Package: intsurv (via r-universe)

October 25, 2026

Title Integrative Survival Modeling

Version 0.3.0

Description Contains implementations of the integrative Cox model with uncertain event times proposed by Wang, et al. (2020) <doi:10.1214/19-AOAS1287>, the regularized Cox cure rate model with uncertain event status proposed by Wang, et al. (2023) <doi:10.1007/s12561-023-09374-w>, and other survival analysis routines including the Cox cure rate model proposed by Kuk and Chen (1992) <doi:10.1093/biomet/79.3.531> via an EM algorithm proposed by Sy and Taylor (2000) <doi:10.1111/j.0006-341X.2000.00227.x>, the regularized Cox cure rate model with elastic net penalty following Masud et al. (2018) <doi:10.1177/0962280216677748>.

Depends R (>= 3.2.3)

Imports Rcpp (>= 0.12.0), methods, stats

Suggests tinytest

LinkingTo Rcpp, RcppArmadillo

License GPL (>= 3)

Collate 'RcppExports.R' 'class.R' 'Survi.R' 'assessment.R' 'bootSe.R' 'coef.R' 'cox_cure.R' 'cox_cure_net.R' 'iCoxph.R' 'intsurv-package.R' 'misc.R' 'prep_model.R' 'print.R' 'show.R' 'simData4cure.R' 'simData4iCoxph.R' 'summary.R'

URL https://wwenjie.org/intsurv, https://github.com/wenjie2wang/intsurv

BugReports https://github.com/wenjie2wang/intsurv/issues

Encoding UTF-8

RoxygenNote 7.3.3

NeedsCompilation yes

Author Wenjie Wang [aut, cre] (ORCID: <https://orcid.org/0000-0003-0363-3180>), Kun Chen [ths] (ORCID: <https://orcid.org/0000-0003-3579-5467>), Jun Yan [ths] (ORCID: <https://orcid.org/0000-0003-4401-7296>)

Maintainer Wenjie Wang <wang@wwenjie.org>

Repository https://cran.r-universe.dev

Date/Publication 2025-09-29 02:40:02 UTC

RemoteUrl https://github.com/cran/intsurv

RemoteRef HEAD

RemoteSha 680f61c8ba79a5e2ece1d61ce0de01d956364c41

Contents

BIC.cox_cure . . . 2
BIC.cox_cure_net . . . 3
bootSe . . . 4
cIndex . . . 6
coef,iCoxph-method . . . 7
coef.cox_cure . . . 8
coef.cox_cure_net . . . 8
cox_cure . . . 9
cox_cure_net . . . 14
iCoxph . . . 19
iCoxph-class . . . 21
iCoxph.control . . . 22
iCoxph.start . . . 23
show-method . . . 24
simData4cure . . . 25
simData4iCoxph . . . 26
summary,iCoxph-method . . . 28
summary.cox_cure . . . 29
summary.iCoxph-class . . . 30
Survi . . . 30
Survi-class . . . 31

Index **32**

`BIC.cox_cure` *Bayesian Information Criterion (BIC)*

Description

Compute Bayesian information criterion (BIC) or Schwarz's Bayesian criterion (SBC) for possibly one or several objects.

Usage

```
## S3 method for class 'cox_cure'
BIC(object, ..., method = c("obs", "effective"))

## S3 method for class 'cox_cure_uncer'
BIC(object, ..., method = c("obs", "effective"))
```

Arguments

`object`	An object for a fitted model.
`...`	Other objects of the same class.
`method`	A character string specifying the method for computing the BIC values. Notice that this argument is placed after `...` and thus must be specified as a named argument. The available options for `cox_cure` objects are `"obs"` for regular BIC based on the number of observations, and `"effective"` for using BIC based on the number of effective sample size for censored data (number of uncensored events) proposed by Volinsky and Raftery (2000). The available options for `cox_cure_mcar` objects are `"obs"` for regular BIC based on the number of observations, and `"certain-event"` for a variant of BIC based on the number of certain uncensored events. For objects of either class, the former method is used by default.

References

Volinsky, C. T., & Raftery, A. E. (2000). Bayesian information criterion for censored survival models. Biometrics, 56(1), 256–262.

Examples

```
## See examples of function 'cox_cure'.
```

`BIC.cox_cure_net` *Bayesian Information Criterion (BIC)*

Description

Compute Bayesian information criterion (BIC) or Schwarz's Bayesian criterion (SBC) from a fitted solution path.

Usage

```
## S3 method for class 'cox_cure_net'
BIC(object, ..., method = c("obs", "effective"))

## S3 method for class 'cox_cure_net_uncer'
BIC(object, ..., method = c("obs", "effective"))
```

Arguments

object	An object for a fitted solution path.
...	Other arguments for future usage. A warning message will be thrown for any invalid argument.
method	A character string specifying the method for computing the BIC values. Notice that this argument is placed after ... and thus must be specified as a named argument. The available options for cox_cure objects are "obs" for regular BIC based on the number of observations, and "effective" for using BIC based on the number of effective sample size for censored data (number of uncensored events) proposed by Volinsky and Raftery (2000). The available options for cox_cure_mcar objects are "obs" for regular BIC based on the number of observations, and "certain-event" for a variant of BIC based on the number of certain uncensored events. For objects of either class, the former method is used by default.

References

Volinsky, C. T., & Raftery, A. E. (2000). Bayesian information criterion for censored survival models. Biometrics, 56(1), 256–262.

Examples

```
## See examples of function 'cox_cure_net'.
```

---

bootSe *Standard Error Estimates through Bootstrap*

Description

For iCoxph-class object, add (or update) standard error (SE) estimates through bootstrap methods, or compute the coefficient estimates from the given number of bootstrap samples.

