Package 'bhm'

July 22, 2025

Type Package

Title Biomarker Threshold Models

Version 1.19

Date 2025-03-29

Author Bingshu E. Chen [aut, cre]

Maintainer Bingshu E. Chen <bingshu.chen@queensu.ca>

Depends R (>= 3.5.0), coda, ggplot2, gridExtra, lpl, MASS, survival

Imports methods

Description Contains tools to fit both predictive and prognostic biomarker effects using biomarker threshold models and continuous threshold models. Evaluate the treatment effect, biomarker effect and treatment-biomarker interaction using probability index measurement. Test for treatment-biomarker interaction using residual bootstrap method.

License GPL (>= 2)

LazyLoad yes

NeedsCompilation no

Repository CRAN

Date/Publication 2025-04-26 20:10:02 UTC

Contents

`bhm-package` *An R package for the biomarker threshold models*

Description

This package fits biomarker threshold regression models for predictive and prognostic biomarker effects with binary data and survival data with an unknown biomarker cutoff point (Chen et al, 2014)<DOI:10.1016/j.csda.2013.05.015>. Multivariable models can also be fitted for adjusted biomarker effect (Fang et al, 2017)<DOI:10.1016/j.csda.2017.02.011>. Tools such as Probability index are included to measure treatment effect, biomarker effect or treatment-biomarker interaction(Jiang et al, 2016)<DOI:10.1002/sim.6907>.

Details

"bhm" is a R package for Biomarker Threshold Models. Please use the following steps to install the most recent version of 'bhm' package:

1. First, you need to install the 'devtools' package. You can skip this step if you have 'devtools' installed in your R. Invoke R and then type

install.packages("devtools")

2. Load the devtools package.

library(devtools)

3. Install "bhm" package from github with R commond

install_github("statapps/bhm")

"bhm" uses different statistical methods to identify cut-point (thershold parameter) for the biomarker in either generalized linear models or Cox proportional hazards model.

A stable version of View the "bhm" package is also available from the Comprehensive R Archive Network (https://CRAN.R-project.org/package=bhm) and can be installed using R command

install.packages("bhm")

Author(s)

Bingshu E. Chen, Tian Fang, Jia Wang, Shuoshuo Liu

Maintainer: Bingshu E. Chen <bingshu.chen@queensu.ca>

References

Chen, B. E., Jiang, W. and Tu, D. (2014). A hierarchical Bayes model for biomarker subset effects in clinical trials. Computational Statistics and Data Analysis. vol 71, page 324-334.

Fang, T., Mackillop, W., Jiang, W., Hildesheim, A., Wacholder, S. and Chen, B. E. (2017). A Bayesian method for risk window estimatin with application to HPV vaccine trial. Computational Statistics and Data Analysis. 112, page 53-62.

Jiang, S., Chen, B. E. and Tu, D.(2016). Inference on treatment-covariate interaction based on a nonparametric measure of treatment effects and censored survival data. Statistics in Medicine. 35, 2715-2725.

Gavanji, P., Chen, B. E. and Jiang, W.(2018). Residual Bootstrap test for interactions in biomarker threshold models with survival data. Statistics in Biosciences.

Chen, B. E. and Wang, J.(2020). Joint modelling of binary response and survival for clustered data in clinical trials. Statistics in Medicine. Vol 39. 326-339.

Liu, S. S. and Chen, B. E. (2020). Continuous threshold models with two-way interactions in sur vival analysis. Canadian Journal of Statistics.

See Also

bhm, brm, coxph, glm

Examples

```
# fit = bhm(y~biomarker+treatment)
# print(summary(fit))
```

---

ars *Function to perform Adaptive Rejection Sampling*

Description

Generates a sequence of random variables using Adaptive Rejection Sampling (ARS).

Usage

```
ars(logpdf, n = 1, lower=-14, upper=15, x0 = 0, ...)
arns(logpdf, n = 1, lower = -5, upper = 5, sigma.offset = 0.05,
              fx.offset = 0.03, K = 100, verbose = FALSE, ...)
```

Arguments

n	Desired sample size.
x0	Initial point.
logpdf	Univariate log target density.
lower	lower limit of the random variable.

upper	upper limit of the random variable.
sigma.offset	offset of sigma for the normal envelope function to ensure it covers the logpdf.
fx.offset	offset of maximum for the normal envelope function to ensure it covers the logpdf.
K	number of points between lower and upper to evaluate logpdf to find the peak and bottom values.
verbose	print out the verbose, default is FALSE.
...	Parameters passed to logpdf

Details

The support of the target density must be a bounded convex set. When this is not the case, the following tricks usually work. If the support is not bounded, restrict it to a bounded set having probability practically one. A workaround, if the support is not convex, is to consider the convex set generated by the support.

Make sure the value returned by logpdf is never smaller than log(.Machine$double.xmin), to avoid divisions by zero.

Value

An n vector, whose elements are the sampled points.

Author(s)

Bingshu E. Chen <bingshu.chen@queensu.ca>

References

Gilks, W.R., Best, N.G. and Tan, K.K.C. (1995) Adaptive rejection Metropolis sampling within Gibbs sampling (Corr: 97V46 p541-542 with Neal, R.M.), *Applied Statistics* **44**:455–472.

Examples

```
#### ==> Warning: running the examples may take a few minutes! <== ####
### Univariate densities
## Normal(mean,1)
normlogdens <- function(x, mean) -(x-mean)^2/2
y <- ars(normlogdens, n = 10, mean = 3.0)
summary(y);
```

bhm *Fitting Biomarker Threshold Models*

Description

{bhm} is a R package for Biomarker Threshold Models. It uses either Hierarchical Bayes method or proflie likehood method (Chen, et al, 2014 and Tian, et al, 2017) to identify a cut-point (thershold parameter) for the biomarker in either generalized linear models or Cox proportional hazards model. The model is specified by giving a symbolic description of the linear predictor and a description of the distribution family.

