Package 'monoreg'

February 11, 2022

Type Package

Title Bayesian Monotonic Regression Using a Marked Point Process Construction

Version 2.0

Date 2022-02-09

Author Olli Saarela, Christian Rohrbeck

Maintainer Olli Saarela <olli.saarela@utoronto.ca>

Description An extended version of the nonparametric Bayesian monotonic regression procedure described in Saarela & Arjas (2011) <DOI:10.1111/j.1467-9469.2010.00716.x>, allowing for multiple additive monotonic components in the linear predictor, and time-to-event outcomes through case-base sampling. The extension and its applications, including estimation of absolute risks, are described in Saarela & Arjas (2015) <DOI:10.1111/sjos.12125>. The package also implements the nonparametric ordinal regression model described in Saarela, Rohrbeck & Arjas <arXiv:2007.01390>.

SystemRequirements GNU GSL

License GPL (>= 2)

Depends R (>= 3.4.0)

NeedsCompilation yes

Repository CRAN

Date/Publication 2022-02-11 13:10:02 UTC

R topics documented:

getcmat — *A matrix of point process configurations in p-dimensional covariate space*

Description

This function returns a matrix where the rows correspond to each possible point process specification in p-dimensional covariate space; this may be helpful in specifying the argument `settozero` in the functions `monoreg` and `monosurv`.

Usage

```
getcmat(p)
```

Arguments

p	Number of covariate axes.

Value

A zero-one matrix with 2^p - 1 rows and p columns.

Author(s)

Olli Saarela <olli.saarela@utoronto.ca>

monoreg — *Bayesian monotonic regression*

Description

This function implements an extended version of the Bayesian monotonic regression procedure for continuous outcomes described in Saarela & Arjas (2011), allowing for multiple additive monotonic components. The simulated example below replicates the one in the reference.

Usage

```
monoreg(niter=15000, burnin=5000, adapt=5000, refresh=10, thin=5,
        birthdeath=10, seed=1, rhoa=0.1, rhob=0.1, deltai=0.1,
        drange=2.0, predict, include, response, offset=NULL,
        axes, covariates, settozero, package)
```

Arguments

niter	Total number of MCMC iterations.
burnin	Number of iterations used for burn-in period.
adapt	Number of iterations used for adapting Metropolis-Hastings proposals.
refresh	Interval for producing summary output of the state of the MCMC sampler.
thin	Interval for saving the state of the MCMC sampler.
birthdeath	Number of birth-death proposals attempted at each iteration.
seed	Seed for the random generator.
rhoa	Shape parameter of a Gamma hyperprior for the Poisson process rate parameters.
rhob	Scale parameter of a Gamma hyperprior for the Poisson process rate parameters.
deltai	Range for a uniform proposal for the function level parameters.
drange	Allowed range for the monotonic function components.
predict	Indicator vector for the observations for which absolute risks are calculated (not included in the likelihood expression).
include	Indicator vector for the observations to be included in the likelihood expression.
response	Vector of response variable values.
offset	Vector of offset terms to be included in the linear predictor (optional).
axes	A matrix where the columns specify the covariate axes in the non-parametrically specified regression functions. Each variable here must be scaled to zero-one interval.
covariates	A matrix of additional covariates to be included in the linear predictor of the events of interest. Must include at least a vector of ones to specify an intercept term.
settozero	A zero-one matrix specifying the point process construction. Each row represents a point process, while columns correspond to the columns of the argument axes, indicating whether the column is one of the dimensions specifying the domain of the point process. (See function getcmat.)
package	An integer vector specifying the additive component into which each point process (row) specified in argument settozero is placed.

Value

A list with elements

steptotal	A sample of total number of points in the marked point process construction.
steps	A sample of the number of points used per each additive component.
rho	A sample of the Poisson process rate parameters (one per each point process specified).
loglik	A sample of log-likelihood values.
beta	A sample of regression coefficients for the variables specified in the argument covariates.

phi	A sample of non-parametric regression function levels for each observation specified in the argument predict.
pred	A sample of predicted means for each observation specified in the argument predict.
sigma	A sample of residual standard error parameters.