Usage

```
bootSe(
  object,
  B = 50,
  se = c("inter-quartile", "mad", "sd"),
  return_beta = FALSE,
  ...
)
```

Arguments

object	iCoxph-class object.
B	A positive integer specifying number of bootstrap samples used for SE estimates. A large number, such as 200, is often needed for a more reliable estimation in practice. If B = 1 is specified, the function will return the covariate coefficient estimates instead of a iCoxph-class object.
se	A character value specifying the way computing SE from bootstrap samples. The default method is based on median absolute deviation and the second method is based on inter-quartile, both of which are based on normality of the bootstrap estimates and provides robust estimates for SE. The third method estimates SE by the standard deviation of the bootstrap estimates.
return_beta	A logical value. If TRUE, the function returns the covariate coefficient estimates from the given number of bootstrap samples, which allows users to split the most computationally intensive step into small pieces that can be computed in a parallel manner. The default value is FALSE.
...	Other arguments for future usage. A warning will be thrown if any invalid argument is specified.

Details

Three different methods are available for computing SE from bootstrap samples through argument se. Given the fact that the bootstrap method is computationally intensive, the function returns the coefficient estimates in a matrix from the given number of bootstrap samples when return_beta = TRUE) is specified, which can be used in parallel computing or high performance computing (HPC) cluster. The SE estimates can be further computed based on estimates from bootstrap samples by users on their own. The return_beta = TRUE is implied, when B = 1 is specified.

Value

iCoxph-class object or a numeric matrix that contains the covariate coefficient estimates from the given number of bootstrap samples in rows.

See Also

iCoxph for fitting integrative Cox model.

Examples

```
## See examples of function 'iCoxph'.
```

cIndex *Concordance Index*

Description

Compute concordance index (C-index or C-statistic) that allows weights for right-censored survival data. For example, Asano and Hirakawa (2017) proposed cure status weighting for cure models, which reduces to Harrell's C-index if weighs are all ones.

Usage

```
cIndex(time, event = NULL, risk_score, weight = NULL)
```

Arguments

time	A numeric vector for observed times
event	A numeric vector for event indicators. If it is NULL (by default) or NA, event will be treated all as ones and the function will compute concordance index for uncensored survival data.
risk_score	A numeric vector representing the risk scores of events.
weight	A optional numeric vector for weights. If it is NULL (by default) or NA, equal weights will be used.

Details

Let r_i, t_i, and δ_i denote the risk score, observed time, and event indicator of i-th subject. The pair of (t_i, δ_i) and (t_j, δ_j), where $i < j$, are defined to be comparable if $t_i < t_j, \delta_i = 1$ or $t_i = t_j, \delta_i = 1, \delta_j = 0$. In the context of survival analysis, the risk scores of a comparable pair are said to be concordant with the survival outcomes if $r_i > r_j$. The C-index is defined as the proportion of the concordant pairs among the comparable pairs. For comparable pair satisfying $t_i < t_j, \delta_i = 1$, we count 0.5 in the numerator of the concordance index for tied risk scores ($r_i = r_j$).

Value

A named numeric vector that consists of

- index: the concordance index.
- concordant: the number of concordant pairs.
- comparable: the number of comparable pairs.
- tied_tisk: the number of comparable pairs having tied risks.

References

Asano, J., & Hirakawa, A. (2017). Assessing the prediction accuracy of a cure model for censored survival data with long-term survivors: Application to breast cancer data. Journal of Biopharmaceutical Statistics, 27(6), 918–932.

Harrell, F. E., Lee, K. L., & Mark, D. B. (1996). Multivariable prognostic models: Issues in developing models, evaluating assumptions and adequacy, and measuring and reducing errors. Statistics in medicine, 15(4), 361–387.

Examples

```
## See examples of function 'cox_cure'.
```

---

coef,iCoxph-method *Estimated Covariates Coefficients*

Description

`coef,iCoxph-method` is an S4 class method that extracts covariate coefficient estimates from `iCoxph-class` object from function `iCoxph`.

Usage

```
## S4 method for signature 'iCoxph'
coef(object, ...)
```

Arguments

`object`	`iCoxph-class` object.
`...`	Other arguments for future usage.

Value

A named numeric vector.

See Also

`iCoxph` for fitting integrative Cox model; `summary,iCoxph-method` for summary of a fitted model.

Examples

```
## See examples of function iCoxph.
```

coef.cox_cure *Estimated Covariate Coefficients*

Description

Extract the covariate coefficient estimates from a fitted Cox cure rate model with possible uncertain event status.

Usage

```
## S3 method for class 'cox_cure'
coef(object, part = c("both", "survival", "cure"), ...)

## S3 method for class 'cox_cure_uncer'
coef(object, part = c("both", "survival", "cure"), ...)
```

Arguments

`object`	Object representing a fitted model.
`part`	A character string specifying the coefficient estimates from a particular model part. The available options are `"both"` for the coefficient estimates from both model parts, `"survival"` for the coefficient estimates from the survival model part, and `"cure"` for the coefficient estimates from the cure rate model part. The default option is `"all"`.
`...`	Other arguments for future usage.

Value

If `part = "both"`, this function returns a list that consists of the following named elements

- `surv`: the coefficient estimates of survival model part.
- `cure`: the coefficient estimates of cure rate model part.

Otherwise, a named numeric vector representing the coefficient estimates of the specified model part will be returned.

coef.cox_cure_net *Estimated Covariate Coefficients*

Description

Extract the covariate coefficient estimates from a solution path of regularized Cox cure rate model.

Usage

```
## S3 method for class 'cox_cure_net'
coef(object, selection = c("bic1", "bic2", "all"), ...)

## S3 method for class 'cox_cure_net_uncer'
coef(object, selection = c("bic1", "bic2", "all"), ...)
```

Arguments

`object`	Object representing a fitted solution path.
`selection`	A character string for specifying the criterion for selection of coefficient estimates. The available options are `"bic1"` for selecting coefficient estimates by regular BIC criterion based on the number of observations, `"bic2"` for a variant BIC criterion based on the effective sample size, and `"all"` for returning the whole solution path. See `BIC.cox_cure_net` for details of selection by BIC.
`...`	Other arguments for future usage.

