

# Package ‘riskRegression’

April 19, 2018

**Type** Package

**Title** Risk Regression Models and Prediction Scores for Survival  
Analysis with Competing Risks

**Version** 2018.04.21

**Depends** R (>= 2.9.0), data.table (>= 1.10.4), ggplot2 (>= 2.1.0),  
prodlim (>= 1.6.1)

**Imports** abind, doParallel, stats, graphics, survival (>= 2.40.1), lava  
(>= 1.4.7), cmprsk, plotrix, timereg (>= 1.9.1), foreach,  
parallel, Rcpp, rms (>= 5.0-0)

**LinkingTo** Rcpp, RcppArmadillo

**Suggests** boot, CoxBoost, Daim, mets, mstate, party, pec, penalized,  
pROC, randomForest, randomForestSRC, rbenchmark, rpart,  
testthat, knitr, rmarkdown

**Maintainer** Thomas Alexander Gerds <tag@biostat.ku.dk>

**Description** Implementation of the following methods for event history analysis.  
Risk regression models for survival endpoints also in the presence of competing  
risks are fitted using binomial regression based on a time sequence of binary  
event status variables. A formula interface for the Fine-Gray regression model  
and an interface for the combination of cause-specific Cox regression models.  
A toolbox for assessing and comparing performance of risk predictions (risk  
markers and risk prediction models). Prediction performance is measured by the  
Brier score and the area under the ROC curve for binary possibly time-dependent  
outcome. Inverse probability of censoring weighting and pseudo values are used  
to deal with right censored data. Lists of risk markers and lists of risk models  
are assessed simultaneously. Cross-validation repeatedly splits the data, trains  
the risk prediction models on one part of each split and then summarizes and  
compares the performance across splits.

**License** GPL (>= 2)

**RoxygenNote** 6.0.1

**NeedsCompilation** yes

**Author** Thomas Alexander Gerds [aut, cre],  
Paul Blanche [ctb],

Ulla Brasch Mogensen [ctb],  
Brice Ozenne [aut]

**Repository** CRAN

**Date/Publication** 2018-04-19 10:14:32 UTC

## R topics documented:

as.data.table.predictCox . . . . .	4
as.data.table.predictCSC . . . . .	4
ate . . . . .	5
autoplot.ate . . . . .	10
autoplot.predictCox . . . . .	11
autoplot.predictCSC . . . . .	12
boot2pvalue . . . . .	13
boxplot.Score . . . . .	15
calcSeCox . . . . .	17
calcSeCSC . . . . .	18
coef.CauseSpecificCox . . . . .	20
coef.riskRegression . . . . .	20
colCenter_cpp . . . . .	21
colCumSum . . . . .	21
colMultiply_cpp . . . . .	22
colScale_cpp . . . . .	23
colSumsCrossprod . . . . .	23
confBandCox . . . . .	24
coxBaseEstimator . . . . .	25
coxCenter . . . . .	25
coxDesign . . . . .	26
coxFormula . . . . .	27
coxLP . . . . .	27
coxN . . . . .	28
coxSpecialStrata . . . . .	29
coxStrata . . . . .	29
coxVarCov . . . . .	30
coxVariableName . . . . .	31
CSC . . . . .	31
discreteRoot . . . . .	34
extractStrata . . . . .	35
FGR . . . . .	35
getSplitMethod . . . . .	37
iidCox . . . . .	39
influenceCoxTest . . . . .	40
ipcw . . . . .	43
Melanoma . . . . .	45
model.matrix.phreg . . . . .	46
Paquid . . . . .	47
penalizedS3 . . . . .	48

plot.riskRegression . . . . .	49
plotAUC . . . . .	50
plotBrier . . . . .	51
plotCalibration . . . . .	52
plotEffects . . . . .	54
plotRisk . . . . .	56
plotROC . . . . .	57
predict.CauseSpecificCox . . . . .	59
predict.FGR . . . . .	62
predict.riskRegression . . . . .	62
predictCox . . . . .	63
predictCoxPL . . . . .	67
predictRisk . . . . .	68
print.ate . . . . .	71
print.CauseSpecificCox . . . . .	72
print.FGR . . . . .	73
print.influenceCoxTest . . . . .	73
print.predictCox . . . . .	74
print.predictCSC . . . . .	75
print.riskRegression . . . . .	76
print.Score . . . . .	76
print.subjectWeights . . . . .	77
reconstructData . . . . .	77
riskRegression . . . . .	78
rowCenter_cpp . . . . .	81
rowCumSum . . . . .	82
rowMultiply_cpp . . . . .	82
rowScale_cpp . . . . .	83
rowSumsCrossprod . . . . .	84
rsquared . . . . .	84
sampleData . . . . .	87
Score.list . . . . .	88
selectCox . . . . .	94
selectJump . . . . .	95
simMelanoma . . . . .	96
sliceMultiply_cpp . . . . .	97
sliceScale_cpp . . . . .	97
splitStrataVar . . . . .	98
subjectWeights . . . . .	99
summary.FGR . . . . .	100
summary.riskRegression . . . . .	101
SurvResponseVar . . . . .	101

as.data.table.predictCox

*Turn predictCox object into a data.table*

---

### Description

Turn predictCox object into a data.table

### Usage

```
## S3 method for class 'predictCox'  
as.data.table(x, keep.rownames = FALSE, ...)
```

### Arguments

x	object obtained with function predictCox
keep.rownames	not used
...	not used

---

as.data.table.predictCSC

*Turn predictCSC object into a data.table*

---

### Description

Turn predictCSC object into a data.table

### Usage

```
## S3 method for class 'predictCSC'  
as.data.table(x, keep.rownames = FALSE, ...)
```

### Arguments

x	object obtained with function predictCSC
keep.rownames	not used
...	not used

---

ate *Compute the average treatment effects via the g-formula*

---

### Description

Use the g-formula to estimate the average treatment effect based on Cox regression with or without competing risks

### Usage

```
ate(object, data, formula, treatment, contrasts = NULL, times, cause,
    landmark, conf.level = 0.95, se = TRUE, band = FALSE, B = 0,
    bootci.method = "perc", nsim.band = ifelse(band, 1000, 0), seed,
    handler = "foreach", mc.cores = 1, verbose = TRUE, store.iid = "full",
    ...)
```

### Arguments

object	outcome model which describes how event risk depends on treatment and covariates. The object carry its own call and have a predictRisk method. See examples.
data	data set in which to evaluate risk predictions based on the outcome model
formula	For analyses with time-dependent covariates, the response formula. See examples.
treatment	name of the treatment variable
contrasts	the levels of the treatment variable to be compared
times	time points at which to evaluate risks
cause	the cause of interest
landmark	for models with time-dependent covariates the landmark time(s) of evaluation. In this case, argument time may only be one value and for the prediction of risks it is assumed that that the covariates do not change between landmark and landmark+time.
conf.level	Numeric value between 0 and 1 (default is 0.05). Confidence level of the confidence intervals.
se	Logical. If TRUE compute standard errors and confidence intervals
band	Logical. If TRUE compute confidence bands across time points.
B	the number of bootstrap replications used to compute the confidence intervals. If it equals 0, then Wald-type confidence intervals are computed. They rely on the standard error estimated using the influence function of the estimator.
bootci.method	Character. Method for constructing bootstrap confidence intervals. Either "perc" (the default), "norm", "basic", "stud", or "bca". Argument passed to boot::boot.ci.
nsim.band	the number of simulations used to compute the quantiles for the confidence bands.

seed	An integer used to generate seeds for bootstrap and to achieve reproducible results.
handler	parallel handler for bootstrap. Either "mclapply" or "foreach". If "foreach" use doParallel to create a cluster.
mc.cores	Passed to parallel::mclapply or doParallel::registerDoParallel. The number of cores to use, i.e. at most how many child processes will be run simultaneously. The option is initialized from environment variable MC_CORES if set.
verbose	Logical. If TRUE inform about estimated run time.
store.iid	Implementation used to estimate the standard error. Can be "full" or "minimal". "minimal" requires less memory but can only estimate the standard for the difference between treatment effects (and not for the ratio).
...	passed to predictRisk

### Value

An object of class `ate` containing:

- `meanRisk`: a `data.table` object containing the ATE at each time and for each treatment level.
- `riskComparison`: a `data.table` object comparing the ATE between two treatment levels.
- `treatment`: the name of the treatment variable.
- `contrasts`: the levels of the treatment variable that were compared.
- `times`: the time points at which the ATE was computed.
- `se`: Logical. if TRUE compute the standard errors and confidence intervals of the ATE
- `B`: the number of bootstrap samples.
- `band`: Logical. If TRUE confidence bands are computed.
- `nsim.band`: the number of simulations used to compute the quantiles for the confidence bands.
- `seeds`: the seed used to generate the bootstrap sample. Not used when the influence function is used to compute the standard errors of the ATE.
- `conf.level`: the confidence level of the confidence intervals.

### Author(s)

Brice Ozenne <broz@sund.ku.dk> and Thomas Alexander Gerds <tag@biostat.ku.dk>

### Examples

```
library(survival)
library(rms)

set.seed(10)

#### Survival settings ####
#### ATE with Cox model ####

## generate data
```

```

n <- 100
dtS <- sampleData(n, outcome="survival")
dtS$time <- round(dtS$time,1)
dtS$X1 <- factor(rbinom(n, prob = c(0.3,0.4) , size = 2), labels = paste0("T",0:2))

## estimate the Cox model
fit <- cph(formula = Surv(time,event)~ X1+X2,data=dtS,y=TRUE,x=TRUE)

## compute the ATE at times 5, 6, 7, and 8 using X1 as the treatment variable
## Not run:
## only punctual estimate (argument se = FALSE)
ateFit1a <- ate(fit, data = dtS, treatment = "X1", times = 5:8,
               se = FALSE)

## standard error / confidence intervals computed using the influence function
## (argument se = TRUE and B = 0)
ateFit1b <- ate(fit, data = dtS, treatment = "X1", times = 5:8,
               se = TRUE, B = 0)

## bootstrap confidence intervals: studentized Wald type
ateFit1c <- ate(fit, data = dtS, treatment = "X1", times = 5,
               seed=3,se = TRUE, B = 100)
## bootstrap confidence intervals: studentized Wald type
ateFit1d <- ate(fit, data = dtS, treatment = "X1", times = 5,
               seed=3,bootci.method="quantile",se = TRUE, B = 100)

## same as before with in addition the confidence bands for the ATE
## (argument band = TRUE)
ateFit1e <- ate(fit, data = dtS, treatment = "X1", times = 5:8,
               se = TRUE, band = TRUE, B = 0)

## standard error / confidence intervals computed using 100 bootstrap samples
## (argument se = TRUE and B = 100)
ateFit1d <- ate(fit, data = dtS, treatment = "X1",
               times = 5:8, se = TRUE, B = 100)
## NOTE: for real applications 100 bootstrap samples is not enough

## same but using 2 cpus for generating and analyzing the bootstrap samples
## (parallel computation, argument mc.cores = 2)
ateFit1e <- ate(fit, data = dtS, treatment = "X1",
               times = 5:8, se = TRUE, B = 100, mc.cores = 2)

## End(Not run)

#### Survival settings without censoring ####
#### ATE with glm                               ####

## generate data
n <- 100
dtS <- sampleData(n, outcome="survival")
dtS[, event5 := eventtime<=5]
dtS[, X2 := as.numeric(X2)]

```

```

## estimate the Cox model
fit <- glm(formula = event5 ~ X1+X2, data=dtS, family = "binomial")

## compute the ATE at times 5 using X1 as the treatment variable
## only punctual estimate (argument se = FALSE)
ateFit1a <- ate(fit, data = dtS, treatment = "X1", times = 5,
               se = FALSE)
ateFit1a

## Not run:
## standard error / confidence intervals computed using the influence function
ateFit1b <- ate(fit, data = dtS, treatment = "X1", times = 5,
               se = TRUE, B = 0)
ateFit1b

## standard error / confidence intervals computed using 100 bootstrap samples
ateFit1d <- ate(fit, data = dtS, treatment = "X1",
               times = 5, se = TRUE, B = 100)
ateFit1d

## using lava
ateLava <- estimate(fit, function(p, data){
  a <- p["(Intercept)"]; b <- p["X11"]; c <- p["X2"];
  R.X11 <- expit(a + b + c * data[["X2"]])
  R.X10 <- expit(a + c * data[["X2"]])
  list(risk0=R.X10,risk1=R.X11,riskdiff=R.X10-R.X11)},
      average=TRUE)
ateLava

## End(Not run)

#### Competing risks settings          #####
#### ATE with cause specific Cox regression #####

## Not run:
## generate data
n <- 500
dt <- sampleData(n, outcome="competing.risks")
dt$time <- round(dt$time,1)
dt$X1 <- factor(rbinom(n, prob = c(0.2,0.3,0.2) , size = 3), labels = paste0("T",0:3))

## estimate cause specific Cox model
fitCR <- CSC(Hist(time,event)~ X1+X8,data=dt,cause=1)

## compute the ATE at times 5, 6, 7, and 8 using X1 as the treatment variable
ateFit2a <- ate(fitCR, data = dt, treatment = "X1", times = 5:8, cause = 1,
               se = FALSE)

## standard error / confidence intervals computed using the influence function
## (argument se = TRUE and B = 0)
ateFit2b <- ate(fitCR, data = dt, treatment = "X1", times = 5:8, cause = 1,
               se = TRUE, B = 0)

```



```

## same as before with in addition the confidence bands for the ATE
## (argument band = TRUE)
ateFit2c <- ate(fitCR, data = dt, treatment = "X1", times = 5:8, cause = 1,
              se = TRUE, band = TRUE, B = 0)

## standard error / confidence intervals computed using 100 bootstrap samples
## (argument se = TRUE and B = 100)
ateFit2d <- ate(fitCR, data = dt, treatment = "X1", times = 5:8, cause = 1,
              se = TRUE, B = 100)
## NOTE: for real applications 100 bootstrap samples is not enough

## same but using 2 cpus for generating and analyzing the bootstrap samples
## (parallel computation, argument mc.cores = 2)
ateFit2e <- ate(fitCR, data = dt, treatment = "X1", times = 5:8, cause = 1,
              se = TRUE, B = 100, mc.cores = 2)

## End(Not run)

#### time-dependent covariates ####
## Not run:
library(survival)
fit <- coxph(Surv(time, status) ~ celltype+karno + age + trt, veteran)
vet2 <- survSplit(Surv(time, status) ~., veteran,
                cut=c(60, 120), episode="timegroup")
fitTD <- coxph(Surv(tstart, time, status) ~ celltype+karno + age + trt,
              data= vet2,x=1)
set.seed(16)
resVet <- ate(fitTD,formula=Hist(entry=tstart,time=time,event=status)~1,
            data = vet2, treatment = "celltype", contrasts = NULL,
            times=5,verbose=1,
            landmark = c(0,30,60,90), cause = 1, B = 20, se = 1,
            band = FALSE, mc.cores=1)
resVet

## End(Not run)

## Not run:
set.seed(137)
d=sampleDataTD(127)
library(survival)
d[,status:=1*(event==1)]
## ignore competing risks
cox1TD <- coxph(Surv(start,time, status,type="counting") ~ X3+X5+X6+X8, data=d)
resTD1 <- ate(cox1TD,formula=Hist(entry=start,time=time,event=status)~1,
            data = d, treatment = "X3", contrasts = NULL,
            times=.5,verbose=1,
            landmark = c(0,0.5,1), B = 20, se = 1,
            band = FALSE, mc.cores=1)
resTD1
## adjust for competing risks
cscTD <- CSC(Hist(time=time, event=event,entry=start) ~ X3+X5+X6+X8, data=d)
set.seed(16)
resTD <- ate(cscTD,formula=Hist(entry=start,time=time,event=event)~1,

```

```

      data = d, treatment = "X3", contrasts = NULL,
      times=.5,verbose=1,
      landmark = c(0,0.5,1), cause = 1, B = 20, se = 1,
      band = FALSE, mc.cores=1)
resTD

## End(Not run)

```

---

autoplot.ate

*Plot predictions from a Cause-specific Cox proportional hazard regression*

---

### Description

Plot predictions from a Cause-specific Cox proportional hazard regression

### Usage

```

## S3 method for class 'ate'
autoplot(object, ci = FALSE, band = FALSE, plot = TRUE,
         digits = 2, alpha = 0.1, ...)

```

### Arguments

object	object obtained with the function predictCox.
ci	Logical. If TRUE display the confidence intervals for the predictions.
band	Logical. If TRUE display the confidence bands for the predictions.
plot	Logical. Should the graphic be plotted.
digits	integer indicating the number of decimal places
alpha	transparency of the confidence bands. Argument passed to ggplot2::geom_ribbon.
...	not used. Only for compatibility with the plot method.

### Examples

```

library(survival)
library(rms)

set.seed(10)
n <- 1e2

## Cox model
dtS <- sampleData(n,outcome="survival")

fit <- cph(formula = Surv(time,event)~ X1+X2,data=dtS,y=TRUE,x=TRUE)

seqTimes <- sort(unique(fit$y[,1]))
seqTimes5 <- seqTimes[seqTimes>5 & seqTimes<10]

```

```
ateFit <- ate(fit, data = dtS, treatment = "X1", contrasts = NULL,
             times = seqTimes, B = 0, band = TRUE, nsim.band = 500, y = TRUE,
             mc.cores=1)
autoplot(ateFit, band = TRUE, ci = TRUE)
```

---

autoplot.predictCox    *Plot predictions from a Cox model*

---

## Description

Plot predictions from a Cox model

## Usage

```
## S3 method for class 'predictCox'
autoplot(object, type = NULL, ci = FALSE,
         band = FALSE, group.by = "row", reduce.data = FALSE, plot = TRUE,
         digits = 2, alpha = NA, ...)
```

## Arguments

object	object obtained with the function predictCox.
type	the type of predicted value to display. Choices are "hazard" the hazard function, "cumhazard" the cumulative hazard function, or "survival" the survival function.
ci	Logical. If TRUE display the confidence intervals for the predictions.
band	Logical. If TRUE display the confidence bands for the predictions.
group.by	The grouping factor used to color the prediction curves. Can be "row", "strata", or "covariates".
reduce.data	Logical. If TRUE only the covariates that does take identical values for all observations are displayed.
plot	Logical. Should the graphic be plotted.
digits	integer indicating the number of decimal places
alpha	transparency of the confidence bands. Argument passed to ggplot2::geom_ribbon.
...	not used. Only for compatibility with the plot method.

## Examples

```
library(survival)
library(ggplot2)

## predictions ##
d <- sampleData(1e2, outcome = "survival")
m.cox <- coxph(Surv(time,event)~ X1 + X2 + X3,
```

```

      data = d, x = TRUE, y = TRUE)
dt.basehaz <- predictCox(m.cox)
ggplot(as.data.table(dt.basehaz), aes(x = time, y = survival)) + geom_point() + geom_line()

pred.cox <- predictCox(m.cox, newdata = d[1:4,],
  times = 1:5, type = "survival", keep.newdata = TRUE)
autoplot(pred.cox)
autoplot(pred.cox, group.by = "covariates")
autoplot(pred.cox, group.by = "covariates", reduce.data = TRUE)

m.cox.strata <- coxph(Surv(time,event)~ strata(X1) + strata(X2) + X3 + X6,
  data = d, x = TRUE, y = TRUE)
pred.cox.strata <- predictCox(m.cox.strata, newdata = d[1,,drop=FALSE],
  time = 1:5, keep.newdata = TRUE)
autoplot(pred.cox.strata, type = "survival")
autoplot(pred.cox.strata, type = "survival", group.by = "strata")
res <- autoplot(pred.cox.strata, type = "survival",
  group.by = "covariates")

# customize display
res$plot + geom_point(data = res$data, size = 5)

## predictions with confidence interval
pred.cox <- predictCox(m.cox, newdata = d[1,,drop=FALSE],
  times = 1:5, type = "survival", se = TRUE, keep.newdata = TRUE)
autoplot(pred.cox, ci = TRUE)

## predictions with confidence bands
pred.cox <- predictCox(m.cox, newdata = d[1,,drop=FALSE],
  times = 1:5, type = "survival", nsim.band = 500, band = TRUE, keep.newdata = TRUE)
autoplot(pred.cox, band = TRUE)

```

---

autoplot.predictCSC *Plot predictions from a Cause-specific Cox proportional hazard regression*

---

## Description

Plot predictions from a Cause-specific Cox proportional hazard regression

## Usage

```

## S3 method for class 'predictCSC'
autoplot(object, ci = FALSE, band = FALSE,
  group.by = "row", reduce.data = FALSE, plot = TRUE, digits = 2,
  alpha = NA, ...)

