ผนวก ง

โปรแกรมประมวลผล

ตัวอย่างโปรแกรมการคำนวณค่าเฉลี่ยของความคลาดเคลื่อนกำลังสองเฉลี่ย (AMSE) และค่าส่วนเบี่ยงเบนมาตรฐานของค่าเฉลี่ยของความคลาดเคลื่อนกำลังสองเฉลี่ย ในกรณีที่ จำนวนตัวแปรอิสระเท่ากับ 3 ค่าส่วนเบี่ยงเบนมาตรฐานเท่ากับ 1 และ ขนาดตัวอย่างเท่ากับ 10

```
# โปรแกรมย่อยในการสร้างตัวแปรตาม (y)
fy <- function(samplesize,sde,x,numx)
    {
        error <- rnorm(samplesize,0,sde)
        ones <- rep(1,samplesize)
        xones <- cbind(ones,x)
        beta <- rep(1,numx + 1)
        y <- (xones %*% beta) + error
        return(y)
    }
# จำนวนตัวแปรอิสระเท่ากับ 3
numx <- 3
# ค่าส่วนเบี่ยงเบนมาตรฐานเท่ากับ 1
sde <- 1
# จำนวนขนาดตัวอย่างเท่ากับ 10
samplesize <- 10
# จำนวนการทำซ้ำเท่ากับ 500
numloop <- 500
# สร้างตัวแปรอิสระที่มีการแจกแจงแบบปกติขนาดตาม samplesize มีค่าเฉลี่ยเป็น
    (0,0,...,numx) และความแปรปรวน เป็นเมตริกซ์เอกลักษณ์ขนาด numx
x <- rmvnorm(samplesize,mean=rep(0,numx),cov=diag(rep(1,numx)),d=numx)
```

```
# กำหนดค่าเริ่มต้นของการคำนวณค่า MSE แต่ละวิธีให้เก็บค่าเท่ากับจำนวน numloop
msesr <- rep(0,numloop)
mseaicri <- rep(0,numloop)
msekicc <- rep(0,numloop)
msekici <- rep(0,numloop)
msegncv <- rep(0,numloop)
# เริ่มต้นโปรแกรม
for ( i in 1:numloop)
    {
        y <- fy(samplesize,sde,x,numx)
# stepwise regression method
        resultsr <- stepwise(x,y,intercept=T)
        numrow <- nrow(resultsr$which)
        regmodel <- lm(y~x[,resultsr$which[numrow,]])
        msesr[i] <- sum(regmodel$residuals^2)/regmodel$df
# AICri & KICc method
        reg <-
        leaps(x,y,rep(1,samplesize),int=T,method="adjr2",keep.int=T,nbest=1,df=sampl
        esize)
        rowwhich <- nrow(reg$which)
        aicri <- matrix(nrow=rowwhich,ncol=1)
        kicc <- matrix(nrow=rowwhich,ncol=1)
        for ( I in 1:rowwhich)
            {
                regmodel <- lm(y~x[,reg$which[I,]])
                v <- sum(regmodel$residuals^2)/samplesize
                a1 <- samplesize*(samplesize+reg$size[I])/(samplesize-reg$size[I]-2)
                a2 <- 2*samplesize*reg$size[I]*(samplesize^(3/4-2/5))/((samplesize-
                reg$size[I]-2)*(samplesize-reg$size[I]))
                aicri[I] <- samplesize*log(v,exp(1))+a1-a2
```

```
            k1 <- samplesize*log(samplesize/(samplesize-reg$size[l]),exp(1))
            k2 <- samplesize*((samplesize+reg$size[l])*(samplesize-
            reg$size[l])+(samplesize-reg$size[l]-2))/((samplesize-reg$size[l]-
            2)*(samplesize-reg$size[l]))
            kicc[l] <- samplesize*log(v,exp(1))+k1+k2
        }
    minaicri <- min(aicri)
    minkicc <- min(kicc)
    for (j in 1:rowwhich)
    {
        if ( aicri[j] == minaicri )
            {
                regmodel <- lm(y~x[,reg$which[j,]])
                mseaicri[i] <- sum(regmodel$residuals^2)/regmodel$df
            }
        if ( kicc[j] == minkicc )
            {
                regmodel <- lm(y~x[,reg$which[j,]])
                msekicc[i] <- sum(regmodel$residual^2)/regmodel$df
            }
    }
# KICI method
    reg <-
    leaps(x,y,rep(1,samplesize),int=T,method="adjr2",keep.int=T,nbest=1,df=sampl
    esize)
    rowwhich <- nrow(reg$which)
    kici <- matrix(nrow=rowwhich,ncol=1)
    for ( l in 1:rowwhich)
        {
            regmodel <- lm(y~x[,reg$which[l,]])
```

```
            v <- sum(regmodel$residuals^2)/samplesize
            b <- matrix(nrow=100,ncol=1)
            for ( k in 1:100)
                {
                    y1 <- rnorm(samplesize,0,1)
                    regmodel1 <- lm(y1~x[,reg$which[l,]])
                    va <- sum(regmodel1$residuals^2)/samplesize
                    regmodel1.p <- predict(regmodel1)
                    yp.sq <- sum(regmodel1.p^2)
