Package 'discSurv'

April 16, 2019

Version 1.4.0

Title Discrete Time Survival Analysis

Date 2019-04-16

Author Thomas Welchowski <welchow@imbie.meb.uni-bonn.de> and Matthias Schmid <matthias.schmid@imbie.uni-bonn.de>

Maintainer Thomas Welchowski <welchow@imbie.meb.uni-bonn.de>

Description Provides data transformations, estimation utilities, predictive evaluation measures and simulation functions for discrete time survival analysis.

Imports functional, mvtnorm, mgcv, data.table

Suggests Matrix, matrixcalc, numDeriv, caret, Ecdat, pec, survival, randomForest, nnet, VGAM

License GPL-3

LazyLoad yes

NeedsCompilation no

Repository CRAN

Date/Publication 2019-04-16 20:12:52 UTC

R topics documented:

`discSurv-package` *Discrete Survival Analysis*

Description

Includes functions for data transformations, estimation, evaluation and simulation of discrete survival analysis. Also discrete life table estimates are available. The most important functions are listed below:

- `contToDisc`: Discretizes continuous time variable into a specified grid of censored data for discrete survival analysis.
- `dataLong`: Transform data from short format into long format for discrete survival analysis and right censoring.
- `dataLongCompRisks`: Transforms short data format to long format for discrete survival modelling in the case of competing risks with right censoring.
- `dataLongTimeDep`: Transforms short data format to long format for discrete survival modelling of single event analysis with right censoring.
- `concorIndex`: Calculates the concordance index for discrete survival models (independent measure of time).
- `simCompRisk`: Simulates responses and covariates of discrete competing risk models.
- `dataLongSubDist`: Converts the data to long format suitable for applying discrete subdistribution hazard modelling (competing risks).

Details

Package:	discSurv
Type:	Package
Version:	1.4.0
Date:	2019-04-16
License:	GPL-3

Author(s)

Thomas Welchowski <welchow@imbie.meb.uni-bonn.de> (Maintainer)

Matthias Schmid <matthias.schmid@imbie.uni-bonn.de>

References

Main references:

Matthias Schmid, Gerhard Tutz and Thomas Welchowski, (2017), *Discrimination Measures for Discrete Time-to-Event Predictions*, Econometrics and Statistics, Elsevier, Doi: 10.1016/j.ecosta.2017.03.008

Gerhard Tutz and Matthias Schmid, (2016), *Modeling discrete time-to-event data*, Springer series in statistics, Doi: 10.1007/978-3-319-28158-2

Further references:

Gerhard Tutz, (2012), *Regression for Categorical Data*, Cambridge University Press

Hajime Uno and Tianxi Cai and Lu Tian and L. J. Wei, (2007), *Evaluating Prediction Rules for t-Year Survivors With Censored Regression Models*, Journal of the American Statistical Association

Gerds T. A. and M. Schumacher, (2006), *Consistent estimation of the expected Brier score in general survival models with right-censored event times*, Biometrical Journal, Vol. 48, No. 6, pages 1029-1040

Roger B. Nelsen, (2006), *An introduction to Copulas*, Springer Science+Business Media, Inc.

Patrick J. Heagerty and Yingye Zheng, (2005), *Survival Model Predictive Accuracy and ROC Curves*, Biometrics 61, 92-105

Steele Fiona and Goldstein Harvey and Browne William, (2004), *A general multilevel multistate competing risks model for event history data* Statistical Modelling, volume 4, pages 145-159

Jerald F. Lawless, (2000), *Statistical Models and Methods for Lifetime Data, 2nd edition*, Wiley series in probability and statistics

Jason P. Fine and Robert J. Gray, (1999), *A proportional hazards model for the subdistribution of a competing risk*, Journal of the American statistical association, Vol. 94, No. 446, pages 496-509

Ludwig Fahrmeir, (1997), *Discrete failure time models*, LMU Sonderforschungsbereich 386, Paper 91, http://epub.ub.uni-muenchen.de/

Tilmann Gneiting and Adrian E. Raftery, (2007), *Strictly proper scoring rules, prediction, and estimation*, Journal of the American Statistical Association 102 (477), 359-376

M. J. van der Laan and J. M. Robins, (2003), *Unified Methods for Censored Longitudinal Data and Causality*, Springer, New York

Wiji Narendranathan and Mark B. Stewart, (1993), *Modelling the probability of leaving unemployment: competing risks models with flexible base-line hazards*, Applied Statistics, pages 63-83

W. A. Thompson Jr., (1977), *On the Treatment of Grouped Observations in Life Studies*, Biometrics, Vol. 33, No. 3

William H. Kruskal, (1958), *Ordinal Measures of Association*, Journal of the American Statistical Association, Vol. 53, No. 284, pages 814-861

`adjDevResid` *Adjusted Deviance Residuals*

Description

Calculates the adjusted deviance residuals. This function only supports generalized, linear models. The adjusted deviance residuals should be approximately normal distributed, in the case of a well fitting model.

Usage

```
adjDevResid(dataSet, survModelFormula, censColumn, linkFunc = "logit", idColumn = NULL)
```

Arguments

dataSet	Original data in short format. Must be of class "data.frame".
survModelFormula	Gives the specified relationship of discrete response and covariates. The formula is designed, that the intercepts for the time dependent base line hazards are always included. Therefore only covariates should be given in this formula. This argument is required to be of class "formula".
censColumn	Gives the column name of the event indicator (1=observed, 0=censored). Must be of type "character".
linkFunc	Specifies the desired link function in use of generalized, linear models.
idColumn	Gives the column name of the identification number of each person. The argument must be of type "character". Default NULL means, that each row equals one person (no repeated measurements).

Value

- Output: List with objects:
  - AdjDevResid: Adjusted deviance residuals as numeric vector.
  - GlmFit: Fit object of class (generalized, linear model used in the calculations)
- Input: A list of given argument input values (saved for reference)

Author(s)

Thomas Welchowski <welchow@imbie.meb.uni-bonn.de>

References

Gerhard Tutz and Matthias Schmid, (2016), *Modeling discrete time-to-event data*, Springer series in statistics, Doi: 10.1007/978-3-319-28158-2

Gerhard Tutz, (2012), *Regression for Categorical Data*, Cambridge University Press

See Also

devResid, brierScore, glm, predErrDiscShort

Examples

```
# Example with cross validation and unemployment data
library(Ecdat)
data(UnempDur)
summary(UnempDur$spell)

# Extract subset of data
set.seed(635)
IDsample <- sample(1:dim(UnempDur)[1], 100)
UnempDurSubset <- UnempDur [IDsample, ]

# Calculate adjusted deviance residuals for the unemployment data subset
```

```
adjDevianceResiduals <- adjDevResid (dataSet=UnempDurSubset,
survModelFormula=spell ~ age + logwage,
censColumn="censor1", linkFunc="logit", idColumn=NULL)
adjDevianceResiduals

# Exclude outliers
adjDevResidWoOut <- adjDevianceResiduals$Output$AdjDevResid [
adjDevianceResiduals$Output$AdjDevResid <
quantile(adjDevianceResiduals$Output$AdjDevResid, prob=0.9) &
adjDevianceResiduals$Output$AdjDevResid >
quantile(adjDevianceResiduals$Output$AdjDevResid, prob=0.1)]

# Compare nonparametric density estimate of adjusted deviance residuals
# with adapted normal distribution
plot(density(adjDevResidWoOut), xlim=c(-10,10),
main="Density comparison: Normal vs nonparametric estimate", lwd=2, las=1, col="red")
dnorm1 <- function (x) {dnorm (x, mean=mean(adjDevResidWoOut), sd=sd(adjDevResidWoOut))}
curve (dnorm1, n=500, add=TRUE, col="black", lwd=2)
legend("topright", legend=c("Normal", "Nonpar"), lty=1, lwd=2, col=c("black", "red"))
```

`adjDevResidShort` *Adjusted Deviance Residuals in short format*

Description

Calculates the adjusted deviance residuals for arbitrary prediction models. The adjusted deviance residuals should be approximately normal distributed, in the case of a well fitting model.

Usage

```
adjDevResidShort(dataSet, hazards)
```

Arguments

`dataSet`	Data set in long format. Must be of class "data.frame".
`hazards`	Estimated hazard rates of the data in long format. Hazard rates are probabilities and therefore restricted to the interval [0, 1]

Value

- Output: List with objects:
  - AdjDevResid: Adjusted deviance residuals as numeric vector
- Input: A list of given argument input values (saved for reference)

Note

The argument *dataSet* must have a response with column name "y". The correct format of the dataset can be augmented by using `dataLong`.

Author(s)

Thomas Welchowski <welchow@imbie.meb.uni-bonn.de>

References

Gerhard Tutz and Matthias Schmid, (2016), *Modeling discrete time-to-event data*, Springer series in statistics, Doi: 10.1007/978-3-319-28158-2

Gerhard Tutz, (2012), *Regression for Categorical Data*, Cambridge University Press

See Also

devResidShort, predErrDiscShort

Examples

```
library(survival)

# Transform data to long format
heart[, "stop"] <- ceiling(heart[, "stop"])
set.seed(0)
Indizes <- sample(unique(heart$id), 25)
randSample <- heart[unlist(sapply(1:length(Indizes),
function(x) which(heart$id==Indizes[x]))),]
heartLong <- dataLongTimeDep(dataSet=randSample,
timeColumn="stop", censColumn="event", idColumn="id", timeAsFactor=FALSE)

# Fit a generalized, additive model and predict hazard rates on data in long format
library(mgcv)
gamFit <- gam(y ~ timeInt + surgery + transplant + s(age), data=heartLong, family="binomial")
hazPreds <- predict(gamFit, type="response")

# Calculate adjusted deviance residuals
devResiduals <- adjDevResidShort (dataSet=heartLong, hazards=hazPreds)$Output$AdjDevResid
devResiduals
```

---

aucUno *Area under the Curve Estimation*

Description

Estimates the time dependent area under the curve given calculated true positive rate and false positive rate. Both objects ("tprObj", "fprObj") and must have identical input arguments, e. g. same relationship of discrete response and covariates and supplied data sources. The values should be above 0.5 for a well predicting model, because random guessing would get this score.

Usage

```
aucUno(tprObj, fprObj)
```

Arguments

tprObj	Object of class "discSurvTprUno". See function tprUno
fprObj	Object of class "discSurvFprUno". See function fprUno

Details

The auc is estimated by numerical integration of the ROC curve.

Value

- Output: A list with objects:
  - Output: Named numeric vector with auc value
  - Input: A list of given argument input values (saved for reference)

Author(s)

Thomas Welchowski <welchow@imbie.meb.uni-bonn.de>

Matthias Schmid <matthias.schmid@imbie.uni-bonn.de>

References

Matthias Schmid, Gerhard Tutz and Thomas Welchowski, (2017), *Discrimination Measures for Discrete Time-to-Event Predictions, Econometrics and Statistics*, Elsevier, DOI: 10.1016/j.ecosta.2017.03.008

Hajime Uno and Tianxi Cai and Lu Tian and L. J. Wei, (2007), *Evaluating Prediction Rules for t-Year Survivors With Censored Regression Models*, Journal of the American Statistical Association

Patrick J. Heagerty and Yingye Zheng, (2005), *Survival Model Predictive Accuracy and ROC Curves*, Biometrics 61, 92-105

See Also

tprUno, fprUno

Examples

```
#####################################################
# Example with cross validation and unemployment data

library(Ecdat)
library(caret)
data(UnempDur)
summary(UnempDur$spell)

# Extract subset of data
set.seed(635)
IDsample <- sample(1:dim(UnempDur)[1], 100)
UnempDurSubset <- UnempDur [IDsample, ]
head(UnempDurSubset)
range(UnempDurSubset$spell)
set.seed(7550)
```

```
CVfolds <- createFolds (y=UnempDurSubset$spell, returnTrain=TRUE, k=2)

# Estimate true positive rate of time interval 7:
# Correspondes to three and a half month duration (each interval is of length two weeks)
tryTPR <- tprUno (timepoint=7, dataSet=UnempDurSubset, trainIndices=CVfolds,
survModelFormula=spell ~ age + logwage, censModelFormula=censor1 ~ 1,
linkFunc="logit", idColumn=NULL, timeAsFactor=FALSE)
tryTPR
plot(tryTPR)

# Estimate false positive rate of time interval 7:
tryFPR <- fprUno (timepoint=7, dataSet=UnempDurSubset, trainIndices=CVfolds,
survModelFormula=spell ~ age + logwage, censModelFormula=censor1 ~ 1,
linkFunc="logit", idColumn=NULL, timeAsFactor=FALSE)
tryFPR
plot(tryFPR)

# Estimate auc
tryAUC <- aucUno (tprObj=tryTPR, fprObj=tryFPR)
tryAUC
plot(tryAUC)

#####################################
# Example National Wilm's Tumor Study

library(survival)
head(nwtco)
summary(nwtco$rel)

# Select subset
set.seed(-375)
Indices <- sample(1:dim(nwtco)[1], 500)
nwtcoSub <- nwtco [Indices, ]

# Convert time range to 30 intervals
intLim <- quantile(nwtcoSub$edrel, prob=seq(0, 1, length.out=30))
intLim [length(intLim)] <- intLim [length(intLim)] + 1
nwtcoSubTemp <- contToDisc(dataSet=nwtcoSub, timeColumn="edrel", intervalLimits=intLim)
nwtcoSubTemp$instit <- factor(nwtcoSubTemp$instit)
nwtcoSubTemp$histol <- factor(nwtcoSubTemp$histol)
nwtcoSubTemp$stage <- factor(nwtcoSubTemp$stage)

# Split in training and test sample
set.seed(-570)
TrainingSample <- sample(1:dim(nwtcoSubTemp)[1], round(dim(nwtcoSubTemp)[1]*0.75))
nwtcoSubTempTrain <- nwtcoSubTemp [TrainingSample, ]
nwtcoSubTempTest <- nwtcoSubTemp [-TrainingSample, ]

# Convert to long format
nwtcoSubTempTrainLong <- dataLong(dataSet=nwtcoSubTempTrain,
timeColumn="timeDisc", censColumn="rel")

# Estimate glm
```

```
inputFormula <- y ~ timeInt + histol + instit + stage
glmFit <- glm(formula=inputFormula, data=nwtcoSubTempTrainLong, family=binomial())
linPreds <- predict(glmFit, newdata=cbind(nwtcoSubTempTest,
timeInt=nwtcoSubTempTest$timeDisc))

# Estimate tpr given one training and one test sample at time interval 5
tprFit <- tprUnoShort (timepoint=5, marker=linPreds,
newTime=nwtcoSubTempTest$timeDisc, newEvent=nwtcoSubTempTest$rel,
trainTime=nwtcoSubTempTrain$timeDisc, trainEvent=nwtcoSubTempTrain$rel)

# Estimate fpr given one training and one test sample at time interval 5
fprFit <- fprUnoShort (timepoint=5, marker=linPreds,
newTime=nwtcoSubTempTest$timeDisc)

# Estimate auc
tryAUC <- aucUno (tprObj=tprFit, fprObj=fprFit)
tryAUC
plot(tryAUC)
```

`brierScore` *Brier Score Calculation*

Description

Computes the brier score of each person based on a generalized, linear model with cross validation.

