รายการอ้างอิง

ภาษาไทย

กิตติ ภักดีวัฒนะกุล และ จำลอง ครูอุตสาหะ. Visual Basic 6 **ฉบับโปรแกรมเมอร์**. พิมพ์ครั้งที่ 8. กรุงเทพมหานคร : ไทยเจริญการพิมพ์, 2543.

จะเด็จ สวรรค์ตรานนท์. **การเปรียบเทียบวิธีที่ใช้สำหรับการเลือกสมการถดถอยที่ดีที่สุด**. วิทยานิพนธ์ปริญญามหาบัณฑิต ภาควิชาสถิติ บัณฑิตวิทยาลัย จุฬาลงกรณ์มหาวิทยาลัย, 2530.

ทรงศิริ แต้สมบัติ. **การวิเคราะห์การถดถอย**. กรุงเทพมหานคร : สำนักพิมพ์มหาวิทยาลัยเกษตรศาสตร์, 2541.

ธีระพร วีระถาวร. **การอนุมานเชิงสถิติขั้นกลาง : โครงสร้างและความหมาย**. กรุงเทพมหานคร : สำนักพิมพ์จุฬาลงกรณ์มหาวิทยาลัย, 2536.

ธีระพร วีระถาวร. **ความน่าจะเป็นกับการประยุกต์**. พิมพ์ครั้งที่ 2. กรุงเทพมหานคร : วิทยพัฒน์, 2539.

ธีระพร วีระถาวร. **ตัวแบบเชิงเส้น ทฤษฎีและการประยุกต์**. กรุงเทพมหานคร : วิทยพัฒน์, 2541.

นพมาศ อัครจันทโชติ. **การเปรียบเทียบวิธีการสร้างตัวแบบในการวิเคราะห์ความถดถอยพหุนามกรณีที่มี 2 ตัวแปรอิสระซึ่งเกิดอันตรกิริยา**. วิทยานิพนธ์ปริญญามหาบัณฑิต ภาควิชาสถิติ บัณฑิตวิทยาลัย จุฬาลงกรณ์มหาวิทยาลัย, 2539.

นุชรินทร์ ทิพยวรรณากร. **การเปรียบเทียบค่าพยากรณ์ที่ได้จากตัวแบบที่คัดเลือกตัวแปรด้วยวิธีเบส์เซียน วิธีการกำจัดตัวแปรแบบถอยหลัง และวิธีการถดถอยแบบขั้นบันได ในการวิเคราะห์ความถดถอยพหุนามแบบลำดับขั้น**. วิทยานิพนธ์ปริญญามหาบัณฑิต ภาควิชาสถิติ บัณฑิตวิทยาลัย จุฬาลงกรณ์มหาวิทยาลัย, 2540.

พจนา แววสวัสดิ์. **การเปรียบเทียบเกณฑ์การคัดเลือกตัวแบบความถดถอยพหุนามแบบติดกลุ่ม**. วิทยานิพนธ์ปริญญามหาบัณฑิต ภาควิชาสถิติ บัณฑิตวิทยาลัย จุฬาลงกรณ์มหาวิทยาลัย, 2543.

ภาษาอังกฤษ

Carlin, B.P., and Chip, S. Bayesian model choice via markov chain monte carlo methods. **Journal of the Royal Statistical Sosiety, Ser.B** 57 (1995) : 473-484.

Chipman, H., Hamada, M., and Wu, C.F.J. A bayesian variable selection approach for analyzing designed experiments with complex aliasing. **Technical Report** Department of Statistics, University of Chicago, 1996.

Draper, N.R., and Smith, H. **Applied regression analysis.** New York : John Wiley & Sons, 1998.

Gelman, A., Carlin, J.B., and Rubin, D.B. **Bayesian data analysis.** London : Chapman & Hall, 1995.

George, E.I., and McCulloch, R.E. Variable selection via gibbs sampling. **Journal of American Statistical Association** 88 (September 1993) : 881-889.

Gilks, W.R., Richardson, S., and Spiegelhalter, D.J. **Markov Chain Monte Carlo in practice.** London : Chapman & Hall, 1996.

Hoeting, J.A., **Account for model uncertainly in linear regression** A dissertation submitted in partial fulfillment of the requirement for the degree of doctor of philosophy, University of Washington, 1994.

Hoeting, J.A., Madigan, D., Raftery, A.E., and Volinsky, C.T. Bayesian model averaging : A tutorial. **Statistical Science** 14, No.4 (1999) : 382-417.

Jeffreys, H. **Theory of probability.** London : Oxford University Press., 1961.

Jose, B. , and Smith A.F.M. **Bayesian theory.** New York : John Wiley & Sons, 1994.

Kass, R.E., and Raftery, A.E. Bayes factors. **Journal of American Statistical Association** 90 (June 1995) : 773-795.

Madigan, D., and Raftery, A.E. Model selection and accounting for model uncertainly in graphical models using occam 's window. **Journal of American Statistical Association** 89 (December 1994) : 1535-1546.

Raftery, A.E. Approximate bayes factor and accounting for model uncertianly in generalized linear models. **Biometrika** 83 (1996) : 251-266.

Raftery, A.E., Madigan, D., and Hoeting, J.A. Bayesian model averaging for linear regression models. **Journal of American Statistical Association** 92 (March 1997) : 179-191.

Schwarz, G. Estimating the dimension of a model. **The Annals of Statistics** 6, No.2 (1978) : 461-464.

Smith, A.F.M., and Roberts, G.O. Bayesian computation via gibbs sampler and related markov chain monte carlo methods. **Journal of the Royal Statistical Sosiety, Ser.B** 55 (1993) : 3-24.

Stone, M. Comments on model selection criteria of Akaike and Schwarz. **Journal of the Royal Statistical Sosiety, Ser.B** 41, No.2 (1979) : 276-278.

ภาคผนวก

การสร้างเลขสุ่มที่มีการแจกแจงแบบสม่ำเสมอ

ในการจำลองแบบจะต้องมีการกำหนดเหตุการณ์ต่าง ๆ ที่เกิดขึ้นในระบบให้ใกล้เคียงกับสถานการณ์จริงที่ต้องการศึกษาให้มากที่สุด เหตุการณ์เหล่านี้ถูกสร้างขึ้นมาโดยอาศัยค่าของตัวแปรสุ่มที่มีการแจกแจงแบบต่าง ๆ และเนื่องจากตัวแปรสุ่มที่มีการแจกแจงแบบอื่น ๆ นั้นต้องสร้างจากตัวแปรสุ่มที่มีการแจกแจงแบบสม่ำเสมอ ดังนั้นการสร้างเลขสุ่มที่มีการแจกแจงแบบสม่ำเสมอจึงมีความสำคัญมากในการจำลองแบบ โดยจะต้องเลือกวิธีการสร้างเลขสุ่มที่ทำให้ได้เลขสุ่มที่มีความถูกต้องและมีคุณสมบัติที่ดีทางสถิติ กล่าวคือเป็นเลขสุ่มที่มีการแจกแจงแบบสม่ำเสมอ และเป็นอิสระซึ่งกันและกัน และนอกจากนี้เลขสุ่มที่ดีควรมีคุณสมบัติดังนี้

- สามารถสร้างเลขสุ่มที่ซ้ำชุดเดิมได้
- เลขสุ่มที่สร้างออกมาจะต้องมีรอบ (period) ยาว
- สามารถสร้างเลขสุ่มด้วยอัตราความเร็วสูง
- ใช้หน่วยความจำในคอมพิวเตอร์น้อย

วิธีการหนึ่งที่นิยมใช้ในการสร้างเลขสุ่ม คือ วิธีการสร้างเลขสุ่มด้วยการใช้เศษจากการหารผลคูณ (Multiplicative Congruential Method) ซึ่งวิธีการนี้จะให้เลขสุ่มที่เรียกว่า เลขคล้ายสุ่ม (pseudo-random number) เนื่องจากเป็นเลขที่เกิดจากการดำเนินการทางคณิตศาสตร์และตรรกศาสตร์ของตัวเลขก่อนหน้า

วิธีการสร้างเลขสุ่มด้วยการใช้เศษจากการหารผลคูณจะทำการหาเลขสุ่มโดยทำการคำนวณจากสมการ

$$X_{i+1} = a \cdot X_i \ (modulo\ m) \qquad , i = 0, 1, 2, 3, \ldots.$$

เมื่อ X_i เป็นเลขคล้ายสุ่มตัวที่ i

X_{i+1} เป็นเลขคล้ายสุ่มตัวที่ i + 1

X_0 เป็นค่าเริ่มต้น (seed)

a เป็นตัวคูณคงที่ (constant multiplier)

m เป็นค่าคงที่

นั่นคือ เลขตัวที่ i + 1 เป็นเศษที่ได้จากการหาร $a \cdot X_i$ ด้วย m จากนั้นคำนวณเลขคล้ายสุ่ม U_{i+1} ที่มีค่าในช่วง (0 , 1) จากสมการ

$$U_{i+1} = \frac{X_{i+1}}{m} \qquad , i = 0, 1, 2, 3, \ldots.$$

เลขคล้ายสุ่มนี้จะมีรอบยาวที่สุดเท่ากับ $m - 1$ โดยจะมีรอบยาวเพียงใดหรือคุณสมบัติดีหรือไม่ขึ้นอยู่กับการเลือกค่า a , m และ X_0 ซึ่งลอว์ และเคลตัน (Law and Kelton ,1982) ได้แนะนำว่า a ควรเป็นจำนวนเต็มบวกที่น้อยกว่า m เมื่อ m เป็นเลขจำนวนเฉพาะ (prime number) ที่ใหญ่ที่สุดที่สามารถเก็บค่าได้ในเครื่องคอมพิวเตอร์ และ X_0 เป็นเลขจำนวนเต็มคี่หรือจำนวนเฉพาะที่น้อยกว่า m

รายละเอียดของโปรแกรมที่ใช้ในการวิจัย

ในการวิจัยครั้งนี้ได้ใช้โปรแกรมภาษาวิชวลเบสิก (Microsoft Visual Basic 6.0) สำหรับการสร้างตัวแบบด้วยวิธี BIC BVS และ SR ซึ่งโปรแกรมภาษาวิชวลเบสิกนี้เป็นโปรแกรมที่ใช้การพัฒนาโปรแกรมบน Windows ส่งผลให้การพัฒนาโปรแกรมทำได้ง่าย และสะดวกยิ่งขึ้น นอกจากนี้โปรแกรมภาษาวิชวลเบสิกยังมีการนำเทคโนโลยีทางด้าน Visualize มาประกอบในการออกแบบจอภาพ ส่งผลให้โปรแกรมที่สร้างขึ้นสามารถพัฒนาไปเป็นโปรแกรมสำเร็จรูปเพื่อการใช้งานภายหลังได้อย่างสะดวก ส่วนการสร้างตัวแบบด้วยวิธี BMA_{OCC} และ BMA_{MC3} นั้นใช้โปรแกรม S-plus 2000 เนื่องจากในงานวิจัยของราฟเทอร์รี่ เมดิแกน และโฮเอ็ททิง (Raftery Madigan and Hoeting ,1997) ซึ่งเป็นผู้นำเสนอวิธีการเฉลี่ยตัวแบบของเบส์ ได้มีการนำเสนออัลกอริทึมของวิธีการเฉลี่ยตัวแบบของเบส์โดยใช้โปรแกรม S-plus 2000 ดังนั้นเพื่อความสะดวกในการพัฒนาโปรแกรมผู้วิจัยจึงได้ใช้โปรแกรม S-plus 2000 ในการเขียนโปแกรมเพื่อสร้างตัวแบบด้วยวิธี BMA_{OCC} และ BMA_{MC3} ดังกล่าว

สำหรับรายละเอียดทั้งหมดของโปรแกรมที่ใช้ในการวิจัยมีดังนี้

ตารางแสดงลักษณะการทำงานของโปรแกรมที่ใช้ในการวิจัย สำหรับการสร้างตัวแบบด้วยวิธี BIC โดยใช้โปรแกรมภาษาวิชวลเบสิก

