

Crossover designs with two binary endpoints

Web appendix

Supplementary Tables

Table A.1: Type I error (in %) of the significance test for crossover design with two binary endpoints when $n = 100$ or 200 , where $p_{j,01}^{(1)} = p_{j,01}^{(2)}$ and $\theta_j^{(1)}$ are set the same for both endpoints, r_h is set the same for both sequence groups, and $\rho_1 = 1$. (i.e. H_{01} is true.) The actual type I errors are obtained by the rejection rate after 10,000 replications based on (7) and the approximated type I errors are obtained by numerical evaluation of (8) using the function `pmvnorm` of the R package `mvtnorm`.

parameters				$n = 100$		$n = 200$	
$p_{j,01}^{(h)}$	$\theta_j^{(1)}$	$\log \rho_2$	r_h	act.	app.	act.	app.
0.1	1.2	1.5	0	2.6	1.1	4.3	1.9
			0.8	2.7	1.4	3.7	2
			2	3.5	1.6	4.8	2.3
		1.25	0	3.1	1.8	4.1	2.4
			0.8	2.4	1.1	3.8	1.9
			2	2.7	1.5	3	2.1
0.15	1	1.5	0	3.2	1.5	4.4	2.2
			0.8	3.4	1.8	3.9	2.3
			2	3	1.4	4.2	2.1
		1.25	0	2.7	1.7	3.8	2.3
			0.8	3.8	1.8	5	2.4
			2	3.3	2	4.3	2.4
0.2	0.8	1.5	0	4.1	1.8	5.2	2.3
			0.8	3.7	2	4.6	2.4
			2	5.2	2.3	5.4	2.5
		1.25	0	4.9	2.4	5.4	2.5
			0.8				
			2				
0.25	0.4	1.5	0				
			0.8				
			2				
		1.25	0				
			0.8				
			2				

Table A.2: Type I error (in %) of the significance test for crossover design with two binary endpoints when $n = 100$ or 200 , where $p_{j,01}^{(1)} = p_{j,01}^{(2)}$ and $\theta_j^{(1)}$ are set the same for both endpoints, r_h is set the same for both sequence groups, and $\rho_1 = \rho_2 = 1$. (i.e. both H_{01} and H_{02} are true.) The actual type I errors are obtained by the rejection rate after 10,000 replications based on (7) and the approximated type I errors are obtained by numerical evaluation of (8) using the function `pmvnorm` of the R package `mvtnorm`.

parameters			$n = 100$		$n = 200$	
$p_{j,01}^{(h)}$	$\theta_j^{(1)}$	r_h	act.	app.	act.	app.
0.1	1.2	0	0.3	0.1	0.3	0.1
		0.8	1.1	0.2	1.1	0.2
0.15	1	0	0.3	0.1	0.3	0.1
		0.8	1.2	0.2	0.8	0.2
0.15	1.2	0	0.3	0.1	0.3	0.1
		0.8	1.1	0.2	1	0.2
0.2	0.8	0	0.4	0.1	0.3	0.1
		0.8	0.8	0.2	0.8	0.2
0.25	0.4	0	0.3	0.1	0.3	0.1
		0.8	0.6	0.2	0.6	0.2

Table A.3: Power (in %) and sample size of the significance test for crossover design with two binary endpoints when $n = 30$ or 50 and the nominal power is 80%, where $p_{j,01}^{(1)} = p_{j,01}^{(2)}$, $\theta_j^{(1)}$ and ρ_j are set the same for both endpoints, r_h is set the same for both sequence groups. The actual power are obtained by the rejection rate after 10,000 replications based on (7) and the approximated powers are obtained by numerical evaluation of (8) using the function `pmvnorm` of the R package `mvtnorm`.

parameters				$n = 30$		$n = 50$		power 80%	
$p_{j,01}^{(h)}$	$\theta_j^{(1)}$	ρ_j	r_h	act.	app.	act.	app.	n_0	act.
0.2	0.1	3	0	16.5	35.1	50	55.6	86	83.3
			0.8	27.1	35.8	56.3	56.1	86	86.3
	0.15	3	0	27	38.8	59.4	62.9	72	81.3
			0.8	37	39.7	66.4	63.4	72	84
0.25	0.1	2.5	0	10.5	26.9	35.8	44.9	108	83.5
			0.8	18	27.7	41.7	45.6	108	85.6
	0.2	2.5	0	26.3	35.2	57.7	60	74	81.1
			0.8	37	36.6	64	60.8	74	83