Usage

```
bhm(x, ...)

## S3 method for class 'formula'
bhm(formula, family, data, control = list(...),...)

# use
#          bhm(y ~ biomarker)
#
# to fit a prognostic model with biomarker term only
#
# use
#
#          bhm(y ~ biomarker+treatment)
#
# to fit a predictive model with interaciton between biomarker
# and treatment, use
#
          bhmFit(x, y, family, control)
#
# to fit a model without the formula
#
# Biomarker shall be in the first dependent variable
#
#  To summary a "bhm" boject,
#
          ## S3 method for class 'bhm'
summary(object, ...)
```

Arguments

formula	an object of class "formula"(or one that can be coerced to that class): a symbolic description of the model to be fitted. The details of model specification are given under 'Details'.

family	a description of the response distribution and link function to be used in the model. The available family function are either "binomial" for fitting a logistic regression model or "surv" for fitting a Cox proportional hazards model
data	an optional data frame, list or environment (or object coercible by 'as.data.frame' to a data frame) containing the variables in the model. If not found in data, the variables are taken from environment(formula), typically the environment from which glm is called.
x, y	For "bhmFit", x is a design matrix of dimension n * p and y is a vector of observations of length n for "glm" models or a "Surv" survival object for "coxph" models.
control	a list of parameters for controlling the fitting process. See "bhmControl" for details
object	object returned from model fit
...	additional arguments to be passed to the low level regression fitting functions (see below)

Details

'biomarker' is a Biomarker variable. This variable is required and shall be the first dependent variable in the formula.

"interaction" is an option of fitting model with itneractin term. When interaction = TRUE, a predictive biomarker model will be fitted. When interaction = FALSE, a prognostic biomarker model will be fitted. Both Biomarker and Treatment variables are required if 'interaction' = TRUE and 'treatment' shall be the second variable in the formula.

"bhmFit" and "bhmGibbs" are the workhorse functions: they are not normally called directly but can be more efficient where the response vector, design matrix and family have already been calculated.

"x.cdf" is a function that maps biomarker values to interval (0, 1) using its empirical cumulative distribution function. After the threshold parameters are identified, the biomarker variable will be transformed back to its original scale.

Value

bhm returns an object of class inheriting from "bhm" which inherits from the class glm or 'coxph'. See later in this section.

The function "summary" (i.e., "summary.bhm") can be used to obtain or print a summary of the results, for example, the 95

An object of class "bhm" is a list containing at least the following components:

c.max	a vector of the mean estimates for the threshold parameter(s)
coefficients	a named vector of coefficients from 'bhm'
c.fit	fitted conditional regression model given c = c.max
cg	Gibbs sample for threshold parmeter c
bg	Gibbs sample for the coefficients beta

Note

The logistic regression part are based on codes wrote by Tian Fang.

Author(s)

Bingshu E. Chen (bingshu.chen@queensu.ca)

References

Chen, B. E., Jiang, W. and Tu, D. (2014). A hierarchical Bayes model for biomarker subset effects in clinical trials. Computational Statistics and Data Analysis. vol 71, page 324-334.

See Also

brm, coxph, glm, glmpLRT, mpl, pIndex, resboot, rmscb, bhmControl

Examples

```
##
## Generate a random data set
n = 300
b = c(0.5, 1, 1.5)
data = gendat.surv(n, c0 = 0.40, beta = b)
age = runif(n, 0, 1)*100
tm = data[, 1]
status = data[, 2]
trt = data[, 3]
ki67 = data[, 4]
## fit a biomarker threshold survival model with one single cut point

# fit = bhm(Surv(tm, status)~ki67+trt+age, interaction = TRUE, B=5, R=10)


## here B=5 and R=10 is used for test run. In general, B > 500 and R > 2000 is
## recommend for the analysis of biomarker variable. To fit a model with
## two cut points, use:
##
##     fit = bhm(Surv(tm, status)~bmk+trt+age, B = 500,  R = 2000, c.n = 2)
##
## To print the output, use
##
#       print(fit)
##
```

bhmControl *Auxiliary function for bhm fitting*

Description

Auxiliary function for bhm fitting. Typically only used internally by 'bhmFit', but may be used to construct a control argument to either function.

Usage

```
bhmControl(method = 'Bayes', interaction, biomarker.main, alpha,
            B, R, thin, epsilon, c.n, beta0, sigma0)
```

Arguments

method	choose either 'Bayes' for Bayes method with MCMC or 'profile' for profile likelihood method with Bootstrap. The default value is 'Bayes'
interaction	an option of fitting model with interaction term When interaction = TRUE, a predictive biomarker model will be fitted When interaction = FALSE, a prognostic biomarker model will be fitted The default value is interaction = TRUE.
biomarker.main	include biomarker main effect, default is TRUE
B	number of burn in
R	number of replications for Bayes meothd or number of Bootstrap for profile likelihood method
thin	thinning parameter for Gibbs samples, default is 2
epsilon	biomarker (transformed) step length for profile likelihood method, default is 0.01
alpha	significance level (e.g. alpha=0.05)
c.n	number of threshold (i.e. the cut point), default is 1
beta0	initial value for mean of the prior distribution of beta, default is 0
sigma0	initial value for variance of the prior distribution of beta, default is 10000

Details

Control is used in model fitting of "bhm".

Value

This function checks the internal consisitency and returns a list of value as inputed to control model fitting of "bhm".

Note

Based on code from Tian Fang.

Author(s)

Bingshu E. Chen

See Also

bhm

Examples

```
## To fit a prognostic model for biomarker with two cut-points,
## 500 burn-in samples and 10000 Gibbs samples,

ctl = bhmControl(interaction = FALSE, B = 500, R = 10000, c.n = 2)

##
## then fit the following model
##
#  fit = bhmFit(x, y, family = 'surv', control = ctl)
##
```

---

brm *Fitting Biomarker Continuous Threshold Models*

Description

{brm} is an R function for Continuous Threshold Models. It uses the maximum likehood method (Liu and Chen, 2020) to identify a cut-point (thershold parameter) for the biomarker in the Cox proportional hazards model. The model is specified by giving a symbolic description of the linear predictor and a description of the distribution family.