```

**Arguments**

object	object obtained with the function predictCox.
ci	Logical. If TRUE display the confidence intervals for the predictions.
band	Logical. If TRUE display the confidence bands for the predictions.
group.by	The grouping factor used to color the prediction curves. Can be "row", "strata", or "covariates".
reduce.data	Logical. If TRUE only the covariates that does take indentical values for all observations are displayed.
plot	Logical. Should the graphic be plotted.
digits	integer indicating the number of decimal places
alpha	transparency of the confidence bands. Argument passed to ggplot2::geom_ribbon.
...	not used. Only for compatibility with the plot method.

**Examples**

```
## no strata
d <- sampleData(1e2, outcome = "competing.risks")
m.CSC <- CSC(Hist(time,event)~ X1 + X2 + X6, data = d)

pred.CSC <- predict(m.CSC, newdata = d[1:2,], time = 1:5, cause = 1)
autoplot(pred.CSC)

pred.CSC <- predict(m.CSC, newdata = d[1:3,],
                    time = 1:5, cause = 1, se = TRUE, keep.newdata = TRUE)

## strata
library(survival)
m.SCSC <- CSC(Hist(time,event)~ strata(X1) + strata(X2) + X6,
              data = d)
pred.SCSC <- predict(m.SCSC, time = 1:3, newdata = d[1:4,],
                    cause = 1, se = TRUE, keep.newdata = TRUE, keep.strata = TRUE)
autoplot(pred.SCSC, group.by = "strata")
```

---

boot2pvalue

*Compute the p.value from the distribution under H1*


---

**Description**

Compute the p.value associated with the estimated statistic using a bootstrap sample of its distribution under H1.

**Usage**

```
boot2pvalue(x, null, estimate = NULL, alternative = "two.sided",
            FUN.ci = quantileCI, tol = .Machine$double.eps^0.5)
```

**Arguments**

<code>x</code>	[numeric vector] a vector of bootstrap estimates of the statistic.
<code>null</code>	[numeric] value of the statistic under the null hypothesis.
<code>estimate</code>	[numeric] the estimated statistic.
<code>alternative</code>	[character] a character string specifying the alternative hypothesis, must be one of "two.sided" (default), "greater" or "less".
<code>FUN.ci</code>	[function] the function used to compute the confidence interval. Must take <code>x</code> , <code>alternative</code> , <code>conf.level</code> and <code>sign.estimate</code> as arguments and only return the relevant limit (either upper or lower) of the confidence interval.
<code>tol</code>	[numeric] the absolute convergence tolerance.

**Details**

For test statistic close to 0, this function returns 1.

For positive test statistic, this function search the quantile  $\alpha$  such that:

- `quantile(x, probs = alpha)=0` when the argument `alternative` is set to "greater".
- `quantile(x, probs = 0.5*alpha)=0` when the argument `alternative` is set to "two.sided".

If the argument `alternative` is set to "less", it returns 1.

For negative test statistic, this function search the quantile  $\alpha$  such that:

- `quantile(x, probs = 1-alpha)=0` when the argument `alternative` is set to "less".
- `quantile(x, probs = 1-0.5*alpha)=0` when the argument `alternative` is set to "two.sided".

If the argument `alternative` is set to "greater", it returns 1.

**Examples**

```
set.seed(10)

#### no effect ####
x <- rnorm(1e3)
boot2pvalue(x, null = 0, estimate = mean(x), alternative = "two.sided")
## expected value of 1
boot2pvalue(x, null = 0, estimate = mean(x), alternative = "greater")
## expected value of 0.5
boot2pvalue(x, null = 0, estimate = mean(x), alternative = "less")
## expected value of 0.5

#### positive effect ####
x <- rnorm(1e3, mean = 1)
boot2pvalue(x, null = 0, estimate = 1, alternative = "two.sided")
## expected value of 0.32 = 2*pnorm(q = 0, mean = -1) = 2*mean(x<=0)
```

```

boot2pvalue(x, null = 0, estimate = 1, alternative = "greater")
## expected value of 0.16 = pnorm(q = 0, mean = 1) = mean(x<=0)
boot2pvalue(x, null = 0, estimate = 1, alternative = "less")
## expected value of 0.84 = 1-pnorm(q = 0, mean = 1) = mean(x>=0)

#### negative effect ####
x <- rnorm(1e3, mean = -1)
boot2pvalue(x, null = 0, estimate = -1, alternative = "two.sided")
## expected value of 0.32 = 2*(1-pnorm(q = 0, mean = -1)) = 2*mean(x>=0)
boot2pvalue(x, null = 0, estimate = -1, alternative = "greater")
## expected value of 0.84 = pnorm(q = 0, mean = -1) = mean(x<=0)
boot2pvalue(x, null = 0, estimate = -1, alternative = "less") # pnorm(q = 0, mean = -1)
## expected value of 0.16 = 1-pnorm(q = 0, mean = -1) = mean(x>=0)

```

---

boxplot.Score

*Boxplot risk quantiles*


---

## Description

Retrospective boxplots of risk quantiles conditional on outcome

## Usage

```

## S3 method for class 'Score'
boxplot(x, model, reference, type, timepoint, overall = 1L,
        lwd = 3, xlim, xlab = "", main, outcome.label, outcome.label.offset = 0,
        event.labels, reline = (type == "diff"), add = FALSE, ...)

```

## Arguments

x	Score object obtained by calling function Score.
model	Choice of risk prediction model
reference	Choice of reference risk prediction model for calculation of risk differences.
type	Either "risk" for predicted risks or "diff" for differences between predicted risks.
timepoint	time point specifying the prediction horizon
overall	Logical. Tag to be documented.
lwd	line width
xlim	x-axis limits
xlab	x-axis label
main	title of plot
outcome.label	Title label for column which shows the outcome status
outcome.label.offset	Vertical offset for outcome.label
event.labels	Labels for the different events (causes).

refline	Logical, for type="diff" only. If TRUE draw a red vertical line at 0.
add	Logical. Tag to be documented.
...	not used

## Examples

```
# binary outcome
db=sampleData(40,outcome="binary")
fitconv=glm(Y~X3+X5,data=db,family=binomial)
fitnew=glm(Y~X1+X3+X5+X6+X7,data=db,family=binomial)
scoreobj=Score(list(new=fitnew,conv=fitconv),
               formula=Y~1,contrasts=list(c(2,1)),
               data=db,summary="riskQuantile",null.model=FALSE)
boxplot(scoreobj)

# survival outcome
library(survival)
ds=sampleData(40,outcome="survival")
fitconv=coxph(Surv(time,event)~X6,data=ds,x=TRUE,y=TRUE)
fitnew=coxph(Surv(time,event)~X6+X9,data=ds,x=TRUE,y=TRUE)
## Not run:
scoreobj=Score(list("conventional model"=fitconv,"new model"=fitnew),
               formula=Hist(time,event)~1, data=ds,
               summary="riskQuantile",metrics=NULL, plots=NULL,
               c(0,0.25,0.5,0.75,1),
               times=5,null.model=FALSE)
boxplot(scoreobj)

scoreobj1=Score(list("conventional model"=fitconv,"new model"=fitnew),
               formula=Hist(time,event)~1, data=ds,
               summary="riskQuantile",metrics=NULL, plots=NULL,
               times=5,null.model=FALSE,compare=list(c(2,1)))
boxplot(scoreobj1)

## End(Not run)

# competing risks outcome
library(survival)
data(Melanoma, package = "riskRegression")
fitconv = CSC(Hist(time,status)~invasion+age+sex,data=Melanoma)
fitnew = CSC(Hist(time,status)~invasion+age+sex+logthick,data=Melanoma)
scoreobj=Score(list("Conventional model"=fitconv,"New model"=fitnew),
               formula=Hist(time,status)~1,
               data=Melanoma,metrics=NULL,summary="riskQuantile",times=5*365.25,null.model=FALSE)
boxplot(scoreobj)

# more than 2 competing risks
m=lava::lvm(~X1+X2+X3)
lava::distribution(m, "eventtime1") <- lava::coxWeibull.lvm(scale = 1/100)
lava::distribution(m, "eventtime2") <- lava::coxWeibull.lvm(scale = 1/100)
lava::distribution(m, "eventtime3") <- lava::coxWeibull.lvm(scale = 1/100)
lava::distribution(m, "censtime") <- lava::coxWeibull.lvm(scale = 1/100)
```



```

lava::regression(m,eventtime2~X3)=1.3
m <- lava::eventTime(m,
time ~ min(eventtime1 = 1, eventtime2 = 2, eventtime3 = 3, censtime = 0), "event")
set.seed(101)
dcr=as.data.table(lava::sim(m,101))
fitOld = CSC(Hist(time,event)~X1+X2,data=dcr)
fitNew = CSC(Hist(time,event)~X1+X2+X3,data=dcr)
scoreobj=Score(list("Conventional model"=fitOld,"New model"=fitNew),
               formula=Hist(time,event)~1,
               data=dcr,summary="riskQuantile",times=5,null.model=FALSE)
boxplot(scoreobj)

```

---

calcSeCox

*Computation of standard errors for predictions*


---

### Description

Compute the standard error associated to the predictions from Cox regression model using a first order von Mises expansion of the functional (cumulative hazard or survival).

### Usage

```

calcSeCox(object, times, nTimes, type, Lambda0, object.n, object.time,
           object.eXb, object.strata, nStrata, new.eXb, new.LPdata, new.strata,
           new.survival, new.cumhazard, nVar, log.transform, export, store.iid)

```

### Arguments

object	The fitted Cox regression model object either obtained with coxph (survival package) or cph (rms package).
times	Vector of times at which to return the estimated hazard/survival.
nTimes	the length of the argument times.
type	One or several strings that match (either in lower or upper case or mixtures) one or several of the strings "hazard", "cumhazard", "survival".
Lambda0	the baseline hazard estimate returned by BaseHazStrata_cpp.
object.n	the number of observations in the dataset used to estimate the object.
object.time	the time to event of the observations used to estimate the object.
object.eXb	the exponential of the linear predictor relative to the observations used to estimate the object.
object.strata	the strata index of the observations used to estimate the object.
nStrata	the number of strata.
new.eXb	the linear predictor evaluated for the new observations
new.LPdata	the variables involved in the linear predictor for the new observations

<code>new.strata</code>	the strata indicator for the new observations
<code>new.survival</code>	the survival evaluated for the new observations
<code>new.cumhazard</code>	the cumulative hazard evaluated for the new observations
<code>nVar</code>	the number of variables that form the linear predictor
<code>log.transform</code>	Should the variance/influence function be computed on the log or log(-log) scale
<code>export</code>	can be "iid" to return the value of the influence function for each observation "se" to return the standard error for a given timepoint
<code>store.iid</code>	Implementation used to estimate the influence function and the standard error. Can be "full" or "minimal". See the details section.

### Details

Can also return the empirical influence function of the functionals cumulative hazard or survival or the sum over the observations of the empirical influence function.

`store.iid="full"` compute the influence function for each observation at each time in the argument times before computing the standard error / influence functions. `store.iid="minimal"` recompute for each subject specific prediction the influence function for the baseline hazard. This avoid to store all the influence functions but may lead to repeated evaluation of the influence function. This solution is therefore efficient more efficient in memory usage but may not be in term of computation time.

### Value

A list optionally containing the standard error for the survival, cumulative hazard and hazard.

### Author(s)

Brice Ozenne broz@sund.ku.dk, Thomas A. Gerds tag@biostat.ku.dk

---

<code>calcSeCSC</code>	<i>Standard error of the absolute risk predicted from cause-specific Cox models</i>
------------------------	---

---

### Description

Standard error of the absolute risk predicted from cause-specific Cox models using a first order von Mises expansion of the absolute risk functional.

### Usage

```
calcSeCSC(object, cif, hazard, cumhazard, object.time, object.maxtime, eXb,
  new.LPdata, new.strata, times, surv.type, new.n, cause, nCause, nVar,
  log.transform, export, store.iid)
```

**Arguments**

<code>object</code>	The fitted cause specific Cox model
<code>cif</code>	the cumulative incidence function at each prediction time for each individual.
<code>hazard</code>	list containing the baseline hazard for each cause in a matrix form. Columns correspond to the strata.
<code>cumhazard</code>	list containing the cumulative baseline hazard for each cause in a matrix form. Columns correspond to the strata.
<code>object.time</code>	a vector containing all the events regardless to the cause.
<code>object.maxtime</code>	a matrix containing the latest event in the strata of the observation for each cause.
<code>eXb</code>	a matrix containing the exponential of the linear predictor evaluated for the new observations (rows) for each cause (columns)
<code>new.LPdata</code>	a list of design matrices for the new observations for each cause.
<code>new.strata</code>	a matrix containing the strata indicator for each observation and each cause.
<code>times</code>	the time points at which to evaluate the predictions.
<code>surv.type</code>	see the <code>surv.type</code> argument of <a href="#">CSC</a> .
<code>new.n</code>	the number of new observations.
<code>cause</code>	the cause of interest.
<code>nCause</code>	the number of causes.
<code>nVar</code>	the number of variables that form the linear predictor in each Cox model
<code>log.transform</code>	Should the variance/influence function be computed on the log(-log) scale
<code>export</code>	can be "iid" to return the value of the influence function for each observation "se" to return the standard error for a given timepoint
<code>store.iid</code>	the method used to compute the influence function and the standard error. Can be "full" or "minimal". See the details section.

**Details**

Can also return the empirical influence function of the functionals cumulative hazard or survival or the sum over the observations of the empirical influence function.

`store.iid="full"` compute the influence function for each observation at each time in the argument `times` before computing the standard error / influence functions. `store.iid="minimal"` recompute for each subject specific prediction the influence function for the baseline hazard. This avoid to store all the influence functions but may lead to repeated evaluation of the influence function. This solution is therefore efficient more efficient in memory usage but may not be in term of computation time.

---

coef.CauseSpecificCox *Extract coefficients from a Cause-Specific Cox regression model*

---

### Description

Extract coefficients from a Cause-Specific Cox regression model

### Usage

```
## S3 method for class 'CauseSpecificCox'
coef(object, ...)
```

### Arguments

object	Object obtained with CSC
...	not used

---

coef.riskRegression *Extract coefficients from riskRegression model*

---

### Description

Extract coefficients from riskRegression model

### Usage

```
## S3 method for class 'riskRegression'
coef(object, digits = 3, eps = 10^-4, ...)
```

### Arguments

object	Object obtained with ARR or LRR or riskRegression
digits	Number of digits
eps	P-values below this number are shown as <eps
...	not used

---

colCenter_cpp	<i>Apply - by column</i>
---------------	--------------------------

---

**Description**

Fast computation of `sweep(X, MARGIN = 1, FUN = "-", STATS = center)`

**Usage**

```
colCenter_cpp(X, center)
```

**Arguments**

X	A matrix.
center	a numeric vector of length equal to the number of rows of x

**Value**

A matrix of same size as X.

**Author(s)**

Brice Ozenne <broz@sund.ku.dk>

**Examples**

```
x <- matrix(1,6,5)
sweep(x, MARGIN = 1, FUN = "-", STATS = 1:6)
colCenter_cpp(x, 1:6 )
```

---

colCumSum	<i>Apply cumsum in each column</i>
-----------	------------------------------------

---

**Description**

Fast computation of `apply(x,2,cumsum)`

**Usage**

```
colCumSum(x)
```

**Arguments**

x	A matrix.
---	-----------

**Value**

A matrix of same size as x.

**Author(s)**

Thomas Alexander Gerds <tag@biostat.ku.dk>

**Examples**

```
x <- matrix(1:8,ncol=2)
colCumSum(x)
```

---

colMultiply_cpp	<i>Apply * by column</i>
-----------------	--------------------------

---

**Description**

Fast computation of `sweep(X, MARGIN = 1, FUN = "*", STATS = scale)`

**Usage**

```
colMultiply_cpp(X, scale)
```

**Arguments**

x	A matrix.
scale	a numeric vector of length equal to the number of rows of x

**Value**

A matrix of same size as X.

**Author(s)**

Brice Ozenne <broz@sund.ku.dk>

**Examples**

```
x <- matrix(1,6,5)
sweep(x, MARGIN = 1, FUN = "*", STATS = 1:6)
colMultiply_cpp(x, 1:6 )
```

---

colScale_cpp	<i>Apply / by column</i>
--------------	--------------------------

---

**Description**

Fast computation of `sweep(X, MARGIN = 1, FUN = "/", STATS = scale)`

**Usage**

```
colScale_cpp(X, scale)
```

**Arguments**

<code>X</code>	A matrix.
<code>scale</code>	a numeric vector of length equal to the number of rows of <code>x</code>

**Value**

A matrix of same size as `X`.

**Author(s)**

Brice Ozenne <broz@sund.ku.dk>

**Examples**

```
x <- matrix(1,6,5)
sweep(x, MARGIN = 1, FUN = "/", STATS = 1:6)
colScale_cpp(x, 1:6 )
```

---

colSumsCrossprod	<i>Apply crossprod and colSums</i>
------------------	------------------------------------

---

**Description**

Fast computation of `crossprod(colSums(X),Y)`

**Usage**

```
colSumsCrossprod(X, Y, transposeY)
```

**Arguments**

<code>X</code>	A matrix with dimensions $k \times n$ . Hence the result of <code>colSums(X)</code> has length $n$ .
<code>Y</code>	A matrix with dimensions $n \times m$ . Can be a matrix with dimension $m \times n$ but then <code>transposeY</code> should be TRUE.
<code>transposeY</code>	Logical. If TRUE transpose <code>Y</code> before matrix multiplication.

**Value**

A vector of length  $m$ .

**Author(s)**

Thomas Alexander Gerds <tag@biostat.ku.dk>

**Examples**

```
x <- matrix(1:8,ncol=2)
y <- matrix(1:16,ncol=8)
colSumsCrossprod(x,y,0)
```

```
x <- matrix(1:8,ncol=2)
y <- matrix(1:16,ncol=2)
colSumsCrossprod(x,y,1)
```

---

confBandCox

*Compute quantiles of a gaussian process*

---

**Description**

Compute quantiles of a gaussian process

**Usage**

```
confBandCox(iid, se, n.sim, conf.level)
```

**Arguments**

<code>iid</code>	The iid decomposition of the estimator over time.
<code>se</code>	The variance of the estimate over time.
<code>n.sim</code>	The number of simulations used to compute the quantiles.
<code>conf.level</code>	Level of confidence.



---

coxBaseEstimator	<i>Extract the type of estimator for the baseline hazard</i>
------------------	--

---

**Description**

Extract the type of estimator for the baseline hazard

**Usage**

```
coxBaseEstimator(object)

## S3 method for class 'coxph'
coxBaseEstimator(object)

## S3 method for class 'phreg'
coxBaseEstimator(object)
```

**Arguments**

object	The fitted Cox regression model object either obtained with coxph (survival package), cph (rms package), or phreg (mets package).
--------	---

**Author(s)**

Brice Ozenne broz@sund.ku.dk

---

coxCenter	<i>Extract the mean value of the covariates</i>
-----------	---

---

**Description**

Extract the mean value of the covariates

**Usage**

```
coxCenter(object)

## S3 method for class 'cph'
coxCenter(object)

## S3 method for class 'coxph'
coxCenter(object)

## S3 method for class 'phreg'
coxCenter(object)
```

**Arguments**

object            The fitted Cox regression model object either obtained with coxph (survival package), cph (rms package), or phreg (mets package).

**Author(s)**

Brice Ozenne broz@sund.ku.dk

---

coxDesign

*Extract the design matrix used to train a Cox model*

---

**Description**

Extract the design matrix used to train a Cox model. Should contain the time of event, the type of event, the variable for the linear predictor, the strata variables and the date of entry (in case of delayed entry).

**Usage**

```
coxDesign(object, center)

## S3 method for class 'coxph'
coxDesign(object, center = FALSE)

## S3 method for class 'phreg'
coxDesign(object, center = FALSE)
```

**Arguments**

object            The fitted Cox regression model object either obtained with coxph (survival package), cph (rms package), or phreg (mets package).

center            logical. Should the variable of the linear predictor be centered ?

