R Codes for

"Sample Size Determination for Stratified Phase II Cancer Trials with Monotone Order Constraints"

1 Functions

Required library: MASS,mvtnorm,survival,quantreg,splines,xtable,quadprog.

Input variables are

- p0 for $\boldsymbol{p}_0 = (p_{10}, \ldots, p_{S0})'$ a vector of the response rates under H_0,
- p for a vector of the response rates $\boldsymbol{p} = (p_1, \ldots, p_S)'$,
- r for the number of responders $\boldsymbol{r} = (r_1, \ldots, r_S)'$
- n the number of patients $(n_1, \ldots, n_S)'$ where patients are stratified into S strata.

Function QP(p0,r,n) returns $\boldsymbol{p}$ which is the solution for $\sup_{p\in\Theta} l(\boldsymbol{p})$ through solving the following constrained convex optimization problem

$$\begin{array}{ll} \min_{\boldsymbol{p}} & -\log l(\boldsymbol{p}) \\ \text{subject to} & 0 \le p_1 \le \cdots \le p_S \le 1 \\ & p_s \ge p_{s0},\ s = 1, \ldots, S. \end{array}$$

```
QP<-function(p0,r,n){
    p_bar<-r/n
    S<-length(p0)
    Dmat<-matrix(0,S,S)
    diag(Dmat)<-n
    dvec<-r
    A_upper<-diag(S)
    A_lower<-matrix(0,nrow = (S-1),ncol = S)
    for(i in 1:(S-1)){
        A_lower[i,i:(i+1)]<-c(-1,1)
    }
    Amat<-rbind(A_upper,A_lower)
    Amat<-t(Amat)
    b_vec<-c(p0,numeric(S-1))
    solve.QP(Dmat,dvec,Amat,b_vec)}
```

Function log_like(p,r,n) returns

$$log\left(\prod_{s=1}^{S} p_s^{r_s}(1-p_s)^{n_s-r_s}\right).$$

```
log_like<-function(p,r,n){
    prod_res<-prod(p^r*(1-p)^(n-r))
    res<-log(prod_res)
    return(res)}
```

2 One stage design

Input variables:

- p for a vector of the response rates $\boldsymbol{p} = (p_1, \ldots, p_S)'$,
- tmpN for a total of N patients
- tmpP for the proportions of patients $(\pi_1, \ldots, \pi_S)'$ where patients are stratified into S strata

Function rep_test(p,tmpN,tmpP) returns our likelihood ratio test statistics,

$$T = -2 \log \left\{ \frac{\sup_{\boldsymbol{p} \in \Theta_0} l(\boldsymbol{p})}{\sup_{\boldsymbol{p} \in \Theta} l(\boldsymbol{p})} \right\},$$

where

$$l(\boldsymbol{p}) = \frac{N!}{n_1! \ldots n_S!} \left(\prod_{s=1}^{S} \pi_s^{n_s} \right) \left\{ \prod_{s=1}^{S} \binom{n_s}{r_s} p_s^{r_s} (1-p_s)^{n_s - r_s} \right\}$$

is the likelihood function.

```
rep_test<-function(p,tmpN,tmpP){
#S is simulation times for an empirical distribution of T.#
S=20000
LRT<-rep(0,S)
LRTstat=matrix(0,nrow=S,ncol=k)
#k is the number of stratum, which should be defined in advance#
na.count<-0
bad_mle.count<-0
for (j in 1:S){
Nk=rmultinom(1, tmpN, tmpP)
Rk=rep(0,k)
Lk0=rep(0,k)
for (i in 1:k){
Rk[i]=rbinom(n=1,Nk[i],p[i])
}
Lk0<-Rk/Nk
#count of how many times N=0
na.ind<-is.na(Lk0)
if(length(which(na.ind==TRUE))>0) na.count<-na.count+1
Rk<-Rk[!na.ind]
Nk<-Nk[!na.ind]
Lk0<-Lk0[!na.ind]
```

```
if(length(which(na.ind==FALSE))==1){Lk_res_final=Lk0}else{
Lk_res_final=rev(QP(rev(p0[!na.ind]),rev(Rk),rev(Nk))$solution)
}
LRT[j]<-(-2)*(log_like(p0[!na.ind],Rk,Nk)
       -log_like(Lk_res_final,Rk,Nk))
LRTstat[j,!na.ind]= (-2)*(log(p0[!na.ind]^Rk*(1-p0[!na.ind])^(Nk-Rk))
       -log(Lk_res_final^Rk*(1-Lk_res_final)^(Nk-Rk)))
if(LRT[j]<0) bad_mle.count<-bad_mle.count+1
}
return(list(test_new=LRT,na.count=na.count))}
#test_new is 20000 test statistics for empirical distribution of T#
```