Supplementary R Codes

[1] 'rbinary.R' for generating correlated bivariate binary variables

```
rbinary <-function(n,p,rho)
{
# generate n bivariate binary row vectors with mean p and correlation rho
# input:
#     n: number of rows
#     p: 1 by 2 mean vector of bivariate variables
#     rho: correlation of bivariate variables
# reference:
#     Qaqish, B. F. (2003). A family of multivariate binary distributions for
#     simulating correlated binary variables with specified marginal means
#     and correlations. Biometrika 90, 455–463.
#
# example:
# n=10; p=c(0.4,0.5); rho=0.5
# y <- rbinary(10,c(0.4,0.5),0.5)

b=rho*sqrt(p[2]*(1-p[2])/p[1]/(1-p[1]))
Y=matrix(0,n,2)
Y[,1]=rbinom(n,1,p[1])
Y[,2]=rbinom(n,1,p[2]+b*(Y[,1]-p[1]))
return(Y)
}
```

[2] 'Mainland_Gart_n.R' for sample size calculation with bivariate binary endpoints

```

Mainland_Gart_n <- function(alpha,beta,p01_1,p01_2,r,theta1,rho)
# return the sample size needed for bivairate Mainland-Gart test
# input:
#   alpha:   type-I error
#   beta:    type-II error
#   p01_1:   1 by 2 vector of baseline probabilities in sequence group AB
#   p01_2:   1 by 2 vector of baseline probabilities in sequence group BA
#   r:       1 by 2 vector of correlation coefficient between baseline events
#   theta1:  1 by 2 vector of p01_1/p10_1
#   rho:     1 by 2 vector of the ratio of the odds

#
# example:
#   alpha=0.05;beta=0.2;p01_1=c(0.05,0.1); p01_2=c(0.25,0.3); r=c(0.5,0.5)
#   theta1=c(5,3); rho=c(1/25,1/9)
#   n0= Mainland_Gart_n(alpha,beta,p01_1,p01_2,r,theta1,rho)
{
  # preliminary
  omega=sqrt(((1+theta1)/(theta1*p01_1)+(1+theta1*rho)/(p01_2*rho*theta1))
  #omega under true model
  p01_1_bar=(1+theta1)*p01_1*(p01_1+p01_2)/((1+theta1)*p01_1+(1+rho*theta1)*p01_2)
  # asymt. Limit of p01_1
  p01_2_bar=(1+rho*theta1)*p01_2*(p01_1+p01_2)/((1+theta1)*p01_1+(1+rho*theta1)*p01_2)
  # asymt. Limit of p01_2
  p10_1_bar=(1+theta1)*theta1*p01_1*(p01_1+rho*p01_2)/((1+theta1)*p01_1+(1+rho*theta1)*p01_2)
  # asymt. Limit of p10_1
  p10_2_bar=(1+rho*theta1)*theta1*p01_2*(p01_1+rho*p01_2)/((1+theta1)*p01_1+(1+rho*theta1)*p01_2)
  # asymt. Limit of p10_2

  omega_bar=sqrt(1/p01_1_bar+1/p10_1_bar+1/p01_2_bar+1/p10_2_bar)
  # asymt. Limit of omega
  b=log(rho)/omega

  Z1Z2_rho=(r[1]/sqrt(t(p01_1)%*(1-p01_1))+r[2]/sqrt(t(p01_2)%*(1-p01_2)))/sqrt((
  1+theta1[1])/theta1[1]/p01_1[1]+(1+theta1[1]*rho[1])/theta1[1]/rho[1]/p01_2[1])/sqrt((
  1+theta1[2])/theta1[2]/p01_1[2]+(1+theta1[2]*rho[2])/theta1[2]/rho[2]/p01_2[2])
  corr=matrix(c(1,Z1Z2_rho,Z1Z2_rho,1),2)      # corr. Coef. of pivotal stats.

