รายการอ้างอิง

ภาษาไทย

กัลยา วานิชย์บัญชา. การวิเคราะห์สถิติ : สถิติสำหรับการบริหารและวิจัย. พิมพ์ครั้งที่6. กรุงเทพฯ : โรงพิมพ์แห่งจุฬาลงกรณ์มหาวิทยาลัย, 2545.

กัลยา วานิชย์บัญชา. การวิเคราะห์ข้อมูลหลายตัวแปร. พิมพ์ครั้งที่ 3. กรุงเทพฯ : สำนักพิมพ์ธรรมสาร, 2551.

ภาษาอังกฤษ

Agresti, A. Categorical Data Analysis, United States of America : John Wiley and Sons, 2002.

Hadjicostas, P. & Hadjinicola, G.C. The asymptotic distribution of the proportion of correct classification for a holdout sample in logistic regression. Journal of Statistical Planning and Inference 92 (2001) : 193-211.

Hadjicostas, P. Maximizing proportions of correct classifications in binary logistic regression. Journal of Applied Statistics 33 (2006) : 629-640.

Hosmer, D.W. & Lemeshow, S. Applied Logistic Regression. New York: Wiley, 2000.

ภาคผนวก

ตัวอย่างการใช้โปรแกรม R ในการดำเนินงานวิจัย

Main Program

```
options(digits=20)
dataD0 <- new("list")
dataD1 <- new("list")
dataD2 <- new("list")
dataD3 <- new("list")
errorTerm <- new("list") #create error term
Y0est  <- new("list")
Y1est  <- new("list")
Y2est  <- new("list")
Y3est  <- new("list")
MD0 <- new("list")
MD1 <- new("list")
MD2 <- new("list")
MD3 <- new("list")

MD0A01 <- new("list")
MD0A05 <- new("list")
MD0A09 <- new("list")
MD1A01 <- new("list")
MD1A05 <- new("list")
MD1A09 <- new("list")
MD2A01 <- new("list")
MD2A05 <- new("list")
MD2A09 <- new("list")
MD3A01 <- new("list")
MD3A05 <- new("list")
MD3A09 <- new("list")
```

```
MD0A01L <- new("list")
MD0A05L <- new("list")
MD0A09L <- new("list")
MD1A01L <- new("list")
MD1A05L <- new("list")
MD1A09L <- new("list")
MD2A01L <- new("list")
MD2A05L <- new("list")
MD2A09L <- new("list")
MD3A01L <- new("list")
MD3A05L <- new("list")
MD3A09L <- new("list")

MD0A01LMI <- new("list")
MD0A05LMI <- new("list")
MD0A09LMI <- new("list")
MD1A01LMI <- new("list")
MD1A05LMI <- new("list")
MD1A09LMI <- new("list")
MD2A01LMI <- new("list")
MD2A05LMI <- new("list")
MD2A09LMI <- new("list")
MD3A01LMI <- new("list")
MD3A05LMI <- new("list")
MD3A09LMI <- new("list")

MD0A01LAI <- new("list")
MD0A05LAI <- new("list")
MD0A09LAI <- new("list")
MD1A01LAI <- new("list")
```

```
MD1A05LAI <- new("list")
MD1A09LAI <- new("list")
MD2A01LAI <- new("list")
MD2A05LAI <- new("list")
MD2A09LAI <- new("list")
MD3A01LAI <- new("list")
MD3A05LAI <- new("list")
MD3A09LAI <- new("list")

MD0A01LCP <- new("list")
MD0A05LCP <- new("list")
MD0A09LCP <- new("list")
MD1A01LCP <- new("list")
MD1A05LCP <- new("list")
MD1A09LCP <- new("list")
MD2A01LCP <- new("list")
MD2A05LCP <- new("list")
MD2A09LCP <- new("list")
MD3A01LCP <- new("list")
MD3A05LCP <- new("list")
MD3A09LCP <- new("list")

MD0A01LCER <- new("list")
MD0A05LCER <- new("list")
MD0A09LCER <- new("list")
MD1A01LCER <- new("list")
MD1A05LCER <- new("list")
MD1A09LCER <- new("list")
MD2A01LCER <- new("list")
MD2A05LCER <- new("list")
```

```
MD2A09LCER <- new("list")
MD3A01LCER <- new("list")
MD3A05LCER <- new("list")
MD3A09LCER <- new("list")

#INPUT
  n <- 20
  p <- c(1, 2, 3, 4, 5, 6)
  b <- 0.1
loop <- 500
seed <- 1
  i <- 1   #start loop, check condition
set.seed(seed)
errorTerm <- numError(n, 0, 5, loop)
while(n/p[i]>=20 & i<=length(p)){
   #Generate random number
   set.seed(seed)
   dataD0[[i]] <- numGen(n, p[i], 0, 0.1, loop)
   set.seed(seed)
   dataD1[[i]] <- numGen(n, p[i], (1/3), 0.1, loop)
   set.seed(seed)
   dataD2[[i]] <- numGen(n, p[i], (2/3), 0.1, loop)
   set.seed(seed)
   dataD3[[i]] <- numGen(n, p[i], 0.99, 0.1, loop)

   #Calcalation Y (dependent variable)
   Y0est[[i]] <- lapply(dataD0[[i]], FUN=tBeta, b)
   Y0est[[i]] <- CalcY(Y0est[[i]], errorTerm)
   Y1est[[i]] <- lapply(dataD1[[i]], FUN=tBeta, b)
