รายการอ้างอิง

ภาษาไทย

มานพ วราภักดิ์. (2550). การประมาณเงินสำรองสำหรับค่าสินไหมทดแทนคงค้างด้วยวิธีบันไดลูกโซ่. *วารสารจุฬาลงกรณ์ธุรกิจปริทัศน์* 29, 111 (มกราคม-มีนาคม 2550).

สำนักงานคณะกรรมการกำกับและส่งเสริม การประกอบธุรกิจประกันภัย และ สำนักงานอัตราเบี้ยประกันวินาศภัย. (2551). *เอกสารทางวิชาการเรื่องการคำนวณเงินสำรองค่าสินไหมทดแทน (Loss Reserving).*

ภาษาอังกฤษ

Christofides, S. (1990). *Regression Models Based on Log-incremental Payments.* Claims Reserving Manual, 2, Institute of Actuaries, London.

Efron, B. (1979). *Bootstrapping methods : Another Look at the Jackknife.* The Annals of Statistics 7 : 1-26.

Efron, B., Tibshirani, R. (1994). *An Introduction to the Bootstrap.* Chapman & Hall/CRC, ISBN10: 0412042312, 31-199.

England, P. D., Verrall, R. J. (1999). *Analytic and bootstrap estimates of prediction errors in claims reserving.* Insurance : Mathematics and Economics, 25, 281-293.

England, P. D., Verrall, R. J. (2002). *Stochastic Claims Reserving in General Insurance.* presented to the Institute of Actuaries. Available from : http://www.emb.com/EMBDOTCOM/Netherlands/Nieuws%20-%20News/SCRinGI-EnglandVerrall.pdf

Mack, T. (1993). *Distribution-Free Calculation of the Standard Error of Chain Ladder Reserve Estimates.* Astin Bulletin, 23, : 2.

Mack, T. (2006:Fall). *Parameter Estimation of Bornhuetter/Ferguson.* Casual Actuarial Society Forum. Available from : http://www.casact.org/pubs/forum/06fforum/145.pdf.

Mack, T. (2008:Fall). *The Prediction Error of Bornhuetter-Ferguson.* Casualty Actuarial Society E-Forum, 222-240. Available from : http://www.casact.org/pubs/forum/08fforum/10Mack%20.pdf.

Pinheiro, P. J. R., Andrade e Silva, J. M., Centeno, Maria de L. (2001). *Bootstrap Methodology in Claims Reserving*. Astin Colloquium International Actuarial Association – Brussels, Belgium. Available from : http://www.actuaries.org/ASTIN/Colloquia/Washington/Pinheiro_Silva_Centeno.pdf.

Renshaw, A. E. (1989). *Chain-Ladder and Interactive Modelling* (Claims Reserving and GLIM). J. I. A., 116, 559-587.

Wright, T. S. (1990). *A Stochastic Method for Claims Reserving in General Insurance.* J.I.A., 117, 667-731.

Zehnwirth, B. (1989). *The Chain-Ladder Technique – a Stochastic Model.* Claims Reserving Manual, 2, Institute of Actuaries, London.

ภาคผนวก

ชุดคำสั่งในการใช้โปรแกรม R สำหรับการหาค่าความคลาดเคลื่อนพยากรณ์โดยใช้เทคนิคบูตสแตรป

1. หาค่าปัจจัยพัฒนาการ (development factor) ของตารางค่าสินไหมทดแทนรูปสามเหลี่ยม (triangle)

```
RationCalc <- function(triTable){
  NCol  <- ncol(triTable)
  Ratio <- new("list")
  for(i in 1:(NCol-1)){
   Ratio[[i]] <- sum(triTable[,1+i],na.rm=T)/sum(
            triTable[-(NCol+1-i),i],na.rm=TRUE)
  }
  Ratio <- unlist(Ratio)
  Ratio
}
```

