Composite endpoint analysis for time-to-event data using Win Ratio

Description

This package uses Win Ratio (WR) as a summary statistic to compare the composite endpoints of time-to-event data between two groups. Options of clustered or independent time-to-event data can be specified.

Usage

```
cWR(treatment, cluster,
    y1, y2,
    delta1, delta2,
    null.WR=1, alpha.sig=0.05)
```

Arguments

`treatment`	An integer vector with code 0 as control group and 1 as treatment group for each subject
`cluster`	An integer vector with unique cluster ID for each cluster. When subjects are independent, the cluster ID is unique for each subject.
`y1`	Let T_H, T_D and T_C be time to non-fatal event, time to fatal event and censoring time, respectively. y1 is a numeric vector with min(T_H, T_D, T_C) for each subject.
`y2`	A numeric vector with min(T_D, T_C) for each subject
`delta1`	An integer vector with code 1 indicating that T_H is observed, 0 otherwise
`delta2`	An integer vector with code 1 indicating that T_D is observed, 0 otherwise
`null.WR`	The null hypothesis of the WR statistic. The default is H0: WR=1 or log(WR)=0.
`alpha.sig`	The significance level, with default value 0.05

Details

The function "cWR" performs significance testing of comparing two composite time-to-event outcomes between groups. The Win Ratio summary statistic is built on the "unmatched" approach described by Pocock et al. (2011). We assume that the composite endpoints can be formulated as semi-competing risk data. Each individual in the study is measured on time to non-fatal (non-terminal) event (e.g. hospitalization) and time to fatal (terminal) event (e.g. death). Specifically, the fatal event is considered clinically more important compared to the non-fatal event. Censoring is allowed, but time to censor needs to be observed.

This function can handle independent data, as well as clustered data. The inference of clustered data is based on the generalized bivariate clustered U-statistics proposed by Zhang and Jeong (2019). This clustered U-statistic accounts for the potential correlations among subjects within a cluster. When the cluster size is 1, it's the independent setting and the inference is the same as the method proposed by Bebu and Lachin (2015).

Note: The option "treatment", "cluster", "y1", "y2", "delta1", "delta2" are required and no defaults are provided. These options have to be vectors with the same length. No missing values are allowed.

Value

`name`	The test name
`U1`	First estimated clustered U-statistic
`U2`	Second estimated clustered U-statistic
`logWR`	Estimated WR on log scale
`se`	Estimated standard error of the WR on log scale
`z`	Test statistic
`ci`	100(1-alpha.sig)% confidence interval
`p`	P-value of the significance testing
`var_cov`	Variance and covariance matrix of the first and second clustered U-statistics

Author(s)

Di Zhang <diz11@pitt.edu>; Jong H. Jeong <jjeong@pitt.edu>

Maintainer: Di Zhang <diz11@pitt.edu>

References

Pocock, S. J., Ariti, C. A., Collier, T. J., andWang, D. (2011). The win ratio: a new approach to the analysis of composite endpoints in clinical trials based on clinical priorities. European heart journal 33, 176-182.

Bebu, I. and Lachin, J. M. (2015). Large sample inference for a win ratio analysis of a composite outcome based on prioritized components. Biostatistics 17, 178-187.

Zhang, D. and Jeong, H. J. Inference on the Win Ratio for Clustered Semi-Competing Risk Data. (Under First Review by the Lifetime Data Analysis, 2019)