Value

A list that consists of the following named elements:

- `surv`: the selected coefficient estimates of survival model part.
- `cure`: the selected coefficient estimates of cure rate model part.

Examples

```
## see examples of function `cox_cure_net`
```

`cox_cure` *Cox Cure Rate Model*

Description

For right-censored data, the function `cox_cure()` trains a Cox cure rate model (Kuk and Chen, 1992; Sy and Taylor, 2000) via an expectation maximization (EM) algorithm; For right-censored data with missing/uncertain event/censoring indicators, the function fits the Cox cure rate model proposed by Wang et al. (2023).

Usage

```
cox_cure(
  surv_formula,
  cure_formula,
  time,
  event,
  data,
```

```
  subset,
  contrasts = NULL,
  bootstrap = 0L,
  surv_mstep = cox_cure.mstep(),
  cure_mstep = cox_cure.mstep(),
  control = cox_cure.control(),
  ...
)

cox_cure.fit(
  surv_x,
  cure_x,
  time,
  event,
  cure_intercept = TRUE,
  bootstrap = 0L,
  surv_mstep = cox_cure.mstep(),
  cure_mstep = cox_cure.mstep(),
  control = cox_cure.control(),
  ...
)

cox_cure.control(
  tail_completion = c("zero", "exp", "tau-zero"),
  tail_tau = Inf,
  maxit = 100,
  epsilon = 1e-04,
  pmin = 1e-05,
  save_call = TRUE,
  verbose = 0,
  ...
)

cox_cure.mstep(
  start = NULL,
  offset = NULL,
  maxit = 10,
  epsilon = 1e-04,
  standardize = TRUE,
  ...
)
```

Arguments

surv_formula	A formula object starting with ~ for the model formula in survival model part. For Cox model, no intercept term is included even if an intercept is specified or implied in the model formula. A model formula with an intercept term only is not allowed.

`cure_formula` A formula object starting with `~` for the model formula in incidence model part. For logistic model, an intercept term is included by default and can be excluded by adding `+ 0` or `- 1` to the model formula.

`time` A numeric vector for the observed survival times.

`event` A numeric vector for the event indicators, where `NA`'s are allowed and represent uncertain event indicators.

`data` An optional data frame, list, or environment that contains the model covariates and response variables (`time` and `event`) If they are not found in data, the variables are taken from the environment of the specified formula, usually the environment from which this function is called.

`subset` An optional logical vector specifying a subset of observations to be used in the fitting process.

`contrasts` An optional list, whose entries are values (numeric matrices or character strings naming functions) to be used as replacement values for the contrasts replacement function and whose names are the names of columns of data containing factors. See `contrasts.arg` of `model.matrix.default` for details.

`bootstrap` An integer representing the number of bootstrap samples for estimating standard errors of the coefficient estimates. The bootstrap procedure will not run if `bootstrap = 0` by default. If `bootstrap > 0`, the specified number of bootstrap samples will be used for estimating the standard errors.

`surv_mstep, cure_mstep`
A named list passed to `cox_cure.mstep()` specifying the control parameters for the corresponding M-steps.

`control` A `cox_cure.control` object that contains the control parameters.

`...` Other arguments passed to the control functions for backward compatibility.

`surv_x` A numeric matrix for the design matrix of the survival model component.

`cure_x` A numeric matrix for the design matrix of the cure rate model component. The design matrix should exclude an intercept term unless we want to fit a model only including the intercept term. In that case, we need further set `cure_intercept = FALSE` to not standardize the intercept term.

`cure_intercept` A logical value specifying whether to add an intercept term to the cure rate model component. If `TRUE` by default, an intercept term is included.

`tail_completion`
A character string specifying the tail completion method for conditional survival function. The available methods are `"zero"` for zero-tail completion after the largest event times (Sy and Taylor, 2000), `"exp"` for exponential-tail completion (Peng, 2003), and `"tau-zero"` for zero-tail completion after a specified `tail_tau`. The default method is the zero-tail completion proposed by Sy and Taylor (2000).

`tail_tau` A numeric number specifying the time of zero-tail completion. It will be used only if `tail_completion = "tau-zero"`. A reasonable choice must be a time point between the largest event time and the largest survival time.

`maxit` A positive integer specifying the maximum iteration number. The default value is `1000`.

epsilon	A positive number specifying the tolerance that determines the convergence of the coefficient estimates. The tolerance is compared with the relative change between estimates from two consecutive iterations, which is measured by the ratio of the L1-norm of their difference to the sum of their L1-norms plus one.
pmin	A positive number specifying the minimum value of probabilities for numerical stability. The default value is 1e-5.
save_call	A logical value indicating if the function call should be saved. For large datasets, saving the function call would increase the size of the returned object dramatically. We may want to set save_call = FALSE if the original function call is not needed.
verbose	A nonnegative integer for verbose outputs, which is mainly useful for debugging.
start	A numeric vector representing the initial values for the underlying model estimation procedure. If standardize is TRUE, the specified initial values will be scaled internally to match the standardized data. The default initial values depend on the specific models and based on the observed data. If inappropriate initial values (in terms of length) are specified, the default values will be used.
offset	A numeric vector specifying the offset term. The length of the specified offset term should be equal to the sample size.
standardize	A logical value specifying if each covariate should be standardized to have mean zero and standard deviation one internally for numerically stability and fair regularization. The default value is TRUE. The coefficient estimates will always be returned in original scales.

Value

A cox_cure object that contains the fitted ordinary Cox cure rate model if none of the event indicators is NA. For right-censored data with uncertain/missing event indicators, a cox_cure_uncer object is returned.

References

Kuk, A. Y. C., & Chen, C. (1992). A mixture model combining logistic regression with proportional hazards regression. *Biometrika*, 79(3), 531–541.

Peng, Y. (2003). Estimating baseline distribution in proportional hazards cure models. *Computational Statistics & Data Analysis*, 42(1-2), 187–201.

