Package ‘coxsei’

February 24, 2015

Type Package
Title Fitting a CoxSEI Model
Version 0.1
Date 2015-02-23
Author Feng Chen <feng.chen@unsw.edu.au>
Maintainer Feng Chen <feng.chen@unsw.edu.au>
Description
It fits a CoxSEI (Cox type Self-Exciting Intensity) model to right-censored counting process data.
License GPL (>= 2)
LazyLoad yes
NeedsCompilation yes
Repository CRAN
Date/Publication 2015-02-24 08:03:34

R topics documented:

coxsei-package ......................................................... 2
coxsei ................................................................. 3
coxseiest ............................................................ 5
coxseifit.ex .......................................................... 7
coxseifit.ex ......................................................... 8
coxseiInt ............................................................. 9
coxseisim ............................................................ 10
CumInt ................................................................. 11
dat ................................................................. 12
Dens ................................................................. 13
Dist ................................................................. 13
Quant ............................................................... 14
RND ................................................................. 15
Surv ................................................................. 16

Index 17
coxsei-package

Fit a Cox-type self-exciting intensity model (CoxSEI) to right-censored counting process data

Description

Fit the CoxSEI model using the partial likelihood method.

Details

Package: coxsei
Type: Package
Version: 0.1
Date: 2015-02-24
License: GPL (>=2)
LazyLoad: yes

To use the package, the data needs to be prepared into a data frame containing a column named y for observed event times in ascending order of each individual process, a column named delta indicating if the event is 'death' (1) or 'censoring' (0), a column named id indicating the process id of each event time, and one or more columns giving the value of any covariate variable at the observed event times of each process. Then call the coxseiest function or the identical but much faster coxseiest2 function to estimate the parametric part of the model and then the coxseiInt function to estimate the cumulative baseline intensity function.

Author(s)

Feng Chen <feng.chen@unsw.edu.au>
Maintainer: Feng Chen <feng.chen@unsw.edu.au>

References


See Also

survival
**Description**

Fit a CoxSEI model to counting process data

**Usage**

```r
## the .default and .coxsei in the following are not necessary
coxsei(x,...)
## Default S3 method:
coxsei(x,y,delta,id,par.init,m=2,mit=1000,tr=TRUE,
       method="L-BFGS-B",lower=c(rep(-Inf,ncol(x)), -Inf, 0),
       upper=rep(Inf,ncol(x) + 2),...)
## S3 method for class 'coxsei'
print(x,...)
## S3 method for class 'coxsei'
plot(x,...)
## S3 method for class 'coxsei'
summary(object,...)
```

**Arguments**

- `x` a covariate matrix, or an object of class `coxsei`
- `y` a vector of observed times
- `delta` the individual/group id to which the event/censoring time correspond
- `id` initial parameter guess to start the iteration
- `par.init` lag parameter as in m-dependence
- `mit` max number of iteration
- `tr` whether to trace the optimization or not
- `method` method used in optimization
- `lower` the lower bound of the parameter space if the L-BFGS-B method of optimization is used.
- `upper` the upper bound of the parameter space if the L-BFGS-B method of optimization is used.
- `...` further arguments to `plot.stepfun`
- `object` an object of the class `coxsei`
Value

an object of class coxsei, basically a list of the following components

- **coefficients**: a numeric vector of coefficients
- **vcov**: the variance-covariance matrix
- **zval**: the vector of z-value of the Wald test statistic
- **pval**: the vector of p-values
- **details.par**: a list returned by the `optim` routine
- **cintfn**: a step function as the estimated cumulative baseline intensity function
- **cintvar**: a step function as the variance of the cumulative baseline intensity function estimator
- **details.cint**: a list containing more details about the cint

Author(s)

Feng Chen <feng.chen@unsw.edu.au>

References


See Also

coxseifit.ex

Examples

data(dat, package="coxsei")
acoxsei <- coxsei(dat[,3:5],dat[,1],dat[,2],dat[,6],
c(0.2*1:3,log(0.07),log(10)))
summary(acoxsei)
plot(acoxsei, do.points=FALSE)
**Function to estimate the parametric part of the Cox (proportional intensity) self-exciting point process (CoxSEI) model**

**Description**

Estimate the parametric part of the CoxSEI model using (conditionally) right-censored counting process data.

