



เอกสารประกอบการสอนเรื่อง

R and Data Mining: Example and Case Studies

ผู้แต่ง Yanchang Zhao

รศ.ดร.กิตติศักดิ์ เกิดประ淑พ (ผู้แปลและเรียบเรียง)



สารบัญ

หน้า

บทที่ 1 บทนำ	3
บทที่ 2 การนำเข้าและการส่งออกข้อมูล	6
บทที่ 3 การสำรวจข้อมูล (Data Exploration)	9
บทที่ 4 Decision Trees และ Random Forest	24
บทที่ 5 สมการทดแทน (Regression)	38
บทที่ 6 โมเดลการจัดกลุ่ม (Clustering)	47
บทที่ 7 Outlier Detection	58
บทที่ 8 การวิเคราะห์อนุกรมเวลา (Time Series and data Mining)	68
บทที่ 9 การหาโมเดล Association Rules	83
บทที่ 10 การทำเหมืองข้อมูล (Text Mining)	98
บทที่ 11 การวิเคราะห์เครือข่ายสังคม (Social Network Analysis)	118
บทที่ 12 กรณีศึกษา 1: การวิเคราะห์และพยากรณ์ดัชนีราคาที่พักอาศัย	126
บทที่ 13 กรณีศึกษา 2: พยากรณ์การตอบสนองของลูกค้าและผลประโยชน์ที่สูงที่สุด	139
บทที่ 14 กรณีศึกษา 3: การสร้างโมเดลเพื่อใช้พยากรณ์ข้อมูลขนาดใหญ่ที่ใช้หน่วยวัดความจำ�อย	145

บทที่ 1 บทนำ

เอกสารเล่มนี้อธิบายการใช้ภาษา R ในงาน Data Mining โดยเน้นที่ตัวอย่างการใช้งานจริงนักศึกษาควรจะมีความรู้ทางด้าน Data Mining มาก่อน เอกสารเล่มนี้เหมาะสมกับนักศึกษาระดับบัณฑิตศึกษา, นักวิจัย, และผู้ที่ต้องการศึกษาเทคโนโลยีที่นำการวิเคราะห์ข้อมูลมาใช้งานกับข้อมูลที่ซับซ้อน บทนำจะอธิบายเกี่ยวกับหลักการพื้นฐานของภาษา R และ Data Mining

1.1 Data Mining

คือกระบวนการเพื่อใช้ค้นหาความรู้ที่น่าสนใจจากข้อมูลจำนวนมาก โดยใช้สาขาต่างๆ เช่น ชั้นทางด้านสถิติ, การเรียนรู้ของเครื่อง, การเข้าถึงข้อมูล, การรับรูปแบบ

เทคนิคหลักของ Data Mining คือ classification & prediction, clustering, outlier detection (การค้นหารูปแบบที่ปรากฏน้อย ๆ), association rule, sequence analysis, time series, text mining (การค้นหาความรู้จากข้อความ)

ในทางธุรกิจ Data Mining Process สามารถพิจารณาเป็น 6 ขั้นตอน (CRISP-DM, Cross Industry Standard Process for Data Mining) คือ

1. การทำความเข้าใจธุรกิจนั้น (business understanding)
2. การทำความเข้าใจข้อมูล (data understanding)
3. การเตรียมข้อมูล (data preparation)
4. การสร้างโมเดล (modeling)
5. การวัดประสิทธิภาพโมเดล (evaluation)
6. การนำไปใช้งาน (deployment)

ในเอกสารเล่มนี้ เน้นที่ ข้อ 4, 5, 6

1.2 ภาษา R

R คือ ภาษาคอมพิวเตอร์ (รวมทั้งส่วนสนับสนุนต่าง ๆ) ที่นิยมใช้ในด้านสถิติและการแสดงข้อมูลในรูปกราฟฟิก สามารถนำ R มาใช้งานได้กว้างขวาง เพราะมีฟังก์ชันสำหรับ (libraries packages) จำนวนมาก และใช้งานได้ฟรี (ประมาณ 4000 packages บนที่เก็บ CRAN) จึงนิยมนำมาใช้งานทางด้านการศึกษา และทางธุรกิจ

ในเอกสารนี้จะใช้งาน packages ที่เกี่ยวกับการวิเคราะห์ข้อมูลซึ่งสามารถนำมาจาก CRAN Task Views (<http://cran.r-project.org/web/views>) ซึ่งมี packages ทางด้าน

1. Machine Learning and Statistical Learning
2. Cluster analysis and Finite Mixture Models
3. Time Series analysis
4. Multivariate Statistics
5. Analysis of spatial data

เอกสารที่สำคัญเกี่ยวกับ R and Data Mining คือ R Reference Card for Data Mining

1.3 ชุดข้อมูล

ชุดข้อมูลที่ใช้ทดลองในเอกสารเล่มนี้คือ

1.3.1 ข้อมูล iris

นิยมใช้ทดลองในงาน classification ข้อมูลประกอบด้วย 3 classes และ 5 คอลัมน์ มี 50 เรコード โดย 1 class จะแยกออกมาอย่างชัดเจนแต่อีก 2 classes จะแยกออกจากกันไม่ชัดเจน ข้อมูลแต่ละคอลัมน์คือ sepal length, sepal width, petal length, petal width, class

คำสั่งภาษา R เพื่อดูลักษณะข้อมูลแต่ละคอลัมน์คือ

```
> str(iris)

'data.frame': 150 obs. of 5 variables:

$ Sepal.Length: num 5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
$ Sepal.Width: num 3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
$ Petal.Length: num 1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
$ Petal.Width: num 0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
$ Species: Factor w/ 3 levels "setosa", "versicolor", ... : 1 1 1 1 1
1 1 1 1 1 ...
```

1.3.2 ข้อมูล bodyfat

อยู่ใน package mboost ประกอบด้วย 71 แถว และ 10 คอลัมน์ (numeric) มีรายละเอียดคือ

1. age คือ อายุ
2. DEXfat คือ ความอ้วนวัดโดย DXA
3. waistcirc คือ ความยาวรอบเอว
4. hipcirc คือ ความยาวรอบสะโพก
5. elbowbreadth คือ ความกว้างของข้อศอก
6. kneebreadth คือ ความกว้างของเข่า
7. anthro3a คือ ผลรวมของค่า log ที่ได้จากการวัดของ anthropometric
8. anthro3b คือ ผลรวมของค่า log ที่ได้จากการวัดของ anthropometric
9. anthro3c คือ ผลรวมของค่า log ที่ได้จากการวัดของ anthropometric
10. anthro4 คือ ผลรวมของค่า log ที่ได้จากการวัดของ anthropometric

คำสั่งภาษา R เพื่อดูดังข้อมูลคือ

```
> data("bodyfat", package = "mboost")
> str(bodyfat)
'data.frame':    71 obs. of 10 variables:
$ age: num 57 65 59 58 60 61 56 60 58 62 ...
$ DEXfat: num 41.7 43.3 35.4 22.8 36.4 ...
$ waistcirc: num 100 99.5 96 72 89.5 83.5 81 89 80 79 ...
$ hipcirc: num 112 116.5 108.5 96.5 100.5 ...
$ elbowbreadth: num 7.1 6.5 6.2 6.1 7.1 6.5 6.9 6.2 6.4 7 ...
$ kneebreadth: num 9.4 8.9 8.9 9.2 10 8.8 8.9 8.5 8.8 8.8 ...
$ anthro3a: num 4.42 4.63 4.12 4.03 4.24 3.55 4.14 4.04 3.91 3.66 ...
$ anthro3b: num 4.95 5.01 4.74 4.48 4.68 4.06 4.52 4.7 4.32 4.21 ...
$ anthro3c: num 4.5 4.48 4.6 3.91 4.15 3.64 4.31 4.47 3.47 3.6 ...
$ anthro4: num 6.13 6.37 5.82 5.66 5.91 5.14 5.69 5.7 5.49 5.25 ...
```

บทที่ 2 การนำเข้าและการส่งออกข้อมูล

บทนี้จะอธิบายการนำเข้าและการส่งออกข้อมูลของโปรแกรมภาษา R โดยสามารถใช้งานข้อมูลชนิดต่างๆ เช่น ไฟล์ชนิด .csv, SAS, ODBC database, Excel

2.1 การบันทึกและเรียกใช้ข้อมูล

ข้อมูลใน R สามารถบันทึกอยู่ใน file .Rdata ด้วยคำสั่ง save() และสามารถนำข้อมูลใน file นั้นมาใช้งานโดยคำสั่ง load() และสามารถใช้คำสั่ง rm() เพื่อลบ object นั้นที่เก็บอยู่ใน R

```
> a <- 1:10  
  
> save(a, file=".data/dumData.Rdata")  
  
> rm(a)  
  
> load("./data/dumData.Rdata")  
  
> print(a)  
[1] 1 2 3 4 5 6 7 8 9 10
```

2.2 การติดต่อกับไฟล์ .csv

ตัวอย่างต่อไปนี้เป็นการสร้าง dataframe ชื่อ df1 และบันทึกใส่ใน .csv file และดึงข้อมูลจากไฟล์นั้นเข้ามาใน dataframe ชื่อ df2

```
> var1 <- 1:5  
  
> var2 <- (1:5) / 10
```

```

> var3 <- c("R", "and", "Data Mining", "Examples", "Case
Studies")

> df1 <- data.frame(var1, var2, var3)

> names(df1) <- c("VariableInt", "VariableReal", "VariableChar")

> write.csv(df1, "./data/dummmyData.csv", row.names = FALSE)

> df2 <- read.csv("./data/dummmyData.csv")

> print(df2)

```

	VariableInt	VariableReal	VariableChar
1	1	0.1	R
2	2	0.2	and
3	3	0.3	Data Mining
4	4	0.4	Examples
5	5	0.5	Case Studies

2.3 การติดต่อกับไฟล์ SAS โดยใช้คำสั่ง read.ss() ซึ่งอยู่ใน package foreign (ข้ามเนื้อหา)

2.4 การติดต่อกับ database ผ่าน ODBC

โดยใช้ package RODBC

2.4.1 อ่านข้อมูลจาก database

โดยใช้คำสั่ง odbcConnect() เพื่อติดต่อ database และใช้ sqlQuery() เพื่อสร้างคำสั่ง SQL เมื่อต้องการ
เลิกใช้ database ต้องใช้คำสั่ง odbcClose()

```

> library(RODBC)

> connection <- odbcConnect(dsn="servername", uid="userid",
pwd="*****")

> query <- "SELECT * FROM lib.table WHERE ..."

> # or read query from file

> # query <- readChar("data/myQuery.sql", nchars=99999)

> myData <- sqlQuery(connection, query, errors=TRUE)

> odbcClose(connection)

```

นอกจากรันคำสั่ง sqlSave() เพื่อเขียนตารางและ sqlUpdate() เพื่อปรับปรุงตาราง

2.4.2 การติดต่อกับ excel file

โดยมีจำนวนแถวไม่เกิน 65,536 แถว

```
> library(RODBC)  
  
> filename <- "data/dummmyData.xls"  
  
> xlsFile <- odbcConnectExcel(filename, readOnly = FALSE)  
  
> sqlSave(xlsFile, a, rownames = FALSE)  
  
> b <- sqlFetch(xlsFile, "a")  
  
> odbcClose(xlsFile)
```

บทที่ 3 การสำรวจข้อมูล (Data Exploration)

บทนี้จะอธิบายตัวอย่างการพิจารณาข้อมูลในແรັ່ມມຸນຕ່າງໆ ເຊັ່ນ ຈຳນວນຄອລິມນີ້ ໂຄງສ້າງຂອງ object ດ້ວຍກາຍາ R ແລະ ອາຍາ R

3.1 การพิจารณาข้อมูล

ໃນບທນີ້ຈະນຳເຫຼືອມູນຕ່າງໆ ໃນການພິຈາລະນາການ ໃນການພິຈາລະນາການ ໃນການພິຈາລະນາການ ໃນການພິຈາລະນາການ ໃນການພິຈາລະນາການ ໃນການພິຈາລະນາການ ໃນການພິຈາລະນາການ ໃນການພິຈາລະນາການ

```
> dim(iris)
[1] 150 5
> names(iris)
[1] "Sepal.Length" "Sepal.Width" "Petal.Length" "Petal.Width"
      "Species"
> str(iris)
'data.frame':   150 obs. of  5 variables:
 $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ Sepal.Width:  num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ Petal.Width:  num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
 $ Species     : Factor w/ 3 levels "setosa","versicolor",...: 1 1 1
1 1 1 1 1 1 ...
```

```

> attributes(iris)

$names
[1] "Sepal.Length" "Sepal.Width" "Petal.Length" "Petal.Width"
"Species"

$row.names
[1]   1   2   3   4   5   6   7   8   9   10  11  12  13  14  15  16  17  18
[19]  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36
[37]  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54
[55]  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72
[73]  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90
[91]  91  92  93  94  95  96  97  98  99 100 101 102 103 104 105 106 107 108
[109] 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126
[127] 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144
[145] 145 146 147 148 149 150

$class
[1] "data.frame"

```

สามารถดู 5 แถวแรกด้วยคำสั่ง `head()` และ 5 แถวสุดท้ายด้วยคำสั่ง `tail()`

```
> iris[1:5, ]
```

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5.0	3.6	1.4	0.2	setosa

```
> head(iris)
```

```

Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1   5.1       3.5        1.4        0.2      setosa
2   4.9       3.0        1.4        0.2      setosa
3   4.7       3.2        1.3        0.2      setosa
4   4.6       3.1        1.5        0.2      setosa
5   5.0       3.6        1.4        0.2      setosa
6   5.4       3.9        1.7        0.4      setosa

```

```
> tail(iris)
```

```

Sepal.Length Sepal.Width Petal.Length Petal.Width Species
145  6.7       3.3        5.7        2.5      virginica
146  6.7       3.0        5.2        2.3      virginica
147  6.3       2.5        5.0        1.9      virginica
148  6.5       3.0        5.2        2.0      virginica
149  6.2       3.4        5.4        2.3      virginica
150  5.9       3.0        5.1        1.8      virginica

```

และสามารถดูเฉพาะคอลัมน์ Sepal.Length เพียง 10 ตัวแรกด้วยคำสั่งต่อไปนี้

```

> iris[1:10, "Sepal.Length"]
[1] 5.1 4.9 4.7 4.6 5.0 5.4 4.6 5.0 4.4 4.9
> iris$Sepal.Length[1:10]
[1] 5.1 4.9 4.7 4.6 5.0 5.4 4.6 5.0 4.4 4.9

```

3.2 การพิจารณาข้อมูลแต่ละคอลัมน์

คุณการกระจายข้อมูลชนิดตัวเลขของคอลัมน์ได้โดยใช้คำสั่ง summary() ซึ่งจะให้ค่า ต่ำสุด, สูงสุด, ค่าเฉลี่ย, ค่ากลาง (median), ค่า Quartile แรก, ค่า Quartile ที่สาม

สำหรับข้อมูลชนิดข้อความ (categorical, nominal, factor) ไม่สามารถคำนวณได้จะแสดงค่าความถี่ของข้อมูลแต่ละตัว

```
> summary(iris)
```

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
Min.:4.300	Min.:2.000	Min.:1.000	Min.:0.100	setosa:50
1st Qu.:5.100	1st Qu.:2.800	1st Qu.:1.600	1st Qu.:0.300	versicolor:50
Median:5.800	Median:3.000	Median:4.350	Median:1.300	virginica:50
Mean:5.843	Mean:3.057	Mean:3.758	Mean:1.199	
3rd Qu.:6.400	3rd Qu.:3.300	3rd Qu.:5.100	3rd Qu.:1.800	
Max.:7.900	Max.:4.400	Max.:6.900	Max.:2.500	

เราสามารถเรียกใช้ฟังก์ชัน mean() เพื่อค่าเฉลี่ย, median() เพื่อค่ากลาง, range() เพื่อช่วง, และ quantile() เพื่อช่วงในรูปเปอร์เซ็นต์

```
> quantile(iris$Sepal.Length)
```

0%	25%	50%	75%	100%
4.3	5.1	5.8	6.4	7.9

```
> quantile(iris$Sepal.Length, c(.1,.3,.65))
```

10%	30%	65%
4.80	5.27	6.20

สามารถเรียกใช้ฟังก์ชัน var() เพื่อค่าความแปรปรวน (variance), hist() เพื่อกราฟ histogram, density() เพื่อความหนาแน่น

```
> var(iris$Sepal.Length)
```

```
[1] 0.6856935
```

```
> hist(iris$Sepal.Length)
```

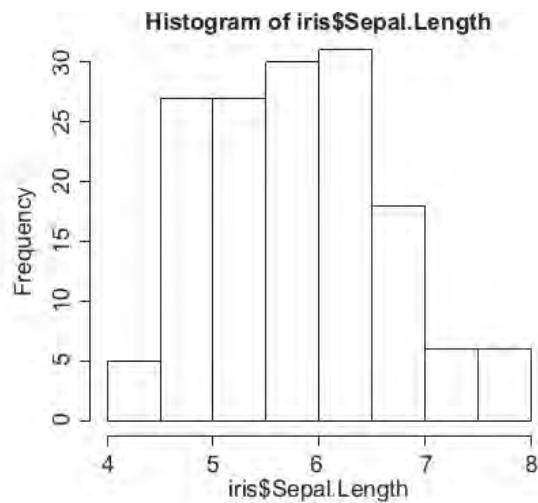


Figure 3.1 Histogram.

```
> plot(density(iris$Sepal.Length))
```

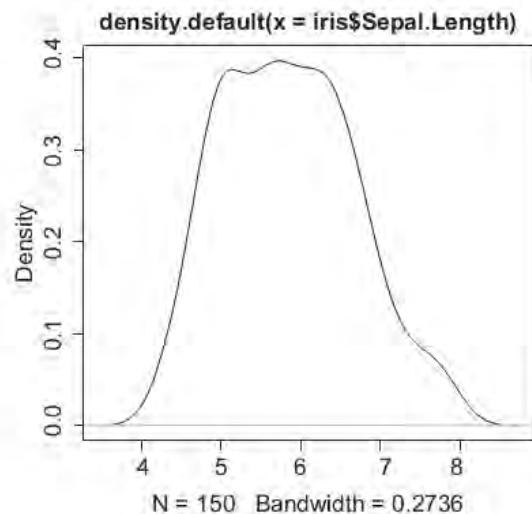


Figure 3.2 Density.

สามารถเรียกใช้ฟังก์ชัน `table()` เพื่อดูค่าความถี่ของข้อมูลชนิด nominal, `pie()` เพื่อดูกราฟพาย, `barplot()` เพื่อดูกราฟบาร์

```
> table(iris$Species)
```

Species	Count
setosa	50
versicolor	50
virginica	50

```
> pie(table(iris$Species))
```

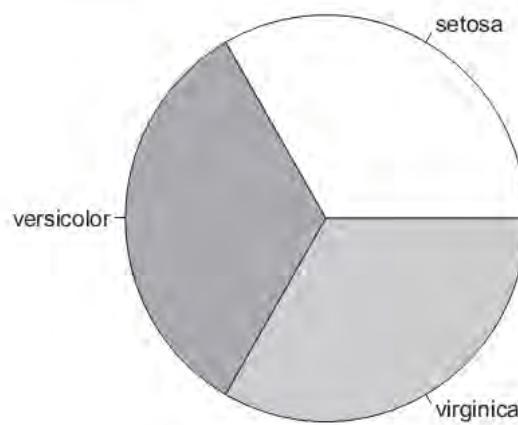


Figure 3.3 Pie chart.

```
> barplot(table(iris$Species))
```

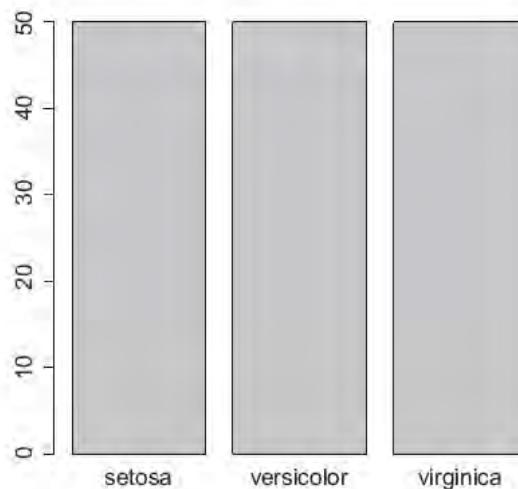


Figure 3.4 Bar chart.

