Package ‘psbcSpeedUp’

July 1, 2024

Title Penalized Semiparametric Bayesian Cox Models

Version 2.0.7

Date 2024-07-01

URL `https://github.com/ocbe-uio/psbcSpeedUp`

BugReports `https://github.com/ocbe-uio/psbcSpeedUp/issues`

Description Algorithms to speed up the Bayesian Lasso Cox model (Lee et al., Int J Biostat, 2011 <doi:10.2202/1557-4679.1301>) and the Bayesian Lasso Cox with mandatory variables (Zucknick et al. Biometrical J, 2015 <doi:10.1002/bimj.201400160>).

License GPL-3

Copyright The C++ files pugixml.cpp, pugixml.h and pugiconfig.h are with Copyright (C) 2006-2018 Arseny Kapoulkine and Copyright (C) 2003 Kristen Wegner.

VignetteBuilder knitr

Depends R (>= 4.0)

Encoding UTF-8

RoxygenNote 7.3.1

LinkingTo Rcpp, RcppArmadillo (>= 0.9.000)

Imports Rcpp, xml2, ggplot2, GGally, MASS, survival, riskRegression, utils, stats

Suggests knitr

LazyData true

NeedsCompilation yes

Author Zhi Zhao [aut, cre],
Manuela Zucknick [aut],
Maral Saadati [aut],
Axel Benner [aut]

Maintainer Zhi Zhao <`zhi.zhao@medisin.uio.no`>

Repository CRAN

Date/Publication 2024-07-01 09:00:02 UTC

Contents

`coef.psbcSpeedUp` *coef method for class* `psbcSpeedUp`

Description

Extract the point estimates of the regression coefficients

Usage

```
## S3 method for class 'psbcSpeedUp'
coef(object, type = "mean", ...)
```

Arguments

`object`	an object of class `psbcSpeedUp`
`type`	type of point estimates of regressions. One of `c("mean", "median")`. Default is `mean`
`...`	not used

Value

Estimated coefficients are from an object of class `psbcSpeedUp`. If the `psbcSpeedUp` specified data standardization, the fitted values are base based on standardized data.

Examples

```
# Load the example dataset
data("exampleData", package = "psbcSpeedUp")
p <- exampleData$p
q <- exampleData$q
survObj <- exampleData[1:3]

# Set hyperparameters
mypriorPara <- list(
  "groupInd" = 1:p, "eta0" = 0.02, "kappa0" = 1, "c0" = 2, "r" = 10 / 9,
  "delta" = 1e-05, "lambdaSq" = 1, "sigmaSq" = runif(1, 0.1, 10),
  "beta.prop.var" = 1, "beta.clin.var" = 1
)
```

```
# run Bayesian Lasso Cox
library("psbcSpeedUp")
set.seed(123)
fitBayesCox <- psbcSpeedUp(survObj,
  p = p, q = q, hyperpar = mypriorPara,
  nIter = 10, burnin = 0, outFilePath = tempdir()
)
coef(fitBayesCox)
```

`exampleData` *Simulated data set*

Description

Simulated data set for a quick test. The data set is a list with six components: survival times `"t"`, event status `"di"`, covariates `"x"`, number of genomics variables `"p"`, number of clinical variables `"1"` and true effects of covariates `"beta_true"`. The R code for generating the simulated data is given in the Examples paragraph.

Usage

```
exampleData
```

Format

An object of class `list` of length 6.