**Author(s)**

Brice Ozenne broz@sund.ku.dk

---

coxFormula	<i>Extract the formula from a Cox model</i>
------------	---

---

**Description**

Extract the formula from a Cox model

**Usage**

```
coxFormula(object)

## S3 method for class 'cph'
coxFormula(object)

## S3 method for class 'coxph'
coxFormula(object)

## S3 method for class 'phreg'
coxFormula(object)
```

**Arguments**

object	The fitted Cox regression model object either obtained with coxph (survival package), cph (rms package), or phreg (mets package).
--------	---

**Author(s)**

Brice Ozenne broz@sund.ku.dk

---

coxLP	<i>Compute the linear predictor of a Cox model</i>
-------	--

---

**Description**

Compute the linear predictor of a Cox model

**Usage**

```
coxLP(object, data, center)

## S3 method for class 'cph'
coxLP(object, data, center)

## S3 method for class 'coxph'
coxLP(object, data, center)

## S3 method for class 'phreg'
coxLP(object, data, center)
```

**Arguments**

object	The fitted Cox regression model object either obtained with coxph (survival package), cph (rms package), or phreg (mets package).
data	a data.frame or a data.table
center	should the linear predictor be computed after centering the covariates

**Details**

In case of empty linear predictor returns a vector of 0 with the same length as the number of rows of the dataset

**Author(s)**

Brice Ozenne broz@sund.ku.dk

---

coxN

*Extract the number of observations from a Cox model*

---

**Description**

Extract the number of observations from a Cox model

**Usage**

```
coxN(object)
```

```
## S3 method for class 'cph'  
coxN(object)
```

```
## S3 method for class 'coxph'  
coxN(object)
```

```
## S3 method for class 'phreg'  
coxN(object)
```

**Arguments**

object	The fitted Cox regression model object either obtained with coxph (survival package), cph (rms package), or phreg (mets package).
--------	---

**Author(s)**

Brice Ozenne broz@sund.ku.dk

---

coxSpecialStrata	<i>Special character for strata in Cox model</i>
------------------	--

---

**Description**

Return the special character used to indicate the strata variables of the Cox model

**Usage**

```
coxSpecialStrata(object)

## S3 method for class 'coxph'
coxSpecialStrata(object)

## S3 method for class 'cph'
coxSpecialStrata(object)

## S3 method for class 'phreg'
coxSpecialStrata(object)
```

**Arguments**

object	The fitted Cox regression model object either obtained with coxph (survival package), cph (rms package), or phreg (mets package).
--------	---

**Author(s)**

Brice Ozenne broz@sund.ku.dk

---

coxStrata	<i>Define the strata for a new dataset</i>
-----------	--

---

**Description**

Define the strata in a dataset to match those of a stratified Cox model

**Usage**

```
coxStrata(object, data, sterms, strata.vars, levels, strata.levels)

## S3 method for class 'coxph'
coxStrata(object, data, sterms, strata.vars, levels,
  strata.levels)

## S3 method for class 'coxph'
coxStrata(object, data, sterms, strata.vars, levels,
```

```

    strata.levels)

## S3 method for class 'phreg'
coxStrata(object, data, sterms, strata.vars, levels,
          strata.levels)

```

### Arguments

object	The fitted Cox regression model object either obtained with <code>coxph</code> (survival package), <code>cph</code> (rms package), or <code>phreg</code> (mets package).
data	a <code>data.frame</code> or a <code>data.table</code>
sterms	terms in the formula corresponding to the strata variables
strata.vars	the name of the variables used to define the strata
levels	the strata levels that have been used to fit the Cox model
strata.levels	a named list containing for each variable used to form the strata all its possible levels

### Details

if no strata variables returns a vector of "1" (factor).

### Author(s)

Brice Ozenne broz@sund.ku.dk

---

coxVarCov

*Extract the variance covariance matrix of the beta from a Cox model*

---

### Description

Extract the variance covariance matrix of the beta from a Cox model

### Usage

```

coxVarCov(object)

## S3 method for class 'cph'
coxVarCov(object)

## S3 method for class 'coxph'
coxVarCov(object)

## S3 method for class 'phreg'
coxVarCov(object)

```

**Arguments**

object            The fitted Cox regression model object either obtained with coxph (survival package), cph (rms package), or phreg (mets package).

**Details**

Should return NULL if the Cox model has no covariate. The rows and columns of the variance covariance matrix must be named with the names used in the design matrix.

**Author(s)**

Brice Ozenne broz@sund.ku.dk

---

coxVariableName	<i>Extract variable names from a model</i>
-----------------	--

---

**Description**

Extract the name of the variables belonging to the linear predictor or used to form the strata

**Usage**

```
coxVariableName(object)
```

**Arguments**

object            The fitted Cox regression model object either obtained with coxph (survival package) or cph (rms package).

**Author(s)**

Brice Ozenne broz@sund.ku.dk

---

CSC	<i>Cause-specific Cox proportional hazard regression</i>
-----	--

---

**Description**

Interface for fitting cause-specific Cox proportional hazard regression models in competing risk.

**Usage**

```
CSC(formula, data, cause, surv.type = "hazard", fitter = "coxph", ...)
```

**Arguments**

formula	Either a single Hist formula or a list of formulas. If it is a list it must contain as many Hist formulas as there are causes when <code>surv.type="hazard"</code> and exactly two formulas when <code>surv.type="survival"</code> . If it is a list the first formula is used for the cause of interest specific Cox regression and the other formula(s) either for the other cause specific Cox regression(s) or for the Cox regression of the combined event where each cause counts as event. Note that when only one formula is given the covariates enter in exactly the same way into all Cox regression analyses.
data	A data in which to fit the models.
cause	The cause of interest. Defaults to the first cause (see Details).
surv.type	Either "hazard" (the default) or "survival". If "hazard" fit cause-specific Cox regression models for all causes. If "survival" fit one cause-specific Cox regression model for the cause of interest and also a Cox regression model for event-free survival.
fitter	Routine to fit the Cox regression models. If <code>coxph</code> use <code>survival::coxph</code> else use <code>rms::cph</code> .
...	Arguments given to <code>coxph</code> .

**Details**

The causes and their order are determined by `prodlim::getStates()` applied to the Hist object.

**Value**

models	a list with the fitted (cause-specific) Cox regression objects
response	the event history response
eventTimes	the sorted (unique) event times
surv.type	the value of <code>surv.type</code>
theCause	the cause of interest. see <code>cause</code>
causes	the other causes

**Author(s)**

Thomas A. Gerds <tag@biostat.ku.dk> and Ulla B. Mogensen

**References**

- J Benichou and Mitchell H Gail. Estimates of absolute cause-specific risk in cohort studies. *Biometrics*, pages 813–826, 1990.
- T.A. Gerds, T.H. Scheike, and P.K. Andersen. Absolute risk regression for competing risks: interpretation, link functions, and prediction. *Statistics in Medicine*, 31(29):3921–3930, 2012.

**See Also**

[coxph](#)



**Examples**

```
library(prodlim)
library(survival)
data(Melanoma)
## fit two cause-specific Cox models
## different formula for the two causes
fit1 <- CSC(list(Hist(time,status)~sex,Hist(time,status)~invasion+epicel+age),
             data=Melanoma)
print(fit1)

## model hazard of all cause mortality instead of hazard of type 2
fit1a <- CSC(list(Hist(time,status)~sex+age,Hist(time,status)~invasion+epicel+log(thick)),
             data=Melanoma,
             surv.type="surv")
print(fit1a)

## special case where cause 2 has no covariates
fit1b <- CSC(list(Hist(time,status)~sex+age,Hist(time,status)~1),
             data=Melanoma)
print(fit1b)
predict(fit1b,cause=1,times=100,newdata=Melanoma)

## same formula for both causes
fit2 <- CSC(Hist(time,status)~invasion+epicel+age,
            data=Melanoma)
print(fit2)

## combine a cause-specific Cox regression model for cause 2
## and a Cox regression model for the event-free survival:
## different formula for cause 2 and event-free survival
fit3 <- CSC(list(Hist(time,status)~sex+invasion+epicel+age,
                Hist(time,status)~invasion+epicel+age),
            surv.type="surv",
            data=Melanoma)
print(fit3)

## same formula for both causes
fit4 <- CSC(Hist(time,status)~invasion+epicel+age,
            data=Melanoma,
            surv.type="surv")
print(fit4)

## strata
fit5 <- CSC(Hist(time,status)~invasion+epicel+age+strata(sex),
            data=Melanoma,
            surv.type="surv")
print(fit5)
```

```

## sanity checks

cox1 <- coxph(Surv(time,status==1)~invasion+epicel+age+strata(sex),data=Melanoma)
cox2 <- coxph(Surv(time,status!=0)~invasion+epicel+age+strata(sex),data=Melanoma)
all.equal(cox1,fit5$models[[1]])
all.equal(cox2,fit5$models[[2]])

## predictions
##
## surv.type = "hazard": predictions for both causes can be extracted
## from the same fit
fit2 <- CSC(Hist(time,status)~invasion+epicel+age, data=Melanoma)
predict(fit2,cause=1,newdata=Melanoma[c(17,99,108),],times=c(100,1000,10000))
predictRisk(fit2,cause=1,newdata=Melanoma[c(17,99,108),],times=c(100,1000,10000))
predictRisk(fit2,cause=2,newdata=Melanoma[c(17,99,108),],times=c(100,1000,10000))
predict(fit2,cause=1,newdata=Melanoma[c(17,99,108),],times=c(100,1000,10000))
predict(fit2,cause=2,newdata=Melanoma[c(17,99,108),],times=c(100,1000,10000))

## surv.type = "surv" we need to change the cause of interest
library(survival)
fit5.2 <- CSC(Hist(time,status)~invasion+epicel+age+strata(sex),
             data=Melanoma,
             surv.type="surv", cause=2)
## now this does not work
try(predictRisk(fit5.2,cause=1,newdata=Melanoma,times=4))

## but this does
predictRisk(fit5.2,cause=2,newdata=Melanoma,times=100)
predict(fit5.2,cause=2,newdata=Melanoma,times=100)
predict(fit5.2,cause=2,newdata=Melanoma[4,],times=100)

```

---

discreteRoot

*Dichotomic search for monotone function*


---

## Description

Find the root of a monotone function on a discrete grid of value using dichotomic search

## Usage

```

discreteRoot(fn, grid, increasing = TRUE, check = TRUE,
            tol = .Machine$double.eps^0.5)

```

## Arguments

fn	[function] objective function to minimize in absolute value.
grid	[vector] possible minimizers.
increasing	[logical] is the function fn increasing?

check            [logical] should the program check that fn takes a different sign for the first vs. the last value of the grid?

tol              [numeric] the absolute convergence tolerance.

### Examples

```
### find the position of a value in a vector
f <- function(x){abs(vec[x]-1)}
discreteRoot(function(x){x},grid = seq(-20,10,1))
```

---

extractStrata            *Extract the information about the strata*

---

### Description

Extract the information about the strata stored in a Cox model

### Usage

```
extractStrata(xterms, xlevels = NULL, special)
```

### Arguments

xterms            ...

xlevels            ...

special            the special character indicating the strata variables

### Author(s)

Brice Ozenne broz@sund.ku.dk and Thomas A. Gerds tag@biostat.ku.dk

---

FGR                      *Formula wrapper for crr from cmprsk*

---

### Description

Formula interface for Fine-Gray regression competing risk models.

### Usage

```
FGR(formula, data, cause = 1, y = TRUE, ...)
```

**Arguments**

formula	A formula whose left hand side is a Hist object – see <a href="#">Hist</a> . The right hand side specifies (a linear combination of) the covariates. See examples below.
data	A data.frame in which all the variables of formula can be interpreted.
cause	The failure type of interest. Defaults to 1.
y	logical value: if TRUE, the response vector is returned in component response.
...	...

**Details**

Formula interface for the function `crr` from the `cmprsk` package.

The function `crr` allows to multiply some covariates by time before they enter the linear predictor. This can be achieved with the formula interface, however, the code becomes a little cumbersome. See the examples.

**Value**

See `crr`.

**Author(s)**

Thomas Alexander Gerds <tag@biostat.ku.dk>

**References**

Gerds, TA and Scheike, T and Andersen, PK (2011) Absolute risk regression for competing risks: interpretation, link functions and prediction Research report 11/7. Department of Biostatistics, University of Copenhagen

**See Also**

[riskRegression](#)

**Examples**

```
library(prodlm)
library(survival)
library(cmprsk)
library(lava)
d <- SimCompRisk(100)
f1 <- FGR(Hist(time,cause)~X1+X2,data=d)
print(f1)

## crr allows that some covariates are multiplied by
## a function of time (see argument tf of crr)
## by FGR uses the identity matrix
f2 <- FGR(Hist(time,cause)~cov2(X1)+X2,data=d)
print(f2)
```

```

## same thing, but more explicit:
f3 <- FGR(Hist(time,cause)~cov2(X1)+cov1(X2),data=d)
print(f3)

## both variables can enter cov2:
f4 <- FGR(Hist(time,cause)~cov2(X1)+cov2(X2),data=d)
print(f4)

## change the function of time
qFun <- function(x){x^2}
noFun <- function(x){x}
sqFun <- function(x){x^0.5}

## multiply X1 by time^2 and X2 by time:
f5 <- FGR(Hist(time,cause)~cov2(X1,tf=qFun)+cov2(X2),data=d)
print(f5)
print(f5$crrFit)
## same results as crr
with(d,crr(ftime=time,
           fstatus=cause,
           cov2=d[,c("X1","X2")],
           tf=function(time){cbind(qFun(time),time)}))

## still same result, but more explicit
f5a <- FGR(Hist(time,cause)~cov2(X1,tf=qFun)+cov2(X2,tf=noFun),data=d)
f5a$crrFit

## multiply X1 by time^2 and X2 by sqrt(time)
f5b <- FGR(Hist(time,cause)~cov2(X1,tf=qFun)+cov2(X2,tf=sqFun),data=d,cause=1)

## additional arguments for crr
f6<- FGR(Hist(time,cause)~X1+X2,data=d, cause=1,gtol=1e-5)
f6
f6a<- FGR(Hist(time,cause)~X1+X2,data=d, cause=1,gtol=0.1)
f6a

```

---

getSplitMethod

*Input for data splitting algorithms*


---

## Description

Parse hyperparameters for data splitting algorithm

## Usage

```
getSplitMethod(split.method, B, N, M, seed)
```

**Arguments**

<code>split.method</code>	A character string specifying the algorithm for data splitting: <ul style="list-style-type: none"> <li>• "loob" leave one out bootstrap</li> <li>• "bootcv" bootstrap cross validation</li> <li>• "cv5" 5-fold cross validation</li> <li>• "loocv" leave one out cross validation aka N-1 fold cross validation</li> <li>• "632plus" Efron's .632+ bootstrap</li> </ul>
<code>B</code>	Number of repetitions of bootstrap or k-fold cross-validation
<code>N</code>	Sample size
<code>M</code>	Subsample size. Default is N (no subsampling).
<code>seed</code>	Integer passed to <code>set.seed</code> . If not given or NA no seed is set.

**Value**

A list with the following elements:

- `split.methodName`: the print name of the algorithm
- `split.method`: the internal name of the algorithm
- `index`: the index for data splitting. For bootstrap splitting this is a matrix with B columns and M rows identifying the in-bag subjects. For k-fold cross-validation this is a matrix with B columns identifying the membership to the k groups.
- `k`: the k of k-fold cross-validation
- `N`: the sample size
- `M`: the subsample size

**Author(s)**

Thomas A. Gerds <tag@biostat.ku.dk>

**See Also**

Score

**Examples**

```
# 3-fold crossvalidation
getSplitMethod("cv3",B=4,N=37)

# bootstrap with replacement
getSplitMethod("loob",B=4,N=37)

# bootstrap without replacement
getSplitMethod("loob",B=4,N=37,M=20)
```

---

iidCox	<i>Extract i.i.d. decomposition from a Cox model</i>
--------	--

---

### Description

Compute the influence function for each observation used to estimate the model

### Usage

```
iidCox(object, newdata = NULL, baseline.iid = TRUE, tau.hazard = NULL,
       store.iid = "full", keep.times = TRUE)
```

### Arguments

object	object The fitted Cox regression model object either obtained with <code>coxph</code> (survival package) or <code>cph</code> (rms package).
newdata	Optional new data at which to do i.i.d. decomposition
baseline.iid	Should the influence function for the baseline hazard be computed.
tau.hazard	the vector of times at which the i.i.d decomposition of the baseline hazard will be computed
store.iid	the method used to compute the influence function and the standard error. Can be "full", "approx" or "minimal". See the details section.
keep.times	Logical. If TRUE add the evaluation times to the output.

### Details

This function implements the first three formula (no number,10,11) of the subsection "Empirical estimates" in (Ozenne et al., 2017).

If there is no event in a strata, the influence function for the baseline hazard is set to 0.

`store.iid` equal to "full" exports the influence function for the coefficients and the baseline hazard at each event time. `store.iid` equal to "approx" does the same except that the terms that do not contribute to the variance are not ignored (i.e. set to 0) `store.iid` equal to "minimal" exports the influence function for the coefficients. For the baseline hazard it only computes the quantities necessary to compute the influence function in order to save memory.

### Value

A list containing:

- `IFbetaInfluence` function for the regression coefficient.
- `IFhazardTime` differential of the influence function of the hazard.
- `IFcumhazardInfluence` function of the cumulative hazard.
- `calcIFhazardElements` used to compute the influence function at a given time.
- `timeTimes` at which the influence function has been evaluated.

- `etime1.minTime` of first event (i.e. jump) in each strata.
- `etime.maxLast` observation time (i.e. jump or censoring) in each strata.
- `indexObsIndex` of the observation in the original dataset.

## References

Brice Ozenne, Anne Lyngholm Sorensen, Thomas Scheike, Christian Torp-Pedersen and Thomas Alexander Gerds. *riskRegression: Predicting the Risk of an Event using Cox Regression Models*. The R Journal (2017) 9:2, pages 440-460.

## Examples

```
library(survival)
library(data.table)
set.seed(10)
d <- sampleData(100, outcome = "survival")[,.(eventtime,event,X1,X6)]
setkey(d, eventtime)

m.cox <- coxph(Surv(eventtime, event) ~ X1+X6, data = d, y = TRUE, x = TRUE)
system.time(IF.cox <- iidCox(m.cox))
system.time(IF.cox_approx <- iidCox(m.cox, store.iid = "approx"))

IF.cox.all <- iidCox(m.cox, tau.hazard = sort(unique(c(7,d$eventtime))))
IF.cox.beta <- iidCox(m.cox, baseline.iid = FALSE)
```

---

influenceCoxTest	<i>Influence test [Experimental!!!]</i>
------------------	---

---

## Description

Compare two estimates using their influence function

## Usage

```
influenceCoxTest(object, ...)

## S3 method for class 'list'
influenceCoxTest(object, newdata, times, type, cause, ...)

## Default S3 method:
influenceCoxTest(object, object2, tanh.transform = FALSE,
  conf.level = 0.95, nsim.band = 10000, ...)
```



**Arguments**

object	either a list of models or an object of class predictCox or predictCSC.
...	additional arguments to be passed to lower level functions.
newdata	A data.frame or data.table containing the values of the predictor variables defining subject specific predictions.
times	Time points at which to evaluate the predictions.
type	the type of predicted value.
cause	Identifies the cause of interest among the competing events.
object2	same as predict1 but for another model.
tanh.transform	Should the comparison be made on the arc-tangent scale or on the original scale ?
conf.level	Level of confidence.
nsim.band	the number of simulations used to compute the quantiles for the confidence bands.