ลำดับที่	ชื่อโปรแกรม	การทำงานของโปรแกรม	ชื่อโปรแกรมย่อยหรือฟังก์ชันที่เรียกใช้
โปรแกรมหลัก	MAIN_BIC	- สร้างข้อมูลตัวแปรตาม - สร้างข้อมูลตัวแปรอิสระ - สร้างตัวแบบด้วยวิธี BIC - คำนวณค่า AMSE และค่าส่วนเบี่ยงเบนมาตรฐานของค่า AMSE จากการทำซ้ำ 500 รอบ	AssignX , AssignTempError , Normal , BIC
โปรแกรมย่อย			
1	BIC	ดำเนินการสร้างตัวแบบตามขั้นตอนของวิธี BIC	FindBestModel , FindSSE
2	FindBestModel	หาตัวแบบที่มีค่า BIC ต่ำสุดจากตัวแบบทุกรูปแบบที่เป็นไปได้	nCr , OperateX , FindSSE
3	OperateX	ทำการหารูปแบบของตัวแบบที่เป็นไปได้ทั้งหมด	-
4	FindSSE	คำนวณค่า SSE	Inverse
5	Inverse	หาเมทริกซ์ผกผัน	-
6	AssignX	อ่านข้อมูลตัวแปรอิสระจากแฟ้มข้อมูล	-
7	AssignTempError	อ่านข้อมูลตัวแปรสุ่มที่มีการแจกแจงแบบสม่ำเสมอจากแฟ้มข้อมูล	-
8	Normal	ทำการแปลงตัวแปรสุ่มที่มีการแจกแจงสม่ำเสมอให้มีการแจกแจงแบบปกติ	-
ฟังก์ชัน			
1	nCr	ทำการคำนวณค่า ${}^{n}C_{r}$	Factorial
2	Factorial	ทำการคำนวณค่าแฟคทอเรียล	-

โปรแกรมสำหรับสร้างตัวแบบด้วยวิธี BIC โดยใช้โปรแกรมภาษาวิชวลเบสิก

```
Private Sub MAIN_BIC()
        Dim Y(110) , X(1 To 110, 1 To 16) , E(110) , MSE(500) , AMSE , STD_AMSE , TempError(50000) , Sigma ,
            Z2 , TempMSE , SumMSE , SumSTD As Double
        Dim n , NumX , nLoop , KK As Integer
        Dim index As Long
        n = 100
        NumX = 5
        Sigma = 20
        nLoop = 500
        Call AssignX(X())
        Call AssignTempError(TempError(), n)
        index = 1
        KK = 0
        SumMSE = 0
        For lp% = 1 To nLoop
                For i% = 1 To n
                        Call Normal(0, Sigma, E(i%), TempError(), index, KK, Z2)
                Next i%
                For i% = 1 To n
                        Y(i%) = 1 + E(i%)
                        For j% = 1 To NumX
                                Y(i%) = Y(i%) + X(i%, j%)
                        Next j%
                Next i%
                Call BIC(Y(), X(), n, NumX, TempMSE)
                MSE(lp%) = TempMSE
                SumMSE = SumMSE + MSE(lp%)
        Next lp%
        AMSE = SumMSE / nLoop
        SumSTD = 0
        For lp% = 1 To nLoop
                SumSTD = SumSTD + (MSE(lp%) - AMSE) ^ 2
```

```
        Next Ip%
        STD_AMSE = (SumSTD / (nLoop - 1)) ^ 0.5
        textAMSE.Text = Format(AMSE, "#####.####")
        textSTD_AMSE.Text = Format(STD_AMSE, "#####.####")
End Sub
*****************************************************************************************************
Sub BIC (ByRef Y() As Double, ByRef X() As Double, ByVal n As Integer, ByVal NumX As Integer,
        ByRef MSE As Double)
        Dim Xstatus(15) , Xstatus1(15) , NumVariable  As Integer
        Dim Yhat(110) , Criterion , Criterion1 , Likelihood , SSE , SumY , MeanY As Double
        '-------------------------------------------------------------
        'Initail  Criterion for Null Model (No variable in Equation)
        '-------------------------------------------------------------
        For i% = 1 To NumX
                Xstatus(i%) = 0
        Next i%
        SumY = 0
        For i% = 1 To n
                SumY = SumY + Y(i%)
        Next i%
        MeanY = SumY / n
        For i% = 1 To n
                Yhat(i%) = MeanY
        Next i%
        SSE = 0
        For i% = 1 To n
                SSE = SSE + ((Y(i%) - Yhat(i%)) ^ 2)
        Next i%
        Likelihood = (2 * 22 / 7 * SSE / n * Exp(1)) ^ (-n / 2)
        Criterion = ((-2) * Log(Likelihood)) + Log(n)
        NumVariable = 0
        MSE = SSE / (n - 1)
        '-------------------------------------------------------
        'Find Best Model from Minimum Criterion (-2BIC)
        '-------------------------------------------------------
```

```
        For i% = 1 To NumX
                Call FindBestModel(Y(), X(), n, NumX, i%, Xstatus1(), Criterion1)
                If Criterion1 < Criterion Then
                        Criterion = Criterion1
                        For j% = 1 To NumX
                                Xstatus(j%) = Xstatus1(j%)
                        Next j%
                        NumVariable = i%
                End If
        Next i%
        If NumVariable > 0 Then
                Call FindSSE(Y(), X(), n, NumX, NumVariable, Xstatus(), SSE)
                MSE = SSE / (n - NumVariable - 1)
        End If
End Sub
**********************************************************************************************************************

Sub FindBestModel (ByRef Y() As Double, ByRef X() As Double, ByVal n As Integer, ByVal NumX As Integer, ByVal
                NumVariable As Integer, ByRef Xstatus1() As Integer, ByRef Criterion1 As Double)
        Dim ModelX(1 To 7000, 1 To 15) , TempXstatus(15) , NumModel  As Integer
        Dim TempSSE , Likelihood , Criterion  As Double
        NumModel = nCr(NumX, NumVariable)
        For a% = 1 To NumModel
                For b% = 1 To NumX
                        ModelX(a%, b%) = 0
                Next b%
        Next a%
        If (NumVariable = 1) Then
                For a% = 1 To NumModel
                        For b% = 1 To NumX
                                If (a% = b%) Then
                                        ModelX(a%, b%) = 1
                                End If
                        Next b%
                Next a%
        ElseIf (NumVariable = NumX) Then
```

```
            For a% = 1 To NumX
                ModelX(1, a%) = 1
            Next a%
        Else
            Call OperateX(NumX, NumVariable, NumModel, ModelX())
        End If
        For a% = 1 To NumX
            TempXstatus(a%) = ModelX(1, a%)
        Next a%
        Call FindSSE(Y(), X(), n, NumX, NumVariable, TempXstatus(), TempSSE)
        Likelihood = (2 * 22 / 7 * TempSSE / n * Exp(1)) ^ (-n / 2)
        Criterion = ((-2) * Log(Likelihood)) + ((NumVariable + 1) * Log(n))
        Criterion1 = Criterion
        For a% = 1 To NumX
            Xstatus1(a%) = TempXstatus(a%)
        Next a%
        If NumVariable < NumX Then
            For a% = 2 To NumModel
                For b% = 1 To NumX
                    TempXstatus(b%) = ModelX(a%, b%)
                Next b%
                Call FindSSE(Y(), X(), n, NumX, NumVariable, TempXstatus(), TempSSE)
                Likelihood = (2 * 22 / 7 * TempSSE / n * Exp(1)) ^ (-n / 2)
                Criterion = ((-2) * Log(Likelihood)) + ((NumVariable + 1) * Log(n))
                If Criterion < Criterion1 Then
                    Criterion1 = Criterion
                    For b% = 1 To NumX
                        Xstatus1(b%) = TempXstatus(b%)
                    Next b%
                End If
            Next a%
        End If
End Sub
```

****

```
Sub OperateX (ByVal NumX As Integer, ByVal NumVariable As Integer, ByVal NumModel As Integer, ByRef ModelX()
    As Integer)
    Dim iFileNum As Integer
    Dim index As String
    Dim PathFileName As String
    iFileNum = FreeFile
    PathFileName = Trim("c:\Thesis\x" + LTrim$(Str$(NumX)) + "_" + LTrim$(Str$(NumVariable)) + ".txt")
    Open PathFileName For Input As #iFileNum
    For i% = 1 To NumModel
        For j% = 1 To NumVariable
            Input #iFileNum, index
            ModelX(i%, Val(Mid(index, 2))) = 1
        Next j%
    Next i%
    Close #iFileNum
End Sub
*****************************************************************************************************************
Sub FindSSE (ByRef Y() As Double, ByRef X() As Double, ByVal n As Integer, ByVal NumX As Integer, ByVal
    NumVariable As Integer, ByRef Xstatus() As Integer, ByRef SSE As Double)
    Dim  Xuse(1 To 110, 1 To 16) , XTX(1 To 16, 1 To 16) , XTY(16) , YP(110) , beta(16)  As  Double
    Dim  PointerVariable As Integer
    For i% = 1 To n
        Xuse(i%, 1) = 1
    Next i%
    For i% = 1 To n
        PointerVariable = 2
        For j% = 2 To NumX + 1
            If Xstatus(j% - 1) = 1 Then
                Xuse(i%, PointerVariable) = X(i%, j% - 1)
                PointerVariable = PointerVariable + 1
            End If
        Next j%
    Next i%
    For i% = 1 To NumVariable + 1
        XTY(i%) = 0
```

```
                For j% = 1 To n
                        XTY(i%) = XTY(i%) + (Xuse(j%, i%) * Y(j%))
                Next j%
        Next i%
        For i% = 1 To NumVariable + 1
                For l% = 1 To NumVariable + 1
                        XTX(i%, l%) = 0
                        For j% = 1 To n
                                XTX(i%, l%) = XTX(i%, l%) + (Xuse(j%, i%) * Xuse(j%, l%))
                        Next j%
                Next l%
        Next i%
        Call Inverse(NumVariable + 1, XTX())
        For i% = 1 To NumVariable + 1
                beta(i%) = 0
                For j% = 1 To NumVariable + 1
                        beta(i%) = beta(i%) + (XTX(i%, j%) * XTY(j%))
                Next j%
        Next i%
        For i% = 1 To n
                YP(i%) = 0
                For j% = 1 To NumVariable + 1
                        YP(i%) = YP(i%) + (beta(j%) * Xuse(i%, j%))
                Next j%
        Next i%
        SSE = 0
        For i% = 1 To n
                SSE = SSE + ((Y(i%) - YP(i%)) ^ 2)
        Next i%
End Sub
**********************************************************************************************************************************
Sub Inverse (ByVal dimension As Integer, ByRef InvMatrix() As Double)
        Dim Temp(1 To 16, 1 To 16) As Double
        For i% = 1 To dimension
                For j% = 1 To dimension
```

```
                Temp(i%, j%) = InvMatrix(i%, j%)
            Next j%
        Next i%
        For I% = 1 To dimension
            Temp(I%, I%) = (-1) / Temp(I%, I%)
            For i% = 1 To dimension
                If (i% - I%) <> 0 Then
                    Temp(i%, I%) = (-Temp(i%, I%)) * Temp(I%, I%)
                End If
            Next i%
            For i% = 1 To dimension
                For j% = 1 To dimension
                    If ((i% - I%) * (j% - I%)) <> 0 Then
                        Temp(i%, j%) = Temp(i%, j%) - (Temp(i%, I%) * Temp(I%, j%))
                    End If
                Next j%
            Next i%
            For j% = 1 To dimension
                If (j% - I%) <> 0 Then
                    Temp(I%, j%) = (-Temp(I%, j%)) * Temp(I%, I%)
                End If
            Next j%
        Next I%
        For i% = 1 To dimension
            For j% = 1 To dimension
                InvMatrix(i%, j%) = -Temp(i%, j%)
            Next j%
        Next i%
End Sub
*****************************************************************************************************************************

Sub AssignX (ByRef X() As Double)
        Dim iFileNum , KK  As Integer
        Dim RdNum As String
        Dim RandomNumber(1500)  , Z2  As Double
        Dim index As Long
```

```
        iFileNum = FreeFile
        Open "c:\thesis\indepX.txt" For Input As #iFileNum
        For i% = 1 To 1500
                Line Input #iFileNum, RdNum
                X(i%) = Val(RdNum)
        Next i%
        Close #iFileNum
End Sub
```

หมายเหตุ

ข้อมูลตัวแปรอิสระสร้างจากฟังก์ชัน rmvnorm() ในโปรแกรม S-plus 2000 แล้วเก็บไว้ในแฟ้มข้อมูลชื่อ indepX.txt ดังนั้นในโปรแกรมย่อยนี้จึงเป็นการอ่านข้อมูลตัวแปรอิสระมาใช้

****

```
Sub AssignTempError(ByRef TempError() As Double, ByVal n As Integer)
        Dim iFileNum , nUse  As Long
        Dim RdNum As String
        nUse = CLng(n)
        iFileNum = FreeFile
        Open "c:\thesis\e500_1.txt" For Input As #iFileNum
        For i& = 1 To 500 * nUse
                Line Input #iFileNum, RdNum
                TempError(i&) = Val(RdNum)
        Next i&
        Close #iFileNum
End Sub
```