Examples

```
# Load the example dataset
data("exampleData", package = "psbcSpeedUp")
str(exampleData)

# ===============
# The code below is to show how to generate the dataset "exampleData.rda"
# ===============

requireNamespace("MASS", quietly = TRUE)

########################### Predefined Functions

Expo <- function(times, surv) {
  z1 <- -log(surv[1])
  t1 <- times[1]
  lambda <- z1 / (t1)
  list(rate = lambda)
```

```
}

Weibull <- function(times, surv) {
  z1 <- -log(surv[1])
  z2 <- -log(surv[2])
  t1 <- times[1]
  t2 <- times[2]
  gamma <- log(z2 / z1) / log(t2 / t1)
  lambda <- z1 / (t1^gamma)
  list(scale = lambda, shape = gamma)
}

########################### Problem Dimensions
n <- 200
p <- 30
q <- 5
s <- 10

############################ Simulate a set of n x p covariates

# effects
bg <- c(0.75, -0.75, 0.5, -0.5, 0.25, -0.25, rep(0, p - 6))
bc <- c(-1.0, 1.0, 0.3, 0, -0.3)
bX <- c(bg, bc)

# covariates
# genomic
means <- rep(0, p)
Sigma <- diag(1, p)
Xg <- MASS::mvrnorm(n, means, Sigma)
# clinical
x1 <- rbinom(n = n, size = 1, prob = 0.7)
x2 <- rbinom(n = n, size = 1, prob = 0.3)
x3 <- rnorm(n = n, mean = 0, sd = 1)
x4 <- rnorm(n = n, mean = 0, sd = 1)
x5 <- rnorm(n = n, mean = 0, sd = 1)
Xc <- cbind(x1, x2, x3, x4, x5)
# all
X <- data.frame(Xg, Xc)
names(X) <- c(paste("G", 1:p, sep = ""), paste("C", 1:q, sep = ""))
X <- scale(X)

# censoring function
# - follow-up time 36 to 72 months
# - administrative censoring: uniform data entry (cens1)
# - loss to follow-up: exponential, 20% loss in 72 months (cens2)
ACT <- 36
FUT <- 72
cens.start <- FUT
cens.end <- ACT + FUT
cens1 <- runif(n, cens.start, cens.end)
loss <- Expo(times = 72, surv = 0.8)
cens2 <- rexp(n, rate = loss$rate)
```

```
cens <- pmin(cens1, cens2)

# survival distribution (Weibull, survival probs 0.5 and 0.9 at 12 and 36 months)
h0 <- round(log(2) / 36, 2)
surv <- Weibull(times = c(12, 36), surv = c(0.9, 0.5))

dt <- (-log(runif(n)) * (1 / surv$scale) * exp(-as.matrix(X) %*% bX))^(1 / surv$shape)

# survival object
status <- ifelse(dt <= cens, 1, 0)
os <- pmin(dt, cens)

exampleData <- list("t" = os, "di" = status, "x" = X, "beta_true" = bX)
```

`plot.psbcSpeedUp` *create a plot of estimated coefficients*

Description

Plot point estimates of regression coefficients and 95% credible intervals

Usage

```
## S3 method for class 'psbcSpeedUp'
plot(x, type = "mean", interval = TRUE, ...)
```

Arguments

`x`	an object of class `psbcSpeedUp` or a matrix. If `x` is a matrix, use `psbcSpeedUp:::plot.psbcSpeedUp(x)`
`type`	type of point estimates of regression coefficients. One of `c("mean", "median")`. Default is `mean`
`interval`	logical argument to show 95% credible intervals. Default is `TRUE`
`...`	additional arguments sent to `ggplot2::geom_point()`

Value

ggplot object

Examples

```
# Load the example dataset
data("exampleData", package = "psbcSpeedUp")
p <- exampleData$p
q <- exampleData$q
survObj <- exampleData[1:3]

# Set hyperparameters
```

```
mypriorPara <- list(
  "groupInd" = 1:p, "eta0" = 0.02, "kappa0" = 1, "c0" = 2, "r" = 10 / 9,
  "delta" = 1e-05, "lambdaSq" = 1, "sigmaSq" = runif(1, 0.1, 10),
  "beta.prop.var" = 1, "beta.clin.var" = 1
)


# run Bayesian Lasso Cox
library("psbcSpeedUp")
set.seed(123)
fitBayesCox <- psbcSpeedUp(survObj,
  p = p, q = q, hyperpar = mypriorPara,
  nIter = 10, burnin = 0, outFilePath = tempdir()
)
plot(fitBayesCox, color = "blue")
```

`plotBrier` *Time-dependent Brier scores*

Description

Predict time-dependent Brier scores based on Cox regression models

Usage

```
plotBrier(object, survObj.new = NULL, method = "mean", times = NULL, ...)
```