Usage

```
brierScore(dataSet, trainIndices, survModelFormula,
linkFunc = "logit", censColumn, idColumn = NULL)
```

Arguments

`dataSet`	Original data in short format. Must be of class "data.frame".
`trainIndices`	Must be a list of indices used for training data sets. E. g. if a 10-fold cross validation should be conducted, then the list should have length 10 and in each element a integer vector of training indices.
`survModelFormula`	Gives the specified relationship of discrete response and covariates. The formula is designed, that the intercepts for the time dependent base line hazards are always included. Therefore only covariates should be given in this formula. This argument is required to be of class "formula".
`linkFunc`	Specifies the desired link function in use of generalized, linear models.
`censColumn`	Gives the column name of the event indicator (1=observed, 0=censored). Must be of type "character".
`idColumn`	Gives the column name of the identification number of each person. The argument must be of type "character". Default NULL means, that each row equals one person (no repeated measurements).

Details

At normal circumstances a covariate free model for the censoring weights is sufficient.

Value

Numeric vector of brier scores for each person in the original data.

Note

It is assumed that all time points up to the last interval [a_q, Inf) are available in short format. If not already present, these can be added manually.

Author(s)

Thomas Welchowski <welchow@imbie.meb.uni-bonn.de>

Matthias Schmid <matthias.schmid@imbie.uni-bonn.de>

References

Gerhard Tutz and Matthias Schmid, (2016), *Modeling discrete time-to-event data*, Springer series in statistics, Doi: 10.1007/978-3-319-28158-2

Gerds T. A. and M. Schumacher, (2006), *Consistent estimation of the expected Brier score in general survival models with right-censored event times*, Biometrical Journal, Vol. 48, No. 6, pages 1029-1040

See Also

adjDevResid, devResid, glm, predErrDiscShort

Examples

```
# Example with cross validation and unemployment data
library(Ecdat)
library(caret)
data(UnempDur)
summary(UnempDur$spell)

# Extract subset of data
set.seed(635)
IDsample <- sample(1:dim(UnempDur)[1], 100)
UnempDurSubset <- UnempDur [IDsample, ]
head(UnempDurSubset)
range(UnempDurSubset$spell)
set.seed(7550)
CVfolds <- createFolds (y=UnempDurSubset$spell, returnTrain=TRUE, k=2)

# Calculate brier score
tryBrierScore <- brierScore (dataSet=UnempDurSubset, trainIndices=CVfolds,
survModelFormula=spell ~ age + logwage, linkFunc="logit",
censColumn="censor1", idColumn=NULL)
tryBrierScore
```

concorIndex *Concordance Index*

Description

Calculates the concordance index for discrete survival models (independent measure of time). This is the probability that, for a pair of randomly chosen comparable samples, the sample with the higher risk prediction will experience an event before the other sample or belongs to a higher binary class.

Usage

```
concorIndex(aucObj, printTimePoints=FALSE)
```

Arguments

aucObj	Object of class "discSurvAucUno". This object is created using the function aucUno
printTimePoints	Should messages be printed for each calculation of a discrete time interval? (logical scalar) Default is FALSE.

Details

The algorithm extracts all necessary information of the auc object (e. g. marginal probabilities and survival functions).

Value

List with objects

- Output: Concordance index (named numeric vector)
- Input: List with all input arguments (saved for reference)

Note

It is assumed that all time points up to the last observed interval [a_q-1, a_q) are available. If not already present, these can be added manually.

Author(s)

Thomas Welchowski <welchow@imbie.meb.uni-bonn.de>

Matthias Schmid <matthias.schmid@imbie.uni-bonn.de>

References

Matthias Schmid, Gerhard Tutz and Thomas Welchowski, (2017), *Discrimination Measures for Discrete Time-to-Event Predictions, Econometrics and Statistics*, Elsevier, DOI: 10.1016/j.ecosta.2017.03.008

Hajime Uno and Tianxi Cai and Lu Tian and L. J. Wei, (2007), *Evaluating Prediction Rules for t-Year Survivors With Censored Regression Models*, Journal of the American Statistical Association

Patrick J. Heagerty and Yingye Zheng, (2005), *Survival Model Predictive Accuracy and ROC Curves*, Biometrics 61, 92-105

See Also

aucUno

Examples

```
# Example with cross validation and unemployment data
library(Ecdat)
library(caret)
data(UnempDur)

# Evaluation of short term prediction for re-employed at full-time job
# Last interval q=14
# -> Set all time points with spell > 13 to time interval 13 and censored
lastObsInterval <- 13
UnempDurSubset <- UnempDur
UnempDurSubset[UnempDurSubset$spell > lastObsInterval, "censor1"] <- 0
UnempDurSubset[UnempDurSubset$spell > lastObsInterval, "spell"] <- lastObsInterval
head(UnempDurSubset)
range(UnempDurSubset$spell)

# Select cross-validation samples
set.seed(7550)
CVfolds <- createFolds (y=UnempDurSubset$spell, returnTrain=TRUE, k=2)

# Continuation ratio model formula
contModForm <- spell ~ logwage + ui + logwage*ui + age

# Estimate true positive rate of time interval 6:
# Correspondes to three and a half month duration (each interval is of length two weeks)
tryTPR <- tprUno (timepoint=6, dataSet=UnempDurSubset, trainIndices=CVfolds,
survModelFormula=contModForm, censModelFormula=censor1 ~ 1,
linkFunc="logit", idColumn=NULL, timeAsFactor=FALSE)
tryTPR
plot(tryTPR)

# Estimate false positive rate of time interval 6:
tryFPR <- fprUno (timepoint=6, dataSet=UnempDurSubset, trainIndices=CVfolds,
survModelFormula=contModForm, censModelFormula=censor1 ~ 1,
linkFunc="logit", idColumn=NULL, timeAsFactor=FALSE)
tryFPR
plot(tryFPR)
```

```
# Estimate AUC rate of time interval 6:
tryAUC <- aucUno (tprObj=tryTPR, fprObj=tryFPR)
tryAUC
plot(tryAUC)

## Not run: # Estimate global concordance index:
tryConcorIndex <- concorIndex (tryAUC, printTimePoints=TRUE)
tryConcorIndex
summary(tryConcorIndex)
## End(Not run)
```

`contToDisc` *Continuous to Discrete Transformation*

Description

Discretizes continuous time variable into a specified grid of censored data for discrete survival analysis. It is a data preprocessing step, before the data can be extendend in long format and further analysed with discrete survival models.

Usage

```
contToDisc(dataSet, timeColumn, intervalLimits, equi=FALSE)
```

Arguments

`dataSet`	Original data in short format. Must be of class "data.frame".
`timeColumn`	Name of the column with discrete survival times. Must be a scalar character value.
`intervalLimits`	Numeric vector of the right interval borders, e. g. if the intervals are [0, a_1), [a_1, a_2), [a_2, a_max), then intervalLimits = c(a_1, a_2, a_max)
`equi`	Specifies if argument *intervalLimits* should be interpreted as number of equidistant intervals. Logical only TRUE or FALSE is allowed.

Value

Gives the data set expanded with a first column "timeDisc". This column includes the discrete time intervals (ordered factor).

Note

In discrete survival analysis the survival times have to be categorized in time intervals. Therefore this function is required, if there are observed continuous survival times.

Author(s)

Thomas Welchowski <welchow@imbie.meb.uni-bonn.de>

References

Gerhard Tutz and Matthias Schmid, (2016), *Modeling discrete time-to-event data*, Springer series in statistics, Doi: 10.1007/978-3-319-28158-2

See Also

dataLong, dataLongTimeDep, dataLongCompRisks

Examples

```
# Example copenhagen stroke study data
library(pec)
data(cost)
head(cost)

# Convert observed times to months
# Right borders of intervals [0, a_1), [a_1, a_2), ... , [a_{\max-1}, a_{\max})
IntBorders <- 1:ceiling(max(cost$time)/30)*30

# Select subsample
subCost <- cost [1:100, ]
CostMonths <- contToDisc (dataSet=subCost, timeColumn="time", intervalLimits=IntBorders)
head(CostMonths)

# Select subsample giving number of equidistant intervals
CostMonths <- contToDisc (dataSet=subCost, timeColumn="time", intervalLimits=10, equi=TRUE)
head(CostMonths)
```

---

dataCensoring *Data Censoring Transformation*

Description

Function for transformation of discrete survival times in censoring encoding. Prior this function the data has to be already transformed to long format. With this new generated variable, the discrete censoring process can be analysed instead of the discrete survival process. In discrete survival analysis this information is used to constructs weights for predictive evaluation measures. It is applicable in single event survival analysis.

Usage

```
dataCensoring(dataSetLong, respColumn, timeColumn)
```

Arguments

dataSetLong	Original data in transformed long format.
respColumn	Name of column of discrete survival response (character scalar).
timeColumn	Name of column of discrete time intervals (character scalar).

Details

The standard procedure is to use functions such as dataLong, dataLongTimeDep, dataLongCompRisks to augment the data set from short format to long format before using dataCensoring.

Value

Original data set as argument *dataSetLong*, but with added censoring process as first variable in column "yCens"

Author(s)

Thomas Welchowski <welchow@imbie.meb.uni-bonn.de>

References

Gerhard Tutz and Matthias Schmid, (2016), *Modeling discrete time-to-event data*, Springer series in statistics, Doi: 10.1007/978-3-319-28158-2

Ludwig Fahrmeir, (1997), *Discrete failure time models*, LMU Sonderforschungsbereich 386, Paper 91, http://epub.ub.uni-muenchen.de/

W. A. Thompson Jr., (1977), *On the Treatment of Grouped Observations in Life Studies*, Biometrics, Vol. 33, No. 3

See Also

dataCensoringShort, contToDisc, dataLong, dataLongTimeDep, dataLongCompRisks

Examples

```
library(pec)
data(cost)
head(cost)
IntBorders <- 1:ceiling(max(cost$time)/30)*30
subCost <- cost [1:100, ]

# Convert from days to months
CostMonths <- contToDisc (dataSet=subCost, timeColumn="time", intervalLimits=IntBorders)
head(CostMonths)

# Convert to long format based on months
CostMonthsLong <- dataLong (dataSet=CostMonths, timeColumn="timeDisc", censColumn="status")
head(CostMonthsLong, 20)

# Generate censoring process variable
CostMonthsCensor <- dataCensoring (dataSetLong=CostMonthsLong,
respColumn="y", timeColumn="timeInt")
head(CostMonthsCensor)
tail(CostMonthsCensor [CostMonthsCensor$obj==1, ], 10)
tail(CostMonthsCensor [CostMonthsCensor$obj==3, ], 10)
```

dataCensoringShort *Data Censoring Transformation for short formats*

Description

Function for transformation of discrete survival times in censoring encoding. In contrast to dataCensoring this function needs the original data in short format as argument. With the new generated variable "yCens", the discrete censoring process can be analyzed instead of the discrete survival process. In discrete survival analysis this information is used to constructs weights for predictive evaluation measures. It is applicable in single event survival analysis.

Usage

```
dataCensoringShort(dataSet, eventColumns, timeColumn)
```

Arguments

dataSet	Original data in transformed long format.
eventColumns	Name of columns of event columns (character vector). The event columns have to be in binary format. If the sum of all events equals zero in a row, then this observation is interpreted as censored.
timeColumn	Name of column with discrete time intervals (character scalar).

Value

Original data set as argument *dataSet*, but with added censoring process as first variable in column "yCens".

Author(s)

Thomas Welchowski <welchow@imbie.meb.uni-bonn.de>

References

Gerhard Tutz and Matthias Schmid, (2016), *Modeling discrete time-to-event data*, Springer series in statistics, Doi: 10.1007/978-3-319-28158-2

Ludwig Fahrmeir, (1997), *Discrete failure time models*, LMU Sonderforschungsbereich 386, Paper 91, http://epub.ub.uni-muenchen.de/

W. A. Thompson Jr., (1977), *On the Treatment of Grouped Observations in Life Studies*, Biometrics, Vol. 33, No. 3

See Also

dataCensoring, contToDisc, dataLong, dataLongTimeDep, dataLongCompRisks

Examples

```
library(pec)
data(cost)
head(cost)
IntBorders <- 1:ceiling(max(cost$time)/30)*30
subCost <- cost [1:100, ]

# Convert from days to months
CostMonths <- contToDisc (dataSet=subCost, timeColumn="time", intervalLimits=IntBorders)
head(CostMonths)

# Generate censoring process variable in short format
CostMonthsCensorShort <- dataCensoringShort (dataSet=CostMonths,
eventColumns="status", timeColumn="time")
head(CostMonthsCensorShort)
```

---

dataLong *Data Long Transformation*

Description

Transform data from short format into long format for discrete survival analysis and right censoring. Data is assumed to include no time varying covariates, e. g. no follow up visits are allowed. It is assumed that the covariates stay constant over time, in which no information is available.

Usage

```
dataLong(dataSet, timeColumn, censColumn, timeAsFactor=TRUE,
remLastInt=FALSE, aggTimeFormat=FALSE, lastTheoInt=NULL)
```

Arguments

dataSet	Original data in short format. Must be of class "data.frame".
timeColumn	Character giving the column name of the observed times. It is required that the observed times are discrete (integer).
censColumn	Character giving the column name of the event indicator. It is required that this is a binary variable with 1=="event" and 0=="censored".
timeAsFactor	Should the time intervals be coded as factor? Default is to use factor. If the argument is false, the column is coded as numeric.
remLastInt	Should the last theoretical interval be removed in long format? Default is no deletion. This is only important, if the short format data includes the last theoretic interval [a_q, Inf). There are only events in the last theoretic interval, so the hazard is always one and these observations have to be excluded for estimation.
aggTimeFormat	Instead of the usual long format, should every obseration have all time intervals? (logical scalar) Default is standard long format. In the case of nonlinear risk score models, the time effect has to be integrated out before these can be applied to the C-index concorIndex.

`lastTheoInt` Gives the number of the last theoretic interval (integer scalar). Only used, if aggTimeFormat==TRUE.

Details

If the data has continuous survival times, the response may be transformed to discrete intervals using function `contToDisc`. If the data set has time varying covariates the function `dataLongTimeDep` should be used instead. In the case of competing risks and no time varying covariates see function `dataLongCompRisks`.