                    b[k] <- (-
                    samplesize*log(va,exp(1)))+(samplesize/va)+(yp.sq/va)+(sample
                    size*va)+yp.sq-2*samplesize
                    }
        biasadj <- mean(b)
        kici[l] <- samplesize*log(v,exp(1))+samplesize+biasadj
            }
        minkici <- min(kici)
        for ( j in 1:rowwhich)
            {
                if ( kici[j] == minkici)
                    {
                        regmodel <- lm(y~x[,reg$which[j,]])
                        msekici[i] <- sum(regmodel$residuals^2)/regmodel$df
                    }
            }
# gncv
    full <- lm(y~x)
    full.cof <- full$coefficients
    full.e <- full$residuals
    abscof <- abs(full.cof)
```

```
a <- numx + 1
newcof <- matrix(nrow=a,ncol=1)
for (l in 1:a)
    {
        if (abscof[l] >= 0.01) newcof[l] <- abscof[l]
        else if (abscof[l] > 0 && abscof[l] < 0.01) newcof[l] <-
        0.01*sign(full.cof[l])
        else newcof[l] <- 0.01
    }
d <- matrix(nrow=samplesize,ncol=1)
ones <- rep(1,samplesize)
xones <- cbind(ones,x)
u <- xones %*% newcof + full.e
for (r in 1:samplesize)
    {
        u1 <- lm(u[-r]~x[-r,])
        u1.p <- xones[r,]%*%u1$coefficients
        d[r] <- (u[r]-u1.p)^2
    }
dn.0 <- sum(d)
dn <- matrix(nrow=numx,ncol=1)
for ( l in 1:numx )
    {
        d1 <- matrix(nrow=samplesize,ncol=1)
        for ( r in 1:samplesize )
            {
                regmodel <- lm( u[-r]~x[-r,-l] )
                xones1 <- c(1,x[r,-l])
                u.p <- xones1 %*% regmodel$coefficients
                d1[r] <- (u[r]-u.p)^2
```

```
            }
        dn[l] <- sum(d1)
    }
delta <- matrix(nrow=numx,ncol=1)
for ( l in 1:numx ) delta[l] <- dn[l]-dn.0
mindelta <- min(delta)
cn.r <- abs(mindelta)
reg <-
leaps(x,y,rep(1,samplesize),int=T,method="adjr2",keep.int=T,nbest=1,df=sampl
esize)
rowwhich <- nrow(reg$which)
gncv <- matrix(nrow=rowwhich,ncol=1)
for ( l in 1:rowwhich )
    {
        regmodel <- lm(y~x[,reg$which[l,]])
        gn <- matrix(nrow=samplesize,ncol=1)
        for ( r in 1:samplesize )
            {
                regmodel2 <- lm(y[-r] ~ x[-r,reg$which[l,]])
                x1 <- c(1,x[r,reg$which[l,]])
                y.hat <- x1 %*% regmodel2$coefficients
                gn[r] <- (y[r]-y.hat)^2
            }
        gncv[l] <- sum(gn)+reg$size[l]*cn.r
    }
mingncv <- min(gncv)
for (j in 1:rowwhich)
    {
        if (gncv[j] == mingncv)
            {
```

```
                    regmodel <- lm(y~x[,reg$which[j,]])
                    msegncv[i] <- sum(regmodel$residuals^2)/regmodel$df
                }
            }
        }
amsesr <- mean(msesr)
amseaicri <- mean(mseaicri)
amsekicc <- mean(msekicc)
amsekici <- mean(msekici)
amsegncv <- mean(msegncv)
stdsr <- matrix(nrow=numloop,ncol=1)
stdaicri <- matrix(nrow=numloop,ncol=1)
stdkicc <- matrix(nrow=numloop,ncol=1)
stdkici <- matrix(nrow=numloop,ncol=1)
stdgncv <- matrix(nrow=numloop,ncol=1)
for ( indexi in 1:numloop)
{
stdaicri[indexi] <- (mseaicri[indexi]-amseaicri)^2
stdkicc[indexi] <- (msekicc[indexi]-amsekicc)^2
stdkici[indexi] <- (msekici[indexi]-amsekici)^2
stdgncv[indexi] <- (msegncv[indexi]-amsegncv)^2
stdsr[indexi] <- (msesr[indexi]-amsesr)^2
}
stdamseaicri <- sqrt(sum(stdaicri)/(numloop-1))
stdamsekicc <- sqrt(sum(stdkicc)/(numloop-1))
stdamsekici <- sqrt(sum(stdkici)/(numloop-1))
stdamsegncv <- sqrt(sum(stdgncv)/(numloop-1))
stdamsesr <- sqrt(sum(stdsr)/(numloop-1))
# ให้แสดงค่า AMSE และ ค่าส่วนเบี่ยงเบนมาตรฐานของ AMSE แต่ละวิธี
print(c(amseaicri,stdamseaicri))
```

```
print(c(amsekicc,stdamsekicc))
print(c(amsekici,stdamsekici))
print(c(amsegncv,stdamsegncv))
print(c(amsesr,stdamsesr))
```