Package 'ELYP'

August 7, 2015

Maintainer Mai Zhou <mai@ms.uky.edu>

Version 0.7-3

Depends R (>= 2.15.0), survival

Suggests emplik

Author Mai Zhou

Description

Empirical likelihood ratio tests for the Yang and Prentice (short/long term hazards ratio) models. Empirical likelihood tests within a Cox model, for parameters defined via both baseline hazard function and regression parameters.

Title Empirical Likelihood Analysis for the Cox Model and Yang-Prentice (2005) Model

License GPL (>= 2)

URL http://www.ms.uky.edu/~mai/EmpLik.html

NeedsCompilation yes

Repository CRAN

Date/Publication 2015-08-07 21:55:47

R topics documented:

`BJfindL2` *Find the Wilks Confidence Interval Lower Bound for Betafun from the 2 dimensional Buckley-James Empirical Likelihood Ratio Function*

Description

This function uses simple search to find the lower level (default 95%) 1 parameter Wilks confidence limits based on the Buckley-James empirical likelihood test function for two dimensional beta's. Betafun determines the 1 parameter we are finding the lower bound.

Usage

```
BJfindL2(NPmle, ConfInt, LLRfn, Betafun, dataMat, level=3.84)
```

Arguments

`NPmle`	a 2-d vector: the NPMLEs: beta1 hat and beta2 hat.
`ConfInt`	a vector of length 2. Approx. length of the 2 conf. intervals for beta1 and beta2.
`LLRfn`	a function that returns -2LLR value.
`Betafun`	a function that takes the input of 2 parameter values (beta1,beta2) and returns a parameter that we wish to find the confidence Interval lower Value.
`dataMat`	matrix of covariates
`level`	confidence level. Use chi-square(df=1), but calibration possible.

Details

Basically we repeatedly testing the value of the 2 parameters, finding the -2LLR values, until we find those Betafun which the -2 log likelihood Ratio value is equal to 3.84 (or other level, if set differently).

Value

A list with the following components:

`Lower` the lower confidence bound.

`minParameterNloglik`
Final values of the 2 parameters, and the log likelihood.

Author(s)

Mai Zhou

References

Zhou, M. and Li, G. (2006). Computing censored empirical likelihood ratio by EM algorithm. *JCGS*

Examples

```
## See the Rd file of BJfindU2 for example.
```

`BJfindU2` *Find the Wilks Confidence Interval Upper Bound for Betafun from the 2 dimensional Buckley-James Empirical Likelihood Ratio Function*

Description

This function uses simple search to find the upper level (default 95%) 1 parameter Wilks confidence limits based on the Buckley-James empirical likelihood test function for two dimensional beta's. The confidece interval is for the 1 parameter, determined by Betafun.

Usage

```
BJfindU2(NPmle, ConfInt, LLRfn, Betafun, dataMat, level=3.84)
```

Arguments

`NPmle`	a 2-d vector: the NPMLEs: beta1 hat and beta2 hat.
`ConfInt`	a vector of length 2. Approx. length of the 2 conf. intervals for beta1 and beta2. May use the SD from bj().
`LLRfn`	a function that returns the -2LLR.
`Betafun`	a function that takes the input of 2 parameter values (beta1, beta2) and returns a parameter that we wish to find its confidence Interval Lower Value.
`dataMat`	matrix of covariates
`level`	confidence level.

Details

Basically we repeatedly testing the value of the 2 parameters, until we find the max of Betafun, provided the -2 log likelihood value is <= 3.84 (or other level, if set differently).

Value

A list with the following components:

`Upper` the upper confidence bound.

`maxParameterNloglik`
Final values of the 2 parameters, and the log likelihood.

Author(s)

Mai Zhou

References

Zhou, M. (2005). Computing censored empirical likelihood ratio by EM algorithm. *JCGS*

Examples

```
# The Stanford Heart Transplant Data: with 152 cases.
# Needs bjtest( ) function from emplik package to run.

## BJloglik <- function(para, dataMat) {
##            yvec <- dataMat[,1]
##            dvec <- dataMat[,2]
##            x <- dataMat[,3:4]
## temp <- bjtest(y=log10(yvec), d=dvec, x=x, beta=para)
## return(temp)
## }

## BJ2fun <- function (b1, b2) {
##    return(b2)
## }

## We first use bj() from Design library to find NPmle and
## the conservative SD of beta1 and beta2

## BJfindU2(NPmle=c(3.52696077,-0.01989555),
##      ConfInt=c(0.3,0.0066), LLRfn=BJloglik,
##      Betafun=BJ2fun,
##      dataMat=cbind(stanford5$time, stanford5$status,
##      rep(1,152),stanford5$age))
##
# This will take (~ 1 min.) to run.
```

CoxEL *Compute Empirical Likelihood and Partial Likelihood of Cox model for Testing the beta and Baseline Jointly.*

Description

This function compute empirical likelihood and partial likelihood for the purpose of testing (jointly) the beta (the regression parameter) and a baseline hazard feature, which is defined by lam and fun.

Usage

```
CoxEL(y, d, Z, beta, lam, fun)
```

Arguments

y	a vector containing the survival times
d	censoring status: 1 - uncensored; 0 - censored.
Z	a matrix of covariates, size nxk; Z=(Z_1i, ..., Z_ki)
beta	a vector =(beta1, ... betak)
lam	a scalar, used in the constraint of baseline: int f(t)dH(t)= Mulam. It is sometime called the tilting amount.
fun	a function in the int f(t)dH(t). Together with lam, it defines the feature of the baseline hazard H(t).