   Y1est[[i]] <- CalcY(Y1est[[i]], errorTerm)
```

```
Y2est[[i]] <- lapply(dataD2[[i]], FUN=tBeta, b)
Y2est[[i]] <- CalcY(Y2est[[i]], errorTerm)
Y3est[[i]] <- lapply(dataD3[[i]], FUN=tBeta, b)
Y3est[[i]] <- CalcY(Y3est[[i]], errorTerm)

#Merge data and transform to data frame
MD0[[i]] <- JoinData(dataD0[[i]], errorTerm, Y0est[[i]])
MD1[[i]] <- JoinData(dataD1[[i]], errorTerm, Y1est[[i]])
MD2[[i]] <- JoinData(dataD2[[i]], errorTerm, Y2est[[i]])
MD3[[i]] <- JoinData(dataD3[[i]], errorTerm, Y3est[[i]])

#Calculation the Ynew in binary(0,1)
MD0A01[[i]] <- CalcYnew(MD0[[i]], 0.1)
MD0A05[[i]] <- CalcYnew(MD0[[i]], 0.5)
MD0A09[[i]] <- CalcYnew(MD0[[i]], 0.9)
MD1A01[[i]] <- CalcYnew(MD1[[i]], 0.1)
MD1A05[[i]] <- CalcYnew(MD1[[i]], 0.5)
MD1A09[[i]] <- CalcYnew(MD1[[i]], 0.9)
MD2A01[[i]] <- CalcYnew(MD2[[i]], 0.1)
MD2A05[[i]] <- CalcYnew(MD2[[i]], 0.5)
MD2A09[[i]] <- CalcYnew(MD2[[i]], 0.9)
MD3A01[[i]] <- CalcYnew(MD3[[i]], 0.1)
MD3A05[[i]] <- CalcYnew(MD3[[i]], 0.5)
MD3A09[[i]] <- CalcYnew(MD3[[i]], 0.9)

#Calculation the Ypred from x independent variables
if(i==1){
   MD0A01L[[i]] <- CalcYpred1x(MD0A01[[i]], "logit")
   MD0A05L[[i]] <- CalcYpred1x(MD0A05[[i]], "logit")
   MD0A09L[[i]] <- CalcYpred1x(MD0A09[[i]], "logit")
```

```
    MD1A01L[[i]] <- CalcYpred1x(MD1A01[[i]], "logit")
    MD1A05L[[i]] <- CalcYpred1x(MD1A05[[i]], "logit")
    MD1A09L[[i]] <- CalcYpred1x(MD1A09[[i]], "logit")
    MD2A01L[[i]] <- CalcYpred1x(MD2A01[[i]], "logit")
    MD2A05L[[i]] <- CalcYpred1x(MD2A05[[i]], "logit")
    MD2A09L[[i]] <- CalcYpred1x(MD2A09[[i]], "logit")
    MD3A01L[[i]] <- CalcYpred1x(MD3A01[[i]], "logit")
    MD3A05L[[i]] <- CalcYpred1x(MD3A05[[i]], "logit")
    MD3A09L[[i]] <- CalcYpred1x(MD3A09[[i]], "logit")
}
if(i==2){
    MD0A01L[[i]] <- CalcYpred2x(MD0A01[[i]], "logit")
    MD0A05L[[i]] <- CalcYpred2x(MD0A05[[i]], "logit")
    MD0A09L[[i]] <- CalcYpred2x(MD0A09[[i]], "logit")
    MD1A01L[[i]] <- CalcYpred2x(MD1A01[[i]], "logit")
    MD1A05L[[i]] <- CalcYpred2x(MD1A05[[i]], "logit")
    MD1A09L[[i]] <- CalcYpred2x(MD1A09[[i]], "logit")
    MD2A01L[[i]] <- CalcYpred2x(MD2A01[[i]], "logit")
    MD2A05L[[i]] <- CalcYpred2x(MD2A05[[i]], "logit")
    MD2A09L[[i]] <- CalcYpred2x(MD2A09[[i]], "logit")
    MD3A01L[[i]] <- CalcYpred2x(MD3A01[[i]], "logit")
    MD3A05L[[i]] <- CalcYpred2x(MD3A05[[i]], "logit")
    MD3A09L[[i]] <- CalcYpred2x(MD3A09[[i]], "logit")
}
if(i==3){
    MD0A01L[[i]] <- CalcYpred3x(MD0A01[[i]], "logit")
    MD0A05L[[i]] <- CalcYpred3x(MD0A05[[i]], "logit")
    MD0A09L[[i]] <- CalcYpred3x(MD0A09[[i]], "logit")
    MD1A01L[[i]] <- CalcYpred3x(MD1A01[[i]], "logit")
    MD1A05L[[i]] <- CalcYpred3x(MD1A05[[i]], "logit")
```

```
    MD1A09L[[i]] <- CalcYpred3x(MD1A09[[i]], "logit")
    MD2A01L[[i]] <- CalcYpred3x(MD2A01[[i]], "logit")
    MD2A05L[[i]] <- CalcYpred3x(MD2A05[[i]], "logit")
    MD2A09L[[i]] <- CalcYpred3x(MD2A09[[i]], "logit")
    MD3A01L[[i]] <- CalcYpred3x(MD3A01[[i]], "logit")
    MD3A05L[[i]] <- CalcYpred3x(MD3A05[[i]], "logit")
    MD3A09L[[i]] <- CalcYpred3x(MD3A09[[i]], "logit")
}
if(i==4){
    MD0A01L[[i]] <- CalcYpred4x(MD0A01[[i]], "logit")
    MD0A05L[[i]] <- CalcYpred4x(MD0A05[[i]], "logit")
    MD0A09L[[i]] <- CalcYpred4x(MD0A09[[i]], "logit")
    MD1A01L[[i]] <- CalcYpred4x(MD1A01[[i]], "logit")
    MD1A05L[[i]] <- CalcYpred4x(MD1A05[[i]], "logit")
    MD1A09L[[i]] <- CalcYpred4x(MD1A09[[i]], "logit")
    MD2A01L[[i]] <- CalcYpred4x(MD2A01[[i]], "logit")
    MD2A05L[[i]] <- CalcYpred4x(MD2A05[[i]], "logit")
    MD2A09L[[i]] <- CalcYpred4x(MD2A09[[i]], "logit")
    MD3A01L[[i]] <- CalcYpred4x(MD3A01[[i]], "logit")
    MD3A05L[[i]] <- CalcYpred4x(MD3A05[[i]], "logit")