Author(s)

Olli Saarela <olli.saarela@utoronto.ca>

References

Saarela O., Arjas E. (2011). A method for Bayesian monotonic multiple regression. Scandinavian Journal of Statistics, 38:499–513.

Examples

```
## Not run:
library(monoreg)
set.seed(1)
# nobs <- 1000
nobs <- 50
sigma <- 0.01
x1 <- runif(nobs)
x2 <- runif(nobs)

# 6 different monotonic regression surfaces:
# mu <- sqrt(x1)
mu <- 0.5 * x1 + 0.5 * x2
# mu <- pmin(x1, x2)
# mu <- 0.25 * x1 + 0.25 * x2 + 0.5 * (x1 + x2 > 1.0)
# mu <- 0.25 * x1 + 0.25 * x2 + 0.5 * (pmax(x1, x2) > 0.5)
# mu <- ifelse((x1 - 1.0)^2 + (x2 - 1.0)^2 < 1.0, sqrt(1.0 - (x1 - 1.0)^2 - (x2 - 1.0)^2), 0.0)

y <- rnorm(nobs, mu, sigma)

# results <- monoreg(niter=15000, burnin=5000, adapt=5000, refresh=10,
results <- monoreg(niter=5000, burnin=2500, adapt=2500, refresh=10,
                   thin=5, birthdeath=10, seed=1, rhoa=0.1, rhob=0.1,
                   deltai=0.1, drange=2.0, predict=rep(1.0, nobs),
                   include=rep(1.0, nobs), response=y, offset=NULL,
                   axes=cbind(x1,x2), covariates=rep(1.0, nobs),
                   settozero=getcmat(2), package=rep(1,3))

# pdf(file.path(getwd(), 'pred3d.pdf'), width=6.0, height=6.0, paper='special')
op <- par(mar=c(2,2,0,0), oma=c(0,0,0,0), mgp=c(2.5,1,0), cex=0.75)
pred <- colMeans(results$pred)
idx <- order(pred, decreasing=TRUE)

tr <- persp(z=matrix(c(NA,NA,NA,NA), 2, 2), zlim=c(0,1),
            xlim=c(0,1), ylim=c(0,1),
```

```
            ticktype='detailed', theta=-45, phi=25, ltheta=25,
            xlab='X1', ylab='X2', zlab='mu')
for (i in 1:nobs) {
    lines(c(trans3d(x1[idx[i]], x2[idx[i]], 0.0, tr)$x,
            trans3d(x1[idx[i]], x2[idx[i]], pred[idx[i]], tr)$x),
          c(trans3d(x1[idx[i]], x2[idx[i]], 0.0, tr)$y,
            trans3d(x1[idx[i]], x2[idx[i]], pred[idx[i]], tr)$y),
          col='gray70')
}
points(trans3d(x1[idx], x2[idx], pred[idx], tr), pch=21, bg='white')
par(op)
# dev.off()

## End(Not run)
```

`monosurv` *Bayesian monotonic regression for time-to-event outcomes*

Description

This function implements an extended version of the Bayesian monotonic regression procedure described in Saarela & Arjas (2011), allowing for multiple additive monotonic components, and time-to-event outcomes through case-base sampling. Logistic/multinomial regression is fitted if no time variable is present. The extension and its applications, including estimation of absolute risks, are described in Saarela & Arjas (2015). The example below does logistic regression; for an example of modeling a time-to-event outcome, please see the documentation for the dataset `risks`.

