Integration of GWAS Data for Related Diseases

Wang Yingxuan; Qin Jiayue; Wuxin

2024-10-18

```r
## Define the data generation function
# input: M, pi00,pi01,pi10,pi11, alpha1,alpha2
# output: P1,P2, Z00,Z01,Z10,Z11
data_generate1 <- function(M,pi00,pi01,pi10,pi11,alpha1,alpha2){
  P1 <- numeric(M)
  P2 <- numeric(M)
  Z00 <- numeric(M)
  Z01 <- numeric(M)
  Z10 <- numeric(M)
  Z11 <- numeric(M)
  C <- c(1:M)
  index00 <- sample(C,M*pi00)
  C1 <- C[-index00]
  index01 <- sample(C1,M*pi01)
  real_index01 <-c()
  k=1
  for (i in index01){
    real_index01[k]=which(C1==i)
    k=k+1
  }
  C2 <- C1[-real_index01]
  index10 <- sample(C2,M*pi10)
  real_index10 <-c()
  k=1
  for (i in index10){
    real_index10[k]=which(C2==i)
    k=k+1
  }
  C3 <- C2[-real_index10]
  index11 <- sample(C3,M*pi11)
  Z00[index00] <- 1
  Z00[-index00] <- 0
  Z01[index01] <- 1
  Z01[-index01] <- 0
  Z10[index10] <- 1
  Z10[-index10] <- 0
  Z11[index11] <- 1
  Z11[-index11] <- 0
  P1[index00] <- runif(M*pi00)
  P1[index01] <- runif(M*pi01)
  P1[index10] <- rbeta(M*pi10,alpha1,1)
  P1[index11] <- rbeta(M*pi11,alpha1,1)
```

```
  P2[index00] <- runif(M*pi00)
  P2[index01] <- rbeta(M*pi01,alpha2,1)
  P2[index10] <- runif(M*pi10)
  P2[index11] <- rbeta(M*pi11,alpha2,1)
  return(list(P1=P1,P2=P2,Z00=Z00,Z01=Z01,Z10=Z10,Z11=Z11))
}
```

```
## Define the function that computes the log-likelihood
# input: P1,P2,pi00,pi01,pi10,pi11,alpha1,alpha2
log_p1 <- function(P1,P2,pi00,pi01,pi10,pi11,alpha1,alpha2){
  return(sum(log(pi00+pi01*alpha2*P2^(alpha2-1)+pi10*alpha1*P1^(alpha1-1)+pi11*alpha1*alpha2*P2^(alpha2
}
```

```
## iteration
# max_iter=10000
# Set the maximum number of iterations to max_iter=10000
## In each iteration, E steps and M steps are carried out to calculate the log-likelihood and judge whe
EM1 <- function(P1,P2,pi00_ini, pi01_ini,pi10_ini,pi11_ini,alpha1_ini,alpha2_ini, max_iter=1000){
  L_ini <- log_p1(P1,P2,pi00_ini, pi01_ini,pi10_ini,pi11_ini,alpha1_ini,alpha2_ini)
  for (iter in 1:max_iter){
  if(iter==1){
    pi00_old <- pi00_ini
    pi01_old <- pi01_ini
    pi10_old <- pi10_ini
    pi11_old <- pi11_ini
    alpha1_old <- alpha1_ini
    alpha2_old <- alpha2_ini
    L_old <- L_ini
  }
  ## E step
  gamma <- pi00_old+pi01_old*alpha2_old*P2^(alpha2_old-1)+alpha1_old*P1^(alpha1_old-1)*pi10_old+pi11_old
  gamma_z00 <- pi00_old/gamma
  gamma_z01 <- pi01_old*alpha2_old*P2^(alpha2_old-1)/gamma
  gamma_z10 <- pi10_old*alpha1_old*P1^(alpha1_old-1)/gamma
  gamma_z11 <- pi11_old*alpha1_old*alpha2_old*P1^(alpha1_old-1)*P2^(alpha2_old-1)/gamma
  ## M step
  pi00_new <- mean(gamma_z00)
  pi01_new <- mean(gamma_z01)
  pi10_new <- mean(gamma_z10)
  pi11_new <- mean(gamma_z11)
  alpha1_new <- -sum(gamma_z11+gamma_z10)/sum(gamma_z11*log(P1)+gamma_z10*log(P1))
  alpha2_new <- -sum(gamma_z11+gamma_z01)/sum(gamma_z11*log(P2)+gamma_z01*log(P2))
  ## compute the log-likelihood
  L_new <- log_p1(P1,P2,pi00_new,pi01_new,pi10_new,pi11_new,alpha1_new,alpha2_new)
  ## whether the algorithm accepts or not
  if(L_new<L_old){
    print("Error: log likelihood is not increasing!")
