HWK#11

library(binom)

newtonraphson <- function(ftn, x0, tol = 1e-9, max.iter = 100) {

x <- x0 # x0: the initial value

fx <- ftn(x)

iter <- 0

while ((max(abs(fx[[1]])) > tol) & (iter < max.iter)) {

x <- x - solve(fx[[2]]) %\*% fx[[1]]

fx <- ftn(x)

iter <- iter + 1

}

if (max(abs(fx[[1]])) > tol) {

cat('Algorithm failed to converge\n')

return(NULL)

} else { # max(abs(fx[[1]])) <= tol

cat("Algorithm converged\n")

return(x)

}

}

# (1) the sample size is 30

betaco <- c(-6,1,0.005)

n <- 30

no.rep <- 1000

pii<-c()

X <- c()

Y <- c()

for(i in 1:n){

set.seed(i)

gpa <- rnorm(n,3.1,0.3)

gre <- rnorm(n,580,80)

(pii[i] <- exp(betaco[1]+betaco[2]\*gpa+betaco[3]\*gre)/(1+exp(betaco[1]+betaco[2]\*gpa+betaco[3]\*gre)))

(Y[i]<- sample(c(0,1),1,c(1-pii[i],pii[i]),replace = F))

# above: data generation process

# below: data analysis process

(X <- cbind(rep(1,length(Y)),gpa,gre))

}

i <- 1

MLE <- matrix(NA,no.rep,3)

MLEglm <- matrix(NA,no.rep,3)

for(i in 1:no.rep){

set.seed(i)

X<- cbind(rep(1,length(Y)),gpa,gre)

ftn <- function(betaco) {

pii <- exp(X%\*%betaco)/ (1+exp(X%\*%betaco))

gradient <- t(X)%\*%(Y-pii)

hessian<- -t(X)%\*%diag(c(pii\*(1-pii)), length(Y))%\*%X

return(list(gradient, hessian))

}

MLE[i,] <- newtonraphson(ftn, c(0,0,0))

MLEglm [i,]<- glm(Y~gpa+gre,family = binomial)$coef

X[i] <- i+1

}

(meanMLE <- colSums(MLE)/no.rep)

(meanMLEglm <- colSums(MLEglm)/no.rep)

(meanMLE-betaco) # bias

(meanMLEglm-betaco) # bias

MLE30 <- MLE

R. output:

> (meanMLE <- colSums(MLE)/no.rep)

[1] -2.2130469319 0.5785163150 0.0005697193

> (meanMLEglm <- colSums(MLEglm)/no.rep)

[1] -2.2130469319 0.5785163150 0.0005697193

> (meanMLE-betaco) # bias

[1] 3.786953068 -0.421483685 -0.004430281

> (meanMLEglm-betaco) # bias

[1] 3.786953068 -0.421483685 -0.004430281

> MLE30 <- MLE

# (2) the sample size is 230

betaco <- c(-6,1,0.005)

n <- 230

no.rep <- 1000

pii<-c()

X <- c()

Y <- c()

for(i in 1:n){

set.seed(i)

gpa <- rnorm(n,3.1,0.3)

gre <- rnorm(n,580,80)

(pii[i] <- exp(betaco[1]+betaco[2]\*gpa+betaco[3]\*gre)/(1+exp(betaco[1]+betaco[2]\*gpa+betaco[3]\*gre)))

(Y[i]<- sample(c(0,1),1,c(1-pii[i],pii[i]),replace = F))

# above: data generation process

# below: data analysis process

(X <- cbind(rep(1,length(Y)),gpa,gre))

}

MLE <- matrix(NA,no.rep,3)

MLEglm <- matrix(NA,no.rep,3)

for(i in 1:no.rep){

set.seed(i)

X<- cbind(rep(1,length(Y)),gpa,gre)

ftn <- function(betaco) {

pii <- exp(X%\*%betaco)/ (1+exp(X%\*%betaco))

gradient <- t(X)%\*%(Y-pii)

hessian<- -t(X)%\*%diag(c(pii\*(1-pii)), length(Y))%\*%X

return(list(gradient, hessian))

}

MLE[i,] <- newtonraphson(ftn, c(0,0,0))

MLEglm [i,]<- glm(Y~gpa+gre,family = binomial)$coef

}

(meanMLE <- colSums(MLE)/no.rep)

(meanMLEglm <- colSums(MLEglm)/no.rep)

(meanMLE-betaco) # bias

(meanMLEglm-betaco) # bias

MLE230 <- MLE

R. output:

(meanMLE <- colSums(MLE)/no.rep)

[1] -2.2130469319 0.5785163150 0.0005697193

> (meanMLEglm <- colSums(MLEglm)/no.rep)