Usage

```
monosurv(niter=15000, burnin=5000, adapt=5000, refresh=10, thin=5,
         birthdeath=10, timevar=0, seed=1, rhoa=0.1, rhob=0.1,
         years=NULL, deltai=0.1, drange=2.0, predict, include,
         casestatus, sprob=NULL, offset=NULL, tstart=NULL, axes,
         covariates, ccovariates=NULL, settozero, package, cr=NULL)
```

Arguments

`niter`	Total number of MCMC iterations.
`burnin`	Number of iterations used for burn-in period.
`adapt`	Number of iterations used for adapting Metropolis-Hastings proposals.
`refresh`	Interval for producing summary output of the state of the MCMC sampler.
`thin`	Interval for saving the state of the MCMC sampler.
`birthdeath`	Number of birth-death proposals attempted at each iteration.
`timevar`	Number identifying the column in argument `axes` representing a time variable. Zero if no time variable is present, in which case a logistic/multinomial regression is fitted (instead of a hazard regression).

seed	Seed for the random generator.
rhoa	Shape parameter of a Gamma hyperprior for the Poisson process rate parameters.
rhob	Scale parameter of a Gamma hyperprior for the Poisson process rate parameters.
years	Time period over which absolute risks are calculated (on a time scale scaled to zero-one interval; optional).
deltai	Range for a uniform proposal for the function level parameters.
drange	Allowed range for the monotonic function components, on log-rate/log-odds scale.
predict	Indicator vector for the observations for which absolute risks are calculated (not included in the likelihood expression).
include	Indicator vector for the observations to be included in the likelihood expression.
casestatus	An integer vector indicating the case status (0=censoring, 1=event of interest, 2=competing event).
sprob	Vector of sampling probabilities/rates for each person/person-moment (optional).
offset	Vector of offset terms to be included in the linear predictor of the events of interest (optional).
tstart	Vector of entry times on a time scale scaled to zero-one interval (optional).
axes	A matrix where the columns specify the covariate axes in the non-parametrically specified regression functions. Each variable here must be scaled to zero-one interval.
covariates	A matrix of additional covariates to be included in the linear predictor of the events of interest. Must include at least a vector of ones to specify an intercept term.
ccovariates	A matrix of additional covariates to be included in the linear predictor of the competing events (optional).
settozero	A zero-one matrix specifying the point process construction. Each row represents a point process, while columns correspond to the columns of the argument axes, indicating whether the column is one of the dimensions specifying the domain of the point process. (See function getcmat.)
package	An integer vector specifying the additive component into which each point process (row) specified in argument settozero is placed.
cr	A zero-one vector indicating the additive components to be placed in the linear predictor of the competing causes (optional).

Value

A list with elements

steptotal	A sample of total number of points in the marked point process construction.
steps	A sample of the number of points used per each additive component.
rho	A sample of the Poisson process rate parameters (one per each point process specified).

loglik	A sample of log-likelihood values.
beta	A sample of regression coefficients for the variables specified in the argument covariates.
betac	A sample of regression coefficients for the variables specified in the argument ccovariates.
phi	A sample of non-parametric regression function levels for each observation specified in the argument predict.
risk	A sample of absolute risks of the event of interest for each observation specified in the argument predict.
crisk	A sample of absolute risks of the competing event for each observation specified in the argument predict.

Author(s)

Olli Saarela <olli.saarela@utoronto.ca>

References

Saarela O., Arjas E. (2011). A method for Bayesian monotonic multiple regression. Scandinavian Journal of Statistics, 38:499–513.

Saarela O., Arjas E. (2015). Non-parametric Bayesian hazard regression for chronic disease risk assessment. Scandinavian Journal of Statistics, 42:609–626.

