formulas, y_dist = y ~ dnorm, data = norm_data,
                       link = list(over = "sd", fun = "log_link"))
summary(norm_mod)


#---------------------------------------------------------------------------
# Example 2: Fitting with censorship
# (data from https://www.itl.nist.gov/div898/handbook/apr/section4/apr413.htm)

failures <- c(55, 187, 216, 240, 244, 335, 361, 373, 375, 386)
fails <- c(failures, rep(500, 10))
status <- c(rep(1, length(failures)), rep(0, 10))
Wei_data <- data.frame(fails = fails, status = status)

# Formulas with linear predictors
formulas <- list(scale.fo=~1, shape.fo=~1)

# Bounds for optimization. Upper bound set with default values (Inf)
start <- list(
  scale = list(Intercept = 100),
  shape = list(Intercept = 10)
)
lower <- list(
  scale = list(Intercept = 0),
  shape = list(Intercept = 0)
)

mod_weibull <- maxlogLreg(formulas, y_dist = Surv(fails, status) ~ dweibull,
                          start = start,
                          lower = lower, data = Wei_data)
summary(mod_weibull)


#---------------------------------------------------------------------------
```

NegInv_link *Negative inverse link function (for estimation with* maxlogL *object)*

Description

NegInv_link object provides a way to implement negative inverse link function that maxlogL needs to perform estimation. See documentation for maxlogL for further information on parameter estimation and implementation of link objects.

Usage

```
NegInv_link()
```

Details

NegInv_link is part of a family of generic functions with no input arguments that defines and returns a list with details of the link function:

1. name: a character string with the name of the link function.
2. g: implementation of the link function as a generic function in R.
3. g_inv: implementation of the inverse link function as a generic function in R.

There is a way to add new mapping functions. The user must specify the details aforesaid.

Value

A list with negative inverse link function, its inverse and its name.

See Also

maxlogL

Other link functions: log_link(), logit_link()

Examples

```
# Estimation of rate parameter in exponential distribution
T <- rexp(n = 1000, rate = 3)
lambda <- maxlogL(x = T, dist = "dexp", start = 5,
                  link = list(over = "rate", fun = "NegInv_link"))
summary(lambda)

# Link function name
fun <- NegInv_link()$name
print(fun)

# Link function
g <- NegInv_link()$g
curve(g(x), from = 0.1, to = 1)

# Inverse link function
ginv <- NegInv_link()$g_inv
curve(ginv(x), from = 0.1, to = 1)
```

`plot.EmpiricalTTT` *Plot method for* `EmpiricalTTT` *objects*

Description

Draws a TTT plot of an `EmpiricalTTT` object, one for each strata.

TTT plots are graphed in the same order in which they appear in the list element `strata` or in the list element `phi_n` of the `EmpiricalTTT` object.

Usage

```
## S3 method for class 'EmpiricalTTT'
plot(
  x,
  add = FALSE,
  grid = FALSE,
  type = "l",
  pch = 1,
  xlab = "i/n",
  ylab = expression(phi[n](i/n)),
  ...
)
```

Arguments

`x`	an object of class `EmpiricalTTT`.
`add`	logical. If TRUE, `plot.EmpiricalTTT` add a TTT plot to an already existing plot.
`grid`	logical. If `TRUE`, plot appears with grid.
`type`	character string (length 1 vector) or vector of 1-character strings indicating the type of plot for each TTT graph. See `plot`.
`pch`	numeric (integer). A vector of plotting characters or symbols when `type = "p"`. See `points`.
`xlab, ylab`	titles for x and y axes, as in `plot`.
`...`	further arguments passed to `matplot`. See the examples and **Details** section for further information.

Details

This method is based on `matplot`. Our function sets some default values for graphic parameters: `type = "l"`, `pch = 1`, `xlab = "i/n"` and `ylab = expression(phi[n](i/n))`. This arguments can be modified by the user.

Author(s)

Jaime Mosquera Gutiérrez, <jmosquerag@unal.edu.co>

See Also

TTTE_Analytical, matplot

Examples

```
library(EstimationTools)

#-------------------------------------------------------------------------------
# First example: Scaled empirical TTT from 'mgus1' data from 'survival' package.

TTT_1 <- TTTE_Analytical(Surv(stop, event == 'pcm') ~1, method = 'cens',
                         data = mgus1, subset=(start == 0))
plot(TTT_1, type = "p")


#-------------------------------------------------------------------------------
# Second example: Scaled empirical TTT using a factor variable with 'aml' data
# from 'survival' package.