Sy, J. P., & Taylor, J. M. (2000). Estimation in a Cox proportional hazards cure model. *Biometrics*, 56(1), 227–236.

Wang, W., Luo, C., Aseltine, R. H., Wang, F., Yan, J., & Chen, K. (2023). Survival Modeling of Suicide Risk with Rare and Uncertain Diagnoses. *Statistics in Biosciences*, 17(1), 1–27.

Examples

```
library(intsurv)

### 1. Cox cure rate model
## simulate right-censored data with a cure fraction
```

```
set.seed(123)
n_obs <- 2e2
p <- 5
x_mat <- matrix(rnorm(n_obs * p), nrow = n_obs, ncol = p)
colnames(x_mat) <- paste0("x", seq_len(p))
cure_beta <- rep(0.5, p)
b0 <- - 1
expit <- binomial()$linkinv
ncure_prob <- expit(as.numeric(b0 + x_mat %*% cure_beta))
is_cure <- 1 - rbinom(n_obs, size = 1, prob = ncure_prob)
surv_beta <- rep(0.5, p)
risk_score <- as.numeric(x_mat %*% surv_beta)
event_time <- rexp(n_obs, exp(as.numeric(x_mat %*% surv_beta)))
censor_time <- 10
event <- ifelse(event_time < censor_time & ! is_cure, 1, 0)
obs_time <- ifelse(event > 0, event_time, censor_time)

## model-fitting for the given design matrices using cox_cure.fit()
fit1 <- cox_cure.fit(x_mat, x_mat, obs_time, event, bootstrap = 30)
summary(fit1)

## coefficient estimates from both model parts
coef(fit1)

## or a particular part
coef(fit1, "surv")
coef(fit1, "cure")


## create a toy example dataset
toy_dat <- data.frame(time = obs_time, status = event)
toy_dat$group <- cut(abs(x_mat[, 1L]), breaks = c(0, 0.5, 1, 2, Inf),
                     labels = LETTERS[1:4])
toy_dat <- cbind(toy_dat, as.data.frame(x_mat[, - 1L, drop = FALSE]))

## model-fitting for the given model formula using cox_cure()
fit2 <- cox_cure(
    ~ x3 + x4 + group,
    ~ group + x3 + offset(x2),
    time = time,
    event = status,
    data = toy_dat,
    subset = group != "D",
    bootstrap = 30
)
summary(fit2)

## get BIC's
BIC(fit1)
BIC(fit2)
BIC(fit1, fit2)

### 2. Cox cure rate model for uncertain event status
```

```
set.seed(123)
n_obs <- 200
p <- 5
x_mat <- matrix(rnorm(n_obs * p), nrow = n_obs, ncol = p)
colnames(x_mat) <- paste0("x", seq_len(p))

## simulate sample data
sim_dat <- simData4cure(nSubject = 200, max_censor = 10, lambda_censor = 0.1,
                        survMat = x_mat, cureMat = x_mat, b0 = 1)
table(sim_dat$case)
table(sim_dat$obs_event, useNA = "ifany")

## use formula
fit3 <- cox_cure(
    ~ x1 + x2 + x3,
    ~ z1 + z2 + z3,
    time = obs_time,
    event = obs_event,
    data = sim_dat
)
summary(fit3)

## use design matrix
fit4 <- cox_cure.fit(x_mat, x_mat,
                     time = sim_dat$obs_time,
                     event = sim_dat$obs_event)
summary(fit4)

## get BIC's
BIC(fit3, fit4)
```

`cox_cure_net` *Regularied Cox Cure Rate Model with Elastic-Net Penalty*

Description

For right-censored data, the function `cox_cure_net()` trains a regularized Cox cure rate model with elastic-net penalty following Masud et al. (2018), and Zou and Hastie (2005). For right-censored data with missing/uncertain event/censoring indicators, it fits the Cox cure rate model proposed by Wang et al. (2023).

Usage

```
cox_cure_net(
  surv_formula,
  cure_formula,
  time,
  event,
  data,
```

```
  subset,
  contrasts = NULL,
  cv_nfolds = 0L,
  surv_net = cox_cure_net.penalty(),
  cure_net = cox_cure_net.penalty(),
  surv_mstep = cox_cure.mstep(),
  cure_mstep = cox_cure.mstep(),
  control = cox_cure.control(),
  ...
)

cox_cure_net.fit(
  surv_x,
  cure_x,
  time,
  event,
  cure_intercept = TRUE,
  cv_nfolds = 0L,
  surv_net = cox_cure_net.penalty(),
  cure_net = cox_cure_net.penalty(),
  surv_mstep = cox_cure.mstep(),
  cure_mstep = cox_cure.mstep(),
  control = cox_cure.control(),
  ...
)

cox_cure_net.penalty(
  nlambda = 10,
  lambda_min_ratio = 0.001,
  alpha = 1,
  lambda = NULL,
  penalty_factor = NULL,
  varying_active = TRUE,
  ...
)
```

Arguments

surv_formula	A formula object starting with ~ for the model formula in survival model part. For Cox model, no intercept term is included even if an intercept is specified or implied in the model formula. A model formula with an intercept term only is not allowed.
cure_formula	A formula object starting with ~ for the model formula in incidence model part. For logistic model, an intercept term is included by default and can be excluded by adding + 0 or - 1 to the model formula.
time	A numeric vector for the observed survival times.
event	A numeric vector for the event indicators, where NA's are allowed and represent uncertain event indicators.

`data` An optional data frame, list, or environment that contains the model covariates and response variables (`time` and `event`) If they are not found in data, the variables are taken from the environment of the specified formula, usually the environment from which this function is called.

`subset` An optional logical vector specifying a subset of observations to be used in the fitting process.

`contrasts` An optional list, whose entries are values (numeric matrices or character strings naming functions) to be used as replacement values for the contrasts replacement function and whose names are the names of columns of data containing factors. See `contrasts.arg` of `model.matrix.default` for details.