Sample size calculation for the case of $S = 2$, $\alpha = 0.05$, $\beta = 0.2$, $\pi_1 = 0.2$, $\pi_2 = 0.8$, $p_{10} = p_{20} = 0.2$, $\Delta_1 = 0.05$ and $\Delta_2 = 0.2$.

```
##SETTING
alpha=0.05
beta=0.2
k=2 #number of stratum#
delta=c(0.2,0.05)#(Delta2,Delta1)#
p0=c(0.2,0.2)#(p10,p20)#
p1=p0+delta
N0v=c()
N1v=c()
P=c(0.8,0.2) #(pi2,pi1)#

##Initialize
Q=0
for (i in 1:k){Q=Q+(p0[i])*P[i]*(log(p0[i]/p1[i])-log((1-p0[i])
                /(1-p1[i])))+P[i]*log((1-p0[i])/(1-p1[i]))}
N1=ceiling((qchisq(1-alpha, k)-qchisq(beta,k))/(2*Q))
power1=0
sign1=0
N_vec=c()
power1_vec=c()
sign1_vec=c()
n=N1-10

##Sample size calculation
while (power1<1-beta) {
tmp_H0<-rep_test(p0,n,P)
test_new_H0<-tmp_H0$test_new
t1=quantile(sort(test_new_H0),1-alpha)
sign1=sum(test_new_H0>t1)/S
tmp_H1<-rep_test(p1,n,P)
test_new_H1<-tmp_H1$test_new
power1=sum(test_new_H1>t1)/S
N_vec=c(N_vec,n)
power1_vec=c(power1_vec,power1)
```

```
sign1_vec=c(sign1_vec,sign1)
n=n+1}

##Output
Tab=cbind(N_vec,sign1_vec,power1_vec)
Tab[min(which(power1_vec>=(1-beta))),c(1,2,3)]
#N, \hat\alpha, 1-\hat\beta#
#N is total sample size#
```

3 Two stage design

Input variables:

- p for a vector of the response rates $\boldsymbol{p} = (p_1, \ldots, p_S)'$,
- tmpN1 for a total of N patients in the first stage,
- tmpN2 for a total of N patients in the second stage
- tmpP for the proportions of patients $(\pi_1, \ldots, \pi_S)'$ where patients are stratified into S strata

Function rep_test(p,tmpN1,tmpN2,tmpP) returns our likelihood ratio test statistics $T^{(1)}$ and T.

```
rep_test2<-function(p,tmpN1,tmpN2,tmpP){
#S is simulation times for an empirical distribution of T1 and T.#
S=20000
LRT_1<-rep(0,S)
LRT_T<-rep(0,S)
for (j in 1:S){
Nk_1=rmultinom(1, tmpN1, tmpP)
Nk_2=rmultinom(1, tmpN2, tmpP)
Nk_T=Nk_1+Nk_2
#k is the number of stratum, which should be defined in advance#
Rk_1=rep(0,k)
Rk_2=rep(0,k)
Lk_1=rep(0,k)
Lk_2=rep(0,k)
for (i in 1:k){
Rk_1[i]=rbinom(n=1,Nk_1[i],p[i])
Rk_2[i]=rbinom(n=1,Nk_2[i],p[i])}
Rk_T=Rk_1+Rk_2
Lk_1<-Rk_1/Nk_1
Rk_1<-Rk_1[!na.ind]
Nk_1<-Nk_1[!na.ind]
Lk_1<-Lk_1[!na.ind]
if(length(which(na.ind==FALSE))==1){Lk_res_final_1=Lk_1}else{
Lk_res_final_1=rev(QP(rev(p0[!na.ind]),rev(Rk_1),rev(Nk_1))$solution)}
LRT_1[j]<-(-2)*(log_like(p0[!na.ind],Rk_1,Nk_1)
            -log_like(Lk_res_final_1,Rk_1,Nk_1))
Lk_T<-Rk_T/Nk_T
Rk_T<-Rk_T[!na.ind]
```

```
Nk_T<-Nk_T[!na.ind]
Lk_T<-Lk_T[!na.ind]
if(length(which(na.ind==FALSE))==1){Lk_res_final_T=Lk_T}else{
Lk_res_final_T=rev(QP(rev(p0[!na.ind]),rev(Rk_T),rev(Nk_T))$solution)}
LRT_T[j]<-(-2)*(log_like(p0[!na.ind],Rk_T,Nk_T)
          -log_like(Lk_res_final_T,Rk_T,Nk_T))}
return(list(test_stage1=LRT_1,test_T=LRT_T))}
#test_stage1 is 20000 test statistics for empirical distribution of T1#
#test_T is 20000 test statistics for empirical distribution of T#
```

