รายการอ้างอิง

ภาษาไทย

จะเด็จ สวรรค์ตรานนท์. การเปรียบเทียบวิธีที่ใช้สำหรับการเลือกสมการถดถอยที่ดีที่สุด. วิทยานิพนธ์ปริญญามหาบัณฑิต ภาควิชาสถิติ บัณฑิตวิทยาลัย จุฬาลงกรณ์มหาวิทยาลัย, 2539.

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

ธีระพร วีระถาวร. ตัวแบบเชิงเส้น ทฤษฎีและการประยุกต์. กรุงเทพมหานคร:บริษัท พิมพ์ดี จำกัด, 2541.

นุชรินทร์ ทิพยวรรณากร. การเปรียบเทียบค่าพยากรณ์ที่ได้จากตัวแบบที่คัดเลือกตัวแปรด้วยวิธีเบย์เซียน วิธีการกำจัดตัวแปรแบบถอยหลัง และวิธีการถดถอยแบบขั้นบันไดในการวิเคราะห์ความถดถอยแบบพหุนามลำดับขั้น. วิทยานิพนธ์ปริญญามหาบัณฑิต ภาควิชาสถิติ บัณฑิตวิทยาลัย จุฬาลงกรณ์มหาวิทยาลัย, 2540.

นพมาศ อัครจันทรโชติ. การเปรียบเทียบวิธีการสร้างตัวแบบในการวิเคราะห์ความถดถอยพหุนามกรณีที่มี 2 ตัวแปรอิสระซึ่งเกิดอันตรกิริยา.วิทยานิพนธ์ปริญญามหาบัณฑิต ภาควิชาสถิติ บัณฑิตวิทยาลัย จุฬาลงกรณ์มหาวิทยาลัย, 2539.

สุชาดา กีระนันทร์. การอนุมานเชิงสถิติ : ทฤษฎีขั้นต้น. กรุงเทพมหานคร : ภาควิชาสถิติ คณะพาณิชยศาสตร์และการบัญชี จุฬาลงกรณ์มหาวิทยาลัย, 2525.

สุพล ดุรงค์วัฒนา. การวิเคราะห์เชิงสถิติ : การวิเคราะห์ความถดถอย. กรุงเทพมหานคร : ภาควิชาสถิติ จุฬาลงกรณ์มหาวิทยาลัย , 2537.

ภาษาต่างประเทศ

Adrian E. Raftery. Bayes factors and BIC : Comment on Weakliem . Technical Report #347, Department of Statistics. University of Washington,1998.

Adrian E. Raftery. Bayesian model selection in social research. Sociological Methodology,1994.

Andrew Gelman, John B.Carlin, and Donald B.Rubin. Bayesian data analysis.London :Chapman & Hall ,1995.

Bernardo, Jos e and Adrian F.M. Smith. Bayesian theory. New York : Wiley ,1994.

Bernd Droge. Some simulation result on cross-validation and competitors for model choice. Technical report. Institute of Mathematics. Humboldt University,1997.

Box,G.E.P..,Tiao,G.C. Bayesian inference in statistical analysis. New York : Wiley Classics,1973.

Bradley P.Carlin and Thomas A.Louis. Baves and Empirical bayes methods for data analysis. New York : Chapman & Hall,1996.

David Madigan and Adrian E. Raftery. Model selection and accounting for model uncertainly in graphical models using occam's window. Technical Report. Department of Statistics, University of Washington.1995.

David G. Kleinbaum, Lawrence L. Kupper,Keith E. Muller and Azhar Nizam. Applied regression analysis and other multivariate methods. Pacific Grove:Duxbury press,1998.

Donna K. Paluer. The schwarz criterion and related methods model selection in linear regression. Technical Report. August 24, 1995.

Edward I. George.The variable selection problem. Technical Report. Department of Statistics.University of Texasat Austin, September 1999.

Elkan F. Halpern. Polynomial regression from a bayesian approach. Journal of The American Statistical Association.Vol.68.No.341(1973):137-147.

H. Linhart and W. Zucchini. Model selection. New York : John Wiley & Sons.1986.

Herbert Lee. Model selection for consumer loan application data. Technical Report Department of Statistics.Carnegie Melllon University. Pittsburgh, PA.15213,1996.

Jennifer A. Hoeting. 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.

John Neter , Michael H. Kutner , Cristopher J. Nachtsheim and William Wasserman Applied linear regression models :Chicago .Times Mirror Higher Education Group,1996.

Joseph E. Cavanaugh and Andrew A. Neath. Generalizing the derivation of the schwarz information criterion. Communications in Statistics – Theory and Methods. Vol. 28 .(1999): 49-66.

Joseph E. Cavanaugh. Unifying the derivations for the akaike and corrected akaike information criteria. Statistics & probability letters Vol. 33. (1997):201-208.

Kass. And Raftery, A.E. Baye factor. Journal Of The American Statistical Association ,Vol. 90.(1995) :377-395.

Larry Wasserman. Bayesian model selection and model averaging. Technical Report #666, Department of Statistics,Carnegie Melllon University,1995.

Norman R. Draper,Harry Smith. Applied regression analysis. New York : John Wiley & Sons,1998.

Ragin C. Charles. The comparative method. Berkeley : University of California Press, 1987.

Robert E. Kass,Larry Wasserman. A reference bayesian test for nested hypotheses and its relationship to the schwarz criterion. Department of Statistics. Technical Report ,Carnegie Mellon University. Pittsburgh, PA. 15213,1994.

Solomon Kullback. Information theory and statistics. New York :Wiley Publication in Statistics Chapman & Hall ,1959.

Steven J. Leon. Linear algebra with applications.New York :Prentice Hall International ,1994.

Terry Sincich. Business statistics by example.{fifth edition}. New York :Prentice Hall International Editions,1991.

Venables, W. and Ripley, B. Modern applied statistics with S-plus. New York : Springer-Verlag,1994.

Zellner, A. An introduction to bayesian inference in econometrics. New York :John Wiley and Sons,1971.