Examples

```
## Not run:
library(monoreg)
set.seed(1)
# nobs <- 1000
nobs <- 50
x1 <- runif(nobs)
x2 <- runif(nobs)

# 6 different monotonic regression surfaces:
# mu <- sqrt(x1)
mu <- 0.5 * x1 + 0.5 * x2
# mu <- pmin(x1, x2)
# mu <- 0.25 * x1 + 0.25 * x2 + 0.5 * (x1 + x2 > 1.0)
# mu <- 0.25 * x1 + 0.25 * x2 + 0.5 * (pmax(x1, x2) > 0.5)
# mu <- ifelse((x1 - 1.0)^2 + (x2 - 1.0)^2 < 1.0, sqrt(1.0 - (x1 - 1.0)^2 - (x2 - 1.0)^2), 0.0)

y <- rbinom(nobs, 1, mu)

# results <- monosurv(niter=15000, burnin=5000, adapt=5000, refresh=10,
results <- monosurv(niter=5000, burnin=2500, adapt=2500, refresh=10,
                    thin=5, birthdeath=10, seed=1,
                    rhoa=0.1, rhob=0.1, deltai=0.5, drange=10.0,
                    predict=rep(1.0, nobs), include=rep(1.0, nobs),
                    casestatus=y, axes=cbind(x1,x2), covariates=rep(1.0, nobs),
```

```
                    settozero=getcmat(2), package=rep(1,3))

# pdf(file.path(getwd(), 'pred3d.pdf'), width=6.0, height=6.0, paper='special')
op <- par(mar=c(2,2,0,0), oma=c(0,0,0,0), mgp=c(2.5,1,0), cex=0.75)
pred <- colMeans(results$risk)
idx <- order(pred, decreasing=TRUE)

tr <- persp(z=matrix(c(NA,NA,NA,NA), 2, 2), zlim=c(0,1),
            xlim=c(0,1), ylim=c(0,1),
            ticktype='detailed', theta=-45, phi=25, ltheta=25,
            xlab='X1', ylab='X2', zlab='mu')
for (i in 1:nobs) {
    lines(c(trans3d(x1[idx[i]], x2[idx[i]], 0.0, tr)$x,
            trans3d(x1[idx[i]], x2[idx[i]], pred[idx[i]], tr)$x),
          c(trans3d(x1[idx[i]], x2[idx[i]], 0.0, tr)$y,
            trans3d(x1[idx[i]], x2[idx[i]], pred[idx[i]], tr)$y),
          col='gray70')
}
points(trans3d(x1[idx], x2[idx], pred[idx], tr), pch=21, bg='white')
par(op)
# dev.off()

## End(Not run)
```

---

ordmonoreg *Bayesian monotonic regression*

Description

This function implements the non-parametric Bayesian monotonic regression procedure for ordinal outcomes described in Saarela, Rohrbeck & Arjas (2022).

Usage

```
ordmonoreg(niter=15000, burnin=5000, adapt=5000, refresh=10, thin=20,
   birthdeath=1, logit=FALSE, gam=FALSE, seed=1, rhoa=0.1, rhob=0.1,
   deltai=0.5, dlower=0, dupper=1, invprob=1.0, dc=0.0,
   predict, include, outcome, axes, covariates=NULL,
   cluster=NULL, ncluster=NULL, settozero)
```

Arguments

niter	Total number of MCMC iterations.
burnin	Number of iterations used for burn-in period.
adapt	Number of iterations used for adapting Metropolis-Hastings proposals.
refresh	Interval for producing summary output of the state of the MCMC sampler.
thin	Interval for saving the state of the MCMC sampler.