`cv_nfolds` A nonnegative integer representing the number of folds in cross-validation.

`surv_net`, `cure_net` Optional lists or `cox_cure_net.penalty` objects specifying the elastic penalties for survival model part and cure rate model part, respectively.

`surv_mstep`, `cure_mstep` A named list passed to `cox_cure.mstep()` specifying the control parameters for the corresponding M-steps.

`control` A `cox_cure.control` object that contains the control parameters.

`...` Other arguments passed to the control functions for backward compatibility.

`surv_x` A numeric matrix for the design matrix of the survival model component.

`cure_x` A numeric matrix for the design matrix of the cure rate model component. The design matrix should exclude an intercept term unless we want to fit a model only including the intercept term. In that case, we need further set `cure_intercept = FALSE` to not standardize the intercept term.

`cure_intercept` A logical value specifying whether to add an intercept term to the cure rate model component. If `TRUE` by default, an intercept term is included.

`nlambda` A positive integer representing the number of lambda parameters.

`lambda_min_ratio` A positive number specifying the ratio between the smallest lambda in the solution path to the large enough lambda that would result in all zero estimates with the lasso penalty.

`alpha` A positive number between 0 and 1 representing the mixing parameter in the elastic net penalty.

`lambda` A numeric vector that consists of nonnegative values representing the sequence of the lambda parameters.

`penalty_factor` A numeric vector that consists of nonnegative penalty factors (or adaptive weights) for the L1-norm of the coefficient estimates.

`varying_active` A logical value. If `TRUE` (by default), the underlying coordinate-descent algorithm will be iterated over varying active sets, which can usually improve the computational efficiency when the number of predictors is large. Otherwise, an ordinary coordinate-descent will be performed.

Details

The model estimation procedure follows expectation maximization (EM) algorithm. Variable selection procedure through regularization by elastic net penalty is developed based on cyclic coordinate descent and majorization-minimization (MM) algorithm.

Value

A `cox_cure` or `cox_cure_net` object that contains the fitted ordinary or regularized Cox cure rate model if none of the event indicators is `NA`. For right-censored data with uncertain/missing event indicators, a `cox_cure_uncer` or `cox_cure_net_uncer` is returned.

References

Kuk, A. Y. C., & Chen, C. (1992). A mixture model combining logistic regression with proportional hazards regression. *Biometrika*, 79(3), 531–541.

Peng, Y. (2003). Estimating baseline distribution in proportional hazards cure models. *Computational Statistics & Data Analysis*, 42(1-2), 187–201.

Sy, J. P., & Taylor, J. M. (2000). Estimation in a Cox proportional hazards cure model. *Biometrics*, 56(1), 227–236.

Masud, A., Tu, W., & Yu, Z. (2018). Variable selection for mixture and promotion time cure rate models. *Statistical methods in medical research*, 27(7), 2185–2199.

Zou, H., & Hastie, T. (2005). Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society*: Series B (Statistical Methodology), 67(2), 301–320.

Wang, W., Luo, C., Aseltine, R. H., Wang, F., Yan, J., & Chen, K. (2023). Survival Modeling of Suicide Risk with Rare and Uncertain Diagnoses. *Statistics in Biosciences*, 17(1), 1–27.

Examples

```
library(intsurv)

### 1. Regularized Cox cure rate model with elastic-net penalty
## simulate a toy right-censored data with a cure fraction
set.seed(123)
n_obs <- 100
p <- 10
x_mat <- matrix(rnorm(n_obs * p), nrow = n_obs, ncol = p)
colnames(x_mat) <- paste0("x", seq_len(p))
surv_beta <- c(rep(0, p - 5), rep(1, 5))
cure_beta <- c(rep(1, 2), rep(0, p - 2))
dat <- simData4cure(nSubject = n_obs, lambda_censor = 0.01,
                    max_censor = 10, survMat = x_mat,
                    survCoef = surv_beta, cureCoef = cure_beta,
                    b0 = 0.5, p1 = 1, p2 = 1, p3 = 1)

## model-fitting for the given design matrices
fit1 <- cox_cure_net.fit(x_mat, x_mat, dat$obs_time, dat$obs_event,
                         surv_net = list(nlambda = 10, alpha = 1),
                         cure_net = list(nlambda = 10, alpha = 0.8))
```

```
## model-fitting for the given model formula
fm <- paste(paste0("x", seq_len(p)), collapse = " + ")
surv_fm <- as.formula(sprintf("~ %s", fm))
cure_fm <- surv_fm
fit2 <- cox_cure_net(surv_fm, cure_fm, data = dat,
                     time = obs_time, event = obs_event)

## summary of BIC's
BIC(fit1)
BIC(fit2)
BIC(fit1)[which.min(BIC(fit1)[, "BIC"]), ]
BIC(fit2)[which.min(BIC(fit2)[, "BIC"]), ]

## list of coefficient estimates based on BIC
coef(fit1)
coef(fit2)


### 2. regularized Cox cure model for uncertain event status
## simulate a toy data
set.seed(123)
n_obs <- 100
p <- 5
x_mat <- matrix(rnorm(n_obs * p), nrow = n_obs, ncol = p)
colnames(x_mat) <- paste0("x", seq_len(p))
surv_beta <- c(rep(0, p - 3), rep(1, 3))
cure_beta <- c(rep(1, 2), rep(0, p - 2))
dat <- simData4cure(nSubject = n_obs, lambda_censor = 0.01,
                    max_censor = 10, survMat = x_mat,
                    survCoef = surv_beta, cureCoef = cure_beta,
                    b0 = 0.5, p1 = 0.95, p2 = 0.95, p3 = 0.95)

## model-fitting from given design matrices
fit1 <- cox_cure_net.fit(
    x_mat, x_mat,
    dat$obs_time, dat$obs_event,
    surv_net = list(nlambda = 5, alpha = 0.5)
)

## model-fitting from given model formula
fm <- paste(paste0("x", seq_len(p)), collapse = " + ")
surv_fm <- as.formula(sprintf("~ %s", fm))
cure_fm <- surv_fm
fit2 <- cox_cure_net(
    surv_fm,
    cure_fm,
    data = dat,
    time = obs_time,
    event = obs_event,
    surv_net = list(nlambda = 5, alpha = 0.9),
    cure_net = list(nlambda = 5, alpha = 0.9)
)
```

```
## summary of BIC's
BIC(fit1)
BIC(fit2)

## list of coefficient estimates based on BIC
coef(fit1)
coef(fit2)
```

---

iCoxph *Integrative Cox Model for Uncertain Event Times*

Description

Fit an integrative Cox model proposed by Wang et al. (2020) for right-censored survival data with uncertain event times due to imperfect data integration.