3.3 การพิจารณาข้อมูลหลายคอลัมน์พร้อม ๆ กัน (multiple variables)

เพื่อใช้พิจารณาความสัมพันธ์ระหว่างคอลัมน์ต่าง ๆ ว่ามีความเกี่ยวข้องมากน้อยเพียงไร เช่น ใช้ฟังก์ชัน cov() เพื่อแสดงความแปรปรวนร่วม (covariance), ใช้ cor() เพื่อแสดงความสัมพันธ์ร่วม (correlation)

```
> cov(iris$Sepal.Length, iris$Petal.Length)
```

```
[1] 1.274315
```

```
> cov(iris[,1:4])
```

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
Sepal.Length	0.6856935	-0.0424340	1.2743154	0.5162707
Sepal.Width	-0.0424340	0.1899794	-0.3296564	-0.1216394
Petal.Length	1.2743154	-0.3296564	3.1162779	1.2956094
Petal.Width	0.5162707	-0.1216394	1.2956094	0.5810063

```
> cor(iris$Sepal.Length, iris$Petal.Length)
```

```
[1] 0.8717538
```

```
> cor(iris[,1:4])
```

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
Sepal.Length	1.0000000	-0.1175698	0.8717538	0.8179411
Sepal.Width	-0.1175698	1.0000000	-0.4284401	-0.3661259
Petal.Length	0.8717538	-0.4284401	1.0000000	0.9628654
Petal.Width	0.8179411	-0.3661259	0.9628654	1.0000000

เราสามารถคำนวณค่าสถิติของคอลัมน์ Sepal.Length ของทุก Species ด้วยฟังก์ชัน aggregate()

```
> aggregate(Sepal.Length ~ Species, summary, data=iris)
```

Species	Sepal.Length.Min.	Sepal.Length.1st Qu.	Sepal.Length.Median
1 setosa	4.300	4.800	5.000
2 versicolor	4.900	5.600	5.900
3 virginica	4.900	6.225	6.500

	Sepal.Length.Mean	Sepal.Length.3rd Qu.	Sepal.Length.Max.
1	5.006	5.200	5.800
2	5.936	6.300	7.000
3	6.588	6.900	7.900

เราสามารถใช้ฟังก์ชัน boxplot() เพื่อแสดงกราฟ box (อาจจะเรียกว่า box-and-whisker plot) เพื่อแสดงค่า median, ค่าค่าว่าไถล์, และค่าที่อยู่นอกกลุ่ม (outlier) โดยปีกกลางคือ ค่า median, ขอบกล่อง บน และล่าง คือ ค่า 75% และ 25% (ระยะห่างคือค่า IR=Intr Quartile Range=ขอบกล่อง บน-ล่าง) ข้อมูล outlier จะแสดงด้วยจุดสีส้ม

```
> boxplot(Sepal.Length~Species, data=iris)
```

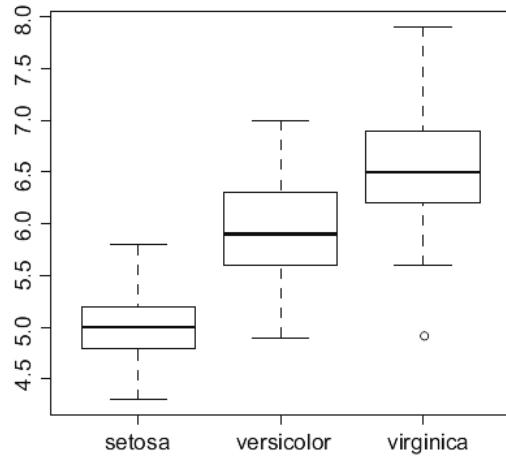


Figure 3.5 Boxplot.

สามารถใช้ฟังก์ชัน `plot()` เพื่อแสดงกราฟจุดที่แสดงความสัมพันธ์ระหว่างแกน x กับแกน y นั่นคือ
ความสัมพันธ์ระหว่าง 2 คอลัมน์ (เราสามารถใช้ `with()` เพื่อไม่ต้องระบุ `iris$` ทุกครั้ง ทำให้สะดวกในการพิมพ์)
ดังนี้

```
> with(iris, plot(Sepal.Length, Sepal.Width, col=Species,  
+ pch=as.numeric(Species)))
```

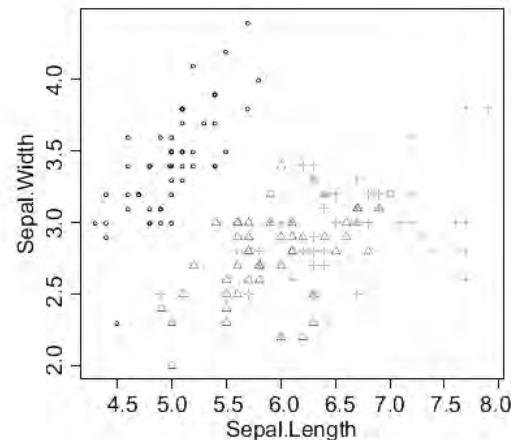


Figure 3.6 Scatter plot.

ในการแสดงกราฟบางครั้งจุดอาจจะซ้อนทับกัน เราสามารถใช้ `jitter()` เพื่อให้เห็นชัดเจนขึ้น

```
> plot(jitter(iris$Sepal.Length), jitter(iris$Sepal.Width))
```

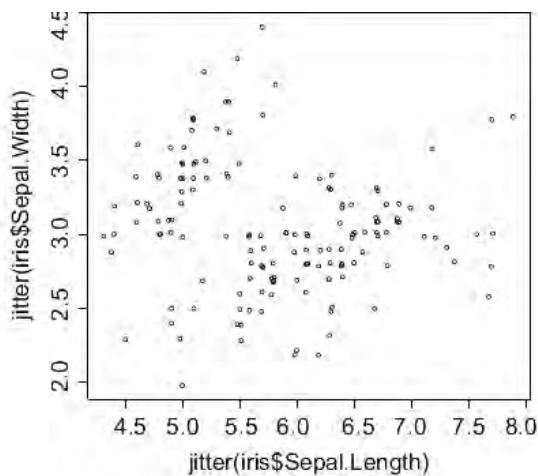


Figure 3.7 Scatter plot with jitter.

สามารถใช้ `pairs()` เพื่อแสดง scatter plots ของทุกคอลัมน์ (ขบวนรุ่น) ที่เป็นไปได้ทั้งหมด

```
> pairs(iris)
```

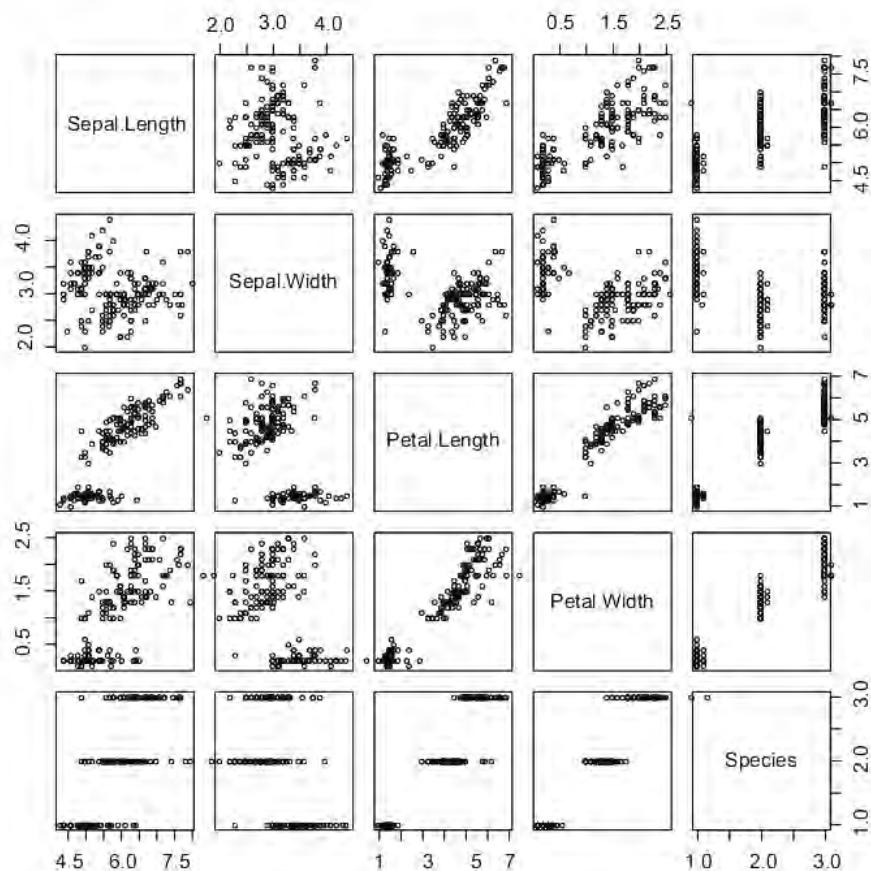


Figure 3.8 A matrix of scatter plots.

3.4 การสำรวจข้อมูลเพิ่มเติม

เป็นการแสดงข้อมูลในรูปแบบที่เกินจำเป็น เช่น 3D, level plot, contour plot, interactive plot, และ parallel coordinates และคงดึงต่อไปนี้

```
> library(scatterplot3d)  
  
> scatterplot3d(iris$Petal.Width, iris$Sepal.Length,  
iris$Sepal.Width)
```

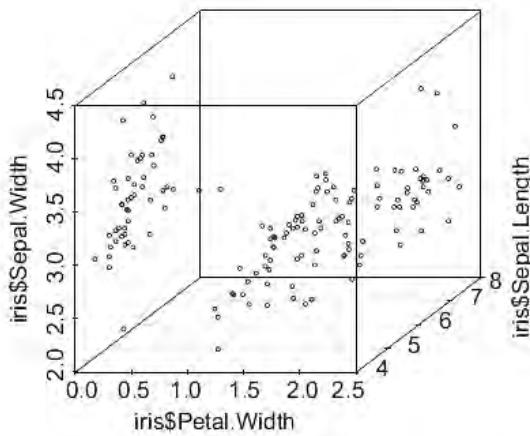


Figure 3.9 3D scatter plot.

package rgl มีฟังก์ชัน plot3d() เพื่อแสดง interactive 3D scatter plot

```
plot3d().  
  
> library(rgl)  
  
> plot3d(iris$Petal.Width, iris$Sepal.Length, iris$Sepal.Width)
```

ฟังก์ชัน heatmap() ใช้แสดงการจับคู่ที่เรียกว่า heat map ในลักษณะ 2D ของข้อมูลเมตริกซ์ เช่น คำสั่งต่อไปนี้แสดงความคล้ายระหว่างคอก iris ด้วยฟังก์ชัน dist()

```
> distMatrix <- as.matrix(dist(iris[, 1:4]))  
  
> heatmap(distMatrix)
```

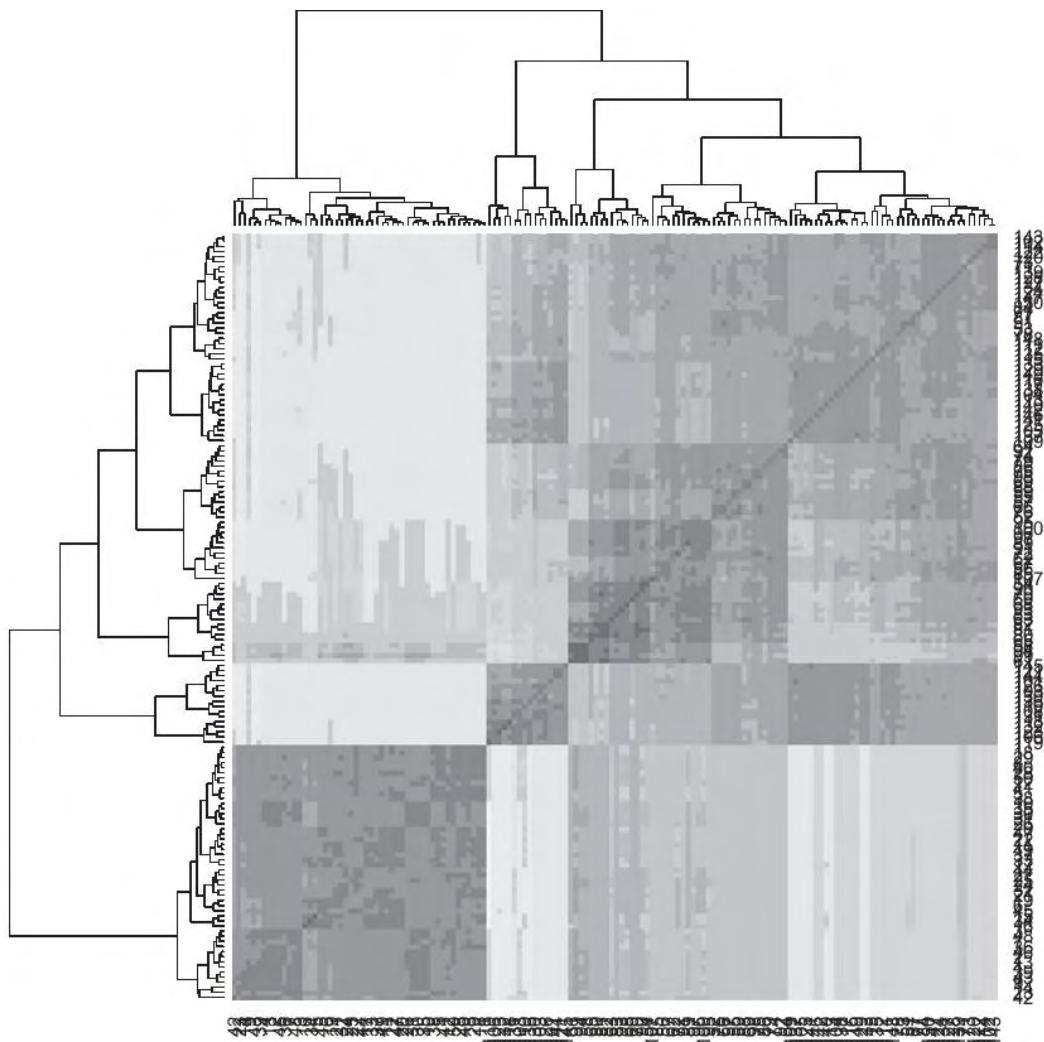


Figure 3.10 Heat map.

เราสามารถใช้ฟังก์ชัน `levelplot()` จาก package `lattice` เพื่อแสดงกราฟชนิด level และสามารถใช้ร่วมกับ `grey.colors()`, `rainbow()` เพื่อแสดงสีต่างๆ

```
> library(lattice)

> levelplot(Petal.Width~Sepal.Length*Sepal.Width, iris, cuts=9,
+           col.regions=grey.colors(10)[10:1])
```

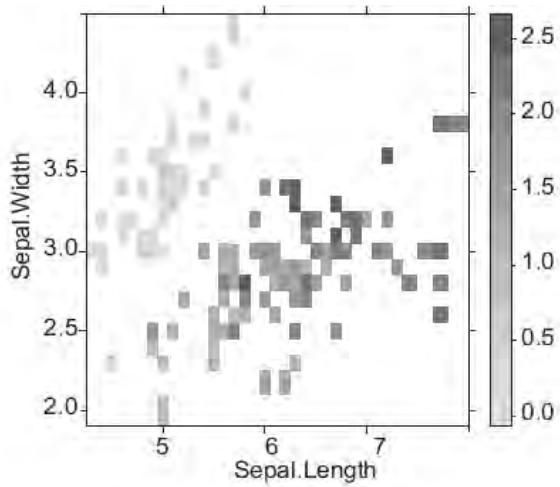


Figure 3.11 Level plot.

พิ้งก์ชัน `contour()` และ `filled.contour()` จาก package `graphics` ใช้แสดงกราฟเชิงพื้นที่ดังต่อไปนี้

```
> filled.contour(volcano, color=terrain.colors, asp=1
+
plot.axes=contour(volcano, add=T))
```

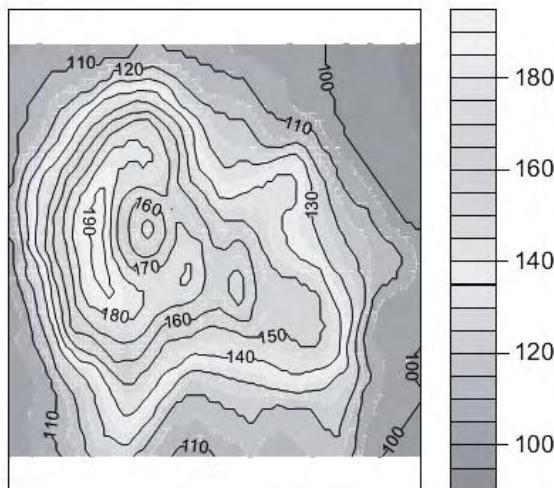


Figure 3.12 Contour.

ในการแสดงข้อมูลตัวเลขที่มี 3 มิติ (3 คอลัมน์) สามารถใช้คำสั่ง persp() ดังนี้

```
> persp(volcano, theta=25, phi=30, expand=0.5,  
col="lightblue")
```

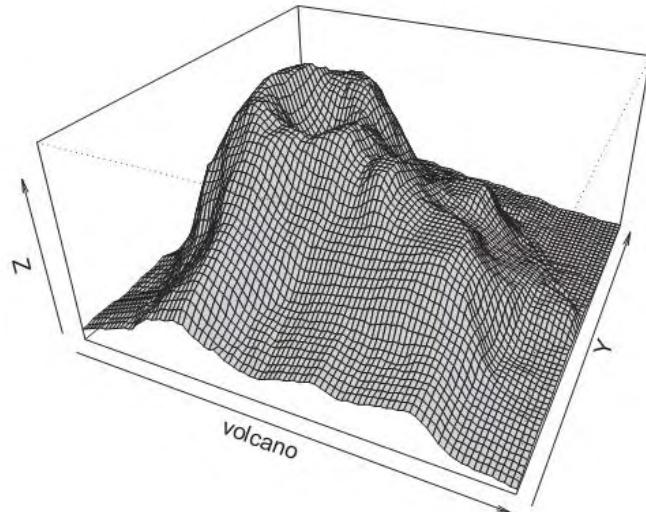


Figure 3.13 3D surface.

ในการแสดงข้อมูลรายเดือนในรูปแบบขนาน (parallel coordinate) โดยการใช้ parcoord() จาก package MASS หรือ parallelplot() จาก package lattice

```
> library(MASS)  
  
> parcoord(iris[1:4], col=iris$Species)
```

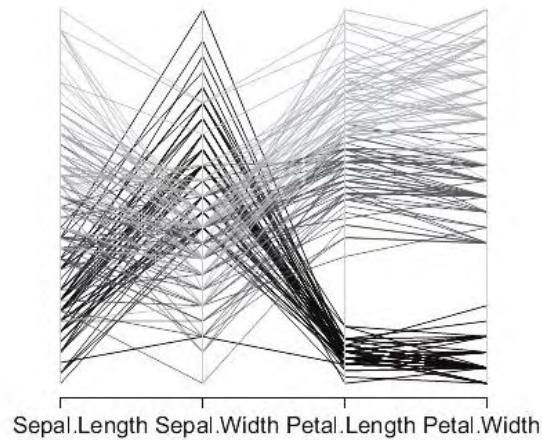


Figure 3.14 Parallel coordinates.

```
> library(lattice)  
  
> parallelplot(~iris[1:4] / Species, data=iris)
```

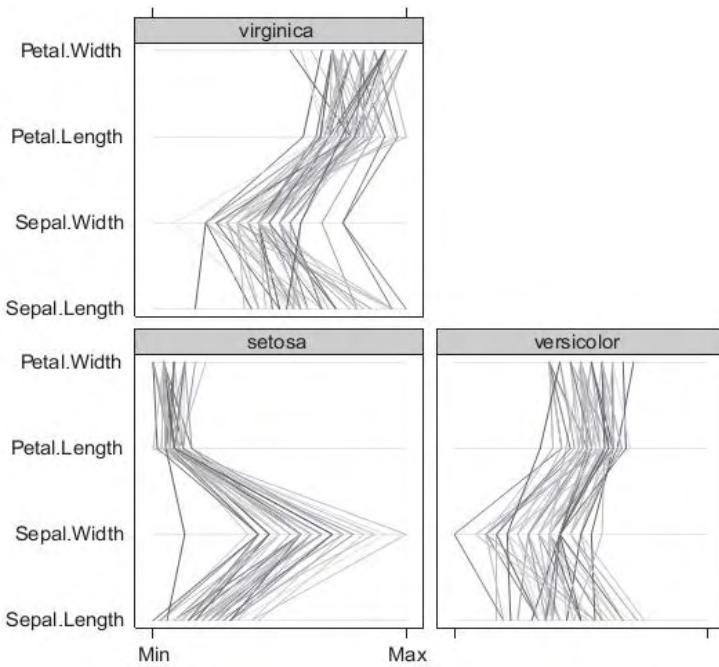


Figure 3.15 Parallel coordinates with package *lattice*.

Package ggplot2 มีฟังก์ชันหลากหลายในการแสดงภาพกราฟฟิก และมีประโยชน์ในการสำรวจข้อมูล เช่นตัวอย่างต่อไปนี้

```
> library(ggplot2)
> qplot(Sepal.Length, Sepal.Width, data=iris, facets=Species ~.)
```

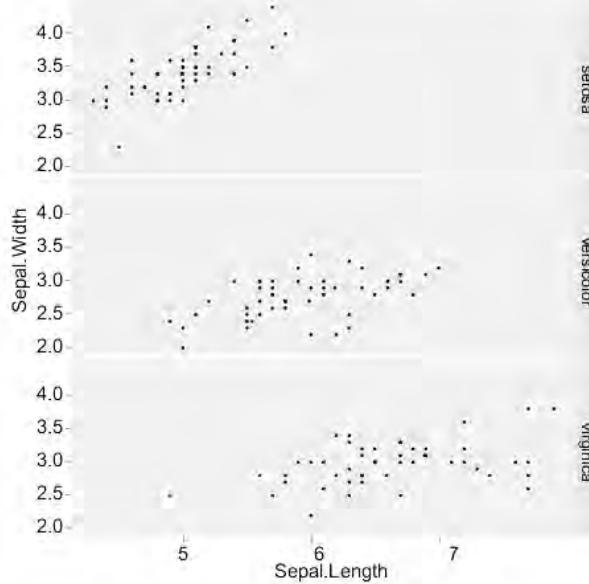


Figure 3.16 Scatter plot with package *ggplot2*.

