Package 'icenReg'

May 29, 2019

Type Package

Title Regression Models for Interval Censored Data

Version 2.0.10

Date 2019-5-28

Author Clifford Anderson-Bergman

Depends survival, Rcpp, coda

Imports foreach, methods, MLEcens

LinkingTo Rcpp, RcppEigen

Maintainer Clifford Anderson-Bergman <pistacliffcho@gmail.com>

Description Regression models for interval censored data. Currently supports Cox-PH, proportional odds, and accelerated failure time models. Allows for semi and fully parametric models (parametric only for accelerated failure time models) and Bayesian parametric models. Includes functions for easy visual diagnostics of model fits and imputation of censored data.

License LGPL (>= 2.0, < 3)

RoxygenNote 6.0.1

NeedsCompilation yes

Repository CRAN

Date/Publication 2019-05-29 06:00:03 UTC

R topics documented:

`bayesControls` *Control parameters for ic_bayes*

Description

Control parameters for ic_bayes

Usage

```
bayesControls(samples = 4000, chains = 4, useMLE_start = TRUE,
  burnIn = 2000, samplesPerUpdate = 1000, initSD = 0.1,
  updateChol = TRUE, acceptRate = 0.25, thin = 5)
```

Arguments

`samples`	Number of samples.
`chains`	Number of MCMC chains to run
`useMLE_start`	Should MLE used for starting point?
`burnIn`	Number of samples discarded for burn in
`samplesPerUpdate`	Number of iterations between updates of proposal covariance matrix
`initSD`	If `useMLE_start == FALSE`, initial standard deviation used
`updateChol`	Should cholesky decomposition be updated?
`acceptRate`	Target acceptance rate
`thin`	Amount of thinning

Details

Control parameters for the MH block updater used by `ic_bayes`.

The `samples` argument dictates how many MCMC samples are taken. One sample will be saved every `thin` iterations, so there will a total of `thin * samples + burnIn` iterations. The burn in samples are not saved at all.

Default behavior is to first calculate the MLE (not the MAP) estimate and use Hessian at the MLE to seed the proposal covariance matrix. After this, an updative covariance matrix is used. In cases with weakly informative likelihoods, using the MLE startpoint may lead to overly diffuse proposal or even undefined starting values. In this case, it suggested to use a cold start by setting `useMLE_start = F` for the `controls` argument. In this case, the initial starting proposal covariance matrix will be a diagonal matrix with `initSD` standard deviations.

`cs2ic` *Convert current status data into interval censored format*

Description

Convert current status data into interval censored format

Usage

```
cs2ic(time, eventOccurred)
```

Arguments

`time`	Time of inspection
`eventOccurred`	Indicator if event has occurred. 0/FALSE implies event has not occurred by inspection time.

Details

Converts current status data to the interval censored format for usage in icenReg.

Examples

```
simData <- simCS_weib()
# Simulate current status data

head(cs2ic(simData$time, simData$event))
# Converting data to current status format

fit <- ic_par(cs2ic(time, event) ~ x1 + x2, data = simData)
# Can be used directly in formula
```

`diag_baseline` *Compare parametric baseline distributions with semi-parametric baseline*

Description

Creates plots to diagnosis fit of different choices of parametric baseline model. Plots the semi paramtric model against different choices of parametric models.

Usage

```
diag_baseline(object, data, model = "ph", weights = NULL,
  dists = c("exponential", "weibull", "gamma", "lnorm", "loglogistic",
  "generalgamma"), cols = NULL, lgdLocation = "bottomleft",
  useMidCovars = T)
```

Arguments

`object`	Either a formula or a model fit with `ic_sp` or `ic_par`
`data`	Data. Unnecessary if `object` is a fit
`model`	Type of model. Choices are `'ph'` or `'po'`
`weights`	Case weights
`dists`	Parametric baseline fits
`cols`	Colors of baseline distributions
`lgdLocation`	Where legend will be placed. See `?legend` for more details
`useMidCovars`	Should the distribution plotted be for covariates = mean values instead of 0

Details

If `useMidCovars = T`, then the survival curves plotted are for fits with the mean covariate value, rather than 0. This is because often the baseline distribution (i.e. with all covariates = 0) will be far away from the majority of the data.

Author(s)

Clifford Anderson-Bergman

Examples

```
data(IR_diabetes)
fit <- ic_par(cbind(left, right) ~ gender,
             data = IR_diabetes)

diag_baseline(fit, lgdLocation = "topright",
             dist = c("exponential", "weibull", "loglogistic"))
```

diag_covar *Evaluate covariate effect for regression model*

Description

Creates plots to diagnosis fit of covariate effect in a regression model. For a given variable, stratifies the data across different levels of the variable and adjusts for all the other covariates included in `fit` and then plots a given function to help diagnosis where covariate effect follows model assumption (i.e. either proportional hazards or proportional odds). See `details` for descriptions of the plots.

If `varName` is not provided, will attempt to figure out how to divide up each covariate and plot all of them, although this may fail.

