Package 'Survgini'

February 19, 2015

Type Package

Title The Gini concentration test for survival data

Version 1.0

Date 2011-11-14

Author Chiara Gigliarano and Marco Bonetti

Maintainer Chiara Gigliarano <c.gigliarano@univpm.it>

Depends R (>= 2.11.0), survival

Description The Gini concentration test for survival data is a nonparametric test based on the Gini index for testing the equality of two survival distributions from the point of view of concentration. The package compares different nonparametric tests (asymptotic Gini test, permutation Gini test, log-rank test, Gray-Tsiatis test and Wilcoxon test) and computes their p-values.

License GPL (>= 2)

LazyLoad yes

Repository CRAN

Date/Publication 2011-11-22 08:45:20

NeedsCompilation no

R topics documented:

`Survgini` *The Gini concentration test for survival data*

Description

Survgini is a nonparametric test for the equality of two survival distributions from the point of view of concentration. Two alternative tests are available, the asymptotic Gini test and the permutation test; the latter should be preferred to the asymptotic test in the case of unbalanced and small groups. The function Survgini compares different nonparametric tests (asymptotic Gini test, permutation Gini test, log-rank test, Gray-Tsiatis test and Wilcoxon test) and computes their p-values.

Usage

```
Survgini (asymptotic = TRUE, permutation = TRUE, M = 500, linearRank = TRUE,
lastEvent = 1, datasetOrig)
```

Arguments

`asymptotic`	a logical value indicating whether the asymptotic Gini test should be computed; the default is TRUE.
`permutation`	a logical value indicating whether the permutation Gini test should be computed; the default is TRUE.
`M`	the number of replications of permutation sampling; default is 500.
`linearRank`	a logical value indicating whether the linear rank tests (log-rank test, Gray-Tsiatis test and Wilcoxon test) should be computed; the default is TRUE.
`lastEvent`	represents the longest follow-up time until which we integrate the restricted Gini index. Choosing lastEvent =1 means integrating the restricted Gini statistic until the last event non-censored, while lastEvent =0 means integrating the restricted Gini statistic until the last observation censored or not.
`datasetOrig`	it must be a 3-columns dataset. The first column refers to the time-to-event, the second column is a censor indicator (=0 if the observation is right-censored, =1 if not), the third column refers to the treatment group (=1 for the first group, and =2 for the second group).

Details

The Gini index is one of the most common statistical indices employed in social sciences for measuring concentration in the distribution of a positive random variable (typically, income). This concentration index may be useful in detecting differences in heterogeneity also between the survival distributions of two groups of individuals to whom different treatments have been administered, thus providing a distinct assessment of such difference compared to what is provided by traditional tests.

In Bonetti, Gigliarano, and Muliere (2009) a nonparametric test has been proposed based on the Gini index for testing the equality of two survival distributions from the point of view of concentration. The authors have derived the asymptotic distribution of the test statistics and performed a simulation analysis, in which they compared the Gini test with other tests for the difference between two survival distributions, such as the log-rank, Wilcoxon, and Gray-Tsiatis tests.

Gigliarano and Bonetti (2011) have compared the asymptotic Gini test with a permutation test, suggesting that the permutation test should be preferred to the asymptotic test in the case of unbalanced and small groups.

The function Survgini compares the Gini test based on asymptotic and on permutation inference with three other tests (log-rank, Wilcoxon, and Gray-Tsiatis) and computes their p-values.

Value

The function Survgini prints a list containing the following components:

`pGiniAs`	the p-value of the asymptotic Gini test (if the argument asymptotic=TRUE is chosen)

`pGiniPerm`	the p-value of the permutation Gini test (if the argument permutation=TRUE is chosen)
`pGT`	the p-value of the Gray-Tsiatis test (if the argument linearRank=TRUE is chosen)
`pLR`	the p-value of the Log-Rank test (if the argument linearRank=TRUE is chosen)
`pW`	the p-value of the Wilcoxon test (if the argument linearRank=TRUE is chosen)

Author(s)

Chiara Gigliarano <`c.gigliarano@univpm.it`>, Marco Bonetti <`marco.bonetti@unibocconi.it`>.
Special thanks to Wai-ki Yip for his helpful assistance in the preparation of the package.

References

Bonetti, M., Gigliarano, C. and Muliere, P. (2009). The Gini concentration test for survival data. Lifetime Data Analysis, 15, 493-518.

Gigliarano, C. and Bonetti, M. (2011). The Gini test for survival data in presence of small and unbalanced groups. In press, Biomedical Statistics and Clinical epidemiology.

