R and WinBUGS codes

1. R code for generation dichotomous outcome

Requires 'MASS' package (https://cran.r-project.org/web/packages/MASS/index.html).

```
# k=number of studies in meta-analysis, tau=heterogeneity variance (τ²),
# B=number of repetitions,
# theta=true overall effect
library (MASS)
# create a function to generate data
generate <-function (tau, theta, k, B) {
# function creates B simulated data sets (treatment effect, within-study
# variance) for dichotomous type of outcome for one scenario with the parameters
# tau, theta and k
# initialize tables
pinakasyi <-matrix (1: k, ncol=1)
pinakasvi <-matrix (1: k, ncol=1)
for (i in 1: B) {
# generate the treatment effect θi for each trial
thi <-rnorm (k, theta, sqrt (tau))
# generate within-study sample sizes for the treatment (T)
# and control  (C) groups, nit and nic, respectively
ni <- sample (20:200, k, replace=T)
nit =nic =ni
# obtain the probability for success in control group
pic <-runif (k, 0.05, 0.65)
# obtain the total number of events ci for the control group
ci <-rbinom (k, nic, pic)
# obtain the probability for success in treatment group
pit =pic*exp (thi) / (1- pic + pic*exp (thi))
# obtain the total number of events αi for treatment group
ai <-rbinom (k, nit, pit)
```

```
# calculate no response bi, di
bi = nit-ai
di = nic-ci
# if any of ai, bi, ci, di is zero put the value 0.5
for (j in 1: k) {
if (ai[j] == 0) {
ai[j] = ai[j] + 0.5
}
if (bi[j] == 0) {
bi[j] = bi[j] + 0.5
}
if (ci[j] == 0) {
ci[j] = ci[j] + 0.5
}
if (di[j] == 0) {
di[j] = di[j] + 0.5
}
}
# calculate the treatment effect log(OR)
yi = log ((ai*di) / (bi*ci))
# estimate the within-study variance
vi = 1/ai + 1/bi + 1/ci + 1/di
pinakasyi <- cbind (pinakasyi, yi)
pinakasvi<- cbind (pinakasvi, vi)
}
# keep data in csv file
write.matrix (pinakasvi, file = "pinakas_vi.csv", sep = ",")
write.matrix (pinakasyi, file = "pinakas_yi.csv", sep = ",")
}
# call function generate to create B simulated data sets (treatment effect, within-study
# variance) for the scenario tau=0, theta=0 and k=10
generate (tau=0, theta=0, k=10, B=1000)
```

2. R code for generation continuous outcome

Requires 'MASS' package (https://cran.r-project.org/web/packages/MASS/index.html).

k=number of studies in meta-analysis, tau=heterogeneity variance (τ^2),

B=number of repetitions,

theta=true overall effect

library(MASS)

create a function to generate data

generate<-function(tau,theta,k,B){

initialize parameters

ni=Azic=Azit=sip=thi=y=s=J=yi=vi=sit=sic=c(NA,k)

initialize tables

pinakasyi<-matrix(1:k,ncol=1)

pinakasvi<-matrix(1:k,ncol=1)

pinakasni<-matrix(1:k,ncol=1)

pinakasJ<-matrix(1:k,ncol=1)

for (i in 1:B){

for(j in 1:k){

generate the treatment effect θ_i for each trial

thi[j] <-rnorm(1,theta,sqrt(tau))

generate within-study sample sizes for the treatment (T) and control (C) groups

nit and nic, respectively

ni[j] <- sample(20:200,1, replace=T)

# simulate n_i observations Z_{ic} for control group with $Z_{ic} \sim N(0,1)$
zic<-rnorm(ni[j],0,1)

# simulate n_i observations Z_{it} for treatment group with $Z_{it} \sim N(\theta_i, 1)$
zit<-rnorm(ni[j],thi[j],1)

Azic[j]=sum(zic)/ni[j]