TTT_2 <- TTTE_Analytical(Surv(time, status) ~ x, method = "cens", data = aml)
plot(TTT_2, type = "l", lty = c(1,1), col = c(2,4))
plot(TTT_2, add = TRUE, type = "p", lty = c(1,1), col = c(2,4), pch = 16)

#-------------------------------------------------------------------------------
# Third example: Non-scaled empirical TTT without a factor (arbitrarily simulated
# data).

y <- rweibull(n=20, shape=1, scale=pi)
TTT_3 <- TTTE_Analytical(y ~ 1, scaled = FALSE)
plot(TTT_3, type = "s", col = 3, lwd = 3)


#-------------------------------------------------------------------------------
# Fourth example: TTT plot for 'carbone' data from 'AdequacyModel' package

if (!require('AdequacyModel')) install.packages('AdequacyModel')
library(AdequacyModel)
data(carbone)
TTT_4 <- TTTE_Analytical(response = carbone, scaled = TRUE)
plot(TTT_4, type = "l", col = "red", lwd = 2, grid = TRUE)


#-------------------------------------------------------------------------------
```

---

plot.HazardShape *Plot of* HazardShape *objects*

Description

Draws the empirical total time on test (TTT) plot and its non-parametric (LOESS) estimated curve useful for identifying hazard shape.

Usage

```
## S3 method for class 'HazardShape'
plot(
  x,
  xlab = "i/n",
  ylab = expression(phi(i/n)),
  xlim = c(0, 1),
  ylim = c(0, 1),
  col = 1,
  lty = NULL,
  lwd = NA,
  main = "",
  curve_options = list(col = 2, lwd = 2, lty = 1),
  par_plot = list(mar = c(5.1, 4.1, 4.1, 2.1)),
  legend_options = NULL,
  ...
)
```

Arguments

x	an object of class initValOW, generated with TTT_hazard_shape.
xlab, ylab	titles for x and y axes, as in plot.
xlim	the x limits (x1, x2) of the plot.
ylim	the y limits (x1, x2) of the plot.
col	The colors for lines and points. Multiple colors can be specified. This is the usual color argument of plot.default.
lty	a vector of line types, see par for further information.
lwd	a vector of line widths, see par for further information.
main	a main title for the plot.
curve_options	a list with further arguments useful for customization of non-parametric estimate plot.
par_plot	some graphical parameters which can be passed to the plot. See **Details** section for further information.
legend_options	a list with fur further arguments useful for customization. See **Details** section for further information. of the legend of the plot.
...	further arguments passed to empirical TTT plot.

Details

This plot complements the use of TTT_hazard_shape. It is always advisable to use this function in order to check the result of non-parametric estimate of TTT plot. See the first example in **Examples** section for an illustration.

par_plot admits some parameters of par function. The following *has preestablished values*:

- mai:the margins can be manipulated with mar. The right margin has a value equals to 7.2 using mar and all new values take it as reference value.
- xpd:Is set as TRUE, and cannot be modified.

On the other hand, legend_options allows many of the parameters of legend function. The following *has preestablished values*:

- x,y:legend is always located on the right side, outside the plot. x and y coordinates cannot be manipulated, instead of this, it exists the argument pos, which can take character or numeric values. In the first case, it can be "top","center" and "bottom", in the later, it can be any value of the y-coordinate, between 0 and 1.
- legend:text of the legend cannot be edited.
- pch:cannot be manipulated, it depends on pch parameter of the plot.
- col:cannot be manipulated, it depends on col parameters of the plot and the curve_options. #'
- lty:cannot be manipulated, it depends on lty parameters of the plot and the curve_options. #'
- lwd:cannot be manipulated, it depends on lwd parameters of the plot and the curve_options. #'
- pt.cex:cannot be manipulated, it depends on cex parameter of the plot.
- xpd:It is set as TRUE, and cannot be modified.

If legend_optinos = "NoLegend", no legend is generated.

The possible arguments for ... can be consulted in plot.default and par.

Author(s)

Jaime Mosquera Gutiérrez <jmosquerag@unal.edu.co>

Examples

```
#-------------------------------------------------------------------------------
# Example 1: Increasing hazard and its corresponding TTT plot with simulated data
hweibull <- function(x, shape, scale){
  dweibull(x, shape, scale)/pweibull(x, shape, scale, lower.tail = FALSE)
  }
curve(hweibull(x, shape = 2.5, scale = pi), from = 0, to = 42,
               col = "red", ylab = "Hazard function", las = 1, lwd = 2)

y <- rweibull(n = 50, shape = 2.5, scale = pi)
my_initial_guess <- TTT_hazard_shape(formula = y ~ 1)
plot(my_initial_guess, par_plot=list(mar=c(3.7,3.7,1,1.5),
```