Value

Original data.frame with three additional columns:

- obj: Index of persons as integer vector
- timeInt: Index of time intervals (factor)
- y: Response in long format as binary vector. 1=="event happens in period timeInt" and 0 otherwise

Author(s)

Thomas Welchowski <welchow@imbie.meb.uni-bonn.de>

Matthias Schmid <matthias.schmid@imbie.uni-bonn.de>

References

Gerhard Tutz and Matthias Schmid, (2016), *Modeling discrete time-to-event data*, Springer series in statistics, Doi: 10.1007/978-3-319-28158-2

Ludwig Fahrmeir, (1997), *Discrete failure time models*, LMU Sonderforschungsbereich 386, Paper 91, http://epub.ub.uni-muenchen.de/

W. A. Thompson Jr., (1977), *On the Treatment of Grouped Observations in Life Studies*, Biometrics, Vol. 33, No. 3

See Also

`contToDisc`, `dataLongTimeDep`, `dataLongCompRisks`

Examples

```
# Example unemployment data
library(Ecdat)
data(UnempDur)

# Select subsample
subUnempDur <- UnempDur [1:100, ]
head(subUnempDur)

# Convert to long format
UnempLong <- dataLong (dataSet=subUnempDur, timeColumn="spell", censColumn="censor1")
```

```
head(UnempLong, 20)

# Is there exactly one observed event of y for each person?
splitUnempLong <- split(UnempLong, UnempLong$obj)
all(sapply(splitUnempLong, function (x) sum(x$y))==subUnempDur$censor1) # TRUE

# Second example: Acute Myelogenous Leukemia survival data
library(survival)
head(leukemia)
leukLong <- dataLong (dataSet=leukemia, timeColumn="time", censColumn="status")
head(leukLong, 30)

# Estimate discrete survival model
estGlm <- glm(formula=y ~ timeInt + x, data=leukLong, family=binomial())
summary(estGlm)

# Estimate survival curves for non-maintained chemotherapy
newDataNonMaintained <- data.frame(timeInt=factor(1:161), x=rep("Nonmaintained"))
predHazNonMain <- predict(estGlm, newdata=newDataNonMaintained, type="response")
predSurvNonMain <- cumprod(1-predHazNonMain)

# Estimate survival curves for maintained chemotherapy
newDataMaintained <- data.frame(timeInt=factor(1:161), x=rep("Maintained"))
predHazMain <- predict(estGlm, newdata=newDataMaintained, type="response")
predSurvMain <- cumprod(1-predHazMain)

# Compare survival curves
plot(x=1:50, y=predSurvMain [1:50], xlab="Time", ylab="S(t)", las=1,
type="l", main="Effect of maintained chemotherapy on survival of leukemia patients")
lines(x=1:161, y=predSurvNonMain, col="red")
legend("topright", legend=c("Maintained chemotherapy", "Non-maintained chemotherapy"),
col=c("black", "red"), lty=rep(1, 2))
# The maintained therapy has clearly a positive effect on survival over the time range
```

`dataLongCompRisks` *Data Long Competing Risks Transformation*

Description

Transforms short data format to long format for discrete survival modelling in the case of competing risks with right censoring. It is assumed that the covariates are not time varying.

Usage

```
dataLongCompRisks(dataSet, timeColumn, eventColumns,
eventColumnsAsFactor=FALSE, timeAsFactor=TRUE)
```

Arguments

dataSet	Original data in short format. Must be of class "data.frame".
timeColumn	Character giving the column name of the observed times. It is required that the observed times are discrete (integer).
eventColumns	Character vector giving the column names of the event indicators (excluding censoring column). It is required that all events are binary encoded. If the sum of all event indicators is zero, then this is interpreted as a censored observation. Alternatively a column name of a factor representing competing events can be given. In this case the argument "eventColumnsAsFactor" has to be set TRUE and the first level is assumed to represent censoring.
eventColumnsAsFactor	Should the argument "eventColumns" be intepreted as column name of a factor variable(logical scalar)? Default is FALSE.
timeAsFactor	Should the time intervals be coded as factor? Default is to use factor. If the argument is false, the column is coded as numeric.

Details

It is assumed, that only one event happens at a specific time point (competing risks). Either the observation is censored or one of the possible events takes place.

Value

Original data set in long format with additional columns

- obj: Gives identification number of objects (row index in short format) (integer)
- timeInt: Gives number of discrete time intervals (factor)
- responses: Columns with dimension count of events + 1 (censoring)
  - e0: No event (observation censored in specific interval)
  - e1: Indicator of first event, 1 if event takes place and 0 otherwise
  - ... ...
  - ek: Indicator of last k-th event, 1 if event takes place and 0 otherwise

Author(s)

Thomas Welchowski <welchow@imbie.meb.uni-bonn.de>

References

Gerhard Tutz and Matthias Schmid, (2016), *Modeling discrete time-to-event data*, Springer series in statistics, Doi: 10.1007/978-3-319-28158-2

Steele Fiona and Goldstein Harvey and Browne William, (2004), *A general multilevel multistate competing risks model for event history data* Statistical Modelling, volume 4, pages 145-159

Wiji Narendranathan and Mark B. Stewart, (1993), *Modelling the probability of leaving unemployment: competing risks models with flexible base-line hazards*, Applied Statistics, pages 63-83

See Also

contToDisc, dataLongTimeDep, dataLongCompRisksTimeDep

Examples

```
# Example with unemployment data
library(Ecdat)
data(UnempDur)

# Select subsample
SubUnempDur <- UnempDur [1:100, ]

# Convert competing risk data to long format
SubUnempDurLong <- dataLongCompRisks (dataSet=SubUnempDur, timeColumn="spell",
eventColumns=c("censor1", "censor2", "censor3", "censor4"))
head(SubUnempDurLong, 20)

# Fit multinomial logit model with VGAM package
# with one coefficient per response
library(VGAM)
multLogitVGM <- vgam(cbind(e0, e1, e2, e3, e4) ~ timeInt + ui + age + logwage,
                     family=multinomial(refLevel=1),
                     data = SubUnempDurLong)
coef(multLogitVGM)

# Alternative: Use nnet
# Convert response to factor
rawResponseMat <- SubUnempDurLong[, c("e0", "e1", "e2", "e3", "e4")]
NewFactor <- factor(unname(apply(rawResponseMat, 1, function(x) which(x == 1))),
                     labels = colnames(rawResponseMat))

# Include recoded response in data
SubUnempDurLong <- cbind(SubUnempDurLong, NewResp=NewFactor)

# Construct formula of mlogit model
mlogitFormula <- formula(NewResp ~ timeInt + ui + age + logwage)

# Fit multinomial logit model
# with one coefficient per response
library(nnet)
multLogitNNET <- multinom(formula=mlogitFormula, data=SubUnempDurLong)
coef(multLogitNNET)
```

---

dataLongCompRisksTimeDep *Data Long Competing Risks Time Dependent Covariates Transformation*

Description

Transforms short data format to long format for discrete survival modelling in the case of competing risks with right censoring. Covariates may vary over time.

Usage

```
dataLongCompRisksTimeDep(dataSet, timeColumn, eventColumns,
eventColumnsAsFactor=FALSE, idColumn, timeAsFactor=TRUE)
```

Arguments

`dataSet`	Original data in short format. Must be of class "data.frame".
`timeColumn`	Character giving the column name of the observed times. It is required that the observed times are discrete (integer).
`eventColumns`	Character vector giving the column names of the event indicators (excluding censoring column). It is required that all events are binary encoded. If the sum of all event indicators is zero, then this is interpreted as a censored observation. Alternatively a column name of a factor representing competing events can be given. In this case the argument "eventColumnsAsFactor" has to be set TRUE and the first level is assumed to represent censoring.
`eventColumnsAsFactor`	Should the argument eventColumns be intepreted as column name of a factor variable(logical scalar)? Default is FALSE.
`idColumn`	Name of column of identification number of persons as character.
`timeAsFactor`	Should the time intervals be coded as factor? Default is to use factor. If the argument is false, the column is coded as numeric.

Details

There may be some intervals, where no additional information on the covariates is observed (e. g. observed values in interval one and three but two is missing). In this case it is assumed, that the values from the last observation stay constant over time until a new measurement was done.

Value

Original data set in long format with additional columns

- obj: Gives identification number of objects (row index in short format) (integer)
- timeInt: Gives number of discrete time intervals (factor)
- responses: Columns with dimension count of events + 1 (censoring)
  - e0: No event (observation censored in specific interval)
  - e1: Indicator of first event, 1 if event takes place and 0 otherwise
  - ... ...
  - ek: Indicator of last k-th event, 1 if event takes place and 0 otherwise

Author(s)

Thomas Welchowski <welchow@imbie.meb.uni-bonn.de>

References

Ludwig Fahrmeir, (1997), *Discrete failure time models*, LMU Sonderforschungsbereich 386, Paper 91, http://epub.ub.uni-muenchen.de/

W. A. Thompson Jr., (1977), *On the Treatment of Grouped Observations in Life Studies*, Biometrics, Vol. 33, No. 3

See Also

contToDisc, dataLong, dataLongCompRisks

Examples

```
# Example Primary Biliary Cirrhosis data
library(survival)

# Convert to months
pbcseq$day <- ceiling(pbcseq$day/30)+1
names(pbcseq) [7] <- "month"
pbcseq$status <- factor(pbcseq$status)

# Convert to long format for time varying effects
pbcseqLong <- dataLongCompRisksTimeDep(dataSet=pbcseq, timeColumn="month",
eventColumns="status", eventColumnsAsFactor=TRUE, idColumn="id",
timeAsFactor=TRUE)
head(pbcseqLong)
```

---

dataLongMultiSpell *Data Long Transformation for multi spell analysis*

Description

Transform data from short format into long format for discrete multi spell survival analysis and right censoring.

Usage

```
dataLongMultiSpell(dataSet, timeColumn, censColumn,
idColumn, timeAsFactor=FALSE)
```

Arguments

dataSet	Original data in short format. Must be of class "data.frame".
timeColumn	Character giving the column name of the observed times. It is required that the observed times are discrete (integer).
censColumn	Character giving the column name of the event status. The event can take multiple values on a discrete scale (0, 1, 2, ...) and repetion of events is allowed. It is assumed that the number zero corresponds to censoring and all number > 0 represent the observed states between transitions.
idColumn	Name of column of identification number of persons as character.
timeAsFactor	Should the time intervals be coded as factor? Default is to use factor. If the argument is false, the column is coded as numeric.

Details

If the data has continuous survival times, the response may be transformed to discrete intervals using function contToDisc. The discrete time variable needs to be strictly increasing for each person, because otherwise the order of the events is not distinguishable. Here is an example data structure in short format prior augmentation with three possible states: \ idColumn=1, 1, ... , 1, 2, 2, ... , n \ timeColumn= t_ID1_1 < t_ID1_1 < ... < t_ID1_k, t_ID2_1 < t_ID2_2 < ... < t_ID2_k, ... \ censColumn=0, 1, ... , 2, 1, 0, ... , 0

Value

Original data.frame with three additional columns:

- obj: Index of persons as integer vector
- timeInt: Index of time intervals (formated as factor or integer)
- e0: Response in long format as binary vector. Event "e0" is assumed to correspond to censoring. If "e0" is coded one in the in the last observed time interval "timeInt" of a person, then this observation was censored.
- e1: Response in long format as binary vector. The event "e1" is the first of the set of possible states "1, 2, 3, ..., X".
- ... Response in long format as binary vectors. These events correspond to the following states "e2, e3, ...".
- eX Response in long format as binary vector. The event "eX" is the last state out of the set of possible states "1, 2, 3, ..., X".

Author(s)

Thomas Welchowski <welchow@imbie.meb.uni-bonn.de>

References

Gerhard Tutz and Matthias Schmid, (2016), *Modeling discrete time-to-event data*, Springer series in statistics, Doi: 10.1007/978-3-319-28158-2

Ludwig Fahrmeir, (1997), *Discrete failure time models*, LMU Sonderforschungsbereich 386, Paper 91, http://epub.ub.uni-muenchen.de/

W. A. Thompson Jr., (1977), *On the Treatment of Grouped Observations in Life Studies*, Biometrics, Vol. 33, No. 3

See Also

contToDisc, dataLongTimeDep, dataLongCompRisks, dataLongCompRisks

Examples

```
############################
# Example of artificial data

# Seed specification
set.seed(-2578)

# Three possible states (0, 1, 2) including censoring
# Discrete time intervals (1, 2, ... , 10)

datFrame <- data.frame(
ID=c(rep(1, 5), rep(2, 3), rep(3, 2), rep(4, 1), rep(5, 3)),
time=c(c(2, 5, 6, 8, 10), c(1, 6, 7), c(9, 10), c(6), c(2, 3, 4)),
state=c(c(0, 0, 2, 1, 0), c(1, 2, 2), c(0, 1), c(2), c(0, 2, 1)),
x=rnorm(n=5+3+2+1+3) )

# Transformation to long format
datFrameLong <- dataLongMultiSpell(dataSet=datFrame, timeColumn="time",
censColumn="state", idColumn="ID")
head(datFrameLong, 25)

# Fit multi state model without autoregressive terms
library(VGAM)
cRm <- vglm(cbind(e0, e1, e2) ~ timeInt + x, data=datFrameLong,
family="multinomial")
summary(cRm)
# -> There is no significant effect of x (as expected).
```

---

dataLongSubDist *Data Matrix and Weights for Discrete Subdistribution Hazard Models*

Description

Generates the augmented data matrix and the weights required for discrete subdistribution hazard modeling with right censoring.

Usage

```
dataLongSubDist(dataSet, timeColumn, eventColumns, eventFocus,
timeAsFactor=TRUE)
```

Arguments

dataSet	Original data in short format. Must be of class "data.frame".
timeColumn	Character specifying the column name of the observed event times. It is required that the observed times are discrete (integer).
eventColumns	Character vector specifying the column names of the event indicators (excluding censoring events). It is required that a 0-1 coding is used for all events. If the sum of the event indicators is zero this will be interpreted as a censoring event.
eventFocus	Column name of the event of interest (type 1 event).
timeAsFactor	Logical indicating whether time should be coded as a factor in the augmented data matrix. If FALSE, a numeric coding will be used.

Details

This function sets up the augmented data matrix and the weights that are needed for weighted maximum likelihood (ML) estimation of the discrete subdistribution model proposed by Berger et al. (2018). The model is a discrete-time extension of the original subdistribution model proposed by Fine and Gray (1999).

Value

Data frame with additional column "subDistWeights". The latter column contains the weights that are needed for fitting a weighted binary regression model, as described in Berger et al. (2018). The weights are calculated by a life table estimator for the censoring event.

Author(s)

Thomas Welchowski <welchow@imbie.meb.uni-bonn.de>

References

Moritz Berger, Matthias Schmid, Thomas Welchowski, Steffen Schmitz-Valckenberg and Jan Beyersmann (2018). *Subdistribution Hazard Models for Competing Risks in Discrete Time*. Submitted manuscript.

Jason P. Fine and Robert J. Gray (1999). *A proportional hazards model for the subdistribution of a competing risk*. Journal of the American Statistical Association 94, pp. 496-509.

See Also

dataLong

Examples

```
# Example with unemployment data
library(Ecdat)
data(UnempDur)

# Generate subsample, reduce number of intervals to k = 5
SubUnempDur <- UnempDur [1:500, ]
SubUnempDur$time <- as.numeric(cut(SubUnempDur$spell, c(0,4,8,16,28)))

# Convert competing risks data to long format
# The event of interest is re-employment at full job
SubUnempDurLong <- dataLongSubDist (dataSet=SubUnempDur, timeColumn="time",
eventColumns=c("censor1", "censor2", "censor3"), eventFocus="censor1")
head(SubUnempDurLong)

# Fit discrete subdistribution hazard model with logistic link function
logisticSubDistr <- glm(y ~ timeInt + ui + age + logwage,
                    family=binomial(), data = SubUnempDurLong,
                    weights=SubUnempDurLong$subDistWeights)
summary(logisticSubDistr)
```

---

dataLongTimeDep *Data Long Time Dependent Covariates*

Description

Transforms short data format to long format for discrete survival modelling of single event analysis with right censoring. Covariates may vary over time.