Azit[j]=sum(zit)/ni[j]

calculate the sample variances for control group and treatment group

sit[j]=(1/(ni[j]-1))*sum((zit-Azit[j])^2)

sic[j]=(1/(ni[j]-1))*sum((zic-Azic[j])^2)

calculate the pooled variance

```
sip[j]=sqrt(((ni[j]-1)*(sit[j])+(ni[j]-1)*(sic[j]))/(2*ni[j]-2))

# calculate the treatment effect and its within-study variance

y[j]=(Azit[j]-Azic[j])/sip[j]

s[j]=(8+(y[j])^2)/(4*ni[j])

# use the correction to avoid biased estimators
J[j]=1-(3/(8*ni[j]-9))

# compute the treatment effect Hedges' g

yi[j]=J[j]*y[j]

# compute the within-study variance

vi[j]=((J[j])^2)*s[j]

}

pinakasyi<-cbind(pinakasyi,yi)

pinakasvi<-cbind(pinakasvi,vi)

pinakasni<-cbind(pinakasni,ni)

pinakasJ<-cbind(pinakasJ,J)

}

# keep data in csv file

write.matrix(pinakasvi, file = "C:/pinakas_vi.csv", sep = ",")

write.matrix(pinakasyi, file = "C:/pinakas_yi.csv", sep = ",")

write.matrix(pinakasni,file="C:/pinakas_ni.csv", sep = ",")

write.matrix(pinakasJ,file="C:/pinakas_J.csv", sep = ",")

}

# call function to create B simulated data sets (treatment effect, within-study variance,

# corrections J and sample sizes n_i)

# for the scenario tau=0, theta=0 and k=10

generate(tau=0,theta=0,k=10,B=1000)
```

3. R code for simulation

```
# An R code for the simulation study which compares the performance of several

# heterogeneity estimators in terms of the assessment criteria.

# This code conducts simulation with dichotomous type of outcome.

# Comments with red color are the differences in code for continuous type of

# outcome.
```

Input: B simulated data sets; treatment effect, within-study variance for

dichotomous outcome

plus corrections J and sample sizes n_i for continuous outcome

Requires ‘metafor’ package (https://cran.r-project.org/web/packages/metafor/)

'R2WinBUGS' package (https://cran.r-project.org/web/packages/R2WinBUGS/)

'boot' package (https://cran.r-project.org/web/packages/boot/)