```
                                        mgp=c(2.5,1,0)))

#-------------------------------------------------------------------------------
```

`predict.maxlogL` *Predict Method for* `maxlogL` *Fits*

Description

This function computes predictions and optionally the estimated standard errors of those predictions from a model fitted with `maxlogLreg`.

Usage

```
## S3 method for class 'maxlogL'
predict(
  object,
  parameter = NULL,
  newdata = NULL,
  type = c("link", "response", "terms"),
  se.fit = FALSE,
  terms = NULL,
  ...
)
```

Arguments

`object`	an object of `maxlogL` class.
`parameter`	a character which specifies the paramter to predict.
`newdata`	a data frame with covariates with which to predict. It is an optional argument, if omitted, the fitted linear predictors or the (distribution) parameter predictions are used.
`type`	a character with the type of prediction required. The default (`type = "link"`) is on the scale of the linear predictors; the alternative `type = "response"` is on the scale of the distribution parameter.
`se.fit`	logical switch indicating if standard errors of predictions are required.
`terms`	A character vector that specifies which terms are required if `type = "terms"`. All terms are returned by default.
`...`	further arguments passed to or from other methods.

Details

This `summary` method computes and displays AIC, BIC, estimates and standard errors from a estimated model stored i a `maxlogL` class object. It also displays and computes Z-score and p values of significance test of parameters.

Value

If se.fit = FALSE, a vector of predictions is returned. For type = "terms", a matrix with a column per term and an attribute "constant" is returned.

If se.fit = TRUE, a list with the following components is obtained:

1. fit: Predictions.
2. se.fit: Estimated standard errors.

Note

Variables are first looked for in newdata and then searched for in the usual way (which will include the environment of the formula used in the fit). A warning will be given if the variables found are not of the same length as those in newdata if it is supplied.

Author(s)

Jaime Mosquera Gutiérrez, <jmosquerag@unal.edu.co>

Examples

```
library(EstimationTools)

#-------------------------------------------------------------------------------
# Example 1: Predictions from simulated normal distribution
n <- 1000
x <- runif(n = n, -5, 6)
y <- rnorm(n = n, mean = -2 + 3 * x, sd = exp(1 + 0.3* x))
norm_data <- data.frame(y = y, x = x)

# It does not matter the order of distribution parameters
formulas <- list(sd.fo = ~ x, mean.fo = ~ x)

norm_mod <- maxlogLreg(formulas, y_dist = y ~ dnorm, data = norm_data,
                       link = list(over = "sd", fun = "log_link"))
predict(norm_mod)


#-------------------------------------------------------------------------------
# Example 2: Predictions using new values for covariates
predict(norm_mod, newdata = data.frame(x=0:6))


#-------------------------------------------------------------------------------
# Example 3: Predictions for another parameter
predict(norm_mod, newdata = data.frame(x=0:6), param = "sd",
       type = "response")

#-------------------------------------------------------------------------------
# Example 4: Model terms
predict(norm_mod, param = "sd", type = "terms")
```

```
#-------------------------------------------------------------------------
```

---

summary.maxlogL *Summarize Maximum Likelihood Estimation*

Description

Displays maximum likelihood estimates computed with maxlogL with its standard errors, AIC and BIC. This is a summary method for maxlogL object.

Usage

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

Arguments

object	an object of maxlogL class which summary is desired.
...	additional arguments affecting the summary produced.

Details

This summary method computes and displays AIC, BIC, estimates and standard errors from a estimated model stored i a maxlogL class object. It also displays and computes Z-score and p values of significance test of parameters.

Value

A list with information that summarize results of a maxlogL class object.

Author(s)

Jaime Mosquera Gutiérrez, <jmosquerag@unal.edu.co>

See Also

maxlogL, maxlogLreg, bootstrap_maxlogL

Examples

```
library(EstimationTools)

#---------------------------------------------------------------------------
### First example: One known parameter

x <- rnorm(n = 10000, mean = 160, sd = 6)
theta_1 <- maxlogL(x = x, dist = 'dnorm', control = list(trace = 1),
                   link = list(over = "sd", fun = "log_link"),
```

```
                 fixed = list(mean = 160))
summary(theta_1)


#-------------------------------------------------------------------------------
# Second example: Binomial probability parameter estimation with variable
# creation

N <- rbinom(n = 100, size = 10, prob = 0.3)
phat <- maxlogL(x = N, dist = 'dbinom', fixed = list(size = 10),
                link = list(over = "prob", fun = "logit_link"))