birthdeath	Number of birth-death proposals attempted at each iteration.
logit	Indicator for fitting the model on logit scale.
gam	Indicator for fitting non-monotonic generalized additive models for covariate effects.
seed	Seed for the random generator.
rhoa	Shape parameter of a Gamma hyperprior for the Poisson process rate parameters.
rhob	Scale parameter of a Gamma hyperprior for the Poisson process rate parameters.
deltai	Range for a uniform proposal for the function level parameters.
dlower	Lower bound for the allowed range for the monotonic function.
dupper	Upper bound for the allowed range for the monotonic function.
invprob	Probability with which to propose keeping the original direction of monotonicity.
dc	First parameter of the conditional beta prior to counter spiking behaviour near the origin - 1.
predict	Indicator vector for the observations for which absolute risks are calculated (not included in the likelihood expression).
include	Indicator vector for the observations to be included in the likelihood expression.
outcome	Vector of outcome variable values.
axes	A matrix where the columns specify the covariate axes in the non-parametrically specified regression functions. Each variable here must be scaled to zero-one interval.
covariates	A matrix of additional covariates to be included in the linear predictor of the events of interest (optional).
cluster	A vector of indicators for cluster membership, numbered from 0 to ncluster-1 (optional).
ncluster	Number of clusters (optional).
settozero	A zero-one matrix specifying the point process construction. Each row represents a point process, while columns correspond to the columns of the argument axes, indicating whether the column is one of the dimensions specifying the domain of the point process. (See function getcmat.)

Value

A list with elements

steptotal	A sample of total number of points in the marked point process construction.
steps	A sample of the number of points used per each additive component.
rho	A sample of the Poisson process rate parameters (one per each point process specified).
loglik	A sample of log-likelihood values.
tau	A sample of random intercept variances.

alpha	A sample of random intercepts.
beta	A sample of regression coefficients for the variables specified in the argument covariates.
lambda	A sample of non-parametric regression function levels for each observation specified in the argument predict.
pred	A sample of predicted means for each observation specified in the argument predict.
sigmasq	A sample of autoregressive prior variances.

Author(s)

Olli Saarela <olli.saarela@utoronto.ca>, Christian Rohrbeck <cr777@bath.ac.uk>

References

Saarela O., Rohrbeck C., Arjas E. (2022). Bayesian non-parametric ordinal regression under a monotonicity constraint. Bayesian Analysis, accepted for publication. arXiv:2007.01390

Examples

```
library(monoreg)
expit <- function(x) {1/(1+exp(-x))}
logit <- function(p) {log(p)-log(1-p)}
set.seed(1)
nobs <- 500
x <- sort(runif(nobs))
ngrid <- 100
xgrid <- seq(1/ngrid, (ngrid-1)/ngrid, by=1/ngrid)
ngrid <- length(xgrid)
ncat <- 4
beta <- 0.75
disc <- c(Inf, Inf, 0.75, 0.5)
gamma <- c(0,0,0.25,0.5)

surv <- matrix(NA, nobs, ncat)
cdf <- matrix(NA, nobs, ncat)
cols <- c('black','red','blue','green')
for (i in 1:ncat) {
    surv[,i] <- expit(logit((ncat-(i-1))/ncat) + beta * x +
                  gamma[i] * (x > disc[i]) - gamma[i] * (x < disc[i]))
    if (i==1)
        plot(x, surv[,i], type='l', col=cols[i], ylim=c(0,1), lwd=2, ylab='S')
    else
        lines(x, surv[,i], col=cols[i], lwd=2)
}
head(surv)

for (i in 1:ncat) {
    if (i<ncat)
        cdf[,i] <- 1.0 - surv[,i+1]
    else
```

```
        cdf[,i] <- 1.0
    if (0) {
        if (i==1)
            plot(x, cdf[,i], type='l', col=cols[i], ylim=c(0,1), lwd=2, ylab='F')
        else
            lines(x, cdf[,i], col=cols[i], lwd=2)
    }
}
head(cdf)

u <- runif(nobs)
y <- rep(NA, nobs)
for (i in 1:nobs)
    y[i] <- findInterval(u[i], cdf[i,]) + 1
table(y)

xwindow <- 0.1/2
mw <- matrix(NA, nobs, ncat)
grid <- sort(x)
for (i in 1:nobs) {
    idx <- (x > grid[i] - xwindow) & (x < grid[i] + xwindow)
    for (j in 1:ncat)
        mw[i,j] <- sum(y[idx] >= j)/sum(idx)
}
for (j in 1:ncat) {
    lines(grid, mw[,j], lty='dashed', col=cols[j])
}

# results <- ordmonoreg(niter=15000, burnin=5000, adapt=5000, refresh=10, thin=5,
results <- ordmonoreg(niter=3000, burnin=1000, adapt=1000, refresh=10, thin=4,
           birthdeath=1, logit=FALSE, gam=FALSE, seed=1, rhoa=0.1, rhob=0.1,
           deltai=0.2, dlower=0, dupper=1, invprob=1.0, dc=0.0,
           predict=c(rep(0,nobs),rep(1,ngrid)),
           include=c(rep(1,nobs),rep(0,ngrid)),
           outcome=c(y, rep(1,ngrid)),
           axes=c(x, xgrid), covariates=NULL, cluster=NULL, ncluster=NULL,
           settozero=getcmat(1))

s <- results$lambda
dim(s)
lines(xgrid, colMeans(subset(s, s[,1]==0)[,2:(ngrid+1)]))
lines(xgrid, colMeans(subset(s, s[,1]==1)[,2:(ngrid+1)]), col='red')
lines(xgrid, colMeans(subset(s, s[,1]==2)[,2:(ngrid+1)]), col='blue')
lines(xgrid, colMeans(subset(s, s[,1]==3)[,2:(ngrid+1)]), col='green')
legend('bottomright', legend=c(expression(P(Y>=1)), expression(P(Y>=2)),
expression(P(Y>=3)), expression(P(Y>=4))),
lwd=2, col=cols)
```

