Package 'icenReg'

December 9, 2015

Type Package

Title Regression Models for Interval Censored Data

Version 1.2.8

Date 2015-10-26

Author Clifford Anderson-Bergman; the Eigen team for Eigen library included; uses Maarloes Maathius's HeightMap algorithm (MLEcens::reduc)

Depends survival, MLEcens

Imports foreach, methods

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

Description Regression models for interval censored data. Currently supports Cox-PH and proportional odds models. Allows for both semi and fully parametric models. Includes functions for easy visual diagnostics of model fits. Includes functions for fitting both the univariate and bivariate NPMLE.

License LGPL (>= 2.0, < 3)

R topics documented:

abs_inv *Loss Function for interval censored cross validation*

Description

A loss function for survival data. Equal to

`mean(abs(1/(pred+1) - 1/(t_val+1 ) ) ) .`

This function heavily penalizes inaccurate predictions that are closer to the origin than those that are farther way. In otherwords, it will favor a model that accurately predicts high risk subjects over a model that accurately predicts loss risk subjects.

Usage

```
abs_inv(pred, t_val)
```

Arguments

`pred`	predicted value
`t_val`	true value

Author(s)

Clifford Anderson-Bergman

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'
dists	parametric baseline fits
cols	colors of baseline distributions
weights	case weights
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(essIncData_small)
useData <- essIncData_small

#Note to user: suggest replacing useData with essIncData
#instead of essIncData_small. Using small dataset to quickly
#pass CRAN tests

fit_po <- ic_sp(Surv(inc_l, inc_u, type = 'interval2') ~ eduLevel * cntry,
                data = useData, bs_samples = 0, model = 'po')

diag_baseline(fit_po)

#Weibull model appears best fit
```

---

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 stratas 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 difference between the plotted lines should be approximately constant. 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

Examples

```
data(essIncData_small)
useData <- essIncData_small

#Note to user: suggest replacing useData with essIncData
#instead of essIncData_small. Using small dataset to quickly
#pass CRAN tests
par(mfrow = c(1,2))

diag_covar(Surv(inc_l, inc_u, type = 'interval2') ~ eduLevel * cntry,
        data = useData, model = 'po')

diag_covar(Surv(inc_l, inc_u, type = 'interval2') ~ eduLevel * cntry,
        data = useData, model = 'ph')
```

---

essIncData *Interval Censored Income Data from European Social Survey*

Description

Dataset containing a sample from the European Social Survey.

In the European Social Survey, income is reported up a to discrete intervals. Within each country, these intervals are disjoint (i.e. [0, 1k), [1k,2k)...). However, across countries, the intervals are not disjoint and so interval censored methods should be used to compare subjects across different countries.

Usage

```
data(essIncData)
```

Format

A data frame with 6712 rows and 4 variables

- `cntry` Country
- `eduLevel` Categorical variable for number of years of reported education
- `inc_l` Lower limit of reported income level (in Euros)
- `inc_u` Upper limit of reported income level (in Euros)

Source

ESS Round 5: European Social Survey Round 5 Data (2010). Data file edition 3.2. Norwegian Social Science Data Services, Norway\- Data Archive and distributor of ESS data.

Examples

```
data(essIncData)

lnormFit <- ic_par(Surv(inc_l, inc_u, type = 'interval2') ~ eduLevel * cntry,
                   data = essIncData,
                   model = 'po',
                   dist = 'loglogistic')

summary(lnormFit)
```

---

essIncData_small *Interval Censored Income Data from European Social Survey*

Description

Dataset containing a sample from the European Social Survey. This is a small subsample from the dataset `essIncData`, used only to run the example very quickly for CRAN. Using the full dataset, running the examples should still be fairly fast; at most a few seconds.

In the European Social Survey, income is reported up a to discrete intervals. Within each country, these intervals are disjoint (i.e. [0, 1k), [1k,2k)...). However, across countries, the intervals are not disjoint and so interval censored methods should be used to compare subjects across different countries.

Usage

