Package 'coxext'

July 29, 2014

Type Package

Title Transformation model and Yang Prentice Model

Version 1.0-14

Date 2014-07-29

Author Yifan Yang <yifan.yang@uky.edu>

Maintainer Yifan Yang <yifan.yang@uky.edu>

Description trans.model returns the beta rho and hazard in transformation model with out covariates.

License GPL

Depends rootSolve, YPmodel, gsl, doParallel

NeedsCompilation yes

Archs i386, x64

R topics documented:

coxext-package *Transformation model and Yang Prentice Model*

Description

`trans.model` returns the NPMLE in transformation model without covariate, while G(x)=x.

`transf2.model` returns the NPMLE in transformation model without covariate, while G(x)=Cox-Box transformation.

`yp.model` returns the NPMLE in Yang-Prentice model without covariate using R(t).

`yplamdaapproach.model` returns the NPMLE in Yang-Prentice model without covariate using hazards Lambda(t).

`hr.yp` calculate the average hazard ratio for Yang-Prentice model.

`hr.trans` calculate the average hazard ratio for transformation model.

`ypnpmle.model` calculates Yang-Prentice model NPMLE with co-variates.

`ypnpmle.modelHessian` calculates Yang-Prentice model NPMLE with co-variates and the hessian (information) matrix.

`ypmodel.chk` calculates an onminbus test for Yang-Prentice model with co-variates.

Details

Package:	coxext
Type:	Package
Version:	1.0
Date:	2014-06-06
License:	BSD

Author(s)

Yifan Yang

References

Yang, Song, and Ross Prentice. "Semiparametric analysis of short-term and long-term hazard ratios with two-sample survival data." Biometrika 92, no. 1 (2005): 1-17.

Zeng, D., and D. Y. Lin. "Maximum likelihood estimation in semiparametric regression models with censored data." Journal of the Royal Statistical Society: Series B (Statistical Methodology) 69, no. 4 (2007): 507-564.

Lin, Danyu Y., Lee-Jen Wei, and Zhiliang Ying. "Checking the Cox model with cumulative sums of martingale-based residuals." Biometrika 80.3 (1993): 557-572.

Chen, Li, D. Y. Lin, and Donglin Zeng. "Checking semiparametric transformation models with censored data." Biostatistics 13.1 (2012): 18-31.

Examples

```
set.seed(123456)
be=c(log(0.9),log(1.2))
aa=0;
tbe=exp(be);
n= 160
z= as.numeric(runif(n)>.5);
g1=exp(-be[1]*z);
g2=exp(-be[2]*z);
rx=g1 /g2 *(runif(n)^(-g2)-1);
x=log(1+rx);
cp=.1
ce=exp(cp/1+runif(n)/2);
y=ce
y[x<=ce]=x[x<=ce]
d=as.numeric(y==x);
# yp.model(y,z,d)
# trans.model(y,z,d)
```

`f.HHmat` *C function calculating HHH matrix*

Description

C function calculating HHH matrix

Usage

```
f.HHmat(
GG1,GG2,GG3,EFpart4,LH,r1i,r2i,Rhatd,d.1,td,n.1
)
```

Arguments

`GG1`	Matrix: subterm 1
`GG2`	Matrix: subterm 1
`GG3`	Matrix: subterm 1
`EFpart4`	Matrix: E F Part4
`LH`	Matrix: score Information Matrix
`r1i`	vector: r1[i]
`r2i`	vector: r2[i]
`Rhatd`	vector: d R(ti) for event points
`d.1`	vector: Index set of events
`td`	number of observations
`n.1`	number of events

Value

Matrix

Examples

```
# It simpilifes this inner tripple loop:
#  for (i in 1: td) {
#   HHH[i,1,] <- GG1[i,]
#   HHH[i,2,] <- GG2[i,]
#   HHH[i,3,] <- GG3[i,]
#   vec.tmp =as.double(r1i[i]+r2i[i]*Rhatd[1:n.1]) *as.double(d.1[1:n.1] <=i) # <=!!!
#   HHH[i,-(1:3),] <- triangleassign(vec.tmp,type=LS) -vv2diag(EF.matpart4[i,])
#  }
#
#  HH.mat <- matrix(0,nrow=td,ncol=n.1)
#
#  for (i in 1:td){
#   HH.mat[i,] <- matrix(LH[i,],nrow=1)%*%HHH[i,,]
#  }
```