k=number of studies in meta-analysis, tau=heterogeneity variance (τ^2),

B=number of repetitions,

a=significance level, theta=true overall effect,

theta0=alternative true overall effect of power.

```
# packages
library(metafor)
library(R2WinBUGS)
library(boot)
# do the simulation
simulation<-function(tau,theta,theta0,k,B,a,fileResult,fileyi,filevi,winbugsdir)
# simulation<-
#function(tau,theta,theta0,k,B,a,fileResult,fileyi,filevi,fileni,fileJ,winbugsdir)
{
# initialize heterogeneity estimators
  t1<-rep(NA,B)
  t2<-rep(NA,B)
  t3<-rep(NA,B)
  t4<-rep(NA,B)
  t5<-rep(NA,B)
  t6<-rep(NA,B)
  t7<-rep(NA,B)
  t8<-rep(NA,B)
  t9<-rep(NA,B)
  t10<-rep(NA,B)
  t11<-rep(NA,B)
```

```
  t12<-rep(NA,B)
  t13<-rep(NA,B)
  t14<-rep(NA,B)
  t15<-rep(NA,B)
  t16<-rep(NA,B)
  t17<-rep(NA,B)
  t18<-rep(NA,B)
  # t19<-rep(NA,B)
  t20<-rep(NA,B)
  t21<-rep(NA,B)
  t22<-rep(NA,B)
# initialize lower and upper interval for summary estimate
  l1<-rep(NA,B)
  r1<-rep(NA,B)
  l2<-rep(NA,B)
  r2<-rep(NA,B)
  l3<-rep(NA,B)
  r3<-rep(NA,B)
  l4<-rep(NA,B)
  r4<-rep(NA,B)
  l5<-rep(NA,B)
  r5<-rep(NA,B)
  l6<-rep(NA,B)
  r6<-rep(NA,B)
  l7<-rep(NA,B)
  r7<-rep(NA,B)
  l8<-rep(NA,B)
  r8<-rep(NA,B)
  l9<-rep(NA,B)
  r9<-rep(NA,B)
```

```
l10<-rep(NA,B)
r10<-rep(NA,B)
l11<-rep(NA,B)
r11<-rep(NA,B)
l12<-rep(NA,B)
r12<-rep(NA,B)
l13<-rep(NA,B)
r13<-rep(NA,B)
l14<-rep(NA,B)
r14<-rep(NA,B)
l15<-rep(NA,B)
r15<-rep(NA,B)
l16<-rep(NA,B)
r16<-rep(NA,B)
l17<-rep(NA,B)
r17<-rep(NA,B)
l18<-rep(NA,B)
r18<-rep(NA,B)
# l19<-rep(NA,B)
# r19<-rep(NA,B)
l20<-rep(NA,B)
r20<-rep(NA,B)
l21<-rep(NA,B)
r21<-rep(NA,B)
l22<-rep(NA,B)
r22<-rep(NA,B)
# initialize counter to compute absolute empirical bias
coo1=coo2=coo3=coo4=coo5=coo6=coo7=coo8=coo9=coo10=coo11=coo12=coo13=
coo14=coo15=coo16=coo17=coo18=coo20=coo21=coo22=0
#coo1=coo2=coo3=coo4=coo5=coo6=coo7=coo8=coo9=coo10=coo11=coo12=coo13
#=coo14=coo15=coo16=coo17=coo18=coo19=coo20=coo21=coo22=0
```

```
#initialize counter to compute how many times the confidence interval doesn't contain
#the value theta, this is useful for computing type error I

count1=count2=count3=count4=count5=count6=count7=count8=count9=count10=count11=count12=count13=count14=count15=count16=count17=count18=count20=count21=count22=0

#count1=count2=count3=count4=count5=count6=count7=count8=count9=count10=

#count11=count12=count13=count14=count15=count16=count17=count18=count19=

#count20=count21=count22=0

# initialize counter to compute how many times the confidence interval of summary

# estimate from simulated data with theta doesn't contain the value theta0 in

# confidence interval, this is useful for computing power.

c1=c2=c3=c4=c5=c6=c7=c8=c9=c10=c11=c12=c13=c14=c15=c16=c17=c18=c20=c21=c22=0

#c1=c2=c3=c4=c5=c6=c7=c8=c9=c10=c11=c12=c13=c14=c15=c16=c17=c18=c19=c
# 20=c21=c22=0

#import tables with B simulated data sets

pinakasyi <- as.matrix(read.table(fileyi, header=T, row.names=1, sep = ",",as.is=F))