```
data(essIncData_small)
```

Format

A data frame with 500 rows and 4 variables

- `cntry` Country
- `eduLevel` Categorical variable for number of years of reported education
- `inc_l` Lower limit of reported income level (in Euros)
- `inc_u` Upper limit of reported income level (in Euros)

Source

ESS Round 5: European Social Survey Round 5 Data (2010). Data file edition 3.2. Norwegian Social Science Data Services, Norway\- Data Archive and distributor of ESS data.

Examples

```
data(essIncData_small)

lnormFit <- ic_par(Surv(inc_l, inc_u, type = 'interval2') ~ eduLevel * cntry,
                   data = essIncData_small,
                   model = 'po',
                   dist = 'loglogistic')

summary(lnormFit)
```

getFitEsts *Get Estimates from icenReg Regression Model*

Description

Gets estimates from a ic_par or ic_sp 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

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.

For ic_par fits, it is worth noting that F^-1(p | x) is approximated using R's optimize function, so there may be some numerical error.

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 a ic_sp model for interval censored data. 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.

Usage

```
getSCurves(fit, newdata)
```

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

Author(s)

Clifford Anderson-Bergman

Examples

```
set.seed(1)

sim_data <- simIC_weib(n = 500, b1 = .3, b2 = -.3,
                       shape = 2, scale = 2,
                       inspections = 6, inspectLength = 1)
fit <- ic_sp(Surv(l, u, type = 'interval2') ~ x1 + x2, data = sim_data, bs_samples = 0)

new_data <- data.frame(x1 = c(0,1), x2 = c(1, 1) )
#want to fit survival curves with above covariates
rownames(new_data) <- c('group 1', 'group 2')
#getSCurves will name the survival curves according to rownames

curveInfo <- getSCurves(fit, new_data)
xs <- curveInfo$Tbull_ints
#Extracting Turnbull intervals
sCurves <- curveInfo$S_curves
#Extracting estimated survival curves

plot(xs[,1], sCurves[[1]], xlab = 'time', ylab = 'S(t)',
     type = 's', ylim = c(0,1),
```

```
     xlim = range(as.numeric(xs), finite = TRUE))
#plotting upper survival curve estimate
lines(xs[,2], sCurves[[1]], type = 's')
#plotting lower survival curve estimate

lines(xs[,1], sCurves[[2]], col = 'blue', type = 's')
lines(xs[,2], sCurves[[2]], col = 'blue', type = 's')
#plotting upper and lower survival curves for group 2

# Actually, all this plotting is a unnecessary:
# plot(fit, new_data) will bascially do this all
# But this is more of a tutorial in case custom
# plots were desired
```

`icenReg_cv` *Get Cross-Validation Statistics from icenReg Regression model*

Description

Computes the cross validation error icenReg regression models by using multiple imputations to get estimates of the censored values and taking the average loss across all imputations. Allows user to provide their own loss function.

Usage