3.5 การบันทึกรูปกราฟลงไฟล์

ทำได้โดยการบันทึกเป็น pdf file ด้วยฟังก์ชัน `pdf()`, เป็น PS file ด้วย `postscript()` เป็น BMP, JPEG, PNG, TIFF ด้วย `bmp()`, `jpg()`, `png()`, `tiff()` ตามลำดับ

```
> # save as a PDF file  
  
> pdf("myPlot.pdf")  
  
> x <- 1:50  
  
> plot(x, log(x))  
  
> graphics.off()  
  
> #  
  
> # save as a postscript file  
  
> postscript("myPlot2.ps")  
  
> x <- -20:20  
  
> plot(x, x^2)  
  
> graphics.off()
```

บทที่ 4 Decision Trees และ Random Forest

Decision Trees (โมเดลต้นไม้ตัดสินใจ) และ Random Forest (โมเดลป่าแบบสุ่ม) เป็นชนิดของโมเดลที่สำคัญในการท classification & prediction เพราะง่ายและอธิบายได้ชัดเจนจากการนี้ยังมีประสิทธิภาพสูง

4.1 การทำโมเดล Decision Tree ด้วย package party

สามารถใช้ฟังก์ชัน ctree() เพื่อสร้างโมเดล Decision Tree เพื่อคาดการณ์คอลัมน์ Species ของดอกไม้ iris ฟังก์ชัน predict() ใช้ในการคาดการณ์ข้อมูลใหม่ว่ามี specie อะไร

จากข้อมูลดอกไม้ iris ทั้งหมดเรานิยมแยกข้อมูลออกเป็น 2 ชุด คือ ชุด Training (ชุดฝึกสอน) มีจำนวน 70% และ ชุด Test (ชุดทดสอบ) มีจำนวน 30% และค่า seed ที่ใช้เป็นการกำหนดการสุ่มเพื่อแบ่งข้อมูลให้ได้ผลลัพธ์การแบ่งเหมือนเดิม (การใช้ seed ถือว่าไม่เป็นการสุ่มที่แท้จริง)

```
> str(iris)

'data.frame': 150 obs. of 5 variables:

$ Sepal.Length: num 5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
$ Sepal.Width: num 3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
$ Petal.Length: num 1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
$ Petal.Width: num 0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
$ Species: Factor w/ 3 levels "setosa","versicolor",...: 1 1 1 1 1
  1 1 1 1 1 ...

> set.seed(1234)

> ind <- sample(2, nrow(iris), replace=TRUE, prob=c(0.7, 0.3))
> trainData <- iris[ind==1,]
> testData <- iris[ind==2,]
```

ฟังก์ชัน ctree() มีพารามิเตอร์คือ MinSplit, MinBucket, MaxSurrogate, MaxDepth ซึ่งจะอธิบายอย่างละเอียดในบทที่ 13 ถ้าไม่มีการระบุพารามิเตอร์ถือว่าใช้ค่า default

```

> library(party)

> myFormula <- Species ~ Sepal.Length + Sepal.Width +
  Petal.Length + Petal.Width

> iris_ctree <- ctree(myFormula, data=trainData)

> # check the prediction

> table(predict(iris_ctree), trainData$Species)

```

	setosa	versicolor	virginica
setosa	40	0	0
versicolor	0	37	3
virginica	0	1	31

เราสามารถสร้างทรีและแสดงกฎได้ดังนี้

```

> print(iris_ctree)

Conditional inference tree with 4 terminal nodes

Response: Species

Inputs: Sepal.Length, Sepal.Width, Petal.Length, Petal.Width

Number of observations: 112

1) Petal.Length <= 1.9; criterion = 1, statistic = 104.643
  2)* weights = 40

  1) Petal.Length > 1.9
    3) Petal.Width <= 1.7; criterion = 1, statistic = 48.939
      4) Petal.Length <= 4.4; criterion = 0.974, statistic = 7.397
        5)* weights = 21
      4) Petal.Length > 4.4
        6)* weights = 19
      3) Petal.Width > 1.7
        7)* weights = 32

```

```
> plot(iris_ctree)
```

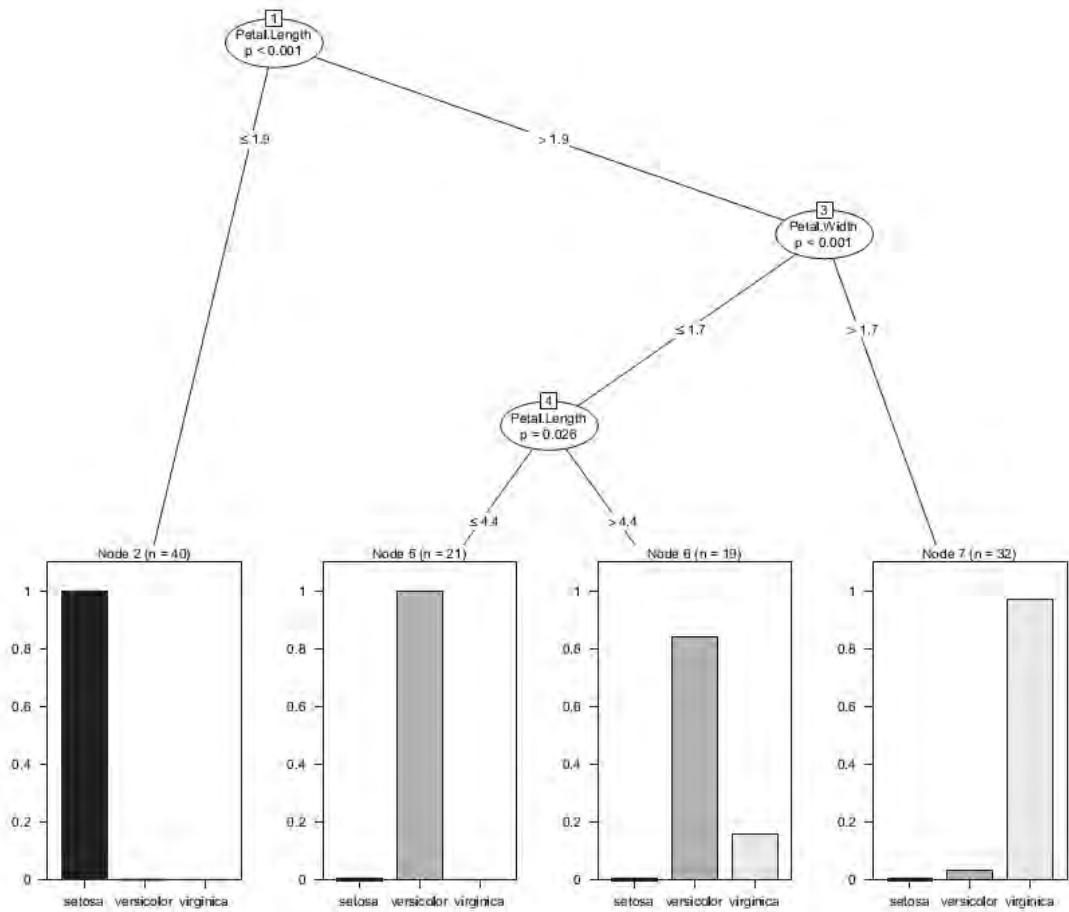


Figure 4.1 Decision tree.

```
> plot(iris_ctree, type="simple")
```

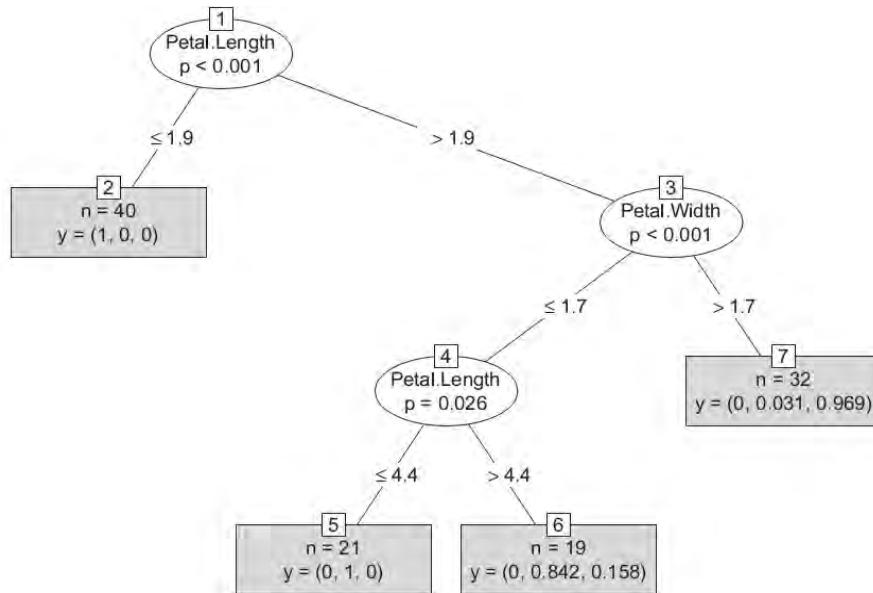


Figure 4.2 Decision tree (simple style).

จากรูปด้านบนในแต่ละโหนดใบจะมีกราฟบาร์เป็นการแสดงความน่าจะเป็นของข้อมูลที่จะอยู่ใน class ต่าง ๆ เช่น โหนดที่ 2 มีค่า “n=40, y(1,0,0)” ความหมายคือมีข้อมูลฝึก 40 ตัวและทุกตัวอยู่ใน class setosa เราสามารถทดสอบข้อมูล Test โดยใช้คำสั่งดังนี้

```
> # predict on test data
> testPred <- predict(iris_ctree, newdata = testData)
> table(testPred, testData$Species)
```

testPred	setosa	versicolor	virginica
setosa	10	0	0
versicolor	0	12	2
virginica	0	0	14

ขุดต้อขของ ctree() ก็อไม่สามารถจัดการ missing value (ใน version 0.9-9995)

4.2 การสร้างโมเดลทรีด้วย package rpart

เราสามารถสร้างโมเดลทรีด้วยฟังก์ชัน rpart() ซึ่งอยู่ใน package rpart และทดสอบข้อมูล Test ด้วยฟังก์ชัน predict() ในที่นี้จะใช้กับข้อมูล bodyfat

ขั้นตอนที่ 1 นำข้อมูล bodyfat เข้ามาใช้ด้วยภาษา R

```
> data("bodyfat", package = "mboost")
> dim(bodyfat)
[1] 71 10
> attributes(bodyfat)

\$names
[1] "age"          "DEXfat"        "waistcirc"     "hipcirc"       "elbowbreadth"
[6] "kneebreadth"  "anthro3a"      "anthro3b"      "anthro3c"      "anthro4"
```

```

$row.names
[1]      "47"  "48"  "49"  "50"  "51"  "52"  "53"  "54"  "55"  "56"  "57"  "58"
[13]     "59"  "60"  "61"  "62"  "63"  "64"  "65"  "66"  "67"  "68"  "69"  "70"
[25]     "71"  "72"  "73"  "74"  "75"  "76"  "77"  "78"  "79"  "80"  "81"  "82"
[37]     "83"  "84"  "85"  "86"  "87"  "88"  "89"  "90"  "91"  "92"  "93"  "94"
[49]     "95"  "96"  "97"  "98"  "99"  "100""101""102""103""104""105""106"
[61]     "107""108""109""110""111""112""113""114""115""116""117"

$class
[1] "data.frame"

> bodyfat[1:5,]

    age DEXfat waistcirc hipcirc elbowbreadth kneebreadth anthro3a
47 57   41.68   100.0     112.0     7.1          9.4        4.42
48 65   43.29   99.5      116.5     6.5          8.9        4.63
49 59   35.41   96.0      108.5     6.2          8.9        4.12
50 58   22.79   72.0      96.5      6.1          9.2        4.03
51 60   36.42   89.5      100.5     7.1         10.0        4.24

anthro3b anthro3c anthro4
4.95      4.50      6.13
5.01      4.48      6.37
4.74      4.60      5.82
4.48      3.91      5.66
4.68      4.15      5.91

```

ขั้นตอนที่ 2 แบ่งข้อมูลออกเป็น 2 กลุ่ม คือ training set, test set และสร้างไมเดลท์

```
> set.seed(1234)

> ind <- sample(2, nrow(bodyfat), replace = TRUE, prob = c(0.7,
  0.3))

> bodyfat.train <- bodyfat[ind==1,]

> bodyfat.test <- bodyfat[ind==2,]

> # train a decision tree

> library(rpart)

> myFormula <- DEXfat ~age + waistcirc + hipcirc + elbowbreadth +
  kneebreadth

> bodyfat_rpart <- rpart(myFormula, data = bodyfat.train,
  +
  control = rpart.control(minsplit = 10))

> attributes(bodyfat_rpart)

$names

[1] "frame"   "where"   "call"     "terms"    "cptable" "splits"
[7] "method"   "parms"   "control"   "functions" "y"        "ordered"

$class

[1] "rpart"

> print(bodyfat_rpart$cptable)

      CP       nsplit rel_error xerror      xstd
1 0.67272638     0 1.0000000 1.0194546 0.18724382
2 0.09390665     1 0.32727362 0.4415438 0.10853044
3 0.06037503     2 0.23336696 0.4271241 0.09362895
4 0.03420446     3 0.17299193 0.3842206 0.09030539
5 0.01708278     4 0.13878747 0.3038187 0.07295556
6 0.01695763     5 0.12170469 0.2739808 0.06599642
7 0.01007079     6 0.10474706 0.2693702 0.06613618
8 0.01000000     7 0.09467627 0.2695358 0.06620732
```

```

3) waistcirc>=88.4 25 1417.1140000 41.34880
   6) waistcirc< 104.75 18 330.5792000 38.09111
      12) hipcirc< 109.9 9 68.9996200 34.37556 *
      13) hipcirc>=109.9 9 13.0832000 41.80667 *
   7) waistcirc>=104.75 7 404.3004000 49.72571 *

> print(bodyfat_rpart)

n = 56

node), split, n, deviance, yval
* denotes terminal node

1) root 56 7265.0290000 30.94589
   2) waistcirc< 88.4 31 960.5381000 22.55645
      4) hipcirc< 96.25 14 222.2648000 18.41143
         8) age< 60.5 9 66.8809600 16.19222 *
         9) age>=60.5 5 31.2769200 22.40600 *
      5) hipcirc>=96.25 17 299.6470000 25.97000
         10) waistcirc< 77.75 6 30.7345500 22.32500 *
            11) waistcirc>=77.75 11 145.7148000 27.95818
               22) hipcirc< 99.5 3 0.2568667 23.74667 *
               23) hipcirc>=99.5 8 72.2933500 29.53750 *
   3) waistcirc>=88.4 25 1417.1140000 41.34880
      6) waistcirc< 104.75 18 330.5792000 38.09111
         12) hipcirc< 109.9 9 68.9996200 34.37556 *
         13) hipcirc>=109.9 9 13.0832000 41.80667 *
      7) waistcirc>=104.75 7 404.3004000 49.72571 *

```

ขั้นตอนที่ 3 แสดงโมเดลทรีที่สร้างขึ้น (ทั้งหมด)

```
> plot(bodyfat_rpart)  
> text(bodyfat_rpart, use.n=T)
```

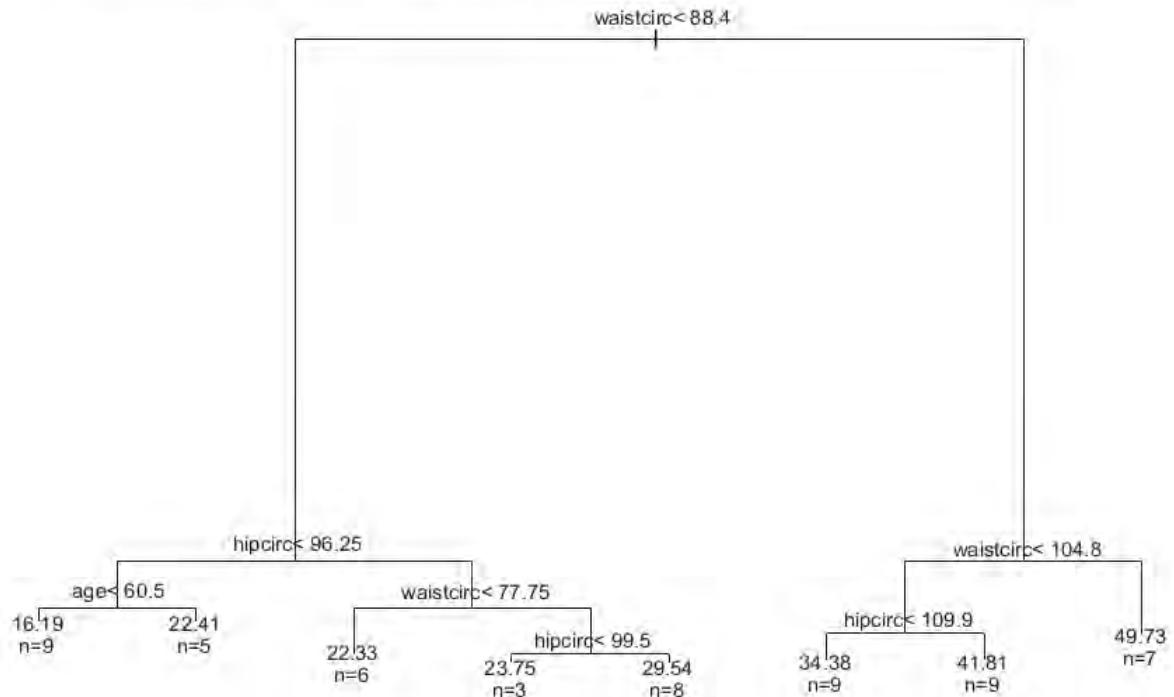


Figure 4.3 Decision tree with package *rpart*.

ขั้นตอนที่ 4 เลือกโมเดลทรีที่มีค่าคาดการณ์ผิดพลาด (prediction error) น้อยที่สุด

```
> opt <- which.min(bodyfat_rpart$cptable[, "xerror"])  
> cp <- bodyfat_rpart$cptable[opt, "CP"]  
> bodyfat_prune <- prune(bodyfat_rpart, cp = cp)  
> print(bodyfat_prune)  
n = 56  
node), split, n, deviance, yval  
* denotes terminal node
```

```

1) root 56 7265.02900 30.94589
2) waistcirc< 88.4 31 960.53810 22.55645
4) hipcirc< 96.25 14 222.26480 18.41143
8) age< 60.5 9 66.88096 16.19222 *
9) age>=60.5 5 31.27692 22.40600 *
5) hipcirc>=96.25 17 299.64700 25.97000
10) waistcirc< 77.75 6 30.73455 22.32500 *
11) waistcirc>=77.75 11 145.71480 27.95818 *
3) waistcirc>=88.4 25 1417.11400 41.34880
6) waistcirc< 104.75 18 330.57920 38.09111
12) hipcirc< 109.9 9 68.99962 34.37556 *
13) hipcirc>=109.9 9 13.08320 41.80667 *
7) waistcirc>=104.75 7 404.30040 49.72571 *
> plot(bodyfat_prune)
> text(bodyfat_prune, use.n=T)

```

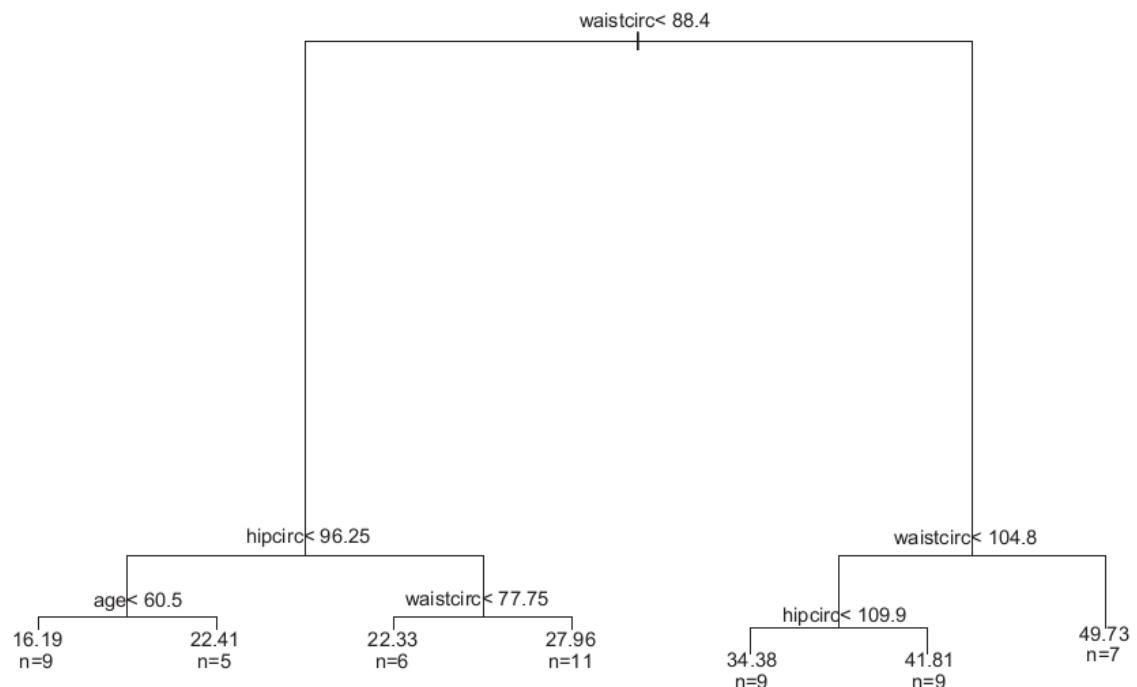


Figure 4.4 Selected decision tree.