`risks` *Absolute risks from 7 survival models*

Description

This example dataset includes time-to-event outcomes and absolute risks for reproducing the ROC curves in Figure 3 of Saarela & Arjas (2015), please see the example code below.

Usage

```
data(risks)
```

Format

A data frame containing the elements

tstop Age at the end of the follow-up (scaled to zero-one interval)

censvar Case status (0=censoring, 1=CVD event, 2=other death).

tstart Age at the start of the follow-up (scaled to zero-one interval).

model1 Absolute risk from model 1.

model2 Absolute risk from model 2.

model3 Absolute risk from model 3.

model4 Absolute risk from model 4.

model5 Absolute risk from model 5.

model6 Absolute risk from model 6.

model7 Absolute risk from model 7.

References

Saarela O., Arjas E. (2015). Non-parametric Bayesian hazard regression for chronic disease risk assessment. Scandinavian Journal of Statistics, 42:609–626.

Examples

```
## Not run:
rm(list=ls())
library(monoreg)
library(eha)

# Read the example data:

data(risks)
ftime <- (risks$tstop - risks$tstart)/max(risks$tstop - risks$tstart)
ftime <- ifelse(ftime == 0, ftime + 0.00001, ftime)
censvar <- risks$censvar
case <- censvar == 1
dcase <- censvar == 2
a <- risks$tstart
a2 <- a^2
nobs <- nrow(risks)

# Fit a simple parametric model to remove the effect of baseline age:
```

```
wmodel <- weibreg(Surv(ftime, case) ~ a + a2, shape=1)
summary(wmodel)
wcoef <- c(coef(wmodel), 0.0)
wlp <- crossprod(t(cbind(a, a2)), wcoef[1:(length(wcoef)-2)])
wpar <- exp(wcoef[(length(wcoef)-1):length(wcoef)])

whaz <- function(x, bz) {
    return(exp(bz) * (wpar[2] / wpar[1]) * (x / wpar[1])^(wpar[2] - 1))
}
chint <- function(x, bz) {
    return(exp(bz) * (x / wpar[1])^wpar[2])
}
croot <- function(x, bz, c) {
    return(chint(x, bz) - c)
}

# Age-matched case-base sample for model validation:

set.seed(1)
crate <- chint(ftime, wlp)
csrate <- cumsum(crate)
m <- sum(case) * 10
persons <- rep(1:nobs, rmultinom(1, m, crate/sum(crate)))
moments <- rep(NA, m)
for (i in 1:m) {
    u <- runif(1, 0.0, crate[persons[i]])
    moments[i] <- uniroot(croot, c(0.0, ftime[persons[i]]), c=u,
                          bz=wlp[persons[i]])$root
}
plot(ecdf(risks$tstart[case]), pch=20, col='red')
plot(ecdf(risks$tstart[persons]), pch=20, col='blue', add=TRUE)

rate <- whaz(moments, wlp[persons])
mrate <- mean(rate)

d <- c(rep(0, m), rep(1, sum(censvar == 1)), rep(2, sum(censvar == 2)), censvar)
mom <- c(moments, ftime[censvar == 1], ftime[censvar == 2], rep(1.0, nobs))
per <- c(persons, (1:nobs)[censvar == 1], (1:nobs)[censvar == 2], 1:nobs)