Usage

```
iCoxph(
  formula,
  data,
  subset,
  na.action,
  contrasts = NULL,
  start = iCoxph.start(),
  control = iCoxph.control(),
  ...
)
```

Arguments

formula	Survi object specifying the covariates and response variable in the model, such as Survi(ID, time, event) ~ x1 + x2.
data	An optional data frame, list, or environment that contains the covariates and response variables included in the model. If not found in data, the variables are taken from environment(formula), usually the environment from which this function is called.
subset	An optional logical vector specifying a subset of observations to be used in the fitting process.
na.action	An optional function that indicates what should the procedure do if the data contains NAs. The default is set by the na.action setting of options. The "factory-fresh" default is na.omit. Other possible values include na.fail, na.exclude, and na.pass. help(na.fail) for details.
contrasts	An optional list, whose entries are values (numeric matrices or character strings naming functions) to be used as replacement values for the contrasts replacement function and whose names are the names of columns of data containing factors. See contrasts.arg of model.matrix.default for details.

start	A list returned by function iCoxph.start specifying starting values of the parameters to be estimated in the model. Please refer to the arguments of iCoxph.start for the available parameters.
control	A list returned by function iCoxph.control specifying control parameters for the model estimation procedure. Please refer to the arguments of iCoxph.control for the available parameters.
...	Other arguments for future usage. A warning will be thrown if any invalid argument is specified.

Value

An iCoxph-class object, whose slots include

- call: Function call.
- formula: Formula used in the model fitting.
- data: A processed data frame used for model fitting.
- nObs: Number of observation.
- estimates:
  - beta: Coefficient estimates.
  - pi: Estimated parameters in prior multinomial distribution indicating the probabilities of corresponding record being true.
  - baseline: A data frame that contains estimated baseline hazard function with columns named time and h0.
- start: The initial guesses beta_mat and pi_mat specified for the parameters to be estimated, and the set of initial guess beta_start and pi_start that resulted in the largest objective function, i.e., the observed-data likelihood function.
- control: The control list specified for model fitting.
- na.action: The procedure specified to deal with missing values in the covariate.
- xlevels: A list that records the levels in each factor variable.
- contrasts: Contrasts specified and used for each factor variable.
- convergeCode: code returned by function nlm, which is an integer indicating why the optimization process terminated. help(nlm) for details.
- logL: A numeric vector containing the observed-data log-likelihood over iterations.

References

Wang, W., Aseltine, R. H., Chen, K., & Yan, J. (2020). Integrative Survival Analysis with Uncertain Event Times in Application to A Suicide Risk Study. *Annals of Applied Statistics*, 14(1), 51–73.

See Also

iCoxph.start and iCoxph.control, respectively, for initial values and control parameters for model fitting; summary,iCoxph-method for summary of fitted model; coef,iCoxph-method for estimated covariate coefficients; bootSe for SE estimates from bootstrap methods.

Examples

```
library(intsurv)

## generate simulated survival data with uncertain records
set.seed(123)
simuDat <- simData4iCoxph(nSubject = 200)

## fit the integertive Cox model
fit <- iCoxph(
    Survi(ID, time, event) ~ x1 + x2 + x3 + x4,
    data = simuDat,
    start = iCoxph.start(methods = "nearest"),
    control = iCoxph.control(tol_beta = 1e-5)
)

## estimated covariate coefficients
coef(fit)

## get SE estimates by bootstrap
fit <- bootSe(fit, B = 30)

## summary of the fitted model
summary(fit)
```

---

iCoxph-class *An S4 Class to Represent a Fitted Integrative Cox Model*

Description

The `iCoxph` class is an S4 class that represents a fitted model. Function `iCoxph` produces objects of this class. See "Slots" for details.

Slots

`call` Function call.

`formula` Model formula.

`nObs` A positive integer.

`data` A data frame.

`estimates` A list.

`start` A list.

`control` A list.

`na.action` A length-one character vector.

`xlevels` A list.

`contrasts` A list.

`convergeCode` A non-negative integer.

`logL` A numeric value.

See Also

iCoxph for details of slots.

---

iCoxph.control *Auxiliary for Controlling iCoxph Fitting*

Description

Auxiliary function for iCoxph that enable users to specify the control parameters of the model estimation procedure. Internally, the arguments cm_gradtol, cm_stepmax, cm_steptol, and cm_max_iter are passed to function nlm as gradtol, stepmax, steptol, and iterlim, respectively.