    break
  }
  if((L_new-L_old)/abs(L_new)<1e-5){
    pi00_est <- pi00_new
    pi01_est <- pi01_new
    pi10_est <- pi10_new
```

```
    pi11_est <- pi11_new
    alpha1_est <- alpha1_new
    alpha2_est <- alpha2_new
    break
  }else{
    pi00_old <- pi00_new
    pi01_old <- pi01_new
    pi10_old <- pi10_new
    pi11_old <- pi11_new
    alpha1_old <- alpha1_new
    alpha2_old <- alpha2_new
    L_old <- L_new
  }
  }
  return(list(pi00 = pi00_new, pi01 = pi01_new, pi10 = pi10_new, pi11 = pi11_new ,alpha1 = alpha1_new, a
}
```

```
# set random seed
set.seed(1)
# generate simulation data
M <- 100000
data <- data_generate1(M,0.7,0.1,0.15,0.05,0.2,0.2)
# Parameter estimates were obtained using the EM algorithm to check for agreement with the previous res
theta_est <- EM1(data$P1,data$P2,pi00_ini=0.6,pi01_ini=0.05,pi10_ini=0.15,pi11_ini=0.01,alpha1_ini=0.2,a
print(theta_est)
```

```
## $pi00
## [1] 0.6986258
##
## $pi01
## [1] 0.0995448
##
## $pi10
## [1] 0.1532531
##
## $pi11
## [1] 0.0485763
##
## $alpha1
## [1] 0.2014528
##
## $alpha2
## [1] 0.1994058
```