colnames(pinakasyi)=NULL

rownames(pinakasyi)=NULL

pinakasvi <- as.matrix(read.table(filevi, header=T, row.names=1, sep = ",",as.is=F))

colnames(pinakasvi)=NULL

rownames(pinakasvi)=NULL

#pinakasni <- as.matrix(read.table(fileni, header=T, row.names=1, sep = ",",as.is=F))

#colnames(pinakasni)=NULL

#rownames(pinakasni)=NULL

#pinakasJ <- as.matrix(read.table(fileJ, header=T, row.names=1, sep = ",",as.is=F))

#colnames(pinakasJ)=NULL

#rownames(pinakasJ)=NULL

# end of import tables

# generate sample sizes from Uniform distribution for each study

# it needs to compute only the RBo estimator

ni<-runif(k,50,500)

# create function metanalysis for computing estimators HM, RBo, RBp, (MBH)

metanalysis<-function(y=yi,v=vi)
```

```
#metanalysis<-function(y=yi,v=vi,n=ni,s=J)
{
# inverse variance weighting for fixed effect model
wi=1/v
# summary estimate for fixed effect model
mestimator=sum(wi*y)/sum(wi)
# Q-statistic
Q=sum(wi*((y-mestimator)^2))
# HM estimator
thm=(Q^2)/((2*(k-1)+Q)*(sum(wi)-(sum(wi^2)/sum(wi))))
ymeso=sum(y)/k
# RBo estimator
trb0=sum((y-ymeso)^2)/(k+1) - ((sum(ni)-k)*(k+1)*sum(v))/(sum(ni-k+2)*k*(k+1))
est.trb0=max(0,trb0)
# RBp estimator
est.trbp=sum((yi-ymeso)^2)/(k+1)
# MBH estimator
#fi=1-(2*n-4)/((s^2)*(2*n-2))
#tmbh=sum((1-fi)*((y-mestimator)^2))/(k-1)-(2/k)*sum(1/n)-(1/k)*sum(fi*(y^2))
#est.mbh=max(0,tmbh)
result2<<-list(thm=thm, est.trb0=est.trb0,est.trbp=est.trbp)
#result2<<-list(thm=thm, est.trb0=est.trb0,est.trbp=est.trbp,est.mbh=est.mbh)
}
# end of function metanalysis
# function randomeffect
# computes summary estimate and its (1 − α)% confidence interval
# for random effects model
randomeffect<-function(timi,y=yi,v=vi)
{
# inverse variance weighting for random effects model
wir=1/(v+timi)
# summary estimate for random effects model
```

```
mre=sum(wir*y)/sum(wir)
# (1 − a)% confidence interval for summary estimate
sem2=1/sqrt(sum(wir))
cl2=mre+qnorm(c(a/2,1-a/2))*sem2
result3<<-list(cl2=cl2)
}
# end of function randomeffect
# function to obtain estimator from data in non-parametric bootstrap
func1 <- function(dat, indices) {
res1 <- rma(yi, vi,data=dat,method="DL",subset=indices)
c(res1$tau2, res1$se.tau2^2)
}
for (i in 1:B)
{   #open big loop
# import data
yi<-pinakasyi[,i]
vi<-pinakasvi[,i]
# ni<-pinakasni[,i]
# J<-pinakasJ[,i])
dat<-data.frame(yi,vi)
# dat<-data.frame(yi,vi,ni,J)
# end of import data
# computation of heterogeneity estimators
# compute heterogeneity variance estimators (DL, HE, HS, SJ, ML, REML, PM,
# DL2, GHO2, SJgho,) and Knapp and Hartung estimators with 'metafor' package.
res1 <- rma(yi=yi, vi=vi,method="DL")
res2 <- rma(yi=yi, vi=vi,method="HE")
res3 <- rma(yi=yi, vi=vi,method="HS")
res4 <- rma(yi=yi, vi=vi,method="SJ")
res5<- rma(yi=yi, vi=vi,method="ML")
res6<- rma(yi=yi, vi=vi,method="REML")
res7<- rma(yi=yi, vi=vi,method="PM")
```

```