**Examples**

```

library(lava)
library(survival)
n <- 100

#### Under H1
set.seed(1)
newdata <- data.frame(X1=1)

## simulate non proportional hazard using lava
m <- lvm()
regression(m) <- y ~ 1
regression(m) <- s ~ exp(-2*X1)
distribution(m,~X1) <- binomial.lvm()
distribution(m,~cens) <- coxWeibull.lvm(scale=1)
distribution(m,~y) <- coxWeibull.lvm(scale=1,shape=~s)
eventTime(m) <- eventtime ~ min(y=1,cens=0)
d <- as.data.table(sim(m,n))
setkey(d, eventtime)

## fit cox models
m.cox <- coxph(Surv(eventtime, status) ~ X1,
               data = d, y = TRUE, x = TRUE)

mStrata.cox <- coxph(Surv(eventtime, status) ~ strata(X1),
                    data = d, y = TRUE, x = TRUE)

## compare models
# one time point
influenceCoxTest(list(m.cox, mStrata.cox), tanh.transform = FALSE,
                  type = "survival", newdata = newdata, times = 0.5)
influenceCoxTest(list(m.cox, mStrata.cox), tanh.transform = TRUE,

```

```

        type = "survival", newdata = newdata, times = 0.5)

# several timepoints
influenceCoxTest(list(m.cox, mStrata.cox), tanh.transform = TRUE,
                 type = "survival", newdata = newdata, times = seq(0.1,1,by=0.1))
influenceCoxTest(list(m.cox, mStrata.cox), tanh.transform = FALSE,
                 type = "survival", newdata = newdata, times = seq(0.1,1,by=0.1))

#### Under H0 (Cox) ####
set.seed(1)
## simulate proportional hazard using lava
m <- lvm()
regression(m) <- y ~ 1
distribution(m,~X1) <- binomial.lvm()
distribution(m,~cens) <- coxWeibull.lvm()
distribution(m,~y) <- coxWeibull.lvm()
eventTime(m) <- eventtime ~ min(y=1,cens=0)
d <- as.data.table(sim(m,n))
setkey(d, eventtime)

## fit cox models
Utime <- sort(unique(d$eventtime))
m.cox <- coxph(Surv(eventtime, status) ~ X1,
              data = d, y = TRUE, x = TRUE)

mStrata.cox <- coxph(Surv(eventtime, status) ~ strata(X1),
                   data = d, y = TRUE, x = TRUE)

p.cox <- predictCox(m.cox, newdata = newdata, time = Utime, type = "survival")
p.coxStrata <- predictCox(mStrata.cox, newdata = newdata, time = Utime, type = "survival")
autoplot(p.cox)
autoplot(p.coxStrata)

## compare models
influenceCoxTest(list(m.cox, mStrata.cox), tanh.transform = TRUE,
                 type = "survival", newdata = newdata, times = Utime[1:(n-10)])
influenceCoxTest(list(m.cox, mStrata.cox), tanh.transform = FALSE,
                 type = "survival", newdata = newdata, times = Utime[1:(n-10)])

#### Under H0 (CSC) ####
set.seed(1)
ff <- ~ f(X1,2) + f(X2,-0.033)
ff <- update(ff, ~ .+ f(X3,0) + f(X4,0) + f(X5,0))
ff <- update(ff, ~ .+ f(X6,0) + f(X7,0) + f(X8,0) + f(X9,0))
d <- sampleData(n, outcome = "competing.risk", formula = ff)
d[,X1:=as.numeric(as.character(X1))]
d[,X2:=as.numeric(as.character(X2))]
d[,X3:=as.numeric(as.character(X3))]
d[,X4:=as.numeric(as.character(X4))]
d[,X5:=as.numeric(as.character(X5))]
setkey(d, time)

Utime <- sort(unique(d$time))

```

```
## fit cox models
m.CSC <- CSC(Hist(time, event) ~ X1 + X2, data = d)
mStrata.CSC <- CSC(Hist(time, event) ~ strata(X1) + X2 + X3, data = d)

## compare models
influenceCoxTest(list(m.CSC, mStrata.CSC), tanh.transform = TRUE,
                  cause = 1, newdata = data.frame(X1=1,X2=1,X3=1), times = Utime[1:20])
influenceCoxTest(list(m.CSC, mStrata.CSC), tanh.transform = FALSE,
                  cause = 1, newdata = data.frame(X1=1,X2=1,X3=1), times = Utime[1:20])
```

ipcw

*Estimation of censoring probabilities***Description**

This function is used internally to obtain inverse of the probability of censoring weights.

**Usage**

```
ipcw(formula, data, method, args, times, subject.times, lag = 1, what,
      keep = NULL)
```

**Arguments**

formula	A survival formula like, $\text{Surv}(\text{time}, \text{status}) \sim 1$ , where as usual $\text{status}=0$ means censored. The status variable is internally reversed for estimation of censoring rather than survival probabilities. Some of the available models (see argument model) will use predictors on the right hand side of the formula.
data	The data used for fitting the censoring model
method	Censoring model used for estimation of the (conditional) censoring distribution.
args	A list of arguments which is passed to method
times	For what="IPCW.times" a vector of times at which to compute the probabilities of not being censored.
subject.times	For what="IPCW.subject.times" a vector of individual times at which the probabilities of not being censored are computed.
lag	If equal to 1 then obtain $G(T_i   X_i)$ , if equal to 0 estimate the conditional censoring distribution at the subject.times, i.e. $(G(T_i   X_i))$ .
what	Decide about what to do: If equal to "IPCW.times" then weights are estimated at given times. If equal to "IPCW.subject.times" then weights are estimated at individual subject.times. If missing then produce both.
keep	Which elements to add to the output. Any subset of the vector $c(\text{"times"}, \text{"fit"}, \text{"call"})$ .

## Details

Inverse of the probability of censoring weights (IPCW) usually refer to the probabilities of not being censored at certain time points. These probabilities are also the values of the conditional survival function of the censoring time given covariates. The function `ipcw` estimates the conditional survival function of the censoring times and derives the weights.

**IMPORTANT:** the data set should be ordered, `order(time, -status)` in order to get the values `IPCW.subject.times` in the right order for some choices of method.

## Value

A list with elements depending on argument `keep`.

<code>times</code>	The times at which weights are estimated
<code>IPCW.times</code>	Estimated weights at times
<code>IPCW.subject.times</code>	Estimated weights at individual time values <code>subject.times</code>
<code>fit</code>	The fitted censoring model
<code>method</code>	The method for modelling the censoring distribution
<code>call</code>	The call

## Author(s)

Thomas A. Gerds <tag@biostat.ku.dk>

## Examples

```
library(prodlim)
library(rms)
dat=SimSurv(30)

dat <- dat[order(dat$time),]

# using the marginal Kaplan-Meier for the censoring times

WKM=ipcw(Hist(time,status)~X2,
  data=dat,
  method="marginal",
  times=sort(unique(dat$time)),
  subject.times=dat$time,keep=c("fit"))
plot(WKM$fit)
WKM$fit

# using the Cox model for the censoring times given X2
library(survival)
WCox=ipcw(Hist(time=time,event=status)~X2,
  data=dat,
  method="cox",
  times=sort(unique(dat$time)),
```

```

    subject.times=dat$time,keep=c("fit"))
WCox$fit

plot(WKM$fit)
lines(sort(unique(dat$time)),
      1-WCox$IPCW.times[1,],
      type="l",
      col=2,
      lty=3,
      lwd=3)
lines(sort(unique(dat$time)),
      1-WCox$IPCW.times[5,],
      type="l",
      col=3,
      lty=3,
      lwd=3)

# using the stratified Kaplan-Meier
# for the censoring times given X2

WKM2=ipcw(Hist(time,status)~X2,
          data=dat,
          method="nonpar",
          times=sort(unique(dat$time)),
          subject.times=dat$time,keep=c("fit"))
plot(WKM2$fit,add=FALSE)

```

---

 Melanoma

*Malignant melanoma data*


---

### Description

In the period 1962-77, 205 patients with malignant melanoma (cancer of the skin) had a radical operation performed at Odense University Hospital, Denmark. All patients were followed until the end of 1977 by which time 134 were still alive while 71 had died (of out whom 57 had died from cancer and 14 from other causes).

### Format

A data frame with 205 observations on the following 12 variables.

**time** time in days from operation

**status** a numeric with values 0=censored 1=death.malignant.melanoma 2=death.other.causes

**event** a factor with levels censored death.malignant.melanoma death.other.causes

**invasion** a factor with levels level.0, level.1, level.2

**ici** inflammatory cell infiltration (IFI): 0, 1, 2 or 3

**epicel** a factor with levels not present present  
**ulcer** a factor with levels not present present  
**thick** tumour thickness (in 1/100 mm)  
**sex** a factor with levels Female Male  
**age** age at operation (years)  
**logthick** tumour thickness on log-scale

### Details

The object of the study was to assess the effect of risk factors on survival. Among such risk factors were the sex and age of the patients and the histological variables tumor thickness and ulceration (absent vs. present).

### Source

<http://192.38.117.59/~linearpredictors/?page=datasets&dataset=Melanoma>

### References

Regression with linear predictors (2010)  
Andersen, P.K. and Skovgaard, L.T.  
Springer Verlag

### Examples

```
data(Melanoma)
```

---

model.matrix.phreg      *Extract design matrix for phreg objects*

---

### Description

Extract design matrix for phreg objects

### Usage

```
## S3 method for class 'phreg'  
model.matrix(object, data)
```

### Arguments

object            a phreg object.  
data              a dataset.

**Details**

mainly a copy paste of the beginning of the phreg function.

---

Paquid

*Paquid sample*

---

**Description**

PAQUID is a prospective cohort study initiated in 1988 in South Western France to explore functional and cerebral ageing. This sample includes n=2561 subjects. Data contains a time-to-event, a type of event and two cognitive scores measured at baseline.

**Format**

A data frame with 2561 observations on the following 4 variables.

`time` the time-to-event (in years).

`status` the type of event  $0$  = censored,  $1$  = dementia onset and  $2$  = death without dementia.

`DSST` score at the Digit Symbol Substitution Score Test. This test explores attention and psychomotor speed.

`MMSE` score at the Mini Mental State Examination. This test is often used as an index of global cognitive performance.

**Source**

The data have been first made publicly available via the package `timeROC`.

**References**

Dartigues, J., Gagnon, M., Barberger-Gateau, P., Letenneur, L., Commenges, D., Sauvel, C., Michel, P., and Salamon, R. (1992). The paquid epidemiological program on brain ageing. *Neuroepidemiology*, 11(1):14–18.

Blanche, P., Dartigues, J. F., & Jacqmin-Gadda, H. (2013). Estimating and comparing time-dependent areas under receiver operating characteristic curves for censored event times with competing risks. *Statistics in Medicine*, 32(30), 5381-5397.

**Examples**

```
data(Paquid)
```

---

 penalizedS3

*S3-wrapper for S4 function penalized*


---

**Description**

S3-wrapper for S4 function penalized

**Usage**

```
penalizedS3(formula, data, type = "ridge", ...)
```

**Arguments**

formula	Communicated outcome and explanatory variables. See examples.
data	Data set in which formula is to be interpreted
type	String specifying the type of penalization. Should match one of the following values: "ridge", "lasso", "elastic.net".
...	Arguments passed to penalized

**Examples**

```
## Not run:
## too slow
library(penalized)
set.seed(8)
d <- sampleData(200,outcome="binary")
newd <- sampleData(80,outcome="binary")
fitridge <- penalizedS3(Y~X1+X2+pen(7:8), data=d, type="ridge",
  standardize=TRUE, model="logistic",trace=FALSE)
fitlasso <- penalizedS3(Y~X1+X2+pen(7:8), data=d, type="lasso",
  standardize=TRUE, model="logistic",trace=FALSE)
# fitnet <- penalizedS3(Y~X1+X2+pen(7:8), data=d, type="elastic.net",
# standardize=TRUE, model="logistic",trace=FALSE)
predictRisk(fitridge,newdata=newd)
predictRisk(fitlasso,newdata=newd)
# predictRisk(fitnet,newdata=newd)
Score(list(fitridge),data=newd,formula=Y~1)
Score(list(fitridge),data=newd,formula=Y~1,split.method="bootcv",B=2)

## End(Not run)
## Not run: data(nki70) ## S4 fit
pen <- penalized(Surv(time, event), penalized = nki70[,8:77],
  unpenalized = ~ER+Age+Diam+N+Grade, data = nki70,
  lambda1 = 1)
penS3 <- penalizedS3(Surv(time,event)~ER+Age+Diam+pen(8:77)+N+Grade,
  data=nki70, lambda1=1)
## or
penS3 <- penalizedS3(Surv(time,event)~ER+pen(TSPYL5,Contig63649_RC)+pen(10:77)+N+Grade,
```



```

                                data=nki70, lambda1=1)
## also this works
penS3 <- penalizedS3(Surv(time,event)~ER+Age+pen(8:33)+Diam+pen(34:77)+N+Grade,
                    data=nki70, lambda1=1)

## End(Not run)

```

---

plot.riskRegression    *Plotting predicted risk*

---

### Description

Show predicted risk obtained by a risk prediction model as a function of time.

### Usage

```

## S3 method for class 'riskRegression'
plot(x,
     cause,
     newdata,
     xlab,
     ylab,
     xlim,
     ylim,
     lwd,
     col,
     lty,
     axes=TRUE,
     percent=TRUE,
     legend=TRUE,
     add=FALSE,
     ...)

```

### Arguments

x	Fitted object obtained with one of ARR, LRR, riskRegression.
cause	For CauseSpecificCox models the cause of interest.
newdata	A data frame containing predictor variable combinations for which to compute predicted risk.
xlab	See plot
ylab	See plot
xlim	See plot
ylim	See plot
lwd	A vector of line thicknesses for the regression coefficients.
col	A vector of colors for the regression coefficients.

lty	A vector of line types for the regression coefficients.
axes	Logical. If FALSE then do not draw axes.
percent	If true the y-axis is labeled in percent.
legend	If true draw a legend.
add	Logical. If TRUE then add lines to an existing plot.
...	Used for transclusion of smart arguments for plot, lines, axis and background. See function <a href="#">SmartControl</a> from prodlim.

**Author(s)**

Thomas Alexander Gerds <tag@biostat.ku.dk>

**Examples**

```
library(survival)
data(Melanoma)
fit.arr <- ARR(Hist(time,status)~invasion+age+strata(sex),data=Melanoma,cause=1)
plot(fit.arr,xlim=c(500,3000))
```

---

plotAUC

*ggplot AUC curve*

---

**Description**

ggplot AUC curves

**Usage**

```
plotAUC(x, models, type = "score", lwd = 2, xlim, ylim, axes = TRUE,
        conf.int = FALSE, ...)
```

**Arguments**

x	Object obtained with <code>Score.list</code>
models	Choice of models to plot
type	Character. Either "score" to show AUC or "contrasts" to show differences between AUC.
lwd	Line width
xlim	Limits for x-axis
ylim	Limits for y-axis
axes	Logical. If TRUE draw axes.
conf.int	Logical. If TRUE draw confidence shadows.
...	Not yet used

**Examples**

```

library(survival)
d=sampleData(100,outcome="survival")
nd=sampleData(100,outcome="survival")
f1=coxph(Surv(time,event)~X1+X6+X8,data=d,x=TRUE,y=TRUE)
f2=coxph(Surv(time,event)~X2+X5+X9,data=d,x=TRUE,y=TRUE)
xx=Score(list(f1,f2), formula=Surv(time,event)~1,
data=nd, metrics="auc", null.model=FALSE, times=seq(3:10))
aucgraph <- plotAUC(xx)
plotAUC(xx,conf.int=TRUE)+ggtitle("AUC")+theme_classic()
plotAUC(xx,type="contrasts")
a=plotAUC(xx,type="contrasts",conf.int=TRUE)
a+theme_bw()

```

---

plotBrier

*Plot Brier curve*


---

**Description**

Plot Brier score curves

**Usage**

```

plotBrier(x, models, which = "score", xlim, ylim, xlab, ylab, col, lwd,
lty = 1, cex = 1, pch = 1, type = "l", axes = 1L, percent = 1L,
conf.int = 0L, legend = 1L, ...)

```

**Arguments**

x	Object obtained with Score
models	Choice of models to plot
which	Character. Either "score" to show AUC or "contrasts" to show differences between AUC.
xlim	Limits for x-axis
ylim	Limits for y-axis
xlab	Label for x-axis
ylab	Label for y-axis
col	line color
lwd	line width
lty	line style
cex	point size
pch	point style

type	line type
axes	Logical. If TRUE draw axes.
percent	Logical. If TRUE scale y-axis in percent.
conf.int	Logical. If TRUE draw confidence shadows.
legend	Logical. If TRUE draw legend.
...	Used for additional control of the subroutines: plot, axis, lines, legend. See <a href="#">SmartControl</a> .

### Examples

```
# survival
library(survival)
ds1=sampleData(40,outcome="survival")
ds2=sampleData(40,outcome="survival")
f1 <- coxph(Surv(time,event)~X1+X3+X5+X7+X9,data=ds1,x=TRUE)
f2 <- coxph(Surv(time,event)~X2+X4+X6+X8+X10,data=ds1,x=TRUE)
xscore <- Score(list(f1,f2),formula=Hist(time,event)~1,data=ds2,times=0:12,metrics="brier")
plotBrier(xscore)
```

---

plotCalibration	<i>Plot Calibration curve</i>
-----------------	-------------------------------

---

### Description

Plot Calibration curve

### Usage

```
plotCalibration(x, models, times, method = "nne", round = TRUE,
  bandwidth = NULL, q = 10, bars = FALSE, hanging = FALSE,
  names = "quantiles", pseudo, rug, show.frequencies = FALSE, plot = TRUE,
  add = FALSE, diag = !add, legend = !add, auc.in.legend = TRUE,
  brier.in.legend = TRUE, axes = !add, xlim = c(0, 1), ylim = c(0, 1),
  xlab = ifelse(bars, "Risk groups", "Predicted risk"),
  ylab = "Observed frequency", col, lwd, lty, pch, type, cause = 1,
  percent = TRUE, na.action = na.fail, cex = 1, ...)
```

### Arguments

x	Object obtained with function Score
models	Choice of models to plot
times	Time point specifying the prediction horizon.
method	The method for estimating the calibration curve(s): "nne": The expected event status is obtained in the nearest neighborhood around the predicted event probabilities. "quantile": The expected event status is obtained in groups defined by quantiles of the predicted event probabilities.

round	If TRUE predicted probabilities are rounded to two digits before smoothing. This may have a considerable effect on computing efficiency in large data sets.
bandwidth	The bandwidth for method="nne"
q	The number of quantiles for method="quantile" and bars=TRUE.
bars	If TRUE, use barplots to show calibration.
hanging	Barplots only. If TRUE, hang bars corresponding to observed frequencies at the value of the corresponding prediction.
names	Barplots only. Names argument passed to names.arg of barplot.
pseudo	If TRUE show pseudo values (only for right censored data).
rug	If TRUE show rug plot at the predictions
show.frequencies	Barplots only. If TRUE, show frequencies above the bars.
plot	If FALSE, do not plot the results, just return a plottable object.
add	If TRUE the line(s) are added to an existing plot.
diag	If FALSE no diagonal line is drawn.
legend	Logical. If TRUE draw legend.
auc.in.legend	Logical. If TRUE add AUC to legend.
brier.in.legend	Logical. If TRUE add Brier score to legend.
axes	If FALSE no axes are drawn.
xlim	Limits of x-axis.
ylim	Limits of y-axis.
xlab	Label for y-axis.
ylab	Label for x-axis.
col	Vector with colors, one for each element of object. Passed to <a href="#">lines</a> .
lwd	Vector with line widths, one for each element of object. Passed to <a href="#">lines</a> .
lty	lwd Vector with line style, one for each element of object. Passed to <a href="#">lines</a> .
pch	Passed to <a href="#">lines</a> .
type	Passed to <a href="#">lines</a> .
cause	For competing risks models, the cause of failure or event of interest
percent	If TRUE axes labels are multiplied by 100 and thus interpretable on a percent scale.
na.action	what to do with NA values. Passed to <a href="#">model.frame</a>
cex	Default cex used for legend and labels.
...	Used to control the subroutines: plot, axis, lines, barplot, legend, addtable2plot, points (pseudo values), rug. See <a href="#">SmartControl</a> .

**Examples**

```

# binary
db=sampleData(100,outcome="binary")
fb1=glm(Y~X1+X5+X7,data=db,family="binomial")
fb2=glm(Y~X1+X3+X6+X7,data=db,family="binomial")
xb=Score(list(model1=fb1,model2=fb2),Y~1,data=db,
          plots="cal")
plotCalibration(xb)

# survival
library(survival)
ds=sampleData(100,outcome="survival")
fs1=coxph(Surv(time,event)~X1+X5+X7,data=ds,x=1)
fs2=coxph(Surv(time,event)~X1+X3+X6+X7,data=ds,x=1)
xs=Score(list(Cox1=fs1,Cox2=fs2),Surv(time,event)~1,data=ds,
             plots="cal",metrics=NULL)
plotCalibration(xs)

# competing risks
data(Melanoma)
f1 <- CSC(Hist(time,status)~age+sex+epicel+ulcer,data=Melanoma)
f2 <- CSC(Hist(time,status)~age+sex+logthick+epicel+ulcer,data=Melanoma)
x <- Score(list(model1=f1,model2=f2),Hist(time,status)~1,data=Melanoma,
            cause= 2,times=5*365.25,plots="cal")
plotCalibration(x)

```

---

plotEffects

*Plotting time-varying effects from a risk regression model.*


---

**Description**

Plot time-varying effects from a risk regression model.

**Usage**

```

plotEffects(x, formula, level, ref.line = TRUE, conf.int = 0.95, xlim, ylim,
           xlab = "Time", ylab = "Cumulative coefficient", col, lty, lwd,
           add = FALSE, legend, axes = TRUE, ...)