Details

This function compute the likelihood when we impose both restriction on beta and on baseline. The restriction on beta is simply by setting beta equal to the given value in the CoxEL input. The restriction on the baseline is via a finite dimensional feature.

lam controls the amount of tilting for the baseline, in the direction of the feature defined by int f(t) dH(t). When lam = 0 means no tilting.

Value

It returns a list containing: (1)logEmpLik: log empirical likelihood value; (2)logPlik: log partial likelihood value; (3)Hazw: the constrained baseline estimator (the jumps); (4)mu: the value of the constraint, int f(t)dH(t)= Mulam i.e., the feature value.

Author(s)

Mai Zhou

References

Zhou, M. (2016). Empirical Likelihood Method in Survival Analysis. Zhou, M. (2005). Cox model with restriction on the baseline hazard. *Tech Report, Univ. of Kentucky, Dept of Statistics*

Examples

```
## censored regression with one right censored observation.
## we check the estimation equation, with the MLE inside myfun7.
y <- c(3, 5.3, 6.4, 9.1, 14.1, 15.4, 18.1, 15.3, 14, 5.8, 7.3, 14.4)
x <- c(1, 1.5, 2,   3,   4,    5,    6,    5,    4,  1,   2,   4.5)
d <- c(1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0)
```

`CoxFindL2` *Find the Wilks Confidence Interval Lower Bound for Efun based on the Empirical Likelihood Ratio Function CoxEL*

Description

This function uses simple search to find the lower level (default 95%) Wilks confidence limits based on the CoxEL( ) likelihood function.

Usage

```
CoxFindL2(BetaMLE, StepSize, Hfun, Efun, y, d, Z, level=3.84)
```

Arguments

`BetaMLE`	a scalar: the NPMLEs: beta1 hat.
`StepSize`	a vector of length 2. Approximate length of the 2 confidence intervals: beta1, and lambda.
`Hfun`	a function that defines the baseline feature: int f(t)dH(t)= mu or sometimes called Mulam.
`Efun`	a function that takes the input of 2 parameter values (beta1 and Mulam) and returns a parameter that we wish to find the confidence interval lower value. The two input variables must be named beta and theta.
`y`	the censored survival times.
`d`	vector of 0, and 1, censoring indicator
`Z`	matrix of covariates
`level`	confidence level. Using chi-square(df=1), but calibration possible.

Details

Basically we repeatedly testing the value of the parameter, until we find those which the -2 log likelihood value is equal to 3.84 (or other level, if set differently).

Value

A list with the following components:

`Lower`	the lower confidence bound.
`maxParameterNloglik`	Final values of the 3 parameters, and the log likelihood.

Author(s)

Mai Zhou

References

Zhou, M. (2002). Computing censored empirical likelihood ratio by EM algorithm. *JCGS*

Examples

```
## We find 95% lower limit of theta= \Lambda_0(300) exp(\beta)
## where \Lambda and \beta are inside a Cox model.
## First we define a function (Hfun) = I[t <= 300], so that
## the baseline feature is \Lambda_0(300). The second function
## we need to define is (Efun) = what we called theta above.

data(smallcell)
myHfun <- function(t){as.numeric(t <= 300)}
myEfun <- function(beta, theta){theta*exp(beta)}

myy <- smallcell$survival
myd <- smallcell$indicator
myZ <- smallcell$arm

CoxFindL2(BetaMLE=0.5337653, StepSize=c(0.1, 3),
          Hfun=myHfun, Efun=myEfun, y=myy, d=myd, Z=myZ)
```

`CoxFindL3` *Find the Wilks Confidence Interval Upper Bound from the Given Empirical Likelihood Ratio Function*

Description

This program uses simple search to find the Lower 95% (or other)Wilks confidence limits based on the log likelihood function (CoxEL) supplied.

Usage

```
CoxFindL3(BetaMLE, StepSize, Hfun, Efun, y, d, Z, level=3.84)
```

Arguments

`BetaMLE`	a vector containing the two NPMLEs: beta1 hat and beta2 hat.
`StepSize`	a vector of length 3. Approximate length of the 3 confidence intervals: beta1, beta2 and lambda.
`Hfun`	a function that used to defines the baseline feature, mu.

Efun	a function that takes the input of 3 parameter values (beta1, beta2 and Mulam) and returns a parameter that we wish to find the confidence Interval Lower Value. The input variables must be called beta and theta. beta: in the form of a 2-d vector (i.e., the beta1, beta2) and theta: (=Mulam)
y	a vector of censored survival time.
d	a vector of 0 and 1, censoring indicator
Z	covariates of the Cox model.
level	confidence level

Details

Basically we repeatedly testing the value of the parameter (defined by Efun), try to go to lower and lower values of the parameter until we find those which the -2 log likelihood value is equal to 3.84 (or other level, if set differently).

Value

A list with the following components:

Lower	the lower confidence bound.
maxParameterNloglik	Final values of the 4 parameters, and the log likelihood.