**Usage**

```r
coxseiestHdatL parNinitL m 
mit 
tr true
method L Bfgs
lower repHInf,ncolHdatI-3,-Inf,0L
upper repHInf,ncolHdatI-3 + 2L
gfun functionHxL paI 
elseifelseHx < 0L repH0L lengthHxIIL pa{Q} * expHpa{R} * xII
})

coxseiestHdatL parNinitL m 
mit 
tr true
method L Bfgs
lower repHInf,ncolHdatI-3,-Inf,0L
upper repHInf,ncolHdatI-3 + 2L
gfun functionHxL paI 
elseifelseHx < 0L repH0L lengthHxIIL pa[1] * expH-`pa[2] * xI
})

coxseiestHdatL parNinitL m 
mit 
tr true
method L Bfgs
lower repHInf,ncolHdatI-3,-Inf,0L
upper repHInf,ncolHdatI-3 + 2L
```

**Arguments**

- `dat`: a data frame with columns `Y` containing the censored event times of each individual process arranged in ascending order with the last time always being the censoring time, `delta` containing the event time indicator with value indicator an event time and 0 a censoring time, `id` specifying the id (process number) of each event time recorded, and the others giving the value of the associated covariate process at the corresponding event times.
- `par.init`: initial guess of the value of the parameters to start the optimization iteration with.
- `m`: order of "autoregression" of the excitation term.
- `mit`: maximum number of iteration in the optimization routine.
- `tr`: if set to TRUE, print some summary information while the optimization routine is running.
- `lower`: vector of lower boundary values of the parameter space.
- `upper`: vector of upper boundary of the parameter space.
- `gfun`: the excitation function. Defaults to the exponential decay function

\[
g(t; \gamma) = \gamma_1 \gamma_2 e^{-\gamma_2 t}
\]
Details

coxseiest uses only R code; coxseiest2 uses external C code, and is expected to be 3~4 times faster than the former; coxseiest3 assumes the excitation function is the exponential function as defaulted by the former two, and hardwares it in the C side of the code, and therefore is much faster than the former two when the exponential excitation function is desired.

Value

A list as that returned by the call to the optimizer routine. For instance,

\[
\begin{align*}
\text{par} & \quad \text{gives the estimate of the parameters} \\
\text{hessian} & \quad \text{gives the inverse of the estimate of the variance-covariance matrix}
\end{align*}
\]

Note

the excitation function has to contain exactly two parameters; a feature that does not seem desirable and might change later.

Author(s)

Feng Chen <feng.chen@unsw.edu.au>

References

coxsei paper

See Also

See optim for the components of the returned value

Examples

data("dat")
## this takes over 15 minutes
##est0 <- coxseiest(dat,par.init=c(0.2,0.4,0.6,0.6,5))
## this one takes about 4 minutes
##est1 <- coxseiest2(dat,par.init=c(0.2,0.4,0.6,0.6,5))
## this one takes about 10 seconds
est2 <- coxseiest3(dat,par.init=c(0.2,0.4,0.6,0.6,5))
fit CoxSEI model using an exponential excitation function

Usage

```r
coxseiexp(Y, delta, id, Z, par.init, m = 2, mit = 1000, tr = TRUE,
method = "L-BFGS-B", lower = c(rep(-Inf, ncol(Z)), -Inf, 0),
upper = rep(Inf, ncol(Z) + 2), ...)
```

Arguments

- `Y`: the observed times (including censoring times)
- `delta`: indicator of event: 1=event, 0=censoring
- `id`: the id of the individual/group the event/censoring corresponds to
- `Z`: covariate matrix
- `par.init`: initial parameter value to start the iteration
- `m`: the lag parameter as in M-dependence
- `mit`: maximum number of iteration allowed in maximizing the log partial likelihood
- `tr`: should the optimization process be 'tr'aced
- `method`: method of optimization; defaults to "L-BFGS-B"
- `lower`: vector of lower boundary values of the parameter space
- `upper`: vector of upper boundary of the parameter space
- `...`: other arguments to be passed to the optimization routine

Value

an object of class "coxsei", basically a list with components

- `coefficients`: a named vector of coefficients
- `vcov`: a symmetric matrix which is supposed to be positive definite when m>0, or with the (np-2)x(np-2) major submatrix positive definite when m=0

Author(s)

