Causal inference on Win Ratio for composite endpoints of semi-competing risk time-to-event data

Description

This function uses Win Ratio (WR) as a summary statistic to compare the composite endpoints of semi-competing risk time-to-event data between two groups, in observational study setting.

Analysis can be done for the following scenarios: (1) Causal inference of independent subjects with confounders (2) Causal inference of cluster-dependent subjects with confounders (observational study with one-level cluster structure). This cluster structure assumes that all clusters include both comparison groups. For instance, patients are nested within hospitals, and each hospital has patients in treatment group and patients in control group. Clusters with only one memberships will be automatically excluded in analysis.

Usage

```
WR.causal(treatment, cluster, y1, y2, delta1, delta2, x.con, x.char, null.WR=1, alpha.sig=0.05, control=NA, n.boot=200)
```

Arguments

`treatment`	Integer vector with code 0 as control group and 1 as treatment group for each subject
`cluster`	Integer vector with unique cluster ID for each cluster. When subjects are independent, the cluster ID is unique for each subject.
`y1`	Numeric vector with min(T_H, T_D, T_C) for each subject, where T_H, T_D and T_C are time to non-fatal event, time to fatal event and censoring time, respectively
`y2`	Numeric vector with min(T_D, T_C) for each subject
`delta1`	Integer vector with code 1 indicating that T_H is observed, 0 otherwise
`delta2`	Integer vector with code 1 indicating that T_D is observed, 0 otherwise
`x.con`	Continuous covariate matrix with observations as rows and variables as columns. All columns have to be numeric type.
`x.char`	Categorical covariate matrix with observations as rows and variables as columns. All columns have to be character type.
`null.WR`	Null hypothesis of the WR statistic. The default is H0: WR=1 or log(WR)=0.
`alpha.sig`	Significance level, with default value 0.05
`control`	List of control options for the estimation of the lagrange multipliers in the estimation of calibration weights. The control options are the same as the ones in R package "nleqslv". The default value is control=NA, and the algorithm uses all default settings in "nleqslv". When specify the options, option names are needed. For instance, control=list(xtol=1e-05, ftol=1e-06).
`n.boot`	Number of bootstrap replications for variance estimation. It's only used for analysis of cluster-dependent data. Default value is 200.

Details

The function "WR.causal" conducts causal inference and significance testing comparing two composite time-to-event outcomes between groups, accounting for confounders. 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 calibration weighted stratified estimator studied by Zhang and Jeong (2019). The construction of the causal inference for Win Ratio is based on the causal U-statistics by Mao (2017), and the calibration weight estimation follows Yang (2018). This weighted stratified estimator accounts for potential correlations among subjects within a cluster and confounder effects. The estimation of the calibration weights requires the estimation of lagrange multipliers through numeric iterations. We adoptted the R package "nleqslv" for the estimation.

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

Warning: The convergence becomes difficult when there are many multi-level categorical variables. We recommend to dichotomize multi-level variables to increase the probability of convergence.

Value

`name`	Test name
`info`	Number of patients and number of clusters in analysis
`clusters`	Clusters that are included in analysis (excluded those with only one membership within clusters)
`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-value`	P-value of the significance testing
`convergence`	Info of convergence for the estimation of lagrange multipliers. Values are the same as the ones in R package "nleqslv".

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.

Mao, L. (2017). On causal estimation using-statistics. Biometrika, 105(1), 215-220.

Yang, S. (2018). Propensity score weighting for causal inference with clustered data. Journal of Causal Inference, 6(2).

Zhang, D. and Jeong, H. J. Causal Inference on Win Ratio for Clustered Semi-Competing Risk Data. (In draft, 2019)