Usage

```
dataLongTimeDep(dataSet, timeColumn, censColumn, idColumn, timeAsFactor=TRUE)
```

Arguments

dataSet	Original data in short format. Must be of class "data.frame".
timeColumn	Character giving the column name of the observed times. It is required that the observed times are discrete (integer).
censColumn	Character giving the column name of the event indicator. It is required that this is a binary variable with 1=="event" and 0=="censored".
idColumn	Name of column of identification number of persons as character.
timeAsFactor	Should the time intervals be coded as factor? Default is to use factor. If the argument is false, the column is coded as numeric.

Details

There may be some intervals, where no additional information on the covariates is observed (e. g. observed values in interval one and three but two is missing). In this case it is assumed, that the values from the last observation stay constant over time until a new measurement was done.

Value

Original data.frame with three additional columns:

- obj: Index of persons as integer vector
- timeInt: Index of time intervals (factor)
- y: Response in long format as binary vector. 1=="event happens in period timeInt" and 0 otherwise

Author(s)

Thomas Welchowski <welchow@imbie.meb.uni-bonn.de>

References

Gerhard Tutz and Matthias Schmid, (2016), *Modeling discrete time-to-event data*, Springer series in statistics, Doi: 10.1007/978-3-319-28158-2

Ludwig Fahrmeir, (1997), *Discrete failure time models*, LMU Sonderforschungsbereich 386, Paper 91, http://epub.ub.uni-muenchen.de/

W. A. Thompson Jr., (1977), *On the Treatment of Grouped Observations in Life Studies*, Biometrics, Vol. 33, No. 3

See Also

contToDisc, dataLong, dataLongCompRisks

Examples

```
# Example Primary Biliary Cirrhosis data
library(survival)
dataSet1 <- pbcseq

# Only event death is of interest
dataSet1$status [dataSet1$status==1] <- 0
dataSet1$status [dataSet1$status==2] <- 1
table(dataSet1$status)

# Convert to months
dataSet1$day <- ceiling(dataSet1$day/30)+1
names(dataSet1) [7] <- "month"

# Convert to long format for time varying effects
pbcseqLong <- dataLongTimeDep (dataSet=dataSet1, timeColumn="month",
censColumn="status", idColumn="id")
pbcseqLong [pbcseqLong$obj==1, ]
```

devResid *Deviance Residuals*

Description

Computes the root of the deviance residuals for evaluation of performance in discrete survival analysis. A generalized, linear model is used for prediction.

Usage

```
devResid(dataSet, survModelFormula, censColumn, linkFunc = "logit", idColumn = NULL)
```

Arguments

dataSet	Original data in short format. Must be of class "data.frame".
survModelFormula	Gives the specified relationship of discrete response and covariates. The formula is designed, that the intercepts for the time dependent base line hazards are always included. Therefore only covariates should be given in this formula. This argument is required to be of class "formula".
censColumn	Gives the column name of the event indicator (1=observed, 0=censored). Must be of type "character".
linkFunc	Specifies the desired link function in use of generalized, linear models.
idColumn	Gives the column name of the identification number of each person. The argument must be of type "character". Default NULL means, that each row equals one person (no repeated measurements).

Value

- Output: List with objects:
  - DevResid: Square root of deviance residuals as numeric vector.
  - GlmFit: Fit object of class (generalized, linear model used in the calculations)
- Input: A list of given argument input values (saved for reference)

Author(s)

Thomas Welchowski <welchow@imbie.meb.uni-bonn.de>

References

Gerhard Tutz and Matthias Schmid, (2016), *Modeling discrete time-to-event data*, Springer series in statistics, Doi: 10.1007/978-3-319-28158-2

Gerhard Tutz, (2012), *Regression for Categorical Data*, Cambridge University Press

See Also

adjDevResid, brierScore, glm, predErrDiscShort

Examples

```
library(Ecdat)
# Example with cross validation and unemployment data
data(UnempDur)
summary(UnempDur$spell)

# Extract subset of data
set.seed(635)
IDsample <- sample(1:dim(UnempDur)[1], 100)
UnempDurSubset <- UnempDur [IDsample, ]

# Calculate deviance residuals for the unemployment data subset
devianceResiduals <- devResid (dataSet=UnempDurSubset, survModelFormula=spell ~ age + logwage,
censColumn="censor1", linkFunc="logit", idColumn=NULL)
devianceResiduals
```

---

devResidShort *Deviance Residuals*

Description

Computes the root of the deviance residuals for evaluation of performance in discrete survival analysis.

Usage

```
devResidShort(dataSet, hazards)
```

Arguments

dataSet	Original data in long format. Must be of class "data.frame". The correct format can be specified with data preparation, see e. g. dataLong.
hazards	Estimated hazard rates of the data in long format. Discrete hazard rates are probabilities and therefore restricted to the interval [0, 1]

Value

- Output: List with objects:
  - DevResid: Square root of deviance residuals as numeric vector.
- Input: A list of given argument input values (saved for reference)

Author(s)

Thomas Welchowski <welchow@imbie.meb.uni-bonn.de>

References

Gerhard Tutz and Matthias Schmid, (2016), *Modeling discrete time-to-event data*, Springer series in statistics, Doi: 10.1007/978-3-319-28158-2

Gerhard Tutz, (2012), *Regression for Categorical Data*, Cambridge University Press

See Also

adjDevResidShort, predErrDiscShort

Examples

```
library(survival)

# Transform data to long format
heart[, "stop"] <- ceiling(heart[, "stop"])
set.seed(0)
Indizes <- sample(unique(heart$id), 25)
randSample <- heart[unlist(sapply(1:length(Indizes),
function(x) which(heart$id==Indizes[x]))),]
heartLong <- dataLongTimeDep(dataSet=randSample,
timeColumn="stop", censColumn="event", idColumn="id", timeAsFactor=FALSE)

# Fit a generalized, additive model and predict hazard rates on data in long format
library(mgcv)
gamFit <- gam(y ~ timeInt + surgery + transplant + s(age), data=heartLong, family="binomial")
hazPreds <- predict(gamFit, type="response")

# Calculate the deviance residuals
devResiduals <- devResidShort (dataSet=heartLong, hazards=hazPreds)$Output$DevResid

# Compare with estimated normal distribution
plot(density(devResiduals),
main="Empirical density vs estimated normal distribution",
las=1, ylim=c(0, 0.5))
tempFunc <- function (x) dnorm(x, mean=mean(devResiduals), sd=sd(devResiduals))
curve(tempFunc, xlim=c(-10, 10), add=TRUE, col="red")
# The empirical density seems like a mixture distribution,
# but is not too far off in with values greater than 3 and less than 1
```

---

estMargProb *Estimated Marginal Probabilities*

Description

Estimates the marginal probability P(T=t|x) based on estimated hazard rates. The hazard rates may or may not depend on covariates. The covariates have to be equal across all estimated hazard rates. Therefore the given hazard rates should only vary over time.

Usage

```
estMargProb(haz)
```

Arguments

haz	Numeric vector of estimated hazard rates.

Details

The argument *haz* must be given for the all intervals [a_0, a_1), [a_1, a_2), ..., [a_q-1, a_q), [a_q, Inf).

Value

Named vector of estimated marginal probabilities.

Note

It is assumed that all time points up to the last interval [a_q, Inf) are available. If not already present, these can be added manually.

Author(s)

Thomas Welchowski <welchow@imbie.meb.uni-bonn.de>

References

Gerhard Tutz and Matthias Schmid, (2016), *Modeling discrete time-to-event data*, Springer series in statistics, Doi: 10.1007/978-3-319-28158-2

See Also

estSurv

Examples

```
# Example unemployment data
library(Ecdat)
data(UnempDur)

# Select subsample
subUnempDur <- UnempDur [1:100, ]

# Convert to long format
UnempLong <- dataLong (dataSet=subUnempDur, timeColumn="spell", censColumn="censor1")
head(UnempLong)

# Estimate binomial model with logit link
Fit <- glm(formula=y ~ timeInt + age + logwage, data=UnempLong, family=binomial())

# Estimate discrete survival function given age, logwage of first person
```

```
hazard <- predict(Fit, newdata=subset(UnempLong, obj==1), type="response")

# Estimate marginal probabilities given age, logwage of first person
MarginalProbCondX <- estMargProb (c(hazard, 1))
MarginalProbCondX
sum(MarginalProbCondX)==1 # TRUE: Marginal probabilities must sum to 1!
```

`estSurv` *Estimated Survival Function*

Description

Estimates the survival function S(T=t|x) based on estimated hazard rates. The hazard rates may or may not depend on covariates. The covariates have to be equal across all estimated hazard rates. Therefore the given hazard rates should only vary over time.

Usage

```
estSurv(haz)
```

Arguments

`haz`	Numeric vector of estimated hazard rates.

Details

The argument *haz* must be given for the all intervals [a_0, a_1), [a_1, a_2), ..., [a_q-1, a_q), [a_q, Inf).

Value

Named vector of estimated probabilities of survival.

Note

It is assumed that all time points up to the last interval [a_q, Inf) are available. If not already present, these can be added manually.

Author(s)

Thomas Welchowski <welchow@imbie.meb.uni-bonn.de>

References

Gerhard Tutz and Matthias Schmid, (2016), *Modeling discrete time-to-event data*, Springer series in statistics, Doi: 10.1007/978-3-319-28158-2

See Also

estMargProb

Examples

```
# Example unemployment data
library(Ecdat)
data(UnempDur)

# Select subsample
subUnempDur <- UnempDur [1:100, ]

# Convert to long format
UnempLong <- dataLong (dataSet=subUnempDur, timeColumn="spell", censColumn="censor1")
head(UnempLong)

# Estimate binomial model with logit link
Fit <- glm(formula=y ~ timeInt + age + logwage, data=UnempLong, family=binomial())

# Estimate discrete survival function given age, logwage of first person
hazard <- predict(Fit, newdata=subset(UnempLong, obj==1), type="response")
SurvivalFuncCondX <- estSurv(c(hazard, 1))
SurvivalFuncCondX
```

---

estSurvCens *Estimated Survival Function*

Description

Estimates the marginal survival function G(T=t) of the censoring process. Compatible with single event and competing risks data.

Usage

```
estSurvCens(dataSet, timeColumn, eventColumns)
```

Arguments

dataSet	Data in original short format (data.frame).
timeColumn	Name of column with discrete time intervals (character scalar).
eventColumns	Names of the event columns of dataSet. In the competing risks case the event columns have to be in dummy encoding format (numeric vectors).

Value

Named vector of estimated survival function of the censoring process for all time points except the last theoretical interval.

Note

In the censoring survival function the last time interval [a_q, Inf) has the probability of zero.

Author(s)

Thomas Welchowski <welchow@imbie.meb.uni-bonn.de>

References

Gerhard Tutz and Matthias Schmid, (2016), *Modeling discrete time-to-event data*, Springer series in statistics, Doi: 10.1007/978-3-319-28158-2

See Also

estSurv

Examples

```
# Load unemployment data
library(Ecdat)
data(UnempDur)

# Select subsample
subUnempDur <- UnempDur [1:100, ]

######################
# Single event example

# Estimate censoring survival function G(t)
estG <- estSurvCens(dataSet=subUnempDur, timeColumn="spell",
eventColumns="censor1")
estG

########################
# Competing risks example

# Estimate censoring survival function G(t)
estG <- estSurvCens(dataSet=subUnempDur, timeColumn="spell",
eventColumns=c("censor1", "censor2", "censor3", "censor4"))
estG
```

---

evalCindex *Wrapper for evaluation of discrete concordance index*

Description

Convenient version to directly compute the discrete concordance without the need to use multiple functions.

Usage

```
evalCindex(marker, newTime, newEvent, trainTime, trainEvent)
```

Arguments

marker	Gives the predicted values of the linear predictor of a regression model (numeric vector). May also be on the response scale.
newTime	New time intervals in the test data (integer vector).
newEvent	New event indicators in the test data (integer vector with 0 or 1).
trainTime	Time intervals in the training data (integer vector).
trainEvent	Event indicators in the training data (integer vector with 0 or 1).

Value

Value of discrete concordance index between zero and one (numeric scalar).

Note

It is assumed that all time points up to the last observed interval [a_q-1, a_q) are available.

Author(s)

Thomas Welchowski <welchow@imbie.meb.uni-bonn.de>

References

Matthias Schmid, Gerhard Tutz and Thomas Welchowski, (2017), *Discrimination Measures for Discrete Time-to-Event Predictions*, Econometrics and Statistics, Elsevier, Doi: 10.1016/j.ecosta.2017.03.008

Hajime Uno and Tianxi Cai and Lu Tian and L. J. Wei, (2007), *Evaluating Prediction Rules for t-Year Survivors With Censored Regression Models*, Journal of the American Statistical Association

Patrick J. Heagerty and Yingye Zheng, (2005), *Survival Model Predictive Accuracy and ROC Curves*, Biometrics 61, 92-105

See Also

tprUnoShort, fprUnoShort, aucUno, concorIndex

Examples

```
################################################
# Example with unemployment data and prior fitting

library(Ecdat)
library(caret)
library(mgcv)
data(UnempDur)
summary(UnempDur$spell)
# Extract subset of data
```

```
set.seed(635)
IDsample <- sample(1:dim(UnempDur)[1], 100)
UnempDurSubset <- UnempDur [IDsample, ]
set.seed(-570)
TrainingSample <- sample(1:100, 75)
UnempDurSubsetTrain <- UnempDurSubset [TrainingSample, ]
UnempDurSubsetTest <- UnempDurSubset [-TrainingSample, ]

# Convert to long format
UnempDurSubsetTrainLong <- dataLong(dataSet=UnempDurSubsetTrain,
timeColumn="spell", censColumn="censor1")

# Estimate gam with smooth baseline
gamFit <- gam(formula=y ~ s(I(as.numeric(as.character(timeInt)))) +
s(age) + s(logwage), data=UnempDurSubsetTrainLong, family=binomial())
gamFitPreds <- predict(gamFit, newdata=cbind(UnempDurSubsetTest,
timeInt=UnempDurSubsetTest$spell))

# Evaluate C-Index based on short data format
evalCindex(marker=gamFitPreds,
newTime=UnempDurSubsetTest$spell,
newEvent=UnempDurSubsetTest$censor1,
trainTime=UnempDurSubsetTrain$spell,
trainEvent=UnempDurSubsetTrain$censor1)

####################################
# Example National Wilm's Tumor Study

library(survival)
head(nwtco)
summary(nwtco$rel)

# Select subset
set.seed(-375)
Indices <- sample(1:dim(nwtco)[1], 500)
nwtcoSub <- nwtco [Indices, ]

# Convert time range to 30 intervals
intLim <- quantile(nwtcoSub$edrel, prob=seq(0, 1, length.out=30))
intLim [length(intLim)] <- intLim [length(intLim)] + 1
nwtcoSubTemp <- contToDisc(dataSet=nwtcoSub, timeColumn="edrel", intervalLimits=intLim)
nwtcoSubTemp$instit <- factor(nwtcoSubTemp$instit)
nwtcoSubTemp$histol <- factor(nwtcoSubTemp$histol)
nwtcoSubTemp$stage <- factor(nwtcoSubTemp$stage)

# Split in training and test sample
set.seed(-570)
TrainingSample <- sample(1:dim(nwtcoSubTemp)[1], round(dim(nwtcoSubTemp)[1]*0.75))
nwtcoSubTempTrain <- nwtcoSubTemp [TrainingSample, ]
nwtcoSubTempTest <- nwtcoSubTemp [-TrainingSample, ]

# Convert to long format
nwtcoSubTempTrainLong <- dataLong(dataSet=nwtcoSubTempTrain,
```

```
timeColumn="timeDisc", censColumn="rel")

# Estimate glm
inputFormula <- y ~ timeInt + histol + instit + stage
glmFit <- glm(formula=inputFormula, data=nwtcoSubTempTrainLong, family=binomial())
linPreds <- predict(glmFit, newdata=cbind(nwtcoSubTempTest,
timeInt=nwtcoSubTempTest$timeDisc))

# Evaluate C-Index based on short data format
evalCindex(marker=linPreds,
newTime=as.numeric(as.character(nwtcoSubTempTest$timeDisc)),
newEvent=nwtcoSubTempTest$rel,
trainTime=as.numeric(as.character(nwtcoSubTempTrain$timeDisc)),
trainEvent=nwtcoSubTempTrain$rel)
```

`evalIntPredErr` *Wrapper function to compute the integrated prediction error*

Description

Convenient version to directly compute the integrated prediction error curve without the need to use multiple functions.