Feng Chen <feng.chen@unsw.edu.au>
Description

Fit a CoxSEI model with exponential function to right censored counting process data

Usage

```r
coxseifit.ex(dat, par.init, m = 2, mit = 1000, tr = TRUE,
method = "L-BFGS-B", lower=c(rep(-Inf,ncol(dat)-3),-Inf,0),
upper=rep(Inf,ncol(dat)-3 + 2),...)
```

Arguments

dat The data
par.init initial value of the regression coefficients and coefficients in the excitation function
m the lag parameter (the m-dependence parameter)
mit maximum number of iterations allowed in the optimizer
tr whether to trace the optimization or not
method the method of optimization used by the optim routine
lower vector of lower boundary values of the parameter space
upper vector of upper boundary of the parameter space
... other arguments to be passed to the optimization routine

Value

A list of some components with kind of self-evident meanings by their name

Author(s)

Feng Chen <feng.chen@unsw.edu.au>

See Also

coxseiest, coxseiInt

Examples

data("dat")
csfit <- coxseifit.ex(dat,c(1:3*0.2,0.7,10))
coef(csfit)
plot(csfit$cfintf,do.points=FALSE)
Calculate the estimator of the cumulative baseline intensity function in the CoxSEI model.

Description

It takes the parameter of the parametric part (or its theorized value) and calculate the values of the estimator at the jump times; it also gives the values of the estimator for the variance of the intensity estimator.

Usage

coxseiInt(dat, parest, hessian=NULL, vcovmat=solve(hessian), m = 2,
gfun = function(x, pa) {
  ifelse(x <= 0, 0, pa[1] * pa[2] * exp(-pa[2] * x))
},
gfungrd = function(x, pa){
  if(length(x)==0)return(matrix(0,2,0));
  rbind(pa[2]*exp(-pa[2]*x),
        pa[1]*exp(-pa[2]*x)*(1-pa[2]*x))
})

Arguments

dat a data frame containing the right-censored counting process data
parest the estimate of parameter of the parametric part of the CoxSEI model
hessian the hessian matrix returned by the optimization procedure in the estimation of the parametric part based on partial likelihood
vcovmat the variance-covariance matrix of the estimator of the the parametric components; defaulted to the inverse of the hessian matrix
m autoregressive order in the excitation part of the intensity
gfun the excitation function; defaults to the exponential decay function
gfungrd derivative/gradient function of the excitation function

Value

a list giving the jump times and values at these of the estimator of the cumulative baseline intensity function.

x the ordered death/event times
y the value of the estimator of the intensity function at the observed death/event times
varest the value of the estimator of the variance of the estimator of the intensity function, at the jump times

The step function can be obtained using stepfun, and plotted by setting type="s" in the plot function.
Note

Currently doesn’t compute the standard error or variance estimator of the baseline cumulative intensity estimator.

Author(s)

Feng Chen <feng.chen@unsw.edu.au>

Examples

data("dat")
est <- coxseiest3(dat,c(0.2,0.4,0.6,log(0.06),log(5)))
pe <- est$par; pe[4:5] <- exp(pe[4:5]);
ve <- diag(pe) %*% solve(est$hessian, diag(pe));
cintest <- coxseiInt(dat,pe,vcovmat=ve)
plot(cintest,type="s")

Description

simulate the sample path of the CoxSEI model with given covariate process values, and excitation function and order of autodependence in the excitation term.

Usage

coxseisim(parreg, par, lmd0 = function(tt) (1 + 0.5 * cos(2 * pi * tt)),
g = function(x, par) {
},
censor = 1, m = 2, trace=TRUE,
Z = function(x) matrix(0, length(x), length(parreg))
)

Arguments

parreg the regression parameter
par parameters of the excitation function
lmd0 the baseline intensity function
g the excitation function
censor the censoring time
m order of autoregression in the excitation component of the intensity process
trace whether to trace the data generation process; defaults to TRUE
Z a function to calculate the covariate values at a specified event time
**Value**

A data frame with provided covariate values and the censoring time, and the generated event times.