```

```

# a subroutine
fp <- function (n)
{ lower_1=qnorm(1-alpha/2)*omega_bar/omega-sqrt(n/2)*b
  upper_2=qnorm(alpha/2)*omega_bar/omega-sqrt(n/2)*b

  pmvnorm(lower=lower_1,upper=Inf,mean=c(0,0),corr)[1]+pmvnorm(lower=-Inf,upper
    =upper_2,mean=c(0,0),corr)[1]-1+beta
} # eq. to be solved

n0=ceiling(uniroot(fp,c(5,100000), tol = 0.0001)$root) # solve eq.
return(2*ceiling(n0/2))      # since n0 is an even number
}

```

[3] 'Mainland_Gart_actpower.R' for actual power calculation with bivariate binary endpoints

```
Mainland_Gart_act_power=function(alpha,p01_1,p01_2,r,theta1,rho,n,simuN)
```

```
# return the actual power for bivairate Mainland-Gart test
```

```
# input:
```

```
#   alpha:   type-I error
```

```
#   p01_1:   1 by 2 vector of baseline probabilities in sequence group AB
```

```
#   p01_2:   1 by 2 vector of baseline probabilities in sequence group BA
```

```
#   r:       1 by 2 vector of correlation coefficient between baseline events
```

```
#   theta1:  1 by 2 vector of p01_1/p10_1
```

```
#   rho:      1 by 2 vector of the ratio of the odds
```

```
#   n:       sample size
```

```
#   simuN:   number of simulation
```

```
# example:
```

```
#   alpha=0.05;p01_1=c(0.1,0.1);p01_2=c(0.1,0.1)
```

```
#   r=c(0.8,0.8);theta1=c(1.1,1.1);rho=c(3.5,3.5);n=200;simuN=1000
```

```
#   resP=Mainland_Gart_act_power(alpha,p01_1,p01_2,r,theta1,rho,n,simuN)
```

```
{
```

```
# preliminary
```

```
z_alpha=qnorm(1-alpha/2);
```

```
condiP10_1=p01_1*theta1/(1-p01_1)      # q_j, the conditional prob
```

```
condiP10_2=p01_2*theta1*rho/(1-p01_2)
```

```
# simulations
```

```
cT=matrix(0,simuN,2);cPower=matrix(0,simuN,1)
```

```
rejRate=0;
```

```
for (i in 1:simuN)
```

```
{
```

```
# generate the bivariate data in group AB
```

```
xi01_1=rbinary(ceiling(n/2),p01_1,r[1]);
```

```
xi10_1=t(rbind(rbinom(ceiling(n/2),1,condiP10_1[1]),rbinom(ceiling(n/2),1,condiP10_1[2]))*(1-xi01_1);
```

```
# generate the bivariate data in group BA
```

```
xi01_2=rbinary(ceiling(n/2),p01_2,r[2]);
```

```
xi10_2=t(rbind(rbinom(ceiling(n/2),1,condiP10_2[1]),rbinom(ceiling(n/2),1,condiP10_2[2]))*(1-xi01_2);
```



```

2[2]))*(1-xi01_2);

# the freq. of discor. pairs in group AB
x10_1=colSums(xi10_1);x01_1=colSums(xi01_1);
# the freq. of discor. pairs in group BA
x10_2=colSums(xi10_2);x01_2=colSums(xi01_2);

# MLEs

p01MLE_1=(x10_1+x01_1+.5)*(x01_1+x01_2+.5)/ceiling(n/2)/(x01_1+x10_1+x01_
2+x10_2+.5)    # MLE of p01_1 under H_0

p01MLE_2=(x10_2+x01_2+.5)*(x01_1+x01_2+.5)/ceiling(n/2)/(x01_1+x10_1+x01_
2+x10_2+.5)    # MLE of p01_2 under H_0

p10MLE_1=(x10_1+x01_1+.5)*(x10_1+x10_2+.5)/ceiling(n/2)/(x01_1+x10_1+x01_
2+x10_2+.5)    # MLE of p10_1 under H_0

p10MLE_2=(x10_2+x01_2+.5)*(x10_1+x10_2+.5)/ceiling(n/2)/(x01_1+x10_1+x01_
2+x10_2+.5)    # MLE of p10_2 under H_0

omega_tilde=sqrt(1/p01MLE_1+1/p10MLE_1+1/p10MLE_2+1/p01MLE_2)    #
The MLE of omega under H_0

# the test statistics t_j

cT[i,]=sqrt(ceiling(n/2))*(log((x01_1+.5)/(x10_1+.5))-log((x01_2+.5)/(x10_2+.5)))/ome
ga_tilde
}

rejRate=mean(apply(abs(cT)>z_alpha,1,all))
}