Examples

```
# Use the dataset aml on survival in patients with Acute Myelogenous Leukemia.
# The dataset contains the following variables:
      # time: survival or censoring time,
      # status: censoring status,
      # x:  maintenance chemotherapy given.

library(survival)
attach(aml)

# We first make data compatible with the SurvGini:
N<-length(aml[,1])
aml2<-matrix(nrow=N, ncol=3)
aml2[,1]<-time
aml2[,2]<-status
for (i in 1:N){
    if (aml[i,3]=="Maintained") aml2[i,3]<-1 else aml2[i,3]<-2
}

Survgini(asymptotic=TRUE, permutation=TRUE, M=500, lastEvent=1, datasetOrig=aml2)


## The function is currently defined as
function(asymptotic=TRUE, permutation=TRUE, M=500, linearRank=TRUE, lastEvent=1, datasetOrig){
    library(survival)
    VarGinicensor<-function(data, Tmax){
        data1<-data.frame(data)
        info<- survfit(Surv(data1[,1], data1[,2])~1 , type='kaplan-meier', data=data1)
        S<-matrix(info$surv)
        T<-matrix(info$time)
```

```
indices <- sum(1*(T<=Tmax))
if (indices==0){
    var<-0
    return(var=var)
}
else{
    Vt<-matrix(ncol=1,nrow=indices)
    Vt[1]<-1*T[1]
    if(indices>=2){
        for (i in 2:indices){
            Vt[i]<-Vt[i-1]+((S[i-1])^2)*(T[i]-T[i-1])
        }
    }
    lastpiecesVt<-((S[indices])^2)*(Tmax-T[indices])
    VtMax<-Vt[indices]+lastpiecesVt
    Wt<-matrix(ncol=1,nrow=indices)
    Wt[1]<-1*T[1]
    if(indices>=2){
        for (i in 2:indices){
            Wt[i]<-Wt[i-1]+S[i-1]*(T[i]-T[i-1])
        }
    }
    lastpiecesWt<-(S[indices])*(Tmax-T[indices])
    WtMax<-Wt[indices]+lastpiecesWt
    mu2<-matrix(ncol=1,nrow=indices)
    mu2<-VtMax-Vt
    mu<-matrix(ncol=1,nrow=indices)
    mu<-WtMax-Wt
    n<-length(data[,1])
    event<-matrix( ncol=1, nrow=indices)
    atrisk<-matrix(ncol=1, nrow=indices)
    for (i in 1:indices){
        event[i]<-info$n.event[i]
        atrisk[i]<-info$n.risk[i]
    }
    dsigma<-matrix(0, ncol=1,nrow=indices)
    for (i in 1:indices){
        if (atrisk[i]>0)
        dsigma[i]<-(n*event[i])/(atrisk[i]^2)
    }
    varistant<-matrix(ncol=1, nrow=indices)
    for (i in 1:indices){
        varistant[i]<-(4*exp(2*log(mu2[i])-2*log(WtMax))+exp(2*log(mu[i])+
        2*log(VtMax)-4*log(WtMax))-4*exp(log(mu[i])+log(mu2[i])+log(VtMax)-
        3*log(WtMax)))*(dsigma[i])
    }
    var<-matrix(ncol=1,nrow=indices)
    var[1]<-varistant[1]
    if(indices>=2){
        for (i in 2:indices){
           var[i]<-var[i-1]+varistant[i]
        }
    }
```

```
            return(var[indices]/n)
        }
    }
    Gcensor2<-function(data,Tmax=max(data[,1])){
        data1 <- data.frame(data)
        info <- survfit(Surv(data[,1], data[,2])~1 , type='kaplan-meier', data=data1)
        K<-length(info$time)
        S<-matrix(info$surv, ncol=1, nrow=K)
        num<-matrix(ncol=1,nrow=K)
        T<-matrix(info$time,ncol=1, nrow=K)
        num[1]<-1*T[1]
        if (K>=2){
            for (i in 2:K)
            num[i]<-num[i-1]+((S[i-1])^2)*(T[i]-T[i-1])
        }
        den <- matrix(ncol=1,nrow=K)
        den[1] <- 1*T[1]
        if (K>=2){
            for (i in 2:K)
            den[i]<-den[i-1]+S[i-1]*(T[i]-T[i-1])
        }
        G <- 1-(num/den)
        indices <- sum(1*(T<Tmax))
        lastpiecesnum <- ((S[indices])^2)*(Tmax-T[indices])
        lastpiecesden <- (S[indices])*(Tmax-T[indices])
        GTmax <- 1-(num[indices]+lastpiecesnum)/(den[indices]+lastpiecesden)
        return(list( G=G,info=info,GTmax=GTmax))
    }
    X <- datasetOrig[,1]
    delta <- datasetOrig[,2]
    Tx <- datasetOrig[,3]
    dataset1Orig <- datasetOrig[Tx==1,]