Usage

```
iCoxph.control(
  tol_beta = 1e-06,
  tol_pi = 1e-08,
  cm_gradtol = 1e-06,
  cm_stepmax = 100,
  cm_steptol = 1e-06,
  cm_max_iter = 100,
  ecm_max_iter = 200,
  ...
)
```

Arguments

tol_beta	A positive value specifying the tolerance that concludes the convergence of the covariate coefficient estimates. The tolerance is compared with the relative change between the estimates from two consecutive iterations that is measured by ratio of the L2-norm of their difference to the sum of their L2-norm. The default value is 1e-6.
tol_pi	A positive value specifying the tolerance that concludes the convergence of the probability estimates of uncertain records being true. The tolerance is compared with the relative change between the estimates from two consecutive iterations measured by ratio of L2-norm of their difference to the L2-norm of their sum. The default value is 1e-8.
cm_gradtol	A positive scalar giving the tolerance at which the scaled gradient is considered close enough to zero to terminate CM steps. The default value is 1e-6.
cm_stepmax	A positive scalar that gives the maximum allowable scaled step length in CM steps. The default value is 1e2.
cm_steptol	A positive scalar providing the minimum allowable relative step length in CM step. The default value is 1e-6.
cm_max_iter	A positive integer specifying the maximum number of iterations to be performed before each CM step is terminated. The default value is 1e2.

ecm_max_iter	A positive integer specifying the maximum number of iterations to be performed before the ECM algorithm is terminated. The default value is 2e2.
...	Other arguments for future usage. A warning will be thrown if any invalid argument is specified.

Value

A list of class intsurv-iCoxph.control containing all specified control parameters.

See Also

iCoxph for fitting integrative Cox model.

Examples

```
## See examples of function 'iCoxph'.
```

---

iCoxph.start *Auxiliary for Starting iCoxph Fitting*

Description

Auxiliary function for iCoxph that enable users to specify the starting values of the model estimation procedure.

Usage

```
iCoxph.start(
  beta_vec = NULL,
  beta_mat = NULL,
  methods = c("nearest_hazard", "unit_hazard"),
  ...
)
```

Arguments

beta_vec	A numeric vector for starting values of coefficient estimates. The default values are the coefficient estimates from the regular Cox model only fitting on records without uncertainty. If censoring rate among subjects having unique certain records is extremely high (> 99 indicator and one predictor, the starting values will be reset to be all zeros.
beta_mat	A numeric matrix that consists of additional starting values of coefficient estimates in columns. The default value is NULL.

methods	A character vector specifying the initialization methods for probabilities of uncertain records being true. The available methods are "nearest_hazard" for initializing baseline hazard by nearest (left) neighbor, and "unit_hazard" for initializing unit baseline hazard. Partial matching on method names is supported for ease of typing. By default, both methods are used. See Wang et al. (2020) for details of the initialization methods.
...	Other arguments for future usage. A warning will be thrown if any invalid argument is specified.

Value

A list of class intsurv-iCoxph.start containing all specified starting values of the parameters to be estimated from the model.

See Also

iCoxph for fitting integrative Cox model.

Examples

```
## See examples of function 'iCoxph'.
```

---

show-method *Show Methods*

Description

S4 class methods that display or summarize certain objects.

Usage

```
## S4 method for signature 'iCoxph'
show(object)

## S4 method for signature 'summary.iCoxph'
show(object)
```

Arguments

object	An object used to dispatch a method.

Details

- For iCoxph-class object, it prints out a brief summary of the fitted model.
- For summary.iCoxph-class object, it prints out summary of a fitted model.

Value

object input (invisibly).

`simData4cure` *Simulate Data from Cox Cure Model with Uncertain Event Status*

Description

Simulate Data from Cox Cure Model with Uncertain Event Status

Usage

```
simData4cure(
  nSubject = 1000,
  shape = 2,
  scale = 0.1,
  lambda_censor = 1,
  max_censor = Inf,
  p1 = 0.9,
  p2 = 0.9,
  p3 = 0.9,
  survMat,
  cureMat = survMat,
  b0 = stats::binomial()$linkfun(0.7),
  survCoef = rep(1, ncol(survMat)),
  cureCoef = rep(1, ncol(cureMat)),
  ...
)
```

Arguments

`nSubject`	A positive integer specifying number of subjects.
`shape`	A positive number specifying the shape parameter of the distribution of the event times.
`scale`	A positive number specifying the scale parameter of the distribution of the event times.
`lambda_censor`	A positive number specifying the rate parameter of the exponential distribution for generating censoring times.
`max_censor`	A positive number specifying the largest censoring time.
`p1`	A number between 0 and 1 specifying the probability of simulating events with observed event indicators given the simulated event times.
`p2`	A number between 0 and 1 specifying the probability of simulating susceptible censoring times with observed event status given the simulated susceptible censoring times.

p3	A number between 0 and 1 specifying the probability of simulating cured censoring times with observed event status given the simulated cured censoring times.
survMat	A numeric matrix representing the design matrix of the survival model part.
cureMat	A numeric matrix representing the design matrix excluding intercept of the cure rate model part.
b0	A number representing the intercept term for the cure rate model part.
survCoef	A numeric vector for the covariate coefficients of the survival model part.
cureCoef	A numeric vector for the covariate coefficients of the cure model part.
...	Other arguments not used currently.

Value

A data.frame with the following columns:

- obs_time: Observed event/survival times.
- obs_event: Observed event status.
- event_time: Underlying true event times.
- censor_time: underlying true censoring times.
- oracle_event: underlying true event indicators.
- oracle_cure: underlying true cure indicators.
- case: underlying true case labels.

References

Wang, W., Luo, C., Aseltine, R. H., Wang, F., Yan, J., & Chen, K. (2023). Survival Modeling of Suicide Risk with Rare and Uncertain Diagnoses. *Statistics in Biosciences*, 17(1), 1–27.

Examples

```
## see examples of function cox_cure()
```

---

simData4iCoxph *Simulated Survival Data with Uncertain Records*

Description

Generate survival data with uncertain records. An integrative Cox model can be fitted for the simulated data by function iCoxph.