2. หาค่าในตารางค่าสินไหมทดแทนรูปสามเหลี่ยม (triangle) ส่วนที่ $i = 2,...,n$ และ $j = n+2-i,...,n$

```
FullTriangle <- function(triTable){
  NCol  <- ncol(triTable)
  Ratio <- RationCalc(triTable)
  devTemp <- as.vector(rev(apply(triTable, 1, FUN=last)))[-length(
  as.vector(rev(apply(triTable, 1, FUN=last))))]
  devRecast <- as.vector(apply(triTable, 1, FUN=last))
  devI  <- devTemp*Ratio
  devII <- devI[-length(devI)]*Ratio[-1]
  devIII <- devII[-length(devII)] *Ratio[-c(1,2)]
  devIV <- devIII[-length(devIII)] *Ratio[-c(1,2,3)]
  dataUnder <- c(devI, devII, devIII, devIV)
  FullTable <- triTable
  posData <- 1
  NColTmp <- NCol-1
  for(i in 1:(NCol-1)){
    for(j in 1:NColTmp){
      FullTable[NCol-j+1, i+j] <- dataUnder[posData]
```

```
            posData <- posData+1
          }
          NColTmp <- NColTmp-1
        }
        FullTable
      }
```

3. หาค่าสินไหมทดแทนจ่ายจนถึงปัจจุบัน

```
last <- function(triTable){
  lastValue <- triTable[sum(!is.na(triTable))]
  lastValue
}
```

4. ค่าที่ปรับด้วยปัจจัยพัฒนาการ

```
RacastCalc <- function(triTable){
  Ratio <- RationCalc(triTable)
  NCol  <- ncol(triTable)
  Temp  <- matrix(0, NCol, NCol)
  RecastPaid <- triTable
  for(i in 1:(NCol-1)){
    nLength <- sum(!is.na(RecastPaid[,NCol-i]))
    for(j in 1:(nLength-1)){
      RecastPaid[j,NCol-i] <- RecastPaid[j,NCol-i+1]/Ratio[NCol-i]
    }
    RecastPaid <- Temp+RecastPaid
  }
  RecastPaid
}
```

5. หาค่าการเพิ่มขึ้นของค่าที่ปรับด้วยปัจจัยพัฒนาการ

```
RecastIncreCalc <- function(triTable){
  NCol  <- ncol(triTable)
  Temp1 <- triTable[,-1]
  Temp2 <- triTable[,-NCol]
```

```
    RecastTemp  <- Temp1-Temp2
    RecastIncre <- triTable
    RecastIncre[,2:NCol] <- RecastTemp
    RecastIncre
}
```

6. หาค่าบนตำแหน่งเปอร์เซ็นต์ไทล์

```
percentile <- function(x, p){
  x <- sort(x)
  x <- x[floor((p/100)*length(x))]
  x
}
```