Author(s)

Mai Zhou

References

Zhou, M. (2002). Computing censored empirical likelihood ratio by EM algorithm. *JCGS*

Examples

```
## Here Mulam is the value of int g(t) d H(t) = Mulam
## For example g(t) = I[ t <= 2.0 ]; look inside myLLfun().

data(GastricCancer)

# The following will take about 0.5 min to run.
# findU3(NPmle=c(1.816674, -1.002082), ConfInt=c(1.2, 0.5, 10),
#         LogLikfn=myLLfun, Pfun=Pfun, dataMat=GastricCancer)
```

CoxFindU2 *Find the Wilks Confidence Interval Upper Bound for Efun from the Empirical Likelihood Ratio Function CoxEL( ).*

Description

This function uses simple search to find the upper 95% Wilks confidence limits based on the log likelihood function supplied. This is a sister function to CoxFindL2().

Usage

```
CoxFindU2(BetaMLE, StepSize, Hfun, Efun, y, d, Z, level=3.84)
```

Arguments

BetaMLE	a scalar: the NPMLE beta1 hat.
StepSize	a vector of length 2. Approximate length of the 2 confidence intervals: beta1, and lambda. It is the initial search step size.
Hfun	a function that defines the baseline feature: mu=int f(t) dH(t).
Efun	a function that takes the input of 2 parameter values (beta1, and Mulam) and returns a parameter that we wish to find the confidence Interval Upper Value. The two input variables must be called beta and theta.
y	a vector of censored survival times.
d	a vector of 0 and 1, censoring indicator.
Z	covariates for the Cox model
level	Confidence Level. Use chi-square(df=1), but calibration possible.

Details

Basically we repeatedly testing the value of the parameter, until we find those which the -2 log likelihood value is equal to 3.84 (or other level, if set differently).

Value

A list with the following components:

Upper	the upper confidence bound.
maxParameterNloglik	Final values of the 4 parameters, and the log likelihood.

Author(s)

Mai Zhou

References

Zhou, M. (2002). Computing censored empirical likelihood ratio by EM algorithm. *JCGS*

Examples

```
## See example in CoxFindL2.
## Here Mulam is the value of int g(t) d H(t) = Mulam
## For example g(t) = I[ t <= 2.0 ]; look inside myLLfun().
```

`CoxFindU3` *Find the Wilks Confidence Interval Upper Bound from the Given Empirical Likelihood Ratio Function*

Description

This program uses simple search to find the upper 95% Wilks confidence limits based on the log likelihood function supplied.

Usage

```
CoxFindU3(BetaMLE, StepSize, Hfun, Efun, y, d, Z, level=3.84)
```

Arguments

`BetaMLE`	a vector containing the two NPMLEs: beta1 hat and beta2 hat.
`StepSize`	a vector of length 3. Approximate length of the 3 confidence intervals: beta1, beta2 and lambda.
`Hfun`	a function that defines the baseline feature: mu.
`Efun`	a function that takes the input of 3 parameter values (beta1, beta2 and Mulam) and returns a parameter that we wish to find the confidence Interval Upper Value. The input variables must be in the form: beta = c(beta1, beta2) and theta = Mulam.
`y`	a matrix.
`d`	a vector of 0 and 1
`Z`	covariates
`level`	confidence level using chi-square(df=1), but calibration possible.

Details

Basically we repeatedly testing the value of the parameter, until we find those which the -2 log likelihood value is equal to 3.84 (or other level, if set differently).

Value

A list with the following components:

Upper	the upper confidence bound.
maxParameterNloglik	Final values of the 4 parameters (beta1, beta2, Mulam, lam), and the log likelihood.

Author(s)

Mai Zhou

References

Zhou, M. (2002). Computing censored empirical likelihood ratio by EM algorithm. *JCGS*

Examples

```
## Here Mulam is the value of int g(t) d H(t) = Mulam
## For example g(t) = I[ t <= 2.0 ]; look inside myLLfun().

data(GastricCancer)

# The following will take about 0.5 min to run.
# findU3(NPmle=c(1.816674, -1.002082), ConfInt=c(1.2, 0.5, 10),
#        LogLikfn=myLLfun, Pfun=Pfun, dataMat=GastricCancer)
```

ELrange *Find the Ractangular parameter region where EL is Only 4 below the Maximum Value.*

Description

This function compute the hazard ratio, given beta1 beta2 a X and Mulam = int g(t) dH(t).

Usage

```
ELrange(mle, loglik, step, DataMat)
```

Arguments

mle	The NPMLE of the parameters value.
loglik	a function. Takes 2 inputs: mle and DataMat. output one scalar = loglik value.
step	a vector, same length as mle. The initial search step.
DataMat	The data matrix, to be used by loglik( ).

Details

Say something.

Value

A list with Step [a vector] and TempV [a matrix]

Author(s)

Mai Zhou

References

Zhou, M. (2002). Computing censored empirical likelihood ratio by EM algorithm. *Tech Report, Univ. of Kentucky, Dept of Statistics*

Examples

```
## censored regression with one right censored observation.
## we check the estimation equation, with the MLE inside myfun7.
y <- c(3, 5.3, 6.4, 9.1, 14.1, 15.4, 18.1, 15.3, 14, 5.8, 7.3, 14.4)
x <- c(1, 1.5, 2,   3,   4,    5,    6,    5,    4,  1,   2,   4.5)
d <- c(1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0)
```

---

findL2d — *Find the Wilks Confidence Interval Lower Bound from the Given 2-d Empirical Likelihood Ratio Function*

Description

This function is a sister function to findU2d( ). It uses simple search algorithm to find the lower 95% Wilks confidence limits based on the log likelihood function supplied. The likelihood have two parameters: beta1, beta2 and the the confidence interval is for a 1-d parameter defined by Pfun(beta1, beta2).