ขั้นตอนที่ 5 โมเดลที่จากขั้นตอนที่ 4 จะถูกนำมาใช้เพื่อ predict ค่า class

และจะเปรียบเทียบกับ class ที่แท้จริงว่าถูกต้องมากน้อยเพียงใด (ฟังก์ชัน abline() ใช้ลากเส้นทแยงมุม)

```
> DEXfat_pred <- predict(bodyfat_prune, newdata=bodyfat.test)

> xlim <- range(bodyfat$DEXfat)

> plot(DEXfat_pred ~ DEXfat, data=bodyfat.test, xlab="Observed",
+       ylab="Predicted", ylim=xlim, xlim=xlim)

> abline(a=0, b=1)
```

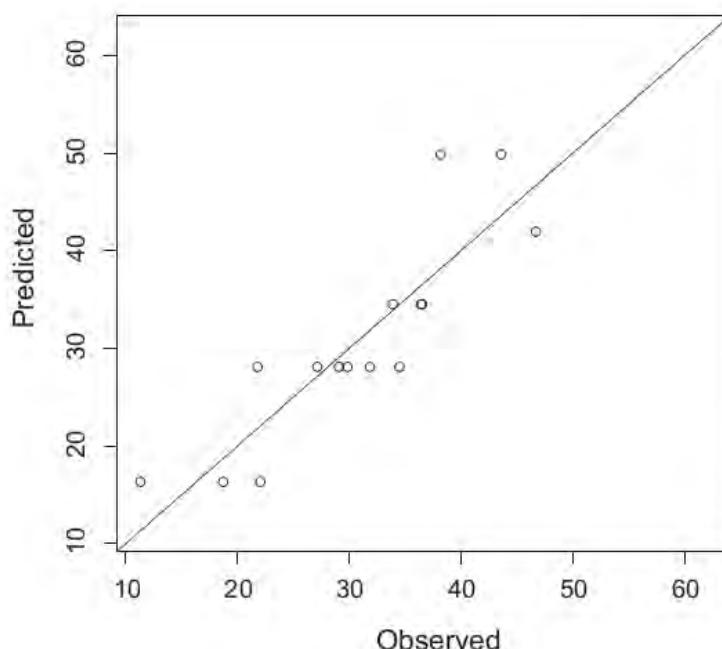


Figure 4.5 Prediction result.

4.3 การสร้างโมเดล Random Forest (กลุ่มของโมเดลที่)

โดยใช้ฟังก์ชัน randomForest() จาก package randomForest

ข้อ ด้อยของฟังก์ชันนี้คือ

- ไม่สามารถใช้กับข้อมูลที่มี missing value (จะต้องจัดการกับ missing value ก่อนทำงาน)
- แต่ละคอลัมน์ที่เป็นชนิด categorical จะมีค่าที่ต่างกันไม่เกิน 32 ค่า (อาจใช้ cforest() จาก package party แทนก็ได้ เพราะไม่มีข้อจำกัดนี้)

ในที่นี้จะใช้ข้อมูล iris ในการศึกษาและทดสอบ

ขั้นตอนที่ 1 แบ่งข้อมูลออกเป็น 2 กลุ่ม คือ training set, test set

```
> ind <- sample(2, nrow(iris), replace=TRUE, prob=c(0.7, 0.3))

> trainData <- iris[ind==1,]

> testData <- iris[ind==2,]
```

ขั้นตอนที่ 2 เรียกใช้ package randomForest

และใช้ข้อมูล training เพื่อสร้างโมเดลโดยระบุ Species~ เพื่อเป็นการนอกว่าใช้พยากรณ์ Species โดยใช้ทุกคอลัมน์ที่เหลือเป็นตัวตั้งต้นในการคาดการณ์

```
> library(randomForest)

> rf <- randomForest(Species ~ ., data=trainData, ntree=100
proximity=TRUE)

> table(predict(rf), trainData$Species)
```

	setosa	versicolor	virginica
setosa	36	0	0
versicolor	0	31	2
virginica	0	1	34

```
> print(rf)
```

Call:

```
randomForest(formula = Species ~ ., data = trainData,
ntree = 100, proximity = TRUE)
```

Type of random forest: classification

Number of trees: 100

No. of variables tried at each split: 2

OOB estimate of error rate: 2.88%

Confusion matrix:

	setosa	versicolor	virginica	class.error
setosa	36	0	0	0.00000000
versicolor	0	31	1	0.03125000
virginica	0	2	34	0.05555556

> attributes(rf)

\$names

```
[1] "call"          "type"          "predicted"      "err.rate"       "err.rate"
[5] "confusion"     "votes"          "oob.times"     "classes"        "classes"
[9] "importance"    "importanceSD"   "localImportance" "proximity"      "proximity"
[13] "ntree"         "mtry"          "forest"         "Y"             "Y"
[17] "test"          "inbag"         "terms"
```

\$class

```
[1] "randomForest.formula" "randomForest"
```

ขั้นตอนที่ 3 แสดงค่าอัตราความผิดพลาด (error rates) ของกลุ่มไม้เดลทีรีบนาดต่าง ๆ

> plot(rf)

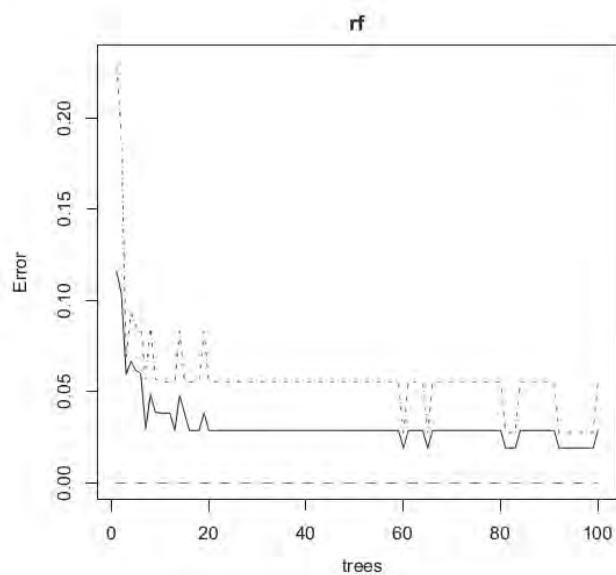


Figure 4.6 Error rate of random forest.

ขั้นตอนที่ 4 แสดงค่าความสำคัญของแต่ละคอลัมน์

ด้วย `importance()` และ `varImpPlot()`

```
> importance(rf)
```

```
MeanDecreaseGini
```

Sepal.Length	6.913882
Sepal.Width	1.282567
Petal.Length	26.267151
Petal.Width	34.163836

```
> varImpPlot(rf)
```

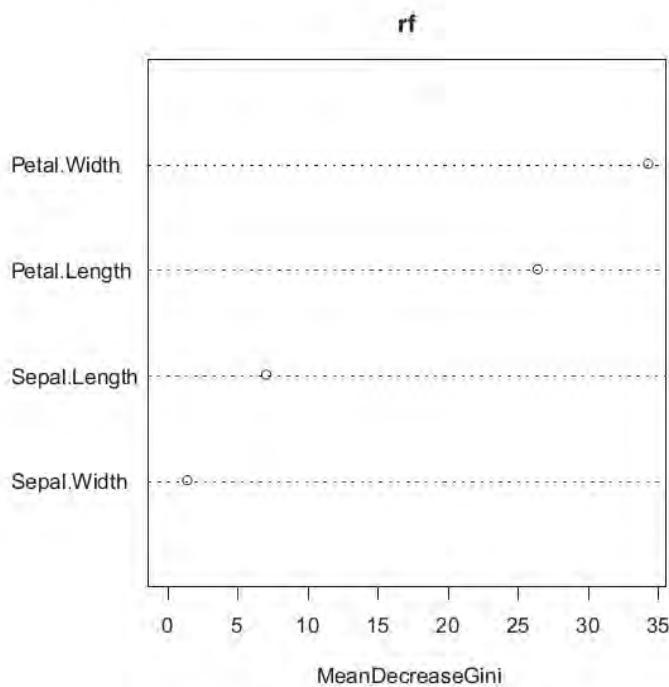


Figure 4.7 Variable importance.

ขั้นตอนที่ 5 ทดสอบโมเดล RandomForest ด้วยข้อมูลทดสอบโดยใช้ `table()` และ `margin()` โดย `margin()` ของแต่ละจุดคือ ค่าพยากรณ์ที่ถูกต้อง (correct classification)

```
> irisPred <- predict(rf, newdata=testData)  
> table(irisPred, testData$Species)
```

irisPred	setosa	versicolor	virginica
setosa	14	0	0
versicolor	0	17	3
virginica	0	1	11

```
> plot(margin(rf, testData$Species))
```

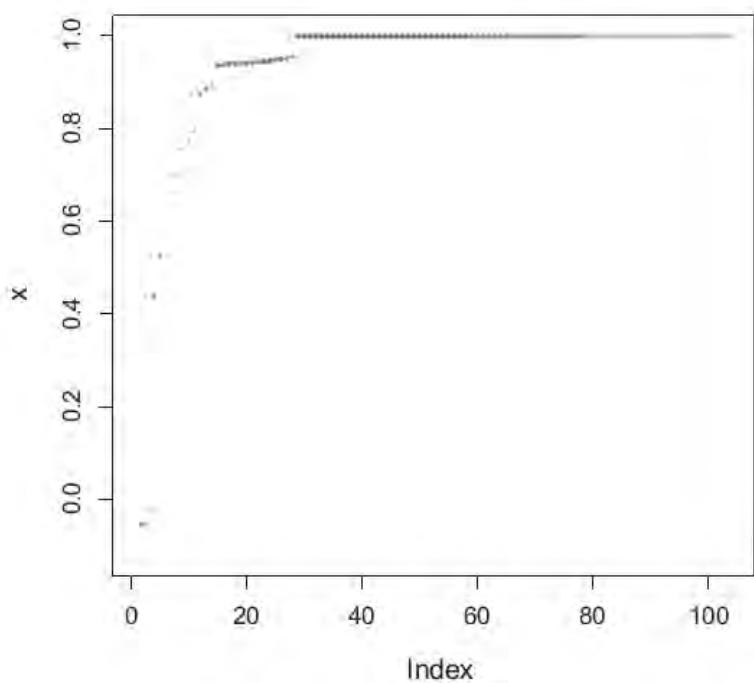


Figure 4.8 Margin of predictions.

บทที่ 5 สมการถดถอย (regression)

Regression คือ พังก์ชัน (สมการ) ระหว่าง ตัวแปรอิสระ (independent) กับ ตัวแปรที่ไม่อิสระ เช่น ขนาดการพยากรณ์ว่าลูกค้าจะชำระเงินรายเดือนเท่าไหร่ โดยดูจาก อายุ, เงินเดือน จึงนำข้อมูลของลูกค้า ในอดีตมาหาความสัมพันธ์ระหว่าง อายุ, เงินเดือน กับการชำระหนี้ ได้สมการคณิตศาสตร์ดังนี้

$$\text{ชำระหนี้รายเดือน} = 1000x\text{อายุ} + \text{เงินเดือน} - 40000$$

เมื่อนำมาใช้งานจะนำอายุของลูกค้า (เช่น 20 ปี), เงินเดือนลูกค้า (เช่น 30000) ใส่ลงในสูตร จะได้ ชำระหนี้รายเดือน = $1000 \times 20 + 30000 - 40000$
= 10000 บาท

จากตัวอย่างนี้ อายุ, เงินเดือน ถูกเรียกว่า ตัวแปรอิสระ (independent variable)

ชำระหนี้รายเดือน ถูกเรียกว่า ตัวแปรไม่อิสระ (dependent variable)

ในทางสถิติ ตัวแปร หมายถึง คอลัมน์ในตาราง

5.1 สมการถดถอยเชิงเส้น (Linear Regression)

มีรูปแบบคือ

$$y = c_0 + c_1 x_1 + c_2 x_2 + \dots + c_k x_k$$

เมื่อ x_i คือ ตัวพยากรณ์ (predictor, independent variable)

y คือ ผลที่ต้องการทราบ (response, dependent variable)

ข้อมูลที่จะใช้ศึกษา คือ ดัชนีผู้บริโภคของคนออสเตรเลีย (Australian Consumer Price Index) ช่วงปี คศ. 2008 ถึง 2010

1. สร้างข้อมูลเอาเองและแสดงข้อมูลในรูป plot ใช้ฟังก์ชัน axis() เพื่อสร้างแกน x, las=3 คือแสดงค่า กำกับในแนวตั้ง

```
> year <- rep(2008:2010, each = 4)  
> quarter <- rep(1:4, 3)  
> cpi <- c(162.2, 164.6, 166.5, 166.0,  
+           166.2, 167.0, 168.6, 169.5,  
+           171.0, 172.1, 173.3, 174.0)  
> plot(cpi, xaxt="n", ylab="CPI", xlab="")
```

```

> # draw x-axis
> axis(1, labels=paste(year,quarter,sep="Q"), at=1:12, las=3)

```

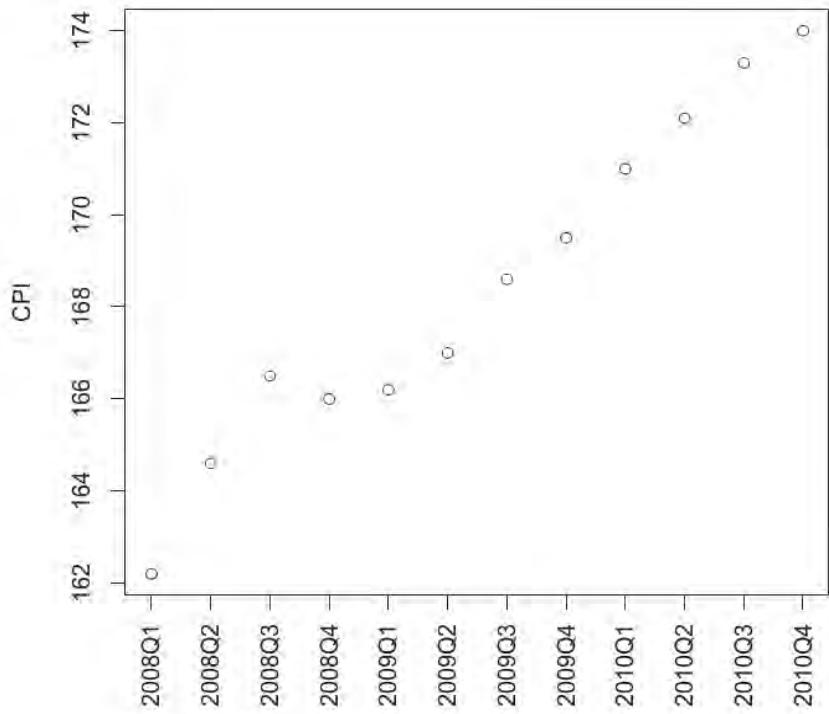


Figure 5.1 Australian CPIs in year 2008 to 2010.

2. แสดง correlation ระหว่าง CPI กับตัวแปรอื่น ๆ (year, quarter)
quarter.

```
> cor(year, cpi)
```

```
[1] 0.9096316
```

```
> cor(quarter, cpi)
```

```
[1] 0.3738028
```

3. สร้างโมเดล regression โดยใช้ lm() โดยใช้ response และ year, quarter เป็น predictor

```
> fit <- lm(cpi ~ year + quarter)
```

```
> fit
```

Call:

```
lm(formula = cpi ~ year + quarter)
```

Coefficients:

	(Intercept)	year	quarter
-7644.488	3.888	1.167	

4. ผลลัพธ์ที่ได้จะอยู่ในรูป

$$cpi = c0 + c1*year + c2*quarter$$

เมื่อ $c0, c1, c2$ คือ coefficient ที่ได้จากข้อ 3

5. ต้องการพยากรณ์ค่า cpi ของปี 2011 (ทุก 3 เดือน) โดยใช้ $predict()$

```
> (cpi2011 <- fit$coefficients[[1]] + fit$coefficients[[2]]*2011 +
+           fit$coefficients[[3]]*(1:4))
[1] 174.4417 175.6083 176.7750 177.9417
```

6. ดูรายละเอียดอื่น ๆ ของโมเดล regression ดังนี้

```
> attributes(fit)
```

\$names

```
[1] "coefficients"   "residuals"     "effects"      "rank"
[5] "fitted.values"  "assign"        "qr"          "df.residual"
[9] "xlevels"         "call"          "terms"        "model"
```

\$class

```
[1] "lm"
```

```
> fit$coefficients
```

	(Intercept)	year	quarter
-7644.487500	3.887500	1.166667	

7. แสดงความผิดพลาด (error, residual) ระหว่างข้อมูลที่มีอยู่จริงกับค่าที่ได้จากการ

```
> # differences between observed values and fitted values
```

```
> residuals(fit)
```

1	2	3	4	5	6
-0.57916667	0.65416667	1.38750000	-0.27916667	-0.46666667	-0.83333333
7	8	9	10	11	12
-0.40000000	-0.66666667	0.44583333	0.37916667	0.41250000	-0.05416667

```
> summary(fit)
```

Call:

```
lm(formula = cpi ~ year + quarter)
```

Residuals:

Min	1Q	Median	3Q	Max
-0.8333	-0.4948	-0.1667	0.4208	1.3875

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)	
(Intercept)	-7644.4875	518.6543	-14.739	1.31e-07	***
year	3.8875	0.2582	15.058	1.09e-07	***
quarter	1.1667	0.1885	6.188	0.000161	***

Signif. codes:	0 ***	0.001 **	0.01 *	0.05 .	0.1

Residual standard error: 0.7302 on 9 degrees of freedom

Multiple R-squared: 0.9672, Adjusted R-squared: 0.9599

F-statistic: 132.5 on 2 and 9 DF, p-value: 2.108e-07

8. แสดงโมเดลที่ได้โดยใช้ `plot()`

```
> plot(fit)
```

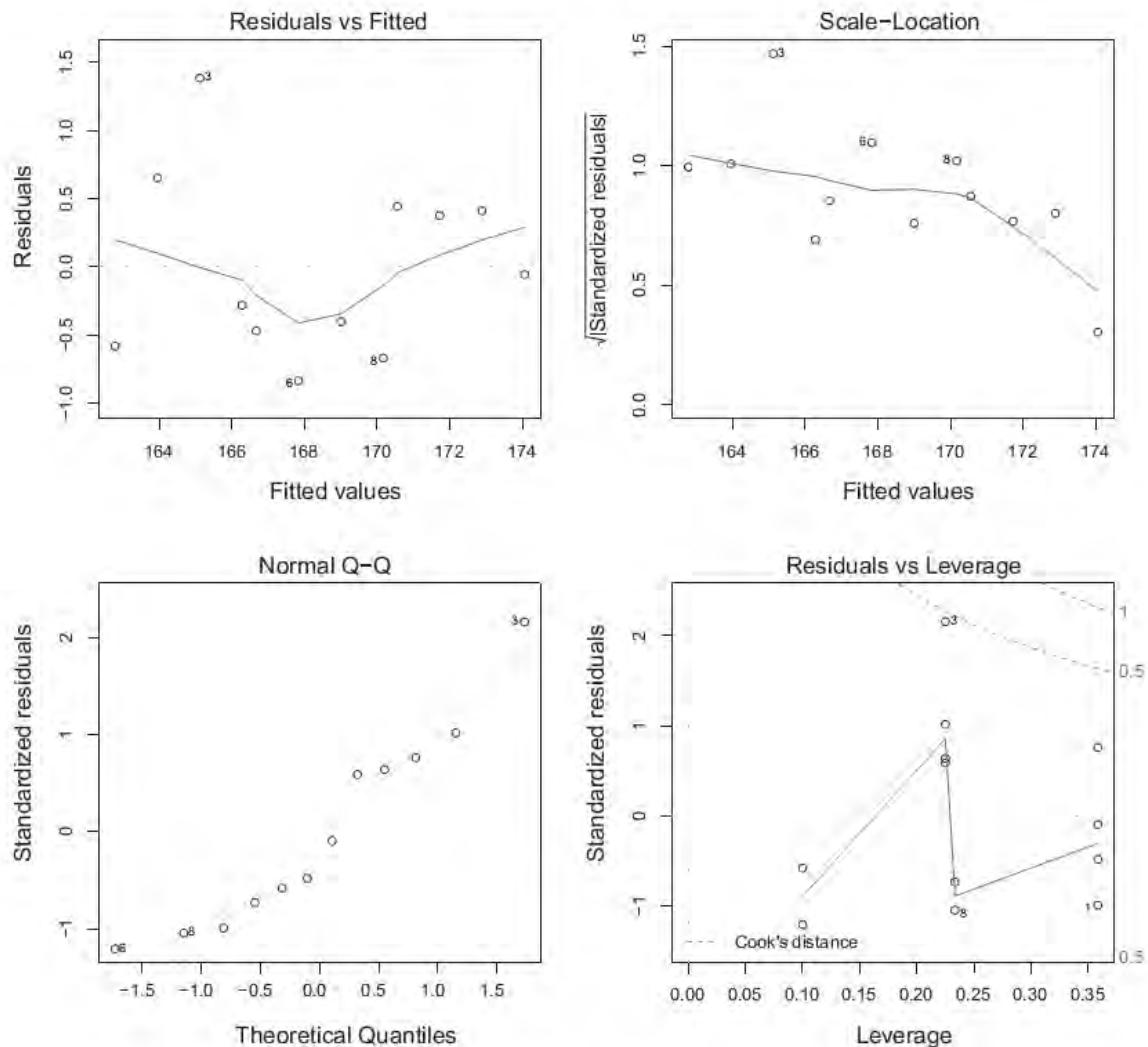


Figure 5.2 Prediction with linear regression model.