7. การหาค่าความคลาดเคลื่อนพยากรณ์ของวิธีบันไดลูกโซ่ (Chain-Ladder Method) ด้วยวิธีบูตสแตรป

```
options(digits=20)
Incurred <- read.csv("the name of the file", row.names=1, header=TRUE)
Paid <- read.csv("the name of the file", row.names=1, header=TRUE)
NCol <- ncol(Paid)
EarnedPremium <- read.csv("the name of the file", row.names=1, header=TRUE)
cumrevEP <- rev(cumsum(EarnedPremium))
cumsumPaid <- t(apply(Paid, 1, FUN=cumsum))

#Step 1
Ratio <- RationCalc(cumsumPaid)
RecastCumsumPaid <- RacastCalc(cumsumPaid)
RecastRatio <- RationCalc(RecastCumsumPaid)
RecastCumsumPaidFull <- FullTriangle(RecastCumsumPaid)
Reserve <- RecastCumsumPaidFull[, NCol]-as.vector(apply(RecastCumsumPaid, 1,
FUN=last))
RecastIncre <- RecastIncreCalc(RecastCumsumPaid)
RecastIncreTemp1 <- RecastCumsumPaidFull[,-ncol(RecastCumsumPaidFull)]
RecastIncreTemp2 <- RecastCumsumPaidFull[,-1]
RecastIncreFull  <- RecastCumsumPaidFull
```

```
RecastIncreFull[,2:ncol(RecastCumsumPaidFull)] <- (RecastIncreTemp2-RecastIncreTemp1)
Residual <- (Paid-RecastIncre)/sqrt(RecastIncre)
SquareResidual <- Residual^2
sumSquare  <- sum(SquareResidual, na.rm=TRUE)
numberData <- sum(!is.na(SquareResidual))
paraEst    <- 2*NCol-1
biasAdjust <- sqrt(numberData/(numberData-paraEst))
scalePara  <- sumSquare/(numberData-paraEst)
ResidualAdj <- Residual*biasAdjust
orgDtSet <- ResidualAdj[!is.na(ResidualAdj)]
sampleSize <- length(orgDtSet)
numLoop <- 1000  #Specify number of loop
seedDOE <- 1  #Specify design of experiment
resBoot      <- new("list")
PseudoDt     <- new("list")
cPseudoDt    <- new("list")
ratioPseudo  <- new("list")
cPseudoDev   <- new("list")
cPseudoDtFull <- new("list")
ReservePseudo <- new("list")
IncrePseudo  <- new("list")
ReserveIncre <- new("list")
resBootH     <- new("list")
PseudoRea    <- new("list")
ReservePRea  <- new("list")
diffReserve  <- new("list")
ReservePE    <- new("list")
sseReserve   <- new("list")

#Step 2
for(i in 1:numLoop){

#Step 2.1
set.seed((seedDOE-1)*numLoop+i)
samDtTmp <- sample(orgDtSet, sampleSize, replace=TRUE)
```

```
resBoot[[i]] <- matrix(c(samDtTmp[1:9], NA, samDtTmp[10:12], rep(NA,2),
samDtTmp[13:14], rep(NA,3), samDtTmp[15], rep(NA,4)), NCol, NCol)
dimnames(resBoot[[i]]) <- list(dimnames(ResidualAdj)[[1]],
dimnames(ResidualAdj)[[2]])
PseudoDt[[i]]  <- resBoot[[i]]*sqrt(RecastIncre)+RecastIncre
cPseudoDt[[i]] <- t(apply(PseudoDt[[i]], 1, FUN=cumsum))
ratioPseudo[[i]] <- RationCalc(cPseudoDt[[i]])
cPseudoDev[[i]] <- as.vector(apply(cPseudoDt[[i]], 1, FUN=last))
cPseudoDtFull[[i]] <- FullTriangle(cPseudoDt[[i]])
ReservePseudo[[i]] <- cPseudoDtFull[[i]][, NCol]-cPseudoDev[[i]]
IncrePseudo[[i]] <- cPseudoDtFull[[i]]
posTarget <- which(!is.na(cPseudoDt[[i]]))
cPseudoF1 <- cPseudoDtFull[[i]][,-1]
cPseudoF2 <- cPseudoDtFull[[i]][,-NCol]
cPseudoTemp  <- cPseudoF1-cPseudoF2
IncrePseudo[[i]][,2:NCol] <- cPseudoTemp
IncrePseudo[[i]] <- as.vector(unlist(as.data.frame(IncrePseudo[[i]])))
IncrePseudo[[i]][posTarget] <- NA
IncrePseudo[[i]] <- matrix(IncrePseudo[[i]], NCol, NCol)
ReserveIncre[[i]] <- apply(IncrePseudo[[i]], 1, FUN=sum, na.rm=TRUE)
IncrePseudoTemp <- as.vector(IncrePseudo[[i]])[as.vector(!is.na(IncrePseudo[[i]]))]
RecastIncreTemp <- as.vector(RecastIncreFull)[as.vector(!is.na(IncrePseudo[[i]]))]
valueCheck     <- (IncrePseudoTemp-RecastIncreTemp)/sqrt(RecastIncreTemp)
set.seed((seedDOE-1)*numLoop+i)
samDtH <- new("list")
for(j in 1:length(valueCheck)){
   samDtH[[j]] <- sample(orgDtSet[orgDtSet>=valueCheck[j]], 1, replace=TRUE)
}

#Step 2.2
resBootH[[i]] <- IncrePseudo[[i]]
resBootH[[i]][!is.na(IncrePseudo[[i]])] <- unlist(samDtH)
PseudoRea[[i]]  <- resBootH[[i]]*sqrt(RecastIncreFull)+RecastIncreFull
posTarget <- which(is.na(cPseudoDt[[i]]))
posAdjust <- which(is.na(PseudoRea[[i]][posTarget]))
```

```
PseudoRea[[i]][posTarget][posAdjust] <- IncrePseudo[[i]][posTarget][posAdjust]
ReservePRea[[i]] <- apply(PseudoRea[[i]], 1, FUN=sum, na.rm=TRUE)

#Step 3
diffReserve[[i]] <- ReservePRea[[i]]-ReserveIncre[[i]]
ReservePE[[i]]   <- Reserve+diffReserve[[i]]
sseReserve[[i]]  <- (Reserve-ReservePE[[i]])^2
}
reservePE <- matrix(unlist(ReservePE), numLoop, NCol, byrow=T)
reservePE <- apply(reservePE, 2, 95, FUN=percentile)
MSEP      <- matrix(unlist(sseReserve), numLoop, NCol, byrow=T)
MSEP      <- apply(MSEP, 2, FUN=mean)
seR       <- sqrt(MSEP)
seRpercent <- (sqrt(MSEP)*100)/Reserve
```