Usage

```
evalIntPredErr(hazPreds, survPreds=NULL, newTimeInput, newEventInput,
trainTimeInput, trainEventInput, testDataLong, tmax=NULL)
```

Arguments

`hazPreds`	Predicted discrete hazards in the test data. The predictions have to be made with a data set in long format with prefix "dataLong", see e. g. `dataLong`, `dataLongTimeDep`.
`survPreds`	Predicted discrete survival functions of all persons (list of numeric vectors). Each list element corresponds to the survival function of one person beginning with the first time interval. The default NULL assumes that arguments "hazPreds" and "testDataLong" are available. If the last two arguments are not available, then argument "survPreds" needs to be specified.
`newTimeInput`	Discrete time intervals in short format of the test set (integer vector).
`newEventInput`	Events in short format in the test set (binary vector).
`trainTimeInput`	Discrete time intervals in short format of the training set (integer vector).
`trainEventInput`	Events in short format in the training set (binary vector).
`testDataLong`	Test data in long format. Needs to be specified only, if argument "survPreds" is NULL. In this case the discrete survival function is calculated based on the predicted hazards. It is assumed that the data was preprocessed with a function with prefix "dataLong", see e. g. `dataLong`, `dataLongTimeDep`.

tmax — Gives the maximum time interval for which prediction errors are calculated. It must be smaller than the maximum observed time in the training data of the object produced by function predErrDiscShort! Default=NULL means, that all observed intervals are used.

Value

Integrated prediction error (numeric scalar).

Author(s)

Thomas Welchowski <welchow@imbie.meb.uni-bonn.de>

References

Gerhard Tutz and Matthias Schmid, (2016), *Modeling discrete time-to-event data*, Springer series in statistics, Doi: 10.1007/978-3-319-28158-2

Tilmann Gneiting and Adrian E. Raftery, (2007), *Strictly proper scoring rules, prediction, and estimation*, Journal of the American Statistical Association 102 (477), 359-376

See Also

predErrDiscShort, intPredErrDisc, aggregate

Examples

```
##########################
# Example with cancer data

library(survival)
head(cancer)

# Data preparation and convertion to 30 intervals
cancerPrep <- cancer
cancerPrep$status <- cancerPrep$status-1
intLim <- quantile(cancerPrep$time, prob=seq(0, 1, length.out=30))
intLim [length(intLim)] <- intLim [length(intLim)] + 1

# Cut discrete time in smaller number of intervals
cancerPrep <- contToDisc(dataSet=cancerPrep, timeColumn="time", intervalLimits=intLim)

# Generate training and test data
set.seed(753)
TrainIndices <- sample (x=1:dim(cancerPrep) [1], size=dim(cancerPrep) [1]*0.75)
TrainCancer <- cancerPrep [TrainIndices, ]
TestCancer <- cancerPrep [-TrainIndices, ]
TrainCancer$timeDisc <- as.numeric(as.character(TrainCancer$timeDisc))
TestCancer$timeDisc <- as.numeric(as.character(TestCancer$timeDisc))

# Convert to long format
LongTrain <- dataLong(dataSet=TrainCancer, timeColumn="timeDisc", censColumn="status")
```

```
LongTest <- dataLong(dataSet=TestCancer, timeColumn="timeDisc", censColumn="status")
# Convert factors
LongTrain$timeInt <- as.numeric(as.character(LongTrain$timeInt))
LongTest$timeInt <- as.numeric(as.character(LongTest$timeInt))
LongTrain$sex <- factor(LongTrain$sex)
LongTest$sex <- factor(LongTest$sex)

# Estimate, for example, a generalized, additive model in discrete survival analysis
library(mgcv)
gamFit <- gam (formula=y ~ s(timeInt) + s(age) + sex + ph.ecog, data=LongTrain, family=binomial())
summary(gamFit)

# 1. Specification of predicted discrete hazards
# Estimate survival function of each person in the test data
testPredHaz <- predict(gamFit, newdata=LongTest, type="response")

# 1.1. Calculate integrated prediction error
evalIntPredErr(hazPreds=testPredHaz, survPreds=NULL,
newTimeInput=TestCancer$timeDisc, newEventInput=TestCancer$status,
trainTimeInput=TrainCancer$timeDisc, trainEventInput=TrainCancer$status,
testDataLong=LongTest)

# 2. Alternative specification
oneMinusPredHaz <- 1-testPredHaz
predSurv <- aggregate(formula=oneMinusPredHaz ~ obj, data=LongTest, FUN=cumprod, na.action=NULL)

# 2.1. Calculate integrated prediction error
evalIntPredErr(survPreds=predSurv[[2]],
newTimeInput=TestCancer$timeDisc, newEventInput=TestCancer$status,
trainTimeInput=TrainCancer$timeDisc, trainEventInput=TrainCancer$status)
```

`fprUno` *False Positive Rate Uno*

Description

Estimates the predictive false positive rate (fpr) based on cross validation and generalized, linear models (see `glm`). The concept was suggested by Uno et al. (see references)

Usage

```
fprUno(timepoint, dataSet, trainIndices, survModelFormula,
censModelFormula, linkFunc = "logit", idColumn = NULL, timeAsFactor=TRUE)
```

Arguments

`timepoint`	Discrete time interval given that the false positive rate is evaluated (integer scalar)

dataSet	Original data in short format. Should be of class "data.frame".
trainIndices	List of Indices from original data used for training (list of integer vectors). The length of the list is equal to the number of cross validation samples.
survModelFormula	Formula of the discrete survival model. It is used in a generalized, linear model.
censModelFormula	Formula of the censoring model. It is used in a generalized, linear model. Usually this is done without covariates.
linkFunc	Link function of the generalized, linear model.
idColumn	Name of the column with identification numbers of persons. Default NULL means, that each row equals one person (no repeated measurements).
timeAsFactor	Should the time intervals be coded as factor? Default is to use factor. If the argument is false, the column is coded as numeric.

Value

- Output: List with objects:
  - Output: Data frame with two columns: "cutoff" gives the different marker values and "fpr" the false positive rates
  - Input: A list of given argument input values (saved for reference). In addition there is the list element `orderMarker`, which gives the indices of the marker values in increasing order.

Author(s)

Thomas Welchowski <welchow@imbie.meb.uni-bonn.de>

Matthias Schmid <matthias.schmid@imbie.uni-bonn.de>

References

Matthias Schmid, Gerhard Tutz and Thomas Welchowski, (2017), *Discrimination Measures for Discrete Time-to-Event Predictions*, Econometrics and Statistics, Elsevier, Doi: 10.1016/j.ecosta.2017.03.008

Hajime Uno and Tianxi Cai and Lu Tian and L. J. Wei, (2007), *Evaluating Prediction Rules for t-Year Survivors With Censored Regression Models*, Journal of the American Statistical Association

Patrick J. Heagerty and Yingye Zheng, (2005), *Survival Model Predictive Accuracy and ROC Curves*, Biometrics 61, 92-105

See Also

tprUno, tprUnoShort, aucUno, concorIndex, createDataPartition, glm

Examples

```
# Example with cross validation and unemployment data
library(Ecdat)
library(caret)
data(UnempDur)
```

```
summary(UnempDur$spell)

# Extract subset of data
set.seed(635)
IDsample <- sample(1:dim(UnempDur)[1], 100)
UnempDurSubset <- UnempDur [IDsample, ]
head(UnempDurSubset)
range(UnempDurSubset$spell)
set.seed(7550)
CVfolds <- createFolds (y=UnempDurSubset$spell, returnTrain=TRUE, k=2)

# Estimate false positive rate of time interval 7:
tryFPR <- fprUno (timepoint=7, dataSet=UnempDurSubset, trainIndices=CVfolds,
survModelFormula=spell ~ age + logwage, censModelFormula=censor1 ~ 1,
linkFunc="logit", idColumn=NULL, timeAsFactor=FALSE)
tryFPR
plot(tryFPR)
```

`fprUnoShort` *False Positive Rate for arbitrary predition models*

Description

Estimates the false positive rate (based on concept of Uno, et al.) for an arbitrary discrete survival prediction model on one test data set.

Usage

```
fprUnoShort(timepoint, marker, newTime)
```

Arguments

`timepoint`	Gives the discrete time interval of which the fpr is evaluated (numeric scalar).
`marker`	Gives the predicted values of the linear predictor of a regression model (numeric vector). May also be on the response scale.
`newTime`	New time intervals in the test data (integer vector).

Details

This function is useful, if other models than generalized, linear models (glm) should be used for prediction. In the case of glm better use the cross validation version `fprUno`.

Value

- Output: A list with objects:
  - Output: Data frame with two columns: "cutoff" gives the different marker values and "fpr" the false positive rates
  - Input: A list of given argument input values (saved for reference). In addition there is the list element `orderMarker`, which gives the indices of the marker values in increasing order.

Note

It is assumed that all time points up to the last observed interval [a_q-1, a_q) are available.

Author(s)

Thomas Welchowski <welchow@imbie.meb.uni-bonn.de>

Matthias Schmid <matthias.schmid@imbie.uni-bonn.de>

References

Matthias Schmid, Gerhard Tutz and Thomas Welchowski, (2017), *Discrimination Measures for Discrete Time-to-Event Predictions*, Econometrics and Statistics, Elsevier, Doi: 10.1016/j.ecosta.2017.03.008

Hajime Uno and Tianxi Cai and Lu Tian and L. J. Wei, (2007), *Evaluating Prediction Rules for t-Year Survivors With Censored Regression Models*, Journal of the American Statistical Association

Patrick J. Heagerty and Yingye Zheng, (2005), *Survival Model Predictive Accuracy and ROC Curves*, Biometrics 61, 92-105

See Also

tprUno, tprUnoShort, aucUno, concorIndex, createDataPartition, glm

Examples

```
################################################
# Example with unemployment data and prior fitting

library(Ecdat)
library(caret)
library(mgcv)
data(UnempDur)
summary(UnempDur$spell)
# Extract subset of data
set.seed(635)
IDsample <- sample(1:dim(UnempDur)[1], 100)
UnempDurSubset <- UnempDur [IDsample, ]
set.seed(-570)
TrainingSample <- sample(1:100, 75)
UnempDurSubsetTrain <- UnempDurSubset [TrainingSample, ]
UnempDurSubsetTest <- UnempDurSubset [-TrainingSample, ]

# Convert to long format
UnempDurSubsetTrainLong <- dataLong(dataSet=UnempDurSubsetTrain,
timeColumn="spell", censColumn="censor1")

# Estimate gam with smooth baseline
gamFit <- gam(formula=y ~ s(I(as.numeric(as.character(timeInt)))) +
s(age) + s(logwage), data=UnempDurSubsetTrainLong, family=binomial())
gamFitPreds <- predict(gamFit, newdata=cbind(UnempDurSubsetTest, timeInt=UnempDurSubsetTest$spell))

# Estimate false positive rate
```

```
fprGamFit <- fprUnoShort (timepoint=1, marker=gamFitPreds,
newTime=UnempDurSubsetTest$spell)
plot(fprGamFit)

####################################
# Example National Wilm's Tumor Study

library(survival)
head(nwtco)
summary(nwtco$rel)

# Select subset
set.seed(-375)
Indices <- sample(1:dim(nwtco)[1], 500)
nwtcoSub <- nwtco [Indices, ]

# Convert time range to 30 intervals
intLim <- quantile(nwtcoSub$edrel, prob=seq(0, 1, length.out=30))
intLim [length(intLim)] <- intLim [length(intLim)] + 1
nwtcoSubTemp <- contToDisc(dataSet=nwtcoSub, timeColumn="edrel", intervalLimits=intLim)
nwtcoSubTemp$instit <- factor(nwtcoSubTemp$instit)
nwtcoSubTemp$histol <- factor(nwtcoSubTemp$histol)
nwtcoSubTemp$stage <- factor(nwtcoSubTemp$stage)

# Split in training and test sample
set.seed(-570)
TrainingSample <- sample(1:dim(nwtcoSubTemp)[1], round(dim(nwtcoSubTemp)[1]*0.75))
nwtcoSubTempTrain <- nwtcoSubTemp [TrainingSample, ]
nwtcoSubTempTest <- nwtcoSubTemp [-TrainingSample, ]

# Convert to long format
nwtcoSubTempTrainLong <- dataLong(dataSet=nwtcoSubTempTrain,
timeColumn="timeDisc", censColumn="rel")

# Estimate glm
inputFormula <- y ~ timeInt + histol + instit + stage
glmFit <- glm(formula=inputFormula, data=nwtcoSubTempTrainLong, family=binomial())
linPreds <- predict(glmFit, newdata=cbind(nwtcoSubTempTest,
timeInt=nwtcoSubTempTest$timeDisc))

# Estimate tpr given one training and one test sample at time interval 10
fprFit <- fprUnoShort (timepoint=10, marker=linPreds,
newTime=nwtcoSubTempTest$timeDisc)
plot(fprFit)
```

---

gumbel *Gumbel Link Function*

Description

Constructs the link function with gumbel distribution in approriate format for use in generalized, linear models

Usage

```
gumbel()
```

Details

Insert this function into a binary regression model

Author(s)

Thomas Welchowski <welchow@imbie.meb.uni-bonn.de>

Matthias Schmid <matthias.schmid@imbie.uni-bonn.de>

References

Gerhard Tutz and Matthias Schmid, (2016), *Modeling discrete time-to-event data*, Springer series in statistics, Doi: 10.1007/978-3-319-28158-2

See Also

glm

Examples

```
# Example with copenhagen stroke study
library(pec)
data(cost)
head(cost)

# Take subsample and convert time to months
costSub <- cost [1:50, ]
costSub$time <- ceiling(costSub$time/30)
costLong <- dataLong(dataSet=costSub, timeColumn="time", censColumn="status")
gumbelModel <- glm(formula=y ~ timeInt + diabetes, data=costLong, family=binomial(link=gumbel()))

# Estimate hazard given prevStroke nad no prevStroke
hazPrevStroke <- predict(gumbelModel, newdata=data.frame(timeInt=factor(1:143),
diabetes=factor(rep("yes", 143), levels=c("no", "yes"))), type="response")
hazWoPrevStroke <- predict(gumbelModel, newdata=data.frame(timeInt=factor(1:143),
diabetes=factor(rep("no", 143), levels=c("no", "yes"))), type="response")

# Estimate survival function
SurvPrevStroke <- cumprod(1-hazPrevStroke)
SurvWoPrevStroke <- cumprod(1-hazWoPrevStroke)

# Example graphics of survival curves with and without diabetes
plot(x=1:143, y=SurvWoPrevStroke, type="l", xlab="Months",
```

```
ylab="S (t|x)", las=1, lwd=2, ylim=c(0,1))
lines(x=1:143, y=SurvPrevStroke, col="red", lwd=2)
legend("topright", legend=c("Without diabetes", "Diabetes"),
lty=1, lwd=2, col=c("black", "red"))
```

`intPredErrDisc` *Integrated Prediction Error Curve*

Description

Estimates the integrated prediction error curve based on previous estimation of the prediction error curve. This measure is time invariant and is based on the weighted quadratic differences of estimated and observed survival functions.