เราสามารถดูโมเดลในลักษณะ 3 มิติโดยใช้ `scatterplot3d()` และ `plane3d()` เพื่อใช้แสดงรูปแบบของโมเดล (พารามิเตอร์ lab ระบุความหนาของเส้นแกน x, y)

```
> library(scatterplot3d)
> s3d <- scatterplot3d(year, quarter, cpi, highlight.3d=T,
  type="h", lab=c(2,3))
```

```
> s3d$plane3d(fit)
```

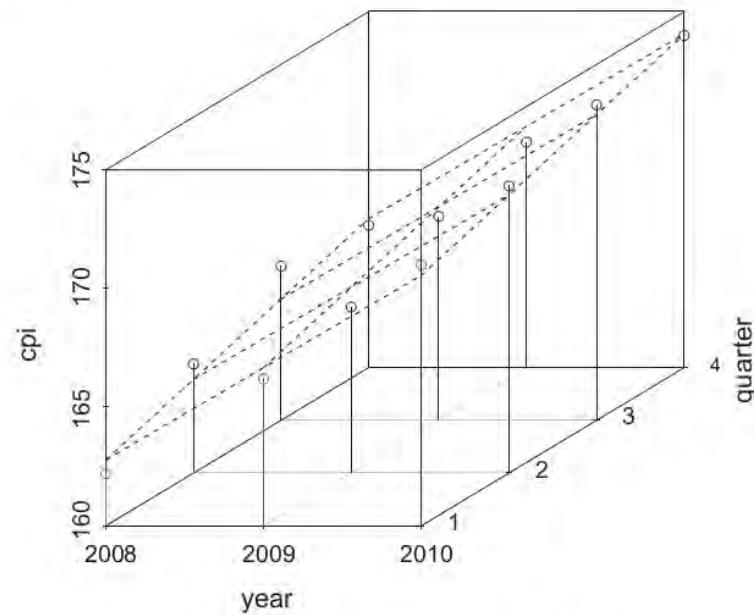


Figure 5.3 A 3D plot of the fitted model.

9. นำโมเดลมาใช้พยากรณ์ข้อมูลในอนาคต คือ CPI คร. 2011

```
> data2011 <- data.frame(year=2011, quarter=1:4)

> cpi2011 <- predict(fit, newdata=data2011)

> style <- c(rep(1,12), rep(2,4))

> plot(c(cpi, cpi2011), xaxt="n", ylab="CPI", xlab="",
+       pch = style, col = style)

> axis(1, at=1:16, las=3,
+       labels=c(paste(year,quarter,sep="Q"), "2011Q1", "2011Q2",
+                 "2011Q3", "2011Q4"))
```

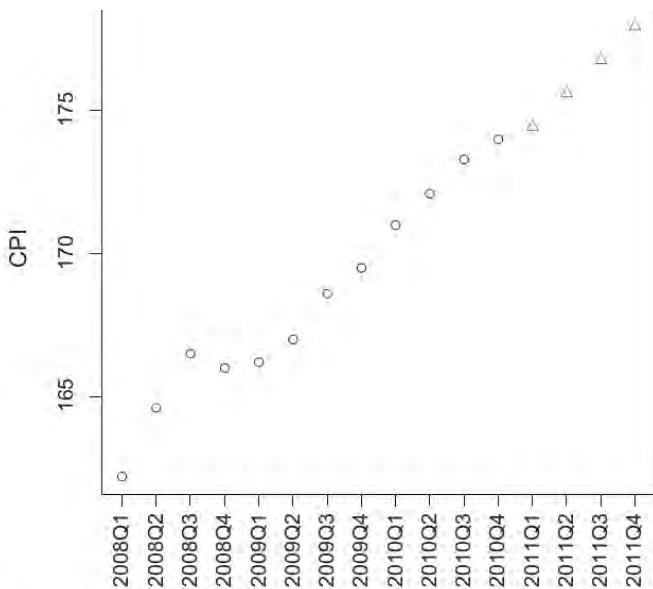


Figure 5.4 Prediction of CPIs in 2011 with linear regression model.

5.2 สมการถดถอยชนิดโลจิสติก (Logistic Regression)

ใช้เพื่อพยากรณ์ความน่าจะเป็นของการเกิดเหตุการณ์โดยนำสมการ logistic มาใช้แทนสมการเชิงเส้น (ของ linear regression) รูปแบบของ Logistic Regression คือ

$$\text{logit}(y) = c_0 + c_1 x_1 + c_2 x_2 + \dots + c_k x_k$$

$$\text{logit}(y) = \ln(y/(1-y))$$

ดังนั้นสามารถเปลี่ยนสมการใหม่ได้เป็น

$$y = \frac{1}{1 + e^{-(c_0 + c_1 x_1 + c_2 x_2 + \dots + c_k x_k)}}.$$

`glm()` ใช้สร้าง logistic regression โดยกำหนดให้ family มีค่า binomial (link = “logit”)

5.3 สมการถดถอยเชิงเส้นรูปแบบทั่วไป (Generalized Linear Regression, GLM)

GLM คือ โมเดลของสมการถดถอยเชิงเส้นในรูปแบบทั่วไปที่สามารถเฉพาะเจาะจงโดยละเอียดโดยการกำหนด link function เพื่อบอกว่าเป็นชนิดอะไร (สามารถระบุขนาดของความแปรปรวนได้) เช่น ชนิดเชิงเส้น, ชนิด logistic, ชนิด Poission

และสามารถกำหนดการกระจายข้อผิดพลาดได้ (error distribution)

```

> data("bodyfat", package = "mboost")

> myFormula <- DEXfat ~age + waistcirc + hipcirc + elbowbreadth +
kneebreadth

> bodyfat.glm <- glm(myFormula, family = gaussian("log"),
data = bodyfat)

```

```
> summary(bodyfat.glm)
```

Call:

```
glm(formula = myFormula, family = gaussian("log"), data = bodyfat)
```

Deviance Residuals:

Min	1Q	Median	3Q	Max
-11.5688	-3.0065	0.1266	2.8310	10.0966

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	0.734293	0.308949	2.377	0.02042 *
age	0.002129	0.001446	1.473	0.14560
waistcirc	0.010489	0.002479	4.231	7.44e-05 ***
hipcirc	0.009702	0.003231	3.003	0.00379 **
elbowbreadth	0.002355	0.045686	0.052	0.95905
kneebreadth	0.063188	0.028193	2.241	0.02843 *

—

Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for gaussian family taken to be 20.31433)

Null deviance: 8536.0 on 70 degrees of freedom

Residual deviance: 1320.4 on 65 degrees of freedom

AIC: 423.02

Number of Fisher Scoring iterations: 5

```
> pred <- predict(bodyfat.glm, type="response")
```

การกำหนด type คือชนิดของการ predict ค่า default คือ เชิงเส้น, ค่า response คือการระบุว่าเป็น response variable

```
> plot(bodyfat$DEXfat, pred, xlab="Observed Values",
      ylab="Predicted Values")
> abline(a=0, b=1)
```

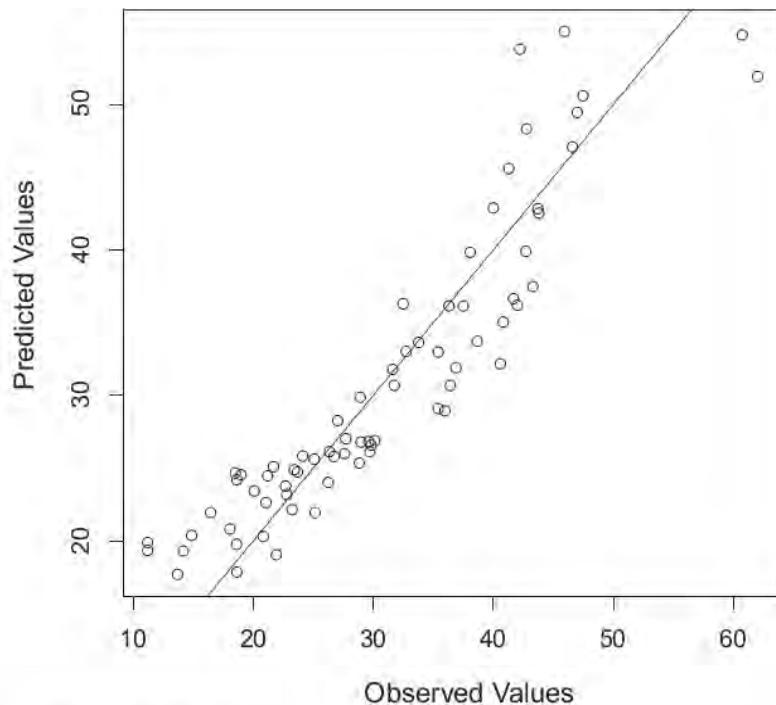


Figure 5.5 Prediction with generalized linear regression model.

ถ้า family = gaussian("identity") จะได้โมเดลที่คล้ายกับ linear และเราสามารถระบุ family = binomial("logit") เพื่อระบุว่าเป็น logistic regression

5.4 สมการถดถอยไม่เชิงเส้น (Non-Linear Regression)

ในขณะที่สมการถดถอยเชิงเส้นจะพิจารณาสมการที่ไกดีข้อมูลทุกจุด ได้ดีที่สุดในแนวเส้นตรงแต่ Non-Linear Regression จะพิจารณาสมการเส้นโดยเพื่อให้เข้าใกล้ข้อมูลทุกจุด โดยจะใช้ฟังก์ชัน nls() วิธีคูณกำรขีบาย การใช้งานฟังก์ชันนี้ทำได้โดยการพิมพ์ ?nls ที่ command line ของ R

บทที่ 6 โนเมเดลการจัดกลุ่ม (Clustering)

Clustering คือชนิดของโมเดลเพื่อใช้อธิบายลักษณะข้อมูลที่คล้ายกันให้อยู่กลุ่มเดียวกันมีหลายชนิด เช่น

k-means clustering, k-medoids clustering, hierarchical clustering และ density based clustering

6.1 k-Means Clustering

ข้อมูลที่ใช้ทดสอบคือ ข้อมูลของไม้ iris มีขั้นตอนการทดลองดังนี้

1. ลบคอลัมน์ species (ระบุ class) ออกจากตาราง (เพราะเป็น data mining ชนิด unsupervised)
 2. ใช้ฟังก์ชัน kmeans() และเก็บ โมเดลที่ถูกสร้างไว้ใน kmeans.result

```
> iris2 <- iris  
> iris2$Species <- NULL  
> (kmeans.result <- kmeans(iris2, 3))  
  
K-means clustering with 3 clusters of sizes 38, 50, 62
```

Cluster means:

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
1	6.850000	3.073684	5.742105	2.071053
2	5.006000	3.428000	1.462000	0.246000
3	5.901613	2.748387	4.393548	1.433871

Clustering vector:

Within cluster sum of squares by cluster:

```
[1] 23.87947 15.15100 39.82097
```

(between SS / total SS = 88.4%)

Available components:

```
[1] "cluster"      "centers"      "totss"       "withinss"     "tot.withinss"  
[6] "betweenss"    "size"
```

3. ต้องการทดสอบว่าผลการจัด cluster ถูกต้องมากน้อยเพียงใด โดยการเปรียบเทียบกับ Species จริง ๆ (ที่ถูกกลบออกไปที่ขั้น 1)

```
> table(iris$Species, kmeans.result$cluster)
```

	1	2	3
setosa	0	50	0
versicolor	2	0	48
virginica	36	0	14

ผลลัพธ์แสดงให้เห็นว่ากลุ่ม stosa สามารถแยกออกจากชั้นเจนจากกลุ่มอื่น ๆ แต่ versicolor และ virginica มีส่วนที่ซ้อนทับกันอยู่ (แยกจากกันไม่ชัดเจน)

4. แสดงการจัดกลุ่มของมาเป็นกราฟ 2 มิติ (จริง ๆ ข้อมูลมี 4 มิติ = 4 คอลัมน์) จึงให้แสดงเพียง 2 คอลัมน์แรกในลักษณะแกน x, y ผลที่ได้จากการทำงานอาจจะแตกต่างกันในแต่ละครั้ง เพราะการสุ่มจะได้ค่าเปลี่ยนไป

```
> plot(iris2[c("Sepal.Length", "Sepal.Width")],  
       col = kmeans.result$cluster  
  
> # plot cluster centers  
  
> points(kmeans.result$centers[, c("Sepal.Length",  
      "Sepal.Width")], col=1:3, pch=8, cex=2)
```

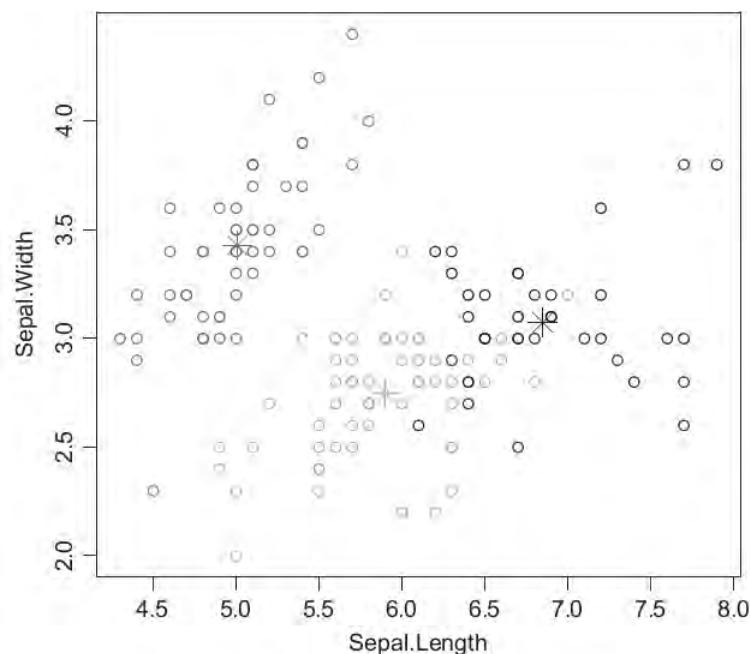


Figure 6.1 Results of k -means clustering.

โดยค่า * คือค่าจุดกึ่งกลางของแต่ละกลุ่ม (3 กลุ่ม)

6.2 k-Medoids Clustering

คล้ายกับการจัดกลุ่ม k-Means สิ่งที่แตกต่างกันคือจุดกึ่งกลางกลุ่มจะต้องมีอยู่จริงภายในกลุ่มนั้น k-Medoids จะใช้ได้กับกรณีที่มี outlier และ noise อัลกอริทึมที่นิยมใช้คือ PAM (Partitoning Around Medoids) แต่ไม่เหมาะสมกับข้อมูลขนาดใหญ่

เทคนิค CLARA พัฒนาขึ้นมาจากการ PAM โดยการใช้ข้อมูลสุ่ม (หลาย ๆ กลุ่ม) ส่งเข้าไปให้ PAM ทำงาน และเลือกโมเดลที่ดีที่สุดเป็นผลลัพธ์ ซึ่งจะทำให้ได้ผลดีกับข้อมูลขนาดใหญ่ ใน package cluster จะมีฟังก์ชัน pam() และ clara() ซึ่งต้องกำหนดค่า k แต่ถ้าต้องการให้หาค่า k อัตโนมัติจะต้องใช้ฟังก์ชัน pamk() ซึ่งอยู่ใน package fpc โดยจะเรียก pam(), clara() และหาค่า k ที่เหมาะสมจากค่า silhouette ที่ดีที่สุด

```
> library(fpc)

> pamk.result <- pamk(iris2)

> # number of clusters

> pamk.result$nc

[1] 2

> # check clustering against actual species

> table(pamk.result$pamobject$clustering, iris$Species)
    setosa  versicolor  virginica
1     50          1          0
2      0         49         50

> layout(matrix(c(1,2),1,2)) # 2 graphs per page
> plot(pamk.result$pamobject)

> layout(matrix(1)) # change back to one graph per page
```

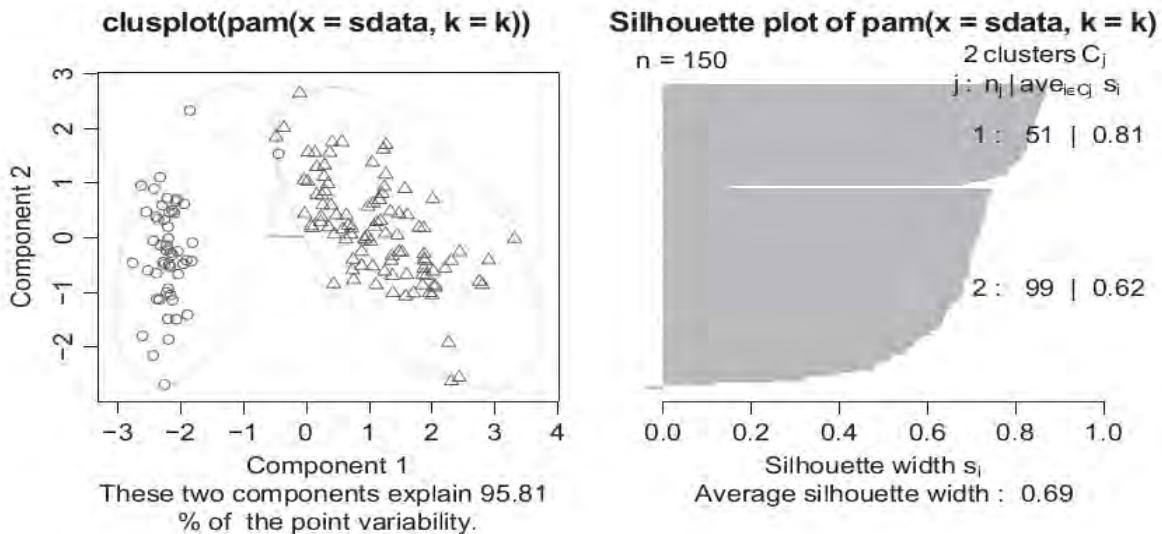


Figure 6.2 Clustering with the k -medoids algorithm—I.

จากผลลัพธ์พบว่า pamk() จัดกลุ่มให้ 2 กลุ่ม โดยกลุ่มแรกคือ setosa และกลุ่มที่สองจะเป็น versicolor รวมกับ virginica รูปทางซ้ายแสดงภาพ 2 มิติที่ได้จาก clusplot เส้นตรงแสดงระยะห่างระหว่าง cluster รูปทางขวาแสดง silhouette ก็สามารถตรวจสอบประสิทธิภาพการจัดกลุ่มว่าดีเพียงใด (ยิ่งมากยิ่งดี) กลุ่มที่ 1 จัดได้ เพราะค่า $s = 0.81$ แต่กลุ่มที่ 2 จัดได้ไม่ดีนัก เพราะ $s = 0.62$

ถ้าต้องการทดลองว่าบังคับให้จัดกลุ่มโดยมี $k = 3$ (3 กลุ่ม) ทำได้ดังนี้

```
> pam.result <- pam(iris2, 3)

> table(pam.result$clustering, iris$Species)

  setosa  versicolor  virginica
1    50        0        0
2     0       48       14
3     0        2       36

> layout(matrix(c(1,2),1,2)) # 2 graphs per page

> plot(pam.result)

> layout(matrix(1)) # change back to one graph per page
```

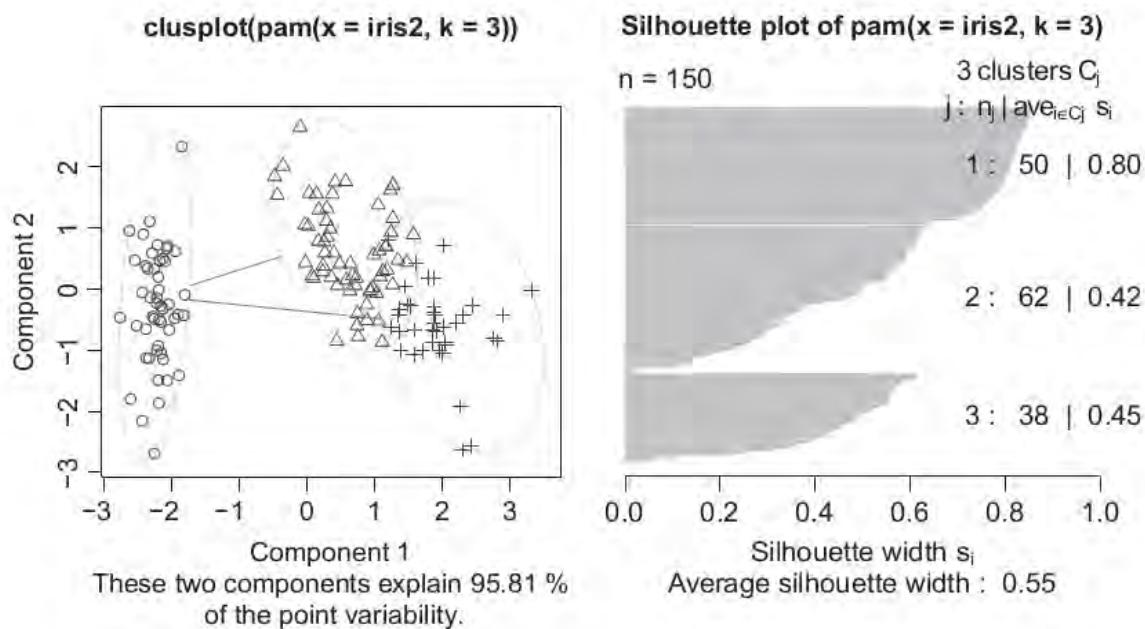


Figure 6.3 Clustering with the k -medoids algorithm—II.