Usage

```
diag_covar(object, varName, data, model, weights = NULL,
  yType = "meanRemovedTransform", factorSplit = TRUE, numericCuts, col,
  xlab, ylab, main, lgdLocation = NULL)
```

Arguments

`object`	Either a formula or a model fit with `ic_sp` or `ic_par`
`varName`	Covariate to split data on. If left blank, will split on each covariate
`data`	Data. Unnecessary if `object` is a fit
`model`	Type of model. Choices are `'ph'` or `'po'`
`weights`	Case weights
`yType`	Type of plot created. See details
`factorSplit`	Should covariate be split as a factor (i.e. by levels)
`numericCuts`	If `fractorSplit == FALSE`, cut points of covariate to stratify data on
`col`	Colors of each subgroup plot. If left blank, will auto pick colors
`xlab`	Label of x axis
`ylab`	Label of y axis
`main`	title of plot
`lgdLocation`	Where legend should be placed. See details

Details

For the Cox-PH and proportional odds models, there exists a transformation of survival curves such that the difference should be constant for subjects with different covariates. In the case of the Cox-PH, this is the log(-log(S(t|X))) transformation, for the proporitonal odds, this is the log(S(t|X) / (1 - S(t|X))) transformation.

The function diag_covar allows the user to easily use these transformations to diagnosis whether such a model is appropriate. In particular, it takes a single covariate and stratifies the data on different levels of that covariate. Then, it fits the semi-parametric regression model (adjusting

for all other covariates in the data set) on each of these stratum and extracts the baseline survival function. If the stratified covariate does follow the regression assumption, the difference between these transformed baseline survival functions should be approximately constant.

To help diagnosis, the default function plotted is the transformed survival functions, with the overall means subtracted off. If the assumption holds true, then the mean removed curves should be approximately parallel lines (with stochastic noise). Other choices of `yType`, the function to plot, are `"transform"`, which is the transformed functions without the means subtracted and `"survival"`, which is the baseline survival distribution is plotted for each strata.

Currently does not support stratifying covariates that are involved in an interaction term.

For variables that are factors, it will create a strata for each level of the covariate, up to 20 levels. If `factorSplit == FALSE`, will divide up numeric covariates according to the cuts provided to numericCuts.

`lgdLocation` is an argument placed to `legend` dictating where the legend will be placed. If `lgdLocation = NULL`, will use standard placement given `yType`. See `?legend` for more details.

Author(s)

Clifford Anderson-Bergman

`getFitEsts` *Get Survival Curve Estimates from icenReg Model*

Description

Gets probability or quantile estimates from a `ic_sp`, `ic_par` or `ic_bayes` object. Provided estimates conditional on regression parameters found in `newdata`.

Usage

```
getFitEsts(fit, newdata, p, q)
```

Arguments

`fit`	model fit with `ic_par` or `ic_sp`
`newdata`	`data.frame` containing covariates
`p`	Percentiles
`q`	Quantiles

Details

For the `ic_sp` and `ic_par`, the MLE estimate is returned. For `ic_bayes`, the MAP estimate is returned. To compute the posterior means, use `sampleSurv`.

If `newdata` is left blank, baseline estimates will be returned (i.e. all covariates = 0). If `p` is provided, will return the estimated F^-1(p | x). If `q` is provided, will return the estimated F(q | x). If neither `p` nor `q` are provided, the estimated conditional median is returned.

In the case of ic_sp, the MLE of the baseline survival is not necessarily unique, as probability mass is assigned to disjoint Turnbull intervals, but the likelihood function is indifferent to how probability mass is assigned within these intervals. In order to have a well defined estimate returned, we assume probability is assigned uniformly in these intervals. In otherwords, we return *a* maximum likelihood estimate, but don't attempt to characterize *all* maximum likelihood estimates with this function. If that is desired, all the information needed can be extracted with getSCurves.

Author(s)

Clifford Anderson-Bergman

Examples

```
simdata <- simIC_weib(n = 500, b1 = .3, b2 = -.3,
inspections = 6, inspectLength = 1)
fit <- ic_par(Surv(l, u, type = 'interval2') ~ x1 + x2,
              data = simdata)
new_data <- data.frame(x1 = c(1,2), x2 = c(-1,1))
rownames(new_data) <- c('grp1', 'grp2')

estQ <- getFitEsts(fit, new_data, p = c(.25, .75))

estP <- getFitEsts(fit, q = 400)
```

---

getSCurves — *Get Estimated Survival Curves from Semi-parametric Model for Interval Censored Data*

Description

Extracts the estimated survival curve(s) from an ic_sp or ic_np model for interval censored data.

Usage

```
getSCurves(fit, newdata = NULL)
```

Arguments

fit	model fit with ic_sp
newdata	data.frame containing covariates for which the survival curve will be fit to. Rownames from newdata will be used to name survival curve. If left blank, baseline covariates will be used

Details

Output will be a list with two elements: the first item will be `$Tbull_ints`, which is the Turnbull intervals. This is a k x 2 matrix, with the first column being the beginning of the Turnbull interval and the second being the end. This is necessary due to the *representational non-uniqueness*; any survival curve that lies between the survival curves created from the upper and lower limits of the Turnbull intervals will have equal likelihood. See example for proper display of this. The second item is `$S_curves`, or the estimated survival probability at each Turnbull interval for individuals with the covariates provided in `newdata`. Note that multiple rows may be provided to newdata, which will result in multiple S_curves.