    MD3A09L[[i]] <- CalcYpred4x(MD3A09[[i]], "logit")
}
if(i==5){
    MD0A01L[[i]] <- CalcYpred5x(MD0A01[[i]], "logit")
    MD0A05L[[i]] <- CalcYpred5x(MD0A05[[i]], "logit")
    MD0A09L[[i]] <- CalcYpred5x(MD0A09[[i]], "logit")
    MD1A01L[[i]] <- CalcYpred5x(MD1A01[[i]], "logit")
    MD1A05L[[i]] <- CalcYpred5x(MD1A05[[i]], "logit")
    MD1A09L[[i]] <- CalcYpred5x(MD1A09[[i]], "logit")
    MD2A01L[[i]] <- CalcYpred5x(MD2A01[[i]], "logit")
```

```
    MD2A05L[[i]] <- CalcYpred5x(MD2A05[[i]], "logit")
    MD2A09L[[i]] <- CalcYpred5x(MD2A09[[i]], "logit")
    MD3A01L[[i]] <- CalcYpred5x(MD3A01[[i]], "logit")
    MD3A05L[[i]] <- CalcYpred5x(MD3A05[[i]], "logit")
    MD3A09L[[i]] <- CalcYpred5x(MD3A09[[i]], "logit")
}
if(i==6){
    MD0A01L[[i]] <- CalcYpred6x(MD0A01[[i]], "logit")
    MD0A05L[[i]] <- CalcYpred6x(MD0A05[[i]], "logit")
    MD0A09L[[i]] <- CalcYpred6x(MD0A09[[i]], "logit")
    MD1A01L[[i]] <- CalcYpred6x(MD1A01[[i]], "logit")
    MD1A05L[[i]] <- CalcYpred6x(MD1A05[[i]], "logit")
    MD1A09L[[i]] <- CalcYpred6x(MD1A09[[i]], "logit")
    MD2A01L[[i]] <- CalcYpred6x(MD2A01[[i]], "logit")
    MD2A05L[[i]] <- CalcYpred6x(MD2A05[[i]], "logit")
    MD2A09L[[i]] <- CalcYpred6x(MD2A09[[i]], "logit")
    MD3A01L[[i]] <- CalcYpred6x(MD3A01[[i]], "logit")
    MD3A05L[[i]] <- CalcYpred6x(MD3A05[[i]], "logit")
    MD3A09L[[i]] <- CalcYpred6x(MD3A09[[i]], "logit")
}
# Calculation the M(i)
MD0A01LMI[[i]] <- CalcMI(MD0A01L[[i]])
MD0A05LMI[[i]] <- CalcMI(MD0A05L[[i]])
MD0A09LMI[[i]] <- CalcMI(MD0A09L[[i]])
MD1A01LMI[[i]] <- CalcMI(MD1A01L[[i]])
MD1A05LMI[[i]] <- CalcMI(MD1A05L[[i]])
MD1A09LMI[[i]] <- CalcMI(MD1A09L[[i]])
MD2A01LMI[[i]] <- CalcMI(MD2A01L[[i]])
MD2A05LMI[[i]] <- CalcMI(MD2A05L[[i]])
MD2A09LMI[[i]] <- CalcMI(MD2A09L[[i]])
```

```
MD3A01LMI[[i]] <- CalcMI(MD3A01L[[i]])
MD3A05LMI[[i]] <- CalcMI(MD3A05L[[i]])
MD3A09LMI[[i]] <- CalcMI(MD3A09L[[i]])

# Calculation the a(i)
MD0A01LAI[[i]] <- CalcAI(MD0A01LMI[[i]])
MD0A05LAI[[i]] <- CalcAI(MD0A05LMI[[i]])
MD0A09LAI[[i]] <- CalcAI(MD0A09LMI[[i]])
MD1A01LAI[[i]] <- CalcAI(MD1A01LMI[[i]])
MD1A05LAI[[i]] <- CalcAI(MD1A05LMI[[i]])
MD1A09LAI[[i]] <- CalcAI(MD1A09LMI[[i]])
MD2A01LAI[[i]] <- CalcAI(MD2A01LMI[[i]])
MD2A05LAI[[i]] <- CalcAI(MD2A05LMI[[i]])
MD2A09LAI[[i]] <- CalcAI(MD2A09LMI[[i]])
MD3A01LAI[[i]] <- CalcAI(MD3A01LMI[[i]])
MD3A05LAI[[i]] <- CalcAI(MD3A05LMI[[i]])
MD3A09LAI[[i]] <- CalcAI(MD3A09LMI[[i]])

# Calculation the cp
MD0A01LCP[[i]] <- cut.of.point(MD0A01LAI[[i]])
MD0A05LCP[[i]] <- cut.of.point(MD0A05LAI[[i]])
MD0A09LCP[[i]] <- cut.of.point(MD0A09LAI[[i]])
MD1A01LCP[[i]] <- cut.of.point(MD1A01LAI[[i]])
MD1A05LCP[[i]] <- cut.of.point(MD1A05LAI[[i]])
MD1A09LCP[[i]] <- cut.of.point(MD1A09LAI[[i]])
MD2A01LCP[[i]] <- cut.of.point(MD2A01LAI[[i]])
MD2A05LCP[[i]] <- cut.of.point(MD2A05LAI[[i]])
MD2A09LCP[[i]] <- cut.of.point(MD2A09LAI[[i]])
MD3A01LCP[[i]] <- cut.of.point(MD3A01LAI[[i]])
MD3A05LCP[[i]] <- cut.of.point(MD3A05LAI[[i]])
```

```
    MD3A09LCP[[i]] <- cut.of.point(MD3A09LAI[[i]])

    #MD0A01LCER[[i]] <- cer.value(MD0A01LAI[[i]])
    #MD0A05LCER[[i]] <- cer.value(MD0A05LAI[[i]])
    #MD0A09LCER[[i]] <- cer.value(MD0A09LAI[[i]])
    #MD1A01LCER[[i]] <- cer.value(MD1A01LAI[[i]])
    #MD1A05LCER[[i]] <- cer.value(MD1A05LAI[[i]])
    #MD1A09LCER[[i]] <- cer.value(MD1A09LAI[[i]])
    #MD2A01LCER[[i]] <- cer.value(MD2A01LAI[[i]])
    #MD2A05LCER[[i]] <- cer.value(MD2A05LAI[[i]])
    #MD2A09LCER[[i]] <- cer.value(MD2A09LAI[[i]])
    #MD3A01LCER[[i]] <- cer.value(MD3A01LAI[[i]])
    #MD3A05LCER[[i]] <- cer.value(MD3A05LAI[[i]])
    #MD3A09LCER[[i]] <- cer.value(MD3A09LAI[[i]])

    print(paste(n, p[i]))  #for checking my program
    i <- i+1
}
```