```
# The estimated value is close to the true value.
```

```
# Calculate the posterior probability
gamma_post <- theta_est$pi00+theta_est$pi01*theta_est$alpha2*data$P2^(theta_est$alpha2-1)+theta_est$alp
z11_post <- theta_est$pi11*theta_est$alpha1*theta_est$alpha2*data$P1^(theta_est$alpha1-1)*data$P2^(theta
z01_post <- theta_est$pi01*theta_est$alpha2*data$P2^(theta_est$alpha2-1)/gamma_post
z10_post <- theta_est$alpha1*data$P1^(theta_est$alpha1-1)*theta_est$pi10/gamma_post
z00_post <- theta_est$pi00/gamma_post
```

```r
z1_post <- z11_post+z10_post
z2_post <- z11_post+z01_post
```

```r
# Given a level alpha that controls the FDR, use the assoc function to identify SNPS associated with th
# The inputs are posterior and alpha.
# posterior is an m-dimensional vector, where each element represents the posterior probability of the
# alpha is the level that controls FDR
# The output of the  function is Z_est, an m-dimensional vector, where each element represents whether
assoc <- function(posterior, alpha){
  M          <- length(posterior)
  fdr        <- 1 - posterior
  rank.fdr   <- rank(fdr)
  sort.fdr   <- sort(fdr)
  cumsum.fdr <- cumsum(sort.fdr)
  sort.FDR   <- cumsum.fdr/seq(1, M, 1)
  FDR        <- sort.FDR[rank.fdr]

  Z_est <- rep(0, M)
  Z_est[which(FDR <= alpha)] <- 1

  return(Z_est)
}

# Use assoc function to identify SNPs associated with disease on simulated data (controlling FDR at lev

Z_est_11 <- assoc(z11_post, 0.1)
sum(Z_est_11)
```

```
## [1] 817
```

```r
## Calculate FDP and power
# Compare the real Z with the Z_est obtained in the previous step, by using table(Z_est, Z)
# V is the number of (Z_est=1, Z=0) in the table, S is the number of (Z_est=1, Z=1), R is the number of
# Calculate FDP using V/R and power using S/(M-M0)
# FDP should be around 0.1, otherwise the code is broken

t<-table(Z_est_11, data$Z11)
t
```

```
##
## Z_est_11     0     1
##        0 94932  4251
##        1    68   749
```

```r
FDP <- t[2, 1]/(t[2, 1] + t[2, 2])
FDP
```

```