Author(s)

Clifford Anderson-Bergman

`ic_bayes` *Bayesian Regression Models for Interval Censored Data*

Description

Fits a Bayesian regression model for interval censored data. Can fit a proportional hazards, proportional odds or accelerated failure time model.

Usage

```
ic_bayes(formula, data, logPriorFxn = function(x) return(0), model = "ph",
  dist = "weibull", weights = NULL, controls = bayesControls(),
  useMCores = F)
```

Arguments

`formula`	Regression formula. Response must be a `Surv` object of type `'interval2'` or `cbind`. See details.
`data`	Dataset
`logPriorFxn`	An R function that computes the log prior
`model`	What type of model to fit. Current choices are "`ph`" (proportional hazards), "`po`" (proportional odds) or "`aft`" (accelerated failure time)
`dist`	What baseline parametric distribution to use. See details for current choices
`weights`	vector of case weights. Not standardized; see details
`controls`	Control parameters passed to samplers
`useMCores`	Should multiple cores be used? Each core is used to run a single chain.

Details

Currently supported distributions choices are "exponential", "weibull", "gamma", "lnorm", "loglogistic" and "generalgamma" (i.e. generalized gamma distribution).

The `logPriorFxn` should take in the a vector of values corresponding to *all* the parameters of the model (baseline parameters first, regression parameters second) and returns the log prior, calculated up to an additive constant. Default behavior is to use a flat prior. See examples for an example of using the log prior function.

Sampling is done by a single MH block updater on all the parameters. See `?bayesControls` for more details.

Response variable should either be of the form `cbind(l, u)` or `Surv(l, u, type = 'interval2')`, where `l` and `u` are the lower and upper ends of the interval known to contain the event of interest. Uncensored data can be included by setting `l == u`, right censored data can be included by setting `u == Inf` or `u == NA` and left censored data can be included by setting `l == 0`.

Does not allow uncensored data points at t = 0 (i.e. `l == u == 0`), as this will lead to a degenerate estimator for most parametric families. Unlike the current implementation of survival's `survreg`, does allow left side of intervals of positive length to 0 and right side to be `Inf`.

In regards to weights, they are not standardized. This means that if weight[i] = 2, this is the equivalent to having two observations with the same values as subject i.

For numeric stability, if abs(right - left) < 10^-6, observation are considered uncensored rather than interval censored with an extremely small interval.

Author(s)

Clifford Anderson-Bergman

Examples

```
data(miceData)

flat_prior_model <- ic_bayes(cbind(l, u) ~ grp, data = miceData)
# Default behavior is flat prior

priorFxn <- function(pars){
 ans <- 0
 ans <- ans + dnorm(pars[1], log = TRUE)
 ans <- ans + dnorm(pars[3], sd = 0.25, log = TRUE)
}
# Prior function puts N(0,1) prior on baseline shape parameter (first parameter)
# flat prior on baseline scale parameter (second parameter)
# and N(0,0.25) on regression parameter (third parameter)

inform_prior_fit <- ic_bayes(cbind(l, u) ~ grp,
                             data = miceData,
                             logPriorFxn = priorFxn)

summary(flat_prior_model)
summary(inform_prior_fit)
# Note tight prior on the regression pulls posterior mean toward 0
```

ic_np *Non-Parametric Estimator for Interval Censored Data*

Description

Fits the non-parametric maximum likelihood estimator (NPMLE) for univariate interval censored data. This is a generalization of the Kaplan-Meier curves that allows for interval censoring. Also referred to as the Turnbull estimator.

Usage

```
ic_np(formula = NULL, data, maxIter = 1000, tol = 10^-10, B = c(0, 1),
  weights = NULL)
```

Arguments

formula	Formula for stratification. If only one group, can be left blank and data must be entered as n x 2 matrix.
data	A data.frame or an n x 2 matrix. See details.
maxIter	Maximum iterations
tol	Numeric tolerance
B	Should intervals be open or closed? See details.
weights	Weights (optional)

Details

data must be an n x 2 matrix or data.frame containing two columns of data representing left and right sides of the censoring interval, denoted L and R. This allows for left censored (L == 0), right censored (R == inf), uncensored (L == R) along with general interval censored observations.

The argument B determines whether the intervals should be open or closed, i.e. B = c(0,1) implies that the event occurs within the interval (l,u]. The exception is that if l == u, it is assumed that the event is uncensored, regardless of B.

The NPMLE is fit using an efficient implementation of the EMICM algorithm.

Author(s)

Clifford Anderson-Bergman

References

Turnbull, B. (1976) The empricial distribution with arbitrarily grouped and censored data *Journal of the Royal Statistical Society B*, vol 38 p290-295

Wellner, J. A., and Zhan, Y. (1997) A hybrid algorithm for computation of the maximum likelihood estimator from censored data, *Journal of the American Statistical Association*, Vol 92, pp945-959

Anderson-Bergman, C. (2016) An efficient implementation of the EMICM algorithm for the interval censored NPMLE *Journal of Computational and Graphical Statistics*, *just accepted*

Examples

```
data(miceData)
fit <- ic_np(cbind(l, u) ~ grp, data = miceData)
# Stratifies fits by group

plot(fit)
```

---

ic_par *Parametric Regression Models for Interval Censored Data*

Description

Fits a parametric regression model for interval censored data. Can fita proportional hazards, proportional odds or accelerated failure time model.