Usage

```
intPredErrDisc(predErrObj, tmax=NULL)
```

Arguments

`predErrObj`	Object of class "discSurvPredErrDisc" generated by function `predErrDiscShort`
`tmax`	Gives the maximum time interval for which prediction errors are calculated. It must be smaller than the maximum observed time in the training data of the object produced by function `predErrDiscShort`! Default=NULL means, that all observed intervals are used.

Value

Named vector with integrated prediction error per time interval.

Author(s)

Thomas Welchowski <welchow@imbie.meb.uni-bonn.de>

References

Gerhard Tutz and Matthias Schmid, (2016), *Modeling discrete time-to-event data*, Springer series in statistics, Doi: 10.1007/978-3-319-28158-2

Tilmann Gneiting and Adrian E. Raftery, (2007), *Strictly proper scoring rules, prediction, and estimation*, Journal of the American Statistical Association 102 (477), 359-376

See Also

`predErrDiscShort`

Examples

```
#####################################################
# Example with cross validation and unemployment data

library(Ecdat)
library(mgcv)
data(UnempDur)
summary(UnempDur$spell)

# Extract subset of data
set.seed(635)
IDsample <- sample(1:dim(UnempDur)[1], 100)
UnempDurSubset <- UnempDur [IDsample, ]
head(UnempDurSubset)
range(UnempDurSubset$spell)

# Generate training and test data
set.seed(7550)
TrainIndices <- sample (x=1:dim(UnempDurSubset) [1], size=75)
TrainUnempDur <- UnempDurSubset [TrainIndices, ]
TestUnempDur <- UnempDurSubset [-TrainIndices, ]

# Convert to long format
LongTrain <- dataLong(dataSet=TrainUnempDur, timeColumn="spell", censColumn="censor1")
LongTest <- dataLong(dataSet=TestUnempDur, timeColumn="spell", censColumn="censor1")
# Convert factor to numeric for smoothing
LongTrain$timeInt <- as.numeric(as.character(LongTrain$timeInt))
LongTest$timeInt <- as.numeric(as.character(LongTest$timeInt))

# Estimate, for example, a generalized, additive model in discrete survival analysis
gamFit <- gam (formula=y ~ s(timeInt) + age + logwage, data=LongTrain, family=binomial())

# Estimate survival function of each person in the test data
oneMinusPredHaz <- 1 - predict(gamFit, newdata=LongTest, type="response")
predSurv <- aggregate(formula=oneMinusPredHaz ~ obj, data=LongTest, FUN=cumprod)

# Prediction error in first and second interval
tryPredErrDisc1 <- predErrDiscShort (timepoints=1,
estSurvList=predSurv [[2]], newTime=TestUnempDur$spell,
 newEvent=TestUnempDur$censor1, trainTime=TrainUnempDur$spell,
 trainEvent=TrainUnempDur$censor1)

# Estimate integrated prediction error
tryIntPredErrDisc <- intPredErrDisc (tryPredErrDisc1)
tryIntPredErrDisc

# Example up to interval 3 (higher intervals are truncated)
tryIntPredErrDisc2 <- intPredErrDisc (tryPredErrDisc1, tmax=3)
tryIntPredErrDisc2

#########################
# Example with cancer data
```

```
library(survival)
head(cancer)

# Data preparation and convertion to 30 intervals
cancerPrep <- cancer
cancerPrep$status <- cancerPrep$status-1
intLim <- quantile(cancerPrep$time, prob=seq(0, 1, length.out=30))
intLim [length(intLim)] <- intLim [length(intLim)] + 1

# Cut discrete time in smaller number of intervals
cancerPrep <- contToDisc(dataSet=cancerPrep, timeColumn="time", intervalLimits=intLim)

# Generate training and test data
set.seed(753)
TrainIndices <- sample (x=1:dim(cancerPrep) [1], size=dim(cancerPrep) [1]*0.75)
TrainCancer <- cancerPrep [TrainIndices, ]
TestCancer <- cancerPrep [-TrainIndices, ]
TrainCancer$timeDisc <- as.numeric(as.character(TrainCancer$timeDisc))
TestCancer$timeDisc <- as.numeric(as.character(TestCancer$timeDisc))

# Convert to long format
LongTrain <- dataLong(dataSet=TrainCancer, timeColumn="timeDisc", censColumn="status")
LongTest <- dataLong(dataSet=TestCancer, timeColumn="timeDisc", censColumn="status")
# Convert factors
LongTrain$timeInt <- as.numeric(as.character(LongTrain$timeInt))
LongTest$timeInt <- as.numeric(as.character(LongTest$timeInt))
LongTrain$sex <- factor(LongTrain$sex)
LongTest$sex <- factor(LongTest$sex)

# Estimate, for example, a generalized, additive model in discrete survival analysis
gamFit <- gam (formula=y ~ s(timeInt) + s(age) + sex + ph.ecog, data=LongTrain, family=binomial())
summary(gamFit)

# Estimate survival function of each person in the test data
oneMinusPredHaz <- 1 - predict(gamFit, newdata=LongTest, type="response")
predSurv <- aggregate(formula=oneMinusPredHaz ~ obj, data=LongTest, FUN=cumprod)

# One survival curve is missing: Replace the missing values,
# with average value of other survival curves
predSurvTemp <- predSurv [[2]]
for(i in 1:length(predSurv [[2]])) {
  lenTemp <- length(predSurv [[2]] [[i]])
  if(lenTemp < 32) {
    predSurvTemp [[i]] <- c(predSurv [[2]] [[i]], rep(NA, 30-lenTemp))
  }
}
# Calculate average survival curve
avgSurv <- rowMeans(do.call(cbind, predSurvTemp), na.rm=TRUE) [1:4]
# Insert into predicted survival curves
predSurvTemp <- predSurv [[2]]
for(i in 3:(length(predSurvTemp)+1)) {
  if(i==3) {
```

```
    predSurvTemp [[i]] <- avgSurv
  }
  else {
    predSurvTemp [[i]] <- predSurv [[2]] [[i-1]]
  }
}
# Check if the length of all survival curves is equal to the observed
# time intervals in test data
all(sapply(1:length(predSurvTemp), function (x) length(predSurvTemp [[x]]))==
as.numeric(as.character(TestCancer$timeDisc))) # TRUE

# Prediction error second interval
tryPredErrDisc1 <- predErrDiscShort (timepoints=2,
estSurvList=predSurvTemp, newTime=TestCancer$timeDisc,
 newEvent=TestCancer$status, trainTime=TrainCancer$timeDisc,
 trainEvent=TrainCancer$status)

# Calculate integrated prediction error
tryIntPredErrDisc <- intPredErrDisc (tryPredErrDisc1)
tryIntPredErrDisc
```

`lifeTable` *Life Table Construction and Estimates*

Description

Constructs a life table and estimates discrete hazard rates, survival functions, cumulative hazard rates and their standard errors without covariates.

Usage

```
lifeTable(dataSet, timeColumn, censColumn, intervalBorders = NULL)
```

Arguments

`dataSet`	Original data in short format. Must be of class "data.frame".
`timeColumn`	Name of the column with discrete survival times. Must be a scalar character vector.
`censColumn`	Gives the column name of the event indicator (1=observed, 0=censored). Must be of type "character".
`intervalBorders`	Optional names of the intervals for each row, e. g. [a_0, a_1), [a_1, a_2), ..., [a_q-1, a_q)

Value

List containing an object of class "data.frame" with following columns

- n: Number of individuals at risk in a given time interval (integer)
- events: Observed number of events in a given time interval (integer)
- dropouts: Observed number of dropouts in a given time interval (integer)
- atRisk: Estimated number of individuals at risk, corrected by dropouts (numeric)
- hazard: Estimated risk of death (without covariates) in a given time interval
- seHazard: Estimated standard deviation of estimated hazard
- S: Estimated survival curve
- seS: Estimated standard deviation of estimated survival function
- cumHazard: Estimated cumulative hazard function
- seCumHazard: Estimated standard deviation of the estimated cumulative hazard function
- margProb: Estimated marginal probability of event in time interval

Author(s)

Thomas Welchowski <welchow@imbie.meb.uni-bonn.de>

Matthias Schmid <matthias.schmid@imbie.uni-bonn.de>

References

Gerhard Tutz and Matthias Schmid, (2016), *Modeling discrete time-to-event data*, Springer series in statistics, Doi: 10.1007/978-3-319-28158-2

Jerald F. Lawless, (2000), *Statistical Models and Methods for Lifetime Data, 2nd edition*, Wiley series in probability and statistics

Examples

```
# Example with unemployment data
library(Ecdat)
data(UnempDur)

# Extract subset of all persons smaller or equal the median of age
UnempDurSubset <- subset(UnempDur, age<=median(UnempDur$age))
LifeTabUnempDur <- lifeTable (dataSet=UnempDurSubset, timeColumn="spell", censColumn="censor1")
LifeTabUnempDur

# Example with monoclonal gammapothy data
library(survival)
head(mgus)
# Extract subset of mgus
subMgus <- mgus [mgus$futime<=median(mgus$futime), ]
# Transform time in days to intervals [0, 1), [1, 2), [2, 3), ... , [12460, 12461)
mgusInt <- subMgus
mgusInt$futime <- mgusInt$futime + 1
LifeTabGamma <- lifeTable (dataSet=mgusInt, timeColumn="futime", censColumn="death")
```

```
head(LifeTabGamma$Output, 25)
plot(x=1:dim(LifeTabGamma$Output)[1], y=LifeTabGamma$Output$hazard, type="l",
xlab="Time interval", ylab="Hazard", las=1,
main="Life table estimated marginal hazard rates")
```

`martingaleResid` *Martingale Residuals*

Description

Estimates the martingale residuals of a generalized, linear model.

Usage

```
martingaleResid(dataSet, survModelFormula, censColumn,
linkFunc = "logit", idColumn = NULL)
```

Arguments

`dataSet`	Original data in short format. Should be of class "data.frame".
`survModelFormula`	Formula of the discrete survival model. It is used in a generalized, linear model.
`censColumn`	Formula of the censoring model. It is used in a generalized, linear model. Usually this is done without covariates.
`linkFunc`	Link function of the generalized, linear model.
`idColumn`	Name of the column with identification numbers of persons. Default NULL means, that each row equals one person (no repeated measurements).

Value

- Output: List with objects:
  - MartingaleResid: Square root of deviance residuals as numeric vector.
  - GlmFit: Fit object of class (generalized, linear model used in the calculations)
- Input: A list of given argument input values (saved for reference)

Author(s)

Thomas Welchowski <welchow@imbie.meb.uni-bonn.de>

References

Gerhard Tutz and Matthias Schmid, (2016), *Modeling discrete time-to-event data*, Springer series in statistics, Doi: 10.1007/978-3-319-28158-2

Terry M. Therneau and Patricia M. Grambsch and Thomas R. Fleming, (1990), *Martingale-Based Residuals for Survival Models*, Biometrica, Vol. 77, No. 1, 147-160

See Also

tprUno, tprUnoShort, aucUno, concorIndex, glm

Examples

```
# Example with cross validation and unemployment data
library(Ecdat)
data(UnempDur)
summary(UnempDur$spell)

# Extract subset of data
set.seed(635)
IDsample <- sample(1:dim(UnempDur)[1], 100)
UnempDurSubset <- UnempDur [IDsample, ]

# Calculate martingale residuals for the unemployment data subset
MartResid <- martingaleResid (dataSet=UnempDurSubset,
survModelFormula=spell ~ age + logwage, censColumn="censor1",
linkFunc="logit", idColumn=NULL)
MartResid
sum(MartResid$Output$MartingaleResid)

# Plot martingale residuals vs each covariate in the event interval
# Dotted line is a loess estimate
plot(MartResid)
```

---

plot.discSurvAdjDevResid

Plot method of class "discSurvAdjDevResid"

Description

Is called implicitly by using standard plot function on an object of class "discSurvAdjDevResid". It plots a qqplot against the normal distribution. If the model fits the data well, it should be approximately normal distributed.

Usage

```
## S3 method for class 'discSurvAdjDevResid'
plot(x, ...)
```

Arguments

x	Object of class "discSurvAdjDevResid"
...	Additional arguments to plot function

Author(s)

Thomas Welchowski <welchow@imbie.meb.uni-bonn.de>

See Also

adjDevResid

`plot.discSurvAucUno` *Plot method of class "discSurvAucUno"*

Description

Plots and ROC curve for discrete survival models. On the x-axis the false positive rate and on the y-axis the true positive rate is shown.

Usage

```
## S3 method for class 'discSurvAucUno'
plot(x, ...)
```

Arguments

x	Object of class "discSurvAucUno"
...	Additional arguments to the plot function

Author(s)

Thomas Welchowski <welchow@imbie.meb.uni-bonn.de>

See Also

aucUno

`plot.discSurvFprUno` *Plot method of class "discSurvFprUno"*

Description

Plots the cutoff values (x-axis) against the false positive rate (y-axis).

Usage

```
## S3 method for class 'discSurvFprUno'
plot(x, ...)
```

Arguments

x	Object of class "discSurvFprUno"
...	Additional arguments to the plot function

Author(s)

Thomas Welchowski <welchow@imbie.meb.uni-bonn.de>

See Also

fprUno

---

plot.discSurvMartingaleResid

Plot method of class "discSurvMartingaleResid"

Description

Gives a different plot of each marginal covariate against the martingale residuals. Additionally a nonparametric loess estimation is done.