Usage

```
brm(x, ...)

## S3 method for class 'formula'
brm(formula, data = list(...), interaction = TRUE,
      method = c("gradient", "profile", "single"), q=1, epsilon = NULL, ...)
# use
#          brm(y ~ biomarker)
# or
#          brm(y ~ biomarker + x1 + x2, interactin = FALSE)
#
# to fit a prognostic model with biomarker term only (will be implement in the future)
#
# use
#
#          brm(y ~ biomarker+treatment+x1+x2+...)
#
```

```
# to fit a predictive model with interaciton between biomarker
# and treatment, adjusted for x1, x2, etc.
#
# use
#          brm(x, y, control, ...)
#
# to fit a model without formula
#
# Biomarker shall be in the first dependent variable
```

Arguments

formula	an object of class "formula"(or one that can be coerced to that class): a symbolic description of the model to be fitted. The details of model specification are given under 'Details'.
data	an optional data frame, list or environment (or object coercible by 'as.data.frame' to a data frame) containing the variables in the model. If not found in data, the variables are taken from environment(formula), typically the environment from which glm is called.
interaction	model with interaction term between biomarker and treatment variables. Default is TRUE.
method	method to fit a brm model, such as a gradient method or a single-index model. The default method is the "gradient" method.
q	nubmer of biomarker variables. If q > 1, a single-index model will be fitted. Default is q = 1.
x	For "brm.default", x is a design matrix of dimension n * p and y is a vector of observations of length n for a "Surv" survival object for "coxph" models.
...	additional arguments to be passed to the low level regression fitting functions (see below).
epsilon	Step width for the profile likelihood method, default is (max(w)-min(w))/20.

Details

'biomarker' is a Biomarker variable. This variable is required and shall be the first dependent variable in the formula.

"interaction" is an option of fitting model with itneractin term. When interaction = TRUE, a predictive biomarker model will be fitted. When interaction = FALSE, a prognostic biomarker model will be fitted. Both Biomarker and Treatment variables are required if 'interaction' = TRUE and 'treatment' shall be the second variable in the formula.

"brm.default" is the workhorse functions: they are not normally called directly but can be more efficient where the response vector, design matrix and family have already been calculated.

Value

brm returns an object of class inheriting from "brm" which inherits from the class glm or 'coxph'. See later in this section.

The function "summary" (i.e., "summary.brm") can be used to obtain or print a summary of the results, for example, the 95

An object of class "brm" is a list containing at least the following components:

`coefficients`	a named vector of coefficients from 'brm'
`c.max`	the maximum likelihood estimate for the threshold parameter(s).
`theta`	a vector contains both the coefficients and c.max.
`var`	the variance matrix of theta.
`loglik`	the log-likelihood with the final values of the coefficients.
`linear.predictors`	the vector of linear predictors, one per subject.
`first`	the first derivative vector at the solution.

Author(s)

Shuoshuo Liu (shuoshuo.liu@psu.edu) and Bingshu E. Chen (bingshu.chen@queensu.ca)

References

Liu, S. S. and Chen, B. E. (2020). Continuous threshold models with two-way interactions in survival analysis. Canadian Journal of Statistics. Vol. 48, page 751-772.

See Also

bhm, coxph, plot.brm, print.brm, residuals.brm, summary.brm,

Examples

```
##
## Generate a random data set
n = 100
b = c(0.5, 1, 1.5)
data = gendat.surv(n, c0 = 0.40, beta = b, type="brm")
age = runif(n, 0, 1)*100
tm = data[, 1]
status = data[, 2]
trt = data[, 3]
ki67 = data[, 4]
## fit a biomarker threshold survival model with one single cut point

   fit = brm(Surv(tm, status)~ki67+trt+age)
##
## fit a prognostic continuous threshold model with biomarker only
##
#    fit = brm(Surv(tm, status)~ki67)
##
## To print the output, use
##
#    print(fit)
##
```

data *dataset*

Description

dataset for biomarker threshold model (bhm)

Usage

```
# to generate survival data, use:

   gendat.surv(n, c0, beta, type=c("brm", "bhm"))

# to generate glm data, use:

   gendat.glm(n, c0, beta)
```

Arguments

n	sample size
c0	cut off point, for example c0 = 0.4
beta	regression coefficient, for example, beta = c(0.3, log(0.5), log(0.25))
type	type of biomarker threshold model, either bhm or brm, default is type = "brm"

Format

The format of the data set for analysis shall be a data frame with a response variable (either a Surv object for Cox model or a glm response variable object) and at least one dependent variable as the biomarker variable.

Details

data set of prostate cancer in the 'survival' package is used as an example in paper by Chen, et al. (2014).

Source

prosate dataset can be loaded with 'library(survival)'.

References

Chen, B. E., Jiang, W. and Tu, D. (2014). A hierarchical Bayes model for biomarker subset eff ects in clinical trials. Computational Statistics and Data Analysis. vol 71, page 324-334.

Examples

```
#data(data)
## maybe str(data) ; plot(data) ...
c0 = 0.4
b = c(-0.5, 1.5, 1.3)
data = gendat.surv(n=30, c0 = c0, beta = b)
```

---

ggkm *Creates a Kaplan-Meier plot with at risk tables below*

Description

Creates a Kaplan-Meier plot with at risk tables below.

Based on the original packge by Michael Way(https://github.com/michaelway/ggkm), I added some minor changes that allow users to:

1) add hazards ratio (HR) and CI for HR to the plot.

2) do the incidence plot instead of KM plot.

3) plot with black and white, sometimes required by a journal.

4) change the aspect ratio of the plot.

Usage

```
ggkm(sfit, table = FALSE, xlabs = "Time-to-event", ylabs = "Survival (%)",
    xlims = c(0, max(sfit$time)), ylims = c(0, 1), ystratalabs = names(sfit$strata),
      ystrataname = "Strata", timeby = signif(max(sfit$time)/7, 1), main = "",
      pval = FALSE, pvposition = c(0.3, 0.6), marks = TRUE, shape = 3,
      legend = TRUE, legendposition = c(0.85, 0.8), ci = FALSE, subs = NULL,
      linecols = "Set1", dashed = FALSE, aspectRatio = 0.7143, black = FALSE,
      data = NULL, HR = FALSE, incid = FALSE, pvaltxt = NULL, hrtxt = NULL, ...)
```

Arguments

sfit	a survfit object
timeby	numeric: control the granularity along the time-axis; defaults to 7 time-points. Default = signif(max(sfit$time)/7, 1)
main	plot title
pval	logical: add the pvalue to the plot?
marks	logical: should censoring marks be added?
subs	= NULL,
table	logical: Create a table graphic below the K-M plot, indicating at-risk numbers?
xlabs	x-axis label
ylabs	y-axis label
xlims	numeric: list of min and max for x-axis. Default = c(0,max(sfit$time))

ylims	numeric: list of min and max for y-axis. Default = c(0,1)
ystratalabs	character list. A list of names for each strata. Default = names(sfit$strata)
ystrataname	The legend name. Default = "Strata"
shape	what shape should the censoring marks be, default is a vertical line
legend	logical. should a legend be added to the plot?
legendposition	numeric. x, y position of the legend if plotted. Default=c(0.85,0.8)
ci	logical. Should confidence intervals be plotted. Default = FALSE
linecols	Character. Colour brewer pallettes too colour lines. Default ="Set1",
dashed	logical. Should a variety of linetypes be used to identify lines. Default = FALSE
pvposition	position for the p-value, default = c(0.3, 0.6).
pvaltxt	text for the p-value, default is NULL.
aspectRatio	add aspect ratio of the plot, default is 0.7134.
black	black and white plot, default is FALSE.
data	data set for the plot, could be useful when one need HR on the plot.
incid	plot incidence curve instead of KM curve, default is FALSE.
HR	add hazards ratio to the plot, default is FALSE.
hrtxt	text for the hazards ratio, default is NULL.
...	additional arguments to be passed to the ggkm function.