7. การหาค่าความคลาดเคลื่อนพยากรณ์ของวิธีบอร์นฮูตเทอร์ เฟอร์กูซัน (Bornhuetter-Ferguson Method) ด้วยวิธีบูตสแตรป

```
options(digits=20)
Incurred <- read.csv(I "the name of the file", row.names=1, header=TRUE)
cumsumIncurred <- t(apply(Incurred, 1, FUN=cumsum))
Paid <- read.csv("the name of the file" ,row.names=1, header=TRUE)
LastDevPaid <- apply(Paid, 1, FUN=last)
NCol <- ncol(Paid)
EarnedPremium <- read.csv(" the name of the file" , row.names=1, header=TRUE)[,position]
cumrevEP <- rev(cumsum(EarnedPremium))
EstPriorUltClaim <- read.csv("08EstPriorUltClaim.csv", row.names=1, header=TRUE)[,position]
ErrorII <- read.csv("09ErrorII.csv", row.names=1, header=TRUE)[,position]
TailILR <- read.csv("10TailILR.csv", row.names=1, header=TRUE)[position,]
cumsumPaid <- t(apply(Paid, 1, FUN=cumsum))

#Step 1
Ratio   <- c(RationCalc(cumsumPaid), 1)
Pattern  <- 1/rev(cumprod(rev(Ratio)))
oPattern <- 1-Pattern
RecastCumsumPaid <- RacastCalc(cumsumPaid)
```

```
RecastIncre <- RecastIncreCalc(RecastCumsumPaid)
Residual <- (Paid-RecastIncre)/sqrt(RecastIncre)
SquareResidual <- Residual^2
sumSquare  <- sum(SquareResidual, na.rm=TRUE)
numberData <- sum(!is.na(SquareResidual))
paraEst    <- 2*NCol-1
biasAdjust <- sqrt(numberData/(numberData-paraEst))
scalePara  <- sumSquare/(numberData-paraEst)
ResidualAdj <- Residual*biasAdjust
ExpUnReport <- EstPriorUltClaim*rev(oPattern)
ReportTD    <- apply(cumsumIncurred, 1, FUN=last)
TotalULT    <- ExpUnReport+ReportTD
PaidTD      <- apply(RecastCumsumPaid, 1, FUN=last)
Reserve     <- TotalULT-PaidTD
TotalReserve <- sum(Reserve)
orgDtSet <- ResidualAdj[!is.na(ResidualAdj)]
orgDtSetTemp <- orgDtSet[order(orgDtSet)][-c(1,2)]
RecastIncreTemp <- RecastIncre[as.vector(!is.na(RecastIncre))]
valueCheck     <- ((RecastIncreTemp/1.5)-RecastIncreTemp)/sqrt(RecastIncreTemp)
valueCheckII   <- (0-RecastIncreTemp)/sqrt(RecastIncreTemp)
sampleSize <- length(orgDtSet)
numLoop <- 1000
seedDOE <- 1
resBoot   <- new("list")
PseudoDt  <- new("list")
cPseudoDt <- new("list")
RatioBoot <- new("list")
PattBoot  <- new("list")
oPattBoot <- new("list")
psDevYear <- new("list")
relatPEP  <- new("list")
lastDev   <- new("list")
relatAdj  <- new("list")
ErrorI    <- new("list")
sqrtError <- new("list")
```

```
EPremAdj  <- new("list")
ULR       <- new("list")
Ultimate  <- new("list")
oPattBootR   <- new("list")
ExpUnReportBoot <- new("list")
ReportTDBoot <- new("list")
TotalULTBoot <- new("list")
PaidTDBoot   <- new("list")
reserveBoot  <- new("list")
resBootH     <- new("list")
PseudoDtH    <- new("list")
cPseudoDtH   <- new("list")
RatioBootH   <- new("list")
PattBootH    <- new("list")
oPattBootH   <- new("list")
psDevYearH   <- new("list")
relatPEPH    <- new("list")
lastDevH     <- new("list")
relatAdjH    <- new("list")