`hr.trans` *Calculating average hazard ratio: Transformation Model*

Description

Calculating average hazard ratio for transformation model: without covariates

Usage

```
hr.trans(.list)
```

Arguments

`.list`	result list

Value

`hr`	hazard ratio
`S`	Survival function

Examples

```
## NOT RUN.
```

`hr.trans2` *Calculating average hazard ratio: Transformation Model*

Description

Calculating average hazard ratio for transformation model: without covariates

Usage

```
hr.trans2(.list)
```

Arguments

.list	result list

Value

hr	hazard ratio
S	Survival function

Examples

```
## NOT RUN.
```

---

hr.yp *Calculating average hazard ratio*

Description

Calculating average hazard ratio: without covariates

Usage

```
hr.yp(.list,type=dR)
```

Arguments

.list	result list
type	'dR' or 'dLambda'

Value

hr	hazard ratio
S	Survival function
R	R(t)

Examples

```
## NOT RUN.
```

`trans.model` *Transformation model: No co-variates*

Description

Transformation model: No co-variates

Usage

```
trans.model(y, z, d, par0 = c(0.1, 0.5), use.iter = TRUE, iter.K = 100,
  iter.tol = 1e-04, report = FALSE, trace = FALSE)
```

Arguments

`y`	observed survival time, min(Y,C)
`z`	0/1, control/trt
`d`	indicators I(y)
`par0`	initial value of beta and rho
`use.iter`	T/F, if TRUE use iteration method, else use multiroot(rootsolve) which is an N-R algorithm
`iter.K`	Max number of iterations. If #iterations > iter.K then stop
`iter.tol`	Tolerance used in iteration. If |f(x)|<tol then stop
`report`	T/F if TRUE, then print a simple checking report
`trace`	T/F if TRUE, then print out the trace report in BFGS

Value

list(beta,rho,lambda,log-likelihood)

Examples

```
set.seed(123456)
be=c(log(0.9),log(1.2))
aa=0;
tbe=exp(be);
n= 160
z= as.numeric(runif(n)>.5);
g1=exp(-be[1]*z);
g2=exp(-be[2]*z);
rx=g1 /g2 *(runif(n)^(-g2)-1);
x=log(1+rx);
cp=.1
ce=exp(cp/1+runif(n)/2);
y=ce
y[x<=ce]=x[x<=ce]
d=as.numeric(y==x);
trans.model(y,z,d)
```

`transf2.model` *Transformation model: No co-variates*

Description

Transformation model: No co-variates

Usage

```
transf2.model(Y, z, d, par.init = c(0.01, 0.01), trace = FALSE, tol
                = 1e-05, iter.K = 200, pse = FALSE, onestep = FALSE,
                gsl = TRUE)
```

Arguments

`Y`	observed survival time, min(Y,C)
`z`	0/1, control/trt
`d`	indicators I(y)
`par.init`	initial value of beta and rho
`trace`	T/F if TRUE, then print out the trace report in BFGS
`tol`	Tolerance used in iteration. If |f(x)|<tol then stop
`iter.K`	Max number of iterations. If #iterations > iter.K then stop
`pse`	Use PSE.
`onestep`	Use onesetp updatiing
`gsl`	Use GNU scientific library routine.

Value

list(beta,nu,DLambda,empirical-log-likelihood,data)

Examples

```
set.seed(123456)
be=c(log(0.9),log(1.2))
aa=0;
tbe=exp(be);
n= 160
z= as.numeric(runif(n)>.5);
g1=exp(-be[1]*z);
g2=exp(-be[2]*z);
rx=g1 /g2 *(runif(n)^(-g2)-1);
x=log(1+rx);
cp=.1
ce=exp(cp/1+runif(n)/2);
y=ce
y[x<=ce]=x[x<=ce]
d=as.numeric(y==x);
transf2.model(y,z,d)
```

triangleassign *C function to generate triangle matrix*

Description

C-function is used, without BLAS.

Usage

```
triangleassign(x,val=NA,type="L",diag=TRUE)
```

Arguments

x	Vector
val	Scalor
type	"L": Lower triangle matrix of val;"LS":1,2,3,0,2,3,0,0,3;"U": Upper triangle matrix of val;"US": similar to "LS"
diag	diag(mat) will used.