```

[4] 'Mainland_Gart_app_power.R' for approximate power calculation with bivariate binary endpoints

```
Mainland_Gart_app_power <- function(alpha,beta,p01_1,p01_2,r,theta1,rho,n)
# return the approximate power for bivairate Mainland-Gart test

# input:
#   alpha:   type-I error
#   beta:    type-II error
#   p01_1:   1 by 2 vector of baseline probabilities in sequence group AB
#   p01_2:   1 by 2 vector of baseline probabilities in sequence group BA
#   r:       1 by 2 vector of correlation coefficient between baseline events
#   theta1:  1 by 2 vector of p01_1/p10_1
#   rho:     1 by 2 vector of the ratio of the odds

# example:
#
#   alpha=0.05;beta=0.2;p01_1=c(0.1,0.1);p01_2=c(0.1,0.1);r=c(0.8,0.8);theta1=c(1.1,1.
#   1);rho=c(3,3);n=200
#   Mainland_Gart_app_power(alpha,p01_1,p01_2,r,theta1,rho,n)

{
# preliminary
omega=sqrt((1+theta1)/(theta1*p01_1)+(1+theta1*rho)/(p01_2*rho*theta1))      #
omega under ture model

p01_1_bar=(1+theta1)*p01_1*(p01_1+p01_2)/((1+theta1)*p01_1+(1+rho*theta1)*p0
1_2)      # asymt. Limit of p01_1

p01_2_bar=(1+rho*theta1)*p01_2*(p01_1+p01_2)/((1+theta1)*p01_1+(1+rho*theta1)
*p01_2)      # asymt. Limit of p01_2

p10_1_bar=(1+theta1)*theta1*p01_1*(p01_1+rho*p01_2)/((1+theta1)*p01_1+(1+rho
*theta1)*p01_2)      # asymt. Limit of p10_1

p10_2_bar=(1+rho*theta1)*theta1*p01_2*(p01_1+rho*p01_2)/((1+theta1)*p01_1+(1
+rho*theta1)*p01_2)      # asymt. Limit of p10_2

omega_bar=sqrt(1/p01_1_bar+1/p10_1_bar+1/p01_2_bar+1/p10_2_bar)
# asymt. Limit of omega
b=log(rho)/omega

Z1Z2_rho=(r[1]/sqrt(t(p01_1)%*%(1-p01_1))+r[2]/sqrt(t(p01_2)%*%(1-p01_2)))/sqrt((
```

```

1+theta1[1])/theta1[1]/p01_1[1]+(1+theta1[1]*rho[1])/theta1[1]/rho[1]/p01_2[1])/sqrt((
1+theta1[2])/theta1[2]/p01_1[2]+(1+theta1[2]*rho[2])

/theta1[2]/rho[2]/p01_2[2])
corr=matrix(c(1,Z1Z2_rho,Z1Z2_rho,1),2)      # corr. Coef. of pivotal stats.

lower_1=qnorm(1-alpha/2)*omega_bar/omega-sqrt(n/2)*b
upper_2=qnorm(alpha/2)*omega_bar/omega-sqrt(n/2)*b

app=pmvnorm(lower=lower_1,upper=Inf,mean=c(0,0),corr)[1]+pmvnorm(lower=-Inf,u
pper=upper_2,mean=c(0,0),corr)[1]      # the app. power
return(app)
}

```

[5] 'Mainland_Gart_adp_n.R' for sample size needed for bivariate Mainland-Gart Test

```
Mainland_Gart_adp_n <- function(gamma,ec,corr,alpha,beta)
  # return sample size needed for bivariate Mainland Test

{
  # Power function
  fp <- function (n)
  {lower_1=qnorm(1-alpha/2)*gamma-sqrt(n/2)*ec
    upper_2=qnorm(alpha/2)*gamma-sqrt(n/2)*ec

    pmvnorm(lower=lower_1,upper=Inf,mean=c(0,0),corr)[1]+pmvnorm(lower=-Inf,upper=upper_2,mean=c(0,0),corr)[1]-1+beta

  } # eq. to be solved

  n0=ceiling(uniroot(fp,c(5,10000), tol = 0.0001)$root) # solve eq.
  return(2*ceiling(n0/2)) # since n0 is an even number
}
```

