Package ‘BayesSurvive’

May 6, 2026

Title Bayesian Survival Models for High-Dimensional Data

Version 0.1.0

Date 2025-03-25

Description An implementation of Bayesian survival models with graph-structured selection priors for sparse identification of omics features predictive of survival (Madjar et al., 2021 <doi:10.1186/s12859-021-04483-z>) and its extension to use a fixed graph via a Markov Random Field (MRF) prior for capturing known structure of omics features, e.g. disease-specific pathways from the Kyoto Encyclopedia of Genes and Genomes database (Hermansen et al., 2025 <doi:10.48550/arXiv.2503.13078>).

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

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

License GPL-3

VignetteBuilder knitr

Depends R (>= 4.1.0)

Encoding UTF-8

RoxygenNote 7.3.2

LinkingTo Rcpp, RcppArmadillo, testthat

Imports Rcpp, ggplot2, GGally, mvtnorm, survival, riskRegression, utils, stats, methods

Suggests knitr, testthat, Matrix

LazyData true

NeedsCompilation yes

Author Zhi Zhao [aut, cre],
Waldir Leoncio [aut],
Katrin Madjar [aut],
Tobias Østmo Hermansen [aut],
Manuela Zucknick [ctb],
Jörg Rahnenführer [ctb]

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

Repository CRAN

Date/Publication 2025-03-25 22:50:23 UTC

Contents

`BayesSurvive` *Fit Bayesian Cox Models*

Description

This is the main function to fit a Bayesian Cox model with graph-structured selection priors for sparse identification of high-dimensional covariates.

Usage

```
BayesSurvive(
  survObj,
  model.type = "Pooled",
  MRF2b = FALSE,
  MRF.G = TRUE,
  g.ini = 0,
  hyperpar = NULL,
  initial = NULL,
  nIter = 1,
  burnin = 0,
  thin = 1,
  output_graph_para = FALSE,
  verbose = TRUE,
  cpp = FALSE
)
```

Arguments

`survObj`	a list containing observed data from `n` subjects with components `t`, `di`, `X`. For graphical learning of the Markov random field prior, `survObj` should be a list of the list with survival and covariates data. For subgroup models with or without graphical learning, `survObj` should be a list of multiple lists with each component list representing each subgroup's survival and covariates data

`model.type`	a method option from `c("Pooled", "CoxBVSSL", "Sub-struct")`. To enable graphical learning for "Pooled" model, please specify `list(survObj)` where `survObj` is the list of `t`, `di` and `X`
`MRF2b`	logical value. `MRF2b = TRUE` means two different hyperparameters b in MRF prior (values b01 and b02) and `MRF2b = FALSE` means one hyperparameter b in MRF prior
`MRF.G`	logical value. `MRF.G = TRUE` is to fix the MRF graph which is provided in the argument `hyperpar`, and `MRF.G = FALSE` is to use graphical model for learning the MRF graph
`g.ini`	initial values for latent edge inclusion indicators in graph, should be a value in [0,1]. 0 or 1: set all random edges to 0 or 1; value in (0,1): rate of indicators randomly set to 1, the remaining indicators are 0
`hyperpar`	a list containing prior parameter values
`initial`	a list containing prior parameters' initial values
`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
`output_graph_para`	allow (`TRUE`) or suppress (`FALSE`) the output for parameters 'G', 'V', 'C' and 'Sig' in the graphical model if `MRF.G = FALSE`
`verbose`	logical value to display the progress of MCMC
`cpp`	logical, whether to use C++ code for faster computation

Value

An object of class `BayesSurvive` is saved as `obj_BayesSurvive.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:
  - "`gamma.p`" - a matrix with MCMC intermediate estimates of the indicator variables of regression coefficients.
  - "`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.
- call - the matched call.