Usage

```
ic_par(formula, data, model = "ph", dist = "weibull", weights = NULL)
```

Arguments

formula	Regression formula. Response must be a Surv object of type 'interval2' or cbind. See details.
data	Dataset
model	What type of model to fit. Current choices are "ph" (proportional hazards), "po" (proportional odds) or "aft" (accelerated failure time)
dist	What baseline parametric distribution to use. See details for current choices
weights	vector of case weights. Not standardized; see details

Details

Currently supported distributions choices are "exponential", "weibull", "gamma", "lnorm", "loglogistic" and "generalgamma" (i.e. generalized gamma distribution).

Response variable should either be of the form `cbind(l, u)` or `Surv(l, u, type = 'interval2')`, where `l` and `u` are the lower and upper ends of the interval known to contain the event of interest. Uncensored data can be included by setting `l == u`, right censored data can be included by setting `u == Inf` or `u == NA` and left censored data can be included by setting `l == 0`.

Does not allow uncensored data points at t = 0 (i.e. `l == u == 0`), as this will lead to a degenerate estimator for most parametric families. Unlike the current implementation of survival's `survreg`, does allow left side of intervals of positive length to 0 and right side to be `Inf`.

In regards to weights, they are not standardized. This means that if weight[i] = 2, this is the equivalent to having two observations with the same values as subject i.

For numeric stability, if abs(right - left) < 10^-6, observation are considered uncensored rather than interval censored with an extremely small interval.

Author(s)

Clifford Anderson-Bergman

Examples

```
data(miceData)

logist_ph_fit <- ic_par(Surv(l, u, type = 'interval2') ~ grp,
                        data = miceData, dist = 'loglogistic')

logist_po_fit <- ic_par(cbind(l, u) ~ grp,
                         data = miceData, dist = 'loglogistic',
                        model = 'po')

summary(logist_ph_fit)
summary(logist_po_fit)
```

---

ic_sample *Draw samples from an icenReg model*

Description

Samples response values from an icenReg fit conditional on covariates, but not censoring intervals. To draw response samples conditional on covariates and restrained to intervals, see `imputeCens`.

Usage

```
ic_sample(fit, newdata = NULL, sampleType = "fullSample", samples = 5)
```

Arguments

`fit`	icenReg model fit
`newdata`	`data.frame` containing covariates. If blank, will use data from model
`sampleType`	type of samples See details for options
`samples`	Number of samples

Details

Returns a matrix of samples. Each row of the matrix corresponds with a subject with the covariates of the corresponding row of `newdata`. For each column of the matrix, the same sampled parameters are used to sample response variables.

If `newdata` is left blank, will provide estimates for original data set.

There are several options for how to sample. To get random samples without accounting for error in the estimated parameters `imputeType ='fixedParSample'` takes a random sample of the response variable, conditional on the response interval, covariates and estimated parameters at the MLE. Alternatively, `imputeType = 'fullSample'` first takes a random sample of the coefficients, (assuming asymptotic normality for the ic_par) and then takes a random sample of the response variable, conditional on the response interval, covariates, and the random sample of the coefficients.

Author(s)

Clifford Anderson-Bergman

Examples

```
simdata <- simIC_weib(n = 500)

fit <- ic_par(cbind(l, u) ~ x1 + x2,
              data = simdata)

newdata = data.frame(x1 = c(0, 1), x2 = c(1,1))

sampleResponses <- ic_sample(fit, newdata = newdata, samples = 100)
```

---

ic_sp *Semi-Parametric models for Interval Censored Data*

Description

Fits a semi-parametric model for interval censored data. Can fit either a Cox-PH model or a proportional odds model.

The covariance matrix for the regression coefficients is estimated via bootstrapping. For large datasets, this can become slow so parallel processing can be used to take advantage of multiple cores via the `foreach` package.

Usage

```
ic_sp(formula, data, model = "ph", weights = NULL, bs_samples = 0,
  useMCores = F, B = c(0, 1), controls = makeCtrls_icsp())
```

Arguments

`formula`	regression formula. Response must be a `Surv` object of type `'interval2'`or `cbind`. See details.
`data`	dataset
`model`	What type of model to fit. Current choices are "ph" (Cox PH) or "po" (proportional odds)
`weights`	Vector of case weights. Not standardized; see details
`bs_samples`	Number of bootstrap samples used for estimation of standard errors
`useMCores`	Should multiple cores be used for bootstrap sample? Does not register cluster (see example)
`B`	Should intervals be open or closed? See details.
`controls`	Advanced control options

Details

Response variable should either be of the form `cbind(l, u)` or `Surv(l, u, type = 'interval2')`, where `l` and `u` are the lower and upper ends of the interval known to contain the event of interest. Uncensored data can be included by setting `l == u`, right censored data can be included by setting `u == Inf` or `u == NA` and left censored data can be included by setting `l == 0`.