หมายเหตุ

ในโปรแกรมย่อยนี้จะอ่านข้อมูลตัวแปรสุ่มที่มีการแจกแจงแบบสม่ำเสมอจากแฟ้มข้อมูลชื่อ e500_1.txt เพื่อนำไปสร้างค่าความคลาดเคลื่อนสุ่มที่มีการแจกแจงแบบปกติต่อไป โดยรายละเอียดของโปรแกรมสำหรับสร้างข้อมูลตัวแปรสุ่มที่มีการแจกแจงแบบสม่ำเสมอคือ

```
Program Uniform;
        Var  DataFile    : Text;
             IX , IY , i   : Longint;
Begin
        assign( DataFile , 'c:\thesis\all_x.txt');
```

```
        rewrite(DataFile);
        IX := 357897;
        For i:= 1 to 1500 do
        begin
                IY:= IX * 16807;
                If IY < 0  then IY:= IY + 2147483647 ;
                WriteIn(DataFile,IY/2147483647) ;
                IX:= IY ;
        end;
        close(DataFile) ;
End.
```

****

```
Sub Normal (ByVal Mean As Double, ByVal Sigma As Double, ByRef X As Double, ByRef RandomNumber() As
    Double, ByRef index As Long, ByRef KK As Integer, ByRef Z2 As Double)
    Dim Z1, R1, R2, PI As Double
    PI = 3.1415926
    If KK <> 1 Then
        R1 = RandomNumber(index)
        R2 = RandomNumber(index + 1)
        Z1 = ((-2) * Log(R1)) ^ 0.5 * Cos(2 * PI * R2)
        Z2 = ((-2) * Log(R1)) ^ 0.5 * Sin(2 * PI * R2)
        X = Z1 * Sigma + Mean
        KK = 1
        index = index + 2
    Else
        X = Z2 * Sigma + Mean
        KK = 0
    End If
End Sub
```

****

```
Function nC r(ByVal n As Integer, ByVal r As Integer) As Integer
    Dim n_r  , StopFact , UseFact   As Integer
    Dim OnProduct As Long
    n_r = n - r
    If (n_r > r) Then
```

```
                StopFact = n_r
                UseFact = r
        Else
                StopFact = r
                UseFact = n_r
        End If
        OnProduct = 1
        Do While (n > StopFact)
                OnProduct = OnProduct * n
                n = n - 1
        Loop
        nCr = OnProduct / Factorial(UseFact)
End Function
**********
Function Factorial (ByVal n As Integer) As Integer
        Factorial = 1
        If (n > 1) Then
                Factorial = Factorial(n - 1) * n
        End If
End Function
**********
```

ตารางแสดงลักษณะการทำงานของโปรแกรมที่ใช้ในการวิจัย สำหรับการสร้างตัวแบบด้วยวิธี BVS โดยใช้โปรแกรมภาษาวิชวลเบสิก

ลำดับที่	ชื่อโปรแกรม	การทำงานของโปรแกรม	ชื่อโปรแกรมย่อยหรือฟังก์ชันที่เรียกใช้
โปรแกรมหลัก	MAIN_BVS	- สร้างข้อมูลตัวแปรตาม - สร้างข้อมูลตัวแปรอิสระ - สร้างตัวแบบด้วยวิธี BVS - คำนวณค่า AMSE และค่าส่วนเบี่ยงเบนมาตรฐานของค่า AMSE จากการทำซ้ำ 500 รอบ	AssignX , AssignTempError , Normal , BVS
โปรแกรมย่อย			
1	BVS	ดำเนินการสร้างตัวแบบตามขั้นตอนของวิธี BVS	Fbeta, CalXTX , CorrX , Finv, Normal, BMSQE , DRXD , CreateRD , Fnormal , IG , Aer2 , Finvgam
2	Fbeta	คำนวณค่าสัมประสิทธิ์ความถดถอย	CalXTX , Finv
3	CalXTX	คำนวณเมทริกซ์ $X'X$	-
4	Fivs	หาเมทริกซ์ผกผัน	-
5	MSQE , BMSQE	คำนวณค่า MSE	-
6	DRXD	คำนวณเมทริกซ์ DRD	-
7	CorrX	คำนวณเมทริกซ์สหสัมพันธ์	-
8	CreateRD	สร้างตัวแปรสุ่มที่มีการแจกแจงแบบสม่ำเสมอ	-
9	Normal	สร้างตัวแปรสุ่มที่มีการแจกแจงแบบปกติ	-
10	IG	สร้างตัวแปรสุ่มที่มีการแจกแจงแบบแกมมาผกผัน	-
11	AssignX	อ่านข้อมูลตัวแปรอิสระจากแฟ้มข้อมูล	-

ตารางแสดงลักษณะการทำงานของโปรแกรมที่ใช้ในการวิจัย สำหรับการสร้างตัวแบบด้วยวิธี BVS โดยใช้โปรแกรมภาษาวิชวลเบสิก (ต่อ)

ลำดับที่	ชื่อโปรแกรม	การทำงานของโปรแกรม	ชื่อโปรแกรมย่อยหรือฟังก์ชันที่เรียกใช้
12	AssignTempError	อ่านข้อมูลตัวแปรสุ่มที่มีการแจกแจงแบบสม่ำเสมอจากแฟ้มข้อมูล	-
ฟังก์ชัน			
1	Factorial	คำนวณค่าแฟคทอเรียล	-
2	Fnormal	คำนวณค่าฟังก์ชันของการแจกแจงแบบปกติ	-
3	Aer2	คำนวณค่าคลาดเคลื่อนกำลังสอง	-
4	Finvgam	คำนวรค่าฟังก์ชันของการแจกแจงแบบแกมมาผกผัน	-