Examples

```
library("BayesSurvive")
set.seed(123)

# Load the example dataset
data("simData", package = "BayesSurvive")
```

```
dataset <- list(
  "X" = simData[[1]]$X,
  "t" = simData[[1]]$time,
  "di" = simData[[1]]$status
)

# Initial value: null model without covariates
initial <- list("gamma.ini" = rep(0, ncol(dataset$X)))
# Hyperparameters
hyperparPooled <- list(
  "c0"     = 2, # prior of baseline hazard
  "tau"    = 0.0375, # sd (spike) for coefficient prior
  "cb"     = 20, # sd (spike) for coefficient prior
  "pi.ga"  = 0.02, # prior variable selection probability for standard Cox models
  "a"      = -4, # hyperparameter in MRF prior
  "b"      = 0.1, # hyperparameter in MRF prior
  "G"      = simData$G # hyperparameter in MRF prior
)


# run Bayesian Cox with graph-structured priors
fit <- BayesSurvive(
  survObj = dataset, hyperpar = hyperparPooled,
  initial = initial, nIter = 50
)

# show posterior mean of coefficients and 95% credible intervals
library("GGally")
plot(fit) +
  coord_flip() +
  theme(axis.text.x = element_text(angle = 90, size = 7))
```

`coef.BayesSurvive` *Create a dataframe of estimated coefficients*

Description

Estimate regression coefficients with posterior mean/median, credible intervals, standard deviation, or MPM estimates, posterior gammas

Usage

```
## S3 method for class 'BayesSurvive'
coef(
  object,
  MPM = FALSE,
  type = "mean",
  CI = 95,
```

```
  SD = FALSE,
  subgroup = 1,
  ...
)
```

Arguments

`object`	an object of class `BayesSurvive`
`MPM`	logical value to obtain MPM coefficients. Default: FALSE
`type`	type of point estimates of regression coefficients. One of `c("mean", "median")`. Default is `mean`
`CI`	size (level, as a percentage) of the credible interval to report. Default: 95, i.e. a 95% credible interval
`SD`	logical value to show each coefficient's standard deviation over MCMC iterations
`subgroup`	index of the subgroup for visualizing posterior coefficients
`...`	other arguments

Value

dataframe object

Examples

```
library("BayesSurvive")
set.seed(123)

# Load the example dataset
data("simData", package = "BayesSurvive")

dataset <- list(
  "X" = simData[[1]]$X,
  "t" = simData[[1]]$time,
  "di" = simData[[1]]$status
)

# Initial value: null model without covariates
initial <- list("gamma.ini" = rep(0, ncol(dataset$X)))
# Hyperparameters
hyperparPooled <- list(
  "c0"     = 2, # prior of baseline hazard
  "tau"    = 0.0375, # sd for coefficient prior
  "cb"     = 20, # sd for coefficient prior
  "pi.ga"  = 0.02, # prior variable selection probability for standard Cox models
  "a"      = -4, # hyperparameter in MRF prior
  "b"      = 0.1, # hyperparameter in MRF prior
  "G"      = simData$G # hyperparameter in MRF prior
)
```

```
# run Bayesian Cox with graph-structured priors
fit <- BayesSurvive(
  survObj = dataset, hyperpar = hyperparPooled,
  initial = initial, nIter = 50
)

# show posterior coefficients
betas <- coef(fit)
head(betas)
```

`func_MCMC` *Function to run MCMC sampling*

Description

This an internal function for MCMC sampling

Usage

```
func_MCMC(
  survObj,
  hyperpar,
  ini,
  nIter,
  thin,
  burnin,
  S,
  method,
  MRF_2b,
  MRF_G,
  output_graph_para,
  verbose,
  cpp = FALSE
)
```

Arguments

`survObj`	a list containing observed data from `n` subjects; `t`, `di`, `X`. See details for more information
`hyperpar`	a list containing prior parameter values
`ini`	a list containing prior parameters' initial values
`nIter`	the number of iterations of the chain
`thin`	thinning MCMC intermediate results to be stored
`burnin`	number of iterations to discard at the start of the chain. Default is 0