```
icenReg_cv(fit, loss_fun = abs_inv, folds = 10,
           numImputes = 100, useMCore = F)
```

Arguments

`fit`	model fit with `ic_par` or `ic_sp`. See details
`loss_fun`	Loss function. See details
`folds`	Number of cross validation folds
`numImputes`	number of imputations of missing responses
`useMCore`	Should multiple cores be used? See details

Details

For interval censored data, the loss function cannot be directly calculated. This is because the response variable is only observed up to an interval and so standard loss functions are not well defined (i.e. (10 - [7, 13])^2 is not a single number. Here the predicted value is 10, and the response is known to be between 7 and 13). To handle this, `icenReg_cv` uses multiple imputations; it first takes a posterior sample of the regression coefficients and conditional on these coefficients and the censored interval for each value, it draws a sample for each of the censored values. This process is repeated `numImputes` times to account for the uncertainty in the imputation.

The cross validation error will be computed using the user provided `loss_fun`. This function must take two vectors of numeric values; the first being the predicted values and the second being the true values. The default function is `abs_inv`, which we believe is well suited in most survival analysis settings. It should be noted that the predicted values used in `icenReg_cv` are the estimated median for a given set of covariates. In many cases, this may not necessarily be the prediction that minimizes the loss function, but it is often still a reasonable estimate.

The object `fit` must be an icenReg regression model, i.e. either a parametric model (via `ic_par`) or a semiparametric model (via `ic_sp`). We caution that the semi-parametric models appear to have heavier bias in estimating the out of sample bias, at least with the `abs_inv` loss function. The reason for this is that the semi-parametric models `only` assign probability mass to to *Turnbull intervals*, which in most cases will be strictly greater than 0, while most parametric models have possible probability on the real line. This biases both the imputations and the estimates of the semi-parametric model to not be close to 0, which is heavily pelanized by the `abs_inv` function. By both imputing no values close to 0 and predicting no values close to 0, the `ic_sp` model is being overly optimistic. In short, caution should be used when using the CV criteria to compare parametric and semi-parametric models.

Multiple cores can be used to speed up the cross validation process. This must be done with the `doParallel` package.

Author(s)

Clifford Anderson-Bergman

Examples

```
##Not run: somewhat computationally intense
# should not take more than a few minutes in worst case scenario

#library(splines)
#library(doParallel)
#simData <- simIC_weib(n = 1000)

# fit1 <- ic_par(cbind(l,u) ~ x1 + x2, data = simData)
     # True model

# fit2 <- ic_par(cbind(l,u) ~ x1 + x2, model = 'po', data = simData)
     # Model with wrong link function

# fit3 <- ic_par(cbind(l,u) ~ bs(x1, df = 6) + x2, data = simData)
     # Overfit model

# myCluster <- makeCluster(5, 'FORK')
# registerDoParellel(myCluster)
      #Setting up to use multiple cores

# icenReg_cv(fit1, useMCore = T)
# icenReg_cv(fit2, useMCore = T)
# icenReg_cv(fit3, useMCore = T)

# stopCluster(myCluster)
```

`ICNPMLE` *Computes the NPMLE for Univariate or Bivariate Interval Censored Data*

Description

Computes the MLE for a Interval Censored Data with a squeezing EM algorithm (*much* faster than the standard EM). Accepts either univariate interval censored data (where `times` is an n x 2 matrix

with `times[,1]` being the left side of the interval and `times[,2]` is the right side), or bivariate interval censored data (where `times` is an n x 4 matrix with `times[,1:2]` being left and right side of the interval for event 1 and `times[,3:4]` being the left and right side of the interval for event 2).

Usage

```
ICNPMLE(times, B = c(1,1), max.inner = 100, max.outer = 100, tol = 1e-10)
```

Arguments

`times`	either an n x 2 or n x 4 `data.frame` or `matrix` of censoring intervals
`B`	A vector indicating whether each end of the intervals are open (0) or closed (1). Alternatively, this could be an n x 2 or n x 4 matrix of indicators for each individual interval
`max.inner`	number of inner loops used in optimization algorithm
`max.outer`	number of outer loops used in optimization algorithm
`tol`	numerical tolerance

Author(s)

Clifford Anderson-Bergman

Also uses Marloes Maathius's `MLEcens::reduc` function for calculation of the clique matrix.

References

Anderson-Bergman, C., (2014) Semi- and non-parametric methods for interval censored data with shape constraints, Ph.D. Thesis

Yu, Y., (2010), Improved EM for Mixture Proportions with Applications to Nonparametric ML Estimation for Censored Data, *preprint*

Maathuis, M., (2005). Reduction algorithm for the NPMLE for the distribution function of bivariate interval censored data. *Journal of Computational and Graphical Statistics* Vol 14 pp 252\- 262

Examples