Value

a matrix will be returned.

Examples

```
triangleassign(1:3,type="LS")
```

vv2diag *C function*

Description

This funtion will genrate a Upper triangle matrix with each row a result of cumsumsurvR. It is a C-function without BLAS support.

Usage

```
vv2diag(x)
```

Arguments

x	A numeric vector

Value

Matrix

Examples

```
vv2diag(1:3)
```

yp.model *Yang Prentice model: No co-variates*

Description

Yang-Prentice model: No co-variates

Usage

```
yp.model(Y, z, d, beta.init = "psu", iter.K = 200, tol = 1e-06,
                 trace = TRUE, pse = TRUE, check = FALSE, usegsl =
                 FALSE)
```

Arguments

Y	observed survival time, min(Y,C)
z	0/1, control/trt
d	indicators I(y)
beta.init	initial value of beta1 and beta2. If beta.init="psu" use psudo-likelihood.
iter.K	Max number of iterations. If #iterations > iter.K then stop
tol	Tolerance used in iteration. If |f(x)|<tol then stop
trace	T/F if TRUE, then print out the trace report in BFGS and iteration step
check	Debug information.
pse	Use PSE.
usegsl	Use GNU scientific library routine.

Value

list(betas,dR,ell)

Examples

```
set.seed(123456)
be=c(log(0.9),log(1.2))
aa=0;
tbe=exp(be);
n= 160
z= as.numeric(runif(n)>.5);
g1=exp(-be[1]*z);
g2=exp(-be[2]*z);
rx=g1 /g2 *(runif(n)^(-g2)-1);
x=log(1+rx);
cp=.1
ce=exp(cp/1+runif(n)/2);
y=ce
y[x<=ce]=x[x<=ce]
d=as.numeric(y==x);
yp.model(y,z,d,trace=TRUE)
```

yp.ZWpse2 *Yang Prentice model: A demo code*

Description

This is function is only for benchmarking.

Usage

```
yp.ZWpse2(Y,Z,delta,W,beta.init=c(0,0),gsl=TRUE,trace=TRUE)
```

Arguments

Y	observed survival time, min(Y,C)
Z	0/1, control/trt
delta	indicators I(y)
W	Covariates
beta.init	initial value of beta1 and beta2. If beta.init="psu" use psudo-likelihood.
trace	T/F if TRUE, then print out the trace report in BFGS and iteration step
gsl	Use GNU scientific library routine.

Value

list(betas,dR,ell)

Examples

```
set.seed(123456)
be=c(log(0.9),log(1.2))
aa=0;
tbe=exp(be);
n= 160
z= as.numeric(runif(n)>.5);
g1=exp(-be[1]*z);
g2=exp(-be[2]*z);
rx=g1 /g2 *(runif(n)^(-g2)-1);
x=log(1+rx);
cp=.1
ce=exp(cp/1+runif(n)/2);
y=ce
y[x<=ce]=x[x<=ce]
d=as.numeric(y==x);
yp.model(y,z,d,trace=TRUE)
```

`yplamdaapproach.model` *Yang Prentice model (using Lambda(t)): No co-variates*

Description

Yang-Prentice model: No co-variates

Usage

```
yplamdaapproach.model(
Y,
z,
d,
beta.init=psu,
  iter.K=100,
tol=1e-6,
trace=TRUE,
pse=FALSE,
onestep=FALSE,
gsl=TRUE)
```

Arguments

`Y`	observed survival time, min(Y,C)
`z`	0/1, control/trt
`d`	indicators I(y)
`beta.init`	initial value of beta1 and beta2. If beta.init="psu" use psudo-likelihood.
`iter.K`	Max number of iterations. If #iterations > iter.K then stop
`tol`	Tolerance used in iteration. If |f(x)|<tol then stop
`pse`	Use PSE.
`trace`	T/F if TRUE, then print out the trace report in BFGS and iteration step
`onestep`	One-step estimation.
`gsl`	Use GNU scientific library routine.