ErrorIH      <- new("list")
sqrtErrorH   <- new("list")
EPremAdjH    <- new("list")
ULRH         <- new("list")
UltimateH    <- new("list")
oPattBootRH  <- new("list")
ExpUnReportBootH <- new("list")
ReportTDBootH <- new("list")
TotalULTBootH <- new("list")
PaidTDBootH   <- new("list")
reserveBootH  <- new("list")
diffReserve   <- new("list")
ReservePE     <- new("list")
sseReserve    <- new("list")
```

```
#Step 2
        for(i in 1:numLoop){

#Step 2.1
        set.seed((seedDOE-1)*numLoop+i)
        samDtTmp <- new("list")
        for(j in 1:length(valueCheck)){
      if(valueCheck[j]==max(orgDtSetTemp)){
        samDtTmp[[j]] <- max(orgDtSetTemp)
      }
      else if(length(orgDtSetTemp[orgDtSetTemp>valueCheck[j]])==1){
        samDtTmp[[j]] <- orgDtSetTemp[orgDtSetTemp>valueCheck[j]]
      }
      else{
      samDtTmp[[j]] <- sample(orgDtSetTemp[orgDtSetTemp>valueCheck[j]], 1, replace=TRUE)
      }
    }
        samDtTmp <- unlist(samDtTmp)
        resBoot[[i]] <- matrix(c(samDtTmp[1:9], NA, samDtTmp[10:12], rep(NA,2),
      samDtTmp[13:14], rep(NA,3), samDtTmp[15], rep(NA,4)), NCol, NCol)
      dimnames(resBoot[[i]]) <- list(dimnames(ResidualAdj)[[1]],
      dimnames(ResidualAdj)[[2]])
      PseudoDt[[i]]  <- resBoot[[i]]*sqrt(RecastIncre)+RecastIncre
      cPseudoDt[[i]] <- t(apply(PseudoDt[[i]], 1, FUN=cumsum))
      RatioBoot[[i]] <- c(RationCalc(cPseudoDt[[i]]), 1)    #Ratio from bootstrap, 1st
      PattBoot[[i]]  <- 1/rev(cumprod(rev(RatioBoot[[i]]))) #Pattern of bootstrap, 1st
      oPattBoot[[i]] <- 1-PattBoot[[i]]                     #1-Pattern of bootstrap, 1st
      psDevYear[[i]] <- apply(PseudoDt[[i]], 2, FUN=sum, na.rm=TRUE)
      relatPEP[[i]]  <- psDevYear[[i]]/cumrevEP
      lastDev[[i]]  <- apply(cPseudoDt[[i]], 1, FUN=last)
      relatAdj[[i]] <- rev(cumsum(relatPEP[[i]]))
      if(length(as.vector(which(lastDev[[i]]<0)))){
    lastDev[[i]][as.vector(which(lastDev[[i]]<0))] <- LastDevPaid[as.vector(
        which(lastDev[[i]]<0))]
  }
```

```
ErrorI[[i]]   <- as.vector(lastDev[[i]]/EarnedPremium/relatAdj[[i]])
sqrtError[[i]] <- sqrt(ErrorI[[i]]*ErrorII)
EPremAdj[[i]]  <- EarnedPremium*sqrtError[[i]]
ULR[[i]]       <- TailILR*sqrtError[[i]]
Ultimate[[i]]  <- EarnedPremium*ULR[[i]]
oPattBootR[[i]]  <- rev(oPattBoot[[i]])
ExpUnReportBoot[[i]] <- Ultimate[[i]]*oPattBootR[[i]]
ReportTDBoot[[i]]   <- apply(cumsumIncurred, 1, FUN=last)
TotalULTBoot[[i]]   <- ExpUnReportBoot[[i]]+ReportTDBoot[[i]]
PaidTDBoot[[i]]     <- apply(cPseudoDt[[i]], 1, FUN=last)
reserveBoot[[i]]    <- TotalULTBoot[[i]]-PaidTDBoot[[i]]

#Step 2.2
  set.seed((seedDOE-1)*numLoop+i)
  samDtH <- new("list")
  for(j in 1:length(valueCheck)){
    if(length(orgDtSet[orgDtSet<samDtTmp[j]])==1){