ภาคผนวก

ตารางแสดงลักษณะการทำงานของฟังก์ชันในโปรแกรม S-plus 2000 ทั้งหมดที่ใช้ในการวิจัย

ลำดับที่	ชื่อฟังก์ชัน	การทำงานของฟังก์ชัน
1	unique(x)	-กำหนดค่าของ x ให้มีระยะห่างเท่าๆ กัน เช่น x(1:35)
2	rnorm(x)	-สร้างเลขสุ่มให้มีการแจกแจงแบบปกติ
3	mean(x)	-คำนวณหาค่าเฉลี่ยของข้อมูล
4	sqrt(x)	-คำนวณหาค่ารากที่สองของข้อมูล
5	var(x)	-คำนวณหาค่าความแปรปรวนของข้อมูล
6	length(x)	-แสดงขนาดหรือจำนวนของข้อมูล
7	round(x,dig)	-ทำการปัดเศษ โดย x แทน ชุดข้อมูลที่ทำการปัดเศษ และ dig แทน การกำหนดทศนิยมตามที่ต้องการ
8	matrix(nrow,ncol)	-ทำการเก็บข้อมูลในรูปของเมทริกซ์ โดย nrow แทน จำนวนแถวที่ต้องการ และ ncol แทน จำนวนสดมส์ที่ต้องการ
9	depen()	-การสร้างข้อมูลของตัวแปรตาม
10	indepen()	-การสร้างข้อมูลของตัวแปรอิสระ
11	data.frame(x)	-การเก็บข้อมูลของตัวแปรอิสระโดย 1 สดมส์ แทน 1 ตัวแปร
12	e()	-กำหนดความคลาดเคลื่อนในสมการความถดถอย
13	leaps.bl()	-สร้างตัวแบบที่เป็นไปได้ทั้งหมดของสมการความถดถอย
14	info	-คำนวณหาค่าความแปรปรวนในตัวแบบ
15	cbind	-เก็บค่าให้อยู่ในรูปแบบของสดมภ์
16	rbind	-เก็บค่าให้อยู่ในรูปแบบของแถว
17	which	-สร้างตัวแบบที่เป็นไปได้ทั้งหมดของสมการความถดถอยโดยส่งค่ากลับมาในรูปตรรกศาสตร์
18	coef	-คำนวณสัมประสิทธิ์ความถดถอย
19	nested	-ทำการทดสอบตัวแบบที่ละคู่
20	size	-แสดงขนาดของตัวแปรในตัวแบบ
21	label	-กำหนดชื่อให้กับตัวแปร

23	occam	-ทำการทดสอบตัวแบบที่ละคู่โดยหลักการของ occam's window
24	dev[k]	-คำนวณผลบวกกำลังสองสำหรับแต่ละตัวแบบ
25	df[k]	-แสดงระดับความเป็นอิสระของตัวแบบที่ k
26	order	-จัดอันดับตัวแบบ
27	which.diff	-เปรียบเทียบตัวแบบติดกลุ่ม
28	se	-คำนวณค่าเบี่ยงเบนมาตรฐาน
29	OR	-กำหนดระดับนัยสำคัญให้กับการทดสอบด้วยเกณฑ์ของเบส์
30	alpha	-กำหนดระดับนัยสำคัญให้กับการทดสอบด้วยเกณฑ์ของอาไคเคะ
31	min	-หาค่าต่ำสุด
32	max	-หาค่าสูงสุด
33	glim	-สร้างสมการถดถอยด้วยตัวประมาณภาวะน่าจะเป็นสูงสุด
34	loop	-กำหนดจำนวนรอบของการทำซ้ำ
35	poly(x)	-แปลง เมทริกส์ X ให้เป็นพหุนามเชิงตั้งฉาก

โปรแกรมสำหรับการดำเนินงานวิจัย

ตัวอย่าง โปรแกรม S-plus 2000 สำหรับการเปรียบเทียบตัวแบบความถดถอยพหุนามแบบติดกลุ่ม กรณีที่ $\alpha = 0.05, \rho(x_1, x_2) = 1$

```
polynomial.sim<-function(x,n,d,ind,inde,loop,n=rep(1,length
                (y)),strict=F,OR=20,alpha=6.635,nbest = 150
                ,scale = 1 )

{
   sim1<-0
   sim2<-0
   sim3<-0
   for(L in 1:loop)
{
```