**Author(s)**

Feng Chen <feng.chen@unsw.edu.au>

**Examples**

```r
nNsmp <- 100;  
z <- matrix(NA,n.smp,3)  
for(i in 1:n.smp)  
z[i,] <- round(c(runif(1,0.5,1.5),runif(1,1.5,2.5),rbinom(1,1,0.5)),2)  
dat <- coxseisim(1:3*0.2,c(0.07,10),censor=rlnorm(1,0,0.1),m=2,  
Z=function(x)matrix(z[i,],length(x),3,byrow=TRUE))  
dat$id <- 1;  
for(i in 2:n.smp){  
datmp <- coxseisim(1:3*0.2,c(0.07,10),censor=rlnorm(1,0,0.1),m=2,  
Z=function(x)matrix(z[i,],length(x),3,byrow=TRUE))  
datmp$id <- i;  
dat <- rbind(dat,datmp)  }
```

---

**CumInt**

**Cumulative intensity function**

**Description**

Calculate the cumulative/integrated hazard/intensity function

**Usage**

```r
CumInt(x, int, ...)
```

**Arguments**

- `x` the value at which to calculate the cumulative function value
- `int` the intensity/hazard rate function. Has to be vectorized.
- `...` the arguments to be passed in to control the behavior of the underlying integrate function.

**Details**

Relies on the numerical integration routine of R.

**Value**

The value(s) of the cumulative hazard function at the specified x value(s).
Warning

The validity of the user supplied intensity function is not checked.

Note

Not intended to be called by the user directly.

Author(s)

Feng Chen <feng.chen@unsw.edu.au>

Examples

curve(CumInt(x, int=function(y)1*(y>=0 & y<2)+3*(y>=2 & y<3)+1*(y>=3)),
0,5,xlab="t",ylab="H(t) of a piece-wise constant hazard fun h(t)"

---

dat  A simulated data set from a CoxSEI model

Description

Simulated from a CoxSEI model with an exponential excitation function and an AR order 2 for the self-excitation effects. Generated using the following code:

```r
set.seed(1); n.smp <- 50;
z <- matrix(NA,n.smp,3); for(i in 1:n.smp) z[i,] <- round(c(runif(1,0.5,1.5),runif(1,1.5,2.5),runif(1,1,2.5)),1);
dat <- coxseisim(1:3*0.2,c(0.07,10),censor=rlnorm(1,0,0.1),m=2, Z=function(x)matrix(x[1],length(x)), id=1);
dat$id <- 1; for(i in 2:n.smp){ dattmp <- coxseisim(1:3*0.2,c(0.07,10),censor=rlnorm(1,0,0.1),m=2, Z=function(x)matrix(x[1],length(x)), id=1);
dattmp$id <- i; dat <- rbind(dat,dattmp) }
```

Usage

```r
data(dat)
```

Format

A data frame with 307 observations on the following 6 variables.

Y  a numeric vector
delta  a numeric vector
Z.1  a numeric vector
Z.2  a numeric vector
Z.3  a numeric vector
id  a numeric vector

Examples

```r
data(dat)
## maybe str(dat); plot(dat) ...
```
**Dens**

*Density function*

**Description**
Evaluate the density function corresponding to the specified intensity/hazard function `int`.

**Usage**
```r
Dens(x, int, ...)
```

**Arguments**
- `x`: the value at which to evaluate the density function
- `int`: the intensity/hazard function. Has to be vectorized.
- `...`: other arguments to be passed to the underlying integrator

**Value**
A numerical value or vector giving the value(s) of the density function

**Note**
Relies on R’s `integrate` function

**Author(s)**
Feng Chen <feng.chen@unsw.edu.au>

**Examples**
```r
set.seed(1); dat <- RND(1000, int=function(x)3*x^2)
hist(dat, freq=FALSE); curve(Dens(x, int=function(x)3*x^2), add=TRUE)
```

---

**Dist**

*Distribution function*

**Description**
Calculate the value at `x` of the distribution function associated with the intensity/hazard function provided through `int`.

**Usage**
```r
Dist(x, int, ...)
```
Arguments

- **x**: the value to evaluate the distribution function at.
- **int**: vectorized function specifying the intensity/hazard function
- **...**: arguments to be passed to the **integrate** function

Value

A number between 0 and 1 inclusive, that gives the value of the distribution function at the specified x value.

Author(s)

Feng Chen <feng.chen@unsw.edu.au>

Examples

```r
curve(Dist(x, int=function(x)3*x^2), 0, 5)
curve(pweibull(x, shape=3), 0, 5, add=TRUE, col=3, lty=3)
```

---

**Quantile function**

calculates the value of the quantile function (inverse of the distribution function) of the survival variable with given intensity/hazard function.