Examples

```
  ## Not run:

# load library
library(gumbel)
library(devtools)

# download and install package through Github
install_github("dee1008/cWR")
library(cWR)

set.seed(123)


#---------1. Data generation for independent semi-competing risk data-----
# joint survival: bivariate exponential with Gumbel-Hougaard copula
# define functions
gumbel_independent<-function(n,n.clust,dim,alpha,lambdaH,lambdaD,etaH,etaD)
{
  exprand <- matrix(rexp(dim * n), c(n, dim))
  unifpirand <- runif(n, 0, pi)
  exprand2 <- rexp(n)
  beta <- 1/alpha
  stablerand <- sin((1 - beta) * unifpirand)^((1 - beta)/beta) *
    (sin(beta * unifpirand))/(sin(unifpirand))^(1/beta)
  stablerand <- stablerand/(exprand2^(alpha - 1))
  unifrand <- invphigumbel(exprand/stablerand, alpha) # generating bivariate uniform random variables for marginal survival funtions--(*)

  clust.siz<- n/n.clust
  frail <- rep(rep(1, n.clust),each=clust.siz)
  matrix(c(-log(unifrand[,1])/(frail*lambdaH*exp(-etaH)),-log(unifrand[,2])/(frail*lambdaD*exp(-etaD))),c(n,dim)) # inverting specific forms of survival functions in (*) to create
  # true bivariate event times adjusted for event types and trt groups
}

gen_independent<-function(n.sub, n.clust, dim, alpha, lambdaH, lambdaD, lambdaC, etaH, etaD, etaC){


  group0<-gumbel_independent(n.sub,n.clust,dim,alpha,lambdaH,lambdaD,0,0)
  group1<-gumbel_independent(n.sub,n.clust,dim,alpha,lambdaH,lambdaD,etaH,etaD)

  true.t<-rbind(group0,group1)
  temp.data<-cbind(true.t,c(rexp(dim(true.t)[1]/2,lambdaC),rexp(dim(true.t)[1]/2,lambdaC*exp(-etaC))))

  t.obs<-apply(temp.data,1,min)
  delH<-rep(0,dim(true.t)[1])
  delD<-rep(0,dim(true.t)[1])
  delH[temp.data[,1]==t.obs]<-1
  delD[temp.data[,2]==t.obs]<-1

  my.data<-cbind(temp.data,t.obs,delH,delD,rep(0:1,each=dim(true.t)[1]/2))
  y1<-rep(0,dim(true.t)[1])
  y2<-rep(0,dim(true.t)[1])

  my.data.f<-data.frame(cbind(my.data,y1,y2))
  names(my.data.f)<-c("t1","t2","c","t.obs","delH","delD","group","y1","y2")

  indi.1<-(my.data.f$c < my.data.f$t1) & (my.data.f$c < my.data.f$t2)
  my.data.f$y1[indi.1]<-my.data.f$c[indi.1]
  my.data.f$y2[indi.1]<-my.data.f$c[indi.1]

  indi.2<-(my.data.f$t2 < my.data.f$t1) & (my.data.f$t1 < my.data.f$c)
  indi.21<-(my.data.f$t2 < my.data.f$c) & (my.data.f$c < my.data.f$t1)
  my.data.f$y1[indi.2 | indi.21]<-my.data.f$t2[indi.2| indi.21]
  my.data.f$y2[indi.2| indi.21]<-my.data.f$t2[indi.2| indi.21]

  indi.3<-(my.data.f$t1 < my.data.f$c) & (my.data.f$c < my.data.f$t2)
  my.data.f$y1[indi.3]<-my.data.f$t1[indi.3]
  my.data.f$y2[indi.3]<-my.data.f$c[indi.3]

  indi.4<-(my.data.f$t1 < my.data.f$t2) & (my.data.f$t2 < my.data.f$c)
  my.data.f$y1[indi.4]<-my.data.f$t1[indi.4]
  my.data.f$y2[indi.4]<-my.data.f$t2[indi.4]
```

```
  my.data.f$delD[indi.4]<-1

  # add cluster information in the data set
  my.data.f$cluster<-rep(1:(2*n.clust),each=n.sub/n.clust)

  names(my.data.f)<-c("time_Non_Fatal","time_Fatal","time_censor","t.obs","delH","delD","treatment","y1","y2","cluster")

  return(my.data.f)

}


# generate independent data
data1<-gen_independent(n.sub=100, n.clust=100, dim=2, alpha=2, lambdaH=0.1, lambdaD=0.08, lambdaC=0.09, etaH=0.2, etaD=0.5, etaC=0.1)

# independent win ratio
ind.wr<-with(data1, cWR(treatment=treatment, cluster=cluster, y1=y1, y2=y2, delta1=delH, delta2=delD, null.WR=1,alpha.sig=0.05))

# logWR
ind.wr$logWR