/*ฟังก์ชันการสร้างตัวแปรอิสระและแปลงข้อมูลเข้าสู่ศูนย์กลาง*/

```
  x1 <- function(x)
  {
  x <- unique(x)
  x<-rnorm(x)
  x<-(x-mean(x))/sqrt(var(x))
  n <- length(x)
  contr <- matrix(nrow = n, ncol =1)
  contr[, 1] <- round(x,dig=5)
  contr
  }

  x12 <- function(x)
  {
  x <- unique(x)
  x<-rnorm(x)
  x <-x^2
  x<-(x-mean(x))/sqrt(var(x))
  n <- length(x)
  contr <- matrix(nrow = n, ncol =1)
  contr[, 1] <- round(x,dig=5)
  contr
  }

  x13<- function(x)
  {
  x <- unique(x)
  x<-rnorm(x)
  x <-x^3
  x<-(x-mean(x))/sqrt(var(x))
  n <- length(x)
  contr <- matrix(nrow = n, ncol =1)
  contr[, 1] <- round(x,dig=5)
  contr
  }
```

```
x14<- function(x)
   {
   x <- unique(x)
   x<-rnorm(x)
   x <-x^4
   x<-(x-mean(x))/sqrt(var(x))
   n <- length(x)
   contr <- matrix(nrow = n, ncol =1)
   contr[, 1] <- round(x,dig=5)
   contr
  }

x15<- function(x)
   {
   x <- unique(x)
   x<-rnorm(x)
   x <- x^5
   x<-(x-mean(x))/sqrt(var(x))
   n <- length(x)
   contr <- matrix(nrow = n, ncol =1)
   contr[, 1] <- round(x,dig=5)
   contr
  }

x16<- function(x)
   {
   x <- unique(x)
   x<-rnorm(x)
   x <-x^6
   x<-(x-mean(x))/sqrt(var(x))
   n <- length(x)
   contr <- matrix(nrow = n, ncol =1)
   contr[, 1] <- round(x,dig=5)
   contr
  }

x2<- function(x)
   {
   x<-unique(x)
   x<-rnorm(x)
   x <-2*x
   x<-(x-mean(x))/round(sqrt(var(x)))
   n <- length(x)
   contr <- matrix(nrow = n, ncol =1)
   contr[, 1] <- round(x,dig=5)
   contr
  }
```

```
x22<- function(x)
   {
   x<-unique(x)
   x<-rnorm(x)
   x <- (2*x)^2
   x<-(x-mean(x))/sqrt(var(x))
   n <- length(x)
   contr <- matrix(nrow = n, ncol =1)
   contr[, 1] <- round(x,dig=5)
   contr
  }


x23<- function(x)
   {
   x<-unique(x)
   x<-rnorm(x)
   x <-(2*x)^3
   x<-(x-mean(x))/sqrt(var(x))
   n <- length(x)
   contr <- matrix(nrow = n, ncol =1)
   contr[, 1] <- round(x,dig=5)
   contr
  }

x24<- function(x)
   {
   x<-unique(x)
   x<-rnorm(x)
   x <- (2*x)^4
   x<-(x-mean(x))/sqrt(var(x))
   n <- length(x)
   contr <- matrix(nrow = n, ncol =1)
   contr[, 1] <- round(x,dig=5)
   contr
  }

x25<- function(x)
   {
   x<-unique(x)
   x<-rnorm(x)
   x <- (2*x)^5
   x<-(x-mean(x))/sqrt(var(x))
   n <- length(x)
   contr <- matrix(nrow = n, ncol =1)
   contr[, 1] <- round(x,dig=5)
   contr
  }
```

```
x26<- function(x)
   {
   x<-unique(x)
   x<-rnorm(x)
   x <- (2*x)^6
   x<-(x-mean(x))/sqrt(var(x))
   n <- length(x)
   contr <- matrix(nrow = n, ncol =1)
   contr[, 1] <- round(x,dig=5)
   contr
  }

x112<- function(x)
   {
   x<-unique(x)
   x<-rnorm(x)
   x<-x*(2*x)
   x<-(x-mean(x))/sqrt(var(x))
   n <- length(x)
   contr <- matrix(nrow = n, ncol =1)
   contr[, 1] <- round(x,dig=5)
   contr
  }


x113<- function(x)
   {
   x<-unique(x)
   x<-rnorm(x)
   x<-x*(2*x)^2
   x<-(x-mean(x))/sqrt(var(x))
   n <- length(x)
   contr <- matrix(nrow = n, ncol =1)
   contr[, 1] <- round(x,dig=5)
   contr
  }

x114<- function(x)
   {
   x<-unique(x)
   x<-rnorm(x)
   x<-x*(2*x)^3
   x<-(x-mean(x))/sqrt(var(x))
   n <- length(x)
   contr <- matrix(nrow = n, ncol =1)
   contr[, 1] <- round(x,dig=5)
   contr
  }
```

```
x115<- function(x)
   {
   x<-unique(x)
   x<-rnorm(x)
   x<-x*(2*x)^4
   x<-(x-mean(x))/sqrt(var(x))
   n <- length(x)
   contr <- matrix(nrow = n, ncol =1)
   contr[, 1] <- round(x,dig=5)
   contr
  }

x116<- function(x)
   {
   x<-unique(x)
   x<-rnorm(x)
   x<-x*(2*x)^2
   x<-(x-mean(x))/sqrt(var(x))
   n <- length(x)
   contr <- matrix(nrow = n, ncol =1)
   contr[, 1] <- round(x,dig=5)
   contr
  }

x212<- function(x)
   {
   x<-unique(x)
   x<-rnorm(x)
   x<-x^2*(2*x)
   x<-(x-mean(x))/sqrt(var(x))
   n <- length(x)
   contr <- matrix(nrow = n, ncol =1)
   contr[, 1] <-round(x,dig=5)
   contr
  }

x213<- function(x)
   {
   x<-unique(x)
   x<-rnorm(x)
   x<-x^2*(2*x)^2
   x<-(x-mean(x))/sqrt(var(x))
   n <- length(x)
   contr <- matrix(nrow = n, ncol =1)
   contr[, 1] <- round(x,dig=5)
   contr
  }
```

```
x215<- function(x)
   {
   x<-unique(x)
   x<-rnorm(x)
   x<-x^2*(2*x)^3
   x<-(x-mean(x))/sqrt(var(x))
   n <- length(x)
   contr <- matrix(nrow = n, ncol =1)
   contr[, 1] <- round(x,dig=5)
   contr
  }

x216<- function(x)
   {
   x<-unique(x)
   x<-rnorm(x)
   x<-x^2*(2*x)^4
   x<-(x-mean(x))/sqrt(var(x))
   n <- length(x)
   contr <- matrix(nrow = n, ncol =1)
   contr[, 1] <- round(x,dig=5)
   contr
  }

x311<- function(x)
   {
   x<-unique(x)
   x<-rnorm(x)
   x<-x^3*(2*x)
   x<-(x-mean(x))/sqrt(var(x))
   n <- length(x)
   contr <- matrix(nrow = n, ncol =1)
   contr[, 1] <- round(x,dig=5)
   contr
  }

x312<- function(x)
   {
   x<-unique(x)
   x<-rnorm(x)
   x<-x^3*(2*x)^2
   x<-(x-mean(x))/sqrt(var(x))
   n <- length(x)
   contr <- matrix(nrow = n, ncol =1)
   contr[, 1] <- round(x,dig=5)
   contr
  }
```

```
x313<- function(x)
   {
   x<-unique(x)
   x<-rnorm(x)
   x<-x^3*(2*x)^4
   x<-(x-mean(x))/sqrt(var(x))
   n <- length(x)
   contr <- matrix(nrow = n, ncol =1)
   contr[, 1] <-round(x,dig=5)
   contr
  }

x411<- function(x)
   {
   x<-unique(x)
   x<-rnorm(x)
   x<-x^4*(2*x)
   x<-(x-mean(x))/sqrt(var(x))
   n <- length(x)
   contr <- matrix(nrow = n, ncol =1)
   contr[, 1] <- round(x,dig=5)
   contr
  }

x412<- function(x)
   {
   x<-unique(x)
   x<-rnorm(x)
   x<-x^4*(2*x)^2
   x<-(x-mean(x))/sqrt(var(x))
   n <- length(x)
   contr <- matrix(nrow = n, ncol =1)
   contr[, 1] <-round(x,dig=5)
   contr
  }

x511<- function(x)
   {
   x<-unique(x)
   x<-rnorm(x)
   x<-x^5*(2*x)^2
   x<-(x-mean(x))/sqrt(var(x))
   n <- length(x)
   contr <- matrix(nrow = n, ncol =1)
   contr[, 1] <- round(x,dig=5)
   contr
  }
```