Usage

```
simData4iCoxph(
  nSubject = 1000,
  beta0Vec,
  xMat,
  maxNum = 2,
  nRecordProb = c(0.9, 0.1),
  matchCensor = 0.1,
  matchEvent = 0.1,
  censorMin = 0.5,
  censorMax = 12.5,
  lambda = 0.005,
  rho = 0.7,
  fakeLambda1 = lambda * exp(-3),
  fakeRho1 = rho,
  fakeLambda2 = lambda * exp(3),
  fakeRho2 = rho,
  mixture = 0.5,
  randomMiss = TRUE,
  eventOnly = FALSE,
  ...
)
```

Arguments

nSubject	Number of subjects.
beta0Vec	Time-invariant covariate coefficients.
xMat	Design matrix. By default, three continuous variables following standard normal distribution and one binary variable following Bernoulli distribution with equal probability are used.
maxNum	Maximum number of uncertain records.
nRecordProb	Probability of the number of uncertain records.
matchCensor	The matching rate for subjects actually having censoring times.
matchEvent	The matching rate for subjects actually having event times.
censorMin	The lower boundary of the uniform distribution for generating censoring time.
censorMax	The upper boundary of the uniform distribution for generating censoring time.
lambda	A positive number, scale parameter in baseline rate function for true event times.
rho	A positive number, shape parameter in baseline rate function for true event times.
fakeLambda1	A positive number, scale parameter in baseline rate function for fake event times from one distribution.
fakeRho1	A positive number, shape parameter in baseline rate function for fake event times from one distribution.
fakeLambda2	A positive number, scale parameter in baseline rate function for fake event times from another distribution.

`fakeRho2`	A positive number, shape parameter in baseline rate function for fake event times from another distribution.
`mixture`	The mixture weights, i.e., the probabilities (summing up to one) of fake event times coming from different mixture components.
`randomMiss`	A logical value specifying whether the labels of the true records are missing completely at random (MCAR) or missing not at random (MNAR). The default value is `TRUE` for MCAR.
`eventOnly`	A logical value specifying whether the uncertain records only include possible events. The default value is `FALSE`, which considers the censoring cases as the possible truth in addition to event records.
`...`	Other arguments for future usage.

Details

The event times are simulated from a Weibull proportional hazard model of given shape and baseline scale. The censoring times follow uniform distribution of specified boundaries.

Value

A data frame with the following columns,

- `ID`: subject ID
- `time`: observed event times
- `event`: event indicators
- `isTure`: latent labels indicating the true records

and the corresponding covariates.

Examples

```
## See examples of function iCoxph
```

`summary,iCoxph-method` *Summary of a Fitted Model*

Description

For `iCoxph` object, the function returns a `summary.iCoxph` object whose slots include

- `call`: Function call of model fitting.
- `coefMat`: Estimated covariate coefficients.
- `logL`: Log-likelihood under observed data.

Usage

```
## S4 method for signature 'iCoxph'
summary(object, showCall = TRUE, ...)
```

Arguments

object	iCoxph-class object.
showCall	A logic value with default TRUE, indicating whether show method prints out the original call information of iCoxph. Set FALSE for a more concise printout.
...	Other arguments for future usage.

Value

summary.iCoxph class object.

See Also

iCoxph for model fitting; coef,iCoxph-method for coefficient estimates.

Examples

```
## See examples of function iCoxph
```

---

summary.cox_cure *Summary of a Fitted Model*

Description

Summarize a fitted Cox cure rate model with possible uncertain event status.

Usage

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

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

Arguments

object	Object representing a fitted model.
...	Other arguments for future usage. A warning will be thrown if any invalid argument is specified.

summary.iCoxph-class *An S4 Class to Represent Summary of a fitted integrative Cox model*

Description

The summary.intsurv class is an S4 class that represents a summarized model. Function summary,iCoxph-method produces objects of this class. See "Slots" for details.

Slots

call Function call.

coefMat A matrix.

logL A numeric value.

See Also

summary,iCoxph-method for meaning of slots.

Survi *Formula Response for Survival Data With Uncertain Event Times*

Description

Survi returns an S4 class that represents formula response for survival data with uncertain records due to imperfect data integration. The last letter 'i' in Survi represents 'integration'.

Usage

```
Survi(ID, time, event, check = TRUE, ...)
```

Arguments

ID	Identificator of each subject.
time	Event times (whether certain or uncertain) or censoring times.
event	The status indicator, 0 = censored, 1 = event.
check	A logical value specifying whether to perform check on input data.
...	Other arguments for future usage. A warning will be thrown if any invalid argument is specified.

Value

Survi object. See Survi-class for details.

Examples

```
## See examples of function 'iCoxph'
```

`Survi-class` *An S4 Class Representing Formula Response*

Description

An S4 Class Representing Formula Response

Slots

`.Data` A numeric matrix object.

`ID` Identificator of each subject.

Index

BIC.cox_cure, 2
BIC.cox_cure_net, 3, *9*
BIC.cox_cure_net_uncer *(*BIC.cox_cure_net*)*, 3
BIC.cox_cure_uncer *(*BIC.cox_cure*)*, 2
bootSe, 4, *20*

cIndex, 6
coef,iCoxph-method, 7
coef.cox_cure, 8
coef.cox_cure_net, 8
coef.cox_cure_net_uncer *(*coef.cox_cure_net*)*, 8
coef.cox_cure_uncer *(*coef.cox_cure*)*, 8
coef.iCoxph *(*coef,iCoxph-method*)*, 7
cox_cure, 9
cox_cure_net, 14

iCoxph, *5*, *7*, 19, *21–24*, *26*, *28, 29*
iCoxph-class, 21
iCoxph.control, *20*, 22
iCoxph.start, *20*, 23

model.matrix.default, *11*, *16*, *19*

na.exclude, *19*
na.fail, *19*
na.omit, *19*
na.pass, *19*
nlm, *20*, *22*

options, *19*

show,iCoxph-method *(*show-method*)*, 24
show,summary.iCoxph-method *(*show-method*)*, 24
show-method, 24
simData4cure, 25
simData4iCoxph, 26
summary,iCoxph-method, 28
summary.cox_cure, 29
summary.cox_cure_uncer *(*summary.cox_cure*)*, 29
summary.iCoxph, *28*
summary.iCoxph *(*summary,iCoxph-method*)*, 28
summary.iCoxph-class, 30
Survi, 30
Survi-class, 31