Usage

```
## S3 method for class 'discSurvMartingaleResid'
plot(x, ...)
```

Arguments

x	Object of class "discSurvMartingaleResid"
...	Additional arguments to the plot function

Author(s)

Thomas Welchowski

See Also

martingaleResid

plot.discSurvTprUno *Plot method of class "discSurvTprUno"*

Description

Plots the cutoff values (x-axis) against the true positive rate (y-axis).

Usage

```
## S3 method for class 'discSurvTprUno'
plot(x, ...)
```

Arguments

x	Object of class "discSurvTprUno"
...	Additional arguments to the plot function

Author(s)

Thomas Welchowski <welchow@imbie.meb.uni-bonn.de>

See Also

tprUno

predErrDiscShort *Prediction Error Curves for arbitrary prediction models*

Description

Estimates prediction error curves of arbitrary prediction models. In prediction error curves the estimated and observed survival functions are compared adjusted by weights at given timepoints.

Usage

```
predErrDiscShort(timepoints, estSurvList, newTime, newEvent, trainTime, trainEvent)
```

Arguments

timepoints	Vector of the number of discrete time intervals. Must be of type integer.
estSurvList	List of persons in the test data. Each element contains a numeric vector of estimated survival functions of all given time points.
newTime	Numeric vector of discrete survival times in the test data.
newEvent	Integer vector of univariate event indicator in the test data.
trainTime	Numeric vector of discrete survival times in the training data.
trainEvent	Integer vector of univariate event indicator in the training data.

Details

The prediction error curves should be smaller than 0.25 for all time points, because this is equivalent to a random assignment error.

Value

- List: List with objects:
  - Output: List with two components
    - predErr: Numeric vector with estimated prediction error values. Names give the evaluation time point.
    - weights: List of weights used in the estimation. Each list component gives the weights of a person in the test data.
  - Input: A list of given argument input values (saved for reference)

Author(s)

Thomas Welchowski <welchow@imbie.meb.uni-bonn.de>

References

Van der Laan M. J. and J. M. Robins, (2003), *Unified Methods for Censored Longitudinal Data and Causality*, Springer, New York

Gerds T. A. and M. Schumacher, (2006), *Consistent estimation of the expected Brier score in general survival models with right-censored event times*, Biometrical Journal 48(6), 1029-1040

See Also

intPredErrDisc, aucUno, gam

Examples

```
# Example with cross validation and unemployment data
library(Ecdat)
library(mgcv)
data(UnempDur)
summary(UnempDur$spell)

# Extract subset of data
set.seed(635)
IDsample <- sample(1:dim(UnempDur)[1], 100)
UnempDurSubset <- UnempDur [IDsample, ]
head(UnempDurSubset)
range(UnempDurSubset$spell)

# Generate training and test data
set.seed(7550)
TrainIndices <- sample (x=1:dim(UnempDurSubset) [1], size=75)
TrainUnempDur <- UnempDurSubset [TrainIndices, ]
TestUnempDur <- UnempDurSubset [-TrainIndices, ]
```

```
# Convert to long format
LongTrain <- dataLong(dataSet=TrainUnempDur, timeColumn="spell", censColumn="censor1")
LongTest <- dataLong(dataSet=TestUnempDur, timeColumn="spell", censColumn="censor1")
# Convert factor to numeric for smoothing
LongTrain$timeInt <- as.numeric(as.character(LongTrain$timeInt))
LongTest$timeInt <- as.numeric(as.character(LongTest$timeInt))

#######################################################################
# Estimate a generalized, additive model in discrete survival analysis

gamFit <- gam (formula=y ~ s(timeInt) + age + logwage, data=LongTrain, family=binomial())

# Estimate survival function of each person in the test data
oneMinusPredHaz <- 1 - predict(gamFit, newdata=LongTest, type="response")
predSurv <- aggregate(formula=oneMinusPredHaz ~ obj, data=LongTest, FUN=cumprod)

# Prediction error in first interval
tryPredErrDisc1 <- predErrDiscShort (timepoints=1,
estSurvList=predSurv [[2]], newTime=TestUnempDur$spell,
newEvent=TestUnempDur$censor1, trainTime=TrainUnempDur$spell,
 trainEvent=TrainUnempDur$censor1)
tryPredErrDisc1
summary(tryPredErrDisc1)

# Prediction error of the 2. to 10. interval
tryPredErrDisc2 <- predErrDiscShort (timepoints=2:10,
estSurvList=predSurv [[2]], newTime=TestUnempDur$spell,
newEvent=TestUnempDur$censor1, trainTime=TrainUnempDur$spell,
trainEvent=TrainUnempDur$censor1)
tryPredErrDisc2
summary(tryPredErrDisc2)

#######################################
# Fit a random discrete survival forest

library(randomForest)
LongTrainRF <- LongTrain
LongTrainRF$y <- factor(LongTrainRF$y)
rfFit <- randomForest (formula=y ~ timeInt + age + logwage, data=LongTrainRF)

# Estimate survival function of each person in the test data
oneMinusPredHaz <- 1 - predict(rfFit, newdata=LongTest, type="prob") [, 2]
predSurv <- aggregate(formula=oneMinusPredHaz ~ obj, data=LongTest, FUN=cumprod)

# Prediction error in first interval
tryPredErrDisc1 <- predErrDiscShort (timepoints=1,
estSurvList=predSurv [[2]], newTime=TestUnempDur$spell,
newEvent=TestUnempDur$censor1, trainTime=TrainUnempDur$spell,
 trainEvent=TrainUnempDur$censor1)
tryPredErrDisc1
summary(tryPredErrDisc1)

# Prediction error of the 2. to 10. interval
```

```
tryPredErrDisc2 <- predErrDiscShort (timepoints=2:10,
estSurvList=predSurv [[2]], newTime=TestUnempDur$spell,
newEvent=TestUnempDur$censor1, trainTime=TrainUnempDur$spell,
trainEvent=TrainUnempDur$censor1)
tryPredErrDisc2
summary(tryPredErrDisc2)
```

---

print.discSurvAdjDevResid *Print method of class "discSurvAdjDevResid"*

Description

Prints the adjusted deviance residuals rounded to 4 digits.

Usage

```
## S3 method for class 'discSurvAdjDevResid'
print(x, ...)
```

Arguments

x	Object of class "discSurvAdjDevResid"
...	Additional arguments to default print

Author(s)

Thomas Welchowski <welchow@imbie.meb.uni-bonn.de>

See Also

adjDevResid

---

print.discSurvAucUno *Print method of class "discSurvAucUno"*

Description

Prints the auc value rounded to 4 digits.

Usage

```
## S3 method for class 'discSurvAucUno'
print(x, ...)
```

Arguments

x	Object of class "discSurvAdjDevResid"
...	Additional arguments to default print

Author(s)

Thomas Welchowski <welchow@imbie.meb.uni-bonn.de>

See Also

aucUno

---

print.discSurvConcorIndex

Print method of class "discSurvConcorIndex"

Description

Displays the concordance index rounded to 4 digits.

Usage

```
## S3 method for class 'discSurvConcorIndex'
print(x, ...)
```

Arguments

x	Object of class "discSurvConcorIndex"
...	Additional arguments to the print function

Author(s)

Thomas Welchowski <welchow@imbie.meb.uni-bonn.de>

See Also

concorIndex

print.discSurvDevResid *Print method of class "discSurvDevResid"*

Description

Displays the deviance residuals rounded to 4 digits.

Usage

```
## S3 method for class 'discSurvDevResid'
print(x, ...)
```

Arguments

x	Object of class "discSurvDevResid"
...	Additional arguments to the print function

Author(s)

Thomas Welchowski <welchow@imbie.meb.uni-bonn.de>

See Also

devResid

print.discSurvFprUno *Print method of class "discSurvFprUno"*

Description

Displays the cutoff values and false positive rates rounded to 4 digits.

Usage

```
## S3 method for class 'discSurvFprUno'
print(x, ...)
```

Arguments

x	Object of class "discSurvFprUno"
...	Additional arguments to the print function

Author(s)

Thomas Welchowski <welchow@imbie.meb.uni-bonn.de>

See Also

fprUno

---

print.discSurvLifeTable

Print method of class "discSurvLifeTable"

Description

Prints the discrete life table estimates rounded to 4 digits.

Usage

```
## S3 method for class 'discSurvLifeTable'
print(x, ...)
```

Arguments

x	Object of class "discSurvLifeTable"
...	Additional arguments to the print function

Author(s)

Thomas Welchowski <welchow@imbie.meb.uni-bonn.de>

See Also

lifeTable

---

print.discSurvMartingaleResid

Print method of class "discSurvMartingaleResid"

Description

Displays martingale residuals rounded to 4 digits.

Usage

```
## S3 method for class 'discSurvMartingaleResid'
print(x, ...)
```

Arguments

x	Object of class "discSurvMartingaleResid"
...	Additional arguments to the print function

Author(s)

Thomas Welchowski <welchow@imbie.meb.uni-bonn.de>

See Also

martingaleResid

---

print.discSurvPredErrDisc

Print method of class "discSurvPredErrDisc"

Description

Displays prediction error rounded to 4 digits.

Usage

```
## S3 method for class 'discSurvPredErrDisc'
print(x, ...)
```

Arguments

x	Object of class "discSurvPredErrDisc"
...	Additional arguments to the print function

Author(s)

Thomas Welchowski <welchow@imbie.meb.uni-bonn.de>

See Also

predErrDiscShort

---

print.discSurvSimCompRisk

Print method of class "discSurvSimCompRisk"

Description

Displays the simulated data set.

Usage

```
## S3 method for class 'discSurvSimCompRisk'
print(x, ...)
```

Arguments

x	Object of class "discSurvSimCompRisk"
...	Additional arguments to the print function

Author(s)

Thomas Welchowski <welchow@imbie.meb.uni-bonn.de>

See Also

simCompRisk

---

print.discSurvTprUno *Print method of class "discSurvTprUno"*

Description

Displays the true positive rate rounded to 4 digits.

Usage

```
## S3 method for class 'discSurvTprUno'
print(x, ...)
```

Arguments

x	Object of class "discSurvTprUno"
...	Additional arguments to the print function

Author(s)

Thomas Welchowski

See Also

tprUno

simCompRisk *Simulation of Competing Risks data*

Description

Simulates responses and covariates of discrete competing risk models. Responses and covariates are modelled separately by gaussian copulas given kendalls tau. This data can further be analysed with discrete competing risk models.

Usage

```
simCompRisk(responseCorr, covariateCorr, sampleSize, covariateSeed = NULL,
covariateQuantFunc, covariateArgs, intercept = TRUE, trueCoef, responseSeed,
responseQuantFunc, responseFixArgs, responseLinPredArgs, censorRN, censorArgs, censorSeed)
```

Arguments

responseCorr	Regulates correlation between event survival times. Numeric Matrix with kendalls tau version b rank correlations. Each cell is restricted to be between -1 and 1. Diagonal elements are always 1.
covariateCorr	Regulates correlation between event covariates. Numeric Matrix with kendalls tau version b rank correlations (each cell is restricted to be between -1 and 1). Diagonal elements are always 1. Uses singular, value decomposition for invertation of covariance matrices.
sampleSize	Integer scalar specifying the number of rows of the data frame.
covariateSeed	Integer scalar, specifying seed of covariate random number generation.
covariateQuantFunc	Character vector, giving the marginal quantile functions of all covariates
covariateArgs	List of Arguments for each quantile function of the marginal distributions. Each list element should be a named numeric vector (names giving the arguments)
intercept	Logical vector, giving if intercept is given in true coefficients for each Event (Default all TRUE)
trueCoef	List of numeric vectors containing the beta of true coefficients, e. g. linear predictor eta = X
responseSeed	Integer scalar, specifying seed of response random number generation
responseQuantFunc	Character vector, giving the marginal quantile functions of all survival
responseFixArgs	List of Arguments for each quantile function of the marginal distributions. Each list element should be a named numeric vector.
responseLinPredArgs	List of lists, specifying the relationship of linear predictor and parameters of the marginal distributions. Each list element is a list of all functional relationships between linear predictor and parameters of one marginal distribution. Each list element is a function giving the functional relationship between linear predictor and one parameter.

censorRN	Integer scalar, specifying seed of censoring random number generation
censorArgs	Named numeric vector, giving all arguments of the marginal censoring distribution
censorSeed	Integer scalar, specifying seed of censoring random number generation

Value

List with following components

- Data: Simulated data of class "data.frame"
- covariateCorr: Original input rank correlation structure of covariates
- samleSize: Sample size
- covariateSeed: Covariate seed
- ... (all arguments specified in Input are saved other the corresponding names)

Author(s)

Thomas Welchowski <welchow@imbie.meb.uni-bonn.de>

References

Wiji Narendranathan and Mark B. Stewart, (1993), *Modelling the probability of leaving unemployment: competing risks models with flexible base-line hazards*, Applied Statistics, pages 63-83

Roger B. Nelsen, (2006), *An introduction to Copulas*, Springer Science+Business Media, Inc.