/*ฟังก์ชันการสร้างตัวแปรอิสระและการแปลงข้อมูลด้วยวิธีพหุนามเชิงตั้งฉาก*/

```
x1<-  function(x, normalize = F)
{
x <- unique(x)
x<-rnorm(x)
n <- length(x)
contr <- matrix(0, nrow = n, ncol = 1)
contr[, 1] <- poly(x)
contr
}

x12<-  function(x, normalize = F)
{
x <- unique(x)
x<-rnorm(x)
n <- length(x)
contr <- matrix(0, nrow = n, ncol = 1)
contr[, 1] <- x
contr[,1] <- lsfit(contr[, 1],poly(x^2))$residuals
if(normalize)
apply(contr, 2, function(x) x/sqrt(sum(x^2)))
else contr
contr
}

x13<- function(x, normalize = F)
{
x<-unique(x)
n <- length(x)
contr <- matrix(0, nrow = n, ncol = 1)
contr[, 1] <- x
contr[,1] <- lsfit(contr[, 1],poly(x^3))$residuals
if(normalize)
apply(contr, 2, function(x) x/sqrt(sum(x^2)))
else contr
contr
}

x14<- function(x, normalize = T)
{
x<-unique(x)
n <- length(x)
contr <- matrix(0, nrow = n, ncol = 1)
contr[, 1] <- round(x,dig=2)
contr[,1] <- lsfit(contr[, 1],poly(x^4))$residuals
if(normalize)
apply(contr, 2, function(x) x/sqrt(sum(x^2)))
else contr
round(contr,dig=4)
}
```

```
x15<-function(x, normalize = T)
{
x <- unique(x)
x<-rnorm(x)
n <- length(x)
contr <- matrix(0, nrow = n, ncol = 1)
contr[, 1] <- x
contr[,1] <- lsfit(contr[, 1],poly(x^5))$residuals
if(normalize)
apply(contr, 2, function(x) x/sqrt(sum(x^2)))
else contr
contr
}

x16<-function(x, normalize = T)
{
x <- unique(x)
x<-rnorm(x)
n <- length(x)
contr <- matrix(0, nrow = n, ncol = 1)
contr[, 1] <- x
contr[,1] <- lsfit(contr[, 1],poly(x^6))$residuals
if(normalize)
apply(contr, 2, function(x) x/sqrt(sum(x^2)))
else contr
round(contr,dig=3)
}

x2<-  function(x, normalize = F)
{
x <- unique(x)
x<-rnorm(x)
n<-length(x)
x <-2*x
n <- length(x)
contr <- matrix(0, nrow = n, ncol = 1)
contr[, 1] <-poly((x))
round(contr,dig=2)
}

x22 <- function(x, normalize = F)
{
x <- unique(x)
x<-rnorm(x)
n<-length(x)
x1<-2*x
contr <- matrix(0, nrow = n, ncol = 1)
contr[, 1] <- x
contr[,1] <- lsfit(contr[, 1],(poly((x1)^2)))$residuals
if(normalize)
apply(contr, 2, function(x) x/sqrt(sum(x^2)))
else floor(contr)
}
```

```
x23<- function(x, normalize = F)
{
x <- unique(x)
x<-rnorm(x)
n<-length(x)
x1<-2*x
contr <- matrix(0, nrow = n, ncol = 1)
contr[, 1] <- x
contr[,1] <- lsfit(contr[, 1],poly(x1^3))$residuals
if(normalize)
apply(contr, 2, function(x) x/sqrt(sum(x^2)))
else round(contr,dig=2)
}

x24<-  function(x, normalize = F)
{
x<-unique(x)
x <- rnorm(x)
n<-length(x)
x1<-2*x
contr <- matrix(0, nrow = n, ncol = 1)
contr[, 1] <-x
contr[,1] <- lsfit(contr[, 1],poly(x^4))$residuals
if(normalize)
apply(contr, 2, function(x) x/sqrt(sum(x^2)))
else round(contr,dig=4)
}

x25<-  function(x, normalize = F)
{
x <- unique(x)
x<-rnorm(x)
n<-length(x)
x1<-2*x
contr <- matrix(0, nrow = n, ncol = 1)
contr[, 1] <- x
contr[,1] <- lsfit(contr[, 1],poly(x1^5))$residuals
if(normalize)
apply(contr, 2, function(x) x/sqrt(sum(x^2)))
else round(contr,dig=3)
}
```

```
x26<-  function(x, normalize = F)
{
x <- unique(x)
x<-rnorm(x)
n<-length(x)
x1<-2*x
n <- length(x)
contr <- matrix(0, nrow = n, ncol = 1)
contr[, 1] <- x
contr[,1] <- lsfit(contr[, 1],poly(x1^6))$residuals
if(normalize)
apply(contr, 2, function(x) x/sqrt(sum(x^2)))
else round(contr,dig=5)
}

x112<-  function(x, normalize = F)
{
x <- unique(x)
x<-rnorm(x)
n<-length(x)
x1<-2*x
n <- length(x)
contr <- matrix(0, nrow = n, ncol = 1)
contr[, 1] <- x
contr[,1] <- lsfit(contr[, 1],poly(x*x1))$residuals
if(normalize)
apply(contr, 2, function(x) x/sqrt(sum(x^2)))
else round(contr,dig=4)
}

x113<-  function(x, normalize = F)
{
x <- unique(x)
x<-rnorm(x)
n<-length(x)
x1<-2*x
n <- length(x)
contr <- matrix(0, nrow = n, ncol = 1)
contr[, 1] <- x
contr[,1] <- lsfit(contr[, 1],poly(x*x1^2))$residuals
if(normalize)
apply(contr, 2, function(x) x/sqrt(sum(x^2)))
else round(contr,dig=2)
}
```

```
x114<-  function(x, normalize = F)
{
x<-unique(x)
x <- rnorm(x)
n<-length(x)
x1<-2*x
n <- length(x)
contr <- matrix(0, nrow = n, ncol = 1)
contr[, 1] <- x
contr[,1] <- lsfit(contr[, 1],poly(x*x1^3))$residuals
if(normalize)
apply(contr, 2, function(x) x/sqrt(sum(x^2)))
else round(contr,dig=1)
}

x115<-  function(x, normalize = F)
{
x<-unique(x)
x <- rnorm(x)
n<-length(x)
x1<-2*x
n <- length(x)
contr <- matrix(0, nrow = n, ncol = 1)
contr[, 1] <- x
contr[,1] <- lsfit(contr[, 1],(poly(x*x1^4)))$residuals
if(normalize)
apply(contr, 2, function(x) x/sqrt(sum(x^2)))
else round(contr,dig=2)
}

x116<-  function(x, normalize = F)
{
x <- unique(x)
x<-rnorm(x)
n<-length(x)
x1<-2*x
n <- length(x)
contr <- matrix(0, nrow = n, ncol = 1)
contr[, 1] <- x
contr[,1] <- lsfit(contr[, 1],poly(x*x1^5))$residuals
if(normalize)
apply(contr, 2, function(x) x/sqrt(sum(x^2)))
else round(contr,dig=2)
}
```

```
x212<-  function(x, normalize = F)
{
x <- unique(x)
x<-rnorm(x)
n<-length(x)
x1<-2*x
n <- length(x)
contr <- matrix(0, nrow = n, ncol = 1)
contr[, 1] <- x
contr[,1] <- lsfit(contr[, 1],poly((x^2)*(x1)))$residuals
if(normalize)
apply(contr, 2, function(x) x/sqrt(sum(x^2)))
else round(contr,dig=3)
}

x213<-  function(x, normalize = F)
{
x<-unique(x)
x <- rnorm(x)
n<-length(x)
x1<-2*x
n <- length(x)
contr <- matrix(0, nrow = n, ncol = 1)
contr[, 1] <- x
contr[,1] <- lsfit(contr[, 1],poly((x^2)*(x1^2)))$residuals
if(normalize)
apply(contr, 2, function(x) x/sqrt(sum(x^2)))
else round(contr,dig=3)
}

x215<-  function(x, normalize = F)
{
x<-unique(x)
x <- rnorm(x)
n<-length(x)
x1<-2*x
n <- length(x)
contr <- matrix(0, nrow = n, ncol = 1)
contr[, 1] <- x
contr[,1] <- lsfit(contr[, 1],poly((x^2)*(x1^3)))$residuals
if(normalize)
apply(contr, 2, function(x) x/sqrt(sum(x^2)))
else round(contr,dig=1)
}
```

```
x216<- function(x, normalize = F)
{
x<-unique(x)
x <- rnorm(x)
n<-length(x)
x1<-2*x
n <- length(x)
contr <- matrix(0, nrow = n, ncol = 1)
contr[, 1] <- x
contr[,1] <- lsfit(contr[, 1],poly((x^2)*(x1^4)))$residuals
if(normalize)
apply(contr, 2, function(x) x/sqrt(sum(x^2)))
else round(contr,dig=3)
}


x311<-  function(x, normalize = F)
{
x<-unique(x)
x <- rnorm(x)
n<-length(x)
x1<-2*x
n <- length(x)
contr <- matrix(0, nrow = n, ncol = 1)
contr[, 1] <- x
contr[,1] <- lsfit(contr[, 1],poly((x^3)*(x1)))$residuals
if(normalize)
apply(contr, 2, function(x) x/sqrt(sum(x^2)))
else round(contr,dig=3)
}

x312<-  function(x, normalize = F)
{
x <- unique(x)
x<-rnorm(x)
n<-length(x)
x1<-2*x
n <- length(x)
contr <- matrix(0, nrow = n, ncol = 1)
contr[, 1] <- x
contr[,1] <- lsfit(contr[, 1],poly((x^3)*(x1^2)))$residuals
if(normalize)
apply(contr, 2, function(x) x/sqrt(sum(x^2)))
else round(contr,dig=3)
}
```

```
x313<-  function(x, normalize = F)
{
x <- unique(x)
x<-rnorm(x)
n<-length(x)
x1<-2*x
n <- length(x)
contr <- matrix(0, nrow = n, ncol = 1)
contr[, 1] <- x
contr[,1] <- lsfit(contr[, 1],poly((x^3)*(x1^3)))$residuals
if(normalize)
apply(contr, 2, function(x) x/sqrt(sum(x^2)))
else round(contr,dig=3)
}

x411<-  function(x, normalize = F)
{
x <- unique(x)
x<-rnorm(x)
n<-length(x)
x1<-2*x
n <- length(x)
contr <- matrix(0, nrow = n, ncol = 1)
contr[, 1] <- x
contr[,1] <- lsfit(contr[, 1],poly((x^4)*(x1)))$residuals
if(normalize)
apply(contr, 2, function(x) x/sqrt(sum(x^2)))
else round(contr,dig=3)
}


x412<-  function(x, normalize = F)
{
x <- unique(x)
x<-rnorm(x)
n<-length(x)
x1<-2*x
n <- length(x)
contr <- matrix(0, nrow = n, ncol = 1)
contr[, 1] <- x
contr[,1] <- lsfit(contr[, 1],poly((x^4)*(x1^2)))$residuals
if(normalize)
apply(contr, 2, function(x) x/sqrt(sum(x^2)))
else round(contr,dig=3)
}
```

```
x511<-  function(x, normalize = F)
{
x <- unique(x)
x<-rnorm(x)
n<-length(x)
x1<-2*x
n <- length(x)
contr <- matrix(0, nrow = n, ncol = 1)
contr[, 1] <- x
contr[,1] <- lsfit(contr[, 1],poly((x^5)*(x1)))$residuals
if(normalize)
apply(contr, 2, function(x) x/sqrt(sum(x^2)))
else round(contr,dig=3)
}
```