res.DL2 <- rma(yi=yi, vi=vi,method="GENQ", weights=1/(vi + res1$tau2))
res8 <- rma(yi=yi, vi=vi,tau2=res.DL2$tau2)
res.HE2 <- rma(yi=yi, vi=vi,method="GENQ", weights=1/(vi + res2$tau2))
res9 <- rma(yi=yi, vi=vi,tau2=res.HE2$tau2)
res10 <- rma(yi=yi, vi=vi, method="SJ", control=list(tau2.init=res2$tau2))
res11 <- rma(yi=yi,vi=vi,method="DL",knha=TRUE)
res12 <- rma(yi=yi,vi=vi,method="HE",knha=TRUE)
res13 <- rma(yi=yi,vi=vi,method="ML",knha=TRUE)
res14 <- rma(yi=yi,vi=vi,method="REML",knha=TRUE)
metanalysis(yi,vi)
# metanalysis(yi,vi,ni,J)
# keep estimators out of function metanalysis
# HM estimator
hm=result2$thm
# RBp estimator
rbp=result2$est.trbp
# RBo estimator
rbo=result2$est.trb0
# MBH estimator
# mbh=result2$est.mbh
# compute DLp estimator
if(res1$tau2<=0)
{
tdlp=0.01
}else{
tdlp=res1$tau2
}
# compute BM estimator
ab=2
resm <- rma(yi=yi, vi=vi,method="ML",tau2=sqrt(res5$tau2))
seml=resm$se.tau2
tml=sqrt(res5$tau2)
```

```
if(tml==0)
{
tbm=(ab-1)*(seml^2)
}else{
tbm=(tml/2+(tml/2)*sqrt(1+(4*(ab-1)*(seml^2))/res5$tau2))^2
}
tbm
# compute FB estimator
# keep data in a list
data<-list(k=k, y=dat$yi,v=dat$vi)
# Use bugs function from package ‘R2WinBUGS’. This function takes data and
# starting values as input. It use a WinBUGS script (model.file) automatically and
# gives the results back to R.
FullyBayesian.sim <- bugs(data,inits=NULL,c("mean","tau2"),
model.file="c:/FullyBayesian1.odc", debug=F, bugs.directory=winbugsdir,
program="WinBUGS")
# keep FB estimator
fb<-FullyBayesian.sim$mean[2]
fullb=fb$tau2
fbb<-FullyBayesian.sim$mean[1]
fulb=fbb$mean      # summary estimate with FB estimator
x<-FullyBayesian.sim$sd[1]
clfb=x$mean       # confidence interval of summary estimate with FB estimator
# non-parametric bootstrap for DL estimator
# bootstrapping with 500 replications with calling the function func1
apot1<- boot(dat, func1, R=500)
# non-parametric bootstrap version of the DL (DLb) estimator
dlb=mean(apot1$t[1:500,1])
# end of non-parametric bootstrap for DL estimator
# end of computation heterogeneity estimators
# computation of (1 − a)% confidence interval for summary estimate
# with ‘metafor’ package
d1=res1$b+qnorm(c(a/2,1-a/2))*res1$se
```

```
d2=res2$b+qnorm(c(a/2,1-a/2))*res2$se
d3=res3$b+qnorm(c(a/2,1-a/2))*res3$se
d4=res4$b+qnorm(c(a/2,1-a/2))*res4$se
d5=res5$b+qnorm(c(a/2,1-a/2))*res5$se
d6=res6$b+qnorm(c(a/2,1-a/2))*res6$se
d7=res7$b+qnorm(c(a/2,1-a/2))*res7$se
d8=res8$b+qnorm(c(a/2,1-a/2))*res8$se
d9=res9$b+qnorm(c(a/2,1-a/2))*res9$se
d10=res10$b+qnorm(c(a/2,1-a/2))*res10$se
d11=res11$b+c(-1,1)*qt(c(1-a/2,1-a/2),df=k-1)*res11$se
d12=res12$b+c(-1,1)*qt(c(1-a/2,1-a/2),df=k-1)*res12$se
d13=res13$b+c(-1,1)*qt(c(1-a/2,1-a/2),df=k-1)*res13$se
d14=res14$b+c(-1,1)*qt(c(1-a/2,1-a/2),df=k-1)*res14$se
#Computation of (1 − a)% confidence interval for summary estimate
# with our custom function
randomeffect(hm)
d15=result3$cl2
randomeffect(rbp)
d16=result3$cl2