โปรแกรมสำหรับสร้างตัวแบบด้วยวิธี BVS โดยใช้โปรแกรมภาษาวิชวลเบสิก

```
Private Sub MAIN_BVS()
        Dim X(1 To 110, 1 To 16) , Y(110) , E(110) , Alpha , CC , Ytemp(110) , Xtemp(1 To 110, 1 To 16) ,
            SumY , MeanY , SumX(16) , MeanX(16) , MSE(NumLoop) , AMSE , STD_AMSE , TempError(NumError) ,
            Sigma , Z2 , TempMSE , SumMSE , SumSTD    As Double
        Dim n, NumX , IP , nLoop , KK  As Integer
        Dim index As Long
        n = 25
        NumX = 5
        Alpha = 0.01
        Sigma = 10
        IP = 1
        CC = 5
        nLoop = NumLoop
        Call AssignX(X())
        Call AssignTempError(TempError(), n)
        index = 1
        KK = 0
        SumMSE = 0
        For Ip% = 1 To nLoop
                For i% = 1 To n
                        Call Normal(0, Sigma, E(i%), TempError(), index, KK, Z2)
                Next i%
                For i% = 1 To n
                        Y(i%) = 1 + E(i%)
                        For j% = 1 To NumX
                                Y(i%) = Y(i%) + X(i%, j%)
                        Next j%
                Next i%
                SumY = 0
                For i% = 1 To n
                        SumY = SumY + Y(i%)
                Next i%
```

```
                MeanY = SumY / n
                For i% = 1 To n
                        Ytemp(i%) = Y(i%) - MeanY
                Next i%
                Call BVS(Ytemp(), Xtemp(), n, NumX, Alpha, IP, CC, TempMSE)
                MSE(lp%) = TempMSE
                SumMSE = SumMSE + MSE(lp%)
        Next lp%
        AMSE = SumMSE / nLoop
        SumSTD = 0
        For ip% = 1 To nLoop
                SumSTD = SumSTD + (MSE(lp%) - AMSE) ^ 2
        Next lp%
        STD_AMSE = (SumSTD / (nLoop - 1)) ^ 0.5
        textAMSE.Text = Format(AMSE, "#####.####")
        textSTD_AMSE.Text = Format(STD_AMSE, "#####.####")
End Sub
*****************************************************************************************************************
Sub BVS (ByRef Y() As Double, ByRef X() As Double, ByVal n As Integer, ByVal NumX As Integer, ByVal Alpha As
        Double, ByVal IP As Integer, ByVal CC As Double, ByRef MSE As Double)
        Dim  Prior(16) , XTX(1 To 16, 1 To 16) , B1(16) , Tau(16) , HyVar(1 To 2, 1 To 16)  , Ppost(Iter) ,
            Diag( 1 To 16, 1 To 16)  ,  XTY(16) ,  RX( 1 To 16 , 1 To 16)  ,  DRD( 1 To 16,  1 To 16)  ,
            Bvar (1 To 16, 1 To 16)  ,  B3 (16)  ,  B2 (16)  ,  FB1(16)  ,  FB2(16)  , TB2(16)  ,
            TDRD( 1 To 16, 1 To 16 )  ,  TVar( 1 To 16, 1 To 16 )  ,  Post(16)  ,  Rpost(16)  ,
            RandomNumber(NumRD)  As  Double
        Dim  Flag(16) , Delta1(16)  , Delta2(16) , Delta3(16) , As Integer
        Dim  SSE , Sigma1 , temp , Due , IAlpha , IA1 , Fact , SSE1 , Lambda , Aerr2 , IBeta , Sigma2 , TBeta ,
             FS1 , FS2 , tmp , Ratio , Pprob , AMSE , Z2  As  Double
        Dim  KK As Integer
        Dim  index As Long
        'SET PRIOR OF DELTA  :  UNIFORM  PRIOR
        For i% = 1 To NumX
                Prior(i%) = 1 / 2
                Flag(i%) = 1
        Next i%
```

```
' Sigma[beta(i)] / Tau(i) = IP
Call Fbeta(n, NumX, X(), Y(), Flag(), XTX(), B1())
Call MSQE(n, NumX, X(), Y(), B1(), SSE, Sigma1)
For i% = 1 To NumX
        temp = Sigma1 / XTX(i%, i%)
        Tau(i%) = (temp ^ 0.5) / IP
Next i%
'CALCULATE VARIANCE OF HYPERPARAMETER BETA
For i% = 1 To NumX
        HyVar(1, i%) = 1 / Tau(i%)
        HyVar(2, i%) = 1 / (CC * Tau(i%))
Next i%
'CALCULATE VARIABLE WHICH CONSTANT IN ALL ITERATION
'SET Due=0 BECAUSE IT MAKE ESTIMATED PARAMETER NOT DIVERGE  Due = 1.99
'CALCULATE ALPHA FOR INVERSE GAMMA (DEPEND ON Due)
IAlpha = (n + Due) / 2
IA1 = IAlpha
Fact = Factorial(IA1)
'SETTING PARAMETER FOR INITIAL ITERATION TO BEGINING
'  Delta1 : DELTA VECTOR OF ZEROTH ITER
'  Delta2 : DELTA VECTOR OF FIRST ITER
'  B1     : BETA VECTOR OF ZEROTH ITER (LS ESTIMATE)
'  B2     : BETA VECTOR OF FIRST ITER (GIBBS SAMPLING ESTIMATE)
'  Sigma1 : SIGMA^2 OF ZEROTH ITER (LS ESTIMATE)
'  Sigma2 : SIGMA^2 OF FIRST ITER (GIBBS SAMPLING ESTIMATE)
'  Lambda = VARIANCE OF SATURATE MEDEL BY LS METHOD
For i% = 1 To NumX
        Delta1(i%) = 1
Next i%
Call BMSQE(n, NumX, X(), Y(), Delta1(), B1(), SSE1, Lambda)
Ppost(1) = 1
For i% = 1 To NumX
        Ppost(1) = Ppost(1) * Prior(i%)
Next i%
'SET DIAGONAL MATRIX OF BETA VARIANCE DEPEND ON DELTA1 VECTOR BY EQUATION 14
```

```
For i% = 1 To NumX
        For j% = 1 To NumX
                If i% = j% Then
                        Diag(i%, i%) = HyVar(2, i%)
                Else
                        Diag(i%, j%) = 0
                End If
        Next j%
Next i%
'CALCULATE VECTOR : Beta(LS,i)* XTX(j,i) = XTY(i)
For i% = 1 To NumX
        XTY(i%) = 0
        For j% = 1 To n
                XTY(i%) = XTY(i%) + (X(j%, i%) * Y(j%))
        Next j%
Next i%
' ----------BEGIN GIBBS----------
'ORDER : ORDER OF ITERATION FOR GIBBS SAMPLING GENERATE PARAMETER BY EQUATION 12
index = 1
KK = 0
Call CreateRD(RandomNumber())
For Order% = 2 To Iter
        'CACULATE CORRELATION MATRIX AND ITS INVERSE
        Call CorrX(n, NumX, X(), Flag(), RX())
        Call Finvs(NumX, RX(), Flag())
        'CALCULATE PRODUCT OF Diag*RX*Diag MATRIX
        Call DRXD(Diag(), RX(), NumX, Flag(), DRD())
        'CALCULATE VARIANCE MATRIX OF BETA VECTOR BY EQUATION 13(a)
        For i% = 1 To NumX
                If Flag(i%) = 1 Then
                        For j% = 1 To NumX
                                BVar(i%, j%) = 0
                                If Flag(j%) = 1 Then
                                        BVar(i%, j%) = (XTX(i%, j%) / Sigma1) + DRD(i%, j%)
                                End If
```

```
                Next j%
            End If
        Next i%
        Call Finvs(NumX, BVar(), Flag())
        'CALCULATE B3 : MEAN OF BETA ESTIMATE VECTOR
        For i% = 1 To NumX
            B3(i%) = 0
            If Flag(i%) = 1 Then
                For j% = 1 To NumX
                    If Flag(j%) = 1 Then
                        B3(i%) = B3(i%) + (BVar(i%, j%) * XTY(j%))
                    End If
                Next j%
                B3(i%) = B3(i%) / Sigma1
            End If
        Next i%
        'GENERATE BETA VECTOR OF FIRST ITERATION BY EQUATION 13
        For i% = 1 To NumX
            B2(i%) = 0
            If Flag(i%) = 1 Then
                Call Normal(B3(i%), BVar(i%, i%), B2(i%), RandomNumber(), index, KK, Z2)
                If Delta1(i%) = 1 Then
                    FB2(i%) = Fnormal(B2(i%), B3(i%), BVar(i%, i%))
                Else
                    FB1(i%) = Fnormal(B2(i%), B3(i%), BVar(i%, i%))
                End If
            End If
        Next i%
        'CALCULATE ABSOLUTE ERROR OF ESTIMATION WITH BETA
        Aerr2 = Aer2(Y(), X(), B2(), Delta1(), NumX, n)
        'CAL PARAMETER OF INVERSE GAMMA (IAlpha,IBeta)
        IBeta = (Aerr2 + (Due * Lambda)) / 2
        'GENERATE Sigma2 OF FIRST ITERATION BY EQUATION 14
        Call IG(IAlpha, IBeta, Sigma2, RandomNumber(), index)
        'INITIAL SOME PARAMETER BEFORE FIND DELTA VECTOR
```

```
For i% = 1 To NumX
        Delta3(i%) = Delta1(i%)
        TB2(i%) = B2(i%)
Next i%
'----------FINDING Delta2 VECTOR COMPONENTWISE BY SAMPLING CONSECUTIVELY---------
'L : ORDER OF Delta2(L) TOFIND NEW VALUE
For l% = 1 To NumX
        If Flag(l%) = 1 Then '*****
                Delta3(l%) = 1 - Delta1(l%)
                If Delta3(l%) = 1 Then
                        Diag(l%, l%) = HyVar(2, l%)
                Else
                        Diag(l%, l%) = HyVar(1, l%)
                End If
                Call DRXD(Diag(), RX(), NumX, Flag(), TDRD())
                For i% = 1 To NumX
                        If Flag(i%) = 1 Then
                                For j% = 1 To NumX
                                        TVar(i%, j%) = 0
                                        If Flag(j%) = 1 Then
                                            TVar(i%, j%) = ( XTX(i%, j%) / Sigma1) +
                                                    TDRD(i%, j%)
                                        End If
                                Next j%
                        End If
                Next i%
                Call Finvs(NumX, TVar(), Flag())
                TB2(l%) = 0
                For j% = 1 To NumX
                        If Flag(j%) = 1 Then
                                TB2(l%) = TB2(l%) + (TVar(l%, j%) * XTY(j%))
                        End If
                Next j%
                TB2(l%) = TB2(l%) / Sigma1
                'GENERATE BETA VECTOR OF SECOND ITERATION BY EQUATION 13
```

```
If Delta3(I%) = 1 Then
        FB2(I%) = Fnormal(B2(I%), TB2(I%), TVar(I%, I%))
Else
        FB1(I%) = Fnormal(B2(I%), TB2(I%), TVar(I%, I%))
End If
'CALCULATE ABSOLUTE ERROR OF ESTIMATION WITH BETA
Aerr2 = Aer2(Y(), X(), TB2(), Delta3(), NumX, n)
'CALCULATE PARAMETER OF INVERSE GAMMA (IAlpha,IBeta)
TBeta = (Aerr2 + (Due * Lambda)) / 2
'GENERATE Sigma2 OF SECOND ITERATION BY EQUATION 14
If Delta3(I%) = 1 Then
        FS1 = Finvgam(Sigma2, IAlpha, IBeta, Fact)
        FS2 = Finvgam(Sigma2, IAlpha, TBeta, Fact)
Else
        FS2 = Finvgam(Sigma2, IAlpha, IBeta, Fact)
        FS1 = Finvgam(Sigma2, IAlpha, TBeta, Fact)
End If
 tmp = 1
If Order% = 2 Then
        For i% = 1 To NumX
                If i% <> I% Then
                        If Delta3(i%) = 1 Then
                                tmp = tmp * FB2(i%)
                        Else
                                tmp = tmp * FB1(i%)
                        End If
                End If
        Next i%
Else
        For i% = 1 To NumX
                If i% <> I% Then
                        If Delta3(i%) = 1 Then
                                tmp = tmp * FB2(i%)
                        Else
                                tmp = tmp * FB1(i%)
```

```
                                        End If
                                    End If
                                Next i%
                            End If
                            Ratio = (FS1 * tmp * (1 - Prior(I%)) * FB1(I%)) + (FS2 * tmp * Prior(I%) *
                                FB2(I%))
                            Ratio = FS2 * tmp * Prior(I%) * FB2(I%) / Ratio
                            Rpost(I%) = Ratio
                            If Ratio >= (1 - Prior(I%)) Then
                                Delta2(I%) = 1
                            Else
                                Delta2(I%) = 0
                            End If
                            'ASSIGN POSTERIOR PROB TO Delta2 VECTOR
                            Delta3(I%) = Delta2(I%)
                            If Delta2(I%) = 1 Then
                                Post(I%) = Ratio
                                Diag(I%, I%) = HyVar(2, I%)
                            Else
                                Post(I%) = 1 - Ratio
                                Diag(I%, I%) = HyVar(1, I%)
                                Flag(I%) = 0
                            End If
                        End If '******
                    Next I%
                    Ppost(Order%) = 1
                    For i% = 1 To NumX
                        Ppost(Order%) = Post(i%) * Ppost(Order%)
                        B1(i%) = B2(i%)
                        Delta1(i%) = Delta2(i%)
                    Next i%
                    Pprob = Ppost(Order%)
                    Sigma1 = Sigma2
                    'CALCULATE XTX MATRIX FOR NEXT LOOP
                    Call CalXTX(X(), n, NumX, Flag(), XTX())
```

```
            For i% = 1 To NumX
                XTY(i%) = 0
                If Flag(i%) = 1 Then
                    For j% = 1 To n
                        XTY(i%) = XTY(i%) + (X(j%, i%) * Y(j%))
                    Next j%
                End If
            Next i%
            Call BMSQE(n, NumX, X(), Y(), Delta1(), B1(), SSE1, Lambda)
        Next Order%
        Call BMSQE(n, NumX, X(), Y(), Delta1(), B1(), SSE1, AMSE)
        MSE = AMSE
End Sub
*****************************************************************************************************************************
Sub Fbeta (ByVal n As Integer, ByVal NumX As Integer, ByRef X() As Double, ByRef Y() As Double, ByRef Flag() As
        Integer, ByRef XTX() As Double, ByRef B1() As Double)
        Dim XTY(16) , a(1 To 16, 1 To 16)  As  Double
        For i% = 1 To NumX
            If (Flag(i%) = 1) Then
                XTY(i%) = 0
                For j% = 1 To n
                    XTY(i%) = XTY(i%) + (X(j%, i%) * Y(j%))
                Next j%
            End If
        Next i%
        Call CalXTX(X(), n, NumX, Flag(), XTX())
        For i% = 1 To NumX
            For j% = 1 To NumX
                a(i%, j%) = XTX(i%, j%)
            Next j%
        Next i%
        Call Finvs(NumX, a(), Flag())
        For i% = 1 To NumX
            If Flag(i%) = 1 Then
                B1(i%) = 0
```

```
                For j% = 1 To NumX
                    If Flag(j%) = 1 Then
                        B1(i%) = B1(i%) + (a(i%, j%) * XTY(j%))
                    End If
                Next j%
            End If
        Next i%
End Sub
'*************************************************************************************************************
Sub CalXTX (ByRef X() As Double, ByVal n As Integer, ByVal NumX As Integer, ByRef Flag() As Integer, ByRef XTX()
        As Double)
    For i% = 1 To NumX
        For j% = 1 To NumX
            XTX(i%, j%) = 0
            If (Flag(i%) = 1) And (Flag(j%) = 1) Then
                For l% = 1 To n
                    XTX(i%, j%) = XTX(i%, j%) + (X(l%, i%) * X(l%, j%))
                Next l%
            End If
        Next j%
    Next i%
End Sub
'*************************************************************************************************************
Sub Finvs (ByVal NumX As Integer, ByRef a() As Double, ByRef Flag() As Integer)
    Dim b(1 To 16, 1 To 16) As Double
    For i% = 1 To NumX
        For j% = 1 To NumX
            b(i%, j%) = a(i%, j%)
        Next j%
    Next i%
    For l% = 1 To NumX
        If Flag(l%) = 1 Then
            b(l%, l%) = (-1) / b(l%, l%)
            For i% = 1 To NumX
                If (Flag(i%) = 1) And ((i% - l%) <> 0) Then
```

```
                    b(i%, l%) = (-b(i%, l%)) * b(l%, l%)
                End If
            Next i%
            For i% = 1 To NumX
                If Flag(i%) = 1 Then
                    For j% = 1 To NumX
                        If (Flag(j%) = 1) And (((i% - l%) * (j% - l%)) <> 0) Then
                            b(i%, j%) = b(i%, j%) - (b(i%, l%) * b(l%, j%))
                        End If
                    Next j%
                End If
            Next i%
            For j% = 1 To NumX
                If (Flag(j%) = 1) And ((j% - l%) <> 0) Then
                    b(l%, j%) = (-b(l%, j%)) * b(l%, l%)
                End If
            Next j%
        End If
    Next l%
    For i% = 1 To NumX
        For j% = 1 To NumX
            If (Flag(i%) = 1) And (Flag(j%) = 1) Then
                a(i%, j%) = -b(i%, j%)
            End If
        Next j%
    Next i%
End Sub
*****************************************************************************************************************************

Sub MSQE (ByVal n As Integer, ByVal NumX As Integer, ByRef X() As Double, ByRef Y() As Double, ByRef B1() As
    Double, ByRef SSE As Double, ByRef MSE As Double)
    Dim YP(110) As Double
    For i% = 1 To n
        YP(i%) = 0
        For j% = 1 To NumX
```

```
                YP(i%) = YP(i%) + (B1(j%) * X(i%, j%))
            Next j%
        Next i%
        SSE = 0
        For i% = 1 To n
            SSE = SSE + ((Y(i%) - YP(i%)) ^ 2)
        Next i%
        MSE = SSE / (n - NumX)
End Sub
*****************************************************************************************************************
Sub BMSQE (ByVal n As Integer, ByVal NumX As Integer, ByRef X() As Double, ByRef Y() As Double, ByRef Delta1()
        As Integer, ByRef B1() As Double, ByRef SSE1 As Double, ByRef MSE1 As Double)
        Dim YP(110) As Double
        Dim IK As Integer
            IK = 0
        For i% = 1 To NumX
            If Delta1(i%) <> 0 Then
                IK = IK + 1
            End If
        Next i%
        For i% = 1 To n
            YP(i%) = 0
            For j% = 1 To NumX
                YP(i%) = YP(i%) + (B1(j%) * X(i%, j%) * Delta1(j%))
            Next j%
        Next i%
        SSE1 = 0
        For i% = 1 To n
            SSE1 = SSE1 + ((Y(i%) - YP(i%)) ^ 2)
        Next i%
        MSE1 = SSE1 / (n - IK)
End Sub
*****************************************************************************************************************
Sub DRXD (ByRef Diag() As Double, ByRef RX() As Double, ByVal NumX As Integer, ByRef Flag() As Integer, ByRef
        DRD() As Double)
```

```
        Dim R1(1 To 16, 1 To 16) As Double
        For i% = 1 To NumX
                For j% = 1 To NumX
                        R1(i%, j%) = 0
                        If (Flag(i%) = 1) And (Flag(j%) = 1) Then
                                For l% = 1 To NumX
                                        If Flag(l%) = 1 Then
                                                R1(i%, j%) = R1(i%, j%) + (Diag(i%, l%) * RX(l%, j%))
                                        End If
                                Next l%
                        End If
                Next j%
        Next i%
        For i% = 1 To NumX
                For j% = 1 To NumX
                        DRD(i%, j%) = 0
                        If (Flag(i%) = 1) And (Flag(j%) = 1) Then
                                For l% = 1 To NumX
                                        If Flag(l%) = 1 Then
                                            DRD(i%, j%) = DRD(i%, j%) + (R1(i%, l%) * Diag(l%, j%))
                                        End If
                                Next l%
                        End If
                Next j%
        Next i%
End Sub
*****************************************************************************************************************************
Sub CorrX (ByVal n As Integer, ByVal NumX As Integer, ByRef X() As Double, ByRef Flag() As Integer, ByRef RX()
        As Double)

        Dim sum, Rsum, SQ, SSQ, SQR, SSQR As Double
        Dim Xmean(16) As Double
        For i% = 1 To NumX
                Xmean(i%) = 0
                If Flag(i%) = 1 Then
```

```
                    sum = 0
                    For j% = 1 To n
                        sum = sum + X(j%, i%)
                    Next j%
                    Xmean(i%) = sum / n
                End If
            Next i%
            For i% = 1 To NumX
                For j% = 1 To NumX
                    RX(i%, j%) = 0
                    If (Flag(i%) = 1) And (Flag(j%) = 1) Then
                        Rsum = 0
                        SQ = 0
                        SSQ = 0
                        For k% = 1 To n
                            Rsum = Rsum + (X(k%, i%) - Xmean(i%)) * (X(k%, j%) - Xmean(j%))
                            SQ = SQ + (X(k%, i%) - Xmean(i%)) ^ 2
                            SSQ = SSQ + (X(k%, j%) - Xmean(j%)) ^ 2
                        Next k%
                        SSQR = SSQ ^ 0.5
                        SQR = SQ ^ 0.5
                        RX(i%, j%) = Rsum / (SSQR * SQR)
                    End If
                Next j%
            Next i%
End Sub
*****************************************************************************************************************************************
Sub CreateRD (ByRef RandomNumber() As Double)
            Dim iFileNum As Integer
            Dim RdNum As String
            iFileNum = FreeFile
            Open "c:\Thesis\Random.txt" For Input As #iFileNum
            For i% = 1 To 20000
                Line Input #iFileNum, RdNum
                RandomNumber(i%) = Val(RdNum)
```

```
        Next i%
        Close #iFileNum
End Sub
```