## [1] 0.08323133
```

```r
power<-t[2,2]/(t[1,2]+t[2,2])
power
```

```
## [1] 0.1498
```

```
#Related to disease 1
Z_est_1 <- assoc(z1_post, 0.1)
sum(Z_est_1)
```

```
## [1] 9159
```

```
t<-table(Z_est_1, data$Z11+data$Z10)
t
```

```
##
## Z_est_1     0     1
##       0 79057 11784
##       1   943  8216
```

```
FDP <- t[2, 1]/(t[2, 1] + t[2, 2])
FDP
```

```
## [1] 0.1029588
```

```
power<-t[2,2]/(t[1,2]+t[2,2])
power
```

```
## [1] 0.4108
```

```
#Related to disease 2
Z_est_2 <- assoc(z2_post, 0.1)
sum(Z_est_2)
```

```
## [1] 6256
```

```
t<-table(Z_est_2, data$Z11+data$Z01)
t
```

```
##
## Z_est_2     0     1
##       0 84395  9349
##       1   605  5651
```

```
FDP <- t[2, 1]/(t[2, 1] + t[2, 2])
FDP
```

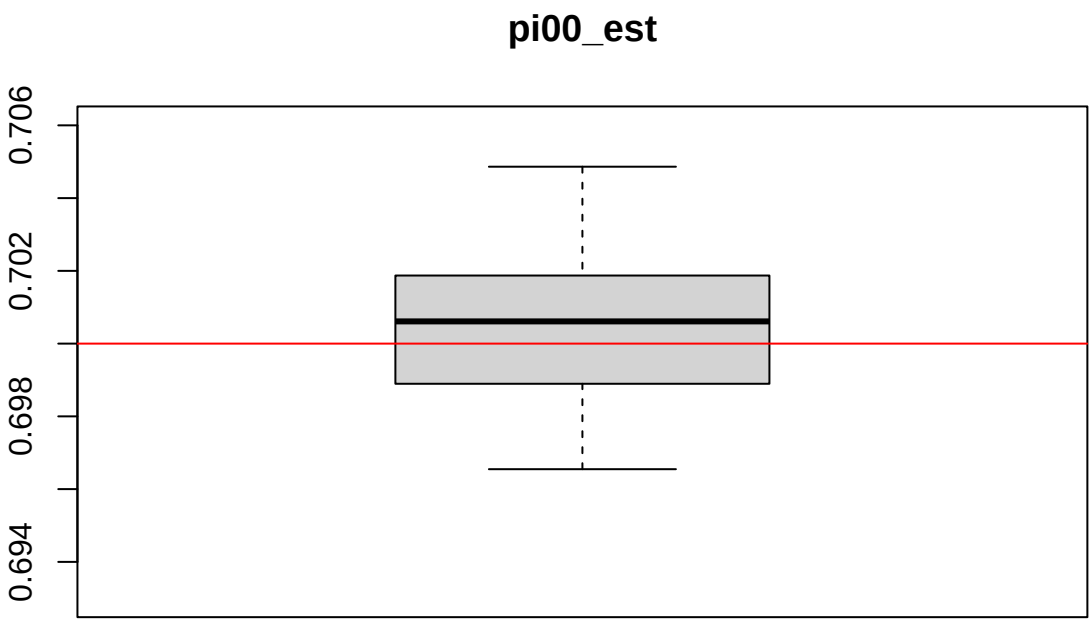
```
## [1] 0.09670716
```

```
power<-t[2,2]/(t[1,2]+t[2,2])
power
```

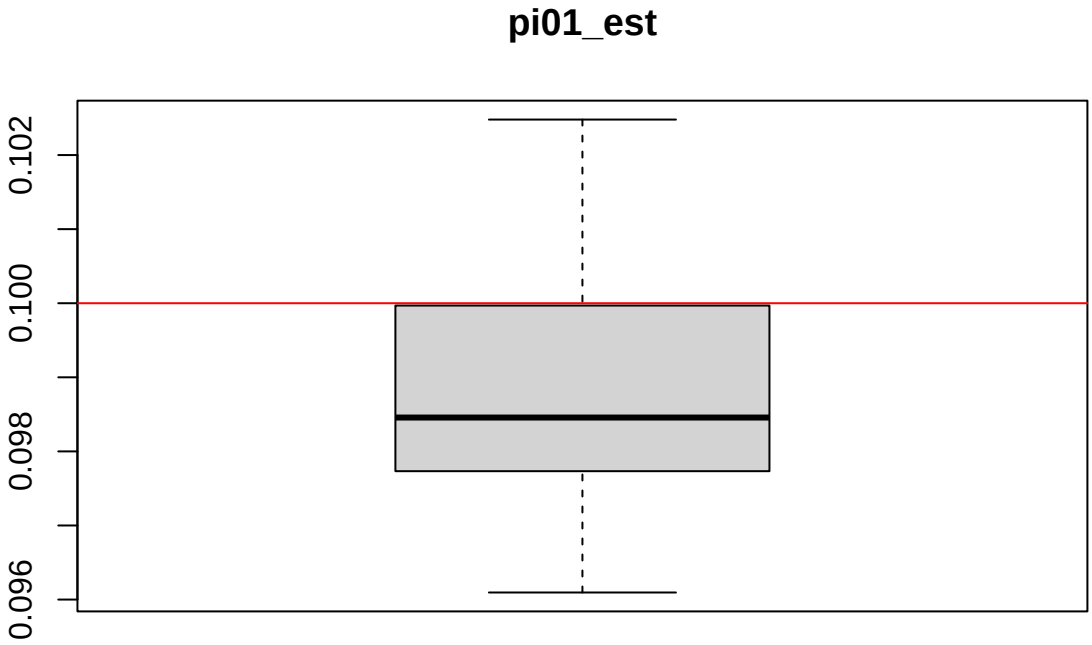
```
## [1] 0.3767333
```

```r
# Repeat 20 times and record pi00_est,pi01_est,pi10_est,pi11_est,alpha1_est, alpha2_est, FDP, power and
rep <- 20