Author(s)

Michael Way(https://github.com/michaelway/ggkm), heavily modified version of a script created by Abhijit Dasgupta with contributions by Gil Tomas. https://statbandit.wordpress.com/2011/03/08/an-enhanced-kaplan-meier-plot/ Michael have packaged this function, added functions to namespace and included a range of new parameters. Bingshu Chen added more options to the plot (see above) and made minor corrections of the R code and the R docments files to pass the R CMD check.

Examples

```
 library(survival)
 data(colon)
 fit <- survfit(Surv(time,status)~rx, data=colon)
# ggkm(fit, timeby=500)
```

glmpLRT *Penalized likelihood ratio test for the generalized linear models.*

Description

{glmpLRT} is an R function for the penalized likelihood ratio test in generalized lienar models. It uses the penalized likehood method (Gavanji, Jiang and Chen, 2021) to identify a cut-point (thershold parameter) for the biomarker in the generalized linear models. The model is specified by giving a symbolic description of the linear predictor and a description of the distribution family.

Usage

```
## S3 method for class 'formula'
glmpLRT(formula, family = binomial, data=list(...), lambda = 15,
  c0 = 0.5, p1 = 1, method = c("pLRT", "Davies", "Bootstrap"), B=10, K = 50,
  epsilon = 0.025,...)
# use
#          glmpLRT(y ~ biomarker)
# or
#          glmpLRT(y ~ biomarker + x1 + x2, p1=0)
#
# to fit a prognostic model with biomarker term adjust for
# covariates x1, x2, etc.
#
# use
#
#          glmpLRT(y ~ biomarker+x1+x2+x3+x4+x5, p1=2...)
#
# to fit a predictive model with interaciton between biomarker
# and x1, x2, adjusted for x3, x4, x5, etc.
#
# use
#          glmpLRT(x, y, control, ...)
#
# to fit a model without formula
#
# Biomarker shall be in the first dependent variable
```

Arguments

formula	an object of class "formula": a sympolic description of the model to be fitted. The details of model specificatoin are given under 'Details'.
family	a description of the error distribution and the link function to be used in the glm model. (See family for details of family functions.)
data	an optional data frame, list or enviroment containing the variables in the model.

method	the method to be used to fit a glmpLRT model. The default method is the "pLRT" method for the penalized likelihood ratio test. Other options include "Davis" for the Davis test and "Bootstrap" for the bootstrap test.
lambda	the tuning parameter for the penalized likelihood function, default is lambda = 15.
c0	the cut point for the threshold function, default is c0 = 0.5. Other options include c0 = 0.25 and c0 = 0.75.
p1	the number of covariates interact with the biomarker variable, the default is p1 = 1. If p1 = 0, then fit a model with biomarker main effect, adjusted for other potential covariates.
B	the number of bootstrap sample if the Bootstrap method is used.
K	smoothing parameter for the indicator function, the large value of K the better approximation of the indicator function, default is K = 50.
epsilon	Step width for the profile likelihood method, default is epsilon = 0.025.
...	additional arguments to be passed to the low level regression fitting functions (see below).

Details

'biomarker' is a Biomarker variable. This variable is required and shall be the first dependent variable in the formula. It is not necessary to use the variable name 'biomarker', other variable name is allowed too.

'p1' controls the number of other covariates that interact with the biomarker variable.

"glmpLRT.default" is the workhorse functions: they are not normally called directly but can be more efficient where the response vector, design matrix and family have already been calculated.

"print", "plot" and "summary" methods can be applied to a fitted "glmpLRT" class to display the results.

Value

glmpLRT returns an object of class inheriting from "glmpLRT" which inherits from the class glm. See later in this section.

An object of class "glmpLRT" is a list containing at least the following components:

coefficients	a named vector of coefficients from 'glmpLRT'
c.max	the maximum likelihood estimate for the threshold parameter(s).
loglik	the log-likelihood with the final values of the coefficients.
linear.predictors	the vector of linear predictors, one per subject.
mpv	p-value for the penalized likelihood ratio test.
rpv	p-value for the Davis test.
bpv	p-value for the Bootstrap test.

Author(s)

Bingshu E. Chen (bingshu.chen@queensu.ca)

References

Gavanji, P., Jiang, W. and Chen, B. E. (2021). Penalized likelihood ratio test for a biomarker threshold effect in clinical trials based on generalized linear models. Canadian Journal of Statistics.

See Also

glmpLRT, glm, plot.glmpLRT, print.glmpLRT

Examples

```
## A simulated example
bmk = rnorm(100, 3, 0.25)
age = rnorm(100, 50, 20)
trt = rbinom(100, 1, 0.5)
lp = exp(log(0.25) + 0.1*ifelse(bmk>2.5, 1, 0) + 0.69*trt)
p = lp/(1+lp)
y = rbinom(100, 1, p)
fit = glmpLRT(y~bmk+trt+age, p1 = 0)
print(fit)
```

---

llm *Fit an L-shape linear model*

Description

Fit an L-shape linear model with the cut point estimated by the profile likelihood method.

Usage

```
## S3 method for class 'formula'
llm(formula, data=list(...), epsilon = 0.025, ...)
```

Arguments

formula	an object of class "formula": a sympolic description of the model to be fitted. The details of model specificatoin are given under 'Details'.
data	an optional data frame, list or enviroment containing the variables in the model.
epsilon	Step width for the profile likelihood method, default is 0.025
...	additional arguments to be passed to the low level regression fitting functions (see below).

Details

Define a L shape linear funcation that change slope at c0:

when x <c0, y = b1 + b2*x

when x>=c0, y = b1 + b2*x + b3*(x-c0) = (b1-b3*c0) + (b2+b3)*x

Value

llm returns an object of class inheriting from "llm" which inherits from the class glm. See later in this section.