[6] 'Mainland_Gart_adp.R' for actual 2-stage rejection rate, N and std(N)

```
Mainland_Gart_adp=function(alpha,beta,p01_1,p01_2,r,theta1,rho,n1,simuN)
{
  # return actual 2-stage rejection rate, estimates of E(N) and std(N)

  #  alpha:    type-I error
  #  beta:     type-II error
  #  p01_1:    1 by 2 vector of baseline probabilities in sequence group AB
  #  p01_2:    1 by 2 vector of baseline probabilities in sequence group BA
  #  r:        1 by 2 vector of correlation coefficient between the two endpoints through
               the
               #correlations of the two baseline events.;
  #  theta1:   1 by 2 vector of p01_1/p10_1
  #  rho:      1 by 2 vector of the ratio of the odds under H1;

  #  example:
  #  alpha=0.05;beta=0.2
  #  p01_1=c(0.2,0.2);p01_2=c(0.2,0.2);r=c(0.8,0.8);theta1=c(0.5,0.5);
  #  rho=c(7.389056,7.389056);n1=100;simuN=1000;
  #  Mainland_Gart_adp(alpha,beta,p01_1,p01_2,r,theta1,rho,n1,simuN)

  # initial values
  #alpha=0.05; beta=0.2;
  z_alpha=qnorm(1-alpha/2)
  condiP10_1=p01_1*theta1/(1-p01_1)      # q_j, the conditional prob
  condiP10_2=p01_2*theta1*rho/(1-p01_2)

  # simu
  cT_b=matrix(0,simuN,2);cn_b=matrix(0,simuN,1)

  for (i in 1:simuN)
  {
    # generate 1-stage data
    # 1-stage data in group AB
    xi01_11=rbinary(ceiling(n1/2),p01_1,r[1]);

    xi10_11=t(rbind(rbinom(ceiling(n1/2),1,condiP10_1[1]),rbinom(ceiling(n1/2),1,condiP
    10_1[2])))*(1-xi01_11);
    # 1-stage data in group BA
    xi01_21=rbinary(ceiling(n1/2),p01_2,r[2]);

    xi10_21=t(rbind(rbinom(ceiling(n1/2),1,condiP10_2[1]),rbinom(ceiling(n1/2),1,condiP
    10_2[2])))*(1-xi01_21);
```

```
## start of blind version of adaptation ##
```

```
#re-estiamte sample size
```

```
er=0;      #initial er
```

```
icov1=cov(xi01_11-xi10_11)
```

```
if (all(diag(icov1)>0)) {er1=cov2cor(icov1)[1,2]}# ensure valid updating
```

```
icov2=cov(xi01_21-xi10_21)
```

```
if (all(diag(icov2)>0)) {er2=cov2cor(icov2)[1,2]}# ensure valid updating
```

```
er=c(er1,er2) # update r
```

```
# esimaor of  $\log(p_{\{j,01\}}^{\{h\}} + \log(p_{\{j,10\}}^{\{h\}}))$ 
```

```
logp1_plus=log(apply(rbind(apply(xi01_11,2,sum),c(0.5,0.5)),2,max)/(n1/2))+log(appl  
y(rbind(apply(xi10_11,2,sum),c(0.5,0.5)),2,max)/(n1/2))
```

```
logp2_plus=log(apply(rbind(apply(xi01_21,2,sum),c(0.5,0.5)),2,max)/(n1/2))+log(appl  
y(rbind(apply(xi10_21,2,sum),c(0.5,0.5)),2,max)/(n1/2))
```

```
# esimaor of  $\log(p_{\{j,01\}}^{\{h\}} - \log(p_{\{j,10\}}^{\{h\}}))$ 
```

```
logp1_minus=log(apply(rbind(apply(xi01_11,2,sum),c(0.5,0.5)),2,max)/(n1/2))-log(ap  
ply(rbind(apply(xi10_11,2,sum),c(0.5,0.5)),2,max)/(n1/2))
```

```
# esimaor of  $\log(p_{\{j,01\}}^{\{h\}} - \log(p_{\{j,10\}}^{\{h\}}))$ 
```

```
logp2_minus=log(apply(rbind(apply(xi01_21,2,sum),c(0.5,0.5)),2,max)/(n1/2))+log(ap  
ply(rbind(apply(xi10_21,2,sum),c(0.5,0.5)),2,max)/(n1/2))
```

```
# update theta
```

```
etheta1=exp(logp1_minus)
```

```
etheta2=exp(logp2_minus)
```

```
# update p01
```

```
ep01_1=exp((-log(etheta1)+logp1_plus)/2)
```

```
ep01_2=exp((-log(rho)-log(etheta1)+logp2_plus)/2)
```

```
# update corr
```

```
Z1Z2_rho=(er[1]/sqrt(t(ep01_1))%*(1-ep01_1))+er[2]/sqrt(t(ep01_2))%*(1-ep01_  
2))/
```

```
sqrt((1+etheta1[1])/etheta1[1]/ep01_1[1]+(1+etheta1[1]*rho[1])/etheta1[1]/rho[1]/ep0
```

```

1_2[1])/

sqrt((1+etheta1[2])/etheta1[2]/ep01_1[2]+(1+etheta1[2]*rho[2])/etheta1[2]/rho[2]/ep0
1_2[2])
corr=matrix(c(1,Z1Z2_erho,Z1Z2_erho,1),2)

omega=sqrt((1+etheta1)/(etheta1*ep01_1)+(1+etheta1*rho)/(ep01_2*rho*etheta1))
# est. true omega

#\bar{p}_j,ab^{(i)} : the asymptotic limit of \widetilde{p}_j,ab^{(i)}

ep01_1_bar=(1+etheta1)*ep01_1*(ep01_1+ep01_2)/((1+etheta1)*ep01_1+(1+rho*et
heta1)*ep01_2)

ep01_2_bar=(1+rho*etheta1)*ep01_2*(ep01_1+ep01_2)/((1+etheta1)*ep01_1+(1+rh
o*etheta1)*ep01_2)

ep10_1_bar=(1+etheta1)*etheta1*ep01_1*(ep01_1+rho*ep01_2)/((1+etheta1)*ep01
_1+(1+rho*etheta1)*ep01_2)

ep10_2_bar=(1+rho*etheta1)*etheta1*ep01_2*(ep01_1+rho*ep01_2)/((1+etheta1)*e
p01_1+(1+rho*etheta1)*ep01_2)

#\bar{w}_j: the asymptotic limit of w_j
omega_bar=(sqrt(1/ep01_1_bar+1/ep10_1_bar+1/ep01_2_bar+1/ep10_2_bar)) # est.
omega_bar
gamma=(omega_bar/omega)    # omega_bar/omega
ec=(log(rho)/omega)

N=Mainland_Gart_adp_n(gamma,ec,corr,alpha,beta) # re-estimate N

# generate 2-stage data
n2=max(N-n1,0);n=max(N,n1)
if (n2>0)
{
  xi01_12=rbinary(ceiling(n2/2),p01_1,r[1]);
  xi01_22=rbinary(ceiling(n2/2),p01_2,r[2]);

  xi10_12=t(rbind(rbinom(ceiling(n2/2),1,condiP10_1[1]),rbinom(ceiling(n2/2),1,condiP
10_1[2])))*(1-xi01_12);

  xi10_22=t(rbind(rbinom(ceiling(n2/2),1,condiP10_2[1]),rbinom(ceiling(n2/2),1,condiP
10_2[2])))*(1-xi01_22)