```

**Arguments**

x	Fitted object obtained with one of ARR, LRR, riskRegression.
formula	A formula to specify the variable(s) whose regression coefficients should be plotted.
level	For categorical variables the level (group) whose contrast to the reference level (group) should be plotted.
ref.line	Logical. If TRUE then add a horizontal line at zero.

conf.int	Logical. If TRUE then add confidence limits. Can be controlled using smart arguments. See examples
xlim	See plot
ylim	See plot
xlab	See plot
ylab	See plot
col	A vector of colors for the regression coefficients.
lty	A vector of line types for the regression coefficients.
lwd	A vector of line thicknesses for the regression coefficients.
add	Logical. If TRUE then add lines to an existing plot.
legend	Logical. If TRUE then add a legend. Can be controlled using smart arguments. See examples.
axes	Logical. If FALSE then do not draw axes.
...	Used for transclusion of smart arguments for plot, axis. See function <a href="#">SmartControl</a> from prodlim.

**Author(s)**

Thomas H. Scheike <ts@biostat.ku.dk>

Thomas A. Gerds <>tag@biostat.ku.dk>

**Examples**

```
library(survival)
library(prodlim)
data(Melanoma)

fit.tarr <- ARR(Hist(time,status)~strata(sex),
               data=Melanoma,
               cause=1)
plotEffects(fit.tarr)

fit.tarr <- ARR(Hist(time,status)~strata(sex)+strata(invasion),
               data=Melanoma,
               cause=1,
               times=seq(800,3000,20))
plotEffects(fit.tarr,formula=~sex)
plotEffects(fit.tarr,formula=~invasion)
plotEffects(fit.tarr,
            formula=~invasion,
            level="invasionlevel.1")

## legend arguments are transcluded:
plotEffects(fit.tarr,
            formula=~invasion,
            legend.bty="b",
```

```

        legend.cex=1)

## and other smart arguments too:
plotEffects(fit.tarr,
            formula=~invasion,
            legend.bty="b",
            axis2.las=2,
            legend.cex=1)

```

---

plotRisk

*plot predicted risks*


---

### Description

plot predicted risks

### Usage

```
plotRisk(x, models, times, xlim = c(0, 1), ylim = c(0, 1), xlab, ylab, col,
        pch = 1, cex = 1, ...)
```

### Arguments

x	Object obtained with function Score
models	Choice of two models to plot. The predicted risks of the first (second) are shown along the x-axis (y-axis).
times	Time point specifying the prediction horizon.
xlim	x-axis limits
ylim	y-axis limits
xlab	x-axis labels
ylab	y-axis labels
col	colour
pch	point type
cex	point size
...	Used to control the subroutines: plot, axis, lines, barplot, legend. See <a href="#">SmartControl</a> .

### Details

Two rival prediction models are applied to the same data.

### Value

a nice graph



**Author(s)**

Thomas A. Gerds <tag@biostat.ku.dk>

**Examples**

```
## uncensored
learndat = sampleData(40,outcome="binary")
testdat = sampleData(40,outcome="binary")
lr1 = glm(Y~X1+X2+X7+X9,data=learndat,family="binomial")
lr2 = glm(Y~X3+X5+X6,data=learndat,family="binomial")
xb=Score(list("LR(X1+X2+X7+X9)"=lr1,"LR(X3+X5+X6)"=lr2),formula=Y~1,
          data=testdat,summary="risks",null.model=0L)
plotRisk(xb)
## survival
library(survival)
learndat = sampleData(40,outcome="survival")
testdat = sampleData(40,outcome="survival")
cox1 = coxph(Surv(time,event)~X1+X2+X7+X9,data=learndat,x=TRUE)
cox2 = coxph(Surv(time,event)~X3+X5+X6,data=learndat,x=TRUE)
xs=Score(list("Cox(X1+X2+X7+X9)"=cox1,"Cox(X3+X5+X6)"=cox2),formula=Surv(time,event)~1,
          data=testdat,summary="risks",null.model=0L)
plotRisk(xs,times=5)
```

---

plotROC

*Plot ROC curves*

---

**Description**

Plot ROC curve

**Usage**

```
plotROC(x, models, times, xlab = "1-Specificity", ylab = "Sensitivity", col,
        lwd, lty = 1, cex = 1, pch = 1, legend = TRUE, auc.in.legend = TRUE,
        brier.in.legend = FALSE, add = FALSE, ...)
```

**Arguments**

x	Object obtained with function Score
models	Choice of models to plot
times	Time point(s) specifying the prediction horizon
xlab	Label for x-axis
ylab	Label for y-axis
col	line color
lwd	line width

lty	line style
cex	point size
pch	point style
legend	logical. If 1L draw a legend with the values of AUC.
auc.in.legend	Logical. If TRUE add AUC to legend.
brier.in.legend	Logical. If TRUE add Brier score to legend.
add	logical. If 1L add lines to an existing plot.
...	Used for additional control of the subroutines: plot, axis, lines, legend, addtable2plot. See <a href="#">SmartControl</a> .

### Examples

```
## binary
set.seed(18)
library(randomForest)
bd1 <- sampleData(40,outcome="binary")
bdt <- sampleData(58,outcome="binary")
bd1[,y:=factor(Y)]
bdt[,y:=factor(Y)]
fb1 <- glm(y~X1+X2+X3+X4+X5+X6+X7+X8+X9+X10,data=bd1,family="binomial")
fb2 <- randomForest(y~X1+X2+X3+X4+X5+X6+X7+X8+X9+X10,data=bd1)
xb <- Score(list("glm"=fb1,"rf"=fb2),y~1,data=bdt,
            plots="roc",metrics=c("auc","brier"))
plotROC(xb,brier.in.legend=1L)

# with cross-validation
## Not run:
xb3 <- Score(list("glm"=fb1,"rf"=fb2),y~1,data=bd1,
            plots="roc",B=3,split.method="bootcv",
            metrics=c("auc"))

## End(Not run)
## survival
set.seed(18)
library(survival)
sd1 <- sampleData(40,outcome="survival")
sdt <- sampleData(58,outcome="survival")
fs1 <- coxph(Surv(time,event)~X3+X5+X6+X7+X8+X10,data=sd1,x=TRUE)
fs2 <- coxph(Surv(time,event)~X1+X2+X9,data=sd1,x=TRUE)
xs <- Score(list(model1=fs1,model2=fs2),Hist(time,event)~1,data=sdt,
            times=5,plots="roc",metrics="auc")
plotROC(xs)
## competing risks
data(Melanoma)
f1 <- CSC(Hist(time,status)~age+sex+epicel+ulcer,data=Melanoma)
f2 <- CSC(Hist(time,status)~age+sex+logthick+epicel+ulcer,data=Melanoma)
x <- Score(list(model1=f1,model2=f2),Hist(time,status)~1,data=Melanoma,
            cause=1,times=5*365.25,plots="roc",metrics="auc")
plotROC(x)
```

---

 predict.CauseSpecificCox

*Predicting absolute risk from cause-specific Cox models*


---

### Description

Apply formula to combine two or more Cox models into absolute risk (cumulative incidence function)

### Usage

```
## S3 method for class 'CauseSpecificCox'
predict(object, newdata, times, cause,
        landmark = NA, keep.times = 1L, keep.newdata = 1L, keep.strata = 1L,
        se = FALSE, band = FALSE, iid = FALSE, average.iid = FALSE,
        nsim.band = 10000, log.transform = TRUE, product.limit = TRUE,
        conf.level = 0.95, store.iid = "full", ...)
```

### Arguments

object	The fitted cause specific Cox model
newdata	A data frame containing the values of the variables in the right hand side of 'coxph' for each subject.
times	Vector of times at which to return the estimated absolute risk.
cause	Identifies the cause of interest among the competing events.
landmark	the starting time for the computation of the cumulative risk
keep.times	Logical. If TRUE add the evaluation times to the output.
keep.newdata	Logical. If TRUE add the value of the covariates used to make the prediction in the output list.
keep.strata	Logical. If TRUE add the value of the strata used to make the prediction in the output list.
se	Logical. If TRUE add the standard errors to the output.
band	Logical. If TRUE add the confidence band to the output.
iid	Logical. If TRUE add the influence function to the output.
average.iid	Logical. If TRUE add the average of the influence function over newdata to the output.
nsim.band	the number of simulations used to compute the quantiles for the confidence bands.
log.transform	Should the confidence intervals/bands be computed on the log(-log) scale and be backtransformed. Otherwise they are computed on the original scale and truncated (if necessary).
product.limit	Logical. If true the survival is computed using the product limit estimator. Otherwise the exponential approximation is used (i.e. $\exp(-\text{cumulative hazard})$ ).

<code>conf.level</code>	Level of confidence.
<code>store.iid</code>	Implementation used to estimate the influence function and the standard error. Can be "full" or "minimal".
<code>...</code>	not used

### Details

This function computes the absolute risk as given by formula 2 of (Ozenne et al., 2017). Confidence intervals and confidence bands can be computed using a first order von Mises expansion. See the section "Construction of the confidence intervals" in (Ozenne et al., 2017).

When setting `log.transform` to TRUE, the standard error that is returned is before back-transformation to the original scale.

A detailed explanation about the meaning of the argument `store.iid` can be found in (Ozenne et al., 2017) Appendix B "Saving the influence functions".

Note: for Cox regression models with time varying covariates it does not make sense to use this function, because the predicted risk has to be a measurable function of the data available at the time origin.

### Value

A list containing:

- `absRisk`: (data table) the predictions for each subject (in rows) and each time (in columns).
- `absRisk.se`: (data table) the standard errors of the predictions.
- `(absRisk.iid)`: (array) the value of the influence of each subject used to fit the object (dim 3) for each subject in `newdata` (dim 1) and each time (dim 2).
- `(absRisk.average.iid)`: (matrix) the average value of the influence over the subjects in `newdata`, for each subject used to fit the object (dim 1) and each time (dim 2).
- `times`: (vector) the evaluation times.

### Author(s)

Brice Ozenne [broz@sund.ku.dk](mailto:broz@sund.ku.dk), Thomas A. Gerds [tag@biostat.ku.dk](mailto>tag@biostat.ku.dk)

### References

Brice Ozenne, Anne Lyngholm Sorensen, Thomas Scheike, Christian Torp-Pedersen and Thomas Alexander Gerds. `riskRegression`: Predicting the Risk of an Event using Cox Regression Models. *The R Journal* (2017) 9:2, pages 440-460.

### Examples

```
library(survival)

## generate data
set.seed(5)
d <- sampleData(80, outcome="comp") ## training dataset
nd <- sampleData(4, outcome="comp") ## validation dataset
```

```

d$time <- round(d$time,1) ## create tied events
ttt <- sort(sample(x = unique(d$time), size = 10))

## estimate a CSC model based on the coxph function
CSC.fit <- CSC(Hist(time,event)~ X3+X8, data=d, method = "breslow")

## compute the absolute risk of cause 1, in the validation dataset
## at time 1:10
CSC.risk <- predict(CSC.fit, newdata=nd, times=1:10, cause=1)
CSC.risk

## add the standard error/confidence intervals
## (computed on the log log scale and backtransformed)
CSC.riskSE <- predict(CSC.fit,newdata=nd,times=1:10,cause=1,se=TRUE,
  log.transform = TRUE)
as.data.table(CSC.riskSE)[1:5]
exp(-exp(
  log(-log(CSC.riskSE$absRisk)) - 1.96 * CSC.riskSE$absRisk.se
))

## extract the iid for the absolute risk
CSC.iid <- predict(CSC.fit, newdata = d, se = TRUE,
  cause = 1, times = ttt[1], iid = TRUE,
  log.transform = FALSE)
rowMeans(CSC.iid$absRisk.iid[,1,]) ## the iid decomposition has 0 expectation
sqrt(rowSums(CSC.iid$absRisk.iid[,1,]^2))[1:5]
as.data.table(CSC.iid)[1:5]

## same but the iid decomposition is averaged over the patients
CSC.aviid <- predict(CSC.fit, newdata = d,
  cause = 1, times = ttt[1],
  average.iid = TRUE,
  log.transform = FALSE)
CSC.aviid$absRisk.average.iid[1:5,]
colMeans(CSC.iid$absRisk.iid[,1,1:5])

## compute absolute risks with CI for cause 2
## (without displaying the value of the covariates)
predict(CSC.fit,newdata=nd,times=1:10,cause=2,se=TRUE,
  log.transform = TRUE, keep.newdata = FALSE)

## other example
library(survival)
CSC.fit.s <- CSC(list(Hist(time,event)~ strata(X1)+X2+X9,
  Hist(time,event)~ X2+strata(X4)+X8+X7),data=d, method = "breslow")
predict(CSC.fit.s,cause=1,times=ttt,se=1L) ## note: absRisk>1 due to small number of observations

## using the cph function instead of coxph
CSC.cph <- CSC(Hist(time,event)~ X1+X2,data=d, method = "breslow", fitter = "cph")#
predict(CSC.cph, newdata = d, cause = 2, times = ttt)

## landmark analysis
T0 <- 1

```

```
predCSC_afterT0 <- predict(CSC.fit, newdata = d, cause = 2, times = ttt[ttt>T0], landmark = T0)
predCSC_afterT0
```

---

predict.FGR	<i>Predict subject specific risks (cumulative incidence) based on Fine-Gray regression model</i>
-------------	--

---

### Description

Predict subject specific risks (cumulative incidence) based on Fine-Gray regression model

### Usage

```
## S3 method for class 'FGR'
predict(object, newdata, times, ...)
```

### Arguments

object	Result of call to FGR
newdata	Predictor values of subjects for who to predict risks
times	Time points at which to evaluate the risks
...	passed to predict.crr

---

predict.riskRegression	<i>Predict individual risk.</i>
------------------------	---------------------------------

---

### Description

Extract predictions from a risk prediction model.

### Usage

```
## S3 method for class 'riskRegression'
predict(object, newdata, ...)
```

### Arguments

object	Fitted object obtained with one of ARR, LRR, riskRegression.
newdata	A data frame containing predictor variable combinations for which to compute predicted risk.
...	not used

**Author(s)**

Thomas H. Scheike <ts@biostat.ku.dk>

Thomas A. Gerds <>tag@biostat.ku.dk>

**References**

Gerds, TA and Scheike, T and Andersen, PK (2011) Absolute risk regression for competing risks: interpretation, link functions and prediction Research report 11/8. Department of Biostatistics, University of Copenhagen

**Examples**

```
data(Melanoma)
library(proplim)
library(survival)

fit.tarr <- ARR(Hist(time,status)~age+invasion+strata(sex),data=Melanoma,cause=1)
predict(fit.tarr,newdata=data.frame(age=48,
  invasion=factor("level.1",
    levels=levels(Melanoma$invasion)),
  sex=factor("Female",levels=levels(Melanoma$sex))))
predict(fit.tarr,newdata=data.frame(age=48,
  invasion=factor("level.1",
    levels=levels(Melanoma$invasion)),
  sex=factor("Male",levels=levels(Melanoma$sex))))
predict(fit.tarr,newdata=data.frame(age=c(48,58,68),
  invasion=factor("level.1",
    levels=levels(Melanoma$invasion)),
  sex=factor("Male",levels=levels(Melanoma$sex))))
predict(fit.tarr,newdata=Melanoma[1:4,])
```

---

predictCox

*Fast computation of survival probabilities, hazards and cumulative hazards from Cox regression models*

---

**Description**

Fast routine to get baseline hazards and subject specific hazards as well as survival probabilities from a `survival::coxph` or `rms::cph` object

**Usage**

```
predictCox(object, newdata = NULL, times, centered = TRUE,
  type = c("cumhazard", "survival"), keep.strata = TRUE,
  keep.times = TRUE, keep.newdata = FALSE, se = FALSE, band = FALSE,
  iid = FALSE, average.iid = FALSE, nsim.band = 10000,
  conf.level = 0.95, log.transform = TRUE, store.iid = "full")
```

**Arguments**

object	The fitted Cox regression model object either obtained with <code>coxph</code> (survival package) or <code>cph</code> (rms package).
newdata	A <code>data.frame</code> or <code>data.table</code> containing the values of the predictor variables defining subject specific predictions. Should have the same structure as the data set used to fit the object.
times	Time points at which to evaluate the predictions.
centered	Logical. If TRUE return prediction at the mean values of the covariates <code>fit\$mean</code> , if FALSE return a prediction for all covariates equal to zero. in the linear predictor. Will be ignored if argument <code>newdata</code> is used. For internal use.
type	<p>the type of predicted value. Choices are</p> <ul style="list-style-type: none"> <li>• "hazard" the baseline hazard function when argument <code>newdata</code> is not used and the hazard function when argument <code>newdata</code> is used.</li> <li>• "cumhazard" the cumulative baseline hazard function when argument <code>newdata</code> is not used and the cumulative hazard function when argument <code>newdata</code> is used.</li> <li>• "survival" the survival baseline hazard function when argument <code>newdata</code> is not used and the cumulative hazard function when argument <code>newdata</code> is used.</li> </ul> <p>Several choices can be combined in a vector of strings that match (no matter the case) strings "hazard", "cumhazard", "survival".</p>
keep.strata	Logical. If TRUE add the ( <code>newdata</code> ) strata to the output. Only if there any.
keep.times	Logical. If TRUE add the evaluation times to the output.
keep.newdata	Logical. If TRUE add the value of the covariates used to make the prediction in the output list.
se	Logical. If TRUE add the standard error to the output.
band	Logical. If TRUE add the confidence band to the output.
iid	Logical. If TRUE add the influence function to the output.
average.iid	Logical. If TRUE add the average of the influence function over <code>newdata</code> to the output.
nsim.band	the number of simulations used to compute the quantiles for the confidence bands.
conf.level	Level of confidence.
log.transform	Should the confidence intervals/bands be computed on the log (hazard) and log(-log) (survival) scale and be backtransformed. Otherwise they are computed on the original scale and truncated (if necessary).
store.iid	Implementation used to estimate the influence function and the standard error. Can be "full" or "minimal".
...	arguments to be passed to the function <code>iidCox</code> .



## Details

When the argument `newdata` is not specified, the function computes the baseline hazard estimate. See (Ozenne et al., 2017) section "Handling of tied event times".

Otherwise the function computes survival probabilities with confidence intervals/bands. See (Ozenne et al., 2017) section "Confidence intervals and confidence bands for survival probabilities". The survival is computed using the exponential approximation (equation 3).

When setting `log.transform` to `TRUE`, the standard error that is returned is before back-transformation to the original scale.

A detailed explanation about the meaning of the argument `store.iid` can be found in (Ozenne et al., 2017) Appendix B "Saving the influence functions".

The function is not compatible with time varying predictor variables.

The centered argument enables us to reproduce the results obtained with the `basehaz` function from the `survival` package but should not be modified by the user.

## Value

A list with some or all of the following elements:

- `times`: the time points at which the other elements are evaluated.
- `hazard`: When argument `newdata` is not used the baseline hazard function, otherwise the predicted hazard function.
- `cumhazard`: When argument `newdata` is not used the cumulative baseline hazard function, otherwise the predicted cumulative hazard function.
- `survival`: When argument `newdata` is not used the survival probabilities corresponding to the baseline hazard, otherwise the predicted survival probabilities.
- `cumhazard.se/survival.se`: The standard errors of the predicted cumulative hazard function/survival.
- `(hazard.iid/cumhazard.iid/survival.iid)`: (array) the value of the influence of each subject used to fit the object (dim 3) for each subject in `newdata` (dim 1) and each time (dim 2).
- `(cumhazard.average.iid/survival.average.iid)`: (array) the average value of the influence over the subjects in `newdata`, for each subject used to fit the model (dim 1) and each time (dim 2).
- `strata`: The strata variable.

## Author(s)

Brice Ozenne [broz@sund.ku.dk](mailto:broz@sund.ku.dk), Thomas A. Gerds [tag@biostat.ku.dk](mailto>tag@biostat.ku.dk)

## References

Brice Ozenne, Anne Lyngholm Sorensen, Thomas Scheike, Christian Torp-Pedersen and Thomas Alexander Gerds. `riskRegression`: Predicting the Risk of an Event using Cox Regression Models. *The R Journal* (2017) 9:2, pages 440-460.