The argument `B` determines whether the intervals should be open or closed, i.e. `B = c(0,1)` implies that the event occurs within the interval `(l,u]`. The exception is that if `l == u`, it is assumed that the event is uncensored, regardless of `B`.

In regards to weights, they are not standardized. This means that if weight[i] = 2, this is the equivalent to having two observations with the same values as subject i.

The algorithm used is inspired by the extended ICM algorithm from Wei Pan 1999. However, it uses a conditional Newton Raphson step (for the regression parameters) and an ICM step (for the baseline survival parameters), rather than one single ICM step (for both sets). In addition, a gradient ascent can also be used to update the baseline parameters. This step is necessary if the data contains many uncensored observations, very similar to how the EM algorithm greatly accelerates the ICM algorithm for the NPMLE (gradient ascent is used rather than the EM, as the M step is not in closed form for semi-parametric models).

Earlier versions of icenReg used an active set algorithm, which was not as fast for large datasets.

Author(s)

Clifford Anderson-Bergman

References

Pan, W., (1999), Extending the iterative convex minorant algorithm to the Cox model for interval-censored data, *Journal of Computational and Graphical Statistics*, Vol 8(1), pp109-120

Wellner, J. A., and Zhan, Y. (1997) A hybrid algorithm for computation of the maximum likelihood estimator from censored data, *Journal of the American Statistical Association*, Vol 92, pp945-959

Anderson-Bergman, C. (preprint) Revisiting the iterative convex minorant algorithm for interval censored survival regression models

Examples

```
set.seed(1)

sim_data <- simIC_weib(n = 100, inspections = 5, inspectLength = 1)
ph_fit <- ic_sp(Surv(l, u, type = 'interval2') ~ x1 + x2,
                data = sim_data)
# Default fits a Cox-PH model

summary(ph_fit)
# Regression estimates close to true 0.5 and -0.5 values


new_data <- data.frame(x1 = c(0,1), x2 = c(1, 1) )
rownames(new_data) <- c('group 1', 'group 2')
plot(ph_fit, new_data)
```

```
# plotting the estimated survival curves

po_fit <- ic_sp(Surv(l, u, type = 'interval2') ~ x1 + x2,
                data = sim_data, model = 'po')
# fits a proportional odds model

summary(po_fit)

# Not run: how to set up multiple cores
# library(doParallel)
# myCluster <- makeCluster(2)
# registerDoParallel(myCluster)
# fit <- ic_sp(Surv(l, u, type = 'interval2') ~ x1 + x2,
#              data = sim_data, useMCores = TRUE
#              bs_samples = 500)
# stopCluster(myCluster)
```

`imputeCens` *Impute Interval Censored Data from icenReg Regression Model*

Description

Imputes censored responses from data.

Usage

```
imputeCens(fit, newdata = NULL, imputeType = "fullSample", samples = 5)
```

Arguments

`fit`	icenReg model fit
`newdata`	`data.frame` containing covariates and censored intervals. If blank, will use data from model
`imputeType`	type of imputation. See details for options
`samples`	Number of imputations (ignored if `imputeType = "median"`)

Details

If `newdata` is left blank, will provide estimates for original data set.

There are several options for how to impute. `imputeType = 'median'` imputes the median time, conditional on the response interval, covariates and regression parameters at the MLE. To get random imputations without accounting for error in the estimated parameters `imputeType ='fixedParSample'` takes a random sample of the response variable, conditional on the response interval, covariates and estimated parameters at the MLE. Finally, `imputeType = 'fullSample'` first takes a random sample of the coefficients, (assuming asymptotic normality) and then takes a random sample of the response variable, conditional on the response interval, covariates, and the random sample of the coefficients.

Author(s)

Clifford Anderson-Bergman

Examples

```
simdata <- simIC_weib(n = 500)

fit <- ic_par(cbind(l, u) ~ x1 + x2,
              data = simdata)

imputedValues <- imputeCens(fit)
```

---

ir_clustBoot *Updates the covariance using cluster bootstrap*

Description

Adjusts error estimates for repeated measures data by use of the cluster bootstrap.

Usage

```
ir_clustBoot(fit, ID, bs_samples = 1000)
```

Arguments

fit	Either an ic_par or ic_sp model
ID	Subject identifier
bs_samples	Number of bootstrap samples

Details

Standard models in icenReg assume independence between each observation. This assumption is broken if we can have multiple observations from a single subject, which can lead to an underestimation of the standard errors. ir_clustBoot addresses this by using a cluster bootstrap to fix up the standard errors.

Note that this requires refitting the model bs_samples, which means this can be fairly time consuming.

References

Sherman, Michael, and Saskia le Cessie. "A comparison between bootstrap methods and generalized estimating equations for correlated outcomes in generalized linear models." Communications in Statistics-Simulation and Computation 26.3 (1997): 901-925.