An object of class "llm" is a list containing at least the following components:

`coefficients`	a named vector of coefficients from 'llm'
`residuals`	the residuals, that is response minus fitted values.
`fitted.values`	the fitted mean values.
`rank`	the numeric rank of the fitted linear model.
`df.residual`	the residual degrees of freedom.
`call`	the matched call.
`terms`	the 'terms' object used.
`c.max`	the maximum likelihood estimate for the threshold parameter(s).
`loglik`	the log-likelihood with the final values of the coefficients.

Author(s)

Bingshu E. Chen (bingshu.chen@queensu.ca)

References

Liu, S. S. and Chen, B. E. (2020). Continuous threshold models with two-way interactions in survival analysis. Canadian Journal of Statistics. Vol. 48, page 751-772.

See Also

brm, lm, glm

Examples

```
#### simulate the data and fit a L-shape model.
n = 50 ; p <- 2
X = matrix(rnorm(n * p), n, p) # no intercept!
w = X[, 1]; age = X[, 2]

wc = w - 0.2; sigma = 0.25
y = rnorm(n, -0.1+0.7*w-1.2*ifelse(wc>0, wc, 0), sigma)

fit=llm(y~w+age)
print(fit)
print(summary(fit))
```

```
#### to plot the L-shape function
# plot(fit)
```

---

mpl *Joint models for clustered data with binary and survival outcomes.*

Description

{mpl} is a function to fit a joint model for clustered binary and survival data using maximum penalized likelihood (MPL) method with Jackknife variance.

Usage

```
mpl(formula, ...)

## S3 method for class 'formula'
mpl(formula, formula.glm, formula.cluster, data, weights=NULL,
    subset = NULL, max.iter=100, tol = 0.005, jackknife=TRUE, ...)
#
# Use:
#
# fit = mpl(Surv(time, status)~w+z, y~x1+x2, ~cluster, data=data)
#
```

Arguments

formula	an object of class "formula"(or one that can be coerced to that class): a symbolic description of the Cox proportiobal hazards model to be fitted for survival data.
formula.glm	an object of class "formula"(or one that can be coerced to that class): a symbolic description of the generalized linear model to be fitted for binary data.
formula.cluster	an object of class "formula"(or one that can be coerced to that class): a symbolic description of the cluster variable.
data	an optional data frame, list or environment (or object coercible by 'as.data.frame' to a data frame) containing the variables in the model. If not found in data, the variables are taken from environment(formula), typically the enviro nment from which mpl is called.
weights	an optional vector of weights to be used in the fitting process. Should be 'NULL' or a numeric vector. If non-NULL, weights options in the glm model and the coxph model will be assinged with the supplied weights.
subset	only a subset of data will be used for model fitting.
max.iter	Maximum number of iterations, default is max.iter = 100
tol	Tolrance for convergence, default is tol = 0.005
jackknife	Jackknife method for variance, default is jackknife = TRUE
...	additional arguments to be passed to the low level regression fitting functions (see below).

Details

mpl(Surv(time, event)~w+z, y~x1+x2, ~cluster) will fit penalized likelihood for binary and survival data with cluster effect. Function print(x) can be used to print a summary of mpl results.

Value

mpl returns an object of class inheriting from "mpl". When jackknife = TRUE, an object of class "mpl" is a list containing the following components:

`theta`	the maximum estimate of the regression coefficients and varaince component
`OR_HR`	Odds ratios (OR) and hazard ratios (HR) for binary and survival outcomes, respectively
`ase`	Asymptotic standard error for theta, which is usually understimated
`jse`	Jackknife standard error of theta based on resampling, this is considered to be more robust

Note

Based on code from J. Wang.

Author(s)

Bingshu E. Chen (bingshu.chen@queensu.ca)

References

Chen, B. E. and Wang, J. (2020). Joint modelling of binary response and survival for clustered data in clinical trials. Statistics in Medicine. Vol 39. 326-339.

See Also

`coxph`, `glm`, `print`.

Examples

```
##
### No run
#
# fit = mpl(Surv(time, event)~trt+ki67, resp~trt+age, ~center.id)
#
```

multiRoot *m-Dimensional Root (Zero) Finding*

Description

The function multiRoot searches for root (i.e, zero) of the vector-valued function func with respect to its first argument using the Gauss-Newton algorithm.

Usage

```
multiRoot(func, theta, ..., verbose = FALSE, maxIter = 50,
       thetaUp = NULL, thetaLow = NULL, tol = .Machine$double.eps^0.25)
```

Arguments

func	a m-vector function for which the root is sought.
theta	the parameter vector first argument to func.
thetaLow	the lower bound of theta.
thetaUp	the upper bound of theta.
verbose	print out the verbose, default is FALSE.
maxIter	the maximum number of iterations, default is 20.
tol	the desired accuracy (convergence tolerance), default is .Machine$double.eps^0.25.
...	an additional named or unmaned arguments to be passed to func.

Details

The function multiRoot finds an numerical approximation to func(theta) = 0 using Newton method: theta = theta - solve(J, func(theta)) when m = p. This function can be used to solve the score function euqations for a maximum log likelihood estimate.

This function make use of numJacobian calculates an numerical approximation to the m by p first order derivative of a m-vector valued function. The parameter theta is updated by the Gauss-Newton method:

theta = theta - solve((t(J) x J), J x func(theta))

When m > p, if the nonlinear system has not solution, the method attempts to find a solution in the non-linear least squares sense (Gauss-Newton algorithm). The sum of square sum(t(U)xU), where U = func(theta), will be minimized.

Value

A list with at least four components:

root	a vector of theta that solves func(theta) = 0.
f.root	a vector of f(root) that evaluates at theta = root.
iter	number of iteratins used in the algorithm.
convergence	1 if the algorithm converges, 0 otherwise.

Author(s)

Bingshu E. Chen (bingshu.chen@queensu.ca)

References

Gauss, Carl Friedrich(1809). Theoria motus corporum coelestium in sectionibus conicis solem ambientum.

See Also

optim (which is preferred) and nlm, nlminb, numJacobian, numScore, optimize and uniroot for one-dimension optimization.

Examples

```
g = function(x, a) (c(x[1]+2*x[2]^3, x[2] - x[3]^3, a*sin(x[1]*x[2])))
theta = c(1, 2, 3)
multiRoot(g, theta, a = -3)
```

---

numHessian *Calculate Hessian or Information Matrix*

Description

Calculate a numerical approximation to the Hessian matrix of a function at a parameter value.