include <- rep(c(1,0), c(m + sum(censvar == 1) + sum(censvar == 2), nobs))
predict <- as.numeric(!include)

offset <- log(sum(crate)/(m * whaz(mom, wlp[per])))
moffset <- rep(log(sum(crate)/(m * mrate)), length(mom))

sprob <- 1/exp(offset)
msprob <- 1/exp(moffset)

stz <- getcmat(2)
settozero <- rbind(stz[1,], stz[1,], stz[2:3,], stz[2:3,])
package <- 1:nrow(settozero)
cr <- c(1,0,rep(1,2),rep(0,2))
```

```
# Fit models removing the age effect:

agecir <- matrix(NA, nobs, 7)
for (i in 1:7) {
    agecir[,i] <- as.numeric(colMeans(
monosurv(niter=15000, burnin=5000, adapt=5000, refresh=10, thin=5,
         birthdeath=10, timevar=1, seed=1, rhoa=0.1, rhob=0.1,
         years=1.0, deltai=0.1, drange=6.0, predict=predict, include=include,
         casestatus=d, sprob=msprob, offset=NULL, tstart=NULL,
         axes=cbind(mom, risks[per,paste('model', i, sep='')]),
         covariates=rep(1.0, length(per)), ccovariates=rep(1.0, length(per)),
         settozero=settozero, package=package, cr=cr)$risk))
    print(i)
}

# Fit models without removing the age effect:

cir <- matrix(NA, nobs, 7)
for (i in 1:7) {
    cir[,i] <- as.numeric(colMeans(
monosurv(niter=15000, burnin=5000, adapt=5000, refresh=10, thin=5,
         birthdeath=10, timevar=1, seed=1, rhoa=0.1, rhob=0.1,
         years=1.0, deltai=0.1, drange=6.0, predict=predict, include=include,
         casestatus=d, sprob=sprob, offset=NULL, tstart=NULL,
         axes=cbind(mom, risks[per,paste('model', i, sep='')]),
         covariates=rep(1.0, length(per)), ccovariates=rep(1.0, length(per)),
         settozero=settozero, package=package, cr=cr)$risk))
    print(i)
}

# Calculate ROC curves:

for (i in 1:7) {
    probs <- as.numeric(risks[,paste('model', i, sep='')])
    cutoffs <- sort(unique(probs), decreasing=TRUE)
    truepos <- rep(NA, length(cutoffs))
    falsepos <- rep(NA, length(cutoffs))
    auc <- rep(0.0, length(cutoffs))
    for (j in 1:length(cutoffs)) {
        ind <- as.numeric(probs > cutoffs[j])
        truepos[j] <- sum(ind * agecir[,i])/sum(agecir[,i])
        falsepos[j] <- sum(ind * (1.0 - agecir[,i]))/sum(1.0 - agecir[,i])
        if (j > 1)
            auc[j] = (truepos[j] + truepos[j-1]) * (falsepos[j] - falsepos[j-1])
    }
    auc <- cumsum(auc) * 0.5
    roc <- cbind(cutoffs, truepos, falsepos, auc)
    save(roc, file=paste('ageroc', i, sep=''))
}

for (i in 1:7) {
    probs <- as.numeric(risks[,paste('model', i, sep='')])
```

```
    cutoffs <- sort(unique(probs), decreasing=TRUE)
    truepos <- rep(NA, length(cutoffs))
    falsepos <- rep(NA, length(cutoffs))