```

```

}else
  {xi01_12=NULL;xi10_12=NULL;xi01_22=NULL;xi10_22=NULL}

# pooled data
xi10_1=rbind(xi10_11,xi10_12);xi01_1=rbind(xi01_11,xi01_12);
xi10_2=rbind(xi10_21,xi10_22);xi01_2=rbind(xi01_21,xi01_22);

x10_1=colSums(xi10_1);x01_1=colSums(xi01_1);x10_2=colSums(xi10_2);x01_2=colSums(xi01_2);

# The MLEs of  $p_j, ab^{(i)}$  under  $H_0$ 

p01MLE_1=(x10_1+x01_1+.5)*(x01_1+x01_2+.5)/ceiling(n/2)/(x01_1+x10_1+x01_2+x10_2+.5)

p01MLE_2=(x10_2+x01_2+.5)*(x01_1+x01_2+.5)/ceiling(n/2)/(x01_1+x10_1+x01_2+x10_2+.5)

p10MLE_1=(x10_1+x01_1+.5)*(x10_1+x10_2+.5)/ceiling(n/2)/(x01_1+x10_1+x01_2+x10_2+.5)

p10MLE_2=(x10_2+x01_2+.5)*(x10_1+x10_2+.5)/ceiling(n/2)/(x01_1+x10_1+x01_2+x10_2+.5)

#the MLE of  $w_j$ :  $\tilde{w}_j$ 
omega_tilde=sqrt(1/p01MLE_1+1/p10MLE_1+1/p10MLE_2+1/p01MLE_2)

#the test statistics  $t_j$  after .5 correction

cT_b[,j]=sqrt(ceiling(n/2))*(log((x01_1+.5)/(x10_1+.5))-log((x01_2+.5)/(x10_2+.5)))/omega_tilde

cn_b[i]=n;

## end of blind version of adaptation ##
}

# end of simu
rejRate_b=mean(apply(abs(cT_b)>z_alpha,1,all))
aveN_b=ceiling(median(cn_b))
stdN_b=mad(cn_b)

return(c(rejRate_b,aveN_b,stdN_b))
}

```