หมายเหตุ

ในโปรแกรมย่อยนี้จะอ่านข้อมูลตัวแปรสุ่มที่มีการแจกแจงแบบสม่ำเสมอจากแฟ้มข้อมูลชื่อ Random.txt เพื่อนำไปสร้างตัวแปรสุ่มที่มีการแจกแจงแบบปกติหรือการแจกแจงแบบแกมมาผกผันต่อไป โดยรายละเอียดของโปรแกรมสำหรับสร้างข้อมูลตัวแปรสุ่มที่มีการแจกแจงแบบสม่ำเสมอคือ

```
Program Uniform;
        Var  DataFile     : Text;
             IX , IY , i    : Longint;
Begin
        assign( DataFile , 'c:\thesis\all_x.txt');
        rewrite(DataFile);
        IX := 357897;
        For i:= 1 to 1500 do
        begin
                IY:= IX * 16807;
                If IY < 0  then IY:= IY + 2147483647 ;
                Writeln(DataFile,IY/2147483647) ;
                IX:= IY ;
        end;
        close(DataFile) ;
End.
```

```
Sub Normal (ByVal Mean As Double, ByVal Sigma As Double, ByRef X As Double, ByRef RandomNumber() As
    Double, ByRef index As Long, ByRef KK As Integer, ByRef Z2 As Double)
    Dim Z1, R1, R2, PI As Double
    PI = 3.1415926
    If KK <> 1 Then
        R1 = RandomNumber(index)
        R2 = RandomNumber(index + 1)
        Z1 = ((-2) * Log(R1)) ^ 0.5 * Cos(2 * PI * R2)
        Z2 = ((-2) * Log(R1)) ^ 0.5 * Sin(2 * PI * R2)
        X = Z1 * Sigma + Mean
```

```
                KK = 1
                index = index + 2
        Else
                X = Z2 * Sigma + Mean
                KK = 0
        End If
End Sub
```

```
Sub IG (ByVal IAlpha As Double, ByVal IBeta As Double, ByRef X As Double, ByRef RandomNumber() As Double,
        ByRef index As Long)
    Dim E, a, b, Q, Zeta, d, R1, R2, V, Y, Z, W, T, Gamma As Double
    E = 2.7182818
    a = 1 / ((2 * IAlpha - 1) ^ 0.5)
    b = IAlpha - Log(4)
    Q = IAlpha + (1 / a)
    Zeta = 4.5
    d = 1 + Log(Zeta)
    Do
            R1 = RandomNumber(index)
            index = index + 1
            R2 = RandomNumber(index)
            index = index + 1
            V = a * Log(R1 / (1 - R1))
            Y = IAlpha * (E ^ V)
            Z = (R1 ^ 2) * R2
            W = b + (Q * V) - Y
            T = W + d - (Zeta * Z)
            If (T >= 0) Or (W >= Log(Z)) Then
                    Gamma = Y
                    Exit Do
            End If
    Loop  While (T < 0) And (W < Log(Z))
    X = IBeta / Gamma
End Sub
```

หมายเหตุ : สำหรับโปรแกรมย่อย AssignX และโปรแกรมย่อย AssignTempError นั้นมีรายละเอียดเช่นเดียวกับในส่วนของโปรแกรมสำหรับสร้างตัวแบบด้วยวิธี BIC ดังที่กล่าวไว้ในตอนต้น จึงไม่ขอกล่าวถึงอีก

```
**********************************************************************************************************************

Function Factorial (ByVal X As Double) As Double
        X = X - 1
        Factorial = ((2 * 22 / 7 * X) ^ 0.5) * (X ^ X / Exp(X))
End Function
**********************************************************************************************************************

Function Fnormal (ByVal X As Double, ByVal Mean As Double, ByVal var As Double) As Double
        Fnormal = 0
        Fnormal = Exp(-((X - Mean) ^ 2) / (2 * var)) / SQR(2 * 22 / 7 * var)
End Function
**********************************************************************************************************************

Function Aer2 (ByRef Y() As Double, ByRef X() As Double, ByRef Beta() As Double, ByRef Delta() As Integer, ByVal
            NumX As Integer, ByVal n As Integer) As Double
        Dim YP As Double
        Aer2 = 0
        For i% = 1 To n
                YP = 0
                For j% = 1 To NumX
                        YP = YP + (X(i%, j%) * Beta(j%) * Delta(j%))
                Next j%
                Aer2 = Aer2 + (Y(i%) - YP) ^ 2
        Next i%
End Function
**********************************************************************************************************************

Function Finvgam (ByVal X As Double, ByVal GAlpha As Double, ByVal GBeta As Double, ByVal Fact As Double)
            As  Double
        Finvgam = 0
        Finvgam = (GBeta ^ GAlpha) * (Exp((-GBeta) / X)) / (Fact * (X ^ (GAlpha + 1)))
End Function
**********************************************************************************************************************
```

ตารางแสดงลักษณะการทำงานของโปรแกรมที่ใช้ในการวิจัย สำหรับการสร้างตัวแบบด้วยวิธี SR โดยใช้โปรแกรมภาษาวิชวลเบสิก

ลำดับที่	ชื่อโปรแกรม	การทำงานของโปรแกรม	ชื่อโปรแกรมย่อยหรือฟังก์ชันที่เรียกใช้
โปรแกรมหลัก	MAIN_SR	- สร้างข้อมูลตัวแปรตาม - สร้างข้อมูลตัวแปรอิสระ - สร้างตัวแบบด้วยวิธี SR - คำนวณค่า AMSE และค่าส่วนเบี่ยงเบนมาตรฐานของค่า AMSE จากการทำซ้ำ 500 รอบ	AssignX , AssignTempError , Normal , SR
โปรแกรมย่อย			
1	Stepwise	ดำเนินการสร้างตัวแบบตามขั้นตอนของวิธี Stepwise	Correlation , Inverse , CoefReg , Ftable
2	Correlation	คำนวณเมทริกซ์สหสัมพันธ์	-
3	Inverse	หาเมทริกซ์ผกผัน	-
4	CoefReg	คำนวณค่าสัมประสิทธิ์ความถดถอย	Inverse
5	Ftable	เก็บค่าสถิติเอฟจากตารางเอฟ	-
6	AssignX	อ่านข้อมูลตัวแปรอิสระจากแฟ้มข้อมูล	-
7	Normal	ทำการแปลงตัวแปรสุ่มที่มีการแจกแจงสม่ำเสมอให้มีการแจกแจงแบบปกติ	-
8	AssignTempError	อ่านข้อมูลตัวแปรสุ่มที่มีการแจกแจงแบบสม่ำเสมอจากแฟ้มข้อมูล	-
ฟังก์ชัน	Tstatistic	คำนวณค่าสถิติที	-

