Package ‘WLreg’

April 18, 2017

Type Package
Version 1.0.0
Date 2017-04-18
Title Regression Analysis Based on Win Loss Endpoints
Description Use various regression models for the analysis of win loss endpoints adjusting for non-binary and multivariate covariates.
Depends R (>= 3.1.2)
Imports inline, stats, survival
License GPL (>= 2)
RoxygenNote 5.0.1
LazyData true
NeedsCompilation yes
Author Xiaodong Luo [aut, cre], Sanofi [cph]
Maintainer Xiaodong Luo <Xiaodong.Luo@sanofi.com>
Repository CRAN
Date/Publication 2017-04-18 05:29:06 UTC

R topics documented:

  winreg ................................................................. 2
  wrlogistic ............................................................ 4
  zinv ................................................................. 8

Index 9
Description

Use two Cox regression models (one for the terminal event and the other for the non-terminal event) to model the win product adjusting for covariates.

Usage

winreg(y1, y2, d1, d2, z)

Arguments

- **y1**: a numeric vector of event times denoting the minimum of event times $T_1$, $T_2$ and censoring time $C$, where the endpoint $T_2$, corresponding to the terminal event, is considered of higher clinical importance than the endpoint $T_1$, corresponding to the non-terminal event. Note that the terminal event may censor the non-terminal event, resulting in informative censoring.
- **y2**: a numeric vector of event times denoting the minimum of event time $T_2$ and censoring time $C$. Clearly, $y_2$ is not smaller than $y_1$.
- **d1**: a numeric vector of event indicators with 1 denoting the non-terminal event is observed and 0 else.
- **d2**: a numeric vector of event indicators with 1 denoting the terminal event is observed and 0 else.
- **z**: a numeric matrix of covariates.

Details

This function uses two Cox regression models (one for the terminal event and the other for the non-terminal event) to model the win product adjusting for covariates.

Value

- **beta1**: Estimated regression parameter based on the non-terminal event times $y_1$, $\exp(\beta1)$ is the adjusted hazard ratio
- **sigma1**: Estimated variance of beta1 using the residual method instead of the inverse of Fisher information
- **tb1**: Wald test statistics based on beta1 and sigma1
- **pb1**: Two-sided p-values of the Wald test statistics tb1
- **beta2**: Estimated regression parameter based on the terminal event times $y_2$, $\exp(\beta2)$ is the adjusted hazard ratio
- **sigma2**: Estimated variance of beta2 using the residual method instead of the inverse of Fisher information
Wald test statistics based on beta2 and sigma2
Two-sided p-values of the Wald test statistics tb2
beta1+beta2, exp(-beta) is the adjusted win product
Estimated variance of beta using the residual method
Wald test statistics based on beta and sigma
Two-sided p-values of the Wald test statistics tb

Author(s)
Xiaodong Luo

References

See Also
wrlogistic

Examples
```r
n<-300
rho<-0.5
b2<-c(1.0,-1.0)
b1<-c(0.5,-0.9)
bc<-c(1.0,0.5)
lambda10<-0.1; lambda20<-0.08; lambdac0<-0.09
lam1<-rep(0,n); lam2<-rep(0,n); lamc<-rep(0,n)
z1<-rep(0,n)
z1[1:(n/2)]<-1
z2<-rnorm(n)
z<-cbind(z1,z2)

lam1<-lam2<-lamc<-rep(0,n)
for (i in 1:n){
  lam1[i]<-lambda10*exp(-sum(z[i,]*b1))
  lam2[i]<-lambda20*exp(-sum(z[i,]*b2))
  lamc[i]<-lambdac0*exp(-sum(z[i,]*bc))
}
tem<-matrix(0,ncol=3,nrow=n)
y2y<-matrix(0,nrow=n,ncol=3)
```
```r
y2y[1]<-rnorm(n);y2y[3]<-rnorm(n)
y2y[2]<-rnorm(n)

y2y[,2]<-y2y[,1]+sqrt(1-rho^2)*y2y[,3]

tem[,1]<--log(1-pnorm(y2y[,1])/lam1)

lam1<-log(1-pnorm(y2y[,2])/lam2)

lamc<-log(1-runif(n))/lamc

y1<-apply(tem,1,min)
y2<-apply(tem[,2:3],1,min)
d1<-as.numeric(tem[,1]<-y1)
d2<-as.numeric(tem[,2]<-y2)

y<-cbind(y1,y2,d1,d2)

z<-as.matrix(z)

aa<-winreg(y1,y2,d1,d2,z)

aa
```

---

**wrlogistic**

Logistic regression for win ratio

---

**Description**

Use a logistic regression model to model win ratio adjusting for covariates with the user-supplied comparison results.