randomeffect(rbo)
d17=result3$cl2
randomeffect(tbm)
d18=result3$cl2
# randomeffect(mbh)
# d19=result3$cl2
randomeffect(tdlp)
d22=result3$cl2
randomeffect(dlb)
d20=result3$cl2
d21= fulb+qnorm(c(a/2,1-a/2))*clfb
# heterogeneity estimators
t1[i]<-res1$tau2
```

```
t2[i]<-res2$tau2
t3[i]<-res3$tau2
t4[i]<-res4$tau2
t5[i]<-res5$tau2
t6[i]<-res6$tau2
t7[i]<-res7$tau2
t8[i]<-res8$tau2
t9[i]<-res9$tau2
t10[i]<-res10$tau2
t11[i]<-res11$tau2
t12[i]<-res12$tau2
t13[i]<-res13$tau2
t14[i]<-res14$tau2
t15[i]<-hm
t16[i]<-rbp
t17[i]<-rbo
t18[i]<-tbm
# t19[i]<-mbh
t20[i]<-dlb
t21[i]<-fullb
t22[i]<-tdlp
# calculations for the absolute empirical bias
coo1=coo1+abs(t1[i]-tau)
coo2=coo2+abs(t2[i]-tau)
coo3=coo3+abs(t3[i]-tau)
coo4=coo4+abs(t4[i]-tau)
coo5=coo5+abs(t5[i]-tau)
coo6=coo6+abs(t6[i]-tau)
coo7=coo7+abs(t7[i]-tau)
coo8=coo8+abs(t8[i]-tau)
coo9=coo9+abs(t9[i]-tau)
coo10=coo10+abs(t10[i]-tau)
```

```
coo11=coo11+abs(t11[i]-tau)
coo12=coo12+abs(t12[i]-tau)
coo13=coo13+abs(t13[i]-tau)
coo14=coo14+abs(t14[i]-tau)
coo15=coo15+abs(t15[i]-tau)
coo16=coo16+abs(t16[i]-tau)
coo17=coo17+abs(t17[i]-tau)
coo18=coo18+abs(t18[i]-tau)
# coo19=coo19+abs(t19[i]-tau)
coo20=coo20+abs(t20[i]-tau)
coo21=coo21+abs(t21[i]-tau)
coo22=coo22+abs(t22[i]-tau)
# lower and upper interval for summary estimate
l1[i]<-d1[1]
r1[i]<-d1[2]
l2[i]<-d2[1]
r2[i]<-d2[2]
l3[i]<-d3[1]
r3[i]<-d3[2]
l4[i]<-d4[1]
r4[i]<-d4[2]
l5[i]<-d5[1]
r5[i]<-d5[2]
l6[i]<-d6[1]
r6[i]<-d6[2]
l7[i]<-d7[1]
r7[i]<-d7[2]
l8[i]<-d8[1]
r8[i]<-d8[2]
l9[i]<-d9[1]
r9[i]<-d9[2]
l10[i]<-d10[1]
```

```
r10[i]<-d10[2]
l11[i]<-d11[1]
r11[i]<-d11[2]
l12[i]<-d12[1]
r12[i]<-d12[2]
l13[i]<-d13[1]
r13[i]<-d13[2]
l14[i]<-d14[1]
r14[i]<-d14[2]
l15[i]<-d15[1]
r15[i]<-d15[2]
l16[i]<-d16[1]
r16[i]<-d16[2]
l17[i]<-d17[1]
r17[i]<-d17[2]
l18[i]<-d18[1]
r18[i]<-d18[2]
# l19[i]<-d19[1]
# r19[i]<-d19[2]
l20[i]<-d20[1]
r20[i]<-d20[2]
l21[i]<-d21[1]
r21[i]<-d21[2]
l22[i]<-d22[1]
r22[i]<-d22[2]
# end of lower and upper interval for summary estimate
# calculations for type error I
if(l1[i]>theta || r1[i]<theta){
count1=count1+1
}
if(l2[i]>theta || r2[i]<theta){
count2=count2+1
```

```
}
if(l3[i]>theta || r3[i]<theta){
count3=count3+1
}
if(l4[i]>theta || r4[i]<theta){
count4=count4+1
}
if(l5[i]>theta || r5[i]<theta){
count5=count5+1
}
if(l6[i]>theta || r6[i]<theta){
count6=count6+1
}
if(l7[i]>theta || r7[i]<theta){
count7=count7+1
}
if(l8[i]>theta || r8[i]<theta){
count8=count8+1
}
if(l9[i]>theta || r9[i]<theta){
count9=count9+1
}
if(l10[i]>theta || r10[i]<theta){
count10=count10+1
}
if(l11[i]>theta || r11[i]<theta){
count11=count11+1
}
if(l12[i]>theta || r12[i]<theta){
count12=count12+1
}
if(l13[i]>theta || r13[i]<theta){
```

```
count13=count13+1
}
if(l14[i]>theta || r14[i]<theta){
count14=count14+1
}
if(l15[i]>theta || r15[i]<theta){
count15=count15+1
}
if(l16[i]>theta || r16[i]<theta){
count16=count16+1
}
if(l17[i]>theta || r17[i]<theta){