/*ฟังก์ชันการสร้างความคลาดเคลื่อนอย่างสุ่มและมีการแจกแจงแบบปกติ*/

```
e<-rnorm(n,0,d)
```

/*ฟังก์ชันการสร้างตัวแปรตาม*/

```
      depen<-
                    {
if(ind==1)y<-1+x1(x)+x2(x)+x112(x)+e       else

if(ind==2)y<-1+x1(x)+x12(x)+x2(x)+x22(x)+x112(x)+e     else

if(ind==3)y<-1+x1(x)+x12(x)+x13(x)+x2(x)+x22(x)+x23(x)+x112(x)
             +x113(x)+x212(x)+e  else

if(ind==4)y<-1+x1(x)+x12(x)+x13(x)+x14(x)+x2(x)+x22(x)+x23(x)
             +x24(x)+x112(x)+x113(x)+x114(x)+x212(x)+x213(x)+x311(x)
             +x411(x)+e   else

if(ind==5)y<-1+x1(x)+x12(x)+x13(x)+x14(x)+x15(x)+x2(x)+x22(x)
             +x23(x)+x24(x)+x25(x)+x112(x)+x113(x)+x114(x)+x115(x)
             +x212(x)+x213(x)+x215(x)+x311(x)+x312(x)+x411(x)+e else

if(ind==6)y<-1+x1(x)+x12(x)+x13(x)+x14(x)+x15(x)+x16(x)+x2(x)+x22(x)
             +x23(x)+x24(x)+x25(x)+x26(x)+x112(x)+x113(x)+x114(x)
             +x115(x)+x116(x)+x212(x)+x213(x)+x215(x)+x216(x)
             +x311(x)+x312(x)+x313(x)+x411(x)+x412(x)+x511(x)+e else

stop("haven't model")
                              }
```

/*ฟังก์ชันการสร้างเมทริกส์ของตัวแปรอิสระ*/

```
indepen<-

        {
if(inde==1) data.frame(x.1=x1(x),x.2=x2(x),x.112=x112(x))  else

if(inde==2) data.frame(x.1=x1(x),x.12=x12(x),x.2=x2(x),x.22=x22(x)
                       ,x.112=x112(x))  else

if(inde==3) data.frame(x.1=x1(x),x.12=x12(x),x.13=x13(x),x.2=x2(x)
                       ,x.22=x22(x),x.23=x23(x),x.112=x112(x)
                       ,x.113=x113(x),x.212=x212(x))   else

if(inde==4) data.frame(x.1=x1(x),x.12=x12(x),x.13=x13(x),x.14=x14(x)
                       ,x.2=x2(x),x.22=x22(x),x.23=x23(x),x.24=x24(x)
                       ,x.112=x112(x),x.113=x113(x),x.114=x114(x)
                       ,x.212=x212(x),x.213=x213(x),x.311=x311(x)
                       ,x.411=x411(x))  else


if(inde==5) data.frame(x.1=x1(x),x.12=x12(x),x.13=x13(x),x.14=x14(x)
                       ,x.15=x15(x),x.2=x2(x),x.22=x22(x),x.23=x23(x)
                       ,x.24=x24(x),x.25=x25(x),x.112=x112(x)
                       ,x.113=x113(x),x.114=x114(x),x.115=x115(x)
                       ,x.212=x212(x),x.213=x213(x),x.215=x215(x)
                       ,x.311=x311(x),x.312=x312(x),x.411=x411(x))
                       else

if(inde==6) data.frame(x.1=x1(x),x.12=x12(x),x.13=x13(x),x.14=x14(x)
                       ,x.15=x15(x),x.16=x16(x),x.2=x2(x),x.22=x22(x)
                       ,x.23=x23(x),x.24=x24(x),x.25=x25(x)
                       ,x.26=x26(x),x.112=x112(x),x.113=x113(x)
                       ,x.114=x114(x),x.115=x115(x),x.116=x116(x)
                       ,x.212=x212(x),x.213=x213(x),x.215=x215(x)
                       ,x.216=x216(x),x.311=x311(x),x.312=x312(x)
                       ,x.313=x313(x),x.411=x411(x),x.412=x412(x)
                       ,x.511=x511(x))       else


stop("haven't model")

}
```

```
###-------------------------------------------------------------###
###                Akaike  Information  Criterion                ###
###-------------------------------------------------------------###
```