[1] -2.2130469319 0.5785163150 0.0005697193

> (meanMLE-betaco) # bias

[1] 3.786953068 -0.421483685 -0.004430281

> (meanMLEglm-betaco) # bias

[1] 3.786953068 -0.421483685 -0.004430281

> MLE230 <- MLE

# (3) the sample size is 430

betaco <- c(-6,1,0.005)

n <- 430

no.rep <- 1000

pii<-c()

X <- c()

Y <- c()

for(i in 1:n){

set.seed(i)

gpa <- rnorm(n,3.1,0.3)

gre <- rnorm(n,580,80)

(pii[i] <- exp(betaco[1]+betaco[2]\*gpa+betaco[3]\*gre)/(1+exp(betaco[1]+betaco[2]\*gpa+betaco[3]\*gre)))

(Y[i]<- sample(c(0,1),1,c(1-pii[i],pii[i]),replace = F))

# above: data generation process

# below: data analysis process

(X <- cbind(rep(1,length(Y)),gpa,gre))

}

MLE <- matrix(NA,no.rep,3)

MLEglm <- matrix(NA,no.rep,3)

for(i in 1:no.rep){

set.seed(i)

X<- cbind(rep(1,length(Y)),gpa,gre)

ftn <- function(betaco) {

pii <- exp(X%\*%betaco)/ (1+exp(X%\*%betaco))

gradient <- t(X)%\*%(Y-pii)

hessian<- -t(X)%\*%diag(c(pii\*(1-pii)), length(Y))%\*%X

return(list(gradient, hessian))

}

MLE[i,] <- newtonraphson(ftn, c(0,0,0))

MLEglm [i,]<- glm(Y~gpa+gre,family = binomial)$coef

}

(meanMLE <- colSums(MLE)/no.rep)

(meanMLEglm <- colSums(MLEglm)/no.rep)

(meanMLE-betaco) # bias

(meanMLEglm-betaco) # bias

MLE430 <- MLE

R. output:

> (meanMLE <- colSums(MLE)/no.rep)

[1] -2.2130469319 0.5785163150 0.0005697193

> (meanMLEglm <- colSums(MLEglm)/no.rep)

[1] -2.2130469319 0.5785163150 0.0005697193

> (meanMLE-betaco) # bias

[1] 3.786953068 -0.421483685 -0.004430281

> (meanMLEglm-betaco) # bias

[1] 3.786953068 -0.421483685 -0.004430281

> MLE430 <- MLE

# (4) the sample size is 630

betaco <- c(-6,1,0.005)

n <- 630

no.rep <- 1000

pii<-c()

X <- c()

Y <- c()

for(i in 1:n){

set.seed(i)

gpa <- rnorm(n,3.1,0.3)

gre <- rnorm(n,580,80)

(pii[i] <- exp(betaco[1]+betaco[2]\*gpa+betaco[3]\*gre)/(1+exp(betaco[1]+betaco[2]\*gpa+betaco[3]\*gre)))

(Y[i]<- sample(c(0,1),1,c(1-pii[i],pii[i]),replace = F))

# above: data generation process

# below: data analysis process

(X <- cbind(rep(1,length(Y)),gpa,gre))

}

MLE <- matrix(NA,no.rep,3)

MLEglm <- matrix(NA,no.rep,3)

for(i in 1:no.rep){

set.seed(i)

X<- cbind(rep(1,length(Y)),gpa,gre)

ftn <- function(betaco) {

pii <- exp(X%\*%betaco)/ (1+exp(X%\*%betaco))

gradient <- t(X)%\*%(Y-pii)

hessian<- -t(X)%\*%diag(c(pii\*(1-pii)), length(Y))%\*%X

return(list(gradient, hessian))

}

MLE[i,] <- newtonraphson(ftn, c(0,0,0))

MLEglm [i,]<- glm(Y~gpa+gre,family = binomial)$coef

}

(meanMLE <- colSums(MLE)/no.rep)

(meanMLEglm <- colSums(MLEglm)/no.rep)

(meanMLE-betaco) # bias

(meanMLEglm-betaco) # bias

MLE630 <- MLE

R. output:

> (meanMLE <- colSums(MLE)/no.rep)

[1] -2.2130469319 0.5785163150 0.0005697193

> (meanMLEglm <- colSums(MLEglm)/no.rep)

[1] -2.2130469319 0.5785163150 0.0005697193

> (meanMLE-betaco) # bias

[1] 3.786953068 -0.421483685 -0.004430281

> (meanMLEglm-betaco) # bias

[1] 3.786953068 -0.421483685 -0.004430281

> MLE630 <- MLE

# make box plots

par(mfrow = c(3,1))

boxplot(MLE30[,1],MLE230[,1],MLE430[,1],MLE630[,1])

abline(h=betaco[1],col=2)

boxplot(MLE30[,2],MLE230[,2],MLE430[,2],MLE630[,2])

abline(h=betaco[2],col=2)

boxplot(MLE30[,3],MLE230[,3],MLE430[,3],MLE630[,3])

abline(h=betaco[3],col=2)