ผลลัพธ์คือ

กลุ่ม 1 คือ กลุ่ม setosa มีการจัดกลุ่มที่ดี มีค่า $s = 0.80$

กลุ่ม 2 คือ กลุ่ม versicolor ผสมกับ virginica มีค่า $s = 0.42$

กลุ่ม 3 คือ กลุ่ม virginica ผสมกับ versicolor มีค่า $s = 0.45$

บางครั้งสังสัยว่าการกำหนดให้ $k = 3$ ดีกว่า $k = 2$ จริงหรือ (ค่า s ไม่ดีกว่า) ในที่นี้เรารู้จำนวน k จากข้อมูลตั้งต้นทำให้ลำเอียง โดยกำหนดค่า k ต้องเป็น 3 (แต่ถ้าดูจาก silhouette แล้ว $k = 2$ ดีที่สุด) ดังนั้นการกำหนดค่า k ที่ดีจะต้องขึ้นกับลักษณะข้อมูลและข้อเท็จจริงบางอย่างที่รู้ด้วยหน้ามาแล้ว

6.3 Hierarchical Clustering (การจัดกลุ่มแบบระดับชั้น)

เป็นการแบ่งกลุ่มแบบล่างขึ้นบนคือพิจารณาแต่ละตัวประกอบตัวใดแล้วจัดให้อยู่ด้วยกันจากนั้นจึงจัดกลุ่มที่คล้ายกันให้รวมกันเป็นกลุ่มใหญ่ขึ้นเรื่อยๆ โดยใช้ `hclust()`

```

> idx <- sample(1:dim(iris)[1], 40)
> irisSample <- iris[idx,]
> irisSample$Species <- NULL
> hc <- hclust(dist(irisSample), method="ave")
> plot(hc, hang = -1, labels=iris$Species[idx])
> # cut tree into 3 clusters
> rect.hclust(hc, k=3)
> groups <- cutree(hc, k=3)

```

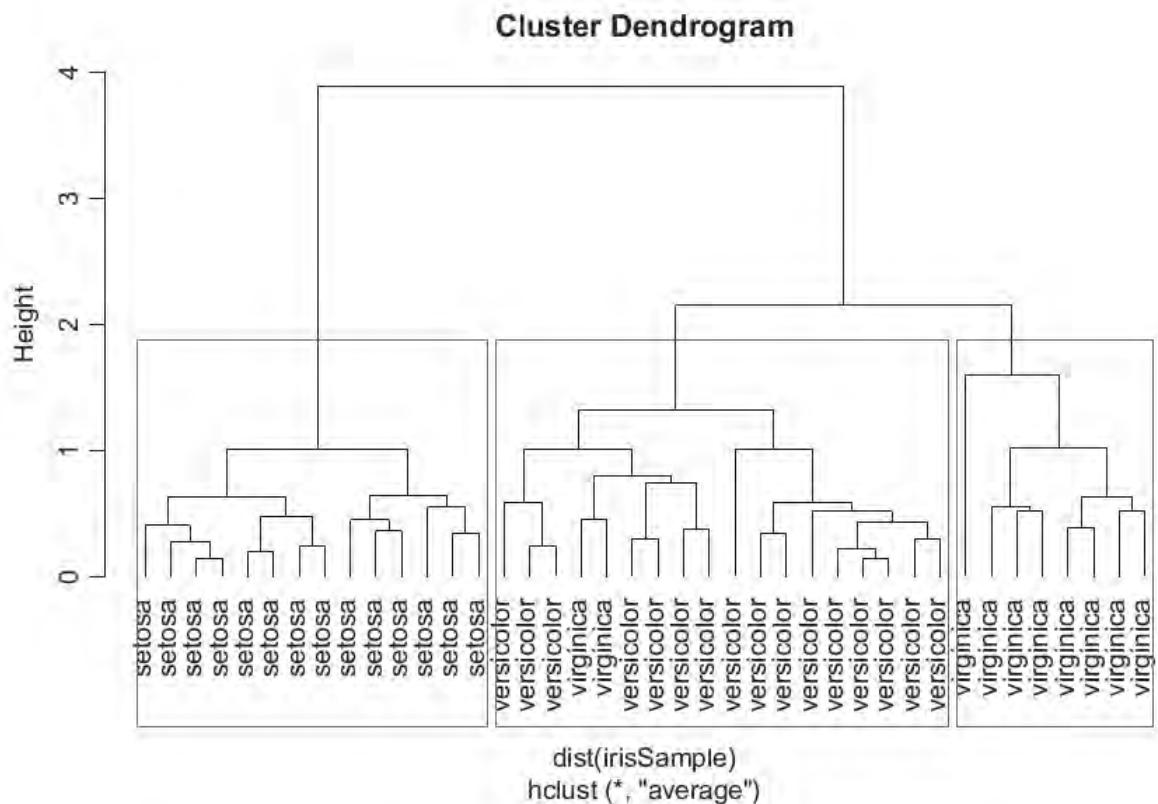


Figure 6.4 Cluster dendrogram.

ผลที่ได้คล้ายกับการจัดกลุ่มแบบ k-means คือ setosa จะถูกจัดกลุ่มออกจากอย่างชัดเจนในขณะที่ versicolor และ virginica ไม่สามารถแยกแยะได้ชัดเจน

6.4 Density-Based Clustering (การจัดกลุ่มด้วยความหนาแน่น)

เทคนิคที่นิยมใช้คือ DBSCAN ซึ่งมีอยู่ใน package fpc โดยทำกับข้อมูลชนิดตัวเลข (numeric) เท่านั้น หลักการคือ จัดกลุ่มตามความหนาแน่น ซึ่งกำหนดโดยพารามิเตอร์ 2 ตัวคือ

1. eps คือระยะห่างภายใน การจัดกลุ่ม (ห้ามเกินค่านี้)

2. MinPts คือจำนวนจุดที่น้อยที่สุด (ห้ามกลุ่มมีจุดน้อยกว่านี้)

ข้อดีของการจัดกลุ่มวิธีนี้คือ รูปร่างของกลุ่มมีหลากหลาย (ไม่จำเป็นต้องเป็นทรงกลม) และทนทานต่อ ข้อมูลที่มี ข้อมูลรบกวน (noise = ค่าผิดพลาด) และ ค่าที่แตกต่างออกไปมาก ๆ (outlier) ต่อไปนี้คือตัวอย่างการ จัดกลุ่มกับข้อมูล iris

```
> library(fpc)

> iris2 <- iris[-5] # remove class tags

> ds <- dbscan(iris2, eps=0.42, MinPts=5)

> # compare clusters with original class labels

> table(ds$cluster, iris$Species)
```

	setosa	versicolor	virginica
0	2	10	17
1	48	0	0
2	0	37	0
3	0	3	33

คำอธิบายตารางด้านบน 伟大 1-3 คือกลุ่มที่ได้ 伟大 0 คือ noise หรือ outlier ซึ่งคือข้อมูลที่ไม่ได้อยู่ใน กำหนดให้อยู่กลุ่มใด ๆ

```
> plot(ds, iris2)
```

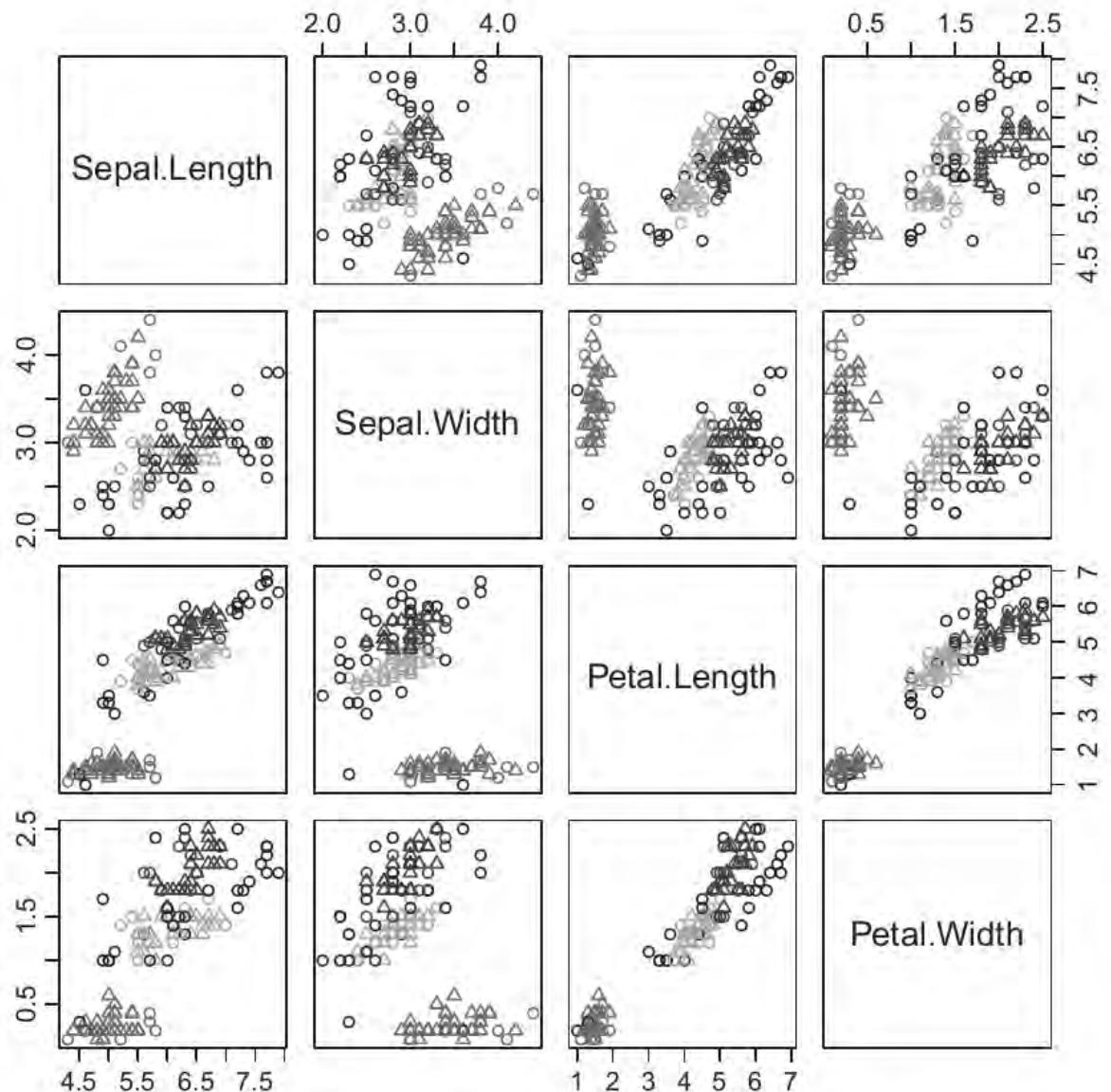


Figure 6.5 Density-based clustering—I.

เราสามารถใช้แสดงการกระจาย (scatter plot) จากข้อมูลคอลัมน์ที่ 1 และ 4 ดังนี้

```
> plot(ds, iris2[c(1, 4)])
```

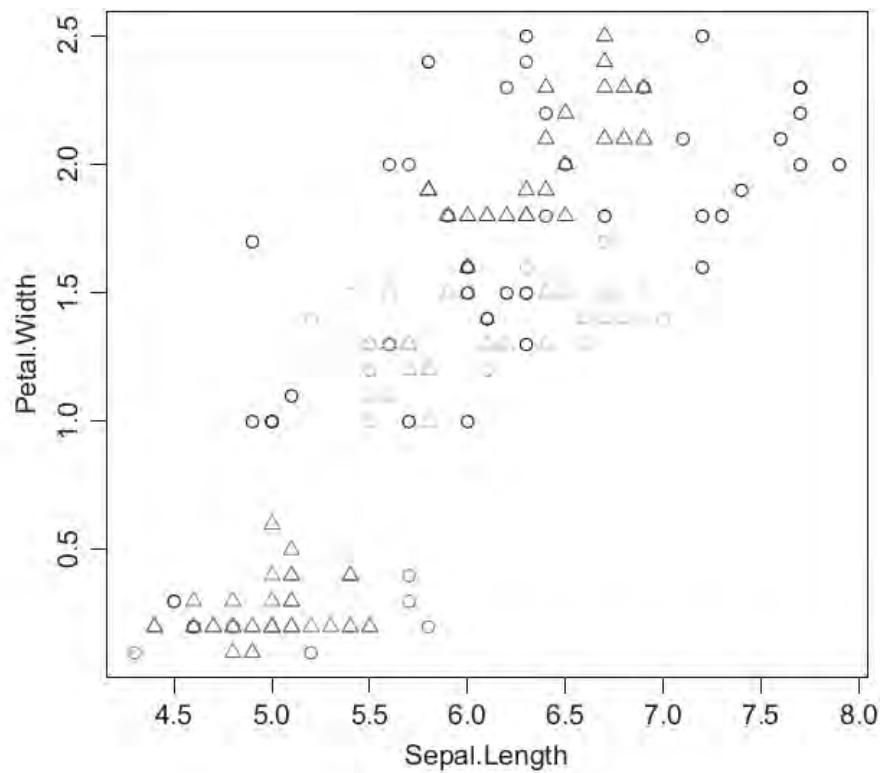


Figure 6.6 Density-based clustering—II.

เราสามารถใช้ `plotcluster()` จาก `fpc` เพื่อแสดงการจัดกลุ่มทั้งหมด

```
> plotcluster(iris2, ds$cluster)
```

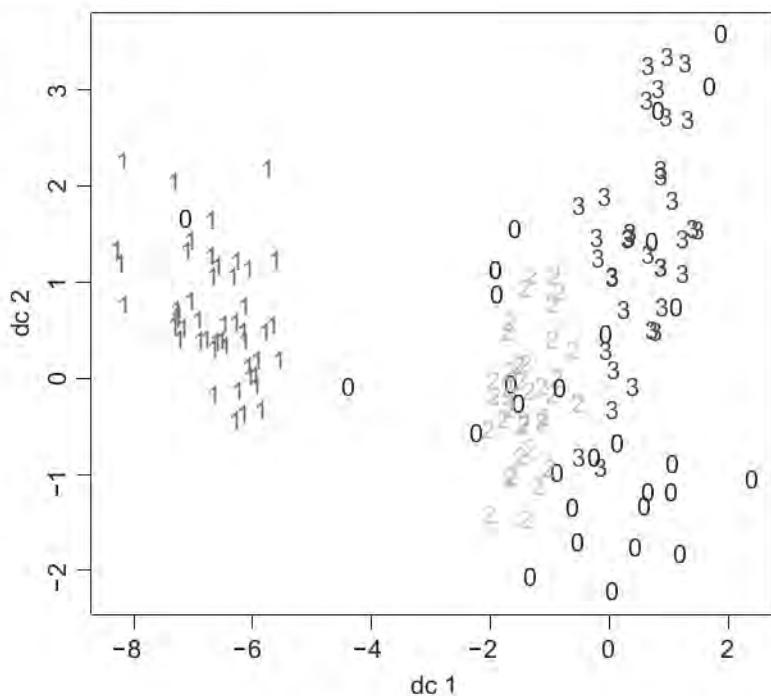


Figure 6.7 Density-based clustering—III.

เราสามารถนำโมเดลที่ได้จากการจัดกลุ่มมาใช้ใหม่เพื่อทำนายข้อมูลอื่น ๆ เช่น จากที่ผ่านมาเราได้โมเดล ds และต้องการนำมาจัดกลุ่มให้กับข้อมูลอื่น เช่น สู่นมา 10 ตัวจากข้อมูลเดิมแล้วดัดแปลงนิดหน่อยโดยการใส่ noise เข้าไป (ค่า noise ที่ใส่เข้าไปเป็นชนิด uniform distribution สร้างโดยใช้ runif())

(ดัดแปลงข้อมูลเพื่อต้องการนำข้อมูลใหม่อื่น ๆ มาใช้งาน)

```
> # create a new dataset for labeling
> set.seed(435)
> idx <- sample(1:nrow(iris), 10)
> newData <- iris[idx, -5]
> newData <- newData + matrix(runif(10*4, min=0, max=0.2),
nrow=10, ncol=4)
> # label new data
> myPred <- predict(ds, iris2, newData)
> # plot result
> plot(iris2[c(1,4)], col=1+ds$cluster)
> points(newData[c(1,4)], pch="ast", col=1+myPred, cex=3)
```

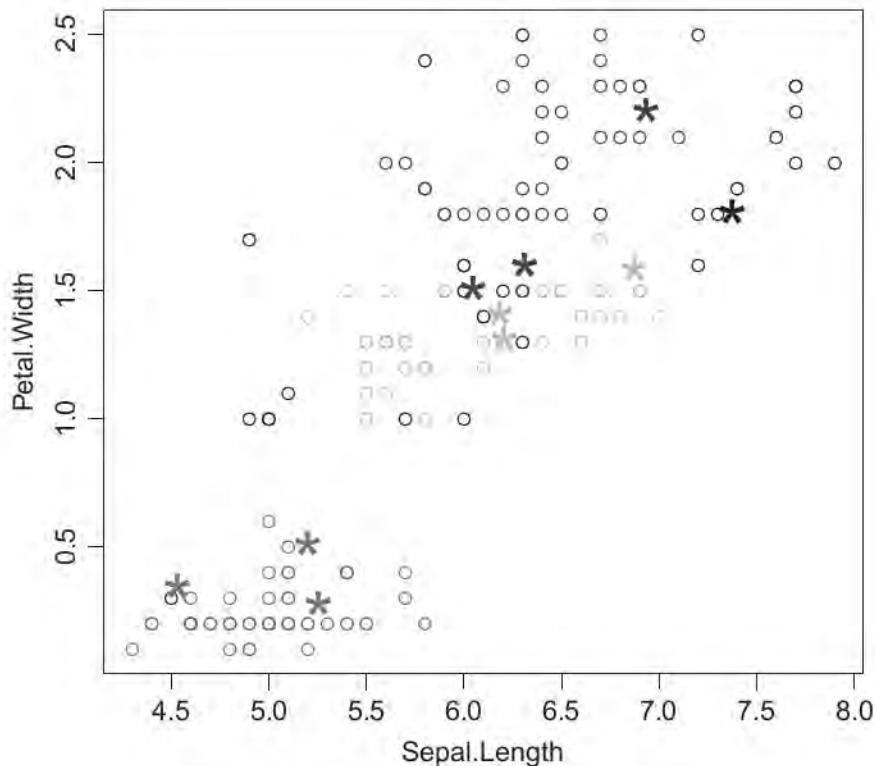


Figure 6.8 Prediction with clustering model.

```

> # check cluster labels
> table(myPred, iris$Species[idx])

myPred   setosa  versicolor  virginica
0          0         0          1
1          3         0          0
2          0         3          0
3          0         1          2

```

ผลที่ได้ข้อมูลอื่น 10 ตัวจะมี 8 ตัวถูกจัดกลุ่มที่ถูกต้อง (ข้อมูล 10 ตัวนี้คือ * ที่แสดงในกราฟ)

บทที่ 7 Outlier Detection

Outlier Detection คือการตรวจจับข้อมูลที่ต่างจากข้อมูลส่วนใหญ่มาก ๆ เราสามารถตรวจจับได้หลายวิธี ในบทนี้จะแสดงตัวอย่างการตรวจจับ outlier ที่มีข้อมูล 1 คอลัมน์ (univariate), การตรวจจับโดยวิธี LOF, โดยวิธี Clustering และโดยใช้ Time series

7.1 Univariate Outlier Detection

เป็นพื้นฐานในการพัฒนาไปสู่ multivariate outlier detection (ข้อมูลมีหลายคอลัมน์)

ฟังก์ชัน `boxplot.stats()` จะให้ค่าสถิติ และกราฟ `boxplot` โดยให้ค่า outlier อยู่ในตัวแปร `out` ซึ่งจะอยู่นอกช่วงออกไปจากกราฟอย่างชัดเจนแสดงเป็นวงกลม ถ้าอยากรู้รายละเอียดการใช้ฟังก์ชันให้พิมพ์ `?boxplot.stats`

```
> set.seed(3147)
```

```
> x <- rnorm(100)
```

```
> summary(x)
```

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
-3.3150	-0.4837	0.1867	0.1098	0.7120	2.6860

```
> # outliers
```

```
> boxplot.stats(x)$out
```

```
[1] -3.315391 2.685922 -3.055717 2.571203
```

```
> boxplot(x)
```

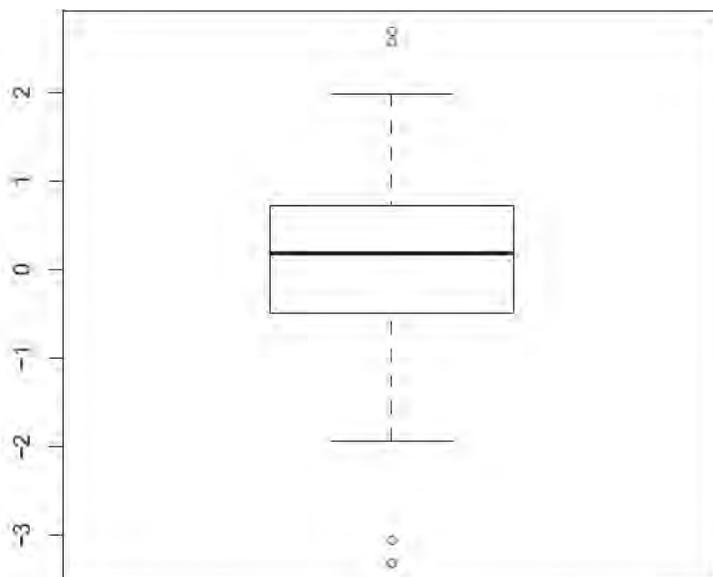


Figure 7.1 Univariate outlier detection with boxplot.