**Usage**

```r
wrlogistic(aindex,z,b0=rep(0,ncol(z)),tol=1.0e-04,maxiter=20)
```  

**Arguments**

- `aindex` a vector that collects the pairwise comparison results. Suppose there are a total of `n` subjects in the study, there are `n(n - 1)/2` elements in `aindex`. The `(i - 1)*(i - 2)/2 + j`-th element, denoted by $C_{ij}$, is the comparison result between subject $i$ and subject $j$, where $i = 2, \ldots, n$ and $j = 1, \ldots, i - 1$. The element $C_{ij}$ is equal to $1$ if subject $i$ wins over subject $j$ on the most important outcome, $C_{ij}$ is equal to $-1$ if subject $i$ loses against subject $j$ on the most important outcome; $C_{ij}$ is equal to $2$ if subject $i$ wins over subject $j$ on the second most important outcome after tie on the most important outcome, $C_{ij}$ is equal to $-2$ if subject $i$ loses against subject $j$ on the second most important outcome after tie on the most important outcome; and so forth until all the outcomes have been used for comparison; then $C_{ij}$ is equal to $0$ if an ultimate tie is resulted.

- `z` a matrix of covariates

- `b0` the initial value of the regression parameter

- `tol` error tolerance

- `maxiter` maximum number of iterations
Details

This function uses a logistic regression model to model win ratio adjusting for covaraites. This function uses the pairwise comparision result supplied by the user which hopefully will speed up the program.

Value

- **b**: Estimated regression parameter, \( \exp(b) \) is the adjusted win ratio
- **ubeta**: The score function
- **vbeta**: The estimated variance of \( \sqrt{n}\times b \)
- **Wald**: Wald test statistics for the estimated parameter b
- **pvalue**: Two-sided p-values of the Wald statistics
- **Imatrix**: The information matrix
- **wtotal**: Total wins
- **ltotal**: Total losses
- **err**: err at convergence
- **iter**: number of iterations performed before convergence

Author(s)

Xiaodong Luo

References


See Also

- **winreg**

Examples

```r
### Generate data
n<-300
rho<-0.5
b2<-c(1.0,-1.0)
b1<-c(0.5,-0.9)
bc<-c(1.0,0.5)
lambda1<-c(0.1,0.08,0.09)
lambda2<-rep(0,n);lamc<-rep(0,n)
```
```R
z1 <- rep(0, n)
z1[1:(n/2)] <- 1
z2 <- rnorm(n)
z <- cbind(z1, z2)

lam1 <- lam2 <- lamc <- rep(0, n)
for (i in 1:n) {
  lam1[i] <- lambda1 * exp(-sum(z[i, ] * b1))
  lam2[i] <- lambda2 * exp(-sum(z[i, ] * b2))
  lamc[i] <- lambdaC * exp(-sum(z[i, ] * bc))
}

tem <- matrix(0, ncol = 3, nrow = n)
yRy <- matrix(0, nrow = n, ncol = 3)
yRy[, 1] <- rnorm(n); yRy[, 3] <- rnorm(n)
yRy[, 2] <- rho * yRy[, 1] + sqrt(1 - rho^2) * yRy[, 3]

for (i in 1:n) {
  tem[, i] <- log((1 - pnorm(yRy[, 1])) / lam1)
  tem[, 2] <- log((1 - pnorm(yRy[, 2])) / lam2)
  tem[, 3] <- log(1 - runif(n)) / lamc
}

y1 <- apply(tem, 1, min)
y2 <- apply(tem[, 2:3], 1, min)
d1 <- as.numeric(tem[, 1] <= y1)
d2 <- as.numeric(tem[, 2] <= y2)

y <- cbind(y1, y2, d1, d2)
z <- as.matrix(z)

### Define the comparison function
comp <- function(y, x) {
  y1i <- y[1]; y2i <- y[2]; d1i <- y[3]; d2i <- y[4]
  y1j <- x[1]; y2j <- x[2]; d1j <- x[3]; d2j <- x[4]
  w2 < 0; w1 < 0; l2 < 0; l1 < 0

  if (d2j == 1 & y2i == y2j) w2 <- 1
  else if (d2i == 1 & y2j == y2i) l2 <- 1

  if (w2 == 0 & l2 == 0 & d1j == 1 & y1j == y1j) w1 <- 1
  else if (w2 == 0 & l2 == 0 & d1i == 1 & y1i == y1i) l1 <- 1

  comp <- 0
  if (w2 == 1) comp <- 1
  else if (l2 == 1) comp <- (-1)
  else if (w1 == 1) comp <- (-2)
  else if (l1 == 1) comp <- (-2)

  comp
}

bin <- rep(0, n*(n-1)/2)
for (i in 2:n) for (j in 1:(i-1)) bin[(i-1)*((i-2)/2+j)] <- comp(y[i, ], y[j, ])
bbz <- wrlogistic(bin, z, b0 = rep(0, ncol(z)), tol = 1e-04, maxiter = 20)
```
Calculate the win, loss, tie result using Fortran loops to speed up the process