Examples

```
# Simulating repeated measures data
simdata = simIC_cluster(nIDs = 10, nPerID = 4)

# Fitting with basic model
fit = ic_par(cbind(l,u) ~ x1 + x2, data = simdata)
fit

# Updating covariance
ir_clustBoot(fit, ID = simdata$ID, bs_samples = 10)
# (Low number of bootstrap samples used for quick testing by CRAN,
# never use this few!!)

# Note that the SE's have changed from above
fit
```

IR_diabetes *Interval censored time from diabetes onset to diabetic nephronpathy*

Description

Data set contains interval censored survival time for time from onset of diabetes to to diabetic nephronpathy. Identical to the `diabetes` dataset found in the package `glrt`.

Fields

`left` left side of observation interval

`right` right side of observation interval

`gender` gender of subject

References

Borch-Johnsens, K, Andersen, P and Decker, T (1985). "The effect of proteinuria on relative mortality in Type I (insulin-dependent) diabetes mellitus." Diabetologia, 28, 590-596.

Examples

```
data(IR_diabetes)
fit <- ic_par(cbind(left, right) ~ gender,
              data = IR_diabetes,
              model = "po",
              dist = "loglogistic")
```

`lines.icenReg_fit` *Plotting for icenReg Fits*

Description

Plotting for icenReg Fits

Usage

```
## S3 method for class 'icenReg_fit'
lines(x, y, newdata = NULL, fun = "surv", cis = F,
  ci_level = 0.9, survRange = c(0.025, 1), evalPoints = 20, ...)
```

Arguments

`x`	icenReg fit
`y`	new data.frame
`newdata`	new data.frame (ignored if `y` is included)
`fun`	Function to be plotted. Options include `"surv"` or `"cdf"`
`cis`	Should confidence/credible interval be plotted?
`ci_level`	Confidence/credible interval
`survRange`	Range of survival curve to be plotted
`evalPoints`	Number of evaluations of survival curve to be plotted.
`...`	additional arguments to be passed to the base `plot` function

Details

Plots survival function from either an `ic_np, ic_sp, ic_par` or `ic_bayes` object. If `newdata` is `NULL`, the baseline distribution is plotted. Otherwise, `newdata` should be a `data.frame` with each row containing a set covariates for which the fit will be plotted. If multiple rows are included, the lines will be colored and a legend will be created using the rownames of `newdata`.

For `ic_np` and `ic_sp`, the MLE is plotted with no intervals (at the time of writing this, there is no formula for standard errors of baseline distributions for these methods).

For `ic_par` and `ic_bayes`, the output plotted is directly extracted from `survCIs`.

If the argument `col` is provided, it will be used to color each survival function in order of the rows provided in `newdata`.

makeCtrls_icsp *Control Parameters for ic_sp*

Description

Control Parameters for ic_sp

Usage

```
makeCtrls_icsp(useGA = T, maxIter = 10000, baseUpdates = 5,
  regStart = NULL)
```

Arguments

useGA	Should constrained gradient ascent step be used?
maxIter	Maximum iterations
baseUpdates	number of baseline updates (ICM + GA) per iteration
regStart	Initial values for regression parameters @description Creates the control options for the ic_sp function. Defaults not intended to be changed for use in standard analyses.

Details

The constrained gradient step, actived by useGA = T, is a step that was added to improve the convergence in a special case. The option to turn it off is only in place to help demonstrate it's utility.

regStart also for seeding of initial value of regression parameters. Intended for use in "warm start" for bootstrap samples and providing fixed regression parameters when calculating fit in qq-plots.

Author(s)

Clifford Anderson-Bergman

miceData *Lung Tumor Interval Censored Data from Hoel and Walburg 1972*

Description

RFM mice were sacrificed and examined for lung tumors. This resulted in current status interval censored data: if the tumor was present, this implied left censoring and if no tumor was present this implied right censoring. Mice were placed in two different groups: conventional environment or germ free environment.

Fields

l left side of observation interval

u right side of observation interval

grp Group for mouse. Either ce (conventional environment) or ge (grem-free environment)

References

Hoel D. and Walburg, H.,(1972), Statistical analysis of survival experiments, *The Annals of Statistics*, 18, 1259-1294

Examples

```
data(miceData)

coxph_fit <- ic_sp(Surv(l, u, type = 'interval2') ~ grp,
                   bs_samples = 50,
                   data = miceData)

#In practice, more bootstrap samples should be used for inference
#Keeping it quick for CRAN testing purposes

summary(coxph_fit)
```

---

plot.icenReg_fit *Plotting for icenReg Fits*

Description

Plotting for icenReg Fits

Usage

```
## S3 method for class 'icenReg_fit'
plot(x, y, newdata = NULL, fun = "surv",
  plot_legend = T, cis = T, ci_level = 0.9, survRange = c(0.025, 1),
  evalPoints = 200, lgdLocation = "topright", xlab = "time", ...)
```

Arguments

x	icenReg fit
y	new data.frame
newdata	new data.frame (ignored if y is included)
fun	Function to be plotted. Options include "surv" or "cdf"
plot_legend	Should legend be plotted?
cis	Should confidence/credible interval be plotted?

ci_level	Confidence/credible interval
survRange	Range of survival curve to be plotted
evalPoints	Number of evaluations of survival curve to be plotted.
lgdLocation	Location of legend; see ?legend for options
xlab	Label of x-axis
...	additional arguments to be passed to the base plot function

Details

Plots survival function from either an ic_np, ic_sp, ic_par or ic_bayes object. If newdata is NULL, the baseline distribution is plotted. Otherwise, newdata should be a data.frame with each row containing a set covariates for which the fit will be plotted. If multiple rows are included, the lines will be colored and a legend will be created using the rownames of newdata.