ในการหา multivariate outlier detection ทำคล้ายกัน โดยทดลองดังนี้

1. สร้าง df เป็นชนิด dataframe มี 2 คอลัมน์ คือ x, y โดยการสุ่มด้วยคำสั่ง rnorm()
2. หา outlier ทั้งแกน x และ y ด้วยระบุ \$out
3. แสดงข้อมูล outlier ด้วยเครื่องหมาย +

```
> y <- rnorm(100)

> df <- data.frame(x, y)

> rm(x, y)

> head(df)

      x          y
1 -3.31539150  0.7619774
2 -0.04765067 -0.6404403
3  0.69720806  0.7645655
4  0.35979073  0.3131930
5  0.18644193  0.1709528
6  0.27493834 -0.8441813

> attach(df)

> # find the index of outliers from x

> (a <- which(x %in% boxplot.stats(x)$out) )

[1] 1 33 64 74

> # find the index of outliers from y

> (b <- which(y %in% boxplot.stats(y)$out) )

[1] 24 25 49 64 74

> detach(df)

> # outliers in both x and y

> (outlier.list1 <- intersect(a,b) )

[1] 64 74
```

```

> plot(df)
> points(df[outlier.list1,], col="red", pch="+", cex=2.5)

```

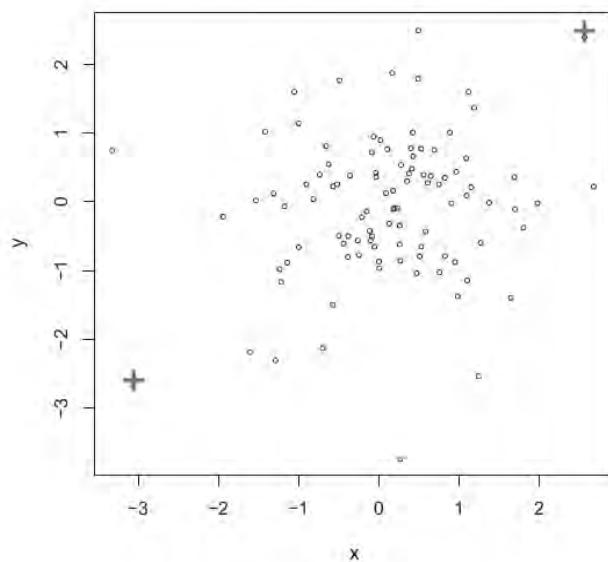


Figure 7.2 Outlier detection—I.

ตัวอย่างต่อไป หา outlier ที่แกน x หรือ y

```

> # outliers in either x or y
> (outlier.list2 <- union(a,b))
[1] 1 33 64 74 24 25 49
> plot(df)
> points(df[outlier.list2,], col="blue", pch="x", cex=2)

```

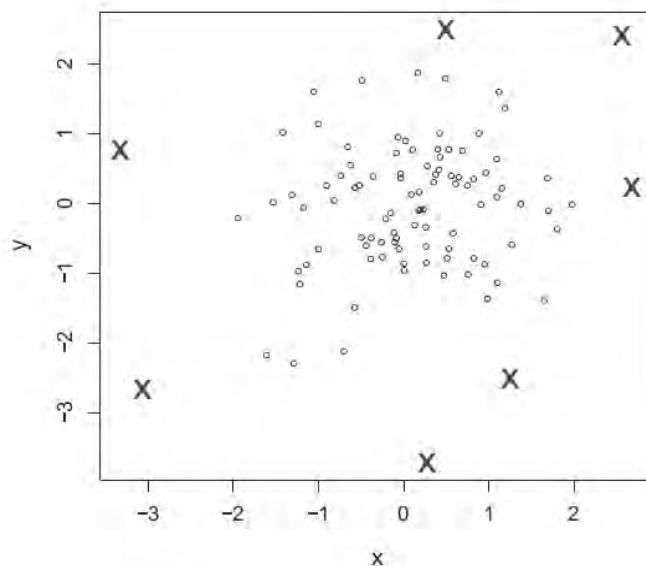


Figure 7.3 Outlier detection—II.

7.2 เทคนิค LOF

LOF (Local Outlier Factor) คือ อัลกอริทึมที่หา outlier โดยวัดจากความหนาแน่นซึ่งจะเปรียบเทียบกับข้อมูลข้างเคียงเมื่อถึงเกณฑ์ที่กำหนดจะรวมเข้าไว้เป็น outlier ข้อมูลที่ใช้จะต้องเป็นชนิดตัวเลขเท่านั้น ใน package DMwR และ dprep มีฟังก์ชัน lofactor() เพื่อใช้ทำ LOF เมื่อ k คือจำนวนข้อมูลข้างเคียงเพื่อใช้พิจารณาว่ากลุ่มนี้นั้นสมควรเป็น outlier หรือไม่ รูป 7.4 แสดงความหนาแน่นของ outlier score (ค่า k) ซึ่งเป็นแกน x

```
> library(DMwR)  
  
> # remove "Species", which is a categorical column  
  
> iris2 <- iris[,1:4]  
  
> outlier.scores <- lofactor(iris2, k=5)  
  
> plot(density(outlier.scores))
```

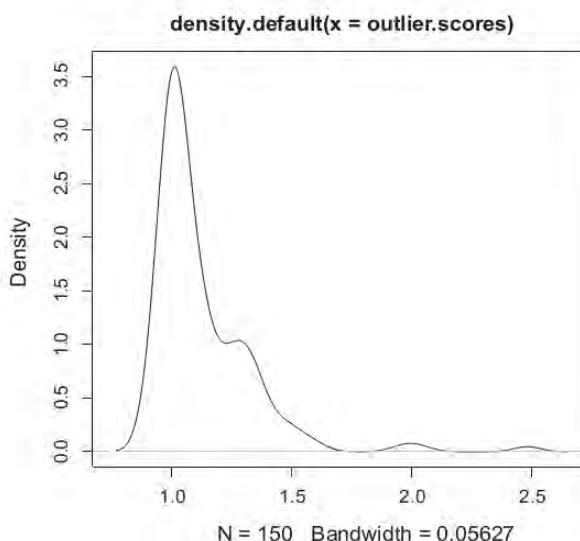


Figure 7.4 Density of outlier factors.

```
> # pick top 5 as outliers  
  
> outliers <- order(outlier.scores, decreasing=T)[1:5]  
  
> # who are outliers  
  
> print(outliers)  
  
[1] 42 107 23 110 63  
  
> print(iris2[outliers,])  
  
  Sepal.Length Sepal.Width Petal.Length Petal.Width  
42       4.5      2.3       1.3        0.3  
107      4.9      2.5       4.5        1.7  
23       4.6      3.6       1.0        0.2  
110      7.2      3.6       6.1        2.5  
63       6.0      2.2       4.0        1.0
```

ต่อไปนี้เป็นการแสดง outlier ด้วย biplot ขององค์ประกอบหลัก 2 ตัวที่สำคัญ (first 2 principle components)

```
> n <- nrow(iris2)
> labels <- 1:n
> labels[-outliers] <- "", ""
> biplot(prcomp(iris2), cex=.8, xlabs=labels)
```

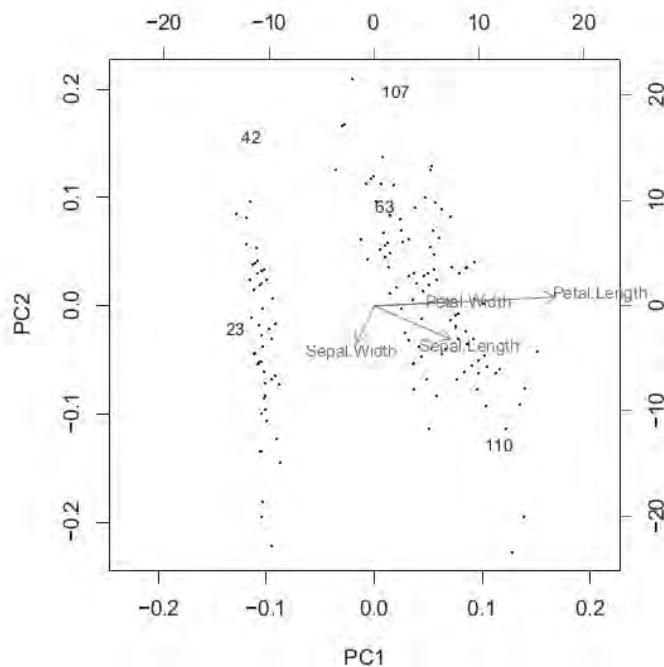


Figure 7.5 Outliers in a biplot of first two principal components.

จากชุดคำสั่งที่ผ่านมาฟังก์ชัน prcomp() จะทำการหา principle components และ biplot แสดงกราฟของ 2 ตัวแรกที่ (2 คอลัมน์ = 2 components = แกน x, y) เส้นที่มีลูกศรแสดงข้อมูลดังเดิมว่ามี 4 คอลัมน์ ค่า outlier ที่คืนพบมี 5 ตัว แสดงด้วยหมายเลขและ

เราสามารถแสดง outlier ด้วย pair plot โดยกำกับด้วย +

```
> pch <- rep(".", n)
> pch[outliers] <- "+"
> col <- rep("black", n)
> col[outliers] <- "red"
> pairs(iris2, pch=pch, col=col)
```

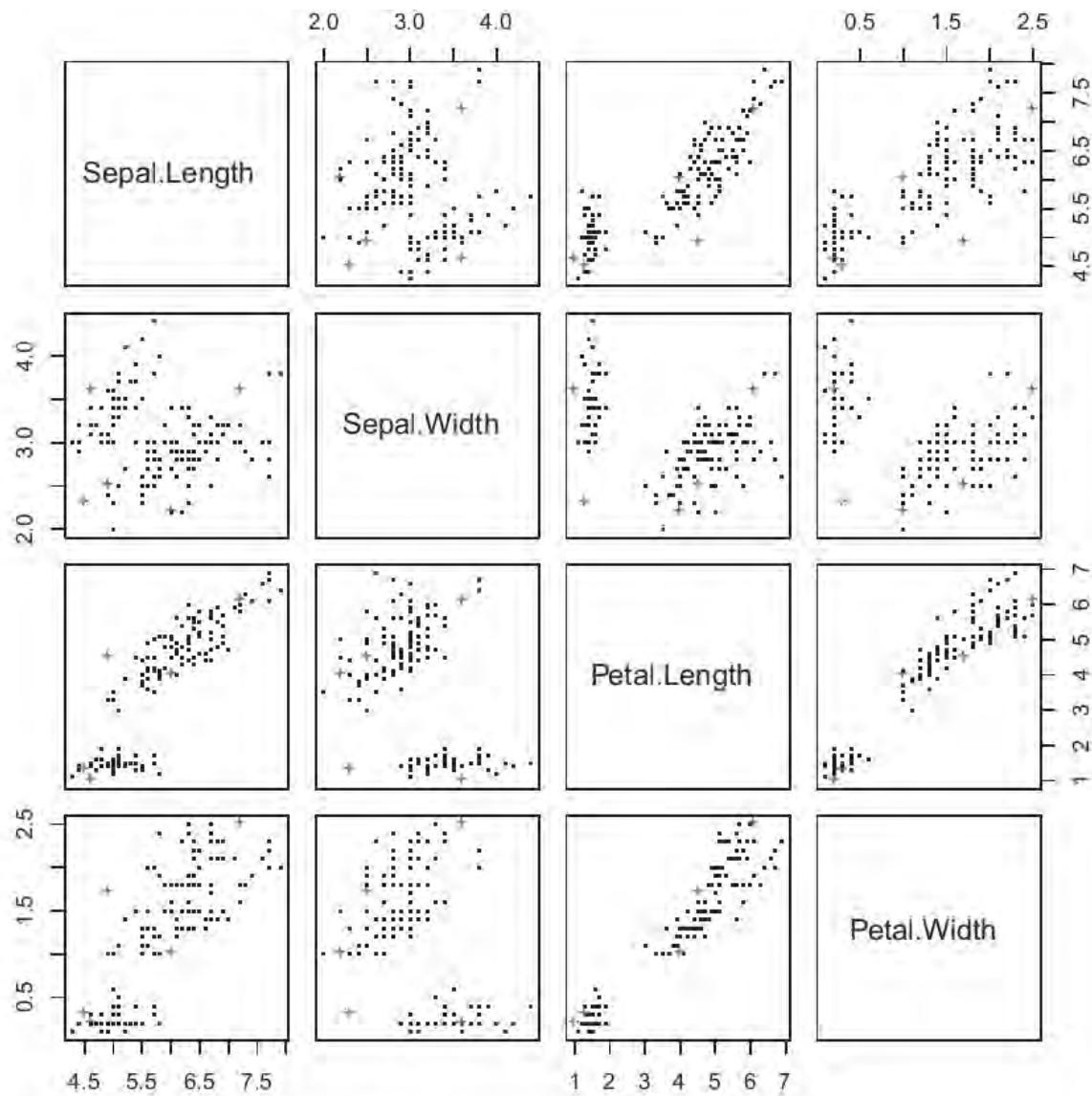


Figure 7.6 Outliers in a matrix of scatter plots.

ใน package Rof จะมี `lof()` เพื่อใช้ทำ LOF ในลักษณะขนาน (parallel computing) จะมีการกำหนดพารามิเตอร์มากขึ้น เช่น ค่า k , ماتริกซ์ระยะทาง (distance matrix), ชุดคำสั่งต่อไปนี้แสดงการใช้ `lof()` เมื่อพบ outlier และจะมีการเลือกที่ดีที่สุดอันดับต้น ๆ

```
> library(Rlof)

> outlier.scores <- lof(iris2, k=5)

> # try with different number of neighbors (k = 5, 6, 7, 8, 9 and 10)

> outlier.scores <- lof(iris2, k=c(5:10))
```

7.3 Outlier Detection by Clustering

เราสามารถหาข้อมูลที่แตกต่างจากส่วนใหญ่โดยการทำ Clustering แนวคิดคือถ้าข้อมูลตัวใดไม่ถูกกำหนดให้อยู่ในกลุ่ม (ส่วนใหญ่) ข้อมูลนั้นจะเป็น outlier การจัดกลุ่มที่นำมาใช้เป็นวิธีทั่วๆไป เช่น DBSCAN, k-means, k-medoid วิธีการคือเมื่อจัดกลุ่มแล้วจะหาว่าข้อมูลตัวไหนอยู่ห่างจากจุดกึ่งกลางมาก ๆ ก็จะจัดให้ข้อมูลนั้นเป็น outlier (วิธี DBSCAN คือดูกลุ่มหรือข้อมูลเดียวที่ห่างจากกลุ่มใหญ่มาก ๆ)

```

> outliers <- order(distances, decreasing=T)[1:5]
> # who are outliers
> print(outliers)
[1] 99 58 94 61 119
> print(iris2[outliers,])

  Sepal.Length Sepal.Width Petal.Length Petal.Width
99      5.1        2.5       3.0         1.1
58      4.9        2.4       3.3         1.0
94      5.0        2.3       3.3         1.0
61      5.0        2.0       3.5         1.0
119     7.7        2.6       6.9         2.3

> # plot clusters
> plot(iris2[,c("Sepal.Length", "Sepal.Width")], pch="o",
+       col=kmeans.result$cluster, cex=0.3)
> # plot cluster centers
> points(kmeans.result$centers[,c("Sepal.Length",
+ "Sepal.Width")], col=1:3, pch=8, cex=1.5)

```

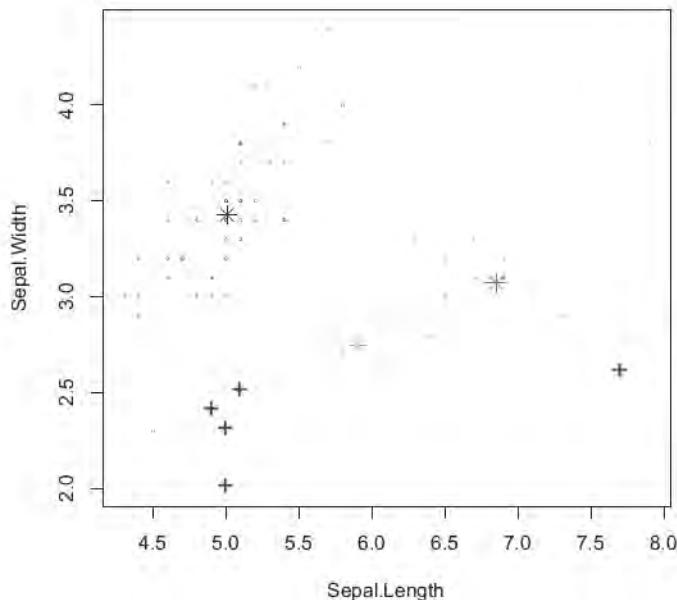


Figure 7.7 Outliers with k -means clustering.

```

> # plot outliers
> points(iris2[outliers, c("Sepal.Length", "Sepal.Width")],
+ pch="+", col=4, cex=1.5)

```

7.4 Outlier Detection from Time Series

Time series คือ ข้อมูลอนุกรมเวลา หมายถึง ข้อมูลที่มีเวลากำกับไว้โดยจะแสดงเรียงตามเวลา และ 1 รุคคอร์ด คือ 1 ชุด ซึ่งมักจะมีหลายคอลัมน์ (ตามเวลาที่บันทึก) การนำไปใช้งานคือเพื่อพยากรณ์ว่าในอนาคตจะเป็นอย่างไร วิธีการหา outlier detection คือ

1. หาสัมประสิทธิ์ (ค่าคงที่) ของสมการที่แทนค่า time series นั้นโดยใช้ `stl()` ซึ่งจะแยกออกเป็นองค์ประกอบต่าง ๆ เช่น seasonal, trend
2. วิเคราะห์หา outlier

```
> # use robust fitting  
  
> f <- stl(AirPassengers, "periodic", robust=TRUE)  
  
> (outliers <- which(f$weights<1e-8))  
  
[1] 79 91 92 102 103 104 114 115 116 126 127 128 138 139 140  
  
> # set layout  
  
> op <- par(mar=c(0, 4, 0, 3), oma=c(5, 0, 4, 0), mfcoll=c(4, 1))  
  
> plot(f, set.pars=NULL)  
  
> sts <- f$time.series  
  
> # plot outliers  
  
> points(time(sts)[outliers], 0.8*sts[, "remainder"] [outliers],  
  pch="x", col="red")  
  
> par(op) # reset layout
```