โปรแกรมสำหรับสร้างตัวแบบด้วยวิธี SR โดยใช้โปรแกรมภาษาวิชวลเบสิก

```
Private Sub MAIN_SR()
    Dim X(1 To nMAX, 1 To paraMAX) , Y(nMAX) , E(nMAX) , Alpha , MSE(NumLoop) , AMSE , STD_AMSE ,
        TempError(NumError) , Sigma , Z2 , TempMSE , SumMSE , SumSTD As Double
    Dim n(4) , NumX(6) , nLoop , KK As Integer
    Dim index As Long
    n(1) = 25
    n(2) = 50
    n(3) = 75
    n(4) = 100
    NumX(1) = 3
    NumX(2) = 5
    NumX(3) = 7
    NumX(4) = 10
    NumX(5) = 12
    NumX(6) = 15
    Alpha = 0.01
    Sigma = 5
    Call AssignX(X(), 100, 15)
    For indexNumX% = 1 To 6
        For indexn% = 1 To 4
            nLoop = NumLoop
            Call AssignTempError(TempError(), n(indexn%))
            index = 1
            KK = 0
            SumMSE = 0
            For lp% = 1 To nLoop
                For i% = 1 To n(indexn%)
                    Call Normal(0, Sigma, E(i%), TempError(), index, KK, Z2)
                Next i%
                For i% = 1 To n(indexn%)
                    Y(i%) = 1 + E(i%)
                    For j% = 1 To NumX(indexNumX%)
```

```
                                        Y(i%) = Y(i%) + X(i%, j%)
                                    Next j%
                                Next i%
                                Call Stepwise(Y(), X(), n(indexn%), NumX(indexNumX%), Alpha, TempMSE)
                                MSE(lp%) = TempMSE
                                SumMSE = SumMSE + MSE(lp%)
                            Next lp%
                            AMSE = SumMSE / nLoop
                            SumSTD = 0
                            For lp% = 1 To nLoop
                                SumSTD = SumSTD + (MSE(lp%) - AMSE) ^ 2
                            Next lp%
                            STD_AMSE = (SumSTD / (nLoop - 1)) ^ 0.5
                            'textAMSE.Text = Format(AMSE, "#####.####")
                            'textSTD_AMSE.Text = Format(STD_AMSE, "#####.####")
                            Printer.Print "*****************************"
                            Printer.Print "n    =", n(indexn%)
                            Printer.Print "NumX =", NumX(indexNumX%)
                            Printer.Print "---------------------------"
                            Printer.Print "AMSE     =", Format(AMSE, "#####.####")
                            Printer.Print "STD_AMSE  =", Format(STD_AMSE, "#####.####")
                            Printer.Print "*****************************"
                    Next indexn%
            Next indexNumX%
End Sub
*****************************************************************************************************************************
Sub Stepwise ( ByRef Y() As Double , ByRef X() As Double , ByVal n As Integer , ByVal NumX As Intege , ByVal Alpha
        As Double, ByRef MSE As Double)
    Dim  Fvalue(120) , Xtemp(1 To 110, 1 To 16) , R(1 To 16, 1 To 16) , Yhat(110) , PartialRyx(15) , Beta(16) ,
        XTY(16) , A(1 To 16, 1 To 16) , Xtemp1(1 To 110, 1 To 16) , PartialF(15) , OrderF(15) , YTY , Rmax ,
        Ftest , SumY , MeanY , PartialRmax , SSR , PartialFmin   As  Double
    Dim  Xstatus(15) As Boolean
    Dim  OrderX(15) , OrderRyx(15) , InVar , OutVar , NumVar , NewVar , MoveVar    As  Integer
    Call Ftable(Alpha, Fvalue())
    YTY = 0
```

```
For i% = 1 To n
        YTY = YTY + (Y(i%) * Y(i%))
Next i%
For i% = 1 To n
        For j% = 1 To NumX
                Xtemp(i%, j%) = X(i%, j%)
        Next j%
Next i%
For i% = 1 To NumX
        Xstatus(i%) = False
Next i%
InVar = 0
OutVar = 0
NumVar = 0
For i% = 1 To NumX
        OrderX(i%) = i%
Next i%
Call Correlation(n, NumX, Y(), Xtemp(), R())
Rmax = 0
For i% = 2 To NumX + 1
        If Abs(R(1, i%)) > Rmax Then
                Rmax = Abs(R(1, i%))
                InVar = i% - 1
        End If
Next i%
NumVar = 1
Ftest = Tstatistic(n, NumVar, Rmax ^ 2) ^ 2
If Ftest < Fvalue(n - NumVar - 1)  Then
        SumY = 0
        For i% = 1 To n
                SumY = SumY + Y(i%)
        Next i%
        MeanY = SumY / n
        For i% = 1 To n
                Yhat(i%) = MeanY
```

```
            Next i%
            MSE = 0

            For i% = 1 To n
                  MSE = MSE + ((Y(i%) - Yhat(i%)) ^ 2)
            Next i%
            MSE = MSE / (n - 1)
            'Debug.Print "*****************************************"
            'Debug.Print "No Variable in Regression Equation"
            'Debug.Print "*****************************************"
            'Debug.Print "Regression Equation  :  Yhat = ", MeanY
            'Debug.Print "*****************************************"
            Exit Sub
End If
Xstatus(InVar) = True
OrderX(NumVar) = InVar
'Debug.Print "---------------------------------------"
'Debug.Print "Input Variable : ", InVar
'Debug.Print "---------------------------------------"
For i% = 1 To n
            Xtemp(i%, 1) = X(i%, InVar)
Next i%
'-----------------------------------------
Do While NumVar < NumX
            For i% = 1 To NumX
                  If Xstatus(i%) = True  Then
                        OrderRyx(i%) = 0
                        PartialRyx(i%) = 0
                  End If
                  If Xstatus(i%) = False Then
                        For j% = 1 To n
                              Xtemp(j%, NumVar + 1) = X(j%, i%)
                        Next j%
                        Call Correlation(n, NumVar + 1, Y(), Xtemp(), R())
                        Call Inverse(NumVar + 2, R())
```

```
                OrderRyx(i%) = i%
                PartialRyx(i%) = ((R(1, NumVar + 2) ^ 2) / (R(1, 1) * R(NumVar + 2 ,
                                NumVar  + 2))) ^ 0.5
            End If
        Next i%
        PartialRmax = 0
        NewVar = 0
        For i% = 1 To NumX
            If (OrderRyx(i%) <> 0) And (PartialRyx(i%) > PartialRmax) Then
                PartialRmax = PartialRyx(i%)
                NewVar = i%
            End If
        Next i%
        Ftest = Tstatistic(n, NumVar + 1, PartialRmax ^ 2) ^ 2
        If Ftest < Fvalue(n - NumVar - 2) Then
            InVar = 0
            If OutVar = 0 Then
                Exit Do
            End If
            If NumVar = 0 Then
                Exit Do
            End If
        End If
        If Ftest >= Fvalue(n - NumVar - 2) Then
            If NewVar = OutVar Then
                Exit Do
            End If
            If NewVar <> OutVar Then
                For i% = 1 To n
                    Xtemp(i%, NumVar + 1) = X(i%, NewVar)
                Next i%
                NumVar = NumVar + 1
                InVar = NewVar
                Xstatus(NewVar) = True
                OrderX(NumVar) = NewVar
```

```
                    Debug.Print "----------------------------------------"
                    'Debug.Print "Input Variable : ", InVar
                    'Debug.Print "----------------------------------------"
            End If
End If
Call CoefReg(n, NumVar, Y(), Xtemp(), Beta(), XTY())
SSR = 0
For i% = 1 To NumVar + 1
            SSR = SSR + (Beta(i%) * XTY(i%))
Next i%
MSE = (YTY - SSR) / (n - NumVar - 1)
For i% = 1 To n
            Xtemp1(i%, 1) = 1
Next i%
For i% = 1 To n
            For j% = 2 To NumVar + 1
                        Xtemp1(i%, j%) = Xtemp(i%, j% - 1)
            Next j%
Next i%
For i% = 1 To NumVar + 1
            For l% = 1 To NumVar + 1
                        A(i%, l%) = 0
                        For j% = 1 To n
                                    A(i%, l%) = A(i%, l%) + (Xtemp1(j%, i%) * Xtemp1(j%, l%))
                        Next j%
            Next l%
Next i%
Call Inverse(NumVar + 1, A())
For i% = 1 To NumVar
            PartialF(i%) = (Beta(i% + 1) / (MSE * A(i% + 1, i% + 1)) ^ 0.5) ^ 2
Next i%
PartialFmin = PartialF(1)
MoveVar = OrderX(1)
For i% = 2 To NumVar
            If PartialF(i%) < PartialFmin Then
```

```
                    PartialFmin = PartialF(i%)
                    MoveVar = OrderX(i%)
                End If
            Next i%
            If PartialFmin >= Fvalue(n - NumVar - 1) Then
                If InVar = 0 Then
                    Exit Do
                End If
            End If
            If PartialFmin < Fvalue(n - NumVar - 1) Then
                OutVar = MoveVar
                NumVar = NumVar - 1
                Xstatus(MoveVar) = False
                For i% = 1 To NumVar + 1
                    If OrderX(i%) = MoveVar Then
                        OrderX(i%) = 0
                    End If
                Next i%
                For i% = 1 To NumVar + 1
                    If (OrderX(i%) = 0) And (i% <> NumVar + 1) Then
                        OrderX(i%) = OrderX(i% + 1)
                        For j% = 1 To n
                            Xtemp(j%, i%) = Xtemp(j%, i% + 1)
                        Next j%
                        OrderX(i% + 1) = 0
                    End If
                Next i%
                'Debug.Print "--------------------------------"
                'Debug.Print "Output Variable : ", OutVar
                'Debug.Print "--------------------------------"
            End If
            If OutVar = InVar Then
                Exit Do
            End If
Loop
```

```
'-------------------------------------------------

Call CoefReg(n, NumVar, Y(), Xtemp(), Beta(), XTY())

For i% = 1 To n

        Yhat(i%) = 0

        For j% = 1 To NumVar

                Yhat(i%) = Yhat(i%) + (Beta(j% + 1) * X(i%, OrderX(j%)))

        Next j%

Next i%

For i% = 1 To n

        Yhat(i%) = Yhat(i%) + Beta(1)

Next i%

MSE = 0

For i% = 1 To n

        MSE = MSE + ((Y(i%) - Yhat(i%)) ^ 2)

Next i%

MSE = MSE / (n - NumVar - 1)

End Sub

'*************************************************************************************************************

Sub Correlation (ByVal n As Integer, ByVal NumX As Integer, ByRef Y() As Double, ByRef Xtemp() As Double,
        ByRef R() As Double)

Dim  sum, Rsum, SQ, SSQ, SQR, SSQR , Xmean(16) , Xuse(1 To 110, 1 To 16)  As Double

Dim NumX1 As Integer

NumX1 = NumX + 1

For i% = 1 To n

        Xuse(i%, 1) = Y(i%)

Next i%

For i% = 1 To n

        For j% = 2 To NumX1

                Xuse(i%, j%) = Xtemp(i%, j% - 1)

        Next j%

Next i%

For i% = 1 To NumX1

        sum = 0

        For j% = 1 To n

                sum = sum + Xuse(j%, i%)
```

```
            Next j%
            Xmean(i%) = sum / n
        Next i%
        For i% = 1 To NumX1
            For j% = 1 To NumX1
                Rsum = 0
                SQ = 0
                SSQ = 0
                For k% = 1 To n
                    Rsum = Rsum + (Xuse(k%, i%) - Xmean(i%)) * (Xuse(k%, j%) - Xmean(j%))
                    SQ = SQ + (Xuse(k%, i%) - Xmean(i%)) ^ 2
                    SSQ = SSQ + (Xuse(k%, j%) - Xmean(j%)) ^ 2
                Next k%
                SSQR = SSQ ^ 0.5
                SQR = SQ ^ 0.5
                R(i%, j%) = Rsum / (SSQR * SQR)
            Next j%
        Next i%
End Sub
*****************************************************************************************************************************************
Sub Inverse (ByVal dimension As Integer, ByRef InvMatrix() As Double)
        Dim Temp(1 To 16, 1 To 16) As Double
        For i% = 1 To dimension
            For j% = 1 To dimension
                Temp(i%, j%) = InvMatrix(i%, j%)
            Next j%
        Next i%
        For l% = 1 To dimension
            Temp(l%, l%) = (-1) / Temp(l%, l%)
            For i% = 1 To dimension
                If (i% - l%) <> 0 Then
                    Temp(i%, l%) = (-Temp(i%, l%)) * Temp(l%, l%)
                End If
            Next i%
            For i% = 1 To dimension
```

```
                    For j% = 1 To dimension
                        If ((i% - l%) * (j% - l%)) <> 0 Then
                            Temp(i%, j%) = Temp(i%, j%) - (Temp(i%, l%) * Temp(l%, j%))
                        End If
                    Next j%
                Next i%
                For j% = 1 To dimension
                    If (j% - l%) <> 0 Then
                        Temp(l%, j%) = (-Temp(l%, j%)) * Temp(l%, l%)
                    End If
                Next j%
            Next l%
            For i% = 1 To dimension
                For j% = 1 To dimension
                    InvMatrix(i%, j%) = -Temp(i%, j%)
                Next j%
            Next i%
End Sub
*****************************************************************************************************************************************
Sub CoefReg (ByVal n As Integer, ByVal NumX As Integer, ByRef Y() As Double, ByRef Xtemp() As Double,
        ByRef Beta() As Double, ByRef XTY() As Double)
    Dim  Xuse(1 To 110, 1 To 16)  , XTX(1 To 16, 1 To 16)  As Double
    Dim  NumX1  As Integer
    NumX1 = NumX + 1
    For i% = 1 To n
        Xuse(i%, 1) = 1
    Next i%
    For i% = 1 To n
        For j% = 2 To NumX1
            Xuse(i%, j%) = Xtemp(i%, j% - 1)
        Next j%
    Next i%
    For i% = 1 To NumX1
        XTY(i%) = 0
        For j% = 1 To n
```

```
                    XTY(i%) = XTY(i%) + (Xuse(j%, i%) * Y(j%))
                Next j%
            Next i%
            For i% = 1 To NumX1
                For l% = 1 To NumX1
                    XTX(i%, l%) = 0
                    For j% = 1 To n
                        XTX(i%, l%) = XTX(i%, l%) + (Xuse(j%, i%) * Xuse(j%, l%))
                    Next j%
                Next l%
            Next i%
            Call Inverse(NumX1, XTX())
            For i% = 1 To NumX1
                Beta(i%) = 0
                For j% = 1 To NumX1
                    Beta(i%) = Beta(i%) + (XTX(i%, j%) * XTY(j%))
                Next j%
            Next i%
End Sub
*****************************************************************************************************************************
Sub Ftable(ByVal Alpha As Double, ByRef Fvalue() As Double)
            If Alpha = 0.01 Then
                Fvalue(1) = 4052
                Fvalue(2) = 98.5
                Fvalue(3) = 34.12
                Fvalue(4) = 21.2
                Fvalue(5) = 16.26
                Fvalue(6) = 13.75
                Fvalue(7) = 12.25
                Fvalue(8) = 11.26
                Fvalue(9) = 10.56
                Fvalue(10) = 10.04
                Fvalue(11) = 9.65
                Fvalue(12) = 9.33
                Fvalue(13) = 9.07
```

```
        Fvalue(14) = 8.86
        Fvalue(15) = 8.68
        Fvalue(16) = 8.53
        Fvalue(17) = 8.4
        Fvalue(18) = 8.29
        Fvalue(19) = 8.18
        Fvalue(20) = 8.1
        Fvalue(21) = 8.02
        Fvalue(22) = 7.95
        Fvalue(23) = 7.88
        Fvalue(24) = 7.82
        Fvalue(25) = 7.77
        Fvalue(26) = 7.72
        Fvalue(27) = 7.68
        Fvalue(28) = 7.64
        Fvalue(29) = 7.6
        Fvalue(30) = 7.56
        For i% = 31 To 39
                Fvalue(i%) = 7.56 - ((i% - 30) * 0.25 / 10)
        Next i%
        Fvalue(40) = 7.31
        For i% = 41 To 59
                Fvalue(i%) = 7.31 - ((i% - 40) * 0.23 / 20)
        Next i%
        Fvalue(60) = 7.08
        For i% = 61 To 119
                Fvalue(i%) = 7.08 - ((i% - 60) * 0.23 / 60)
        Next i%
        Fvalue(120) = 6.85
End If
If Alpha = 0.05 Then
        Fvalue(1) = 161.4
        Fvalue(2) = 18.51
        Fvalue(3) = 10.13
        Fvalue(4) = 7.71
```

```
Fvalue(5) = 6.61
Fvalue(6) = 5.99
Fvalue(7) = 5.59
Fvalue(8) = 5.32
Fvalue(9) = 5.12
Fvalue(10) = 4.96
Fvalue(11) = 4.84
Fvalue(12) = 4.75
Fvalue(13) = 4.67
Fvalue(14) = 4.6
Fvalue(15) = 4.54
Fvalue(16) = 4.49
Fvalue(17) = 4.45
Fvalue(18) = 4.41
Fvalue(19) = 4.38
Fvalue(20) = 4.35
Fvalue(21) = 4.32
Fvalue(22) = 4.3
Fvalue(23) = 4.28
Fvalue(24) = 4.26
Fvalue(25) = 4.24
Fvalue(26) = 4.23
Fvalue(27) = 4.21
Fvalue(28) = 4.2
Fvalue(29) = 4.18
Fvalue(30) = 4.17
For i% = 31 To 39
        Fvalue(i%) = 4.17 - ((i% - 30) * 0.09 / 10)
Next i%
Fvalue(40) = 4.08
For i% = 41 To 59
        Fvalue(i%) = 4.08 - ((i% - 40) * 0.08 / 20)
Next i%
Fvalue(60) = 4
For i% = 61 To 119
```

```
                Fvalue(i%) = 4 - ((i% - 60) * 0.08 / 60)
            Next i%
            Fvalue(120) = 3.92
        End If
End Sub
```