    auc <- rep(0.0, length(cutoffs))
    for (j in 1:length(cutoffs)) {
        ind <- as.numeric(probs > cutoffs[j])
        truepos[j] <- sum(ind * cir[,i])/sum(cir[,i])
        falsepos[j] <- sum(ind * (1.0 - cir[,i]))/sum(1.0 - cir[,i])
        if (j > 1)
            auc[j] = (truepos[j] + truepos[j-1]) * (falsepos[j] - falsepos[j-1])
    }
    auc <- cumsum(auc) * 0.5
    roc <- cbind(cutoffs, truepos, falsepos, auc)
    save(roc, file=paste('roc', i, sep=''))
}

# Plot ROC curves:

# postscript(file.path(getwd(), 'rocs.eps'), paper='special', width=10, height=5,
#            horizontal=FALSE)
op <- par(cex=1, mar=c(3.75,3.75,0.25,0.25), mfrow=c(1,2), mgp=c(2.5,1,0))

plot(1, xlim=c(0,1), ylim=c(0,1), type='n', xlab='False positive fraction',
     ylab='True positive fraction')
abline(0, 1, lty='dashed')
cols=c('darkgray','red','blue','darkgreen','orange','purple','magenta')
aucs <- NULL
for (i in 1:7) {
    load(file=paste('roc', i, sep=''))
    aucs <- c(aucs, max(roc[,4]))
    lines(roc[,3], roc[,2], type='s', lwd=2, col=cols[i])
    for (j in c(0.05,0.1,0.15,0.2)) {
        tp <- approx(roc[,1], roc[,2], xout=j)$y
        fp <- approx(roc[,1], roc[,3], xout=j)$y
        idx <- nobs - findInterval(j,sort(roc[,1]))
            points(fp, tp, col=cols[i], pch=20)
        if (i == 1)
            text(fp, tp-0.015, labels=j, pos=4, offset=0.25, col=cols[i],
                 cex=0.9)
    }
}
legend('bottomright', legend=paste('Model ', 1:7, '; AUC=',
       format(round(aucs, 3), nsmall=3, scientific=FALSE), sep=''),
        col=cols, lty=rep('solid',7), lwd=rep(2,7))

plot(1, xlim=c(0,1), ylim=c(0,1), type='n', xlab='False positive fraction',
     ylab='True positive fraction')
abline(0, 1, lty='dashed')
cols=c('darkgray','red','blue','darkgreen','orange','purple','magenta')
aucs <- NULL
for (i in 1:7) {
    load(file=paste('ageroc', i, sep=''))
    aucs <- c(aucs, max(roc[,4]))
```

```
    lines(roc[,3], roc[,2], type='s', lwd=2, col=cols[i])
    for (j in c(0.05,0.1,0.15,0.2)) {
        tp <- approx(roc[,1], roc[,2], xout=j)$y
        fp <- approx(roc[,1], roc[,3], xout=j)$y
        idx <- nobs - findInterval(j,sort(roc[,1]))
        points(fp, tp, col=cols[i], pch=20)
        if (i == 1)
            text(fp, tp-0.015, labels=j, pos=4, offset=0.25, col=cols[i],
                 cex=0.9)
    }
}
legend('bottomright', legend=paste('Model ', 1:7, '; AUC=',
       format(round(aucs, 3), nsmall=3, scientific=FALSE), sep=''),
        col=cols, lty=rep('solid',7), lwd=rep(2,7))

par(op)
# dev.off()

## End(Not run)
```

Index