For ic_np and ic_sp, the MLE is plotted with no intervals (at the time of writing this, there is no formula for standard errors of baseline distributions for these methods).

For ic_par and ic_bayes, the output plotted is directly extracted from survCIs.

If the argument col is provided, it will be used to color each survival function in order of the rows provided in newdata.

Examples

```
# Fitting mice data set
data(miceData)
miceFit <- ic_par(cbind(l, u) ~ grp, data = miceData)

# Creating covariates we want plotted
newData <- data.frame(grp = c("ce", "ge"))
# Naming rows for legend
rownames(newData) <- c("Conventional", "Germ-Free")

plot(miceFit, newdata = newData,
     col = c('blue', 'orange'))
```

---

predict.icenReg_fit *Predictions from icenReg Regression Model*

Description

Gets various estimates from an ic_np, ic_sp or ic_par object.

Usage

```
## S3 method for class 'icenReg_fit'
predict(object, type = "response", newdata = NULL,
  ...)
```

Arguments

object	Model fit with ic_par or ic_sp
type	type of prediction. Options include "lp", "response"
newdata	data.frame containing covariates
...	other arguments (will be ignored)

Details

If newdata is left blank, will provide estimates for original data set.

For the argument type, there are two options. "lp" provides the linear predictor for each subject (i.e. in a proportional hazards model, this is the log-hazards ratio, in proportional odds, the log proporitonal odds), "response" provides the median response value for each subject, *conditional on that subject's covariates, but ignoring their actual response interval*. Use imputeCens to impute the censored values.

Author(s)

Clifford Anderson-Bergman

Examples

```
simdata <- simIC_weib(n = 500, b1 = .3, b2 = -.3,
                      inspections = 6,
                      inspectLength = 1)

fit <- ic_par(cbind(l, u) ~ x1 + x2,
              data = simdata)

imputedValues <- predict(fit)
```

---

sampleSurv *Samples fitted survival function*

Description

Samples fitted survival function

Usage

```
sampleSurv(fit, newdata = NULL, p = NULL, q = NULL, samples = 100)
```

Arguments

fit	Either an ic_bayes or ic_par fit
newdata	A data.frame with a single row of covariates
p	A set of survival probabilities to sample corresponding time for
q	A set of times to sample corresponding cumulative probability for
samples	Number of samples to draw

Details

For Bayesian models, draws samples from the survival distribution with a given set of covariates. Does this by first drawing a set of parameters (both regression and baseline) from `fit$samples` and then computing the quantiles of the distribution (if `p` is provided) or the CDF at `q`.

If a `ic_par` model is provided, the procedure is the same, but the sampled parameters are drawn using the normal approximation.

Not compatible with `ic_np` or `ic_sp` objects.

Author(s)

Clifford Anderson-Bergman

Examples

```
data("IR_diabetes")
fit <- ic_par(cbind(left, right) ~ gender, data = IR_diabetes)

newdata <- data.frame(gender = "male")
time_samps <- sampleSurv(fit, newdata,
                         p = c(0.5, .9),
                         samples = 100)
# 100 samples of the median and 90th percentile for males

prob_samps <- sampleSurv(fit, newdata,
                         q = c(10, 20),
                         samples = 100)
# 100 samples of the cumulative probability at t = 10 and 20 for males
```

---

simCS_weib *Simulate Current Status Data*

Description

Simulates current status data from a survival regression model with a Weibull baseline distribution.

Usage

```
simCS_weib(n = 100, b1 = 0.5, b2 = -0.5, model = "ph", shape = 2,
  scale = 2)
```

Arguments

`n`	Number of observations
`b1`	Regression coefficient 1
`b2`	Regression coefficient 2
`model`	Regression model to use. Choices are `"ph"`, `"po"` or `"aft"`
`shape`	Baseline shape parameter
`scale`	Baseline scale parameter

Details

Exact event times are simulated according to the given survival regression model. Two covariates are used; `x1 = rnorm(n), x2 = 1 - 2 * rbinom(n, 1, .5)`. After event times are simulated, current status inspection times are simulated following the exact same conditional distribution as event time (so each event time necessarily has probability 0.5 of being right censored).

Returns data in current status format, i.e. inspection time and event indicator. Use `cs2ic` to convert to interval censored format (see example).

Examples

```
simData <- simCS_weib()
fit <- ic_par(cs2ic(time, event) ~ x1 + x2, data = simData)
```

---

simDC_weib *Simulate Doubly Censored Data*

Description

Simulates doubly censored data from a survival regression model with a Weibull baseline distribution.