Usage

```
numHessian(func, theta, h = 0.0001, method=c("fast", "easy"), ...)
```

Arguments

func	a function for which the first (vector) argument is used as a parameter vector.
theta	the parameter vector first argument to func.
h	the step used in the numerical calculation.
method	one of "fast" or "easy" indicating the method to use for the approximation.
...	additional named or unmaned arguments to be passed to func.

Details

The function numHessian calculates an numerical approximation to the p by p second order derivative of a scalar real valued function with p-vector argument theta. This function can be used to check if the information matrix of a log likelihood is correct or not.

Value

An p by p matrix of the Hessian of the function calculated at the point theta. If the func is a log likelihood function, then the negative of the p by p matrix is the information matrix.

See Also

numScore

Examples

```
g = function(x, a) (x[1]+2*x[2]^3 - x[3]^3 + a*sin(x[1]*x[2]))
x0= c(1, 2, 3)
numHessian(g, theta = x0, a = 9)
numHessian(g, theta = x0, method = 'easy', a = 9)
```

---

numScore *Calculate the Score / Jacobian Function*

Description

Calculate a numerical approximation to the Score function of a function at a parameter value.

Usage

```
numScore(func, theta, h = 0.0001, ...)
numJacobian(func, theta, m, h = 0.0001, ...)
```

Arguments

func	a function for which the first (vector) argument is used as a parameter vector.
theta	the parameter vector first argument to func.
h	the step used in the numerical calculation.
m	the dimension of the function f(theta), default is 2.
...	additional named or unmaned arguments to be passed to func.

Details

The function numScore calculates an numerical approximation to the p by 1 first order derivative of a scalar real valued function with p-vector argument theta. This function can be used to check if the score function of a log likelihood is correct or not.

The function numJacobian calculates an numerical approximation to the m by p first order derivative of a m-vector real valued function with p-vector argument theta. This function can be used to find the solution of score functions for a log likelihood using the multiRoot function.

Value

An p by 1 vector of the score of the function calculated at the point theta. If the func is a log likelihood function, then the p by 1 vector is the score function.

See Also

numHessian multiRoot

Examples

```
g = function(x, a) (x[1]+2*x[2]^3 - x[3]^3 + a*sin(x[1]*x[2]))
x0 = c(1, 2, 3)
numScore(g, x0, a = -3)
```

---

pIndex *Probability Index for Survival Time Difference*

Description

{pIndex} is a function to estimate and test differce of survival time among groups. It is defined as p = Pr{T_1 < T_2 }, where T_1 is survival time for subjects in group 1 and T_2 is surval time in group 2.

Usage

```
pIndex(x, ...)

## S3 method for class 'formula'
pIndex(formula, data, control = list(...),...)
###To estimate probability index for treatment and control groups (define by trt):
#
# fit = pIndex(Surv(time, status) ~ trt)
#
###To estimate probability index difference for treatment and control
###groups (define by trt) between biomarker postive and biomarker negative
###subjects(i.e. Treatment-biomarker interaction):
#
# fit = pIndex(Surv(time, status) ~ trt+biomarker)
#
```

Arguments

formula	an object of class "formula"(or one that can be coerced to that class): a sy mbolic description of the model to be fitted. The details of model specification are given under 'Details'.
data	an optional data frame, list or environment (or object coercible by 'as.data.frame' to a data frame) containing the variables in the model. If not found in data, the variables are taken from environment(formula), typically the enviro nment from which pIndex is called.
x	Here covariate x is a design matrix of dimension n * 1 (for two sample test) or dimension n * 2 (for treatment * biomarker interaction).
control	a list of parameters for controlling the fitting process. See 'pIndexControl' for details
...	additional arguments to be passed to the low level regression fitting functions (see below).

Details

pIndex(y~x) will estimate probability index of two groups (eg. treatment vs control) define by x. pIndex(y~x1 + x2) will estimate the difference of probability index of x1 (eg. treatment vs control) between biomarker positive and biomarker negative groups (x2). Function print(x) can be used to print a summary of pIndex results.

Value

pIndex returns an object of class inheriting from "pIndex". When B > 0, an object of class "pIndex" is a list containing at least the following components:

`theta`	the estimated probability index
`theta.b`	Bootstrap or Jackknife sample of the probability index
`sd`	standard deviation of theta based on resampling
`ci`	(1-alpha) percent confidence interval based on resampling

Note

This function is part of the bhm package.

Author(s)

Bingshu E. Chen (bingshu.chen@queensu.ca)

References

Jiang, S., Chen, B. E. and Tu, D.(2016). Inference on treatment-covariate interaction based on a nonparametric measure of treatment effects and censored survival data. Statistics in Medicine. 35, 2715-2725.

See Also

bhm, pIndexControl,

Examples

```
##
## Generate a random data set
n = 50
b = c(0.5, 1, 1.5)
data = gendat.surv(n, c0 = 0.40, beta = b, type='brm')
age = runif(n, 0, 1)*100
tm = data[, 1]
status = data[, 2]
trt = data[, 3]
ki67 = data[, 4]
#
### No run
#
# fit = pIndex(Surv(tm, status) ~ trt + ki67)
#
```

pIndexControl *Auxiliary function for pIndex fitting*

Description

Auxiliary function for pIndex fitting. Typically only used internally by 'pIndexFit', but may be used to construct a control argument to either function.

Usage

```
  pIndexControl(method = c("Efron", "Elc", "Elw", "Pic"),
                model = c("default", "local", "threshold"),
       ci = c("Bootstrap", "Jackknife"), weights = NULL,
kernel = NULL, h = 0.1, w  = seq(0.05, 0.95, 0.05),
alpha = 0.05, B = 0, pct = 0.5, tau=NULL)
```

Arguments

method	choose either 'Efron' for Efron method, 'Elc' for conditional empirical likelihood, 'Elw' for weighted empirical likelihood method, and 'Pic' for piecewise exponential distribution. The default value is 'Efron'
model	'default' for default pIndex model, 'local' for kernel method, 'threshold' for threshold method
ci	Method to construct confidence interval, 'Bootstrap' for Bootstrap method and 'Jackknife' for Jackknife method
weights	case weight
kernel	kernel funtion types, including "gaussian", "epanechnikov", "rectangular", "triangular", "biweiht", "cosine", "optcosine". The default value is 'gaussian'
h	bandwidth, defaul is 0.1
w	percentile of biomarker value for local fit
B	number of Bootstrap sample
alpha	significance level (e.g. alpha=0.05)
pct	Percentile of threshold (i.e. the cut point), default is 0.5
tau	maximum time tau to be used for pIndex

Details

Control is used in model fitting of 'pIndex'.