Value

list(betas,dR,ell)

Examples

```
set.seed(123456)
be=c(log(0.9),log(1.2))
aa=0;
tbe=exp(be);
n= 160
z= as.numeric(runif(n)>.5);
g1=exp(-be[1]*z);
g2=exp(-be[2]*z);
rx=g1 /g2 *(runif(n)^(-g2)-1);
```

```
x=log(1+rx);
cp=.1
ce=exp(cp/1+runif(n)/2);
y=ce
y[x<=ce]=x[x<=ce]
d=as.numeric(y==x);
yp.model(y,z,d,trace=TRUE)
```

---

ypmodel.chk *Model check for Yang-Prentice model with co-variates*

Description

The model check function will calculate the following process and report the max absolute value of it.

$$\begin{aligned} T(\omega \le \omega_K, t = X_{t_u}) &= \sum_{i=1}^{n} I(i : \omega_i \le \omega_K) \int_0^{X_{t_u}} d\hat{M}_i(s) \\ &= \sum_{i:\omega_i \le \omega_K} \left\{ N_i(t_u) - \sum_{k=1}^{u} \frac{I(X_i \ge X_{t_k})}{r_{1i} + r_{2i} R(X_{t_k})} \Delta R(X_{t_k}) \right\} \end{aligned} \quad , \forall \delta_{t_u} = 1, K = 1, \ldots, t_d.$$

Usage

```
ypmodel.chk(
Y,W,Z,delta,simulationtime=1000,
beta.init=rep(0,3),gsl=TRUE,trace=FALSE,BFGS=TRUE,iter.K=500,tol=1e-6,iter.optim=500,
PLOT=TRUE,
cores=0
)
```

Arguments

Y	observed survival time, min(Y,C)
W	Covariate variable. Now it only allows 1 covariate, in future more variables may be allowed.
Z	0/1, control/trt
simulationtime	Number of simulation used to approximate Wn.
delta	indicators I(y)
beta.init	initial value of beta1 and beta2. By default, it is a vector of 0.
iter.K	Max number of iterations used in updating dR. If iterations > iter.K then stop.
iter.optim	Max number of iterations used in Newton types of optimaization.
tol	Tolerance used in iteration. If |f(x)|<tol then stop
trace	T/F if TRUE, then print out the trace report in BFGS and iteration step
gsl	gsl=TRUE, then GNU scientific library is used. Otherwise, a built-in R routine (optim) is used.
BFGS	BFGS= TRUE, then Broyden Fletcher Goldfarb Shanno algorithm will be used. Otherwise, Nelder and Mead algorithm will be used.
PLOT	If PLOT=TRUE, then R will generate a plot the simulated process. The option is useless if cores>0.
cores	An MPI option. If cores>0, then R will call a parallel function to approximate Wn. set.seed() will not effect this option unless you choose to use

Value

score	Score Process.
Hessian	Hessian matrix. (without minus sign, without inverse operation.)
re	return the output of NPMLE, which is the same as the utput of ypnpmle.model.

Examples

```
# generate data
set.seed(123456)
N=200
W     = matrix(runif(N),ncol=1)
Z     = as.numeric(runif(N)>.5);
b=c(log(.5),log(1.5))
u=exp(0.5*W)
g1=exp(-b[1]*Z);
g2=exp(-b[2]*Z);
rx=g1 /g2 *( runif(N)^(-g2/u)-1);
x=log(1+rx);
cp=1;tau=5;
ce=exp(cp/1+rnorm(N)/2);
ce[ce>tau]=tau
Y=ce
Y[x<=ce]=x[x<=ce]
delta=as.numeric(Y==x);
# result: not run
#ypmodel.chk(y1,matrix(W,ncol=1),Z,delta,simulationtime=1000,PLOT=TRUE) -> re
```

---

ypnpmle.model *Yang Prentice model with co-variates*

Description

Yang-Prentice model: With covariates

Usage

```
ypnpmle.model(
Y,
W,
z,
d,
beta.init=rep(0,3),
  iter.K=200,
iter.optim=200,
tol=1e-6,
trace=TRUE,
check=FALSE,
gsl=TRUE,
BFGS=TRUE
)
```

Arguments

Y	observed survival time, min(Y,C)
W	Covariate variable. Now it only allows 1 covariate, in future more variables may be allowed.
z	0/1, control/trt
d	indicators I(y)
beta.init	initial value of beta1 and beta2. By default, it is a vector of 0.
iter.K	Max number of iterations used in updating dR. If #iterations > iter.K then stop.
iter.optim	Max number of iterations used in Newton types of optimaization.
tol	Tolerance used in iteration. If |f(x)|<tol then stop
trace	T/F if TRUE, then print out the trace report in BFGS and iteration step
check	Debug output.
gsl	gsl=TRUE, then GNU scientific library is used. Otherwise, a built-in R routine (optim) is used.
BFGS	BFGS= TRUE, then Broyden Fletcher Goldfarb Shanno algorithm algorithm will be used. Otherwise, Nelder and Mead algorithm will be used.