****

หมายเหตุ : สำหรับโปรแกรมย่อย AssignX และโปรแกรมย่อย AssignTempError นั้นมีรายละเอียดเช่นเดียวกับในส่วนของโปรแกรมสำหรับสร้างตัวแบบด้วยวิธี BIC ดังที่กล่าวไว้ในตอนต้น จึงไม่ขอกล่าวถึงอีก

****

```
Sub Normal (ByVal Mean As Double, ByVal Sigma As Double, ByRef X As Double, ByRef RandomNumber() As
    Double, ByRef index As Long, ByRef KK As Integer, ByRef Z2 As Double)
    Dim Z1, R1, R2, PI As Double
    PI = 3.1415926
    If KK <> 1 Then
        R1 = RandomNumber(index)
        R2 = RandomNumber(index + 1)
        Z1 = ((-2) * Log(R1)) ^ 0.5 * Cos(2 * PI * R2)
        Z2 = ((-2) * Log(R1)) ^ 0.5 * Sin(2 * PI * R2)
        X = Z1 * Sigma + Mean
        KK = 1
        index = index + 2
    Else
        X = Z2 * Sigma + Mean
        KK = 0
    End If
End Sub
```

****

```
Function Tstatistic (ByVal n As Integer, ByVal NumVar As Integer, ByVal RR As Double) As Double
    Tstatistic = (RR ^ 0.5) * (((n - NumVar - 1) / (1 - RR)) ^ 0.5)
End Function
```

****

การทำงานของโปรแกรมที่ใช้ในการวิจัย สำหรับการสร้างตัวแบบด้วยวิธี BMA_{OCC} โดยใช้โปรแกรม S-plus 2000

โปรแกรมที่ใช้ในการสร้างตัวแบบด้วยวิธี BMA_{OCC} นั้นสร้างขึ้นโดยใช้โปรแกรม S-plus 2000 เขียนเป็นฟังก์ชันชื่อ BMA.OCC(....) เป็นฟังก์ชันหลักฟังก์ชันเดียวที่ใช้ในการหาปริภูมิตัวแบบและหาค่าความน่าจะเป็นภายหลัง รูปแบบการทำงานจึงต่างกับการเขียนโปรแกรมด้วยภาษาวิชวลเบสิก และเนื่องจากโปรแกรม S-plus 2000 มีฟังก์ชันสำเร็จรูปทางคณิตศาสตร์และสถิติให้เรียกใช้ได้ การทำงานของโปรแกรมจึงมีความสะดวกรวดเร็ว ไม่จำเป็นต้องมีโปรแกรมย่อยหรือฟังก์ชันที่สร้างขึ้นเองสำหรับเรียกใช้ โดยรายละเอียดของโปรแกรมมีดังนี้

โปรแกรมสำหรับสร้างตัวแบบด้วยวิธี BMA_{OCC} โดยใช้โปรแกรม S-plus 2000

```
BMA.OCC(x, y, wt = rep(1, length(y)), strict = F, OR = 20)
{
    ### Inputs:
    ### x          matrix of independent variables
    ### y          dependent variable
    ### wt         weights for regression
    ### strict     = T will return a more parsimonious set of models (Occam's Window).
    ###              Any modelwith a more likely submodel will be eliminated.
    ###            = F (default).  Returns all models within 1/OR in posteriormodel prob.
    ### OR         maximum ratio for excluding models in Occam's window  Default is 20.

    ### Outputs:
    ### postprob   posterior probabilities of the models selected
    ### label      labels identifying the models selected
    ### r2         R2 values for the models
    ### bic        values of bic for the models
    ### size       the number of independent variables in each of the models
    ### which      a logical matrix with one row per model and one column per variable indicating
                   whether that variable is in the model
    ### probne0    posterior probability that each variable is non-zero (in %)
    ### postmean   posterior mean of each coefficient (from model mixing)
    ### postsd     posterior standard deviation of each coefficient (from model mixing)
    ### mle        matrix(like the matrix which) giving the MLE estimate of each coefficient for each model
    ### se         matrix (like mle) giving the corresponding standard errors for each model

##############################################################################################
```

```
### Set original names of x-variables, if there are none to start with.

 x <- data.frame(x)
 if(is.null(dimnames(x))) dimnames(x) <- list(NULL,paste("X",1:ncol(x),sep=""))
 y <- as.numeric(y)
 options(contrasts=c("contr.treatment","contr.treatment"))


### Removing NA's


 x2 <- na.omit(data.frame(x))
 used <- match(row.names(data.frame(x)), row.names(x2))
 omitted <- seq(nrow(x))[is.na(used)]
 if(length(omitted) > 0)
 {
          wt <<- wt [ - omitted]
          x <<- x2
          y <<- y[ - omitted]
          warning(paste("There were ",length(omitted),"records deleted due to NA's"))
 }

### Creating design matrix

 cdf <-cbind.data.frame(y=y,x)
 mm <- model.matrix(formula(cdf),data=cdf)[,-1,drop=F]
 x <- mm


### If there are more than 30 columns in x, reduce to 30 using backwards elimination.

 if(ncol(x) > 30)
 {
  back <- stepwise(x, y, wt, method = "backward")
  which.back30 <- back$which[(ncol(x) - 30), ]
  x <- x[, which.back30]
 }


### Execute leaps and bounds algorithm
 nvar <- length(x[1, ])


if (nvar >2)
 {
  a <- leaps(x, y, wt = wt, method = "r2", names = dimnames(x)[[2]])
  a$r2 <- pmin(pmax(0, a$r2), 99.9)       # Correct for incorrect R2 for leaps
  x.lm <- cbind.data.frame(y = y, as.data.frame(x[,a$which[2, ,drop=F]]), w =wt)
  lm.fix <- lm(y ~ . - w, weights = w, data = x.lm)
  r2.fix <- summary(lm.fix)$r.sq
  N <- ncol(x)
```

```
  magic <- N * log(1 - a$r2[2]/100) - N * log(1 - r2.fix)
  a$r2 <- 1 - (1 - a$r2/100) * exp( - magic/N)          # Include the null model
  r2 <- round(c(0, a$r2) * 100, 3)
  size <- c(1, a$size)
  label <- c("NULL", a$label)
  which <- rbind(rep(F, ncol(x)), a$which)
 }
 else
  {
   r2_bic_NULL
   nmod <- switch(ncol(x),2,4)
   bic <- label <- rep(0, nmod)
   model.fits <- as.list(rep(0, nmod))
   which <- matrix(c(F, T, F, T, F, F, T, T), nmod, nmod/2)
   size <- c(1 ,2,2,3)[1:nmod]
   sep <- if(all(nchar(dimnames(x)[[2]]) == 1)) "" else ","
   for(k in 1:nmod) {
          if (k==1) {
           label[k] <- "NULL"
           lm1 <- lm(y~1,w=wt)
          }
           else {
            label[k] <- paste(dimnames(x)[[2]][which[k, ]], collapse = sep)
            x.lm <- cbind.data.frame(y = y, x=x[,which[k, ,drop=F]], wt = wt)
            lm1 <- lm(y~.-wt,data=x.lm,w=wt)
           }
          r2[k] <- summary(lm1)$r.sq*100
   }
  }

### Calculate BIC and apply the first OW rule.
 n <- length(y)
 bic <- n * log(1 - r2/100) + (size - 1) * log(n)
 occam <- bic - min(bic) < 2 * log(OR)
 r2 <- r2[occam]
 size <- size[occam]
 label <-label[occam]
 which <- which[occam, ,drop=F]
 bic <- bic[occam]
 postprob <- exp(-0.5 * bic)/sum(exp(-0.5 * bic))

### Order models in descending order of posterior probability
 order.bic <- order(bic, size, label)
```

```
 r2 <- r2[order.bic]
 size <- size[order.bic]
 label <- label[order.bic]
 which <- which[order.bic,  , drop=F]
 bic <- bic[order.bic]
 postprob <- postprob[order.bic]

### Apply the second rule to eliminate models with better ones nested withinthem.
 if(strict) {
  nmod <- length(bic)
  if (nmod>1)
   {
          occam <- rep(T, nmod)
          for(k in (2:nmod)) {
           for(j in (1:(k - 1))) {
            which.diff <- which[k,  ] - which[j,  ]
            if(all(which.diff >= 0))
             occam[k] <- F
           }
          }
          r2 <- r2[occam]
          size <- size[occam]
          label <- label[occam]
          nmod <- sum(occam)
          which <- which[occam,  ,drop=F]
          bic <- bic[occam]
          postprob <- postprob[occam]
          postprob <- postprob/sum(postprob)
   }
 }


### Calculate Pr[beta_i != 0 | D] for each coefficient beta_i
 probne0 <- round(100 * t(which) %*% as.matrix(postprob), 1)
 nmod <- length(bic)

### Calculate E[beta_i|D] and SD[beta_i|D] for each coefficient beta_i
 model.fits <- as.list(rep(0, nmod))
 for(k in (1:nmod)) {
  if(sum(which[k,  ]) != 0) {
   model.fits[[k]] <- ls.print(lsfit(x[, which[k,  ],drop=F], y, wt=wt),
                                          print.it = F)$coef.table
  }
```

```
    else model.fits[[k]] <- ls.print(lsfit(rep(1,length(y)),y, wt=wt,int=F),
                                          print.it = F)$coef.table
  }
  Ebi <- rep(0, (nvar+1))
  SDbi <- rep(0,(nvar+1))
  EbiMk <- matrix(rep(0, nmod * (nvar+1)), nrow = nmod)
  sebiMk <- matrix(rep(0, nmod * (nvar+1)), nrow = nmod)
  for(i in 1:(nvar+1)) {
   if((i==1)||(sum(which[, (i-1)] != 0))) {
    for(k in (1:nmod)) {
          if((i==1)||(which[k, (i-1)] == T)) {

###Find position of beta_i in model k
           if (i==1) pos <- 1
            else pos <- 1+sum(which[k, (1:(i-1))])
           EbiMk[k, i] <- model.fits[[k]][pos , 1]
           sebiMk[k, i] <- model.fits[[k]][pos , 2]
          }
    }
    Ebi[i] <- as.numeric(sum(postprob * EbiMk[, i]))
    SDbi[i] <- sqrt(postprob%*%(sebiMk[,i]^2) + postprob%*%((EbiMk[,
i]-Ebi[i])^2))
   }
  }

### Output

  dimnames(which)_list(NULL,dimnames(x)[[2]])
  dimnames(EbiMk)_dimnames(sebiMk)_list(NULL,c("Int",dimnames(x)[[2]]))

  result.list<-list(postprob = postprob, label = label, r2 = r2, bic = bic,
     size = (size - 1), which = which, probne0 = c(probne0), postmean = Ebi,
     postsd = SDbi, mle = EbiMk, se = sebiMk)

#result.list$mle[1,1]
#result.list$mle[1,2]
#result.list$mle[1,3]
#result.list$mle[1,4]

yhat<-matrix(nrow=nmax,ncol=1)

indexi<-1
sum<-0
while (indexi <= nmax)
{
```

```
        yhat[indexi]<-result.list$mle[1,1]
        indexj<-1
        while (indexj <= numxmax)
        {
                yhat[indexi]<-yhat[indexi]+ result.list$mle[1,indexj+1]*x[indexi,indexj]
                indexj<-indexj+1
        }
        sum<-sum+((y[indexi]-yhat[indexi])*(y[indexi]-yhat[indexi]))
        indexi<-indexi+1
}
yhat
sum
mse[loop]<-sum/(nmax-result.list$size[1]-1)

loop<-loop+1
}

amse<-sum(mse)/numloop
std<-matrix(nrow=numloop,ncol=1)

indexi<-1
while (indexi <= numloop )
{
        std[indexi]<-(mse[indexi]-amse)^2
        indexi<-indexi+1
}

stdamse<-sqrt(sum(std)/(numloop-1))
nmax
numxmax
amse
stdamse
```

การทำงานของโปรแกรมที่ใช้ในการวิจัย สำหรับการสร้างตัวแบบด้วยวิธี BMA_{MC3} โดยใช้โปรแกรม S-plus 2000