pi00_est <- pi01_est <-pi10_est <- pi11_est <- alpha1_est <- alpha2_est <- Z11_FDP <- Z1_FDP <- Z2_FDP <
for (i in 1:rep) {
  set.seed(i)
  data <- data_generate1(M,0.7,0.1,0.15,0.05,0.2,0.2)
  theta_est <- EM1(data$P1,data$P2,pi00_ini=0.6,pi01_ini=0.05,pi10_ini=0.15,pi11_ini=0.01,alpha1_ini=0.1
  pi00_est[i] <- theta_est$pi00
  pi01_est[i] <- theta_est$pi01
  pi10_est[i] <- theta_est$pi10
  pi11_est[i] <- theta_est$pi11
  alpha1_est[i] <- theta_est$alpha1
  alpha2_est[i] <- theta_est$alpha2
  gamma_post <- theta_est$pi00+theta_est$pi01*theta_est$alpha2*data$P2^(theta_est$alpha2-1)+theta_est$al
  z11_post <- theta_est$pi11*theta_est$alpha1*theta_est$alpha2*data$P1^(theta_est$alpha1-1)*data$P2^(the
  z01_post <- theta_est$pi01*theta_est$alpha2*data$P2^(theta_est$alpha2-1)/gamma_post
  z10_post <- theta_est$alpha1*data$P1^(theta_est$alpha1-1)*theta_est$pi10/gamma_post
  z00_post <- theta_est$pi00/gamma_post
  z1_post <- z11_post+z10_post
  z2_post <- z11_post+z01_post
  Z_est_11 <- assoc(z11_post, 0.1)
  t11<-table(Z_est_11, data$Z11)