Using the "inline" package to convert the code into Fortran

```fortran
#install.packages("inline") #Install the package "inline'
library("inline") ###Load the package "inline"

# The use of `inline` needs `rtools` and `gcc`
# in the PATH environment of R.
# The following code will put these two into
# the PATH for the current R session ONLY.

rtools <- "C:\Rtools\bin"
gcc <- "C:\Rtools\gcc-4.6.3\bin"
path <- strsplit(Sys.getenv("PATH"), ";")[[1]]
new_path <- c(rtools, gcc, path)
new_path <- new_path[((tolower(new_path))]
Sys.setenv(PATH = paste(new_path, collapse = ";"))

codex4 <- 
integer::i,j,indexij,d1i,d2i,d1j,d2j,w1,\l1,\l2,\l1
double precision::y1i,y2i,y1j,y2j
do i=2,n,1
  y1i=y(i,1);y2i=y(i,2);d1i=dnint(y(i,3));d2i=dnint(y(i,4))
do j=1,(i-1),1
  y1j=y(j,1);y2j=y(j,2);d1j=dnint(y(j,3));d2j=dnint(y(j,4))
  indexij=(i-1)*(i-2)/2+j
  w2=0;w1=0;l2=0;l1=0
  if (d2j==1 .and. y2i>=y2j) then
    w2=1
  else if (d2i==1 .and. y2j>=y2i) then
    l2=1
  else if (d1j==1 .and. y1i>=y1j) then
    w1=1
  else if (d1i==1 .and. y1j>=y1i) then
    l1=1
  end if
  aindex(indexij)=0
  if (w2==1) then
    aindex(indexij)=1
  else if (l2==1) then
    aindex(indexij)=-1
  else if (w2==0 .and. l2==0 .and. w1==1) then
    aindex(indexij)=2
  else if (w2==0 .and. l2==0 .and. l1==1) then
    aindex(indexij)=-2
  end if
end do
```

end do
zinv

Description
This will calculate the inverse matrix by Gauss elimination method

Usage
zinv(y)

Arguments
y          a square matrix

Details
Inverse matrix

Value
y
the inverse of y

Note
This provides the inverse matrix using Gauss elimination method, this program performs satisfactorily when the size of the matrix is less than 50

Author(s)
Xiaodong Luo

Examples
y<-matrix(c(1,2,0,1),ncol=2,nrow=2)
zinv(y)
Index

*Topic Cox regression
   winreg, 2
*Topic Win loss regression
   winreg, 2
*Topic conditional logistic regression
   wrlogistic, 4
*Topic logistic regression
   wrlogistic, 4
*Topic pairwise comparison
   wrlogistic, 4
*Topic weighted
   wrlogistic, 4
*Topic win loss statistics
   winreg, 2
*Topic win product
   winreg, 2
*Topic win ratio
   winreg, 2
   wrlogistic, 4

winreg, 2, 5
wrlogistic, 3, 4
zinv, 8