**Examples**

```

library(survival)

## generate data
set.seed(10)
d <- sampleData(40,outcome="survival") ## training dataset
nd <- sampleData(4,outcome="survival") ## validation dataset
d$time <- round(d$time,1) ## create tied events
# table(duplicated(d$time))

## estimate a stratified Cox model
fit <- coxph(Surv(time,event)~X1 + strata(X2) + X6,
             data=d, ties="breslow", x = TRUE, y = TRUE)

## compute the baseline cumulative hazard
fit.haz <- predictCox(fit)
cbind(survival::basehaz(fit), fit.haz$cumhazard)

## compute individual specific cumulative hazard and survival probabilities
predictCox(fit, newdata=nd, times=c(3,8))

## add confidence intervals computed on the original scale
CI.original <- predictCox(fit, newdata=nd, times=c(3,8), se = TRUE, log.transform = FALSE)
as.data.table(CI.original)
CI.original$cumhazard + 1.96 * CI.original$cumhazard.se

## add confidence intervals computed on the log (or log-log) scale
## and backtransformed
CI.log <- predictCox(fit, newdata=nd, times=c(3,8), se = TRUE, log.transform = TRUE)
as.data.table(CI.log)
exp(log(CI.log$cumhazard) + 1.96 * CI.original$cumhazard.se/CI.log$cumhazard)
exp(log(CI.log$cumhazard) + 1.96 * CI.log$cumhazard.se)

## export iid decomposition relative to the survival probabilities
CI.iid <- predictCox(fit, newdata = d, times = 5, iid = TRUE, se = TRUE,
                   log.transform = FALSE)
as.data.table(CI.iid)[1:5]
rowMeans(CI.iid$survival.iid[,1,]) ## the iid decomposition has 0 expectation
sqrt(rowSums(CI.iid$survival.iid[1:5,1,]^2))

## same but the iid decomposition is averaged over the patients
CI.aviid <- predictCox(fit, newdata = d, times = 5,
                    average.iid = TRUE, log.transform = FALSE)
CI.aviid$survival.average.iid[1:5,]
colMeans(CI.iid$survival.iid[,1,1:5])

## export confidence bands (by default computed on the log scale and backtransformed)
predictCox(fit, newdata=nd, times=c(3,8), se = TRUE, band = TRUE)

## other examples
# one strata variable
fitS <- coxph(Surv(time,event)~strata(X1)+X2,

```

```

data=d, ties="breslow", x = TRUE, y = TRUE)

predictCox(fitS)
predictCox(fitS, newdata=nd, times = 1)

# two strata variables
set.seed(1)
d$U=sample(letters[1:5],replace=TRUE,size=NROW(d))
d$V=sample(letters[4:10],replace=TRUE,size=NROW(d))
nd$U=sample(letters[1:5],replace=TRUE,size=NROW(nd))
nd$V=sample(letters[4:10],replace=TRUE,size=NROW(nd))
fit2S <- coxph(Surv(time,event)~X1+strata(U)+strata(V)+X2,
              data=d, ties="breslow", x = TRUE, y = TRUE)

cbind(survival::basehaz(fit2S),predictCox(fit2S,type="cumhazard")$cumhazard)
predictCox(fit2S)
predictCox(fitS, newdata=nd, times = 3)

# left truncation
test2 <- list(start=c(1,2,5,2,1,7,3,4,8,8),
              stop=c(2,3,6,7,8,9,9,9,14,17),
              event=c(1,1,1,1,1,1,1,0,0,0),
              x=c(1,0,0,1,0,1,1,1,0,0))
m.cph <- coxph(Surv(start, stop, event) ~ 1, test2, x = TRUE)
as.data.table(predictCox(m.cph))

basehaz(m.cph)

```

---

predictCoxPL	<i>Computation of survival probabilities from Cox regression models using the product limit estimator.</i>
--------------	--

---

### Description

Same as predictCox except that the survival is estimated using the product limit estimator.

### Usage

```

predictCoxPL(object, newdata, times, type = c("cumhazard", "survival"),
             se = FALSE, band = FALSE, ...)

```

### Arguments

object	The fitted Cox regression model object either obtained with coxph (survival package) or cph (rms package).
newdata	A data.frame or data.table containing the values of the predictor variables defining subject specific predictions. Should have the same structure as the data set used to fit the object.
times	Time points at which to evaluate the predictions.

type	the type of predicted value. Choices are "hazard", "cumhazard", and "survival". See <a href="#">predictCox</a> for more details.
se	Logical. If TRUE add the standard error to the output.
band	Logical. If TRUE add the confidence band to the output.
...	additional arguments to be passed to <a href="#">predictCox</a> .

## Examples

```
library(survival)

set.seed(10)
d <- sampleData(40,outcome="survival")
nd <- sampleData(4,outcome="survival")
d$time <- round(d$time,1)
fit <- coxph(Surv(time,event)~X1 + strata(X2) + X6,
             data=d, ties="breslow", x = TRUE, y = TRUE)
predictCoxPL(fit, newdata = d, times = 1:5)
fit <- coxph(Surv(time,event)~X1 + X2 + X6,
             data=d, ties="breslow", x = TRUE, y = TRUE)
predictCoxPL(fit, newdata = d, times = 1:5)

#### Compare exp to product limit
set.seed(10)
A <- predictCoxPL(fit, newdata = d[1:5], times = 1:5, se = TRUE, band = TRUE, log.transform = FALSE)
set.seed(10)
B <- predictCox(fit, newdata = d[1:5], times = 1:5, se = TRUE, band = TRUE, log.transform = FALSE)

A$survival - B$survival
A$survival.lower - B$survival.lower
A$survival.upper - B$survival.upper
A$survival.lowerBand - B$survival.lowerBand
A$survival.upperBand - B$survival.upperBand
```

---

predictRisk

*Extracting predicting risks from regression models*

---

## Description

Extract event probabilities from fitted regression models and machine learning objects.

## Usage

```
## S3 method for class 'glm'
predictRisk(object,newdata,...)
## S3 method for class 'cox.aalen'
predictRisk(object,newdata,times,...)
## S3 method for class 'cph'
```

```

predictRisk(object,newdata,times,...)
## S3 method for class 'coxph'
predictRisk(object,newdata,times,...)
## S3 method for class 'matrix'
predictRisk(object,newdata,times,cause,...)
## S3 method for class 'selectCox'
predictRisk(object,newdata,times,...)
## S3 method for class 'psm'
predictRisk(object,newdata,times,...)
## S3 method for class 'survfit'
predictRisk(object,newdata,times,...)
## S3 method for class 'riskRegression'
predictRisk(object,newdata,times,cause,...)
## S3 method for class 'prodlm'
predictRisk(object,newdata,times,cause,...)
## S3 method for class 'rfsrc'
predictRisk(object,newdata,times,cause,...)
## S3 method for class 'FGR'
predictRisk(object,newdata,times,cause,...)
## S3 method for class 'CauseSpecificCox'
predictRisk(object,newdata,times,cause,...)

```

### Arguments

object	A fitted model from which to extract predicted event probabilities
newdata	A data frame containing predictor variable combinations for which to compute predicted event probabilities.
...	Additional arguments that are passed on to the current method.
times	A vector of times in the range of the response variable, for which the cumulative incidences event probabilities are computed.
cause	Identifies the cause of interest among the competing events.

### Details

The function `predictRisk` is a generic function, meaning that it invokes specifically designed functions depending on the 'class' of the first argument.

See [predictRisk](#).

In uncensored binary outcome data there is no need to choose a time point.

When operating on models for survival analysis (without competing risks) the function still predicts the risk, as  $1 - S(t|X)$  where  $S(t|X)$  is survival chance of a subject characterized by  $X$ .

When there are competing risks (and the data are right censored) one needs to specify both the time horizon for prediction (can be a vector) and the cause of the event. The function then extracts the absolute risks  $F_c(t|X)$  aka the cumulative incidence of an event of type/cause  $c$  until time  $t$  for a subject characterized by  $X$ . Depending on the model it may or not be possible to predict the risk of all causes in a competing risks setting. For example. a cause-specific Cox (CSC) object allows to predict both cases whereas a Fine-Gray regression model (FGR) is specific to one of the causes.

**Value**

For binary outcome a vector with predicted risks. For survival outcome with and without competing risks a matrix with as many rows as `NROW(newdata)` and as many columns as `length(times)`. Each entry is a probability and in rows the values should be increasing.

**Author(s)**

Thomas A. Gerds <tag@biostat.ku.dk>

**See Also**

See [predictRisk](#).

**Examples**

```
## binary outcome
library(rms)
set.seed(7)
x <- abs(rnorm(20))
d <- data.frame(y=rbinom(20,1,x/max(x)),x=x,z=rnorm(20))
nd <- data.frame(y=rbinom(8,1,x/max(x)),x=abs(rnorm(8)),z=rnorm(8))
fit <- lrm(y~x+z,d)
predictRisk(fit,newdata=nd)

## survival outcome
# generate survival data
library(prodlim)
set.seed(100)
d <- sampleData(100,outcome="survival")
d[,X1:=as.numeric(as.character(X1))]
d[,X2:=as.numeric(as.character(X2))]
# then fit a Cox model
library(rms)
cphmodel <- cph(Surv(time,event)~X1+X2,data=d,surv=TRUE,x=TRUE,y=TRUE)
# or via survival
library(survival)
coxphmodel <- coxph(Surv(time,event)~X1+X2,data=d,x=TRUE,y=TRUE)

# Extract predicted survival probabilities
# at selected time-points:
ttt <- quantile(d$time)
# for selected predictor values:
ndat <- data.frame(X1=c(0.25,0.25,-0.05,0.05),X2=c(0,1,0,1))
# as follows
predictRisk(cphmodel,newdata=ndat,times=ttt)
predictRisk(coxphmodel,newdata=ndat,times=ttt)

# stratified cox model
sfit <- coxph(Surv(time,event)~strata(X1)+X2,data=d,x=TRUE,y=TRUE)
predictRisk(sfit,newdata=d[1:3,],times=c(1,3,5,10))

## simulate learning and validation data
```

```

learndat <- sampleData(100,outcome="survival")
valdat <- sampleData(100,outcome="survival")
## use the learning data to fit a Cox model
library(survival)
fitCox <- coxph(Surv(time,event)~X1+X2,data=learndat,x=TRUE,y=TRUE)
## suppose we want to predict the survival probabilities for all subjects
## in the validation data at the following time points:
## 0, 12, 24, 36, 48, 60
psurv <- predictRisk(fitCox,newdata=valdat,times=seq(0,60,12))
## This is a matrix with survival probabilities
## one column for each of the 5 time points
## one row for each validation set individual

# Do the same for a randomSurvivalForest model
# library(randomForestSRC)
# rsfmodel <- rfsrc(Surv(time,event)~X1+X2,data=learndat)
# prsfsurv=predictRisk(rsfmodel,newdata=valdat,times=seq(0,60,12))
# plot(psurv,prsfsurv)

## Cox with ridge option
f1 <- coxph(Surv(time,event)~X1+X2,data=learndat,x=TRUE,y=TRUE)
f2 <- coxph(Surv(time,event)~ridge(X1)+ridge(X2),data=learndat,x=TRUE,y=TRUE)
## Not run:
plot(predictRisk(f1,newdata=valdat,times=10),
      riskRegression:::predictRisk.coxph(f2,newdata=valdat,times=10),
      xlim=c(0,1),
      ylim=c(0,1),
      xlab="Unpenalized predicted survival chance at 10",
      ylab="Ridge predicted survival chance at 10")

## End(Not run)

## competing risks

library(survival)
library(riskRegression)
library(prodlim)
train <- SimCompRisk(100)
test <- SimCompRisk(10)
cox.fit <- CSC(Hist(time,cause)~X1+X2,data=train)
predictRisk(cox.fit,newdata=test,times=seq(1:10),cause=1)

## with strata
cox.fit2 <- CSC(list(Hist(time,cause)~strata(X1)+X2,Hist(time,cause)~X1+X2),data=train)
predictRisk(cox.fit2,newdata=test,times=seq(1:10),cause=1)

```

**Description**

Print average treatment effects

**Usage**

```
## S3 method for class 'ate'
print(x, bootci.method = x$bootci.method, digits = 3, ...)
```

**Arguments**

<code>x</code>	object obtained with function <code>ate</code>
<code>bootci.method</code>	Character. Method for constructing bootstrap confidence intervals. Either "perc" (the default), "norm", "basic", "stud", or "bca". Argument passed to <code>boot::boot.ci</code> .
<code>digits</code>	Number of digits
<code>...</code>	passed to <code>print</code>

**Details**

When using bootstrap resampling the p-values are computing using a test-inversion method, i.e. find the critical confidence level such that one side of the confidence interval overlap the null hypothesis. The p-value is 1 minus the critical confidence level.

---

```
print.CauseSpecificCox
```

*Print of a Cause-Specific Cox regression model*

---

**Description**

Print of a Cause-Specific Cox regression model

**Usage**

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

**Arguments**

<code>x</code>	Object obtained with CSC
<code>...</code>	Passed to <code>print</code>



---

print.FGR	<i>Print of a Fine-Gray regression model</i>
-----------	--

---

**Description**

Print of a Fine-Gray regression model

**Usage**

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

**Arguments**

x	Object fitted with function FGR
...	passed to cmprsk::summary.crr

---

print.influenceCoxTest	<i>Print the results of the influence test</i>
------------------------	--

---

**Description**

Print the results of the influence test

**Usage**

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

**Arguments**

x	Object obtained with influenceCoxTest
...	Passed to print

---

```
print.predictCox      Print predictions from a Cox model
```

---

### Description

Print predictions from a Cox model

### Usage

```
## S3 method for class 'predictCox'
print(x, digits = 3, ...)
```

### Arguments

x	object obtained with the function predictCox.
digits	integer indicating the number of decimal places.
...	Passed to print.

### Examples

```
library(survival)
library(rms)

set.seed(10)
d <- sampleData(1e2, outcome = "survival")
m.cox <- coxph(Surv(time,event)~ X1 + X2 + X3,
              data = d, x = TRUE, y = TRUE)
predictCox(m.cox)

pred <- predictCox(m.cox, newdata = d[1:5,],
                  times = 1:5, type = "survival")
pred

pred.data <- predictCox(m.cox, newdata = d[1:4,],se=1L,
                       times = 1:5, type = "survival", keep.newdata = TRUE)
pred.data

m.cox <- coxph(Surv(time,event)~ strata(X1) + strata(X2) + X3 + X6,
              data = d, x = TRUE, y = TRUE)
pred.cox <- predictCox(m.cox, newdata = d[c(1:5,10,50),],
                      time = 1:5)
pred.cox

m.cox <- cph(Surv(time,event)~ strat(X1) + strat(X2) + X3 + X6,
            data = d, x = TRUE, y = TRUE)
pred.cox <- predictCox(m.cox, newdata = d[c(1:5,10,50),],
                      time = 1:5)
pred.cox
```

```

pred.dataci <- predictCox(m.cox, newdata = d[1:5,],
                        times = 1:5, keep.newdata = TRUE, se = TRUE)
pred.dataci

```

---

print.predictCSC	<i>Print predictions from a Cause-specific Cox proportional hazard regression</i>
------------------	---

---

### Description

Print predictions from a Cause-specific Cox proportional hazard regression

### Usage

```

## S3 method for class 'predictCSC'
print(x, digits = 3, ...)

```

### Arguments

x	object obtained with the function predictCox.
digits	integer indicating the number of decimal places.
...	Passed to print.

### Examples

```

## no strata
d <- sampleData(1e2, outcome = "competing.risks")
m.CSC <- CSC(Hist(time,event)~ X1 + X2 + X6, data = d)
pred.CSC <- predict(m.CSC, time = 1:5, cause = 1,
                  se = TRUE, keep.newdata = TRUE)

pred.CSC
print(pred.CSC, ci = TRUE)

## strata
library(survival)
m.SCSC <- CSC(Hist(time,event)~ strata(X1) + strata(X2) + X6,
             data = d)
pred.SCSC <- predict(m.SCSC, time = 1:5, cause = 1,
                   se = TRUE, keep.newdata = TRUE, keep.strata = TRUE)
pred.SCSC
print(pred.SCSC, ci = TRUE)

```

---

`print.riskRegression` *Print function for riskRegression models*

---

### Description

Print function for riskRegression models

### Usage

```
## S3 method for class 'riskRegression'
print(x, times, digits = 3, eps = 10^-4,
      verbose = TRUE, conf.int = 0.95, ...)
```

### Arguments

<code>x</code>	Object obtained with ARR, LRR or riskRegression
<code>times</code>	Time points at which to show time-dependent coefficients
<code>digits</code>	Number of digits for all numbers but p-values
<code>eps</code>	p-values smaller than this number are shown as such
<code>verbose</code>	Level of verbosity
<code>conf.int</code>	level of confidence. default is 0.95
<code>...</code>	not used

---

`print.Score` *Print Score object*

---

### Description

Print method for risk prediction scores

### Usage

```
## S3 method for class 'Score'
print(x, digits = 3, ...)
```

### Arguments

<code>x</code>	Object obtained with Score.list
<code>digits</code>	Number of digits
<code>...</code>	passed to print

---

`print.subjectWeights`    *Print subject weights*

---

**Description**

Print subject weights

**Usage**

```
## S3 method for class 'subjectWeights'  
print(x, digits = 3, ...)
```

**Arguments**

<code>x</code>	Subject weights
<code>digits</code>	Digits
<code>...</code>	not used

---

`reconstructData`    *Reconstruct the original dataset*

---

**Description**

Reconstruct the original dataset from the elements stored in the coxph object

**Usage**

```
reconstructData(object)
```

**Arguments**

<code>object</code>	a coxph object.
---------------------	-----------------

**Author(s)**

Brice Ozenne broz@sund.ku.dk and Thomas A. Gerds tag@biostat.ku.dk

---

riskRegression	<i>Risk Regression Fits a regression model for the risk of an event – allowing for competing risks.</i>
----------------	---

---

### Description

This is a wrapper for the function `comp.risk` from the `timereg` package. The main difference is one marks variables in the formula that should have a time-dependent effect whereas in `comp.risk` one marks variables that should have a time constant (proportional) effect.

### Usage

```
riskRegression(formula, data, times, link = "relative", cause,
  conf.int = TRUE, cens.model, cens.formula, max.iter = 50,
  conservative = TRUE, ...)
```

### Arguments

formula	Formula where the left hand side specifies the event history <code>event.history</code> and the right hand side the linear predictor. See examples.
data	The data for fitting the model in which includes all the variables included in formula.
times	Vector of times. For each time point in <code>times</code> estimate the baseline risk and the timevarying coefficients.
link	"relative" for the absolute risk regression model. "logistic" for the logistic risk regression model. "prop" for the Fine-Gray regression model.
cause	The cause of interest.
conf.int	If TRUE return the iid decomposition, that can be used to construct confidence bands for predictions.
cens.model	Specified the model for the (conditional) censoring distribution used for deriving weights (IFPW). Defaults to "KM" (the Kaplan-Meier method ignoring covariates) alternatively it may be "Cox" (Cox regression).
cens.formula	Right hand side of the formula used for fitting the censoring model. If not specified the right hand side of <code>formula</code> is used.
max.iter	Maximal number of iterations.
conservative	If TRUE use variance formula that ignores the contribution by the estimate of the inverse of the probability of censoring weights
...	Further arguments passed to <code>comp.risk</code>

### Author(s)

Thomas A. Gerds <tag@biostat.ku.dk>, Thomas H. Scheike <ts@biostat.ku.dk>

## References

Gerds, TA and Scheike, T and Andersen, PK (2011) Absolute risk regression for competing risks: interpretation, link functions and prediction Research report 11/8. Department of Biostatistics, University of Copenhagen

Scheike, Zhang and Gerds (2008), Predicting cumulative incidence probability by direct binomial regression, *Biometrika*, 95, 205-220.

Scheike and Zhang (2007), Flexible competing risks regression modelling and goodness of fit, *LIDA*, 14, 464-483.

Martinussen and Scheike (2006), *Dynamic regression models for survival data*, Springer.