Usage

```
findL2d(NPmle, ConfInt, LogLikfn, Pfun, dataMat, level=3.84)
```

Arguments

NPmle	a vector containing the two NPMLE: beta1 hat and beta2 hat.
ConfInt	a vector of length 2. These are APPROXIMATE length of confidence intervals, as initial guess.
LogLikfn	a function that takes input of beta=(beta1, beta2) and dataMat, and output the log likelihood value.

Pfun	A function of 2 variables: beta1 and beta2. Must be able to take a vector input. Example: Pfun(x1, x2)= x1.
dataMat	a matrix of data. for the function LogLikfn.
level	Confidence level. Default to 3.84 (95 percent).

Details

Basically we repeatedly testing the value of the parameter, until we find those which the -2 log likelihood value is equal to 3.84 (or other level, if set differently).

Value

A list with the following components:

Lower	the lower confidence bound for Pfun.
minParameterNloglik	Final values of the 2 parameters, and the log likelihood.

Author(s)

Mai Zhou

References

Zhou, M. (2002). Computing censored empirical likelihood ratio by EM algorithm. *JCGS*

Examples

```
## example with tied observations
x <- c(1, 1.5, 2, 3, 4, 5, 6, 5, 4, 1, 2, 4.5)
d <- c(1,   1, 0, 1, 0, 1, 1, 1, 1, 0, 0,   1)
```

---

findL3 *Find the Wilks Confidence Interval Lower Bound from the Given Empirical Likelihood Ratio Function*

Description

This program is the sister program to the findU3( ). It uses simple search to find the lower 95% Wilks confidence limits based on the log likelihood function supplied.

Usage

```
findL3(NPmle, ConfInt, LogLikfn, Pfun, level=3.84, dataMat)
```

Arguments

NPmle	a vector containing the two NPMLE: beta1 hat and beta2 hat.
ConfInt	a vector of length 3.
LogLikfn	a function that compute the loglikelihood. Typically this has three parameters: beta1, beta2 and lam, in a Yang-Prentice model context.
Pfun	a function that takes the input of 3 parameter values (beta1,beta2 and Mulam) and returns a parameter that we wish to find the confidence Interval of (here only the Lower Value).
level	confidence level. Default to 3.84 for 95 percent.
dataMat	a matrix.

Details

The empirical likelihood for Y-P model has parameters: beta1, beta2 and a baseline. The baseline is converted to a 1-d parameter feature via Hfun, and then amount controled by lam.

Basically we repeatedly testing the value of the parameter, until we find those which the -2 log likelihood value is equal to 3.84 (or other level, if set differently).

Value

A list with the following components:

Lower	the lower confidence bound.
minParameterNloglik	Final values of the 4 parameters, and the log likelihood.

Author(s)

Mai Zhou

References

Zhou, M. (2002). Computing censored empirical likelihood ratio by EM algorithm. *JCGS*

Examples

```
## Here Mulam is the value of int g(t) d H(t) = Mulam
## For example g(t) = I[ t <= 2.0 ]; look inside myLLfun().

Pfun <- function(b1, b2, Mulam) {
alpha <- exp(-Mulam)
TrtCon <- 1/(alpha*exp(-b1) + (1-alpha)*exp(-b2))
return(TrtCon)
}

data(GastricCancer)

# The following will take about 10 sec. to run on i7 CPU.
# findL3(NPmle=c(1.816674, -1.002082), ConfInt=c(1.2, 0.5, 10),
```

```
#          LogLikfn=myLLfun, Pfun=Pfun, dataMat=GastricCancer)
```

---

findL4 *Find the Wilks Confidence Interval Lower Bound from the Given Empirical Likelihood Ratio Function*

Description

This program uses simple search to find the upper 95% Wilks confidence limits based on the log likelihood function supplied. Caution: it takes about 1 min. to run on a data set of 90 obs. [GastricCancer]

Usage

```
findL4(NPmle, ConfInt, LogLikfn2, Pfun, dataMat, level=3.84)
```

Arguments

NPmle	a vector containing the three NPMLEs: beta1 hat, beta2 hat and alpha hat. from a Y-P model.
ConfInt	a vector of length 4. Approx. length of the 4 conf. intervals: beta1, beta2, alpha and lambda.
LogLikfn2	a function that compute the empirical likelihood of the Y-P model. given the parameters beta1, beta2, alpha, and lam.
Pfun	a function that takes the input of 3 parameter values (beta1,beta2 and Mulam) and returns a parameter that we wish to find the confidence Interval Lower Value.
dataMat	a matrix.
level	The significance level. Default to 3.84; corresponds to a 95 percent confidence interval.

Details

Basically we repeatedly testing the value of the parameter, until we find those which the -2 log likelihood value is equal to 3.84 (or other level, if set differently).

Value

A list with the following components:

Lower	the lower confidence bound.
minParameterNloglik	Final values of the 4 parameters, and the log likelihood.