/*ฟังก์ชันการกำหนดตัวแบบที่เป็นไปได้ทั้งหมด*/

```
leaps.bl <- function(info, coef, names.arg, nbest = nbest)
    {
      names.arg <- names.arg
      if(is.null(names.arg))
      names.arg <- c(as.character(1:9), LETTERS, letters)[1: ncol
      (info)]
      if(length(names.arg) < ncol(info))
      stop("Too few names")
      bIb <- coef %*% info %*% coef
      kx <- ncol(info)
      maxreg <- nbest * kx
      if(kx < 0)    stop("Too few independent variables")
      if(kx >= 31)  stop("Problem too large")
      imeth <- 1
      df <- kx + 1
      Ib <- info %*% coef
      rr <- cbind(info, Ib)
      rr <- rbind(rr, c(Ib, bIb))
      ans <- .Fortran("leaps",
                  ## Fortran routine
                  as.single(rr),
                  as.integer(kx),
                  as.integer(kx + 1),
                  as.integer(df),
                  as.integer(imeth),
                  as.integer(nbest),
                  as.single(0.0001),
                  regid = integer(maxreg),
                  Cp = single(maxreg),
                  size = integer(maxreg),
                  nreg = integer(1),
      single((nbest + 4) * (kx + 1) + (((kx + 1) * (kx + 2))/ 2) *
            ((2 * (kx + 3))/3 + 1)),
      integer(4 * (kx + 1)^2 + 8 * (kx + 1) + (nbest + 1) * (kx +
             1)))[c("Cp", "size",
      "nreg", "regid")]
      nreg <- ans$nreg
      Cp <- ans$Cp
      length(Cp) <- nreg
      size <- ans$size
      length(size) <- nreg
      which <- matrix(T, nreg, kx)
      z <- ans$regid
      length(z) <- nreg
      which <- matrix(as.logical((rep.int(z, kx) %/% rep.int(2^((kx -
                      1):0), rep.int(length(z),   kx))) %% 2), byrow
                      = F, ncol= kx)
      label <- character(nreg)
      sep <- if(all(nchar(names.arg) == 1)) "" else ","
      for(i in 1:nreg)
      label[i] <- paste(names.arg[which[i,  ]], collapse =  sep)
      ans <- list(Cp, size = size, label = label, which = which)
      names(ans)[1] <- "r2"
      ans
    }
```

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


  x01<-indepen
  y<-as.numeric(depen)
  names.x <- paste("X",1:ncol(x01),sep="")
  names.arg <- dimnames(x01)[[2]]
  if(is.null(dimnames(x01)[[2]]))   names.arg <- names.x
  if(is.null(dimnames(x01)[[1]])) dimnames(x01)[[1]]<-paste
    ("R",1:nrow(x01),sep="")
  x2 <- na.omit(data.frame(x01))
  used <- match(dimnames(x01)[[1]], dimnames(x2)[[1]])
  omitted <- seq(nrow(x01))[is.na(used)]
  if(length(omitted) > 0) {
    warning(paste("there were ",length(omitted),"records deleted due
            to NA's"))
    n <- n[ - omitted]
    x01 <- data.matrix(x2)
    y <- y[ - omitted]
  }

  cdf <-cbind.data.frame(y=y,x01)
  mm <- model.matrix(formula(cdf),data=cdf)[,-1,drop=F]
  x01 <- mm
  glim.out <- glim(x01,y,n,error="gaussian",link ="identity",
                   scale="pearson")


### If there are more than 30 columns in x.


  if(ncol(x01) > 30) {
    while(ncol(x01) > 30) {
      glim.out <- glim(x01,y,n,error="gaussian",link = "identity",
                       scale ="pearson")
      coef <- glim.out$coef
      se <- sqrt(diag(glim.out$var))
      pvec <- signif(1 - pchisq((coef/se)^2, 1), 3)
      x01 <- x01[,  - rev(order(pvec))[1]]
    }
  }
```

```
names.arg <- dimnames(x01)[[2]]
 glim.out <- glim(x01, y, n, error = "gaussian", link = "identity",
                  scale ="pearson")
 coef <- glim.out$coef[-1]
 info <- solve(glim.out$var[-1,-1])
 names.arg <- dimnames(x01)[[2]]
 glim.out <- glim(x01, y, n, error = "gaussian", link = "identity",
                  scale ="pearson")
 if(ncol(x01) > 2) {
   glim.out <- glim(x01, y, n,error = "gaussian", link = "identity",
                    scale ="pearson")

a <- leaps.bl(info = info, coef = coef, names.arg = names.arg,
      nbest = nbest)
```

/*ฟังก์ชันการสร้าง null model */

```
    a$r2 <- pmin(pmax(0, a$r2), 99.9)      # Include the null model
    a$r2 <- c(0, a$r2)/100
    a$size <- c(0, a$size)
    a$label <- c("NULL", a$label)
    a$which <- rbind(rep(F, ncol(x01)), a$which)
    nobs <- length(y)
    nmod <- length(a$size)
    dev <- glim.out$deviance[2]
    bIb <- as.numeric(coef %*% info %*% coef)
    lrt <- bIb - (a$r2 * bIb)
    aic <- lrt+2

### Eliminate models with little evidence
```

/*ฟังก์ชันการสร้างตัวแบบติดกลุ่ม*/

```
    nested <- aic - min(aic) < alpha
    r2 <- a$r2[nested]
    size <- a$size[nested]
    label <- a$label[nested]
    which <- a$which[nested,  ,drop=F]
    aic <- aic[nested]
    mse<-mse[nested]
```

```
### Calculate AIC exactly for the remaining models using "glim",
```

/*คำนวณค่า AIC สำหรับแต่ละตัวแบบ*/