Value

par	NPMLE for coefficients. The last two are corresponding to beta1 and beta2 in Y-P model.
dR	NPMLE for R(t).
data	Sorted dara.
ell	Log-empirical likelihood.

Examples

```
# generate data
set.seed(123456)
N=200
W     = matrix(runif(N),ncol=1)
Z     = as.numeric(runif(N)>.5);
b=c(log(.5),log(1.5))
u=exp(0.5*W)
g1=exp(-b[1]*Z);
g2=exp(-b[2]*Z);
rx=g1 /g2 *( runif(N)^(-g2/u)-1);
x=log(1+rx);
cp=1;tau=5;
ce=exp(cp/1+rnorm(N)/2);
ce[ce>tau]=tau
Y=ce
Y[x<=ce]=x[x<=ce]
delta=as.numeric(Y==x);
# result
re = ypnpmle.model(Y,W,Z,delta,beta.init=rep(0,3),gsl=TRUE,BFGS=TRUE,iter.K=100,tol=1e-6,iter.optim=100)
```

`ypnpmle.modelHessian` *Hessian Matrix for Yang-Prentice model with co-variates*

Description

Hessian matrix and scores for Yang-Prentice model: With covariates

Usage

```
ypnpmle.modelHessian(
Y,
W,
z,
d,
beta.init=rep(0,3),
  iter.K=200,
iter.optim=200,
tol=1e-6,
trace=TRUE,
check=FALSE,
gsl=TRUE,
BFGS=TRUE,
re= NA
)
```

Arguments

`Y`	observed survival time, min(Y,C)
`W`	Covariate variable. Now it only allows 1 covariate, in future more variables may be allowed.
`z`	0/1, control/trt
`d`	indicators I(y)
`beta.init`	initial value of beta1 and beta2. By default, it is a vector of 0.
`iter.K`	Max number of iterations used in updating dR. If #iterations > iter.K then stop.
`iter.optim`	Max number of iterations used in Newton types of optimaization.
`tol`	Tolerance used in iteration. If |f(x)|<tol then stop
`trace`	T/F if TRUE, then print out the trace report in BFGS and iteration step
`check`	Debug output.
`gsl`	gsl=TRUE, then GNU scientific library is used. Otherwise, a built-in R routine (optim) is used.
`BFGS`	BFGS= TRUE, then Broyden Fletcher Goldfarb Shanno algorithm will be used. Otherwise, Nelder and Mead algorithm will be used.
`re`	by default, ypnpmle.modelHessian will cal `ypnpmle.model` to calculate NPMLE. But one could specify the result. The given `re` must be a `list` with same entries as the output of `ypnpmle.model`.

Value

score	Score Process.
Hessian	Hessian matrix. (without minus sign, without inverse operation.)
re	return the output of NPMLE, which is the same as the utput of ypnpmle.model.

Examples

```
# generate data
set.seed(123456)
N=200
W     = matrix(runif(N),ncol=1)
Z     = as.numeric(runif(N)>.5);
b=c(log(.5),log(1.5))
u=exp(0.5*W)
g1=exp(-b[1]*Z);
g2=exp(-b[2]*Z);
rx=g1 /g2 *( runif(N)^(-g2/u)-1);
x=log(1+rx);
cp=1;tau=5;
ce=exp(cp/1+rnorm(N)/2);
ce[ce>tau]=tau
Y=ce
Y[x<=ce]=x[x<=ce]
delta=as.numeric(Y==x);
# result
re = ypnpmle.modelHessian(Y,W,Z,delta,beta.init=rep(0,3),gsl=TRUE,BFGS=TRUE,iter.K=100,tol=1e-6,iter.optim=
```

Index