  Z11_FDP[i] <- t11[2, 1]/(t11[2, 1] + t11[2, 2])
  Z11_power[i] <-t11[2,2]/(t11[1,2]+t11[2,2])
  Z_est_1 <- assoc(z1_post, 0.1)
  t1<-table(Z_est_1, data$Z11+data$Z10)
  Z1_FDP[i] <- t1[2, 1]/(t1[2, 1] + t1[2, 2])
  Z1_power[i] <-t1[2,2]/(t1[1,2]+t1[2,2])
  Z_est_2 <- assoc(z2_post, 0.1)
  t2<-table(Z_est_2, data$Z11+data$Z01)
  Z2_FDP[i] <- t2[2, 1]/(t2[2, 1] + t2[2, 2])
  Z2_power[i] <-t2[2,2]/(t2[1,2]+t2[2,2])
}
```

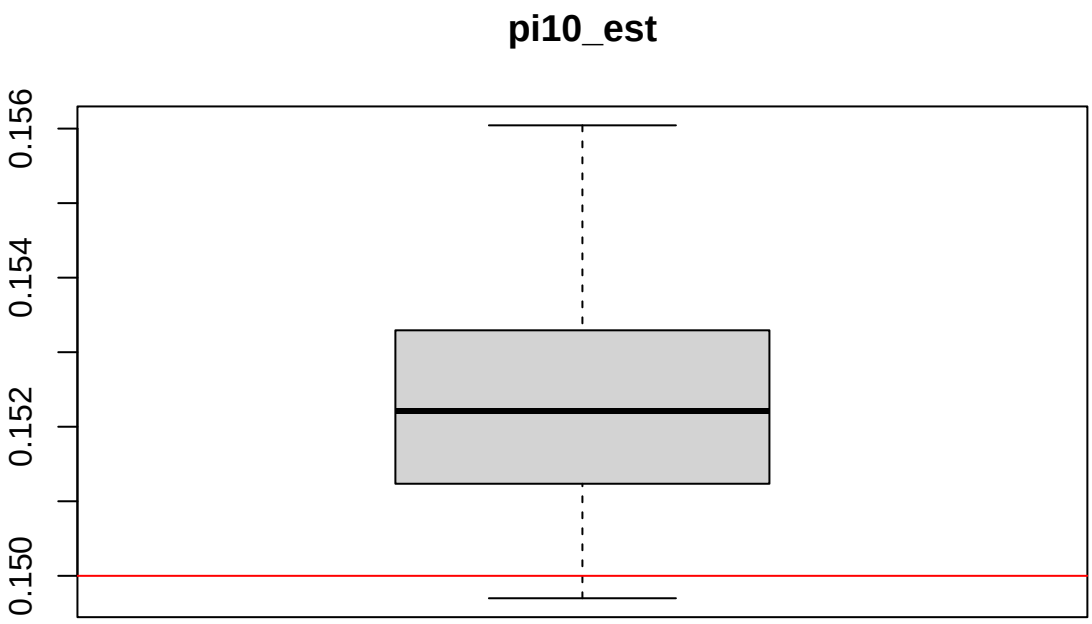
```r
boxplot(pi00_est,ylim=c(0.693,0.706))
title("pi00_est")
abline(h=0.7,col="red")
```



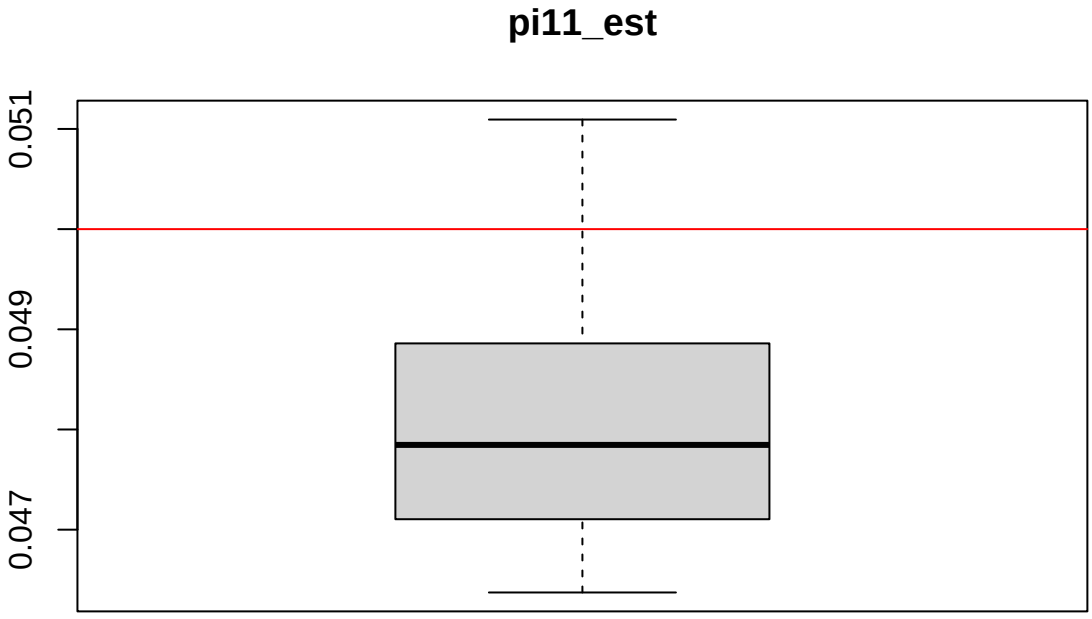

```
boxplot(pi01_est)
title("pi01_est")
abline(h=0.1,col="red")
```



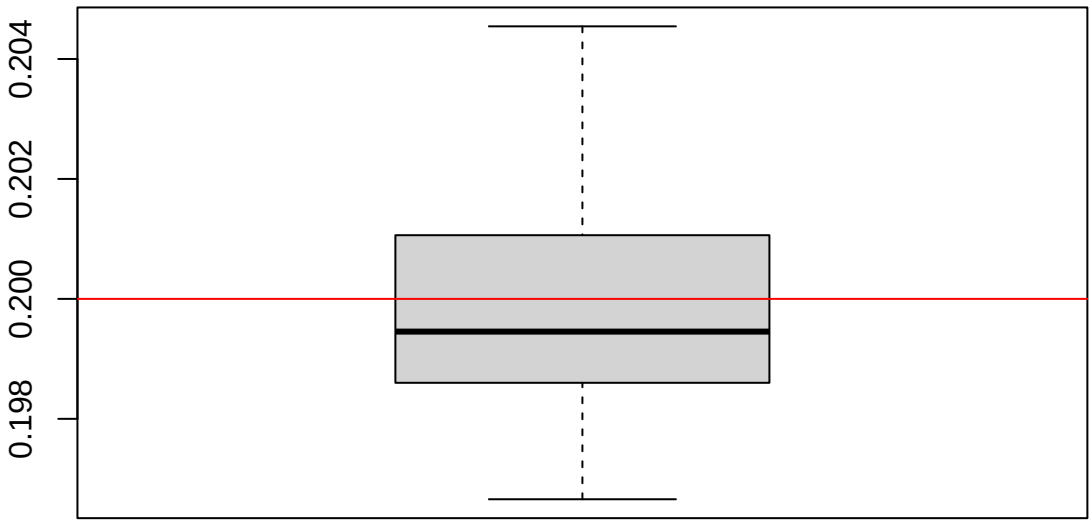

```
boxplot(pi10_est)
title("pi10_est")
abline(h=0.15,col="red")
```



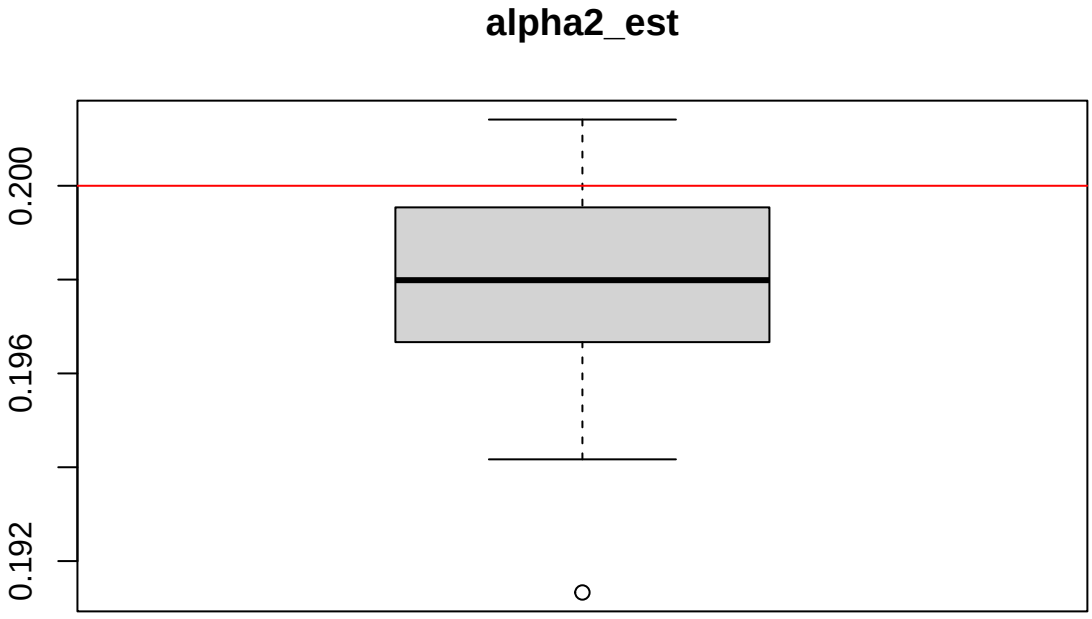

```
boxplot(pi11_est)
title("pi11_est")
abline(h=0.05,col="red")
```



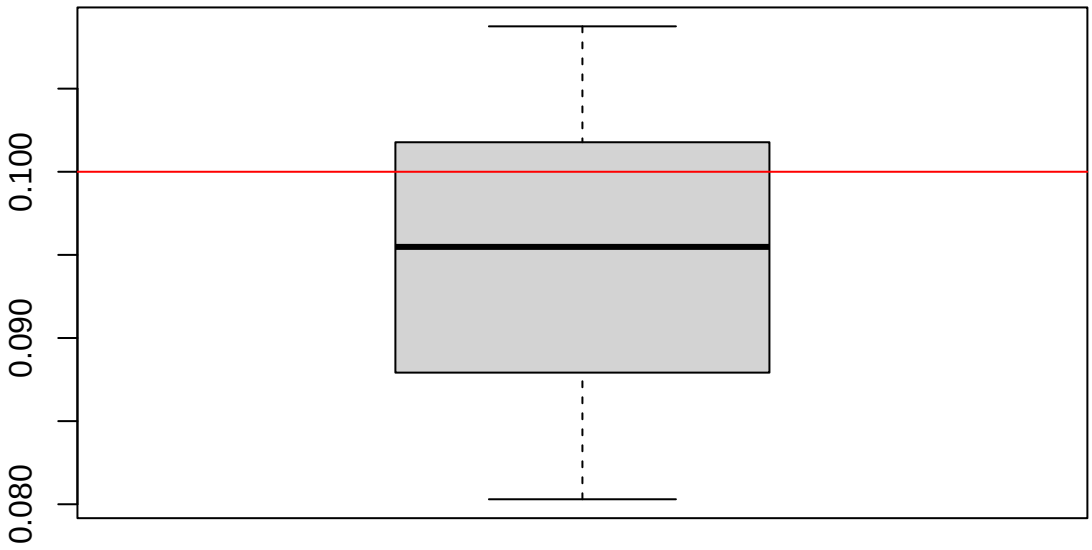

```
boxplot(alpha1_est)
title("alpha1_est")
abline(h=0.2,col="red")
```