## Standard error calculation method
print(phat$outputs$StdE_Method)

## 'summary' method
summary(phat)

#-------------------------------------------------------------------------------
# Third example: Binomial probability parameter estimation with no variable
# creation

N <- rbinom(n = 100, size = 10, prob = 0.3)
summary(maxlogL(x = N, dist = 'dbinom', fixed = list(size = 10),
                link = list(over = "prob", fun = "logit_link")))

#-------------------------------------------------------------------------------
# Fourth example: Estimation in a regression model with simulated normal data
n <- 1000
x <- runif(n = n, -5, 6)
y <- rnorm(n = n, mean = -2 + 3 * x, sd = exp(1 + 0.3* x))
norm_data <- data.frame(y = y, x = x)
formulas <- list(sd.fo = ~ x, mean.fo = ~ x)

norm_mod <- maxlogLreg(formulas, y_dist = y ~ dnorm, data = norm_data,
                       link = list(over = "sd", fun = "log_link"))

## 'summary' method
summary(norm_mod)


#-------------------------------------------------------------------------------
```

`TTTE_Analytical` *Empirical Total Time on Test (TTT), analytic version.*

Description

This function allows to compute the TTT curve from a formula containing a factor type variable (classification variable).

Usage

```
TTTE_Analytical(
  formula,
  response = NULL,
  scaled = TRUE,
  data,
  method = c("Barlow", "censored"),
  partition_method = NULL,
  silent = FALSE,
  ...
)
```

Arguments

formula	an object of class formula with the response on the left of an operator ~. The right side can be a factor variable as term or an 1 if a classification by factor levels is not desired.
response	an optional numeric vector with data of the response variable. Using this argument is equivalent to define a formula with the right side such as ~ 1. See the fourth example below.
scaled	logical. If TRUE (default value), scaled TTT is computed.
data	an optional data frame containing the variables (response and the factor, if it is desired). If data is not specified, the variables are taken from the environment from which TTT_analytical is called.
method	a character specifying the method of computation. There are two options available: 'Barlow' and 'censored'. Further information can be found in the **Details** section.
partition_method	a list specifying cluster formation when the covariate in formula is numeric, or when the data has several covariates. 'quantile-based' method is the only one currently available (See the last example).
silent	logical. If TRUE, warnings of TTTE_Analytical are suppressed.
...	further arguments passing to survfit.

Details

When method argument is set as 'Barlow', this function uses the original expression of empirical TTT presented by Barlow (1979) and used by Aarset (1987):

$$\phi_n\left(\frac{r}{n}\right) = \frac{\left(\sum_{i=1}^{r} T_{(i)}\right) + (n-r)T_{(r)}}{\sum_{i=1}^{n} T_i}$$

where $T_{(r)}$ is the r^{th} order statistic, with $r = 1, 2, \ldots, n$, and n is the sample size. On the other hand, the option *'censored'* is an implementation based on integrals presented in Westberg and KlefsjÃ¶ (1994), and using survfit to compute the Kaplan-Meier estimator:

$$\phi_n\left(\frac{r}{n}\right) = \sum_{j=1}^{r}\left[\prod_{i=1}^{j}\left(1 - \frac{d_i}{n_i}\right)\right]\left(T_{(j)} - T_{(j-1)}\right)$$

Value

A list with class object `Empirical.TTT` containing a list with the following information:

`i/n‘`	A matrix containing the empirical quantiles. This matrix has the number of columns equals to the number of levels of the factor considered (number of strata).
`phi_n`	A matrix containing the values of empirical TTT. his matrix has the number of columns equals to the number of levels of the factor considered (number of strata).
`strata`	A numeric named vector storing the number of observations per strata, and the name of each strata (names of the levels of the factor).

Author(s)

Jaime Mosquera Gutiérrez, <jmosquerag@unal.edu.co>

References

Barlow RE (1979). "Geometry of the total time on test transform." *Naval Research Logistics Quarterly*, **26**(3), 393–402. ISSN 00281441, doi: 10.1002/nav.3800260303, http://doi.wiley.com/10.1002/nav.3800260303.

Aarset MV (1987). "How to Identify a Bathtub Hazard Rate." *IEEE Transactions on Reliability*, **R-36**(1), 106–108. ISSN 15581721, doi: 10.1109/TR.1987.5222310, https://doi.org/10.1109/TR.1987.5222310.

KlefsjÃ¶ B (1991). "TTT-plotting - a tool for both theoretical and practical problems." *Journal of Statistical Planning and Inference*, **29**(1-2), 99–110. ISSN 03783758, doi: 10.1016/0378-3758(92)90125C, https://linkinghub.elsevier.com/retrieve/pii/037837589290125C.

Westberg U, KlefsjÃ¶ B (1994). "TTT-plotting for censored data based on the piecewise exponential estimator." *International Journal of Reliability, Quality and Safety Engineering*, **01**(01), 1–13. ISSN 0218-5393, doi: 10.1142/S0218539394000027, https://www.worldscientific.com/doi/abs/10.1142/S0218539394000027.

See Also

`plot.EmpiricalTTT`

Examples

```
library(EstimationTools)

#---------------------------------------------------------------------------
# Example 1: Scaled empirical TTT from 'mgus1' data from 'survival' package.