    dataset2Orig <- datasetOrig[Tx==2,]
    N<-length(datasetOrig[,1])
    N1<-length(dataset1Orig[,1])
    N2<-length(dataset2Orig[,1])
   if (lastEvent==1) A1<-dataset1Orig[,1]*1*(dataset1Orig[,2]==1) else A1<-dataset1Orig[,1]
   if (lastEvent==1) A2<-dataset2Orig[,1]*1*(dataset2Orig[,2]==1) else A2<-dataset2Orig[,1]
    Tmax1<-max(A1)+0.001
    Tmax2<-max(A2)+0.001
    Tmaxnum<-Tmax1*1*(N1>=N2)+Tmax2*1*(N2>N1)
    Ginisurv1 <- Gcensor2(dataset1Orig[,-3], Tmax=Tmaxnum)
    Ginisurv2 <- Gcensor2(dataset2Orig[,-3], Tmax=Tmaxnum)
    Ginis1Orig<- Ginisurv1$GTmax
    Ginis2Orig<- Ginisurv2$GTmax
    teststatGiniOrig<-(Ginis1Orig - Ginis2Orig)^2
    if (asymptotic==TRUE){
        VarasintGini1 <- VarGinicensor(dataset1Orig[,-3], Tmax1)
        VarasintGini2 <- VarGinicensor(dataset2Orig[,-3], Tmax2)
        teststatGiniAs<-(Ginis1Orig - Ginis2Orig)/sqrt(VarasintGini1 + VarasintGini2)
        pGiniAs<-1-pchisq((teststatGiniAs^2), df=1)
    }
    teststatGiniPerm<-rep(NA, M)
    if (permutation==TRUE){
        for(msims in 1:M){
            label<-sample(Tx)
            dataset1<-matrix(ncol=2, nrow=N1)
            dataset2<-matrix(ncol=2, nrow=N2)
            dataset1[,1]<-datasetOrig[label==1,1]
            dataset1[,2]<-datasetOrig[label==1,2]
            dataset2[,1]<-datasetOrig[label==2,1]
            dataset2[,2]<-datasetOrig[label==2,2]
          if (lastEvent==1) A1Perm<-dataset1[,1]*1*(dataset1[,2]==1) else A1Perm<-dataset1[,1]
          if (lastEvent==1) A2Perm<-dataset2[,1]*1*(dataset2[,2]==1) else A2Perm<-dataset2[,1]
            Tmax1Perm<-max(A1Perm)+0.001
            Tmax2Perm<-max(A2Perm)+0.001
            TmaxnumPerm<-Tmax1Perm*1*(N1>=N2)+Tmax2Perm*1*(N2>N1)
            Ginisurv1Perm <- Gcensor2(dataset1, Tmax=TmaxnumPerm)
            Ginisurv2Perm <- Gcensor2(dataset2, Tmax=TmaxnumPerm)
            Ginis1Perm<- Ginisurv1Perm$GTmax
            Ginis2Perm<- Ginisurv2Perm$GTmax
            teststatGiniPerm[msims]<-(Ginis1Perm-Ginis2Perm)^2
        }
        pGiniPerm<- sum( 1*(teststatGiniPerm > teststatGiniOrig) )/M
    }
    if (linearRank==TRUE){
        datatemp <- list(X=X,delta=delta, Tx=Tx)
        teststatGT <- survdiff(Surv(X,delta)~Tx,datatemp,rho=-1)$chisq
        teststatLR <- survdiff(Surv(X,delta)~Tx,datatemp,rho=0)$chisq
        teststatW <- survdiff(Surv(X,delta)~Tx,datatemp,rho=1)$chisq
        pLR<-1-pchisq(teststatLR, df=1)
        pGT<-1-pchisq(teststatGT, df=1)
        pW<-1-pchisq(teststatW, df=1)
    }
    if (permutation==TRUE & asymptotic==TRUE & linearRank==TRUE)
print(c(pGiniAs=pGiniAs, pGiniPerm=pGiniPerm, pGT=pGT,pLR=pLR,pW=pW), digits=5)
    else{
        if (permutation==FALSE & asymptotic==TRUE & linearRank==TRUE)
            print(c(pGiniAs=pGiniAs, pGT=pGT,pLR=pLR,pW=pW), digits=5)
        if (permutation==TRUE & asymptotic==FALSE & linearRank==TRUE)
            print(c(pGiniPerm=pGiniPerm, pGT=pGT,pLR=pLR,pW=pW), digits=5)
        if (permutation==TRUE & asymptotic==TRUE & linearRank==FALSE)
            print(c(pGiniAs=pGiniAs, pGiniPerm=pGiniPerm), digits=5)
        if (permutation==FALSE & asymptotic==FALSE & linearRank==TRUE)
            print(c(pGT=pGT,pLR=pLR,pW=pW), digits=5)
        if (permutation==FALSE & asymptotic==TRUE & linearRank==FALSE)
            print(c(pGiniAs=pGiniAs), digits=5)
        if (permutation==TRUE & asymptotic==FALSE & linearRank==FALSE)
            print(c(pGiniPerm=pGiniPerm ), digits=5)
        if (permutation==FALSE & asymptotic==FALSE & linearRank==FALSE)
            print("No test chosen", digits=5)
    }
  }
```

Index