Generalized Linear Models, Link: Logit

```
slcList <- length(MD1A01LCP)
slcList <- ifelse(slcList<2, 2, slcList)
MD0A01Ldt <- unlistMatrix(MD0A01LCP, loop)  #1,2,...
MD0A05Ldt <- unlistMatrix(MD0A05LCP, loop)  #1,2
MD0A09Ldt <- unlistMatrix(MD0A09LCP, loop)  #1,2
MD1A01Ldt <- unlistMatrix(MD1A01LCP, loop)[2:slcList]  #2,...
MD1A05Ldt <- unlistMatrix(MD1A05LCP, loop)[2:slcList]  #2,...
MD1A09Ldt <- unlistMatrix(MD1A09LCP, loop)[2:slcList]  #2,...
MD2A01Ldt <- unlistMatrix(MD2A01LCP, loop)[2:slcList]  #2,...
MD2A05Ldt <- unlistMatrix(MD2A05LCP, loop)[2:slcList]  #2,...
MD2A09Ldt <- unlistMatrix(MD2A09LCP, loop)[2:slcList]  #2,...
```

```
MD3A01Ldt <- unlistMatrix(MD3A01LCP, loop)[2:slcList] #2,...
MD3A05Ldt <- unlistMatrix(MD3A05LCP, loop)[2:slcList] #2,...
MD3A09Ldt <- unlistMatrix(MD3A09LCP, loop)[2:slcList] #2,...

if(n!=20){
  #Logit: Calculation percent of C0
  MD0A01LC0 <- lapply(MD0A01Ldt,FUN=function(x){mean(x[,1])*100})
  MD0A05LC0 <- lapply(MD0A05Ldt,FUN=function(x){mean(x[,1])*100})
  MD0A09LC0 <- lapply(MD0A09Ldt,FUN=function(x){mean(x[,1])*100})
  MD1A01LC0 <- lapply(MD1A01Ldt,FUN=function(x){mean(x[,1])*100})
  MD1A05LC0 <- lapply(MD1A05Ldt,FUN=function(x){mean(x[,1])*100})
  MD1A09LC0 <- lapply(MD1A09Ldt,FUN=function(x){mean(x[,1])*100})
  MD2A01LC0 <- lapply(MD2A01Ldt,FUN=function(x){mean(x[,1])*100})
  MD2A05LC0 <- lapply(MD2A05Ldt,FUN=function(x){mean(x[,1])*100})
  MD2A09LC0 <- lapply(MD2A09Ldt,FUN=function(x){mean(x[,1])*100})
  MD3A01LC0 <- lapply(MD3A01Ldt,FUN=function(x){mean(x[,1])*100})
  MD3A05LC0 <- lapply(MD3A05Ldt,FUN=function(x){mean(x[,1])*100})
  MD3A09LC0 <- lapply(MD3A09Ldt,FUN=function(x){mean(x[,1])*100})
  #Logit: Confidence Interval
  MD0A01LCI <- lapply(MD0A01Ldt,FUN=function(x){percentile(x[,1])})
  MD0A05LCI <- lapply(MD0A05Ldt,FUN=function(x){percentile(x[,1])})
  MD0A09LCI <- lapply(MD0A09Ldt,FUN=function(x){percentile(x[,1])})
  MD1A01LCI <- lapply(MD1A01Ldt,FUN=function(x){percentile(x[,1])})
  MD1A05LCI <- lapply(MD1A05Ldt,FUN=function(x){percentile(x[,1])})
  MD1A09LCI <- lapply(MD1A09Ldt,FUN=function(x){percentile(x[,1])})
  MD2A01LCI <- lapply(MD2A01Ldt,FUN=function(x){percentile(x[,1])})
  MD2A05LCI <- lapply(MD2A05Ldt,FUN=function(x){percentile(x[,1])})
  MD2A09LCI <- lapply(MD2A09Ldt,FUN=function(x){percentile(x[,1])})
  MD3A01LCI <- lapply(MD3A01Ldt,FUN=function(x){percentile(x[,1])})
  MD3A05LCI <- lapply(MD3A05Ldt,FUN=function(x){percentile(x[,1])})
```

```
MD3A09LCI <- lapply(MD3A09Ldt,FUN=function(x){percentile(x[,1])})

pTemp <- c(rep(c(1:(n/20)),3), rep(c(2:(n/20)),9))
nTemp <- rep(n, length(pTemp))
#aTemp <- c(rep(0.1,(n/20)), rep(0.5,(n/20)), rep(0.9,(n/20)),
#          rep(c(0.1,0.5,0.9), (3*((n/20)-1))))
aTemp <- c(rep(0.1,(n/20)), rep(0.5,(n/20)), rep(0.9,(n/20)),
        rep(c(rep(0.1,(n/20-1)), rep(0.5,(n/20-1)), rep(0.9,(n/20-1))),3))
mTemp <- c(rep(0, (3*(n/20))), rep(0.33, (3*(n/20-1))),
        rep(0.67, (3*(n/20-1))), rep(0.99, (3*(n/20-1))))
perTempL <- c(unlist(MD0A01LC0),unlist(MD0A05LC0),unlist(MD0A09LC0),
        unlist(MD1A01LC0),unlist(MD1A05LC0),unlist(MD1A09LC0),
        unlist(MD2A01LC0),unlist(MD2A05LC0),unlist(MD2A09LC0),
        unlist(MD3A01LC0),unlist(MD3A05LC0),unlist(MD3A09LC0))
DataFinalL <- matrix(NA, length(pTemp), 5)
colnames(DataFinalL) <- c("n","p","m","a","percent")
DataFinalL[,1] <- nTemp
DataFinalL[,2] <- pTemp
DataFinalL[,3] <- mTemp
DataFinalL[,4] <- aTemp
DataFinalL[,5] <- perTempL
#Confidence Interval
CILMatrix <- matrix(c(unlist(MD0A01LCI),unlist(MD0A05LCI),unlist(MD0A09LCI),
        unlist(MD1A01LCI),unlist(MD1A05LCI),unlist(MD1A09LCI),
        unlist(MD2A01LCI),unlist(MD2A05LCI),unlist(MD2A09LCI),
        unlist(MD3A01LCI),unlist(MD3A05LCI),unlist(MD3A09LCI)),
        length(pTemp), 2, byrow=T)
colnames(CILMatrix) <- c("CI.Lower","CI.Upper")
allInfLogit <- cbind(DataFinalL, CILMatrix)
}
```

```
if(n==20){
   # Calculation percent of C0
   MD0A01LC0 <- lapply(MD0A01Ldt,FUN=function(x){mean(x[,1])*100})
   MD0A05LC0 <- lapply(MD0A05Ldt,FUN=function(x){mean(x[,1])*100})
   MD0A09LC0 <- lapply(MD0A09Ldt,FUN=function(x){mean(x[,1])*100})
   #Logit: Confidence Interval
   MD0A01LCI <- lapply(MD0A01Ldt,FUN=function(x){percentile(x[,1])})
   MD0A05LCI <- lapply(MD0A05Ldt,FUN=function(x){percentile(x[,1])})
   MD0A09LCI <- lapply(MD0A09Ldt,FUN=function(x){percentile(x[,1])})
   pTemp <- rep(1,3)
   nTemp <- rep(n, 3)
   aTemp <- c(0.1,0.5,0.9)
   mTemp <- rep(0,3)
   perTempL <- c(unlist(MD0A01LC0),unlist(MD0A05LC0),unlist(MD0A09LC0))
   DataFinalL <- matrix(NA, length(pTemp), 5)
   colnames(DataFinalL) <- c("n","p","m","a","percent")
   DataFinalL[,1] <- nTemp
   DataFinalL[,2] <- pTemp
   DataFinalL[,3] <- mTemp
   DataFinalL[,4] <- aTemp
   DataFinalL[,5] <- perTempL
   #Confidence Interval
   CILMatrix <- matrix(c(unlist(MD0A01LCI),unlist(MD0A05LCI),unlist(MD0A09LCI)),
         length(pTemp), 2, byrow=T)
   colnames(CILMatrix) <- c("CI.Lower","CI.Upper")
   allInfLogit <- cbind(DataFinalL, CILMatrix)
}
#Export data into Excel
write.csv(allInfLogit, paste("D:/LogitALL","_n",n,".csv",sep=""), row.names=F)
write.csv(DataFinalL, paste("D:/LogitDataInf","_n",n,".csv",sep=""), row.names=F)
```