      samDtH[[j]] <- min(orgDtSet[order(orgDtSet)][-1])
    }
    else if(length(orgDtSet[orgDtSet<=samDtTmp[j] & orgDtSet>valueCheckII[j]])==1){
      samDtH[[j]] <- orgDtSet[orgDtSet<=samDtTmp[j]]
    }
    else{
      samDtH[[j]] <- sample(orgDtSet[orgDtSet<samDtTmp[j] &
                orgDtSet>valueCheckII[j]], 1, replace=TRUE)
    }
  }
  samDtH <- unlist(samDtH)
  resBootH[[i]] <- matrix(c(samDtH[1:9], NA, samDtH[10:12], rep(NA,2),
  samDtH[13:14], rep(NA,3), samDtH[15], rep(NA,4)), NCol, NCol)
  PseudoDtH[[i]]  <- resBootH[[i]]*sqrt(RecastIncre)+RecastIncre
  cPseudoDtH[[i]] <- t(apply(PseudoDtH[[i]], 1, FUN=cumsum))
  RatioBootH[[i]] <- c(RationCalc(cPseudoDtH[[i]]), 1)
  PattBootH[[i]]  <- 1/rev(cumprod(rev(RatioBootH[[i]])))
  oPattBootH[[i]] <- 1-PattBootH[[i]]
```

```
    psDevYearH[[i]] <- apply(PseudoDtH[[i]], 2, FUN=sum, na.rm=TRUE)
    relatPEPH[[i]]  <- psDevYearH[[i]]/cumrevEP
    lastDevH[[i]]   <- apply(cPseudoDtH[[i]], 1, FUN=last)
    relatAdjH[[i]]  <- rev(cumsum(relatPEPH[[i]]))
    if(length(as.vector(which(lastDevH[[i]]<0)))){
      lastDevH[[i]][as.vector(which(lastDevH[[i]]<0))] <- LastDevPaid[as.vector(
        which(lastDevH[[i]]<0))]
    }
    ErrorIH[[i]]    <- as.vector(lastDevH[[i]]/EarnedPremium/relatAdjH[[i]])
    sqrtErrorH[[i]] <- sqrt(ErrorIH[[i]]*ErrorII)
    EPremAdjH[[i]]  <- EarnedPremium*sqrtErrorH[[i]]
    ULRH[[i]]       <- TailILR*sqrtErrorH[[i]]
    UltimateH[[i]]  <- EarnedPremium*ULRH[[i]]
    oPattBootRH[[i]]  <- rev(oPattBootH[[i]])
    ExpUnReportBootH[[i]] <- UltimateH[[i]]*oPattBootRH[[i]]
    ReportTDBootH[[i]]    <- apply(cumsumIncurred, 1, FUN=last)
    TotalULTBootH[[i]]    <- ExpUnReportBootH[[i]]+ReportTDBootH[[i]]
    PaidTDBootH[[i]]      <- apply(cPseudoDtH[[i]], 1, FUN=last)
    reserveBootH[[i]]     <- TotalULTBootH[[i]]-PaidTDBootH[[i]]

#Step 3
    diffReserve[[i]] <- reserveBootH[[i]]-reserveBoot[[i]]
    ReservePE[[i]]   <- Reserve+diffReserve[[i]]
    sseReserve[[i]]  <- (Reserve-ReservePE[[i]])^2
    }
    reservePE <- matrix(unlist(ReservePE), numLoop, NCol, byrow=T)
    reservePE <- apply(reservePE, 2, 95, FUN=percentile)
    MSEP  <- matrix(unlist(sseReserve), numLoop, NCol, byrow=T)
    MSEP  <- apply(MSEP, 2, FUN=mean, na.rm=T)
    seR   <- sqrt(MSEP)
    seRpercent <- (sqrt(MSEP)*100)/Reserve
```

ประวัติผู้เขียนวิทยานิพนธ์

นายไพรวุฒิ อชินีทองคำ เกิดเมื่อวันที่ 4 มิถุนายน 2529 ที่จังหวัดกรุงเทพมหานคร สำเร็จการศึกษาวิทยาศาสตรบัณฑิต สาขาสถิติประยุกต์ ภาควิชาสถิติประยุกต์ คณะวิทยาศาสตร์ประยุกต์ มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ เมื่อปีการศึกษา 2550 และเข้าศึกษาต่อในหลักสูตร วิทยาศาสตรมหาบัณฑิต สาขาการประกันภัย ภาควิชาสถิติ คณะพาณิชยศาสตร์และการบัญชี จุฬาลงกรณ์มหาวิทยาลัย เมื่อปีการศึกษา 2551

การติดต่อ E-mail : nonbug56@hotmail.com