Author(s)

Mai Zhou

References

Zhou, M. (2002). Computing censored empirical likelihood ratio by EM algorithm. *JCGS*

Examples

```
## Here Mulam is the value of int g(t) d H(t) = Mulam
## For example g(t) = I[ t <= 2.0 ]; look inside myLLfun().

data(GastricCancer)

# The following will take about 0.5 min to run.
# findU3(NPmle=c(1.816674, -1.002082), ConfInt=c(1.2, 0.5, 10),
#           LogLikfn=myLLfun, Pfun=Pfun, dataMat=GastricCancer)
```

---

findU2d *Find the Wilks Confidence Interval Upper Bound from the Given 2-d Empirical Likelihood Ratio Function*

Description

This program uses simple search algorithm to find the upper 95% Wilks confidence limits based on the log likelihood function supplied. The likelihood have two parameters beta1, beta2 and the the confidence interval is for a 1-d parameter =Pfun(beta1,beta2).

Usage

```
findU2d(NPmle, ConfInt, LogLikfn, Pfun, dataMat, level=3.84)
```

Arguments

NPmle	a vector containing the two NPMLE: beta1 hat and beta2 hat.
ConfInt	a vector of length 2. These are APPROXIMATE length of confidence intervals, as initial guess.
LogLikfn	a function that takes the input of beta and dataMay and output the logliklihood value.
Pfun	A function of 2 variables: beta1 and beta2. Must be able to take vector input. Example: Pfun(x1, x2)= x1.
dataMat	a matrix of data. for the function LogLikfn.
level	Confidence level. Default to 3.84 (95 percent).

Details

Basically we repeatedly testing the value of the parameter, until we find those which the -2 log likelihood value is equal to 3.84 (or other level, if set differently).

Value

A list with the following components:

`Upper` the upper confidence bound for Pfun.

`maxParameterNloglik`
Final values of the 2 parameters, and the log likelihood.

Author(s)

Mai Zhou

References

Zhou, M. (2002). Computing censored empirical likelihood ratio by EM algorithm. *JCGS*

Examples

```
## example with tied observations
x <- c(1, 1.5, 2, 3, 4, 5, 6, 5, 4, 1, 2, 4.5)
d <- c(1,   1, 0, 1, 0, 1, 1, 1, 1, 0, 0,   1)
```

`findU3` *Find the Wilks Confidence Interval Upper Bound from the Given Empirical Likelihood Ratio Function*

Description

This program uses simple search to find the upper 95% Wilks confidence limits based on the log likelihood function supplied.

Usage

```
findU3(NPmle, ConfInt, LogLikfn, Pfun, level=3.84, dataMat)
```

Arguments

`NPmle`	a vector containing the two NPMLEs: beta1 hat and beta2 hat.
`ConfInt`	a vector of length 3. Approximate length of the 3 confidence intervals: beta1, beta2 and lambda. They are used as initial search steps.
`LogLikfn`	a function that takes input of beta1, beta2 lam, dataMat, and output the log likelihood value.
`Pfun`	a function that takes the input of 3 parameter values (beta1,beta2 and Mulam) and returns a parameter that we wish to find the confidence Interval Upper Value.
`level`	confidence level, default to 3.84 for 95 percent.
`dataMat`	a matrix. for the loglik computation.

Details

Basically we repeatedly testing the value of the parameter, until we find those which the -2 log likelihood value is equal to 3.84 (or other level, if set differently).

Value

A list with the following components:

`Upper` the upper confidence bound.

`maxParameterNloglik`
Final values of the 4 parameters, and the log likelihood.

Author(s)

Mai Zhou

References

Zhou, M. (2002). Computing censored empirical likelihood ratio by EM algorithm. *JCGS*

Examples

```
## Here Mulam is the value of int g(t) d H(t) = Mulam
## For example g(t) = I[ t <= 2.0 ]; look inside myLLfun().

data(GastricCancer)

# The following will take about 10 sec. to run on an i7-4690 CPU.
# findU3(NPmle=c(1.816674, -1.002082), ConfInt=c(1.2, 0.5, 10),
#          LogLikfn=myLLfun, Pfun=Pfun, dataMat=GastricCancer)
```

`findU32` *Find the Wilks Confidence Interval Upper Bound from the Given Empirical Likelihood Ratio Function*

Description

This program uses simple search to find the upper 95% Wilks confidence limits based on the log likelihood function supplied.

Usage

```
findU32(NPmle, ConfInt, LogLikfn, Pfun, dataMat, level=3.84)
```

Arguments

NPmle	a vector containing the two NPMLEs: beta1 hat and beta2 hat.
ConfInt	a vector of length 3. Approximate length of the 3 confidence intervals: beta1, beta2 and lambda. They are used as initial search step size.
LogLikfn	a function that computes the loglikelihood.
Pfun	a function that takes the input of 3 parameter values (beta1, beta2 and Mulam) and returns a parameter that we wish to find the confidence Interval Upper Value.
dataMat	a matrix.
level	Significance level. Default to 3.84 (95 percent).

Details

This is a sister function to findU3( ). A bit faster. Use about half the time compared to findU3().

Basically we repeatedly testing the value of the parameter, until we find those which the -2 log likelihood value is equal to 3.84 (or other level, if set differently).

Value

A list with the following components:

Upper	the upper confidence bound.
maxParameterNloglik	Final values of the 4 parameters, and the log likelihood.