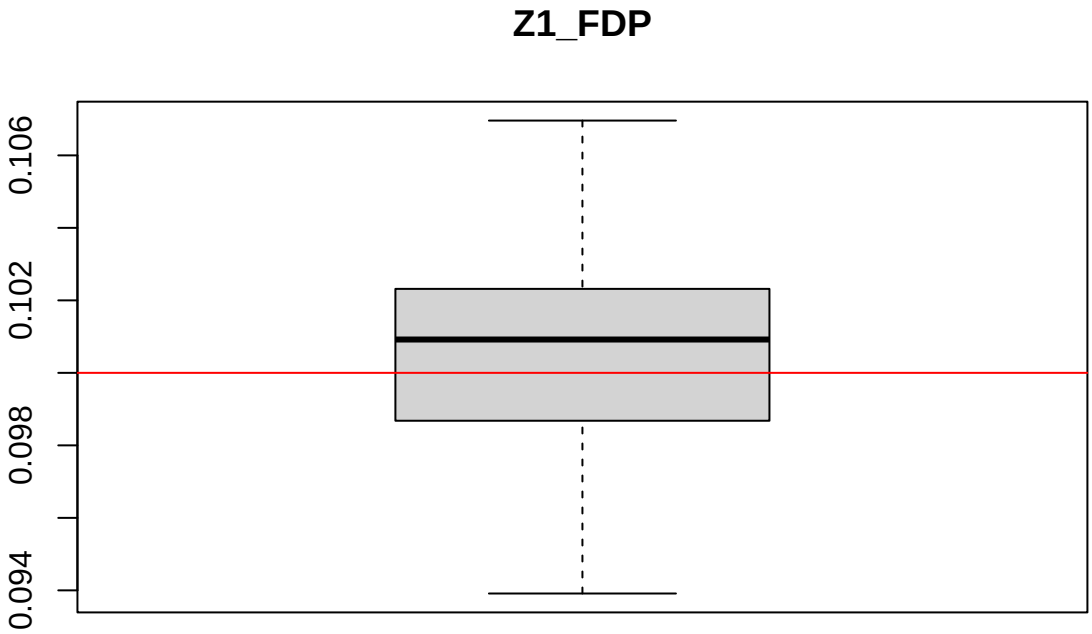

```
boxplot(alpha2_est)
title("alpha2_est")
abline(h=0.2,col="red")
```



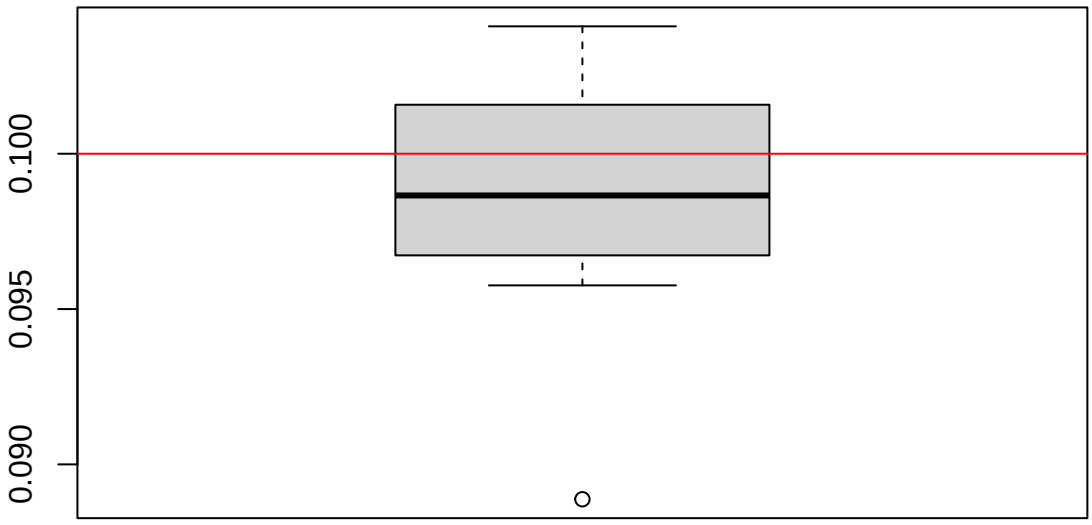

```
boxplot(Z11_FDP)
title("Z11_FDP")
abline(h=0.1,col="red")
```



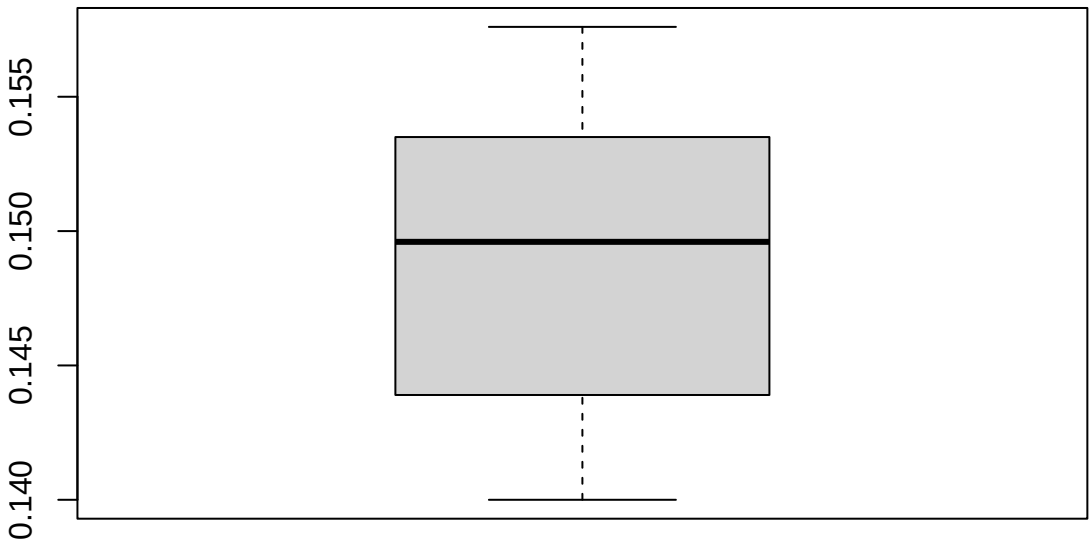

```
boxplot(Z1_FDP)
title("Z1_FDP")
abline(h=0.1,col="red")
```



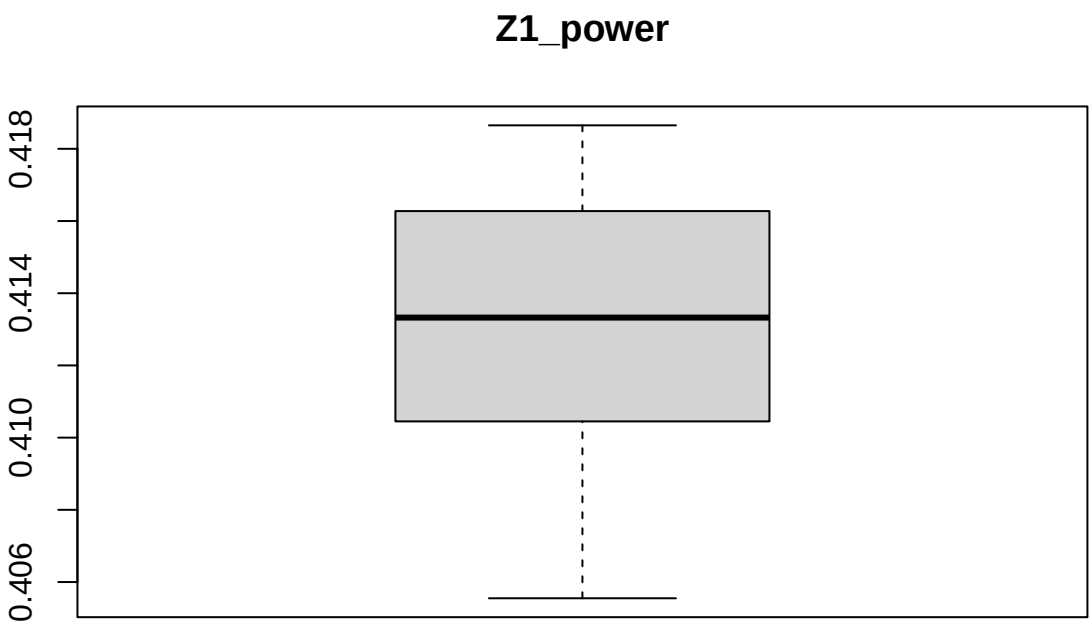

```
boxplot(Z2_FDP)
title("Z2_FDP")
abline(h=0.1,col="red")
```



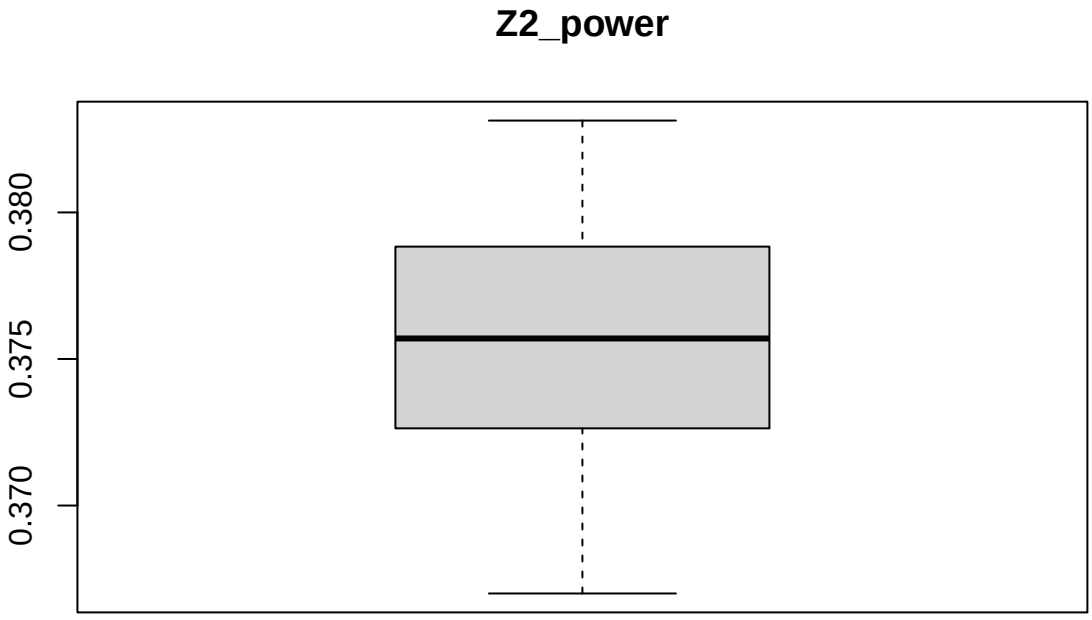

```
boxplot(Z11_power)
title("Z11_power")
```




```
boxplot(Z1_power)
title("Z1_power")
```




```
boxplot(Z2_power)
title("Z2_power")
```




```
## Define the function that calculates the log-likelihood
# input: p and pi1, alpha

log_X <- function(P, pi1, alpha){
  return(sum(log(pi1*alpha*(P^(alpha-1))+1-pi1)))
}

## Define the function that implements the EM algorithm
# The EM function takes P, pi1_ini, alpha_ini, max_iter, tol and outputs pi1_est, alpha_est
EM <- function(P, pi1_ini, alpha_ini, max_iter=10000, tol=1e-6){
```