Value

This function checks the internal consisitency and returns a list of value as inputed to control model fit of pIndex.

Note

Based on code from Bingshu E. Chen.

Author(s)

Bingshu E. Chen

See Also

bhm, pIndex

Examples

```
## To calculate the probability index for a biomarker with conditional empirical likelihood method,
## and the corresponding 90 percent CI using Bootstrap method with 10000 bootstrap sample

ctl = pIndexControl(method = 'Elc', ci = 'Bootstrap', B = 10000, alpha = 0.1)

##
## then fit the following model
##
#  fit = pIndex(y~x1 + x2, family = 'surv', control = ctl)
##
```

---

plot *Plot a fitted biomarker threhold model*

Description

Several different type of plots can be produced for biomarker threshold mdels. Plot method is used to provide a summary of outputs from "bhm", "pIndex", "resboot".

Use "methods(plot)" and the documentation for these for other plot methods.

Usage

```
## S3 method for class 'bhm'
plot(x, type = c("profile", "density"), ...)
## S3 method for class 'brm'
plot(x, type = c("HR"), ...)
## S3 method for class 'pIndex'
plot(x, ...)
## S3 method for class 'resboot'
plot(x, ...)
## S3 method for class 'residuals.brm'
plot(x, type="Martingale", ...)
```

Arguments

x	a class returned from "bhm", "pIndex" or "resboot" fit.
type	type of plot in bhm object, "profile" to plot profile likelihood, "density" to plot trace and density of the threshold distribution. "HR" to plot hazard ratio of the "brm" boject.
...	other options used in plot().

Details

plot.bhm is called to plot either the profilelihood function or the threshold density function.

plot.pIndex is called to plot local probability index (pIndex) of a continuous biomarker.

plot.resboot is called to plot the bootstrap distribution of the likelihood ratio test statistics for biomarker threshold models (resboot).

The default method, plot.default has its own help page. Use methods("plot") to get all the methods for the plot generic.

Author(s)

Bingshu E. Chen

See Also

The default method for plot plot.default. glm bhm pIndex resboot

Examples

```
#
#  plot(fit)
#
######## plot for bhm object
#
#  plot(fit, type = 'density')
#
```

---

print *print a fitted object or a summary of fitted object*

Description

print and summary are used to provide a short summary of outputs from "bhm", "brm", "mpl", "pIndex", "resboot".

Usage

```
## S3 method for class 'bhm'
print(x, ...)
## S3 method for class 'brm'
print(x, digits = 4, ...)
## S3 method for class 'mpl'
print(x, digits = 3, ...)
## S3 method for class 'pIndex'
print(x, ...)
## S3 method for class 'picreg'
print(x, digits=3, ...)
## S3 method for class 'resboot'
print(x, ...)
## S3 method for class 'summary.bhm'
print(x, ...)
```

Arguments

x	a class returned from bhm, pIndex or resboot fit
digits	number if digits to be printed
...	other options used in print()

Details

print.bhm is called to print object or summary of object from the biomarker threshold models bhm. print.pIndex is called to print object or summary of object from the probability index model pIndex. print.resboot is called to print object or summary of object from the residuall bootstrap method for biomarker threshold models resboot. summary(fit) provides detail summary of 'bhm' model fit, including parameter estimates, standard errors, and 95 percent CIs.

The default method, print.default has its own help page. Use methods("print") to get all the methods for the print generic.

Author(s)

Bingshu E. Chen

See Also

The default method for print print.default. Other methods include glm, bhm, brm, mpl, pIndex, resboot.

Examples

```
#
#  print(fit)
#
```

resboot *Rresidual Bootstrap Test (RBT) for treatment-biomarker interaction*

Description

{resboot} is a function to test the existance of treatment-biomarker interaction in biomarker threshold model

g(Y) = b0+b1*I(w>c) + b2*z + b3*I(w>c)*z.

Usage

```
resboot(x, ...)

## S3 method for class 'formula'
resboot(formula, family, data=list(...), B = 100, epsilon = 0.01, ...)
#
###To test the null hypothesis of interaction between treatment variable
###(define by z) and biomarker variables (define by w) for survival dataa,
###use:
#
# fit = resboot(Surv(time, status) ~ w + z + w:z)
#
```

Arguments

formula	an object of class "formula"(or one that can be coerced to that class): a symbolic description of the model to be fitted. The details of model specification are given under 'Details'.
family	default is family = 'Surv' for survival data.
data	an optional data frame, list or environment (or object coercible by 'as.data.frame' to a data frame) containing the variables in the model. If not found in data, the variables are taken from environment(formula), typically the enviro nment from which resboot is called.
x	Here covariate x is a design matrix of dimension n * 1 (for two sample test) or dimension n * 2 (for treatment * biomarker interaction).
B	Number of bootstraps, default is B = 100
epsilon	Biomarker (transformed) step length for profile likelihood method, default is epsilon = 0.01
...	additional arguments to be passed to the low level regression fitting functions (see below).

Details

resboot(y~w + z + w:z) will give residual bootstrap p-value for interaction between biomarker variable (w) and treatment variable (z). The null hypothesis is given by H0: b3 = 0, where b3 is the regression coefficient for the interaction term I(w>c)*z. Function print(x) can be used to print a summary of resboot results.

Value

resboot returns an object of class inheriting from "resboot". When B > 0, an object of class "resboot" is a list containing at least the following components:

`theta`	the estimated maximum of likelihood ratio statistics
`theta.b`	Bootstrap sample of theta
`sd`	standard deviation of theta based on resampling
`ci`	(1-alpha) percent confidence interval for theta based on resampling

Note

Based on code from Parisa Gavanji.

Author(s)

Bingshu E. Chen (bingshu.chen@queensu.ca)

References

Gavanji, P., Chen, B. E. and Jiang, W.(2018). Residual Bootstrap test for interactions in biomarker threshold models with survival data. Statistics in Biosciences.