count17=count17+1
}
if(l18[i]>theta || r18[i]<theta){
count18=count18+1
}
# if(l19[i]>theta || r19[i]<theta){
# count19=count19+1
# }
if(l20[i]>theta || r20[i]<theta){
count20=count20+1
}
if(l21[i]>theta || r21[i]<theta){
count21=count21+1
}
if(l22[i]>theta || r22[i]<theta){
count22=count22+1
}
# end calculations for type error I
# calculations for power
if(theta0<l1[i] || r1[i]<theta0){
```

```
c1=c1+1
}
if(theta0<l2[i] || r2[i]<theta0){
c2=c2+1
}
if(theta0<l3[i] || r3[i]<theta0){
c3=c3+1
}
if(theta0<l4[i] || r4[i]<theta0){
c4=c4+1
}
if(theta0<l5[i] || r5[i]<theta0){
c5=c5+1
}
if(theta0<l6[i] || r6[i]<theta0){
c6=c6+1
}
if(theta0<l7[i] || r7[i]<theta0){
c7=c7+1
}
if(theta0<l8[i] || r8[i]<theta0){
c8=c8+1
}
if(theta0<l9[i] || r9[i]<theta0){
c9=c9+1
}
if(theta0<l10[i] || r10[i]<theta0){
c10=c10+1
}
if(theta0<l11[i] || r11[i]<theta0){
c11=c11+1
}
```

```
if(theta0<l12[i] || r12[i]<theta0){
c12=c12+1
}
if(theta0<l13[i] || r13[i]<theta0){
c13=c13+1
}
if(theta0<l14[i] || r14[i]<theta0){
c14=c14+1
}
if(theta0<l15[i] || r15[i]<theta0){
c15=c15+1
}
if(theta0<l16[i] || r16[i]<theta0){
c16=c16+1
}
if(theta0<l17[i] || r17[i]<theta0){
c17=c17+1
}
if(theta0<l18[i] || r18[i]<theta0){
c18=c18+1
}
# if(theta0<l19[i] || r19[i]<theta0){
# c19=c19+1
# }
if(theta0<l20[i] || r20[i]<theta0){
c20=c20+1
}
if(theta0<l21[i] || r21[i]<theta0){
c21=c21+1
}
if(theta0<l22[i] || r22[i]<theta0){
c22=c22+1
```

```
}
# end of calculations for power
print(i)
} # close big loop
# absolute empirical bias
bias1=coo1/B
bias2=coo2/B
bias3=coo3/B
bias4=coo4/B
bias5=coo5/B
bias6=coo6/B
bias7=coo7/B
bias8=coo8/B
bias9=coo9/B
bias10=coo10/B
bias11=coo11/B
bias12=coo12/B
bias13=coo13/B
bias14=coo14/B
bias15=coo15/B
bias16=coo16/B
bias17=coo17/B
bias18=coo18/B
# bias19=coo19/B
bias20=coo20/B
bias21=coo21/B
bias22=coo22/B

# empirical type error I
typerror1=count1/B
typerror2=count2/B
typerror3=count3/B
```

```
typerror4=count4/B
typerror5=count5/B
typerror6=count6/B
typerror7=count7/B
typerror8=count8/B
typerror9=count9/B
typerror10=count10/B
typerror11=count11/B
typerror12=count12/B
typerror13=count13/B
typerror14=count14/B
typerror15=count15/B
typerror16=count16/B
typerror17=count17/B
typerror18=count18/B
# typerror19=count19/B
typerror20=count20/B
typerror21=count21/B
typerror22=count22/B
# power
pow1=c1/B
pow2=c2/B
pow3=c3/B
pow4=c4/B
pow5=c5/B
pow6=c6/B
pow7=c7/B
pow8=c8/B
pow9=c9/B
pow10=c10/B
pow11=c11/B
pow12=c12/B
```

```
pow13=c13/B

pow14=c14/B

pow15=c15/B

pow16=c16/B

pow17=c17/B

pow18=c18/B

# pow19=c19/B

pow20=c20/B

pow21=c21/B

pow22=c22/B

# keep assessment criteria in vectors

w1=c(bias1,bias2,bias3,bias4,bias5,bias6,bias7,bias8,bias9,bias10,bias11,bias12,bias1
3,bias14,bias15,bias16,bias17,bias18,bias20,bias21,bias22)

#w1=c(bias1,bias2,bias3,bias4,bias5,bias6,bias7,bias8,bias9,bias10,bias11,bias12,bias
# 13,bias14,bias15,bias16,bias17,bias18,bias19,bias20,bias21,bias22)

bia=round(w1, digits=5)