```
simData <- simBVCen(500)

fit <- ICNPMLE(simData)

fit
```

---

ic_par *Parametric Regression Models for Interval Censored Data*

Description

Fits a parametric regression model for interval censored data. Can fit either a Cox-PH model or a proportional odds 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" (Cox PH) or "po" (proportional odds)
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 is 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(Surv(l, u, type = 'interval2') ~ grp,
                        data = miceData, dist = 'loglogistic', model = 'po')

summary(logist_ph_fit)
summary(logist_po_fit)
```

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, seed = NULL,
      useGA = T, maxIter = 1000, baseUpdates = 5)
```

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)
`seed`	Seed for bootstrap. If `seed == NULL`, a random seed is still used. See details
`useGA`	Should a gradient ascent step be used in addition to an icm? See details
`maxIter`	Maximum iterations
`baseUpdates`	number of baseline updates (ICM + GA) per iteration

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`.

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.

It is very important to note that a random seed is *always* set if bs_samples > 0 (via `set.seed(seed)`), which can create problems in simulation studies if the same seed is set in every call to ic_sp during a simulation study. If `seed == NULL`, then the starting seed will be `round(runif(0, max = 10^8))`, which should be approximately equivalent to not setting a seed.

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

Examples

```
set.seed(1)

sim_data <- simIC_weib(n = 500, 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, type = 'FORK')
# 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", numImputes = 5)
```

Arguments

`fit`	model fit with `ic_par` or `ic_sp`
`newdata`	data.frame containing covariates and censored intervals. If blank, will use data from model
`imputeType`	type of imputation. See details for options
`numImputes`	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. `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.

NOTE: for the fully-parametric model, sampling the coefficients includes sampling the baseline parameters. However, for the semi-parametric model, the baseline parameters are **treated as fixed**, as we are unaware of any sort of asymptotic approximation of their distribution.

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)

imputedValues <- imputeCens(fit)
```

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.

Usage

```
data(miceData)
```

Format

A data frame with 144 rows and 3 variables

- `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)

Source

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)
```

optCliq *Computes the MLE for a Binary Mixture Model.*

Description

Computes the MLE for a Binary Mixture Model. Used internally for ICNPMLE, but may be useful in other problems. In the abstraction, solves the problem

$$\arg\max_{p} \sum_{i=1}^{n} \log\left(\sum_{j=1} p_j C_{ij}\right)$$

where C_{ij} is an indicator of whether the i-th observation could have come from the j-th source. C is referred to as the *clique matrix*.

Usage

```
optCliq(cliqMat, tol = 10^-10,
        inner_loops = 100, outer_loops = 20)
```

Arguments

`cliqMat`	n x m clique matrix. n = number of observations, m = number of components
`tol`	numerical tolerance
`inner_loops`	number of inner loops used
`outer_loops`	number of outer loops used

Examples

```
testData <- simBVCen()
#simulate bivariate interval censored data

cliqMat <- MLEcens::reduc(testData, cm = TRUE)$cm
#computes the cliqMat associated with data

cliqFit <- optCliq(cliqMat)
#optimizes the component weights for clique matrix

cliqFit
```

`predict.icenReg_fit` *Predictions from icenReg Regression Model*

Description

Gets various estimates from a `ic_par` or `ic_sp` 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 Cox-PH, 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(Surv(l, u, type = 'interval2') ~ x1 + x2,
                    data = simdata)

imputedValues <- predict(fit)
```

---

simBVCen *Simulates Bivariate Interval Censored Data*

Description

Simulates Bivariate Interval Censored Data

Usage

```
simBVCen(n = 1000)
```

Arguments

n	number of observations simulated

Examples

```
testData <- simBVCen()
#simulate bivariate interval censored data

bvcenFit <- ICNPMLE(testData)

bvcenFit
```

`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 = 0.5)
```

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
```