Examples

```
  ## Not run:


# load library
library(gumbel)
library(copula)

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

library(cWR)

set.seed(123)



#---------1. Data generation for independent semi-competing risk data with confounders-----
# joint survival: bivariate exponential with Gumbel-Hougaard copula
# define functions
gumbel_causal_individual<-function(n.sub,dim,alpha,lambdaH,lambdaD,etaH,etaD,cov.H, cov.D)
{
  exprand <- matrix(rexp(dim * n.sub), c(n.sub, dim))
  unifpirand <- runif(n.sub, 0, pi)
  exprand2 <- rexp(n.sub)
  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--(*)

  matrix(c(-log(unifrand[,1])/(lambdaH*exp(-etaH+cov.H)),-log(unifrand[,2])/(lambdaD*exp(-etaD+cov.D))),c(n.sub,dim)) # inverting specific forms of survival functions in (*) to create
  # true bivariate event times adjusted for event types and trt groups
}


gen_causal_individual<-function(N,dim,alpha,lambdaH,lambdaD,lambdaC,etaH,outcome.H.eta,etaD,outcome.D.eta,etaC,ps.model.phi){

  # generate covariates z1 and z2
  z1<-rnorm(N)
  z2<-sample(c(1,0), N, prob=c(0.5, 0.5), replace=TRUE)

  # generate treatment status
  # design matrix X
  X<-cbind(rep(1,N), z1, z2)
  # calculate probability of getting treatment for each obs
  p<-exp(X%*%ps.model.phi)/(1+exp(X%*%ps.model.phi))
  trt<-sapply(p, function(x) sample(c(1,0), 1, prob=c(x, (1-x))))
  # number of treatment obs
  m<-sum(trt)
  # number of control obs
  n<-length(trt)-sum(trt)
  # percentage of treatment and control obs
  percent<-c("control%"=n/N, "trt%"=m/N)

  # combine data trt, z1, z2
  temp<-cbind(trt, z1, z2)
  # separate treatment temp data
  temp.trt<-temp[(temp[,1]==1),]
  temp.con<-temp[(temp[,1]==0),]

  # generate time to events: time to non-fatal and time to fatal events
  group0<-gumbel_causal_individual(n,dim,alpha,lambdaH,lambdaD,0,0,temp.con[,2:3]%*%outcome.H.eta, temp.con[,2:3]%*%outcome.D.eta)
  group1<-gumbel_causal_individual(m,dim,alpha,lambdaH,lambdaD,etaH,etaD, temp.trt[,2:3]%*%outcome.H.eta, temp.trt[,2:3]%*%outcome.D.eta)

  # combine time to events and time to censoring
  true.t<-rbind(group0,group1)
  temp.data<-cbind(true.t,c(rexp(n,lambdaC),rexp(m,lambdaC*exp(-etaC))))

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

```r
  delH[temp.data[,1]==t.obs]<-1
  delD[temp.data[,2]==t.obs]<-1

  my.data<-cbind(temp.data,t.obs,delH,delD,rbind(temp.con, temp.trt))
  y1<-rep(0,n+m)
  y2<-rep(0,n+m)

  my.data.f<-data.frame(cbind(my.data,y1,y2))
  names(my.data.f)<-c("t1","t2","c","t.obs","delta1","delta2","group","z1","z2","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


  names(my.data.f)<-c("time_Non_Fatal","time_Fatal","time_censor","t.obs","delta1","delta2","treatment","z1","z2","y1","y2")



  output<-list(my.data.f, n, m, percent)
  names(output)<-c("data", "#control", "#trt", "assignment
  return(output)

}



# generate independent data
data1<-gen_causal_individual(N=1000,dim=2, alpha=2, lambdaH=0.1, lambdaD=0.08, lambdaC=0.09, etaH=0, outcome.H.eta=c(0.1, 0.3), etaD=0, outcome.D.eta=c(0.2, 0.4), etaC=0.1, ps.model.phi=c(-0.2, 0.5, 0.5))$data

# generate cluster variable
data1$cluster<-seq(1:nrow(data1))

# generate continuous covariate matrix
x.con<-as.matrix(data1[, c("z1","z2")])

# independent win ratio
ind.wr<-with(data1, WR.causal(treatment=treatment, cluster=cluster, y1=y1, y2=y2, delta1=delta1, delta2=delta2, x.con=x.con, n.boot=2))


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







#---------2. Data generation for cluster-dependent semi-competing risk data with confounders-----

gumbel_causal_PScluster<-function(n.sub,dim,alpha,lambdaH,lambdaD,etaH,etaD,cov.H, cov.D, frail)
{
  exprand <- matrix(rexp(dim * n.sub), c(n.sub, dim))
  unifpirand <- runif(n.sub, 0, pi)
  exprand2 <- rexp(n.sub)
  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--(*)

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


gen_causal_PScluster<-function(N, n.cluster, shape, rate, dim, alpha, lambdaH, lambdaD, lambdaC, etaH, outcome.H.eta, etaD, outcome.D.eta, etaC, ps.model.phi){