Arguments

`object`	fitted object obtained with `psbcSpeedUp`
`survObj.new`	a list containing observed data from new subjects with components `t`, `di`, `x`. If `NULL`, the prediction is based on the training data
`method`	option to use the posterior mean (`"mean"`) of coefficients for prediction or Bayesian model averaging (`"BMA"`) for prediction
`times`	maximum time point to evaluate the prediction
`...`	not used

Details

psbcSpeedUp

Examples

```
# Load the example dataset
data("exampleData", package = "psbcSpeedUp")
p <- exampleData$p
q <- exampleData$q
survObj <- exampleData[1:3]

# Set hyperparameters
mypriorPara <- list(
  "groupInd" = 1:p, "eta0" = 0.02, "kappa0" = 1, "c0" = 2, "r" = 10 / 9,
  "delta" = 1e-05, "lambdaSq" = 1, "sigmaSq" = runif(1, 0.1, 10),
  "beta.prop.var" = 1, "beta.clin.var" = 1)


# run Bayesian Lasso Cox
library("psbcSpeedUp")
set.seed(123)
fitBayesCox <- psbcSpeedUp(survObj,
  p = p, q = q, hyperpar = mypriorPara,
  nIter = 10, burnin = 0, outFilePath = tempdir()
)
# predict survival probabilities of the train data
plotBrier(fitBayesCox, times = 80)
```

`predict.psbcSpeedUp` *Predict survival risk*

Description

Predict survival probability, (cumulative) hazard or (integrated) Brier scores based on Cox regression models

Usage

```
## S3 method for class 'psbcSpeedUp'
predict(
  object,
  survObj.new = NULL,
  type = "brier",
  method = "mean",
  times = NULL,
  ...
)
```

Arguments

`object`	fitted object obtained with `psbcSpeedUp`
`survObj.new`	a list containing observed data from new subjects with components `t`, `di`, `x`. If `NULL`, the prediction is based on the training data. If `type` is among `c("hazard", "cumhazard", "survival")`, only `survObj.new$x` is needed
`type`	option to chose for predicting survival probabilities (one of `c('hazard','cumhazard','survival')`) or brier scores (`type="brier"`)
`method`	option to use the posterior mean (`"mean"`) of coefficients for prediction or Bayesian model averaging (`"BMA"`) for prediction
`times`	time points at which to evaluate the risks. If `NULL` (default), the event/censoring times are used. If `type="brier"`, the largest one of the `times` is used
`...`	not used

Details

psbcSpeedUp

Examples

```
# Load the example dataset
data("exampleData", package = "psbcSpeedUp")
p <- exampleData$p
q <- exampleData$q
survObj <- exampleData[1:3]

# Set hyperparameters
mypriorPara <- list(
  "groupInd" = 1:p, "eta0" = 0.02, "kappa0" = 1, "c0" = 2, "r" = 10 / 9,
  "delta" = 1e-05, "lambdaSq" = 1, "sigmaSq" = runif(1, 0.1, 10),
  "beta.prop.var" = 1, "beta.clin.var" = 1)


# run Bayesian Lasso Cox
library("psbcSpeedUp")
library("survival")
set.seed(123)
fitBayesCox <- psbcSpeedUp(survObj,
  p = p, q = q, hyperpar = mypriorPara,
  nIter = 10, burnin = 0, outFilePath = tempdir()
)
# predict survival probabilities of the train data
predict(fitBayesCox)
```

`psbcSpeedUp` *Function to Fit the Bayesian Cox Lasso Model*

Description

This a speed-up and extended version of the function `psbcGL()` in the R package `psbcGrouup`

Usage

```
psbcSpeedUp(
  survObj = NULL,
  p = 0,
  q = 0,
  hyperpar = list(),
  nIter = 1,
  burnin = 0,
  thin = 1,
  rw = FALSE,
  outFilePath,
  tmpFolder = "tmp/"
)
```