Function

```
# Create matrix with the correlation value
cor.matrix <- function(p=1, corv=0){
   m1 <- 1
   m2 <- matrix(c(1, corv, corv, 1), 2, 2)
   dimnames(m2) <- list(c("x1","x2"), c("x1","x2"))
   m3 <- matrix(c(1, corv, corv^2, corv, 1, corv, corv^2, corv, 1), 3, 3)
   dimnames(m3) <- list(c("x1","x2","x3"), c("x1","x2","x3"))
   m4 <- matrix(c(1, corv, corv^2, corv^3, corv, 1, corv, corv^2, corv^2,
        corv, 1, corv, corv^3, corv^2, corv, 1), 4, 4)
   dimnames(m4) <- list(c("x1","x2","x3","x4"), c("x1","x2","x3","x4"))
   m5 <- matrix(c(1, corv, corv^2, corv^3, corv^4, corv, 1, corv, corv^2,
        corv^3, corv^2, corv, 1, corv, corv^2, corv^3, corv^2, corv,
        1, corv, corv^4, corv^3, corv^2, corv, 1), 5, 5)
   dimnames(m5) <- list(c("x1","x2","x3","x4","x5"), c("x1","x2","x3",
        "x4","x5"))
   m6 <- matrix(c(1, corv, corv^2, corv^3, corv^4, corv^5, corv, 1, corv,
        corv^2, corv^3, corv^4, corv^2, corv, 1, corv, corv^2, corv^3,
        corv^3, corv^2, corv, 1, corv, corv^2, corv^4, corv^3, corv^2,
        corv, 1, corv, corv^5, corv^4, corv^3, corv^2, corv, 1), 6, 6)
   dimnames(m6) <- list(c("x1","x2","x3","x4","x5","x6"), c("x1","x2",
        "x3","x4","x5","x6"))
   cor.mtx <- list(m1, m2, m3, m4, m5, m6)
   output <- cor.mtx[[p]]
   return(output)
}
# Number generator with correlation matrix
numGen <- function(n, p=1, cor.value, error=0.01, nloop=1){
   np <- p
   cv <- cor.value
```

```
    MC <- cor.matrix(np, cv)
    chol.matr <- chol(MC)
    num <- new("list")
    for(i in 1:nloop){
      stop.loop <- 0
      while(stop.loop<1){
        num[[i]] <- matrix(runif(n*p, -(n/2), n/2), n)
        num[[i]] <- num[[i]]%*%chol.matr
        if(max(abs(cor(num[[i]])-MC))<=error){
          stop.loop <- stop.loop+1
        }
        stop.loop
      }
    }
    output <- num
    return(output)
}
# Generator the error term
numError <- function(n, Mean=0, Var=1, nloop=1){
    num <- new("list")
    for(i in 1:nloop){
      num[[i]] <- rnorm(n, Mean, Var)
    }
    output <- num
    return(output)
}
# Calculation Y, Y=b0+b1X1+b2X2+...+bnXn
tBeta <- function(x, b){
    b+apply(rep(b, ncol(x))*x, 1, sum)
}
```

```
# Calculation dependent variable
CalcY <- function(x, y){
   Length <- length(x)
   Y.calc <- new("list")
   for(i in 1:Length){
      Y.calc[[i]] <- x[[i]]+y[[i]]
   }
   output <- Y.calc
   return(output)
}
# Merge data and transform to data frame
JoinData <- function(x, y, z){
   Length   <- length(x)
   DataTemp <- new("list")
   for(i in 1:Length){      .
      DataTemp[[i]] <- cbind(x[[i]], y[[i]], z[[i]])
      DataTemp[[i]] <- as.data.frame(DataTemp[[i]][order(DataTemp[[i]]
                  [,ncol(DataTemp[[i]])),])
   }
   output <- DataTemp
   return(output)
}
# Calculation the y in binary
CalcYnew <- function(dataProg, Prob){
   Length <- length(dataProg)
   Ynew   <- new("list")
   for(i in 1:Length){
      Ynew <- rep(1, nrow(dataProg[[i]]))
      dataProg[[i]]$Ynew <- Ynew
      dataProg[[i]]$Ynew[1:(Prob*nrow(dataProg[[i]]))] <- 0
```

```
    }
    output <- dataProg
    return(output)
}
# Calculation the Ypred from 1 independent variables
CalcYpred1x <- function(dataProg, Link="logit"){
    Length <- length(dataProg)
    Ypred  <- new("list")
    for(i in 1:Length){
        dataProgTr <- dataProg[[i]][,-c((ncol(dataProg[[i]])-1),
            (ncol(dataProg[[i]])-2))]
        Ypred[[i]] <- as.vector(fitted(glm(dataProgTr[,ncol(dataProgTr)]~