See Also

`bhm` `coxph`

Examples

```
##
## Generate a random data set
n = 30
b = c(0.5, 1, 1.5)
data = gendat.surv(n, c0 = 0.40, beta = b)
tm = data[, 1]
status = data[, 2]
trt = data[, 3]
ki67 = data[, 4]
#
### No run
#
# fit = resboot(Surv(tm, status) ~ ki67+trt+ki67:trt)
#
```

rmscb *Fitting Restricted Mean Survival Time Models with a Continuous Biomarker*

Description

{rmscb} is an R function for restricted mean survival time (RMST) as a continuous function for a biomarker variables. The model is specified by giving a symbolic description of the linear predictor and a description of the distribution family.

Usage

```
  rmscb(x, ...)

  ## S3 method for class 'formula'
rmscb(formula, data, subset, na.action, tau=5, h=0.2, w0=NULL,
       sig.level = 0.95, rho = 2,...)
# use
#          rmscb(y ~ biomarker)
#
# to fit a prognostic model with biomarker term only
#
# use
#          rmscb(y ~ biomarker + trt)
#
# to fit the difference of RMSTs between two treatment groups.
#
# use
   ## Default S3 method:
rmscb(x, y, control, ...)
#
# to fit a model without formula, where the biomarker shall be in the
# first dependent variable
```

Arguments

formula	an object of class "formula"(or one that can be coerced to that class): a symbolic description of the model to be fitted. The details of model specification are given under 'Details'.
data	an optional data frame, list or environment (or object coercible by 'as.data.frame' to a data frame) containing the variables in the model. If not found in data, the variables are taken from environment(formula), typically the environment from which "rmscb" is called.
tau	a prespecified time point at which the restricted mean survival time will be calculated.
subset	an optional vector specifying a subset of observations to be used in the fitting process.

na.action	a function which indicates what should happen when the data contain NAs. The default is set by the 'na.action' setting of 'options', and is 'na.fail' if that is unset. The 'factory-fresh' default is 'na.omit'. Another possible value is 'NULL', no action. Value 'na.exclude' can be useful.
h	the bandwidth, default is h = 0.2, if h = NULL, then the bandwidth will be selected using the cross validation method.
w0	the values of biomarker at which the RMST E(T|w=w0) will be estimated.
sig.level	the significant level of the simultaneuous confidence band will be constructed.
rho	the mode for the prediction error used in the cross validation bandwidth selection.
x	for 'rmscb.default', x is a design matrix of dimension n * p
y	y is a vector of observations of length n for a "Surv" survival object.
control	a list of parameters for controlling the fitting process. See "rmsControl" for details
...	additional arguments to be passed to the low level regression fitting functions (see below).

Details

'biomarker' is a Biomarker variable. This variable is required and shall be the first dependent variable in the formula.

'rmscb.default' is the workhorse functions: they are not normally called directly but can be more efficient where the response vector, design matrix and family have already been calculated.

Value

rmscb returns an object of class inheriting from "rmscb". See later in this section.

The function "summary" (i.e., "summary.rmscb") can be used to obtain or print a summary of the results, for example, the 95 percent CI of the parameters.

An object of class "rmscb" is a list containing at least the following components:

w0	w0 from the input.
rms	a named vector of restriected mean survival time from the "rmscb".
LB	lower bound of the simultaneuous confidence band.
UB	upper bound of the simultaneuous confidence band.

Author(s)

Wen Teng, Wenyu Jiang and Bingshu E. Chen (bingshu.chen@queensu.ca)

References

Teng, W., Jiang, W. and Chen, B. E. (2022). Continuous threshold models with two-way interactions in survival analysis. Statistics in Medicine, submitted.

Examples

```
##
## Generate a random data set
n = 100
age = runif(n, 0, 1)*100
tm = rexp(n, 1/10)
status = rbinom(n, 1, 0.5)
trt= rbinom(n, 1, 0.5)

## fit a restricted mean survival time with one biomarker

   fit = rmscb(Surv(tm, status)~age)
   print(fit)
## plot(fit)
## summary(fit)
```

---

rpicexp *The Piecewise Exponential Distribution*

Description

Density, distribution function, quantile function, hazard function h(t), cumulative hazard function H(t), and random generation for the piecewise exponential distribution with rate equal to 'rate' and cut points equal to 'cuts'.

Usage

```
dpicexp(x, rate=1, cuts=c(0, 10), log = FALSE)
ppicexp(q, rate=1, cuts=c(0, 10), lower.tail = TRUE, index = NULL)
qpicexp(p, rate=1, cuts=c(0, 10), lower.tail = TRUE)
rpicexp(n, rate=1, cuts=c(0, 10))

hpicexp(x, rate, cuts, index=NULL)
Hpicexp(x, rate, cuts, index=NULL)
#
## to fit a piece exponential survival model use:
#
# picfit(y, cuts=c(0, 10))
#
```

Arguments

x, q	vector of quantiles.
p	vector of probabilities.
n	number of observations. If 'length(n) > 1', the length is taken to be the number required.
rate	vector rate parameter, defaulting to 1.

cuts	cut points, defaulting 0 to 10.
log	logical; if TRUE, probability p are given as log(p).
lower.tail	logical; if TRUE(default), probabilities are P[X <= x], otherwise, P[X>x].
index	index of x, q in the interval defined by cuts, it saves time if index is known. For example, find index by index = findInterval(x, cuts)

Details

If the rate is not specified, it assumes the default value of 1.

Value

dpicexp gives the density, ppicexp gives the distribution function, qpicexp gives the quantile function, and rpicexp generates random deviates.

The length of the result is determined by n for rpicexp.

Only the first elements of the logical arguments are used.

Note

The cumulative hazard H(t) = -log(1-F(t)) is log(1-ppicexp(t, rate, cuts)), or more efficiently call function Hpicexp(t, rate, cuts).

Author(s)

Bingshu E. Chen (bingshu.chen@queensu.ca)

References

Chen, B. E., Cook, R. J., Lawless, J. F. and Zhan, M. (2005). Statistical methods for multivariate interval-censored recurrent events. Statistics in Medicine. Vol 24, 671-691.

See Also

exp for the exponential function.

Distributions for other standard distributions, including dgamma for the gamma distribution and dweibull for the Weibull distribution.

Examples

```
##
### No run
# n = 100
# rate = c(1, 1, 0.5, 0.125)
# cuts = c(0, 1, 2.5, 5, 10)
# x = rpicexp(n, rate, cuts)
#
### compare rexp and rpicexp
#
#print(ppicexp(2.5, rate = .5))
```

```
#print(pexp(2.5, rate = 0.5))
#
#
```

Index