TTT_1 <- TTTE_Analytical(Surv(stop, event == 'pcm') ~1, method = 'cens',
                         data = mgus1, subset=(start == 0))
head(TTT_1$`i/n`)
head(TTT_1$phi_n)
print(TTT_1$strata)
```

```
#-------------------------------------------------------------------------------
# Example 2: Scaled empirical TTT using a factor variable with 'aml' data
# from 'survival' package.

TTT_2 <- TTTE_Analytical(Surv(time, status) ~ x, method = "cens", data = aml)
head(TTT_2$`i/n`)
head(TTT_2$phi_n)
print(TTT_2$strata)

#-------------------------------------------------------------------------------
# Example 3: Non-scaled empirical TTT without a factor (arbitrarily simulated
# data).

set.seed(911211)
y <- rweibull(n=20, shape=1, scale=pi)
TTT_3 <- TTTE_Analytical(y ~ 1, scaled = FALSE)
head(TTT_3$`i/n`)
head(TTT_3$phi_n)
print(TTT_3$strata)


#-------------------------------------------------------------------------------
# Example 4: non-scaled empirical TTT without a factor (arbitrarily simulated
# data) using the 'response' argument (this is equivalent to Third example).

set.seed(911211)
y <- rweibull(n=20, shape=1, scale=pi)
TTT_4 <- TTTE_Analytical(response = y, scaled = FALSE)
head(TTT_4$`i/n`)
head(TTT_4$phi_n)
print(TTT_4$strata)

#-------------------------------------------------------------------------------
# Eample 5: empirical TTT with a continuously variant term for the shape
# parameter in Weibull distribution.

x <- runif(50, 0, 10)
shape <- 0.1 + 0.1*x
y <- rweibull(n = 50, shape = shape, scale = pi)

partitions <- list(method='quantile-based',
                   folds=5)
TTT_5 <- TTTE_Analytical(y ~ x, partition_method = partitions)
head(TTT_5$`i/n`)
head(TTT_5$phi_n)
print(TTT_5$strata)
plot(TTT_5) # Observe changes in Empirical TTT

#-------------------------------------------------------------------------------
```

TTT_hazard_shape *Hazard Shape estimation from TTT plot*

Description

This function can be used so as to estimate hazard shape corresponding to a given data set.

Usage

```
TTT_hazard_shape(
  formula,
  data = NULL,
  local_reg = loess.options(),
  interpolation = interp.options(),
  silent = FALSE,
  ...
)
```

Arguments

formula	an object of class formula with the response on the left of an operator ~. The right side must be 1.
data	an optional data frame containing the response variables. If data is not specified, the variables are taken from the environment from which TTT_hazard_shape is called.
local_reg	a list of control parameters for LOESS. See loess.options.
interpolation	a list of control parameters for interpolation function. See interp.options.
silent	logical. If TRUE, warnings of TTT_hazard_shape are suppressed.
...	further arguments passed to TTTE_Analytical.

Details

This function performs a non-parametric estimation of the empirical total time on test (TTT) plot. Then, this estimated curve can be used so as to get suggestions about initial values and the search region for parameters based on hazard shape associated to the shape of empirical TTT plot.

Use Hazard_Shape function to get the results for shape estimation.

Author(s)

Jaime Mosquera Gutiérrez <jmosquerag@unal.edu.co>

See Also

Hazard_Shape, plot.HazardShape

Examples

```
#-------------------------------------------------------------------------------
# Example 1: Increasing hazard and its corresponding TTT plot with simulated data
hweibull <- function(x, shape, scale){
  dweibull(x, shape, scale)/pweibull(x, shape, scale, lower.tail = FALSE)
  }
curve(hweibull(x, shape = 2.5, scale = pi), from = 0, to = 42,
               col = "red", ylab = "Hazard function", las = 1, lwd = 2)

y <- rweibull(n = 50, shape = 2.5, scale = pi)
my_initial_guess <- TTT_hazard_shape(formula = y ~ 1)
my_initial_guess$hazard_type


#-------------------------------------------------------------------------------
```

Index