            dataProgTr[,1], family=binomial(link=Link),
            data=dataProgTr)))
        dataProg[[i]]$Ypred <- Ypred[[i]]
    }
    output <- dataProg
    return(output)
}
# Calculation the Ypred from 2 independent variables
CalcYpred2x <- function(dataProg, Link="logit"){
    Length <- length(dataProg)
    Ypred  <- new("list")
    for(i in 1:Length){
        dataProgTr <- dataProg[[i]][,-c((ncol(dataProg[[i]])-1),
            (ncol(dataProg[[i]])-2))]
        Ypred[[i]] <- as.vector(fitted(glm(dataProgTr[,ncol(dataProgTr)]~
            dataProgTr[,1]+dataProgTr[,2], family=binomial(link=Link),
            data=dataProgTr)))
        dataProg[[i]]$Ypred <- Ypred[[i]]
```

```
    }
    output <- dataProg
    return(output)
}
# Calculation the Ypred from 3 independent variables
CalcYpred3x <- function(dataProg, Link="logit"){
    Length <- length(dataProg)
    Ypred  <- new("list")
    for(i in 1:Length){
        dataProgTr <- dataProg[[i]][,-c((ncol(dataProg[[i]])-1),
            (ncol(dataProg[[i]])-2))]
        Ypred[[i]] <- as.vector(fitted(glm(dataProgTr[,ncol(dataProgTr)]~
            dataProgTr[,1]+dataProgTr[,2]+dataProgTr[,3], family=binomial(link=Link),
            data=dataProgTr)))
        dataProg[[i]]$Ypred <- Ypred[[i]]
    }
    output <- dataProg
    return(output)
}
# Calculation the Ypred from 4 independent variables
CalcYpred4x <- function(dataProg, Link="logit"){
    Length <- length(dataProg)
    Ypred  <- new("list")
    for(i in 1:Length){
        dataProgTr <- dataProg[[i]][,-c((ncol(dataProg[[i]])-1),
            (ncol(dataProg[[i]])-2))]
        Ypred[[i]] <- as.vector(fitted(glm(dataProgTr[,ncol(dataProgTr)]~
            dataProgTr[,1]+dataProgTr[,2]+dataProgTr[,3]+dataProgTr[,4],
            family=binomial(link=Link), data=dataProgTr)))
        dataProg[[i]]$Ypred <- Ypred[[i]]
```

```
  }
  output <- dataProg
  return(output)
}
# Calculation the Ypred from 5 independent variables
CalcYpred5x <- function(dataProg, Link="logit"){
  Length <- length(dataProg)
  Ypred  <- new("list")
  for(i in 1:Length){
    dataProgTr <- dataProg[[i]][,-c((ncol(dataProg[[i]])-1),
      (ncol(dataProg[[i]])-2))]
    Ypred[[i]] <- as.vector(fitted(glm(dataProgTr[,ncol(dataProgTr)]~
      dataProgTr[,1]+dataProgTr[,2]+dataProgTr[,3]+dataProgTr[,4]+
      dataProgTr[,5], family=binomial(link=Link), data=dataProgTr)))
    dataProg[[i]]$Ypred <- Ypred[[i]]
  }
  output <- dataProg
  return(output)
}
# Calculation the Ypred from 6 independent variables
CalcYpred6x <- function(dataProg, Link="logit"){
  Length <- length(dataProg)
  Ypred  <- new("list")
  for(i in 1:Length){
    dataProgTr <- dataProg[[i]][,-c((ncol(dataProg[[i]])-1),
      (ncol(dataProg[[i]])-2))]
    Ypred[[i]] <- as.vector(fitted(glm(dataProgTr[,ncol(dataProgTr)]~
      dataProgTr[,1]+dataProgTr[,2]+dataProgTr[,3]+dataProgTr[,4]+
      dataProgTr[,5]+dataProgTr[,6], family=binomial(link=Link),
      data=dataProgTr)))
```

```
    dataProg[[i]]$Ypred <- Ypred[[i]]
  }
  output <- dataProg
  return(output)
}
# Calculation the M(i)
CalcMI <- function(dataProg){
  Length <- length(dataProg)
  mRank  <- new("list")
  for(i in 1:Length){
    dataProg[[i]] <- dataProg[[i]][order(dataProg[[i]]$Ypred),]
    mRank[[i]]    <- rank(dataProg[[i]]$Ypred)
    mTemp1  <- cbind(as.vector(unlist(mRank[[i]])), order(order(unlist(mRank[[i]]))))
    rewRank <-
as.numeric(names(table(unlist(mRank[[i]]))[table(unlist(mRank[[i]]))!=1]))
    if(length(rewRank)!=0){