Steele Fiona and Goldstein Harvey and Browne William, (2004), *A general multilevel multistate competing risks model for event history data* Statistical Modelling, volume 4, pages 145-159

See Also

dataLongCompRisks, vgam

Examples

```
library(Matrix)
library(matrixcalc)

########################
# Design of Simulation

# Monte carlo samples for each design = 10
# SampleSize = 100 in each replicate

# Responses:
# R1 i.n.d. ~ Weibull (shape=0.5, scale=exp(eta))
# E(R1) = gamma (3)*exp(eta)
# R2 i.n.d. ~ Weibull (shape=1, scale=exp(eta))
# E(R2) = gamma (2)*exp(eta)
# R3 i.n.d. ~ Weibull (shape=2, scale=exp(eta))
# E(R3) = gamma (1.5)*exp(eta)
```

```
# Random Censoring
# Z = 2/3 * (gamma (3) + gamma (2) + gamma (1.5)) = 2.590818
# Independent of survival times C_i i.i.d ~ Exp (lambda=Z)

# Correlation structure of survival times:
# Specify with kendalls tau -> spearmans rho
# Kendalls tau matrix:
# R1   R2   R3
# R1  1   0.3  0.4
# R2  0.3   1  0.5
# R3  0.4 0.5    1

# Covariates:
# V1: Binary variable ~ Bin (n=2, pi=0.5) -> E (V1) = 1
# V2: Continuous positive variable ~ Gamma (shape=1, scale=1) -> E (V2) = 1
# V3: Continuous variable ~ Normal (mu=0, sigma^2=1) -> E (V5) = 0

# True linear predictor:
# eta = X %*% beta
# beta_1 = c(-0.5, 1, 0.5)
# beta_2 = c(1, -1, 1)
# beta_3 = c(-0.5, -0.5, 2)

# Correlation Structure in Covariates:
# Specify with kendalls tau -> pearsons rho
# V1    V2     V3
# V1       1 -0.25      0
# V2   -0.25     1   0.25
# V3       0  0.25      1

# Design Correlation Matrix of covariates
DesignCovariateCor <- diag(3)
DesignCovariateCor [lower.tri(DesignCovariateCor)] <- c(-0.25, 0, 0.25)
DesignCovariateCor [upper.tri(DesignCovariateCor)] <- c(-0.25, 0, 0.25)

# Check if correlation matrix is symmetric
sum(DesignCovariateCor-t(DesignCovariateCor))==0 # TRUE -> ok
# Check if correlation matrix is positive definite
is.positive.definite (DesignCovariateCor) # TRUE -> ok
# Check if correlation matrix is transformed pearson matrix is positive, definite
is.positive.definite (apply(DesignCovariateCor, c(1,2), tauToPearson)) # TRUE -> ok

# Design Correlation Matrix positive definite after transformation
DesignResponseCor <- diag(3)
DesignResponseCor [lower.tri(DesignResponseCor)] <- c(0.3, 0.4, 0.5)
DesignResponseCor [upper.tri(DesignResponseCor)] <- c(0.3, 0.4, 0.5)

# Check if response correlation matrix is symmetric
sum(DesignResponseCor-t(DesignResponseCor))==0
# Check if response correlation matrix is positive definite
is.positive.definite (DesignResponseCor)
# Check if response correlation matrix is transformed pearson matrix is positive, definite
```

```
is.positive.definite (apply(DesignResponseCor, c(1,2), tauToPearson))

# Simulate raw data
DesignSampleSize <- 100
MonteCarloSamples <- 10
SimOutput <- vector("list", MonteCarloSamples)
for(j in 1:MonteCarloSamples) {
  SimOutput [[j]] <- simCompRisk (responseCorr=DesignResponseCor, covariateCorr=DesignCovariateCor,
  covariateSeed=NULL, sampleSize=100, covariateQuantFunc=c("qbinom", "qgamma", "qnorm"),
  covariateArgs=list(c(size=2, prob=0.5), c(shape=1, scale=1), c(mean=0, sd=1)),
  intercept=c(FALSE, FALSE, FALSE),
  trueCoef=list(c(-0.5, 1, 0.5), c(1, -1, 1), c(-0.5, -0.5, 2)),
  responseSeed=NULL, responseQuantFunc=c("qweibull", "qweibull", "qweibull"),
  responseFixArgs=list(c(shape=0.5), c(shape=1), c(shape=2)),
  responseLinPredArgs=list(list(scale=function (x) {exp(x)}),
  list(scale=function (x) {exp(x)}),
  list(scale=function (x) {exp(x)})), censorRN="rexp",
  censorArgs=c(rate=2/3 * (gamma (3) + gamma (2) + gamma (1.5))), censorSeed=NULL)
}
SimOutput [[1]]
SimOutput [[5]]

OverviewData <- data.frame(Min=rep(NA, MonteCarloSamples),
Max=rep(NA, MonteCarloSamples), Censrate=rep(NA, MonteCarloSamples))
for(j in 1:MonteCarloSamples) {

  # Calculate Range of observed time
  OverviewData [j, 1:2] <- range (SimOutput [[j]]$Data [, "Time"])

  # Calculate censoring rate
  OverviewData [j, "Censrate"] <- mean(SimOutput [[j]]$Data [, "Censor"])

}
OverviewData
```

`summary.discSurvConcorIndex` *Summary method of class "discSurvConcorIndex"*

Description

Displays the concordance index, auc values over all time points, survival function and marginal probabilities each rounded to 4 digits.

Usage

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

Arguments

object	Object of class "discSurvConcorIndex"
...	Additional arguments to generic summary function

Author(s)

Thomas Welchowski

See Also

concorIndex

---

summary.discSurvPredErrDisc

Summary method of class "discSurvPredErrDisc"

Description

Prints the prediction errors and corresponding weights given by the survival function of the censoring process. All values are rounded to 4 digits.

Usage

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

Arguments

object	Object of class "discSurvPredErrDisc"
...	Additional arguments to generic summary function

Author(s)

Thomas Welchowski

See Also

predErrDiscShort

`tauToPearson` *Transformation of Tau to Pearson correlation*

Description

If the two variables are bivariate normal distributed, this formula maps kendalls tau to bravais pearson correlation coefficient rho.

Usage

```
tauToPearson(Tau)
```

Arguments

`Tau`	Numeric vector in the interval [-1, 1]

Details

This relationship holds only in the case if the two random variables are jointly normal distributed.

Value

Vector of bravais pearson correlation coefficient rho

Author(s)

Thomas Welchowski <welchow@imbie.meb.uni-bonn.de>

References

William H. Kruskal, (1958), *Ordinal Measures of Association*, Journal of the American Statistical Association, Vol. 53, No. 284, pp. 814-861

See Also

`simCompRisk`

Examples

```
# Plot of relation between kendalls tau and pearson correlation
MaxDisc1 <- nlminb(start=0.5, objective=function (x) -abs(tauToPearson (x)-x))
MaxDisc2 <- nlminb(start=-0.5, objective=function (x) -abs(tauToPearson (x)-x))
plot(x=seq(-1,1,length.out=500), y=tauToPearson(seq(-1,1,length.out=500)),
xlab=expression(tau), ylab=expression(rho), type="l", las=1,
main="Relationship between tau and pearson correlation (bivariate normal)", lwd=2)
lines(x=seq(-1,1,length.out=500), y=seq(-1,1,length.out=500), lty=2)
segments(x0=0, y0=-1.25, x1=0, y1=0, lty=2)
segments(x0=-1.25, y0=0, x1=0, y1=0, lty=2)
segments(x0=MaxDisc1$par, y0=-1.25, x1=MaxDisc1$par, y1=tauToPearson (MaxDisc1$par), lty=2)
```

```
segments(x0=MaxDisc2$par, y0=-1.25, x1=MaxDisc2$par, y1=tauToPearson (MaxDisc2$par), lty=2)

# The maximum discrepancy between pearson and spearman is at
# a kendalls tau value about 0.56 and -0.56
```

---

tprUno *True positive Rate Uno*

Description

Estimates the true positive rate based on Uno et al. to evaluate the predictive accuracy of discrete generalized, linear survival models by cross validation.

Usage

```
tprUno(timepoint, dataSet, trainIndices, survModelFormula,
censModelFormula, linkFunc = "logit", idColumn = NULL, timeAsFactor=TRUE)
```

Arguments

timepoint	Discrete time interval given that the false positive rate is evaluated (integer scalar)
dataSet	Original data in short format. Should be of class "data.frame".
trainIndices	List of Indices from original data used for training (list of integer vectors). The length of the list is equal to the number of cross validation samples.
survModelFormula	Formula of the discrete survival model. It is used in a generalized, linear model.
censModelFormula	Formula of the censoring model. It is used in a generalized, linear model. Usually this is done without covariates.
linkFunc	Link function of the generalized, linear model.
idColumn	Name of the column with identification numbers of persons. Default NULL means, that each row equals one person (no repeated measurements).
timeAsFactor	Should the time intervals be coded as factor? Default is to use factor. If the argument is false, the column is coded as numeric.

Details

The formula survModelFormula must be in a specific structure: The response on the left side of the formula is the time of the short data format. On the right side are the covariates without time, e. g. Time ~ X1 + X2 if there are only two covariates. The time will be added automatically.

The next formula survModelFormula has similiar structure. The difference is the left side of the formula: This must be the

Value

List with objects

- Output: Data frame with two columns: "cutoff" gives the different marker values and "tpr" the true positive rates
- Input: A list of given argument input values (saved for reference). In addition there is the list element `orderMarker`, which gives the indices of the marker values in increasing order.

Author(s)

Thomas Welchowski <welchow@imbie.meb.uni-bonn.de>

Matthias Schmid <matthias.schmid@imbie.uni-bonn.de>

References

Matthias Schmid, Gerhard Tutz and Thomas Welchowski, (2017), *Discrimination Measures for Discrete Time-to-Event Predictions*, Econometrics and Statistics, Elsevier, Doi: 10.1016/j.ecosta.2017.03.008

Hajime Uno and Tianxi Cai and Lu Tian and L. J. Wei, (2007), *Evaluating Prediction Rules for t-Year Survivors With Censored Regression Models*, Journal of the American Statistical Association

Patrick J. Heagerty and Yingye Zheng, (2005), *Survival Model Predictive Accuracy and ROC Curves*, Biometrics 61, 92-105

See Also

`tprUnoShort`, `fprUno`, `fprUnoShort`, `aucUno`, `concorIndex`, `createDataPartition`, `glm`

Examples

```
# Example with cross validation and unemployment data
library(Ecdat)
library(caret)
data(UnempDur)
summary(UnempDur$spell)

# Extract subset of data
set.seed(635)
IDsample <- sample(1:dim(UnempDur)[1], 100)
UnempDurSubset <- UnempDur [IDsample, ]
head(UnempDurSubset)
range(UnempDurSubset$spell)
set.seed(7550)
CVfolds <- createFolds (y=UnempDurSubset$spell, returnTrain=TRUE, k=2)

# Estimate true positive rate of time interval 7:
# Correspondes to three and a half month duration (each interval is of length two weeks)
tryTPR <- tprUno (timepoint=7, dataSet=UnempDurSubset,
trainIndices=CVfolds, survModelFormula=spell ~ age + logwage,
censModelFormula=censor1 ~ 1, linkFunc="logit", idColumn=NULL)
tryTPR
plot(tryTPR)
```

tprUnoShort *True Positive Rate for arbitrary predition models*

Description

Estimates the true positive rate (based on concept of Uno, et al.) for an arbitrary discrete survival prediction model on one test data set.

Usage

```
tprUnoShort(timepoint, marker, newTime, newEvent, trainTime, trainEvent)
```

Arguments

timepoint	Gives the discrete time interval of which the tpr is evaluated (numeric scalar).
marker	Gives the predicted values of the linear predictor of a regression model (numeric vector). May also be on the response scale.
newTime	New time intervals in the test data (integer vector).
newEvent	New event indicators in the test data (integer vector with 0 or 1).
trainTime	Time intervals in the training data (integer vector).
trainEvent	Event indicators in the training data (integer vector with 0 or 1).

Details

This function is useful, if other models than generalized, linear models (glm) should be used for prediction. In the case of glm better use the cross validation version tprUno.

Value

List with objects

- Output Data frame with two columns: "cutoff" gives the different marker values and "fpr" the false positive rates
- Input A list of given argument input values (saved for reference). Another list element is selectInd, which gives the selected indices of the marker values with time intervals available in both training and test sets. In addition there is the list element orderMarker, which gives the indices of the marker values in increasing order.

Note

It is assumed that all time points up to the last observed interval [a_q-1, a_q) are available.

Author(s)

Thomas Welchowski <welchow@imbie.meb.uni-bonn.de>

Matthias Schmid <matthias.schmid@imbie.uni-bonn.de>

References

Matthias Schmid, Gerhard Tutz and Thomas Welchowski, (2017), *Discrimination Measures for Discrete Time-to-Event Predictions*, Econometrics and Statistics, Elsevier, Doi: 10.1016/j.ecosta.2017.03.008

Hajime Uno and Tianxi Cai and Lu Tian and L. J. Wei, (2007), *Evaluating Prediction Rules for t-Year Survivors With Censored Regression Models*, Journal of the American Statistical Association

Patrick J. Heagerty and Yingye Zheng, (2005), *Survival Model Predictive Accuracy and ROC Curves*, Biometrics 61, 92-105

See Also

tprUno, aucUno, concorIndex, createDataPartition, glm

Examples

```
################################################
# Example with unemployment data and prior fitting

library(Ecdat)
library(caret)
library(mgcv)
data(UnempDur)
summary(UnempDur$spell)
# Extract subset of data
set.seed(635)
IDsample <- sample(1:dim(UnempDur)[1], 100)
UnempDurSubset <- UnempDur [IDsample, ]
set.seed(-570)
TrainingSample <- sample(1:100, 75)
UnempDurSubsetTrain <- UnempDurSubset [TrainingSample, ]
UnempDurSubsetTest <- UnempDurSubset [-TrainingSample, ]

# Convert to long format
UnempDurSubsetTrainLong <- dataLong(dataSet=UnempDurSubsetTrain,
timeColumn="spell", censColumn="censor1")

# Estimate gam with smooth baseline
gamFit <- gam(formula=y ~ s(I(as.numeric(as.character(timeInt)))) +
s(age) + s(logwage), data=UnempDurSubsetTrainLong, family=binomial())
gamFitPreds <- predict(gamFit, newdata=cbind(UnempDurSubsetTest,
timeInt=UnempDurSubsetTest$spell))

# Estimate tpr given one training and one test sample
tprGamFit <- tprUnoShort (timepoint=1, marker=gamFitPreds,
newTime=UnempDurSubsetTest$spell, newEvent=UnempDurSubsetTest$censor1,
trainTime=UnempDurSubsetTrain$spell, trainEvent=UnempDurSubsetTrain$censor1)
plot(tprGamFit)

####################################
# Example National Wilm's Tumor Study

library(survival)
```

```
head(nwtco)
summary(nwtco$rel)

# Select subset
set.seed(-375)
Indices <- sample(1:dim(nwtco)[1], 500)
nwtcoSub <- nwtco [Indices, ]

# Convert time range to 30 intervals
intLim <- quantile(nwtcoSub$edrel, prob=seq(0, 1, length.out=30))
intLim [length(intLim)] <- intLim [length(intLim)] + 1
nwtcoSubTemp <- contToDisc(dataSet=nwtcoSub, timeColumn="edrel", intervalLimits=intLim)
nwtcoSubTemp$instit <- factor(nwtcoSubTemp$instit)
nwtcoSubTemp$histol <- factor(nwtcoSubTemp$histol)
nwtcoSubTemp$stage <- factor(nwtcoSubTemp$stage)

# Split in training and test sample
set.seed(-570)
TrainingSample <- sample(1:dim(nwtcoSubTemp)[1], round(dim(nwtcoSubTemp)[1]*0.75))
nwtcoSubTempTrain <- nwtcoSubTemp [TrainingSample, ]
nwtcoSubTempTest <- nwtcoSubTemp [-TrainingSample, ]

# Convert to long format
nwtcoSubTempTrainLong <- dataLong(dataSet=nwtcoSubTempTrain,
timeColumn="timeDisc", censColumn="rel")

# Estimate glm
inputFormula <- y ~ timeInt + histol + instit + stage
glmFit <- glm(formula=inputFormula, data=nwtcoSubTempTrainLong, family=binomial())
linPreds <- predict(glmFit, newdata=cbind(nwtcoSubTempTest,
timeInt=nwtcoSubTempTest$timeDisc))

# Estimate tpr given one training and one test sample at time interval 5
tprFit <- tprUnoShort (timepoint=5, marker=linPreds,
newTime=nwtcoSubTempTest$timeDisc, newEvent=nwtcoSubTempTest$rel,
trainTime=nwtcoSubTempTrain$timeDisc, trainEvent=nwtcoSubTempTrain$rel)
plot(tprFit)
```

Index