Usage

```r
Quant(p, int, tolerance = .Machine$double.eps, ...)
```

Arguments

- **p**: the (probability) values to calculate the quantiles at
- **int**: the intensity/hazard function. Has to be vectorized.
- **tolerance**: tolerated numerical error in inverting the distribution function.
- **...**: arguments to be passed to **CumInt** (eventually to **integrate**)

Value

A numerical value or vector giving the values of the quantile function at x

Author(s)

Feng Chen <feng.chen@unsw.edu.au>
Examples
curve(Quant(x, int=function(x)3*x^2), from=1e-3, to=1 - 1e-3)
curve(qweibull(x, shape=3), col=3, lty=3, add=TRUE)

Description

RND takes a vectorized positive R function defined on positive reals and returns a vector of \( n \) values of the random variable (survival time) with the specified function as its hazard/intensity rate function.

Usage

\[
\text{RND}(n, \text{int}, \text{tol} = .Machine$double.eps^0.5, \text{epsabs} = 1e-10, \text{epsrel} = 1e-10, \text{limit} = 1000)
\]

Arguments

- \( n \): number of observations.
- \( \text{int} \): hazard rate function of the survival variable, or the intensity function of the one-event point process counting the number (0 or 1) of deaths by following a sample of the surviving subject.
- \( \text{tol} \): tolerance of the numerical error in calculating the inverse of the cumulative distribution function of the survival variable. Defaults to the square root of the machine epsilon.
- \( \text{epsabs} \): maximum absolute error to be tolerated by the integrator.
- \( \text{epsrel} \): maximum relative error to be tolerated by the integrator.
- \( \text{limit} \): maximum number of iterations permitted by the integrator.
- \( \ldots \): other variables to be passed to the underlying functions such as the quantile functions etc.

Value

A vector of \( n \) observations of the survival variable with the supplied intensity/hazard function.

Author(s)

Feng Chen <feng.chen@unsw.edu.au>

Examples

set.seed(1)
dat <- RND(100, int=function(x)x^2)
ks.test(dat, pweibull, shape=3) # p-value = 0.6058
qqplot(dat, rweibull(100, shape=3))
Survival function

Description

Evaluate the survival function corresponding to the given intensity/hazard function.

Usage

```r
Surv(x, int, ...)
```

Arguments

- `x`: value to calculate the value of the survival function for
- `int`: the intensity/hazard function
- `...`: further arguments to be passed to `cumint`

Value

A numerical value or vector giving the value(s) of the survival function at `x`

Author(s)

Feng Chen <feng.chen@unsw.edu.au>

Examples

```r
curve(Surv(x, int=function(x)3*x^2), from=0, to=5)
curve(pweibull(x, shape=3, lower=FALSE), add=TRUE, col=2, lty=3)
```
Index

*Topic \textsc{ CoxseiModels }
coxseiest, 5
coxseiexp, 7
coxseiInt, 9
*Topic \textsc{ CoxseiRegression }
coxseiest, 5
coxseiexp, 7
coxseiInt, 9
*Topic \textsc{ CoxseiSurvival }
coxseiest, 5
coxseiexp, 7
coxseiInt, 9
coxseisim, 10
cumint, 11
dat, 12
dens, 13
dist, 13
quant, 14
RND, 15
Surv, 16
*Topic \textsc{ Datasets }
dat, 12
*Topic \textsc{ Package }
coxsei-package, 2
*Topic \textsc{ Regression }
coxsei, 3
coxseifit.ex, 8
*Topic \textsc{ Survival }
coxsei, 3
coxsei-package, 2
coxseifit.ex, 8
coxsei, 3
coxsei-package, 2
coxseiest, 5, 8
coxseiest2(coxseiest), 5
coxseiest3(coxseiest), 5
coxseiexp, 7
coxseifit.ex, 4, 8
coxseiInt, 8, 9
coxseisim, 10
cumint, 11
dat, 12
dens, 13
dist, 13
optim, 6
plot.coxsei(coxsei), 3
print.coxsei(coxsei), 3
print.summary.coxsei(coxsei), 3
quant, 14
RND, 15
summary.coxsei(coxsei), 3
Surv, 16
survival, 2