```
    model.fits <- as.list(rep(0, length(label)))
    dev <- rep(0, length(label))
    df <- rep(0, length(label))
    for(k in (1:length(label))) {
    if(sum(which[k,  ]) == 0)
         glim.out <- glim(rep(1,nobs),y,n,error = "gaussian",link =
                     "identity",intercept= F,scale ="pearson")
      else {
 x.mat <- matrix(x01[, which[k,  ]], nrow = nobs, dimnames = list
                (NULL, names.arg[which[k,  ]]))

glim.out <- glim(x.mat, y, n, error = "gaussian", link = "identity",
                  scale = "pearson")
      }
      dev[k] <- glim.out$deviance[2]
      df[k] <- glim.out$df[2]
      aic[k] <- -(dev[k])+2*df[k]
      mse[k]<-(dev[k])/(df[k]-1)
      model.fits[[k]] <- matrix(rep(0,2*sum(which[k, ])+ 2),ncol = 2)
            }
  }

   else {
      nmod <- switch(ncol(x01),2,4)
      dev <- df <- aic <-mse<- 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(0, 1, 1, 2)[1:nmod]
      names.arg <- dimnames(x01)[[2]]
      if(is.null(names.arg))
      names.arg <- c(as.character(1:9),LETTERS,letters)[1: ncol(x01)]
      sep <- if(all(nchar(names.arg) == 1)) "" else ","
      nobs <- length(y)
      ncases <- sum(n)
```

```
for(k in 1:nmod) {
      if(sum(which[k,  ]) == 0)
     glim.out <-glim(rep(1,nobs),y,n,error="gaussian",link="identity"
                     , intercept = F,  scale  ="pearson")
        else {
     x.mat <- matrix(x01[, which[k,  ]], nrow = nobs,dimnames = list
                    (NULL, names.arg[which[k,  ]]))
     glim.out <- glim(x.mat, y, n,error="gaussian",link = "identity",
                      scale ="pearson" )
        }
      dev[k] <- glim.out$deviance[2]
      df[k] <- glim.out$df[2]
      aic[k] <- -(dev[k])+2*df[k]
      mse[k]<-(dev[k])/(df[k]-1)
      label[k] <- paste(names.arg[which[k,  ]], collapse = sep)
      if(k == 1)
        label[k] <- "NULL"
      model.fits[[k]]<- matrix(rep(0,2*sum(which[k, ]) +2), ncol = 2)
        }
    }

nested <- aic - min(aic) < alpha
  dev <- dev[nested]
  df <- df[nested]
  size <- size[nested]
  label <- label[nested]
  which <- matrix(which[nested,  ,drop=F], nrow = sum(nested))
  aic <- aic[nested]
  mse<-mse[nested]

  ### Order models in descending order of Akaike information
criterion
```

/*จัดเรียงค่า AIC จากน้อยไปหามาก*/

```
  order.aic <- order(aic, size, label)
  dev <- dev[order.aic]
  df <- df[order.aic]
  size <- size[order.aic]
  label <- label[order.aic]
  which <- matrix(which[order.aic, ,drop=F ], nrow = sum(nested))
  aic <- bic[order.aic]
  mse<-mse[order.aic]
```

```
### Apply the second rule to eliminate models with better ones nested
within them.
```

/*ทำการทดสอบสมมติฐานแบบติดกลุ่ม*/

```
  if(strict) {
    nmod <- length(aic)
    nested <- 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))
        nested[k] <- F
      }
    }
    dev <- dev[nested]
    df <- df[nested]
    size <- size[nested]
    label <- label[nested]
    which <- which[nested, ,drop=F ]
    aic <- aic[nested]
    mse<-mse[nested]
  }
```

```
###-------------------------------------------------------------###
###                      Backward  Elimination                   ###
###-------------------------------------------------------------###
```

/*ฟังก์ชันการคำนวณวิธีการกำจัดตัวแปรแบบถอยหลัง*/

```
back1<-stepwise(x01, y, method = "backward",f.crit=0.05)
sse1<-back1$rss
sse11<-max(sse1)
ss1<-back1$which
ss2<-back1$f.stat
size2<-back1$size
size22<-max(size2)
n<-length(y)
mse1<-(sse11)/(n-size22-1)
```

ต้นฉบับ หน้าขาดหาย

```
single((nbest + 4) * (kx + 1) + (((kx + 1) * (kx + 2))/ 2) *
            ((2 * (kx + 3))/3 + 1)),
      integer(4 * (kx + 1)^2 + 8 * (kx + 1) + (nbest + 1) * (kx +
             1)))[c("Cp", "size",
      "nreg", "regid")]
      nreg <- ans$nreg
      Cp <- ans$Cp
      length(Cp) <- nreg
      size <- ans$size
      length(size) <- nreg
      which <- matrix(T, nreg, kx)
      z <- ans$regid
      length(z) <- nreg
      which <- matrix(as.logical((rep.int(z, kx) %/% rep.int(2^((kx -
                      1):0), rep.int(length(z),   kx))) %% 2), byrow
                      = F, ncol= kx)
      label <- character(nreg)
      sep <- if(all(nchar(names.arg) == 1)) "" else ","
      for(i in 1:nreg)
      label[i] <- paste(names.arg[which[i,  ]], collapse =  sep)
      ans <- list(Cp, size = size, label = label, which = which)
      names(ans)[1] <- "r2"
      ans
    }

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

  x02<-indepen
  y<-as.numeric(depen)
  names.x <- paste("X",1:ncol(x02),sep="")
  names.arg <- dimnames(x02)[[2]]
  if(is.null(dimnames(x02)[[2]]))   names.arg <- names.x
  if(is.null(dimnames(x02)[[1]])) dimnames(x02)[[1]]<-paste
    ("R",1:nrow(x02),sep="")
  x2 <- na.omit(data.frame(x02))
  used <- match(dimnames(x02)[[1]], dimnames(x2)[[1]])
  omitted <- seq(nrow(x02))[is.na(used)]
```

```
if(length(omitted) > 0) {
    warning(paste("there were ",length(omitted),"records deleted due
            to NA's"))
    n <- n[ - omitted]
    x02 <- data.matrix(x2)
    y <- y[ - omitted]
  }
  cdf <-cbind.data.frame(y=y,x02)
  mm <- model.matrix(formula(cdf),data=cdf)[,-1,drop=F]
  x02 <- mm
  glim.out <- glim(x02, y, n,error="gaussian",link ="identity"
                  ,scale="pearson")

### If there are more than 30 columns in x

  if(ncol(x02) > 30) {
    while(ncol(x02) > 30) {
      glim.out <- glim(x02,y,n,error = "gaussian", link = "identity",
                       scale ="pearson")
      coef <- glim.out$coef
      se <- sqrt(diag(glim.out$var))
      pvec <- signif(1 - pchisq((coef/se)^2, 1), 3)
      x02 <- x02[,  - rev(order(pvec))[1]]
    }
  }
names.arg <- dimnames(x02)[[2]]
  glim.out <- glim(x02, y, n, error = "gaussian", link = "identity",
                  scale ="pearson")
  coef <- glim.out$coef[-1]
  info <- solve(glim.out$var[-1,-1])
  if(ncol(x02) > 2) {
    glim.out <- glim(x02, y, n, error="gaussian", link = "identity",
                     scale ="pearson")
    a <- leaps.bl(info = info, coef = coef, names.arg = names.arg,
nbest = nbest)
```

/*ฟังก์ชันการสร้างตัวแบบ null model*/