  # generate covariates z1 and z2, z3=z1*z2
  z1<-rnorm(N)
  #z2<-sample(c(1,0), N, prob=c(0.5, 0.5), replace=TRUE)
  z2<-rnorm(N, mean=1, sd=2)


  # generate clusters
  cluster<-rep(1:n.cluster, each=N/n.cluster)

  # generate cluster effect: use copula to create joint distribution of normal and gamma
  # constructs an elliptical copula
  myCop <- normalCopula(param=c(0.4), dim=2, dispstr="un")
  # creates a multivariate distribution via copula
  myMvd <- mvdc(copula=myCop, margins=c("norm", "gamma"),
                paramMargins=list(list(mean=0,  sd=2),
                                  list(shape=shape, scale=1/rate))
  )
  reffect <- rMvdc(n.cluster, myMvd)
  r1<-rep(reffect[,1], each=N/n.cluster) # cluster effect for treatment
  r2<-rep(reffect[,2], each=N/n.cluster) # cluster effect for outcome



  # generate treatment status
  # design matrix X
  X<-cbind(rep(1,N), z1, z2)
  # calculate probability of getting treatment for each obs
  p<-exp(X%*%ps.model.phi+r1)/(1+exp(X%*%ps.model.phi+r1))
  trt<-sapply(p, function(x) sample(c(1,0), 1, prob=c(x, (1-x))))
  # number of treatment obs
  m<-sum(trt)
  # number of control obs
  n<-length(trt)-sum(trt)
  # percentage of treatment and control obs
  percent<-c("control%"=n/N, "trt%"=m/N)


  # combine data trt, z1, z2, z3, cluster ID, cluster effect
  temp<-cbind(trt, z1, z2, cluster, r1, r2)
  # separate treatment temp data
  temp.trt<-temp[(temp[,1]==1),]
  temp.con<-temp[(temp[,1]==0),]

  # generate time to events: time to non-fatal and time to fatal events
  group0<-gumbel_causal_PScluster(n,dim,alpha,lambdaH,lambdaD,0,0,
                                  temp.con[,2:3]%*%outcome.H.eta,
                                  temp.con[,2:3]%*%outcome.D.eta, temp.con[,6])
  group1<-gumbel_causal_PScluster(m,dim,alpha,lambdaH,lambdaD,etaH,etaD,
                                  temp.trt[,2:3]%*%outcome.H.eta,
                                  temp.trt[,2:3]%*%outcome.D.eta,temp.trt[,6])

  # combine time to events and time to censoring
  true.t<-rbind(group0,group1)
  temp.data<-cbind(true.t,c(rexp(n,lambdaC),rexp(m,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,rbind(temp.con, temp.trt))
  y1<-rep(0,n+m)
  y2<-rep(0,n+m)

  my.data.f<-data.frame(cbind(my.data,y1,y2))
  names(my.data.f)<-c("t1","t2","c","t.obs","delta1","delta2","group","z1","z2","cluster","r1", "r2","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


  names(my.data.f)<-c("time_Non_Fatal","time_Fatal","time_censor","t.obs","delta1","delta2","treatment","z1","z2","cluster","cluster.trt.effect","cluster.outcome.effect","y1","y2")



  output<-list(my.data.f, n, m, percent)
  names(output)<-c("data", "#control", "#trt", "assignment
  return(output)

}


# generate cluster data
data2<-gen_causal_PScluster(N=1000,n.cluster=50,shape=15,rate=15,dim=2,alpha=2,lambdaH=0.1,lambdaD=0.08,lambdaC=0.09,etaH=0,outcome.H.eta=c(0.1, 0.3),etaD=0,outcome.D.eta=c(0.2, 0.4),etaC=0.1,ps.model.phi=c(-0.2,  0.5,  0.5))$data

# generate continuous covariate matrix
x.con<-as.matrix(data2[, c("z1","z2")])

# cluster win ratio
clus.wr<-with(data2, WR.causal(treatment=treatment, cluster=cluster, y1=y1, y2=y2, delta1=delta1, delta2=delta2, x.con=x.con, n.boot=2))


# cluster logWR
clus.wr$logWR
# se of logWR
clus.wr$se
# 95% CI of logWR
clus.wr$ci
# p-value
clus.wr$`p-value`






## End(Not run)
```