`S`	the number of subgroups
`method`	a method option from `c("Pooled", "CoxBVSSL", "Sub-struct")`
`MRF_2b`	two different b in MRF prior for subgraphs G_ss and G_rs
`MRF_G`	logical value. `MRF_G = TRUE` is to fix the MRF graph which is provided in the argument `hyperpar`, and `MRF_G = FALSE` is to use graphical model for learning the MRF graph
`output_graph_para`	allow (`TRUE`) or suppress (`FALSE`) the output for parameters 'G', 'V', 'C' and 'Sig' in the graphical model if `MRF_G = FALSE`
`verbose`	logical value to display the progress of MCMC
`cpp`	logical, whether to use C++ code for faster computation

Value

A list object saving the MCMC results with components including 'gamma.p', 'beta.p', 'h.p', 'gamma.margin', 'beta.margin', 's', 'eta0', 'kappa0', 'c0', 'pi.ga', 'tau', 'cb', 'accept.RW', 'log.jpost', 'log.like', 'post.gamma'

`func_MCMC_graph` *Function to learn MRF graph*

Description

This an internal function for MCMC sampling

Usage

```
func_MCMC_graph(sobj, hyperpar, ini, S, method, MRF_2b, cpp = FALSE)
```

Arguments

`sobj`	a list containing observed data from `n` subjects; `t`, `di`, `X`. See details for more information
`hyperpar`	a list containing prior parameter values
`ini`	a list containing prior parameters' ini values
`S`	the number of subgroups
`method`	a method option from `c("Pooled", "CoxBVSSL", "Sub-struct")`
`MRF_2b`	two different b in MRF prior for subgraphs G_ss and G_rs
`cpp`	logical, whether to use C++ code for faster computation

Value

A list object with components "Sig" the updated covariance matrices, "G.ini" the updated graph, "V.ini" the updated variances for precision matrices in all subgroups, "C.ini" the updated precision matrices omega for each subgroup

`plot.BayesSurvive` *Create a plot of estimated coefficients*

Description

Plot point estimates of regression coefficients and 95% credible intervals

Usage

```
## S3 method for class 'BayesSurvive'
plot(x, type = "mean", interval = TRUE, subgroup = 1, ...)
```

Arguments

`x`	an object of class `BayesSurvive` or a matrix. If `x` is a matrix, use `BayesSurvive:::plot.BayesSurvive(`
`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`
`subgroup`	index of the subgroup for visualizing posterior coefficients
`...`	additional arguments sent to `ggplot2::geom_point()`

Value

ggplot object

Examples

```
library("BayesSurvive")
set.seed(123)

# Load the example dataset
data("simData", package = "BayesSurvive")

dataset <- list(
  "X" = simData[[1]]$X,
  "t" = simData[[1]]$time,
  "di" = simData[[1]]$status
)

# Initial value: null model without covariates
initial <- list("gamma.ini" = rep(0, ncol(dataset$X)))
# Hyperparameters
hyperparPooled <- list(
  "c0"     = 2, # prior of baseline hazard
  "tau"    = 0.0375, # sd for coefficient prior
  "cb"     = 20, # sd for coefficient prior
  "pi.ga"  = 0.02, # prior variable selection probability for standard Cox models
  "a"      = -4, # hyperparameter in MRF prior
```

```
  "b"      = 0.1, # hyperparameter in MRF prior
  "G"      = simData$G # hyperparameter in MRF prior
)


# run Bayesian Cox with graph-structured priors
fit <- BayesSurvive(
  survObj = dataset, hyperpar = hyperparPooled,
  initial = initial, nIter = 50
)

# show posterior mean of coefficients and 95% credible intervals
library("GGally")
plot(fit) +
  coord_flip() +
  theme(axis.text.x = element_text(angle = 90, size = 7))
```

`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,
  subgroup = 1
)
```

Arguments

`object`	fitted object obtained with `BayesSurvive`
`survObj.new`	a list containing observed data from new subjects with components `t`, `di`, `X`
`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
`subgroup`	index of the subgroup in `survObj.new` for prediction. Default value is 1

Value

A `ggplot2::ggplot` object. See `?ggplot2::ggplot` for more details of the object.

Examples

```
library("BayesSurvive")
set.seed(123)