Author(s)

Mai Zhou

References

Zhou, M. (2002). Computing censored empirical likelihood ratio by EM algorithm. *JCGS*

Examples

```
## Here Mulam is the value of int g(t) d H(t) = Mulam
## For example g(t) = I[ t <= 2.0 ]; look inside myLLfun().

data(GastricCancer)

# The following will take about 0.5 min to run.
# findU32(NPmle=c(1.816674, -1.002082), ConfInt=c(1.2, 0.5, 10),
#            LogLikfn=myLLfun, Pfun=Pfun, dataMat=GastricCancer)
```

`findU4` *Find the Wilks Confidence Interval Upper Bound from the Given Empirical Likelihood Ratio Function*

Description

This program uses simple search to find the upper 95% Wilks confidence limits based on the log likelihood function supplied. Caution: it take about 3 min. to run on a data set of 90 obs. [GastricCancer]

Usage

```
findU4(NPmle, ConfInt, LogLikfn2, Pfun, dataMat, level=3.84)
```

Arguments

`NPmle`	a vector containing the three NPMLEs: beta1 hat, beta2 hat and alpha hat.
`ConfInt`	a vector of length 4. Approximate length of the 4 confidence intervals: beta1, beta2, alpha and lambda. They are the initial search step.
`LogLikfn2`	a function that computes the loglikelihood.
`Pfun`	a function that takes the input of 3 parameter values (beta1, beta2 and Mulam) and returns a parameter that we wish to find the confidence Interval Upper Value.
`dataMat`	a matrix.
`level`	The significance level. Default to 3.84.

Details

Basically we repeatedly testing the value of the parameter, until we find those which the -2 log likelihood value is equal to 3.84 (or other level, if set differently).

Value

A list with the following components:

`Upper`	the upper confidence bound.
`maxParameterNloglik`	Final values of the 4 parameters, and the log likelihood.

Author(s)

Mai Zhou

References

Zhou, M. (2002). Computing censored empirical likelihood ratio by EM algorithm. *JCGS*

Examples

```
## Here Mulam is the value of int g(t) d H(t) = Mulam
## For example g(t) = I[ t <= 2.0 ]; look inside myLLfun().

data(GastricCancer)

# The following will take about 0.5 min to run.
# findU3(NPmle=c(1.816674, -1.002082), ConfInt=c(1.2, 0.5, 10),
#           LogLikfn=myLLfun, Pfun=Pfun, dataMat=GastricCancer)
```

`fitYP3` *Compute Baseline Hazard for the Given Data, Given Parameters: beta1, beta2, lam, and fun. Also, Given the Baseline, Compute the empirical likelihood value.*

Description

This function fits the baseline for given beta1 beta2 and lam; and then compute the empirical likelihood.

Usage

```
fitYP3(Y, d, Z, beta1, beta2, lam, fun)
```

Arguments

`Y`	a vector containing the observed survival times.
`d`	a vector containing the censoring indicators, 1-uncensored; 0-right censored.
`Z`	a matrix of covariates (Xi and Zi)...
`beta1`	a vector = (alpha, beta1).
`beta2`	a vector = (alpha, beta2).
`lam`	a scalar. tilting parameter for the baseline. When lam=0, then there is no tilting.
`fun`	a function. It determine what feature of the baseline lam tilts.

Details

This function computes the log empirical likelihood. The parameters are given: beta1, beta2 and lam.

What baseline feature the lam corresponds to is determined by the fun(t), as in int f(t) dH(t). This integral value for the lam is also in the output as Mulam.

Value

A list with the following components:

LogEmpLik	this is actually the log empirical likelihood value.
Mulam	The value of int f(t) d H(t) for corresponding lam. This is also called a baseline feature.
BaseHazw	The baseline hazard jumps.

Author(s)

Mai Zhou

References

Zhou, M. (2002). Computing censored empirical likelihood ratio by EM algorithm. *Tech Report, Univ. of Kentucky, Dept of Statistics*

Examples

```
## censored regression with one right censored observation.
## we check the estimation equation, with the MLE inside myfun7.
y <- c(3, 5.3, 6.4, 9.1, 14.1, 15.4, 18.1, 15.3, 14, 5.8, 7.3, 14.4)
x <- c(1, 1.5, 2,   3,   4,    5,    6,    5,    4,  1,   2,   4.5)
d <- c(1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0)
```

---

fitYP4 *Compute Alpha and Baseline Hazard for the Given Data, Given Parameters beta1, beta2. Also, compute the empirical likelihood value.*

Description

This function finds the NPMLE of alpha and baseline, for the given beta1 and beta2. and then compute the empirical likelihood.

Usage

```
fitYP4(Y, d, Z, beta1=1, beta2=-1, maxiter=60)
```

Arguments

Y	a vector containing the observed survival times.
d	a vector containing the censoring indicators, 1-uncensored; 0-right censored.
Z	a vector of ...
beta1	a scalar. short term
beta2	a scalar. long term
maxiter	an integer.

Details

Difference to the function `fitYP3`: there is no constraint on the baseline. So, there is no lam input.

On the other hand, it try to find the NPMLE of alpha, via cox model iteration. So, it will output alpha hat.

Value

A list with the following components (may be I should also return the baseline Surv?):

`EmpLik`	this is actually the log empirical likelihood value.
`BaselineH`	The baseline hazard estimate.
`alpha`	The regression coefficient estimate, that is proportional hazard.