```
    a$r2 <- pmin(pmax(0, a$r2), 99.9)      # Include the null model
    a$r2 <- c(0, a$r2)/100
    a$size <- c(0, a$size)
    a$label <- c("NULL", a$label)
    a$which <- rbind(rep(F, ncol(x02)), a$which)
    nobs <- length(y)
    ncases <- sum(n)
    nmod <- length(a$size)
    bIb <- as.numeric(coef %*% info %*% coef)
    lrt <- bIb - (a$r2 * bIb)
    bic <- lrt+log(nobs)

### Eliminate models with little evidence
```

/*ฟังก์ชันการสร้างตัวแบบติดกลุ่ม*/

```
    occam <- bic - min(bic) < 2* log(OR)
    r2 <- a$r2[occam]
    size <- a$size[occam]
    label <- a$label[occam]
    which <- a$which[occam,  ,drop=F]
    bic <- bic[occam]

### Calculate BIC exactly for the remaining models using "glim",
```

/*คำนวณค่า BIC สำหรับแต่ละตัวแบบ

```
    model.fits <- as.list(rep(0, length(label)))
    dev <- rep(0, length(label))
    df <- rep(0, length(label))
    for(k in (1:length(label))) {
      if(sum(which[k,  ]) == 0)
      glim.out <- glim(rep(1, nobs), y, n, error =  "gaussian", link
= "identity", intercept = F, scale = "pearson")
```

```
else {
       x.mat <- matrix(x02[, which[k,  ]], nrow = nobs,dimnames =
list(NULL, names.arg[which[k,  ]]))
glim.out <- glim(x.mat, y, n, error = "gaussian", link = "identity",
scale = "pearson")
      }
      dev[k] <- glim.out$deviance[2]
      df[k] <- glim.out$df[2]
      bic[k] <- nobs*log(df[k]+(dev[k]/scale))
      mse3[k]<-(dev[k])/(df[k]-1)
      model.fits[[k]] <- matrix(rep(0, 2 * sum(which[k,  ]) + 2),
                          ncol = 2)
      model.fits[[k]][, 1] <- glim.out$coef     # CTV 7/95
      model.fits[[k]][, 2] <- sqrt(diag(glim.out$var))
    }      # for loop
  }
    else {  # ncol  < 2
      nmod <- switch(ncol(x02),2,4)
      dev <- df <- bic <-mse3<- 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(0, 1, 1, 2)[1:nmod]
      names.arg <- dimnames(x02)[[2]]
      if(is.null(names.arg))
      names.arg <- c(as.character(1:9), LETTERS, letters)[1: ncol
                    (x02)]
sep <- if(all(nchar(names.arg) == 1)) "" else ","
      nobs <- length(y)
      ncases <- sum(n)
      for(k in 1:nmod) {
      if(sum(which[k,  ]) == 0)
glim.out <- glim(rep(1,nobs),y,n,error="gaussian", link = "identity",
                  intercept = F,  scale ="pearson")
        else {
x.mat <- matrix(x02[, which[k,  ]], nrow = nobs,dimnames = list(NULL,
                names.arg[which[k,  ]]))
    glim.out <- glim(x.mat,y,n,error="gaussian", link = "identity",
                 scale ="pearson" )
        }
      dev[k] <- glim.out$deviance[2]
      df[k] <- glim.out$df[2]
      bic[k] <- nobs*log(df[k]+(dev[k]/scale))
      mse3[k]<-(dev[k])/(df[k]-1)
      label[k] <- paste(names.arg[which[k,  ]], collapse =  sep)
```

```
if(k == 1)
        label[k] <- "NULL"
      model.fits[[k]] <- matrix(rep(0,2*sum(which[k, ]) +2),ncol = 2)
      model.fits[[k]][, 1] <- glim.out$coef
      model.fits[[k]][, 2] <- sqrt(diag(glim.out$var))
      }
    }
```

/*คัดเลือกตัวแบบด้วยวิธีการ Occam's Window*/

```
  occam <- bic - min(bic) < 2 * log(OR)
  dev <- dev[occam]
  df <- df[occam]
  size <- size[occam]
  label <- label[occam]
  which <- matrix(which[occam,  ,drop=F], nrow = sum(occam))
  bic <- bic[occam]
  mse3<-mse3[occam]
  model.fits <- model.fits[occam]

  ### Order models in descending order of posterior probability
```

/*จัดเรียงค่า BIC จากน้อยไปหามาก*/

```
  order.bic <- order(bic, size, label)
  dev <- dev[order.bic]
  df <- df[order.bic]
  size <- size[order.bic]
  label <- label[order.bic]
  which <- matrix(which[order.bic, ,drop=F ], nrow = sum(occam))
  bic <- bic[order.bic]
  mse3<-mse3[order.bic]
  model.fits <- model.fits[order.bic]
```

```
### Apply the second rule to eliminate models with better ones nested
within them.
```

/*คัดเลือกตัวแบบด้วยวิธีการ Occam's Window*/

```
if(strict) {
    nmod <- length(bic)
    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
      }
    }
    dev <- dev[occam]
    df <- df[occam]
    size <- size[occam]
    label <- label[occam]
    which <- which[occam, ,drop=F ]
    bic <- bic[occam]
    mse3<-mse3[occam]
    model.fits <- model.fits[occam]
  }
mmse1<-mse1
mmse2<-mse[1]
mmse3<-mse3[1]
sim1<-sim1+mmse1
sim2<-sim2+mmse2
sim3<-sim3+mmse3
cat("number of loop:",L,fill=T)

}
```

/*คำนวณหาค่า AMSE ทั้ง 3 เกณฑ์*/

```
aBW<-sum(sim1)/loop
aAIC<-sum(sim2)/loop
aBIC<-sum(sim3)/loop
print(cbind(aBW,aAIC,aBIC))

}
```

/*จบการทำงานของโปรแกรม*/

ประวัติผู้เขียนวิทยานิพนธ์

นาย พจนา แววสวัสดิ์ เกิดเมื่อวันที่ 19 กรกฎาคม 2519 ที่ จ. พังงา สำเร็จการศึกษาปริญญาตรีวิทยาศาสตร์บัณฑิต สาขา สถิติประยุกต์ จาก คณะวิทยาศาสตร์ มหาวิทยาลัยหอการค้าไทย เมื่อ พ.ศ.2540 เข้าศึกษาต่อในระดับปริญญามหาบัณฑิต สาขาสถิติ ภาควิชาสถิติ คณะพาณิชยศาสตร์และการบัญชี จุฬาลงกรณ์มหาวิทยาลัย เมื่อ พ.ศ.2541