Sample size calculation for the case of $S = 2$, $\alpha = 0.05$, $\beta = 0.2$, $\pi_1 = 0.2$, $\pi_2 = 0.8$, $p_{10} = p_{20} = 0.2$, $\Delta_1 = 0.05$ and $\Delta_2 = 0.2$.

```
##Setting
alpha=0.05
beta=0.2
k=2 #number of stratum#
delta=c(0.2,0.05)#(Delta2,Delta1)#
p0=c(0.2,0.2)#(p10,p20)#
p1=p0+delta
N0v=c()
N1v=c()
P=c(0.8,0.2) #(pi2,pi1)#

##Initialize
Q=0
for (i in 1:k){Q=Q+(p0[i])*P[i]*(log(p0[i]/p1[i])-log((1-p0[i])
/(1-p1[i])))+P[i]*log((1-p0[i])/(1-p1[i]))}
N1=ceiling((qchisq(1-alpha, k)-qchisq(beta,k))/(2*Q))
n=N1-10
r=numeric(0))
ind_M=1

##Sample size calculation
M=data.frame(c1=numeric(0),c=numeric(0),n1=numeric(0),n=numeric(0)
  ,EN_1=numeric(0),pet=numeric(0),sig=numeric(0),power=numeric(0))
ind_n=0
while (ind_n==0) {
ind_M_n=0
M_tmp=data.frame(c1=numeric(0),c=numeric(0),n1=numeric(0),n=numeric(0)
      ,EN_1=numeric(0),pet=numeric(0),sig=numeric(0),power=numeric(0))
for (n1 in 1:(n-1)){
tmp_H0<-rep_test2(p0,n1,n-n1,P)
test_H0_1<-tmp_H0$test_stage1
test_H0_T<-tmp_H0$test_T
tmp_H1<-rep_test2(p1,n1,n-n1,P)
test_H1_1<-tmp_H1$test_stage1
test_H1_T<-tmp_H1$test_T
for (pet in rev(PET)){
```

```
c1= quantile(sort(test_H0_1),pet)
c=quantile(sort(test_H0_T[which(test_H0_1>c1)]),1-alpha/(1-pet))
power=sum(test_H1_1[which(test_H1_T>c)]>c1)/S
if (power< 1-beta) {} else {
EN_1=n1+(1-pet)*(n-n1)
sig=sum(test_H0_1[which(test_H0_T>c)]>c1)/S
ind_M_n=ind_M_n+1
M[ind_M,]=c(c1,c,n1,n,EN_1,pet,sig,power)
M_tmp[ind_M_n,]=c(c1,c,n1,n,EN_1,pet,sig,power)
write.table(M,append=F,col.names=F,row.names = F,"M.R")
ind_M=ind_M+1
ind_n=ind_n+1
break}
}
}
if (ind_M_n>0){
minmax=M_tmp[which(M_tmp[,5]==min(M_tmp[,5])),c(3,4,5,7,8)]}
n=n+1}
EN_min=minmax
ind_EN=0
while (ind_EN==0) {
ind_M_n=0
M_tmp=data.frame(c1=numeric(0),c=numeric(0),n1=numeric(0),n=numeric(0)
      ,EN_1=numeric(0),pet=numeric(0),sig=numeric(0),power=numeric(0))
for (n1 in 1:(n-1)){
tmp_H0<-rep_test2(p0,n1,n-n1,P)
test_H0_1<-tmp_H0$test_stage1
test_H0_T<-tmp_H0$test_T
tmp_H1<-rep_test2(p1,n1,n-n1,P)
test_H1_1<-tmp_H1$test_stage1
test_H1_T<-tmp_H1$test_T
for (pet in rev(PET)){
c1= quantile(sort(test_H0_1),pet)
c=quantile(sort(test_H0_T[which(test_H0_1>c1)]),1-alpha/(1-pet))
power=sum(test_H1_1[which(test_H1_T>c)]>c1)/S
if (power< 1-beta) {} else {
EN_1=n1+(1-pet)*(n-n1)
sig=sum(test_H0_1[which(test_H0_T>c)]>c1)/S
ind_M_n=ind_M_n+1
M[ind_M,]=c(c1,c,n1,n,EN_1,pet,sig,power)
M_tmp[ind_M_n,]=c(c1,c,n1,n,EN_1,pet,sig,power)
write.table(M,append=F,col.names=F,row.names = F,"M.R")
ind_M=ind_M+1
break}
}
}
if (ind_M_n>0){
EN_now=M_tmp[which(M_tmp[,5]==min(M_tmp[,5])),5]
if (EN_now<EN_min) {
```

```
ind_EN=0
EN_min=EN_now
optimal=M_tmp[which(M_tmp[,5]==min(M_tmp[,5])),c(3,4,5,7,8)]
n=n+1}else{ind_EN=1}
}
}

##Output
minmax
#N^{(1)}, N, EN(\bfp_0), \hat\alpha, 1-\hat\beta of minmax design#
optimal
#N^{(1)}, N, EN(\bfp_0), \hat\alpha, 1-\hat\beta of optimal design#
#N^{(1)} is sample size in the frist stage#
#N is total sample size#
```