Author(s)

Mai Zhou

References

Zhou, M. (2002). Computing censored empirical likelihood ratio by EM algorithm. *Tech Report, Univ. of Kentucky, Dept of Statistics*

Examples

```
## censored regression with one right censored observation.
## we check the estimation equation, with the MLE inside myfun7.
y <- c(3, 5.3, 6.4, 9.1, 14.1, 15.4, 18.1, 15.3, 14, 5.8, 7.3, 14.4)
x <- c(1, 1.5, 2,   3,   4,    5,    6,    5,    4,  1,   2,   4.5)
d <- c(1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0)
```

---

fitYP41 *Compute the Baseline Hazard for the Given Data, given Parameters beta1, beta2. Also, compute the empirical likelihood value.*

Description

This function finds the NPMLE of baseline, for the given beta1 and beta2, and then compute the empirical likelihood. The model used is the simple YP model, without alpha.

Usage

```
fitYP41(Y, d, Z, beta1=1, beta2=-1, maxiter=60)
```

Arguments

Y	a vector containing the observed survival times.
d	a vector containing the censoring indicators, 1-uncensored; 0-right censored.
Z	a vector of covariates ...
beta1	a scalar. short term
beta2	a scalar. long term
maxiter	an integer.

Details

Similar to the function `fitYP4` except using a simple YP model (without alpha).

Value

A list with the following components (may be I should also return the baseline?):

EmpLik	this is actually the log empirical likelihood value.
BaselineH	The baseline hazard estimate.

Author(s)

Mai Zhou

References

Zhou, M. (2002). Computing censored empirical likelihood ratio by EM algorithm. *Tech Report, Univ. of Kentucky, Dept of Statistics*

Examples

```
## find NPMLE of beta1 and beta2 in the simple YP model.
data(GastricCancer)
optim(par=c(1.5,-1.5), fn= myffitYP41,
                      myY=GastricCancer[1,],
   myd=GastricCancer[2,],
   myZ=GastricCancer[4,])
```

GastricCancer *Gastric Cancer Data*

Description

There are 90 observations on 4 variables. Times is the survival times in years. Status is the censoring status. ax is the covariate that satisfy proportional hazards assumptions. zo is the covariate that indicating the two treatments, which we suppose follow the Y-P model. For more details please see the reference below.

Data are from Ying, Z., Jung, SH, and Wei, LJ (1995). Median regression analysis with censored data. Journal of the American Statistical Association, 90, 178-184.

Usage

```
data(GastricCancer)
```

Format

A data frame containing 90 observations on 4 variables:

[,1] "times"
[,2] "status"
[,3] "ax"
[,4] "zo"

References

Ying, Z., Jung, SH, and Wei, LJ (1995). Median regression analysis with censored data. Journal of the American Statistical Association, 90, 178-184.

myLLfun *Compute Baseline Hazard for the Given Data and Parameters beta1, beta2, lam. Also Compute the empirical likelihood value.*

Description

This function is similar to `fitYP3`. Just streamline input and output.

Usage

```
myLLfun(mle, dataMat, fun)
```

Arguments

mle	a vector of length 3, containing the parameter values: beta1, beta2 and lam. They do not have to be the MLE.
dataMat	a matrix of 4 by n. But the 4th row do not matter, since alpha=0 here always.
fun	a function, used in the definition of int f(t)dH(t)= Mulam.

Details

We assume a Y-P model. and with the given parameters (in the input mle) we compute the baseline hazard and compute the (parameter constrained) empirical likelihood value.

Value

A list with the following components:

Mulam	The value of int f(t) d H(t) for corresponding lam. Notice lam, beta1, beta2 determine the baseline H(t).
Loglik	The log empirical likelihood.

Author(s)

Mai Zhou

References

Zhou, M. (2002). Computing censored empirical likelihood ratio by EM algorithm. *Tech Report, Univ. of Kentucky, Dept of Statistics*

Examples

```
## censored regression with one right censored observation.
## we check the estimation equation, with the MLE inside myfun7.
y <- c(3, 5.3, 6.4, 9.1, 14.1, 15.4, 18.1, 15.3, 14, 5.8, 7.3, 14.4)
x <- c(1, 1.5, 2,   3,   4,    5,    6,    5,    4,  1,   2,   4.5)
d <- c(1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0)
```

---

myLLfun2 *Compute Baseline Hazard for the Given Data and Parameters beta1, beta2, alpha, lam. Also Compute the empirical likelihood value.*

Description

This function is similar to `fitYP3`. Just streamline input and output.

Usage

```
myLLfun2(mle, dataMat, fun)
```

Arguments

mle	a vector of length 4, containing the parameter values beta1, beta2, alpha, and lam. They do not have to be MLE.
dataMat	a matrix of 4 by n. They are (Y d X Z).
fun	a function, used in define the int f(t)dH(t)= Mulam.

Details

We assume a Y-P model. Say something.

Value

A list with the following components:

Mulam	The value of int f(t) d H(t) for corresponding lam. Notice lam, beta1, beta2 determine the baseline H(t).
Loglik	The log empirical likelihood.

Author(s)

Mai Zhou

References

Zhou, M. (2002). Computing censored empirical likelihood ratio by EM algorithm. *Tech Report, Univ. of Kentucky, Dept of Statistics*

Examples

```
## censored regression with one right censored observation.
## we check the estimation equation, with the MLE inside myfun7.
y <- c(3, 5.3, 6.4, 9.1, 14.1, 15.4, 18.1, 15.3, 14, 5.8, 7.3, 14.4)
x <- c(1, 1.5, 2,   3,   4,    5,    6,    5,    4,  1,   2,   4.5)
d <- c(1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0)
```

Pfun *The Hazard Ratio in YP Model as a Function of beta1 beta2 and Mulam.*

Description

This function compute the hazard ratio, given beta1 beta2 and Mulam = int g(t) dH(t) .