โปรแกรมที่ใช้ในการสร้างตัวแบบด้วยวิธี BMA_{MC3} นั้นสร้างขึ้นโดยใช้โปรแกรม S-plus 2000 เขียนเป็นฟังก์ชันชื่อ BMA.MC3(....) เป็นฟังก์ชันหลักในการทำงานเพื่อหาปริภูมิตัวแบบและคำนวณความน่าจะเป็นภายหลัง ส่วนฟังก์ชันอื่น ๆ ที่ใช้เป็นส่วนประกอบในการทำงานคือ

1) MC3.REG.chosse เป็นฟังก์ชันที่ถูกเรียกใช้โดยฟังก์ชันหลัก BMA.MC3(....) มีหน้าที่ในการสุ่มหาตัวแบบรูปแบบอื่น ๆ ที่จะนำมาพิจารณาเข้าสู่ปริภูมิตัวแบบ?ฉ รูปแบบการทำงานจึงต่างกับการเขียนโปรแกรมด้วยภาษาวิชวลเบสิก

2) MC3.REG.logpost เป็นฟังก์ชันที่ถูกเรียกใช้โดยฟังก์ชันหลัก BMA.MC3(....)มีหน้าที่ในการคำนวณค่าล็อกของความน่าจะเป็นภายหลัง

โดยรายละเอียดของโปรแกรมมีดังนี้

โปรแกรมสำหรับสร้างตัวแบบด้วยวิธี BMA_{MC3} โดยใช้โปรแกรม S-plus 2000

```
**********************************************************************************************************************

BMA.MC3<-function(all.y,all.x,num.its,M0.var,M0.out,outs.list,PI,K,nu,lambda,phi)
{
#Inputs:
#  all.y -  the response matrix (1 column)
#  all.x -  the matrix of all possible covariates
#  num.its -  the number of iterations in the Markov chain
#  M0.var -  the starting model variable set in T/F format that is the
#            same length as the number of predictors. For example, if you
#            have 3 predictors and the starting model is X_1 and X_3, then
#            M0.var would be c(T,F,T)
#            NOTE:  starting predictor model cannot be the null model.
#  M0.out -  the starting model outlier set in T/F format (see above)
#           this can be NULL only if outs.list is NULL, otherwise must be
#           the same length as outs.list (but can be a vector of all "F's")
#  outs.list -  the list of all potential outlier locations
#             (e.g. c(10,12) means the 10th and 12th points are
#              potential outliers) - can be NULL
```

```
#  PI -  a hyperparameter indicating the probability of an
#        outlier (for Hastings ratio)
#  K -  a hyperparameter indicating the outlier inflation factor
#  nu, lambda, phi - regression hyperparameters

#Outputs:
#  model.matrix -  matrix of selected models (described below)

        Ys <<- scale(all.y)
        Xs <<- scale(all.x)
        M0.var<<-M0.var
        M0.out<<-M0.out
        outs.list <<- outs.list
        K<<-K
        nu<<-nu
        PI<<-PI
        lambda<<-lambda
        phi<<-phi
        flag <<- 1
        outcnt<<-sum(outs.list)
        big.list<<-matrix(0,1,4)
        big.list[1,1]<<-sum(2^((0:(length(M0.var)-1))[M0.var]))+1
        if (sum(M0.out)!=0) big.list[1,2]<<-sum(2^((0:(length(M0.out)-1))[M0.out]))+1

        else big.list[1,2]<<-1

        if (outcnt!=0) big.list[1,3]<<-(dim(Ys)[1]-sum(M0.out))*log(1-PI)+
sum(M0.out)*log(PI)+MC3.REG.logpost(Ys,Xs,M0.var, sum(M0.var),outs.list[M0.out],K,nu,lambda,phi)
        else big.list[1,3]<<-MC3.REG.logpost(Ys,Xs,M0.var, sum(M0.var),outs.list[M0.out],K,nu,lambda,phi)


        i<-1
        while (i <= num.its)
        {
                if (flag==1)
                {
                if (sum(M0.var)!=0) M0.1<<-sum(2^((0:(length(M0.var)-1))[M0.var]))+1
                else M0.1<<-1

                if (sum(M0.out)!=0) M0.2<<-sum(2^((0:(length(M0.out)-1))[M0.out]))+1
                else M0.2<<-1
                }

                M1 <- MC3.REG.choose(M0.var,M0.out)

                if (sum(M1$var)!=0) M1.1<<-sum(2^((0:(length(M0.var)-1))[M1$var]))+1
```

```
                else M1.1<-1

                if (sum(M1$out)!=0) M1.2<<-sum(2^((0:(length(M0.out)-1))[M1$out]))+1
                else M1.2<-1

                if (sum(big.list[,1]==M1.1 & big.list[,2]==M1.2)==0)
                {
                        if (M1.1==1)
                        {
                        if (outcnt!=0) a<-(dim(Ys)[1]-sum(M1$out))*log(1-PI)+
                    sum(M1$out)*log(PI)+MC3.REG.logpost(Ys,Xs,0,0,outs.list[M1$out],K,nu,lambda,phi)
                        else a<-MC3.REG.logpost(Ys,Xs,0,0,outs.list[M1$out],K,nu,lambda,phi)
                        }
                        else
                        {
                        if (outcnt!=0) a<-(dim(Ys)[1]-sum(M1$out))*log(1-
                        PI)+sum(M1$out)*log(PI)+MC3.REG.logpost(Ys,Xs,M1$var,
                        sum(M1$var),outs.list[M1$out],K,nu,lambda,phi)
                        else a<-MC3.REG.logpost(Ys,Xs,M1$var,sum(M1$var),outs.list[M1$out],
                                                K,nu,lambda,phi)
                        }
                        big.list<<-rbind(big.list,c(M1.1,M1.2,a,0))
                }
                BF <- exp(big.list[big.list[,1]==M1.1 & big.list[,2]==M1.2,3]-  big.list[big.list[,1]==M0.1 &
                        big.list[,2]==M0.2,3])

                if (BF >= 1) flag <<- 1
                else flag <<- rbinom(1,1,BF)

                if (flag == 1)
                {
                        M0.var <<- M1$var
                        M0.out <<- M1$out
                        M0.1 <<- M1.1
                        M0.2 <<- M1.2
                }
        big.list[big.list[,1]==M0.1 & big.list[,2]==M0.2,4] <<-big.list[big.list[,1]==M0.1 & big.list[,2]==M0.2,4]+1

                i<-i+1
        }

        var.vect<<-matrix(as.logical(rep(big.list[,1]-1,rep(length(M0.var),length(big.list[,1])))) %/%
2^(0:(length(M0.var) - 1)) %% 2),ncol=length(M0.var),byrow=T)

        n.var<-length(M0.var)
        ndx<-1:n.var
        Xn<-rep("X",n.var)
        labs<-paste(Xn,ndx,sep="")
```

```
        dimnames(var.vect)<-list(c(1:length(var.vect[,1])),labs)

        postprob<<-matrix((exp(big.list[,3]))/(sum(exp(big.list[,3]))),ncol=1)

        dimnames(postprob)[2]_list(c("Post.Mod.Pr."))

        visits<<-matrix(big.list[,4],ncol=1)

        dimnames(visits)[2]<-list(c("#visits"))

        if (length(outs.list)!=0) {out.vect<<-matrix(as.logical(rep(big.list[,2]-1,
rep(length(outs.list),length(big.list[,2])))  %/% 2^(0:(length(outs.list) - 1)) %%2),ncol=length(outs.list),byrow=T)
                dimnames(out.vect)<-list(c(1:length(out.vect[,1])),c(outs.list))
                model.matrix<<-cbind(var.vect,out.vect,postprob,visits)}
        else model.matrix<<-cbind(var.vect,postprob,visits)


        colno<-length(M0.var)+length(M0.out)+1
        model.matrix<<-model.matrix[order(-model.matrix[,colno]),]

        return(model.matrix)

}
*****************************************************************************************************************************************

MC3.REG.choose<-function(M0.var,M0.out)
{
        var <- M0.var
        in.or.out <- sample(c(1:length(M0.var),rep(0,length(M0.out))),1)
        if (in.or.out == 0)
        {
          out<-M0.out
          in.or.out2 <- sample(1:length(M0.out),1)
          out[in.or.out2]<-!M0.out[in.or.out2]
        }
        else
        {
          var[in.or.out]<- !M0.var[in.or.out]
          out <- M0.out  #stay at same outliers, choose a predictor
        }
        return(var,out)
}
*****************************************************************************************************************************************

MC3.REG.logpost<-function(Y, X, model.vect, p, i, K, nu, lambda, phi)
{
        n <- dim(Y)[1]
        ones <- rep(1, n)
```

```
        A <- cbind(ones, X[, model.vect])
        V <- diag(c(1, rep(phi^2, p)))
        ones[i] <- K^2
        det <- diag(ones) + A %*% V %*% t(A)
        divs <- prod(eigen(det, T, T)$values)^0.5
        denom <- (t(Y) %*% solve(det, Y)) + (nu * lambda)
        lgamma((n + nu)/2) + log(nu * lambda) * (nu/2) - log(pi) * (n/2) -
        lgamma(nu/2) - log(divs) - ((nu + n)/2) * log(denom)
}

# ------------------------------------------------
#                Main  Program
# ------------------------------------------------

seterror<-matrix(scan("e_25.txt"),ncol=1,byrow=T)
setx<-matrix(scan("x_1500.txt"),ncol=1,byrow=T)

indexk<-1
indexi<-1

nmax<-100
numxmax<-15

datax<-matrix(nrow=nmax,ncol=numxmax)

while (indexi <= nmax)
{
        indexj<-1
        while (indexj <= numxmax)
        {
                datax[indexi,indexj]<-setx[indexk]
                indexk<-indexk+1
                indexj<-indexj+1
        }
        indexi<-indexi+1
}

numloop<-500
loop<-1
count<-1
numx<-15
samplesize<-100
x<-matrix(nrow=samplesize,ncol=numx)
indexi<-1

while (indexi <= samplesize)
{
        indexj<-1
```

```
        while (indexj <= numx)
        {
                x[indexi,indexj]<-datax[indexi,indexj]
                indexj<-indexj+1
        }
        indexi<-indexi+1
}

mse<-matrix(nrow=numloop,ncol=1)

e<-matrix(nrow=samplesize,ncol=1)
y<-matrix(nrow=samplesize,ncol=1)

while (loop <= numloop)
{
        indexi<-1
        while (indexi <= samplesize)
        {
                e[indexi]<-seterror[count]
                count<-count+1
                indexi<-indexi+1
        }

        indexi<-1
        while(indexi <= samplesize)
        {
                y[indexi]<-1+e[indexi]
                indexj<-1
                while(indexj <= numx)
                {
                        y[indexi]<-y[indexi]+x[indexi,indexj]
                        indexj<-indexj+1
                }
                indexi<-indexi+1
        }

        result<-MC3.REG(y,x,30000,rep(T,numx),NULL,NULL,0,0,nu=2.58,lambda=0.28,phi=2.85)
        indexj<-1
        indexk<-0

        while (indexj <= numx)
        {
                if (result[1,indexj]!=0)
                {
                        indexk<-indexk+1
                }
                indexj<-indexj+1
        }
```

```
        xtemp<-matrix(nrow=samplesize,ncol=indexk)
        indexj<-1
        indexk<-1

        while (indexj <= numx)
        {
                if (result[1,indexj]!=0)
                {
                        xtemp[,indexk]<-x[,indexj]
                        indexk<-indexk+1
                }
                indexj<-indexj+1
        }

        regmodel<-lm(y~xtemp)
        mse[loop]<-sum(regmodel$residuals^2)/(regmodel$df)
        loop<-loop+1
}
amse<-sum(mse)/numloop
std<-matrix(nrow=numloop,ncol=1)
indexi<-1
while (indexi <= numloop)
{
        std[indexi]<-(mse[indexi]-amse)^2
        indexi<-indexi+1
}

stdamse<-sqrt(sum(std)/(numloop-1))
samplesize
numx
numloop
amse
stdamse
mse
```

****

ประวัติผู้เขียนวิทยานิพนธ์

นายนิทัสน์ สุขสุวรรณ เกิดเมื่อวันที่ 4 พฤศจิกายน 2521 ที่จังหวัดกรุงเทพมหานคร สำเร็จการศึกษาระดับปริญญาตรีวิทยาศาสตรบัณฑิต เกียรตินิยมอันดับหนึ่ง สาขาสถิติ จากคณะวิทยาศาสตร์ มหาวิทยาลัยเกษตรศาสตร์ เมื่อ พ.ศ.2542 และเข้าศึกษาต่อในระดับปริญญามหาบัณฑิต สาขาสถิติ ภาควิชาสถิติ คณะพาณิชยศาสตร์และการบัญชี จุฬาลงกรณ์มหาวิทยาลัย เมื่อ พ.ศ.2542