Usage

```
simDC_weib(n = 100, b1 = 0.5, b2 = -0.5, model = "ph", shape = 2,
  scale = 2, lowerLimit = 0.75, upperLimit = 2)
```

Arguments

`n`	Number of observations
`b1`	Regression coefficient 1
`b2`	Regression coefficient 2
`model`	Regression model to use. Choices are `"ph"`, `"po"` or `"aft"`
`shape`	Baseline shape parameter
`scale`	Baseline scale parameter
`lowerLimit`	Lower censoring threshold
`upperLimit`	Upper censoring threshold

Details

Exact event times are simulated according to the given survival regression model. Two covariates are used; `x1 = rnorm(n), x2 = 1 - 2 * rbinom(n, 1, .5)`. After event times are simulated, all values less than `lowerLimit` are left censored and all values less than `upperLimit` are right censored.

Examples

```
simData <- simDC_weib()
fit <- ic_par(cbind(l, u) ~ x1 + x2, data = simData)
```

`simIC_cluster` *Simulates data with multiple observations per subject*

Description

Simulates data in which each subject is observed several times. In this case, the covariance matrix should be updated with `ir_clustBoot`.

Usage

```
simIC_cluster(nIDs = 50, nPerID = 5)
```

Arguments

`nIDs`	Number of subjects
`nPerID`	Number of observations per subject

`simIC_weib` *Simulates interval censored data from regression model with a Weibull baseline*

Description

Simulates interval censored data from a regression model with a weibull baseline distribution. Used for demonstration

Usage

```
simIC_weib(n = 100, b1 = 0.5, b2 = -0.5, model = "ph", shape = 2,
  scale = 2, inspections = 2, inspectLength = 2.5, rndDigits = NULL,
  prob_cen = 1)
```

Arguments

`n`	Number of samples simulated
`b1`	Value of first regression coefficient
`b2`	Value of second regression coefficient
`model`	Type of regression model. Options are 'po' (prop. odds) and 'ph' (Cox PH)
`shape`	shape parameter of baseline distribution
`scale`	scale parameter of baseline distribution

inspections	number of inspections times of censoring process
inspectLength	max length of inspection interval
rndDigits	number of digits to which the inspection time is rounded to, creating a discrete inspection time. If rndDigits = NULL, the inspection time is not rounded, resulting in a continuous inspection time
prob_cen	probability event being censored. If event is uncensored, l == u

Details

Exact event times are simulated according to regression model: covariate x1 is distributed rnorm(n) and covariate x2 is distributed 1 - 2 * rbinom(n, 1, 0.5). Event times are then censored with a case II interval censoring mechanism with inspections different inspection times. Time between inspections is distributed as runif(min = 0, max = inspectLength). Note that the user should be careful in simulation studies not to simulate data where nearly all the data is right censored (or more over, all the data with x2 = 1 or -1) or this can result in degenerate solutions!

Author(s)

Clifford Anderson-Bergman

Examples

```
set.seed(1)
sim_data <- simIC_weib(n = 500, b1 = .3, b2 = -.3, model = 'ph',
                       shape = 2, scale = 2, inspections = 6,
                       inspectLength = 1)
#simulates data from a cox-ph with beta weibull distribution.

diag_covar(Surv(l, u, type = 'interval2') ~ x1 + x2,
           data = sim_data, model = 'po')
diag_covar(Surv(l, u, type = 'interval2') ~ x1 + x2,
           data = sim_data, model = 'ph')

#'ph' fit looks better than 'po'; the difference between the transformed survival
#function looks more constant
```

---

survCIs *Confidence/Credible intervals for survival curves*

Description

Confidence/Credible intervals for survival curves

Usage

```
survCIs(fit, newdata = NULL, p = NULL, q = NULL, ci_level = 0.95,
  MC_samps = 4000)
```

Arguments

fit	Fitted model from ic_par or ic_bayes
newdata	data.frame containing covariates for survival curves
p	Percentiles of distribution to sample
q	Times of disitribution to sample. Only p OR q should be specified, not p AND q
ci_level	Confidence/credible level
MC_samps	Number of Monte Carlo samples taken

Details

Creates a set of confidence intervals for the survival curves conditional on the covariates provided in newdata. Several rows can be provided in newdata; this will lead to several sets of confidence/credible intervals.

For Bayesian models, these are draws directly from the posterior; a set of parameters drawn from those saved in fit$samples repeatedly and then for each set of parameters, the given set of quantiles is calculated. For parametric models, the procedure is virtually the same, but rather than randomly drawing rows from saved samples, random samples are drawn using the asymptotic normal approximation of the estimator.

This function is not compatible with ic_np or ic_sp objects, as the distribution of the baseline distribution of these estimators is still an open question.

Author(s)

Clifford Anderson-Bergman

Examples

```
data("IR_diabetes")
fit <- ic_par(cbind(left, right) ~ gender,
              data = IR_diabetes)

# Getting confidence intervals for survival curves
# for males and females
newdata <- data.frame(gender = c("male", "female"))
rownames(newdata) <- c("Males", "Females")
diab_cis <- survCIs(fit, newdata)
diab_cis

# Can add this to any plot
plot(fit, newdata = newdata,
     cis = FALSE)
# Would have been included by default
lines(diab_cis, col = c("black", "red"))
```

Index