## Examples

```
data(Melanoma,package="riskRegression")
## tumor thickness on the log-scale
Melanoma$logthick <- log(Melanoma$thick)

# Single binary factor

## absolute risk regression
library(survival)
library(prodlim)
fit.arr <- ARR(Hist(time,status)~sex,data=Melanoma,cause=1)
print(fit.arr)
# show predicted cumulative incidences
plot(fit.arr,col=3:4,newdata=data.frame(sex=c("Female","Male")))

## compare with non-parametric Aalen-Johansen estimate
library(prodlim)
fit.aj <- prodlim(Hist(time,status)~sex,data=Melanoma)
plot(fit.aj,conf.int=FALSE)
plot(fit.arr,add=TRUE,col=3:4,newdata=data.frame(sex=c("Female","Male")))

## with time-dependent effect
fit.tarr <- ARR(Hist(time,status)~strata(sex),data=Melanoma,cause=1)
plot(fit.tarr,newdata=data.frame(sex=c("Female","Male")))

## logistic risk regression
fit.lrr <- LRR(Hist(time,status)~sex,data=Melanoma,cause=1)
summary(fit.lrr)

# Single continuous factor

## tumor thickness on the log-scale
Melanoma$logthick <- log(Melanoma$thick)

## absolute risk regression
fit2.arr <- ARR(Hist(time,status)~logthick,data=Melanoma,cause=1)
```

```

print(fit2.arr)
# show predicted cumulative incidences
plot(fit2.arr,col=1:5,newdata=data.frame(logthick=quantile(Melanoma$logthick)))

## comparison with nearest neighbor non-parametric Aalen-Johansen estimate
library(prodlim)
fit2.aj <- prodlim(Hist(time,status)~logthick,data=Melanoma)
plot(fit2.aj,conf.int=FALSE,newdata=data.frame(logthick=quantile(Melanoma$logthick)))
plot(fit2.arr,add=TRUE,col=1:5,lty=3,newdata=data.frame(logthick=quantile(Melanoma$logthick)))

## logistic risk regression
fit2.lrr <- LRR(Hist(time,status)~logthick,data=Melanoma,cause=1)
summary(fit2.lrr)

## change model for censoring weights
library(rms)
fit2a.lrr <- LRR(Hist(time,status)~logthick,
                data=Melanoma,
                cause=1,
                cens.model="cox",
                cens.formula=~sex+epicel+ulcer+age+logthick)
summary(fit2a.lrr)

## compare prediction performance
Score(list(ARR=fit2.arr,AJ=fit2.aj,LRR=fit2.lrr),formula=Hist(time,status)~1,data=Melanoma)

# multiple regression
library(riskRegression)
library(prodlim)
# absolute risk model
multi.arr <- ARR(Hist(time,status)~logthick+sex+age+ulcer,data=Melanoma,cause=1)

# stratified model allowing different baseline risk for the two gender
multi.arr <- ARR(Hist(time,status)~thick+strata(sex)+age+ulcer,data=Melanoma,cause=1)

# stratify by a continuous variable: strata(age)
multi.arr <- ARR(Hist(time,status)~tp(thick,power=0)+strata(age)+sex+ulcer,
                data=Melanoma,
                cause=1)

fit.arr2a <- ARR(Hist(time,status)~tp(thick,power=1),data=Melanoma,cause=1)
summary(fit.arr2a)
fit.arr2b <- ARR(Hist(time,status)~timevar(thick),data=Melanoma,cause=1)
summary(fit.arr2b)

## logistic risk model
fit.lrr <- LRR(Hist(time,status)~thick,data=Melanoma,cause=1)
summary(fit.lrr)

```



```
## nearest neighbor non-parametric Aalen-Johansen estimate
library(prodlim)
fit.aj <- prodlim(Hist(time,status)~thick,data=Melanoma)
plot(fit.aj,conf.int=FALSE)

# prediction performance
x <- Score(list(fit.arr2a,fit.arr2b,fit.lrr),
            data=Melanoma,
            formula=Hist(time,status)~1,
            cause=1,
            split.method="none")
```

---

rowCenter_cpp	<i>Apply - by row</i>
---------------	-----------------------

---

### Description

Fast computation of `sweep(X, MARGIN = 2, FUN = "-", STATS = center)`

### Usage

```
rowCenter_cpp(X, center)
```

### Arguments

<code>X</code>	A matrix.
<code>center</code>	a numeric vector of length equal to the number of rows of <code>x</code>

### Value

A matrix of same size as `X`.

### Author(s)

Brice Ozenne <broz@sund.ku.dk>

### Examples

```
x <- matrix(1,6,5)
sweep(x, MARGIN = 2, FUN = "-", STATS = 1:5)
rowCenter_cpp(x, 1:5 )

rowCenter_cpp(x, colMeans(x) )
```

---

rowCumSum	<i>Apply cumsum in each row</i>
-----------	---------------------------------

---

**Description**

Fast computation of `t(apply(x,1,cumsum))`

**Usage**

```
rowCumSum(x)
```

**Arguments**

`x`                    A matrix.

**Value**

A matrix of same size as `x`.

**Author(s)**

Thomas Alexander Gerds <tag@biostat.ku.dk>

**Examples**

```
x <- matrix(1:8,ncol=2)
rowCumSum(x)
```

---

rowMultiply_cpp	<i>Apply * by row</i>
-----------------	-----------------------

---

**Description**

Fast computation of `sweep(X, MARGIN = 2, FUN = "*", STATS = scale)`

**Usage**

```
rowMultiply_cpp(X, scale)
```

**Arguments**

`X`                    A matrix.  
`scale`                a numeric vector of length equal to the number of rows of `x`

**Value**

A matrix of same size as `X`.

**Author(s)**

Brice Ozenne <broz@sund.ku.dk>

**Examples**

```
x <- matrix(1,6,5)
sweep(x, MARGIN = 2, FUN = "*", STATS = 1:5)
rowMultiply_cpp(x, 1:5 )

rowMultiply_cpp(x, 1/colMeans(x) )
```

---

rowScale_cpp	<i>Apply / by row</i>
--------------	-----------------------

---

**Description**

Fast computation of `sweep(X, MARGIN = 2, FUN = "/", STATS = scale)`

**Usage**

```
rowScale_cpp(X, scale)
```

**Arguments**

<code>X</code>	A matrix.
<code>scale</code>	a numeric vector of length equal to the number of rows of <code>x</code>

**Value**

A matrix of same size as `X`.

**Author(s)**

Brice Ozenne <broz@sund.ku.dk>

**Examples**

```
x <- matrix(1,6,5)
sweep(x, MARGIN = 2, FUN = "/", STATS = 1:5)
rowScale_cpp(x, 1:5 )

rowScale_cpp(x, colMeans(x) )
```

---

rowSumsCrossprod	<i>Apply crossprod and rowSums</i>
------------------	------------------------------------

---

**Description**

Fast computation of `crossprod(rowSums(X),Y)`

**Usage**

```
rowSumsCrossprod(X, Y, transposeY)
```

**Arguments**

`X` A matrix with dimensions  $n*k$ . Hence the result of `rowSums(X)` has length  $n$ .

`Y` A matrix with dimensions  $n*m$ . Can be a matrix with dimension  $m*n$  but then `transposeY` should be TRUE.

`transposeY` Logical. If TRUE transpose `Y` before matrix multiplication.

**Value**

A vector of length  $m$ .

**Author(s)**

Thomas Alexander Gerds <tag@biostat.ku.dk>

**Examples**

```
x <- matrix(1:10,nrow=5)
y <- matrix(1:20,ncol=4)
rowSumsCrossprod(x,y,0)

x <- matrix(1:10,nrow=5)
y <- matrix(1:20,ncol=5)
rowSumsCrossprod(x,y,1)
```

---

rsquared	<i>Explained variation for settings with binary, survival and competing risk outcome</i>
----------	--

---

**Description**

General  $R^2$  for binary outcome and right censored time to event (survival) outcome also with competing risks

**Usage**

```

rsquared(object,...)
IPA(object,...)
## Default S3 method:
rsquared(object,formula,newdata,times,cause,...)
## S3 method for class 'glm'
rsquared(object,formula,newdata,...)
## S3 method for class 'coxph'
rsquared(object,formula,newdata,times,...)
## S3 method for class 'CauseSpecificCox'
rsquared(object,formula,newdata,times,cause,...)
## Default S3 method:
IPA(object,formula,newdata,times,cause,...)
## S3 method for class 'glm'
IPA(object,formula,newdata,...)
## S3 method for class 'coxph'
IPA(object,formula,newdata,times,...)
## S3 method for class 'CauseSpecificCox'
IPA(object,formula,newdata,times,cause,...)

```

**Arguments**

object	Model for which we want $R^2$
...	passed to <code>riskRegression::Score</code>
newdata	Optional validation data set in which to compute $R^2$
formula	Formula passed to <code>Score</code> . If not provided, try to use the formula of the call of object, if any.
cause	For competing risk models the event of interest
times	Vector of time points used as prediction horizon for the computation of Brier scores.

**Details**

$R^2$  is calculated based on the model's predicted risks. The Brier score of the model is compared to the Brier score of the null model.

**Value**

Data frame with explained variation values for the full model.

**Author(s)**

Thomas A. Gerds <tag@biostat.ku.dk>

**See Also**

Score

**Examples**

```

# binary outcome
library(lava)
set.seed(18)
learndat <- sampleData(48,outcome="binary")
lr1 = glm(Y~X1+X2+X7+X9,data=learndat,family=binomial)
rsquared(lr1)

## validation data
valdat=sampleData(94,outcome="binary")
rsquared(lr1,newdata=valdat)

## predicted risks externally given
p1=predictRisk(lr1,newdata=valdat)
rsquared(p1,formula=Y~1,valdat)

# survival
library(survival)
data(pbc)
pbc=na.omit(pbc)
pbctest=(1:NROW(pbc)) %in% sample(1:NROW(pbc),size=.632*NROW(pbc))
pbclearn=pbc[pbctest,]
cox1= coxph(Surv(time,status!=0)~age+sex+log(bili)+log(albumin)+log(protime),
            data=pbclearn,x=TRUE)

## same data
rsquared(cox1,formula=Surv(time,status!=0)~1,times=1000)

## validation data
pbcval=pbc[!pbctest,]
rsquared(cox1,formula=Surv(time,status!=0)~1,newdata=pbcval,times=1000)

## predicted risks externally given
p2=predictRisk(cox1,newdata=pbcval,times=1000)
rsquared(cox1,formula=Surv(time,status!=0)~1,newdata=pbcval,times=1000)

# competing risks
data(Melanoma)
Melanomatest=(1:NROW(Melanoma)) %in% sample(1:NROW(Melanoma),size=.632*NROW(Melanoma))
Melanomalearn=Melanoma[Melanomatest,]
fit1 <- CSC(list(Hist(time,status)~sex,
                Hist(time,status)~invasion+epicel+age),
            data=Melanoma)
rsquared(fit1,times=1000,cause=2)

## validation data
Melanomaval=Melanoma[!Melanomatest,]
rsquared(fit1,formula=Hist(time,status)~1,newdata=Melanomaval,times=1000)

## predicted risks externally given
p3= predictRisk(fit1,cause=1,newdata=Melanomaval,times=1000)

```

```
rsquared(p3, formula=Hist(time, status)~1, cause=1, newdata=Melanomaval, times=1000)
```

---

sampleData

*Simulate data with binary or time-to-event outcome*


---

## Description

Simulate data with binary outcome and 10 covariates.

## Usage

```
sampleData(n, outcome="competing.risks",
  formula= ~ f(X1,2)+f(X2,-0.033)+f(X3,0.4)+f(X6,.1)+f(X7,-.1)+f(X8,.5)+f(X9,-1))
sampleDataTD(n, n.intervals=5, outcome="competing.risks",
  formula= ~ f(X1,2)+f(X2,-0.033)+f(X3,0.4)+f(X6,.1)+f(X7,-.1)+f(X8,.5)+f(X9,-1))
```

## Arguments

n	Sample size
outcome	Character vector. Response variables are generated according to keywords: "binary" = binary response, "survival" = survival response, "competing.risks" = competing risks response
formula	Specify regression coefficients
n.intervals	sampleDataTD only: the maximum number of episodes in which the covariates are updated.

## Details

For the actual lava::regression parameters see the function definition.

## Value

Simulated data as data.table with n rows and the following columns: Y (binary outcome), time (non-binary outcome), event (non-binary outcome), X1-X5 (binary predictors), X6-X10 (continuous predictors)

## Author(s)

Thomas A. Gerds <tag@biostat.ku.dk>

## See Also

lvm

**Examples**

```
sampleData(10,outcome="binary")
sampleData(10,outcome="survival")
sampleData(10,outcome="competing.risks")
```

Score.list

*Score risk predictions***Description**

Methods to score the predictive performance of risk markers and risk prediction models

**Usage**

```
## S3 method for class 'list'
Score(object, formula, data, metrics = c("auc", "brier"),
      summary = NULL, plots = NULL, cause, times, landmarks,
      use.event.times = FALSE, null.model = TRUE, se.fit = TRUE,
      conservative = FALSE, multi.split.test = FALSE, conf.int = 0.95,
      contrasts = TRUE, probs = c(0, 0.25, 0.5, 0.75, 1),
      cens.method = "outside.ipcw", cens.model = "cox", split.method, B, M,
      seed, trainseeds, keep, predictRisk.args, debug = 0L, useEventTimes,
      nullModel, censMethod, censModel, splitMethod, ...)
```

**Arguments**

object	List of risk predictions (see details and examples).
formula	A formula which identifies the outcome (left hand side). E.g., $Y \sim 1$ for binary and $\text{Hist}(\text{time}, \text{status}) \sim 1$ for time-to-event outcome. In right censored data, the right hand side of the formula is used to estimate the inverse probability of censoring weights (IPCW) model.
data	data.frame or data.table in which the formula can be interpreted.
metrics	Character vector specifying which metrics to apply. Case does not matter. Choices are "AUC" and "Brier".
summary	Character vector specifying which summary statistics to apply to the predicted risks. The only choice is "riskQuantile" which enables (time-point specific) boxplots of predicted risks conditional on the outcome (at the time-point). Set to NULL to avoid estimation of retrospective risk quantiles.
plots	Character vector specifying for which plots to put data into the result. Currently implemented are "ROC", "Calibration" and "boxplot". In addition, one can plot AUC and Brier score as function of time as soon as times has at least two different values.
cause	Event of interest. Used for binary outcome $Y$ to specify that risks are risks of the event $Y=\text{event}$ and for competing risks outcome to specify the cause of interest.



<code>times</code>	For survival and competing risks outcome: list of prediction horizons. All times which are greater than the maximal observed time in the data set are automatically removed. Note that the object returned by the function may become huge when the prediction performance is estimated at many prediction horizons.
<code>landmarks</code>	Not yet implemented.
<code>use.event.times</code>	If TRUE merge all unique event times with the vector given by argument <code>times</code> .
<code>null.model</code>	If TRUE fit a risk prediction model which ignores the covariates and predicts the same value for all subjects. The model is fitted using data and the left hand side of formula. For binary outcome this is just the empirical prevalence. For (right censored) time to event outcome, the null models are equal to the Kaplan-Meier estimator (no competing risks) and the Aalen-Johansen estimator (with competing risks).
<code>se.fit</code>	Logical or 0 or 1. If FALSE or 0 do not calculate standard errors.
<code>conservative</code>	Logical, only relevant in right censored data. If TRUE ignore variability of the estimate of the inverse probability of censoring weights when calculating standard errors for prediction performance parameters. This can potentially reduce computation time and memory usage at a usually very small expense of a slightly higher standard error.
<code>multi.split.test</code>	Logical or 0 or 1. If FALSE or 0 do not calculate multi-split tests. This argument is ignored when <code>split.method</code> is "none".
<code>conf.int</code>	Either logical or a numeric value between 0 and 1. In right censored data, confidence intervals are based on Blanche et al (see references). Setting FALSE prevents the computation confidence intervals. TRUE means compute 95 percent confidence intervals and corresponding p-values for AUC and Brier score. If set to 0.87, the level of significance is 13 percent. So, do not set it to 0.87.
<code>contrasts</code>	Either logical or a list of contrasts. A list of contrasts defines which risk prediction models (markers) should be contrasted with respect to their prediction performance. If TRUE do all possible comparisons. For example, when object is a list with two risk prediction models and <code>null.model=TRUE</code> setting TRUE is equivalent to <code>list(c(0, 1, 2), c(1, 2))</code> where <code>c(0, 1, 2)</code> codes for the two comparisons: 1 vs 0 and 2 vs 0 (positive integers refer to elements of object, 0 refers to the benchmark null model which ignores the covariates). This again is equivalent to explicitly setting <code>list(c(0, 1), c(0, 2), c(1, 2))</code> . A more complex example: Suppose object has 7 elements and you want to do the following 3 comparisons: 6 vs 3, 2 vs 5 and 2 vs 3, you should set <code>contrasts=c(6, 3), c(2, 5, 3)</code> .
<code>probs</code>	Quantiles for retrospective summary statistics of the predicted risks. This affects the result of the function <code>boxplot.Score</code> .
<code>cens.method</code>	Method for dealing with right censored data. Either "outside.ipcw" or "inside.ipcw". "outside.ipcw" Here IPCW refers to inverse probability of censoring weights. The attributes <code>inside</code> and <code>outside</code> are only relevant for cross-validation ( <code>split.method</code> ) where the <code>inside</code> method computes the IPCW model in the the loop in each of the current test data sets whereas the <code>outside</code> method computes the IPCW model only once in the full data. Also, jackknife pseudo-values may become another option for all statistics. Right now they are only used for calibration curves.

<code>cens.model</code>	Model for estimating inverse probability of censored weights. Implemented are the Kaplan-Meier method ("km") and Cox regression ("cox") both applied to the censored times. If the right hand side of formula does not specify covariates, the Kaplan-Meier method is used even if this argument is set to "cox".
<code>split.method</code>	Method for cross-validation. Right now the only choice is <code>bootcv</code> in which case bootstrap learning sets are drawn with or without replacement (argument <code>M</code> ) from data. The data not included in the current bootstrap learning set are used as validation set to compute the prediction performance.
<code>B</code>	Number of bootstrap sets for cross-validation.
<code>M</code>	Size of subsamples for bootstrap cross-validation. If specified it has to be an integer smaller than the size of data.
<code>seed</code>	Super seed for setting training data seeds when randomly splitting (bootstrapping) the data during cross-validation.
<code>trainseeds</code>	Seeds for training models during cross-validation.
<code>keep</code>	list of characters (not case sensitive) which determines additional output. "residuals" provides Brier score residuals and "splitindex" provides sampling index used to split the data into training and validation sets. "vcov" provides the variance-covariance matrix of the estimated parameters.
<code>predictRisk.args</code>	A list of argument-lists to control how risks are predicted. The names of the lists should be the S3-classes of the object. The argument-lists are then passed on to the S3-class specific <code>predictRisk</code> method. For example, if your object contains one or several random forest model fitted with the function <code>randomForestSRC::rfsrc</code> then you can specify additional arguments for the function <code>riskRegression::predictRisk.rfsrc</code> which will pass these on to the function <code>randomForestSRC::predict.rfsrc</code> . A specific example in this case would be <code>list(rfsrc=list(na.action="na.i</code> A more flexible approach is to write a new <code>predictRisk</code> S3-method. See Details.
<code>debug</code>	Logical. If TRUE indicate landmark in progress of the program.
<code>useEventTimes</code>	obsolete.
<code>nullModel</code>	obsolete.
<code>censMethod</code>	obsolete.
<code>censModel</code>	obsolete.
<code>splitMethod</code>	obsolete.
<code>...</code>	Named list containing additional arguments that are passed on to the <code>predictRisk</code> methods corresponding to object. See examples.

## Details

The function implements a toolbox for the risk prediction modeller: all tools work for the three outcomes: (1) binary (uncensored), (2) right censored time to event without competing risks, (3) right censored time to event with competing risks

Computed are the (time-dependent) Brier score and the (time-dependent) area under the ROC curve for a list of risk prediction models either in external validation data or in the learning data using bootstrap cross-validation. The function optionally provides results for plotting (time-point specific)

ROC curves, for (time-point specific) calibration curves and for (time-point specific) retrospective boxplots.

For uncensored binary outcome the DeLong-DeLong test is used to contrast AUC of rival models. In right censored survival data (with and without competing risks) the p-values correspond to Wald tests based on standard errors obtained with an estimate of the influence function as described in detail in the appendix of Blanche et al. (2015).

This function works with one or multiple models that predict the risk of an event  $R(t|X)$  for a subject characterized by predictors  $X$  at time  $t$ . With binary endpoints (outcome 0/1 without time component) the risk is simply  $R(X)$ . In case of a survival object without competing risks the function still works with predicted event probabilities, i.e.,  $R(t|X)=1-S(t|X)$  where  $S(t|X)$  is the predicted survival chance for subject  $X$  at time  $t$ .

The already existing predictRisk methods (see methods(predictRisk)) may not cover all models and methods for predicting risks. But users can quickly extend the package as explained in detail in Mogensen et al. (2012) for the predecessors `pec::predictSurvProb` and `pec::predictEventProb` which have been unified as `riskRegression::predictRisk`.