      mTemp2  <- mTemp1[!is.na(match(mTemp1[,1], rewRank)),]
      newRank <- aggregate(mTemp2[,2], list(mTemp2[,1]), FUN=max)
      newRank <- newRank[match(mTemp1[,1], newRank[,1]), 2]
      mRank[[i]][!is.na(newRank)] <- newRank[!is.na(newRank)]
      dataProg[[i]]$MI <- mRank[[i]]
    }
    else{
      dataProg[[i]]$MI <- mRank[[i]]
    }
  }
  output <- dataProg
  return(output)
}
# Calculation the a(i)
```

```
CalcAI <- function(dataProg){
  Length <- length(dataProg)
  AI    <- new("list")
  for(i in 1:Length){
    MI <- c(0, dataProg[[i]]$MI)
    Yn <- c(dataProg[[i]]$Ynew, NA)
    iTemp <- c(0:(length(MI)-1))
    Co <- rep(-1, length(MI))

    Case1st <- c(iTemp[-length(MI)]+1<=MI[-length(MI)], NA)
    Case2nd <- c(MI[-length(MI)]<iTemp[-length(MI)]+1, NA)
    Pos1st  <- rep(NA, length(MI))
    Pos2nd  <- rep(NA, length(MI))
    Pos1st[which(!is.na(Case2nd) & Case2nd==1)] <- MI[which(!is.na(Case2nd) &
Case2nd==1)]+1
    Pos2nd[which(!is.na(Case2nd) & Case2nd==1)+1] <- MI[which(!is.na(Case2nd) &
Case2nd==1)+1]
    dataTemp <- as.data.frame(cbind(Yn, MI, iTemp, Co, Case1st, Case2nd, Pos1st,
Pos2nd))

    AI[[i]] <- c(0, rep(NA, length(MI)))
    for(j in 2:(length(AI[[i]])-1)){
      if(Case2nd[j-1]==1){
        AI[[i]][j] <- AI[[i]][j-1]+sum(Co[Pos1st[j-1]:Pos2nd[j+1-1]]^Yn[Pos1st[j-
1]:Pos2nd[j+1-1]])
      }
      else{
        AI[[i]][j] <- AI[[i]][j-1]
      }
    }
```

```
    AI[[i]] <- AI[[i]][2:(length(AI[[i]])-1)]
    dataProg[[i]]$AI <- AI[[i]]
  }
  output <- dataProg
  return(output)
}
# Description          : Calculation the cut of point
cut.of.point <- function(dataProg){
  Length <- length(dataProg)
  lower  <- new("list")
  upper  <- new("list")
  for(i in 1:Length){
    ai <- dataProg[[i]]$AI
    y.pred <- dataProg[[i]]$Ypred
    lower[[i]] <- ifelse(which(ai==max(ai))[1]==length(ai),
      y.pred[which(ai==max(ai))[1]],
      ifelse(which(ai==max(ai))==0, 0,
      y.pred[which(ai==max(ai))[1]]))
    upper[[i]] <- ifelse(which(ai==max(ai))[1]==length(ai) | max(which(
      ai==max(ai)))==length(ai), 1,
      ifelse(which(ai==max(ai))[1]==0, y.pred[1],
      y.pred[max(which(ai==max(ai)))+1]))
  }
  output <- list(lower, upper)
  names(output) <- c("lower","upper")
  return(output)
}
cer.value <- function(dataProg){
  Length <- length(dataProg)
  CER  <- new("list")
```

```
  for(i in 1:Length){
    nDt <- length(dataProg[[i]]$AI)
    posLow <- which(dataProg[[i]]$AI==max(dataProg[[i]]$AI))[1]
    nX1 <- sum(dataProg[[i]]$Ynew[1:posLow]==0)
    nX2 <- sum(dataProg[[i]]$Ynew[posLow:nDt]==1)
    CER[[i]] <- (nX1+nX2)/nDt
  }
  output <- CER
  return(output)
}
# Create new dataset, unlist to matrix
unlistMatrix <- function(dataProg, Loop){
  Length <- length(dataProg)
  dataMt <- new("list")
  for(i in 1:Length){
    dataMt[[i]] <- matrix(unlist(dataProg[[i]]), Loop, length(dataProg[[i]]))
    colnames(dataMt[[i]]) <- c("LOWER", "UPPER")
  }
  output <- dataMt
  return(output)
}
# Get data from percentile rank
#ROUND(((500+1)*2.5)/100,0)
percentile <- function(dataProg, alpha=0.05){
  dataProg <- dataProg[order(dataProg)]
  posL <- round((((length(dataProg)+1)*((alpha/2)*100))/100),0)
  posL <- ifelse(posL<1, 1, posL)
  posU <- round((((length(dataProg)+1)*((1-(alpha/2))*100))/100),0)
  posU <- ifelse(posU>length(dataProg), length(dataProg), posU)
  LCI  <- dataProg[posL]
```

```
    UCI  <- dataProg[posU]
    output <- matrix(c(LCI, UCI),1,2)
    colnames(output) <- c("LOWER", "UPPER")
    return(output)
}
```