Usage

```
Pfun(b1, b2, Mulam)
```

Arguments

b1	a scalar, the parameter value.
b2	a scalar, parameter.
Mulam	It is int f(t)dH(t)= Mulam.

Details

Say something.

Value

A scalar which is the hazard ratio.

Author(s)

Mai Zhou

References

Zhou, M. (2002). Computing censored empirical likelihood ratio by EM algorithm. *Tech Report, Univ. of Kentucky, Dept of Statistics*

Examples

```
## censored regression with one right censored observation.
## we check the estimation equation, with the MLE inside myfun7.
y <- c(3, 5.3, 6.4, 9.1, 14.1, 15.4, 18.1, 15.3, 14, 5.8, 7.3, 14.4)
x <- c(1, 1.5, 2,   3,   4,    5,    6,    5,    4,  1,   2,   4.5)
d <- c(1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0)
```

Pfun2 — *The Hazard Ratio in YP Model as a Function of beta1, beta2, a, X, and Mulam.*

Description

This function compute the hazard ratio based on a Yang-Prentice model, given beta1, beta2, a, X and Mulam = int g(t) dH(t).

Usage

```
Pfun2(b1, b2, a, X, Mulam)
```

Arguments

b1	parameter value. = short term hazard ratio
b2	parameter: long term hazard ratio.
a	parameter
X	covariate
Mulam	it is int f(t)dH(t)= Mulam.

Details

The flexibility also rest on the definition of Mulam: it can using any f(t) function. If we use indicator I[t <= t0] then Mulam is just the baseline cumulative hazard funtion at t0. Where do you define the Mulam? (in fitYP3....)

Value

A scalar which is the hazard ratio.

Author(s)

Mai Zhou

References

Zhou, M. (2002). Computing censored empirical likelihood ratio by EM algorithm. *Tech Report, Univ. of Kentucky, Dept of Statistics*

Examples

```
## censored regression with one right censored observation.
## we check the estimation equation, with the MLE inside myfun7.
y <- c(3, 5.3, 6.4, 9.1, 14.1, 15.4, 18.1, 15.3, 14, 5.8, 7.3, 14.4)
x <- c(1, 1.5, 2,   3,   4,    5,    6,    5,    4,  1,   2,   4.5)
d <- c(1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0)
```

---

simuDataYP — *Generate random times that follow the YP model with the Given Parameters th1, th2, and alphaX.*

Description

This function is for simulations. It generates data from Yang-Prentice model with given/known parameters and may be used later to see how well some estimation procedure works on them. th1 = exp(beta1), th2 = exp(beta2), alphaX = $a' X$. There is always a covariate Z that indicates the two samples, and the hazards of the two treatments follows the Yang–Prentice model. The baseline hazard of sample one (where Z=0) is taken to be exponential.

Usage

```
simuDataYP(n1, n2, th1, th2, cens, alphaX)
```

Arguments

n1	sample size of first arm.
n2	sample size of second arm.
th1	the parameter of th1=exp(beta1). Short term.
th2	the parameter of th2=exp(beta2). Long term.
cens	logical, Either TRUE or FALSE.
alphaX	a vector of length n1+n2. It is the inner product of alpha and covariates X....the part that is proportional hazards. This way, alpha can be p dimensional, However alpha times X is always a vector of length n1+n2.

Details

The hazard of the generated survival times, Y, have hazard function that is proportional to exp(alphaX).

The hazard of arm 1 is constant, just exp(alphaX). The hazard of arm 2 is given as exp(alphaX) / [1/th1 S_0(t) + 1/th2 F_0(t)]

where S_0 and F_0 are survival function and CDF of a standard exponential random variable.

Value

A list with the following components:

Y	The survival times, possibly right censored.
d	The censoring status.
Zmat	the covariates used in generating random times.

Author(s)

Mai Zhou

References

Yang and Prientice. (2005). Semiparametric analysis of short term/long term hazard ratios with two sample survival data. *Biometrika*

Examples

```
## generate data and covariates.
X <- -99:100/50        ## the covariate for alpha, 200 long
temp <- simuDataYP(n1=100, n2=100, th1=exp(1), th2=exp(-1), cens=TRUE, alphaX = -0.5*X)
## this generate a sample of censored data with n=200. beta1=1, beta2=-1, alpha= -0.5.
## and the design matrix or covariance matrix is
Zmat <- cbind(X, temp$Zmat)
```

`smallcell` *Smallcell Lung Cancer Data*

Description

There are 121 observations on 4 variables. Arm is the indication of two treatments. Entry is the age of the patient at entry. Survival is the survival time and indicator is the censoring indicator (right censoring). For more details please see the reference below.

Data are from Ying, Z., Jung, SH, and Wei, LJ (1995). Median regression analysis with censored data. Journal of the American Statistical Association, 90, 178-184.

Usage

```
data(smallcell)
```

Format

A data frame containing 121 observations on 4 variables:

[,1] "arm"
[,2] "entry"
[,3] "survival"
[,4] "indicator"

References

Ying, Z., Jung, SH, and Wei, LJ (1995). Median regression analysis with censored data. Journal of the American Statistical Association, 90, 178-184.

Index