w5
=c(typerror1,typerror2,typerror3,typerror4,typerror5,typerror6,typerror7,typerror8,typ
error9,typerror10,typerror11,typerror12,typerror13,typerror14,typerror15,typerror16,t
yperror17, typerror18,typerror20,typerror21,typerror22)

#w5
#=c(typerror1,typerror2,typerror3,typerror4,typerror5,typerror6,typerror7,typerror8,

# typerror9,typerror10,typerror11,typerror12,typerror13,typerror14,typerror15,

# typerror16,typerror17,typerror18, typerror19,typerror20,typerror21,typerror22)

type_error_I=round(w5, digits=5)

w6=c(pow1,pow2,pow3,pow4,pow5,
pow6,pow7,pow8,pow9,pow10,pow11,pow12,pow13,pow14,pow15,pow16,pow17,p
ow18, pow20,pow21,pow22)

# w6=c(pow1,pow2,pow3,pow4,pow5,pow6,pow7,pow8,pow9,pow10,pow11,

# pow12,pow13,pow14,pow15,pow16,pow17,pow18, pow19, pow20,pow21,pow22)

power=round(w6, digits=5)

result<-bia

result<-cbind(result,type_error_I)

result<-cbind(result,power)

# write csv file
```

```
write.csv2(result, fileResult,
row.names=c("DL","HE","HS","SJ","ML","REML","EB","DL2","GHO2","SJgho","DLknha","GHOknha","MLknha","REMLknha","HM","RBp","RBo","BM","DLb","FB","DLp"))
# write.csv2(result, fileResult,
# row.names=c("DL","HE","HS","SJ","ML","REML","EB","DL2","GHO2","SJgho",
# "DLknha","GHOknha","MLknha","REMLknha","HM","RBp","RBo","BM",
# "MBH","DLb","FB","DLp")
}
winbugsdir="C:/Program Files/WinBUGS"
# call function simulation for scenario tau=0, theta=0, theta0=0,
# k=10, B=1000, a=0.05
simulation(tau=0,theta=0,theta0=0,k=10,B=1000,a=0.05,fileResult="C:/senario tau=0 theta=0 k=10
a=0,05.csv",fileyi="C:/pinakas_yi.csv",filevi="C:/pinakas_vi.csv",winbugsdir)
# simulation(tau=0,theta=0,theta0=0,k=10,B=1000,a=0.05,fileResult="C:/senario
# tau=0 theta=0 k=10,
# a=0,05.csv",fileyi="C:/pinakas_yi.csv",filevi="C:/pinakas_vi.csv",
# fileni="C:/pinakas_ni.csv",fileJ="C:/pinakas_J.csv", winbugsdir)
# for another scenario with dichotomous type of outcome, e.g. tau=0, theta=0, k=20
# just change
simulation(tau=0,theta=0,theta0=0,k=20,B=1000,a=0.05,fileResult="C:/senario tau=0 theta=0 k=20,
a=0,05.csv",fileyi="C:/pinakas_yi.csv",filevi="C:/pinakas_vi.csv",winbugsdir)
# where fileyi and filevi must have the appropriate tables with generated data for the
# scenario tau=0, theta=0 and k=20
```

4. WinBUGS code for MCMC

```
model{
# Run MCMC for FB estimator

for(i in 1:k){
w[i]<-1/v[i]
y[i]~dnorm(theta[i],w[i])
theta[i]~dnorm(mean,prec)
}

# prior for summary estimate from Uniform(0, 10^6)
mean~dnorm(0,0.000001)
```

informative prior $logN(-2.56, 1.74^2)$ on the untransformed τ^2 scale for
dichotomous outcome
u~dnorm(-2.56,0.3302946228)
tau2<-exp(u)
prec<-1/tau2
}

informative prior $log(\tau^2) \sim t(-3.44, 2.59^2, 5)$ for continuous outcome just change

u~dt(-3.44,0.1490735081, 5)

tau2<-exp(u)
prec<-1/tau2

vague prior $Uniform(0,100)$ for continuous and dichotomous outcome just change

tau2<-pow(tau,2)
prec<-1/tau2
tau~dunif(0,100)