Multiple regression Model

```
#Get or set current directory
setwd("D:/Output")
getFiles <- list.files("D:/Output", full.names=TRUE, pattern=".csv")
posFiles <- which(as.vector(regexpr("Logit", getFiles))!=(-1))
getFiles <- getFiles[posFiles]
allData  <- lapply(getFiles, FUN=read.csv)
#Merge Datasets
allDataFrame <- allData[[1]]
for(i in 1:(length(allData)-1)){
        allDataFrame <- rbind(allDataFrame, allData[[i+1]])
        print(i)
}
allDataFrame <- allDataFrame[order(allDataFrame[,1]),]

MLogit <- lm(percent~n+p+m+a+(n*p)+(n*m)+(n*a)+(p*m)+(p*a)+(m*a)+
        (n*p*m)+(n*p*a)+(n*m*a)+(p*m*a)+(n*p*m*a), data=allDataFrame)
summary(MLogit)
```

ประวัติผู้เขียนวิทยานิพนธ์

นางสาวศิริวิตรา ศิริวิสุทธิรัตน์ เกิดเมื่อวันเสาร์ที่ 30 มีนาคม พ.ศ. 2528 สำเร็จการศึกษาระดับปริญญาวิทยาศาสตรบัณฑิต (วท.บ.) สาขาสถิติ ภาควิชาคณิตศาสตร์ คณะวิทยาศาสตร์ มหาวิทยาลัยศรีนครินทรวิโรฒ ในปีการศึกษา 2549 และเข้าศึกษาต่อในหลักสูตรสถิติศาสตรมหาบัณฑิต (สต.ม.) สาขาสถิติ ภาควิชาสถิติ คณะพาณิชยศาสตร์และการบัญชี จุฬาลงกรณ์มหาวิทยาลัย ในปี พ.ศ. 2551