Arguments

`survObj`	a list containing observed data from `n` subjects; `t`, `di`, `x`. See details for more information
`p`	number of covariates for variable selection
`q`	number of mandatory covariates
`hyperpar`	a list containing prior parameter values; among `c('groupInd', 'beta.ini', 'eta0', 'kappa0', 'c0', 'r', 'delta','lambdaSq', 'sigmaSq', 'tauSq', 's', 'h', 'beta.prop.var','beta.clin.var')`. See details for more information
`nIter`	the number of iterations of the chain
`burnin`	number of iterations to discard at the start of the chain. Default is 0
`thin`	thinning MCMC intermediate results to be stored
`rw`	when setting to "TRUE", the conventional random walk Metropolis Hastings algorithm is used. Otherwise, the mean and the variance of the proposal density is updated using the jumping rule described in Lee et al. (2011)
`outFilePath`	path to where the output files are to be written
`tmpFolder`	the path to a temporary folder where intermediate data files are stored (will be erased at the end of the chain). It is specified relative to `outFilePath`

Details

psbcSpeedUp

`t`	a vector of `n` times to the event
`di`	a vector of `n` censoring indicators for the event time (1=event occurred, 0=censored)
`x`	covariate matrix, `n` observations by `p` variables
`groupInd`	a vector of `p` group indicator for each variable
`beta.ini`	the starting values for coefficients β
`eta0`	scale parameter of gamma process prior for the cumulative baseline hazard, $\eta_0 > 0$
`kappa0`	shape parameter of gamma process prior for the cumulative baseline hazard, $\kappa_0 > 0$
`c0`	the confidence parameter of gamma process prior for the cumulative baseline hazard, $c_0 > 0$
`r`	the shape parameter of the gamma prior for λ^2
`delta`	the rate parameter of the gamma prior for λ^2
`lambdaSq`	the starting value for λ^2
`sigmaSq`	the starting value for σ^2
`tauSq`	the starting values for τ^2
`s`	the set of time partitions for specification of the cumulative baseline hazard function
`h`	the starting values for h
`beta.prop.var`	the variance of the proposal density for β in a random walk M-H sampler
`beta.clin.var`	the starting value for the variance of β

Value

An object of class `psbcSpeedUp` is saved as `obj_psbcSpeedUp.rda` in the output file, including the following components:

- input - a list of all input parameters by the user
- output - a list of the all output estimates:
  - "`beta.p`" - a matrix with MCMC intermediate estimates of the regression coefficients.
  - "`h.p`" - a matrix with MCMC intermediate estimates of the increments in the cumulative baseline hazard in each interval.
  - "`tauSq.p`" - a vector MCMC intermediate estimates of the hyperparameter "tauSq".
  - "`sigmaSq.p`" - a vector MCMC intermediate estimates of the hyperparameter "sigmaSq".
  - "`lambdaSq.p`" - a vector MCMC intermediate estimates of the hyperparameter "lambdaSq".
  - "`accept.rate`" - a vector acceptance rates of individual regression coefficients.
- call - the matched call.

References

Lee KH, Chakraborty S, and Sun J (2011). Bayesian Variable Selection in Semiparametric Proportional Hazards Model for High Dimensional Survival Data. *The International Journal of Biostatistics*, 7(1):1-32.

Zucknick M, Saadati M, and Benner A (2015). Nonidentical twins: Comparison of frequentist and Bayesian lasso for Cox models. *Biometrical Journal*, 57(6):959–81.

Examples

```
# Load the example dataset
data("exampleData", package = "psbcSpeedUp")
p <- exampleData$p
q <- exampleData$q
survObj <- exampleData[1:3]

# Set hyperparameters
mypriorPara <- list(
  "groupInd" = 1:p, "eta0" = 0.02, "kappa0" = 1, "c0" = 2, "r" = 10 / 9,
  "delta" = 1e-05, "lambdaSq" = 1, "sigmaSq" = runif(1, 0.1, 10),
  "beta.prop.var" = 1, "beta.clin.var" = 1
)


# run Bayesian Lasso Cox
library("psbcSpeedUp")
set.seed(123)
fitBayesCox <- psbcSpeedUp(survObj,
  p = p, q = q, hyperpar = mypriorPara,
  nIter = 10, burnin = 0, outFilePath = tempdir()
)
plot(fitBayesCox, color = "blue")
```

Index