# se of logWR
ind.wr$se
# 95
ind.wr$ci
# p-value
ind.wr$p



#------2. Data generation for clustered semi-competing risk data--------
# joint survival: clustered bivariate exponential with Gumbel-Hougaard copula
# define functions
gumbel_cluster<-function(n,n.clust,dim,alpha,lambdaH,lambdaD,etaH,etaD,shape,rate)
{
  exprand <- matrix(rexp(dim * n), c(n, dim))
  unifpirand <- runif(n, 0, pi)
  exprand2 <- rexp(n)
  beta <- 1/alpha
  stablerand <- sin((1 - beta) * unifpirand)^((1 - beta)/beta) *
    (sin(beta * unifpirand))/(sin(unifpirand))^(1/beta)
  stablerand <- stablerand/(exprand2^(alpha - 1))
  unifrand <- invphigumbel(exprand/stablerand, alpha) # generating bivariate uniform random variables for marginal survival funtions--(*)

  clust.siz<- n/n.clust
  frail <- rep(rgamma(n.clust,shape=shape,rate=rate),each=clust.siz)
  matrix(c(-log(unifrand[,1])/(frail*lambdaH*exp(-etaH)),-log(unifrand[,2])/(frail*lambdaD*exp(-etaD))),c(n,dim)) # inverting specific forms of survival functions in (*) to create
  # true bivariate event times adjusted for event types and trt groups
}

gen_cluster<-function(n.sub, n.clust, dim, alpha, lambdaH, lambdaD, lambdaC, etaH, etaD, etaC, shape, rate){


  group0<-gumbel_cluster(n.sub,n.clust,dim,alpha,lambdaH,lambdaD,0,0,shape,rate)
  group1<-gumbel_cluster(n.sub,n.clust,dim,alpha,lambdaH,lambdaD,etaH,etaD,shape,rate)

  true.t<-rbind(group0,group1)
  temp.data<-cbind(true.t,c(rexp(dim(true.t)[1]/2,lambdaC),rexp(dim(true.t)[1]/2,lambdaC*exp(-etaC))))

  t.obs<-apply(temp.data,1,min)
  delH<-rep(0,dim(true.t)[1])
  delD<-rep(0,dim(true.t)[1])
  delH[temp.data[,1]==t.obs]<-1
  delD[temp.data[,2]==t.obs]<-1

  my.data<-cbind(temp.data,t.obs,delH,delD,rep(0:1,each=dim(true.t)[1]/2))
  y1<-rep(0,dim(true.t)[1])
  y2<-rep(0,dim(true.t)[1])

  my.data.f<-data.frame(cbind(my.data,y1,y2))
  names(my.data.f)<-c("t1","t2","c","t.obs","delH","delD","group","y1","y2")

  indi.1<-(my.data.f$c < my.data.f$t1) & (my.data.f$c < my.data.f$t2)
  my.data.f$y1[indi.1]<-my.data.f$c[indi.1]
  my.data.f$y2[indi.1]<-my.data.f$c[indi.1]

  indi.2<-(my.data.f$t2 < my.data.f$t1) & (my.data.f$t1 < my.data.f$c)
  indi.21<-(my.data.f$t2 < my.data.f$c) & (my.data.f$c < my.data.f$t1)
  my.data.f$y1[indi.2 | indi.21]<-my.data.f$t2[indi.2| indi.21]
  my.data.f$y2[indi.2| indi.21]<-my.data.f$t2[indi.2| indi.21]

  indi.3<-(my.data.f$t1 < my.data.f$c) & (my.data.f$c < my.data.f$t2)
  my.data.f$y1[indi.3]<-my.data.f$t1[indi.3]
  my.data.f$y2[indi.3]<-my.data.f$c[indi.3]

  indi.4<-(my.data.f$t1 < my.data.f$t2) & (my.data.f$t2 < my.data.f$c)
  my.data.f$y1[indi.4]<-my.data.f$t1[indi.4]
  my.data.f$y2[indi.4]<-my.data.f$t2[indi.4]

  my.data.f$delD[indi.4]<-1

  # add cluster information in the data set
  my.data.f$cluster<-rep(1:(2*n.clust),each=n.sub/n.clust)

  names(my.data.f)<-c("time_Non_Fatal","time_Fatal","time_censor","t.obs","delH","delD","treatment","y1","y2","cluster")

  return(my.data.f)

}

# generate clustered data
data2<-gen_cluster(n.sub=500, n.clust=25, dim=2, alpha=2, lambdaH=0.1, lambdaD=0.08, lambdaC=0.09, etaH=0.2, etaD=0.5, etaC=0.1, shape=15, rate=15)

# clustered win ratio
clus.wr<-with(data2, cWR(treatment=treatment, cluster=cluster, y1=y1, y2=y2, delta1=delH, delta2=delD, null.WR=1,alpha.sig=0.05))

# logWR
clus.wr$logWR
# se of logWR
clus.wr$se
# 95
clus.wr$ci
# p-value
clus.wr$p




## End(Not run)
```