Bootstrap Crossvalidation (see also Gerds & Schumacher 2007 and Mogensen et al. 2012)

$B=10$ ,  $M$  (not specified or  $M=NROW(data)$ ) Training of each of the models in each of 10 bootstrap data sets (learning data sets). Learning data sets are obtained by sampling  $NROW(data)$  subjects of the data set with replacement. There are roughly  $.632*NROW(data)$  subjects in the learning data (inbag) and  $.368*NROW(data)$  subjects not in the validation data sets (out-of-bag).

These are used to estimate the scores: AUC, Brier, etc. Reported are averages across the 10 splits.

```
## Bootstrap with replacement set.seed(13) N=17 data = data.frame(id=1:N, y=rbinom(N,1,.3),x=rnorm(N))
boot.index = sample(1:N,size=N,replace=TRUE) boot.index inbag = 1:N outofbag = !inbag
learn.data = data[inbag] val.data = data[outofbag] riskRegression::getSplitMethod("bootcv",B=10,N=17)
NOTE: the number .632 is the expected probability to draw one subject (for example subject
1) with replacement from the data, which does not depend on the sample size: B=10000 N=137
mean(sapply(1:B, function(b){match(1,sample(1:N,size=N,replace=TRUE),nomatch=0)}))
N=30 mean(sapply(1:B, function(b){match(1,sample(1:N,size=N,replace=TRUE),nomatch=0)}))
N=300 mean(sapply(1:B, function(b){match(1,sample(1:N,size=N,replace=TRUE),nomatch=0)}))
```

```
## Bootstrap without replacement (training size set to be 70 percent of data) B=10, M=.7
```

Training of each of the models in each of 10 bootstrap data sets (learning data sets). Learning data sets are obtained by sampling  $round(.8*NROW(data))$  subjects of the data set without replacement. There are  $NROW(data)-round(.8*NROW(data))$  subjects not in the learning data sets. These are used to estimate the scores: AUC, Brier, etc. Reported are averages across the 10 splits.

```
set.seed(13) N=17 data = data.frame(id=1:N, y=rbinom(N,1,.3),x=rnorm(N))
boot.index = sample(1:N,size=M,replace=FALSE) boot.index inbag = 1:N outofbag = !inbag
learn.data = data[inbag] val.data = data[outofbag] riskRegression::getSplitMethod("bootcv",B=10,N=17)
```

## Value

List with scores and assessments of contrasts, i.e., tests and confidence limits for performance and difference in performance (AUC and Brier), summaries and plots. Most elements are `indata.table` format.

## Author(s)

Thomas A Gerds <tag@biostat.ku.dk> and Paul Blanche <paul.blanche@univ-ubs.fr>

## References

- Ulla B. Mogensen, Hemant Ishwaran, Thomas A. Gerds (2012). Evaluating Random Forests for Survival Analysis Using Prediction Error Curves. *Journal of Statistical Software*, 50(11), 1-23. URL <http://www.jstatsoft.org/v50/i11/>.
- Paul Blanche, Cecile Proust-Lima, Lucie Loubere, Claudine Berr, Jean- Francois Dartigues, and Helene Jacqmin-Gadda. Quantifying and comparing dynamic predictive accuracy of joint models for longitudinal marker and time-to-event in presence of censoring and competing risks. *Biometrics*, 71 (1):102–113, 2015.
- P. Blanche, J-F Dartigues, and H. Jacqmin-Gadda. Estimating and comparing time-dependent areas under receiver operating characteristic curves for censored event times with competing risks. *Statistics in Medicine*, 32(30):5381–5397, 2013.
- E. Graf et al. (1999), Assessment and comparison of prognostic classification schemes for survival data. *Statistics in Medicine*, vol 18, pp= 2529–2545.
- Efron, Tibshirani (1997) *Journal of the American Statistical Association* 92, 548–560 Improvement On Cross-Validation: The .632+ Bootstrap Method.
- Gerds, Schumacher (2006), Consistent estimation of the expected Brier score in general survival models with right-censored event times. *Biometrical Journal*, vol 48, 1029–1040.
- Thomas A. Gerds, Martin Schumacher (2007) Efron-Type Measures of Prediction Error for Survival Analysis *Biometrics*, 63(4), 1283–1287 doi:10.1111/j.1541-0420.2007.00832.x
- Martin Schumacher, Harald Binder, and Thomas Gerds. Assessment of survival prediction models based on microarray data. *Bioinformatics*, 23(14):1768-74, 2007.
- Mark A. van de Wiel, Johannes Berkhof, and Wessel N. van Wieringen Testing the prediction error difference between 2 predictors *Biostatistics* (2009) 10(3): 550-560 doi:10.1093/biostatistics/kxp011

## Examples

```
# binary outcome
library(lava)
set.seed(18)
learndat <- sampleData(48,outcome="binary")
testdat <- sampleData(40,outcome="binary")

## score logistic regression models
lr1 = glm(Y~X1+X2+X7+X9,data=learndat,family=binomial)
lr2 = glm(Y~X3+X5,data=learndat,family=binomial)
Score(list("LR(X1+X2+X7+X9)"=lr1,"LR(X3+X5)"=lr2),formula=Y~1,data=testdat)

## ROC curve and calibration plot
xb=Score(list("LR(X1+X2+X7+X9)"=lr1,"LR(X3+X5+X6)"=lr2),formula=Y~1,
          data=testdat,plots=c("calibration","ROC"))
plotROC(xb)
plotCalibration(xb)

## compute AUC for a list of continuous markers
markers = as.list(testdat[,.(X6,X7,X8,X9,X10)])
Score(markers,formula=Y~1,data=testdat,metrics=c("auc"))
```

```

# cross-validation
## Not run:
learndat=sampleData(400,outcome="binary")
lr1a = glm(Y~X6,data=learndat,family=binomial)
lr2a = glm(Y~X7+X8+X9,data=learndat,family=binomial)
## bootstrap cross-validation
x1=Score(list("LR1"=lr1a,"LR2"=lr2a),formula=Y~1,data=learndat,split.method="bootcv",B=100)
x1
## leave-one-out and leave-pair-out bootstrap
x2=Score(list("LR1"=lr1a,"LR2"=lr2a),formula=Y~1,data=learndat,
          split.method="loob",
          B=100,plots="calibration")
x2

## End(Not run)
# survival outcome

# Score Cox regression models
library(survival)
library(rms)
library(prodlm)
set.seed(18)
trainSurv <- sampleData(100,outcome="survival")
testSurv <- sampleData(40,outcome="survival")
cox1 = coxph(Surv(time,event)~X1+X2+X7+X9,data=trainSurv, y=TRUE, x = TRUE)
cox2 = coxph(Surv(time,event)~X3+X5+X6,data=trainSurv, y=TRUE, x = TRUE)
xs=Score(list("Cox(X1+X2+X7+X9)"=cox1,"Cox(X3+X5+X6)"=cox2),
         formula=Surv(time,event)~1,data=testSurv,conf.int=FALSE,times=c(5,8))
xs

# time-dependent AUC for list of markers
survmarkers = as.list(testSurv[,.(X6,X7,X8,X9,X10)])
Score(survmarkers,
      formula=Surv(time,event)~1,metrics="auc",data=testSurv,
      conf.int=TRUE,times=c(5,8))

# compare models on test data
Score(list("Cox(X1+X2+X7+X9)"=cox1,"Cox(X3+X5+X6)"=cox2),
      formula=Surv(time,event)~1,data=testSurv,conf.int=TRUE,times=c(5,8))

# crossvalidation models in traindata
## Not run:
library(survival)
set.seed(18)
trainSurv <- sampleData(400,outcome="survival")
cox1 = coxph(Surv(time,event)~X1+X2+X7+X9,data=trainSurv, y=TRUE, x = TRUE)
cox2 = coxph(Surv(time,event)~X3+X5+X6,data=trainSurv, y=TRUE, x = TRUE)
x1 = Score(list("Cox(X1+X2+X7+X9)"=cox1,"Cox(X3+X5+X6)"=cox2),
          formula=Surv(time,event)~1,data=trainSurv,conf.int=TRUE,times=c(5,8),
          split.method="loob",B=100,plots="calibration")

x2= Score(list("Cox(X1+X2+X7+X9)"=cox1,"Cox(X3+X5+X6)"=cox2),
          formula=Surv(time,event)~1,data=trainSurv,conf.int=TRUE,times=c(5,8),

```

```

split.method="bootcv",B=100)

## End(Not run)

# restrict number of comparisons
## Not run:
Score(list("Cox(X1+X2+X7+X9)"=cox1,"Cox(X3+X5+X6)"=cox2),
       formula=Surv(time,event)~1,data=trainSurv,contrasts=TRUE,
       null.model=FALSE,conf.int=TRUE,times=c(5,8),split.method="bootcv",B=3)

## End(Not run)
# competing risks outcome
set.seed(18)
trainCR <- sampleData(40,outcome="competing.risks")
testCR <- sampleData(40,outcome="competing.risks")
library(riskRegression)
library(cmprsk)
# Cause-specific Cox regression
csc1 = CSC(Hist(time,event)~X1+X2+X7+X9,data=trainCR)
csc2 = CSC(Hist(time,event)~X3+X5+X6,data=trainCR)
# Fine-Gray regression
fgr1 = FGR(Hist(time,event)~X1+X2+X7+X9,data=trainCR,cause=1)
fgr2 = FGR(Hist(time,event)~X3+X5+X6,data=trainCR,cause=1)
Score(list("CSC(X1+X2+X7+X9)"=csc1,"CSC(X3+X5+X6)"=csc2,
          "FGR(X1+X2+X7+X9)"=fgr1,"FGR(X3+X5+X6)"=fgr2),
       formula=Hist(time,event)~1,data=testCR,se.fit=1L,times=c(5,8))

## Not run:
# reproduce some results of Table IV of Blanche et al. Stat Med 2013
data(Paquid)
ResPaquid <- Score(list("DSST"--Paquid$DSST,"MMSE"--Paquid$MMSE),
                  formula=Hist(time,status)~1,
                  data=Paquid,
                  null.model = FALSE,
                  conf.int=TRUE,
                  metrics=c("auc"),
                  times=c(3,5,10),
                  plots="ROC")

ResPaquid
plotROC(ResPaquid,time=5)

## End(Not run)

```

---

selectCox

*Backward variable selection in the Cox regression model*


---

### Description

This is a wrapper function which first selects variables in the Cox regression model using `fastbw` from the `rms` package and then returns a fitted Cox regression model with the selected variables.

**Usage**

```
selectCox(formula, data, rule = "aic")
```

**Arguments**

formula	A formula object with a Surv object on the left-hand side and all the variables on the right-hand side.
data	Name of an data frame containing all needed variables.
rule	The method for selecting variables. See <a href="#">fastbw</a> for details.

**Details**

This function first calls `cph` then `fastbw` and finally `cph` again.

**References**

Ulla B. Mogensen, Hemant Ishwaran, Thomas A. Gerds (2012). Evaluating Random Forests for Survival Analysis Using Prediction Error Curves. *Journal of Statistical Software*, 50(11), 1-23. URL <http://www.jstatsoft.org/v50/i11/>.

**Examples**

```
library(pec)
data(GBSG2)
library(survival)
f <- selectCox(Surv(time,cens)~horTh+age+menostat+tsize+tgrade+pnodes+progre+estrec ,
               data=GBSG2)
```

---

selectJump	<i>Evaluate the influence function at selected times</i>
------------	--

---

**Description**

Evaluate the influence function at selected times

**Usage**

```
selectJump(IF, times, type)
```

**Arguments**

IF	influence function returned by <code>iidCox</code>
times	the times at which the influence function should be assessed
type	can be "hazard" or/and "cumhazard".

**Value**

An object with the same dimensions as IF

**Author(s)**

Brice Ozenne broz@sund.ku.dk

---

simMelanoma

*Simulate data alike the Melanoma data*

---

**Description**

Simulate data alike the Melanoma data

**Usage**

```
simMelanoma(n)
```

**Arguments**

n                    sample size

**Details**

This is based on the functionality of `library(lava)`.

**Value**

data table of size n

**Author(s)**

Thomas A. Gerds <tag@biostat.ku.dk>

**Examples**

```
set.seed(71)
simMelanoma(3)
```



---

sliceMultiply\_cpp      *Apply \* by slice*

---

**Description**

Fast computation of `sweep(X, MARGIN = 1:2, FUN = "*", STATS = scale)`

**Usage**

```
sliceMultiply_cpp(X, M)
```

**Arguments**

`X`                    An array.  
`M`                    A matrix with the same number of row and columns as `X`.

**Value**

An array of same size as `X`.

**Author(s)**

Brice Ozenne <broz@sund.ku.dk>

**Examples**

```
x <- array(1, dim = c(2,6,5))
M <- matrix(1:12,2,6)
sweep(x, MARGIN = 1:2, FUN = "*", STATS = M)
sliceMultiply_cpp(x, M)
```

---

sliceScale\_cpp      *Apply / by slice*

---

**Description**

Fast computation of `sweep(X, MARGIN = 1:2, FUN = "/", STATS = scale)`

**Usage**

```
sliceScale_cpp(X, M)
```

**Arguments**

X                    An array.  
M                    A matrix with the same number of row and columns as X.

**Value**

An array of same size as X.

**Author(s)**

Brice Ozenne <broz@sund.ku.dk>

**Examples**

```
x <- array(1, dim = c(2,6,5))  
M <- matrix(1:12,2,6)  
sweep(x, MARGIN = 1:2, FUN = "/", STATS = M)  
sliceScale_cpp(x, M)
```

---

splitStrataVar                    *Reconstruct each of the strata variables*

---

**Description**

Reconstruct each of the strata variables from the strata variable stored in the coxph object.

**Usage**

```
splitStrataVar(object)
```

**Arguments**

object                a coxph object.

**Author(s)**

Brice Ozenne broz@sund.ku.dk and Thomas A. Gerds tag@biostat.ku.dk

---

subjectWeights	<i>Estimation of censoring probabilities at subject specific times</i>
----------------	--

---

**Description**

This function is used internally to construct pseudo values by inverse of the probability of censoring weights.

**Usage**

```
subjectWeights(formula, data, method = c("cox", "marginal", "km", "nonpar",
    "forest", "none"), args, lag = 1)
```

**Arguments**

formula	A survival formula like, $\text{Surv}(\text{time}, \text{status}) \sim 1$ or $\text{Hist}(\text{time}, \text{status}) \sim 1$ where $\text{status} = 0$ means censored. The status variable is internally reversed for estimation of censoring rather than survival probabilities. Some of the available models, see argument <code>model</code> , will use predictors on the right hand side of the formula.
data	The data used for fitting the censoring model
method	Censoring model used for estimation of the (conditional) censoring distribution.
args	Arguments passed to the fitter of the method.
lag	If equal to 1 then obtain $G(T_i   X_i)$ , if equal to 0 estimate the conditional censoring distribution at the <code>subject.times</code> , i.e. $(G(T_i   X_i))$ .

**Details**

Inverse of the probability of censoring weights usually refer to the probabilities of not being censored at certain time points. These probabilities are also the values of the conditional survival function of the censoring time given covariates. The function `subjectWeights` estimates the conditional survival function of the censoring times and derives the weights.

**IMPORTANT:** the data set should be ordered, `order(time, -status)` in order to get the weights in the right order for some choices of method.

**Value**

times	The times at which weights are estimated
weights	Estimated weights at individual time values <code>subject.times</code>
lag	The time lag.
fit	The fitted censoring model
method	The method for modelling the censoring distribution
call	The call

**Author(s)**

Thomas A. Gerds <tag@biostat.ku.dk>

**Examples**

```
library(proplim)
library(survival)
dat=SimSurv(300)

dat <- dat[order(dat$time,-dat$status),]

# using the marginal Kaplan-Meier for the censoring times

WKM=subjectWeights(Hist(time,status)~X2,data=dat,method="marginal")
plot(WKM$fit)
WKM$fit
WKM$weights

# using the Cox model for the censoring times given X2

WCox=subjectWeights(Surv(time,status)~X2,data=dat,method="cox")
WCox
plot(WCox$weights,WKM$weights)

# using the stratified Kaplan-Meier for the censoring times given X2

WKM2 <- subjectWeights(Surv(time,status)~X2,data=dat,method="nonpar")
plot(WKM2$fit,add=FALSE)
```

---

summary.FGR

*Summary of a Fine-Gray regression model*


---

**Description**

Summary of a Fine-Gray regression model

**Usage**

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

**Arguments**

object	Object fitted with function FGR
...	passed to cmprsk::summary.crr

---

summary.riskRegression

*Summary of a risk regression model*


---

**Description**

Summary of a risk regression model

**Usage**

```
## S3 method for class 'riskRegression'
summary(object, times, digits = 3,
        pvalue.digits = 4, eps = 10^-4, verbose = TRUE, ...)
```

**Arguments**

object	Object obtained with ARR, LRR or riskRegression
times	Time points at which to show time-dependent coefficients
digits	Number of digits for all numbers but p-values
pvalue.digits	Number of digits for p-values
eps	p-values smaller than this number are shown as such
verbose	Level of verbosity
...	not used

---

SurvResponseVar

*Extract the time and event variable from a Cox model*


---

**Description**

Extract the time and event variable from a Cox model

**Usage**

```
SurvResponseVar(formula)
```

**Arguments**

formula	a formula
---------	-----------

**Author(s)**

Brice Ozenne broz@sund.ku.dk

# Index

## \*Topic **datasets**

Melanoma, [45](#)

Paquid, [47](#)

## \*Topic **survival**

CSC, [31](#)

FGR, [35](#)

ipcw, [43](#)

plot.riskRegression, [49](#)

plotEffects, [54](#)

predict.riskRegression, [62](#)

predictRisk, [68](#)

riskRegression, [78](#)

selectCox, [94](#)

subjectWeights, [99](#)

ARR (riskRegression), [78](#)

as.data.table.predictCox, [4](#)

as.data.table.predictCSC, [4](#)

ate, [5](#)

autoplot.ate, [10](#)

autoplot.predictCox, [11](#)

autoplot.predictCSC, [12](#)

boot2pvalue, [13](#)

boxplot.Score, [15](#)

calcSeCox, [17](#)

calcSeCSC, [18](#)

coef.CauseSpecificCox, [20](#)

coef.riskRegression, [20](#)

colCenter\_cpp, [21](#)

colCumSum, [21](#)

colMultiply\_cpp, [22](#)

colScale\_cpp, [23](#)

colSumsCrossprod, [23](#)

confBandCox, [24](#)

coxBaseEstimator, [25](#)

coxCenter, [25](#)

coxDesign, [26](#)

coxFormula, [27](#)

coxLP, [27](#)

coxN, [28](#)

coxph, [32](#)

coxSpecialStrata, [29](#)

coxStrata, [29](#)

coxVarCov, [30](#)

coxVariableName, [31](#)

CSC, [19](#), [31](#)

discreteRoot, [34](#)

extractStrata, [35](#)

fastbw, [95](#)

FGR, [35](#)

getSplitMethod, [37](#)

Hist, [36](#)

iidCox, [39](#)

influenceCoxTest, [40](#)

IPA (rsquared), [84](#)

ipcw, [43](#)

lines, [53](#)

LRR (riskRegression), [78](#)

Melanoma, [45](#)

model.frame, [53](#)

model.matrix.phreg, [46](#)

Paquid, [47](#)

penalizedS3, [48](#)

plot.riskRegression, [49](#)

plotAUC, [50](#)

plotBrier, [51](#)

plotCalibration, [52](#)

plotEffects, [54](#)

plotRisk, [56](#)

plotROC, [57](#)

predict.CauseSpecificCox, 59  
predict.FGR, 62  
predict.riskRegression, 62  
predictCox, 63, 68  
predictCoxPL, 67  
predictRisk, 68, 69, 70  
print.ate, 71  
print.CauseSpecificCox, 72  
print.FGR, 73  
print.influenceCoxTest, 73  
print.predictCox, 74  
print.predictCSC, 75  
print.riskRegression, 76  
print.Score, 76  
print.subjectWeights, 77

reconstructData, 77  
riskRegression, 36, 78  
rowCenter\_cpp, 81  
rowCumSum, 82  
rowMultiply\_cpp, 82  
rowScale\_cpp, 83  
rowSumsCrossprod, 84  
rsquared, 84

sampleData, 87  
sampleDataTD (sampleData), 87  
Score (Score.list), 88  
Score.list, 88  
selectCox, 94  
selectJump, 95  
simMelanoma, 96  
sliceMultiply\_cpp, 97  
sliceScale\_cpp, 97  
SmartControl, 50, 52, 53, 55, 56, 58  
splitStrataVar, 98  
subjectWeights, 99  
summary.FGR, 100  
summary.riskRegression, 101  
SurvResponseVar, 101