# Load the example dataset
data("simData", package = "BayesSurvive")

dataset <- list(
  "X" = simData[[1]]$X,
  "t" = simData[[1]]$time,
  "di" = simData[[1]]$status
)

# Initial value: null model without covariates
initial <- list("gamma.ini" = rep(0, ncol(dataset$X)))
# Hyperparameters
hyperparPooled <- list(
  "c0"     = 2, # prior of baseline hazard
  "tau"    = 0.0375, # sd for coefficient prior
  "cb"     = 20, # sd for coefficient prior
  "pi.ga"  = 0.02, # prior variable selection probability for standard Cox models
  "a"      = -4, # hyperparameter in MRF prior
  "b"      = 0.1, # hyperparameter in MRF prior
  "G"      = simData$G # hyperparameter in MRF prior
)


# run Bayesian Cox with graph-structured priors
fit <- BayesSurvive(
  survObj = dataset, hyperpar = hyperparPooled,
  initial = initial, nIter = 50
)
# predict survival probabilities of the train data
plotBrier(fit, survObj.new = dataset)
```

`predict.BayesSurvive` *Predict survival risk*

Description

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

Usage

```
## S3 method for class 'BayesSurvive'
predict(
```

```
  object,
  survObj.new,
  type = "brier",
  method = "mean",
  times = NULL,
  subgroup = 1,
  verbose = TRUE,
  ...
)
```

Arguments

`object`	fitted object obtained with `BayesSurvive`
`survObj.new`	a list containing observed data from new subjects with components `t`, `di`, `x`. If `type` is among `c("hazard", "cumhazard", "survival")`, only `survObj.new$X` is needed.
`type`	option to chose for predicting brier scores with `type="brier"` or one of `type=c("brier", "hazard", "cumhazard", "survival"))`
`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
`subgroup`	index of the subgroup in `survObj.new` for prediction. Default value is 1
`verbose`	logical value to print IBS of the NULL model and the Bayesian Cox model
`...`	not used

Value

A list object with 5 components if `type="brier"` including "model", "times", "Brier", "IBS" and "IPA" (Index of Prediction Accuracy), otherwise a list of 7 components with the first component as the specified argument `type` and "se", "band", "type", "diag", "baseline" and "times", see function `riskRegression::predictCox` for details

Examples

```
library("BayesSurvive")
set.seed(123)

# Load the example dataset
data("simData", package = "BayesSurvive")

dataset <- list(
  "X" = simData[[1]]$X,
  "t" = simData[[1]]$time,
  "di" = simData[[1]]$status
)

# Initial value: null model without covariates
```

```
initial <- list("gamma.ini" = rep(0, ncol(dataset$X)))
# Hyperparameters
hyperparPooled <- list(
  "c0"     = 2, # prior of baseline hazard
  "tau"    = 0.0375, # sd for coefficient prior
  "cb"     = 20, # sd for coefficient prior
  "pi.ga"  = 0.02, # prior variable selection probability for standard Cox models
  "a"      = -4, # hyperparameter in MRF prior
  "b"      = 0.1, # hyperparameter in MRF prior
  "G"      = simData$G # hyperparameter in MRF prior
)


# run Bayesian Cox with graph-structured priors
fit <- BayesSurvive(
  survObj = dataset, hyperpar = hyperparPooled,
  initial = initial, nIter = 50
)
# predict survival probabilities of the train data
predict(fit, survObj.new = dataset)
```

`simData` *Simulated survival data*

Description

Simulated data set for a quick test. The data set is a list with six components: covariates `"X"`, survival times `"time"`, event status `"status"`. The R code for generating the simulated data is given in the Examples.

Usage

```
simData
```

Format

An object of class `list` of length 3.

`UpdateGamma` *Subfunctions to update parameters*

Description

This contains subfunctions to update parameters gammas, betas, baseline hazard and graph learning parameters

Usage

```
UpdateGamma(sobj, hyperpar, ini, S, method, MRF_G, MRF_2b, cpp = FALSE)
```

Arguments

`sobj`	a list containing observed data
`hyperpar`	a list containing prior parameter values
`ini`	a list containing prior parameters' initial values
`S`	the number of subgroups
`method`	a method option from `c("Pooled", "CoxBVSSL", "Sub-struct", "Subgroup")`
`MRF_G`	logical value. `MRF_G = TRUE` is to fix the MRF graph which is provided in the argument `hyperpar`, and `MRF_G = FALSE` is to use graphical model for learning the MRF graph
`MRF_2b`	two different b in MRF prior for subgraphs G_ss and G_rs
`cpp`	logical, whether to use C++ code for faster computation

Value

A list object with two components for the latent variable selection indicators gamma with either independent Bernoulli prior

---

`UpdateRPlee11` *Update coefficients of Bayesian Cox Models*

Description

This an internal function to update coefficients of the Bayesian Cox Lasso Model

Usage

```
UpdateRPlee11(sobj, hyperpar, ini, S, method, MRF_G, cpp = FALSE)
```

Arguments

`sobj`	a list containing observed data from `n` subjects; `t`, `di`, `X`. See details for more information
`hyperpar`	a list containing prior parameter values
`ini`	a list containing prior parameters' initial values
`S`	the number of subgroups
`method`	a method option from `c("Pooled", "CoxBVSSL", "Sub-struct")`
`MRF_G`	logical value. `MRF_G = TRUE` is to fix the MRF graph which is provided in the argument `hyperpar`, and `MRF_G = FALSE` is to use graphical model for learning the MRF graph
`cpp`	logical, whether to use C++ code for faster computation

Value

A list object with component 'beta.ini' for the updated coefficients and component 'acceptlee' for the MCMC acceptance rate

---

`VS` *Function to perform variable selection*

Description

Perform variable selection using the 95 neighborhood criterion (SNC), median probability model (MPM) or Bayesian false discovery rate (FDR). Note that the Bayesian FDR only applies for each subgroup if there are subgroups.

Usage

```
VS(x, method = "FDR", threshold = NA, subgroup = 1)
```

Arguments

`x`	fitted object obtained with `BayesSurvive`, or a matrix/array, or a list consisting of matrices and arrays
`method`	variable selection method to choose from `c("CI", "SNC", "MPM", "FDR")`. Default is "FDR"
`threshold`	SNC threshold value (default 0.5) or the Bayesian expected false discovery rate threshold (default 0.05)
`subgroup`	index(es) of subgroup(s) for visualizing variable selection

Value

A boolean vector of selected (= TRUE) and rejected (= FALSE) variables for one group or a list for multiple groups

References

Lee KH, Chakraborty S, Sun J (2015). Survival prediction and variable selection with simultaneous shrinkage and grouping priors. Statistical Analysis and Data Mining, 8:114-127

Newton MA, Noueiry A, Sarkar D, Ahlquist P (2004). Detecting differential gene expression with a semiparametric hierarchical mixture method. Biostatistics, 5(2), 155-76

Examples

```
library("BayesSurvive")
set.seed(123)

# Load the example dataset
data("simData", package = "BayesSurvive")

dataset <- list(
  "X" = simData[[1]]$X,
  "t" = simData[[1]]$time,
  "di" = simData[[1]]$status
)

# Initial value: null model without covariates
initial <- list("gamma.ini" = rep(0, ncol(dataset$X)))
# Hyperparameters
hyperparPooled <- list(
  "c0"     = 2, # prior of baseline hazard
  "tau"    = 0.0375, # sd for coefficient prior
  "cb"     = 20, # sd for coefficient prior
  "pi.ga"  = 0.02, # prior variable selection probability for standard Cox models
  "a"      = -4, # hyperparameter in MRF prior
  "b"      = 0.1, # hyperparameter in MRF prior
  "G"      = simData$G # hyperparameter in MRF prior
)


# run Bayesian Cox with graph-structured priors
fit <- BayesSurvive(
  survObj = dataset, hyperpar = hyperparPooled,
  initial = initial, nIter = 50, burnin = 30
)
# show variable selection
VS(fit, method = "FDR")
```

Index