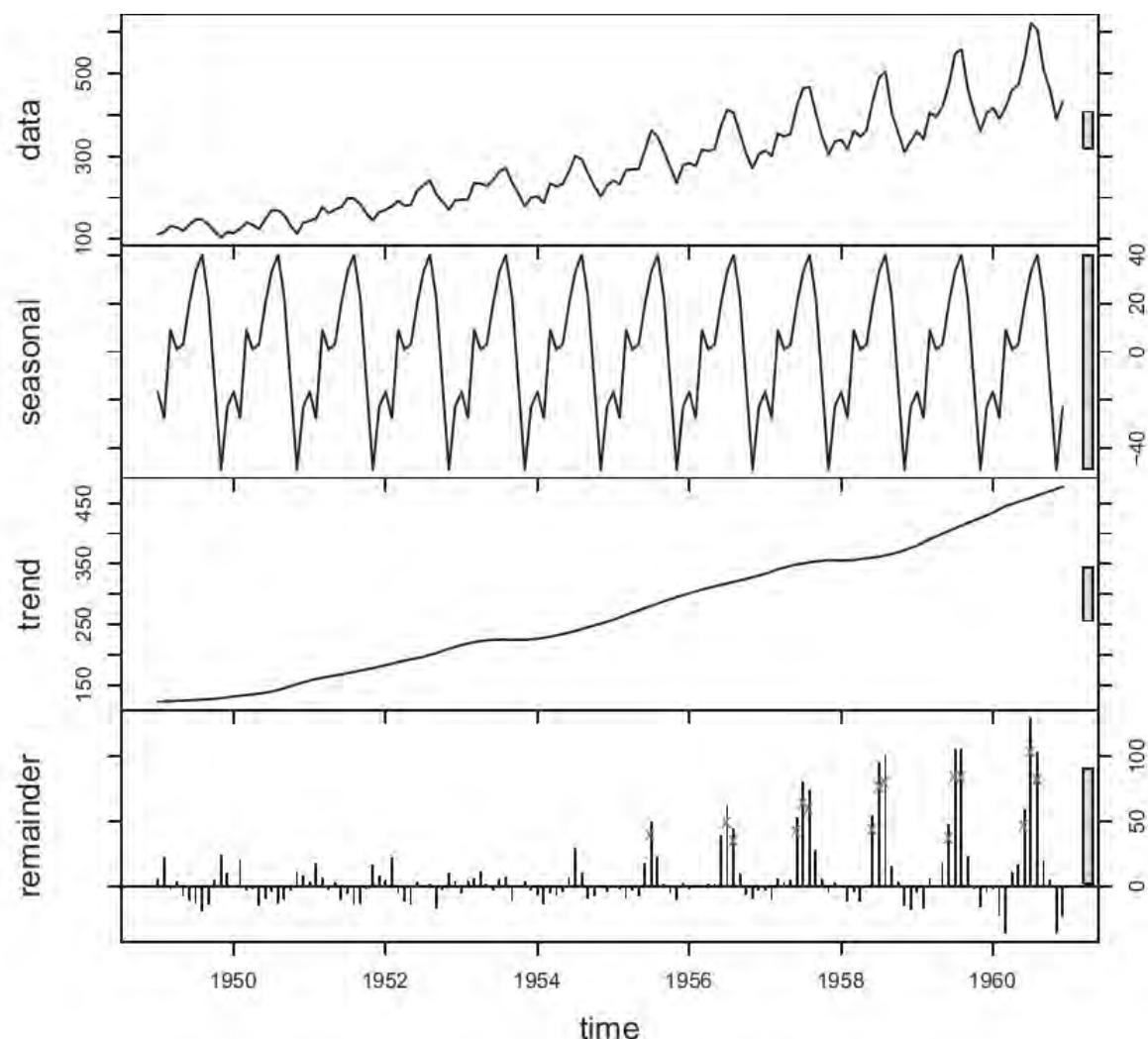


Figure 7.8 Outliers in time series data.

ค่า outlier คือ x (สีแดง)

7.5 บทสรุป

อัลกอริทึม LOF เหมาะสำหรับการหา outlier เนพาะภายในขอบเขต (local) และทำงานกับข้อมูลตัวเลขเท่านั้น package Rlof ทำงานบน processor คลายตัวในลักษณะการคำนวณในแบบบานาน (อาจจะทำไม่ได้บน window) อัลกอริทึม AVF (Attribute Value Frequency) สามารถทำงานกับข้อมูล categorical และสามารถทำงานได้กับข้อมูลขนาดใหญ่ได้อย่างรวดเร็ว

package อื่นของภาษา R ที่สามารถทำงานกับ outlier คือ

1. extremevalues : univariate outlier detection
2. mvoutlier : multivariate outlier detection
3. outliers : ทำกับ outlier ทั่วไป

บทที่ 8 การวิเคราะห์อนุกรมเวลา (Time Series and Data Mining)

ในบทนี้จะอธิบายและยกตัวอย่างเกี่ยวกับการแยกองค์ประกอบอนุกรมเวลา (time series decomposition) การพยากรณ์ (forecasting) การจัดกลุ่ม และการจำแนก เนื้อหาจะอธิบายเกี่ยวกับพื้นฐานของ time series ตัวอย่างการแยกรายละเอียดอนุกรมเวลาออกเป็นแนวโน้ม (trend) ฤดูกาล (seasonal) และเชิงสุ่ม (random composition) นอกจากนี้ยังอธิบายถึงโมเดล ARIMA ใช้ในการพิจารณาค่าเฉลี่ยเป็นช่วง ๆ และ DTW (Dynamic Time Warping) ใช้ในการหาความเหมือนของอนุกรมเวลา

8.1 Time series ในภาษา R

คลาส ts แทนข้อมูลที่ถูกสุ่มในช่วงเวลาเท่า ๆ กัน โดยมี 7 ช่วงแทนสัปดาห์, 12 ช่วงแทนเดือน, และ 4 ช่วงแทนไตรมาส (quarterly series) ตัวอย่างต่อไปนี้แสดงการสร้าง time series โดยใช้ 30 ค่า Frequency = 12, start = c(2011,3)

```
> a <- ts(1:30, frequency=12, start=c(2011,3))
> print(a)

      Jan   Feb   Mar   Apr   May   Jun   Jul   Aug   Sep   Oct   Nov   Dec
2011          1     2     3     4     5     6     7     8     9     10
2012  11    12    13    14    15    16    17    18    19    20    21    22
2013  23    24    25    26    27    28    29    30

> str(a)
Time-Series [1:30] from 2011 to 2014: 1 2 3 4 5 6 7 8 9 10 ...
> attributes(a)

$tsp
[1] 2011.167 2013.583 12.000
$class
[1] "ts"
```

8.2 Time Series Decomposition

คือการวิเคราะห์อนุกรมเวลาออกเป็นส่วนประกอบย่อยคือ trend, seasonal, cyclical, irregular component (แนวโน้ม, ฤดูกาล, วัฏจักร, สิ่งผิดปกติหรือข้อผิดพลาด) ในบางครั้งเราระบุ irregular component ว่าเป็น residual (ข้อผิดพลาด)

ข้อมูล AirPassengers ใช้แสดงการวิเคราะห์อนุกรมเวลา ประกอบด้วยข้อมูลรายเดือนของผู้โดยสารสายการบิน ช่วงปี ก.ศ. 1949–1960 มีข้อมูลจำนวน $12 \times 12 = 144$ เรคคอร์ด

```
> plot(AirPassengers)
```

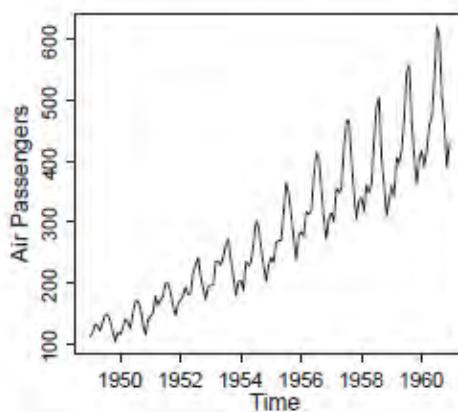


Figure 8.1 A time series of AirPassengers.

ฟังก์ชัน decompose() ใช้เพื่อแสดงองค์ประกอบต่าง ๆ

```
> # decompose time series
> apts <- ts(AirPassengers, frequency=12)
> f <- decompose(apts)
> # seasonal figures
> f$figure
[1] -24.748737 -36.188131 -2.241162 -8.036616 -4.506313 35.402778
[7] 63.830808 62.823232 16.520202 -20.642677 -53.593434 -28.619949

> plot(f$figure, type="b", xaxt="n", xlab="")
> # get names of 12 months in English words
> monthNames <- months(ISOdate(2011,1:12,1))
> # label x-axis with month names
> # las is set to 2 for vertical label orientation
```

จากรูป 8.3 ภาพแรกคือ ข้อมูลดั้งเดิมของ AirPassengers, ภาพที่สองคือ แสดง trend, ภาพที่สามแสดง seasonal, และภาพสุดท้ายคือ ข้อมูลที่เหลือซึ่งได้จากการตัด trend และ seasonal ออกไป

เกี่ยวกับการแยกองค์ประกอบใน package stats มีฟังก์ชัน stl(), ใน package timsac มีฟังก์ชัน decompose() และใน package ast มีฟังก์ชัน tsr()

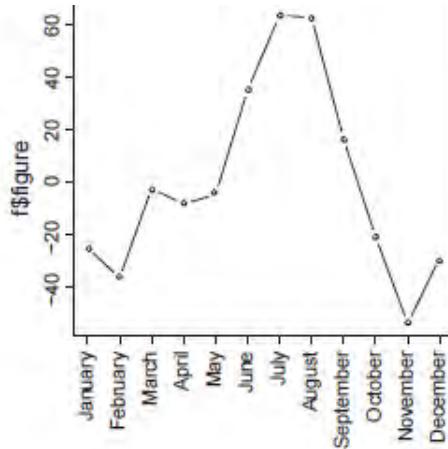


Figure 8.2 Seasonal component.

```
> plot(f)
```

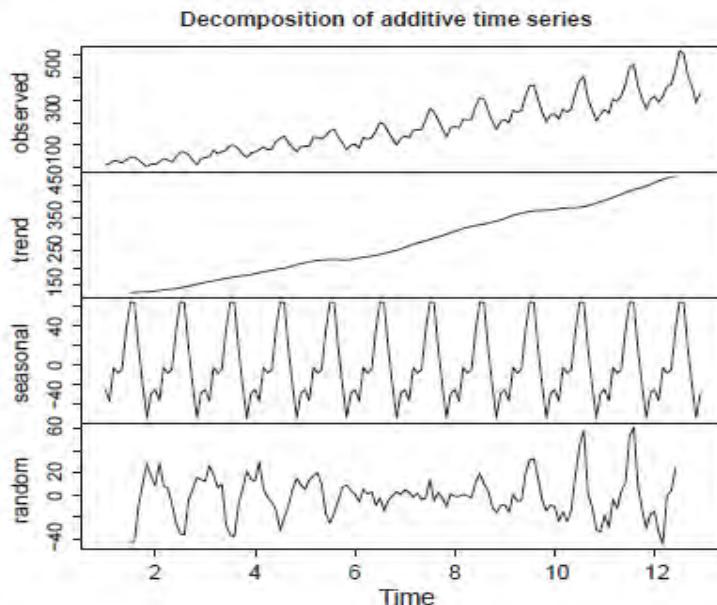


Figure 8.3 Time series decomposition.

8.3 การพยากรณ์ข้อมูล Time Series (Time series forecasting)

โดยการพิจารณาจากข้อมูลในอดีต เช่น การพยากรณ์ดัชนีหุ้นในตลาดหลักทรัพย์จากดัชนีในอดีต โดยเดลที่นิยมใช้มี 2 ชนิด คือ ARMA model และ ARIMA model
(autoregressive moving average, autoregressive integrated moving average)

ต่อไปนี้เป็นตัวอย่าง โดยเดลชนิด ARIMA ที่ได้จากการพยากรณ์เวลาแล้วนำไปพยากรณ์ได้

```
> fit <- arima(AirPassengers, order=c(1,0,0), list(order=c(2,1,0),
+ period=12))
> fore <- predict(fit, n.ahead=24)
> # error bounds at 95% confidence level
> U <- fore$pred +2*fore$se
> L <- fore$pred - 2*fore$se
> ts.plot(AirPassengers, fore$pred, U, L, col=c(1,2,4,4),
+ lty=c(1,1,2,2))
> legend("topleft", c("Actual", "Forecast",
+ "Error Bounds(95% Confidence)"), col=c(1,2,4), lty=c(1,1,2))
```

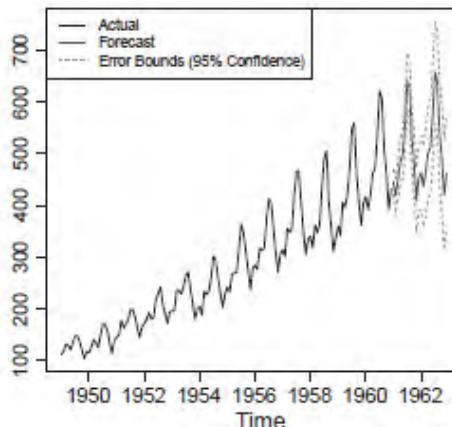


Figure 8.4 Time series forecast.

จากรูปข้างบนเส้นที่บันทึกแสดงค่าที่ได้จากการพยากรณ์และเส้นประคือค่าพิเศษเมื่อพิจารณาค่าความเชื่อมั่นที่ 95%

8.4 การจัดกลุ่มอนุกรมเวลา (Time series clustering)

เป็นการจัดกลุ่ม Time Series หลาย ๆ ตัวเข้าเป็นกลุ่มที่มีลักษณะเหมือนกันเข้าไว้ด้วยกัน การวัดความเหมือนมีหลายวิธี เช่น วิธี Euclidean, วิธี Manhattan, วิธี Maximum norm, วิธี Hamming, วิธีวัดมุนระหว่าง 2 เวกเตอร์ (inner product), และวิธี Dynamic Time warping (DTW)

8.4.1 การวัดความเหมือนกันด้วยวิธี DTW

เป็นการวัดว่าข้อมูล Time Series ทั้งสองเหมือนกันมากน้อยเพียงใด โดยใช้ฟังก์ชัน dtw() จาก package dtw โดยมีอีก 2 ฟังก์ชันที่หาระยะทาง (ความเหมือน) ที่ดีที่สุดคือ dtwDist() หรือ dist()

```
> library(dtw)
> idx <- seq(0, 2*pi, len=100)
> a <- sin(idx) + runif(100)/10
> b <- cos(idx)
> align <- dtw(a, b, step=asymmetricP1, keep=T)
> dtwPlotTwoWay(align)
```

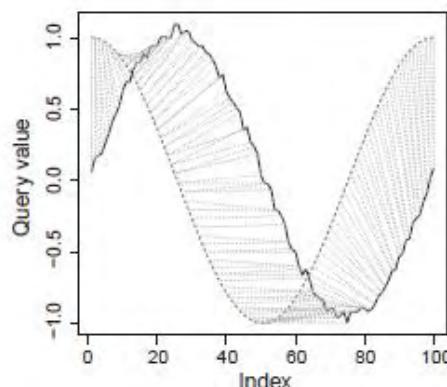


Figure 8.5 Alignment with dynamic time warping.

8.4.2 ข้อมูลอนุกรมเวลาที่สร้างขึ้นเพื่อใช้ศึกษา Time Series

ข้อมูลนี้เราสร้างขึ้นมาเองเพื่อใช้เป็นตัวตั้งต้นในการศึกษาโมเดลด้าน Time Series เราให้ชื่อข้อมูลนี้ว่า Synthetic Chart Time Series โดยมีทั้งหมด 60 เรคคอร์ด มี 600 คอลัมน์ประกอบด้วย 6 classes คือ

1. 1-100 : Normal
2. 101-200 : Cyclic
3. 201-300 : Increasing trend
4. 301-400 : Decreasing trend
5. 401-500 : Upward shift
6. 501-600 : Downward shift

```

> sc <- read.table("./data/synthetic_control.data", header=F,
+ sep="")
> # show one sample from each class
> idx <- c(1,101,201,301,401,501)
> sample1 <- t(sc[idx,])
> plot.ts(sample1, main="")

```

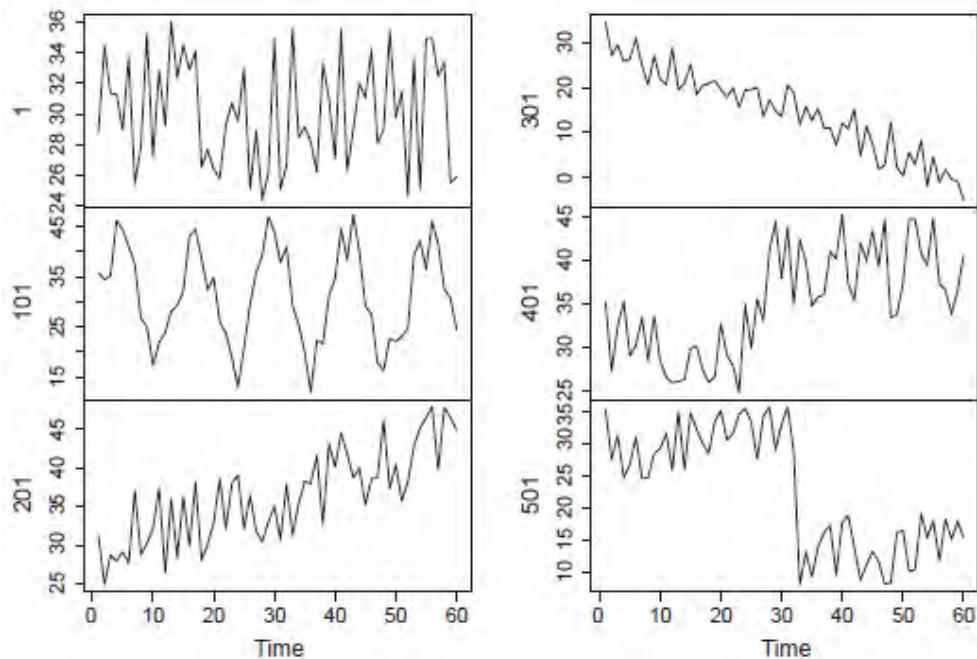
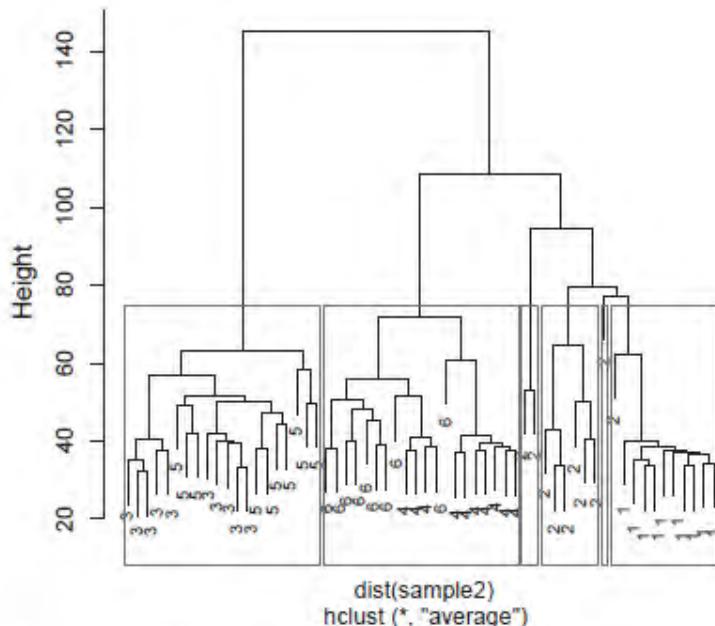


Figure 8.6 Six classes in synthetic control chart time series.

8.4.3 การจัดกลุ่มแบบระดับชั้นด้วยการวัดระยะยูคลิดีเดียน (Hierarchical Clustering with Euclidean Distance)

เริ่มต้นเราเลือกข้อมูล 10 เรคคอร์ดแบบสุ่มมาจากแต่ละ class เพื่อให้เห็นการจัดกลุ่มง่าย ๆ ถ้าใช้ข้อมูลทั้งหมดจะซับซ้อนเกินไป

```
> set.seed(6218)
> n <- 10
> s <- sample(1:100, n)
> idx <- c(s, 100+s, 200+s, 300+s, 400+s, 500+s)
> sample2 <- sc[idx, ]
> observedLabels <- rep(1:6, each=n)
> # hierarchical clustering with Euclidean distance
> hc <- hclust(dist(sample2), method="average")
> plot(hc, labels=observedLabels, main="")
> # cut tree to get 6 clusters
> rect.hclust(hc, k=6)
```



```

> memb <- cutree(hc, k=6)
> table(observedLabels, memb)

            memb
observedLabels 1 2 3 4 5 6
1              10 0 0 0 0 0
2              1 6 2 1 0 0
3              0 0 0 0 10 0
4              0 0 0 0 0 10
5              0 0 0 0 10 0
6              0 0 0 0 0 10

```

ผลลัพธ์ แสดงให้เห็นชัดเจนว่า Increasing trend (class 3) และ Upward shift (class 5) ไม่สามารถแยกออกจากกันได้ชัดเจนทำองเดียวกับ class 4 และ class 6 ที่ผสมรวมอยู่ด้วยกัน

8.4.4 การจัดกลุ่มแบบระดับชั้นด้วยการวัดระยะ DTW (Hierarchical Clustering with DTW Distance)

โดยใช้คำสั่งต่อไปนี้

```

> library(dtw)
> distMatrix <- dist(sample2, method="DTW")
> hc <- hclust(distMatrix, method="average")
> plot(hc, labels=observedLabels, main="")
> # cut tree to get 6 clusters
> rect.hclust(hc, k=6)

```

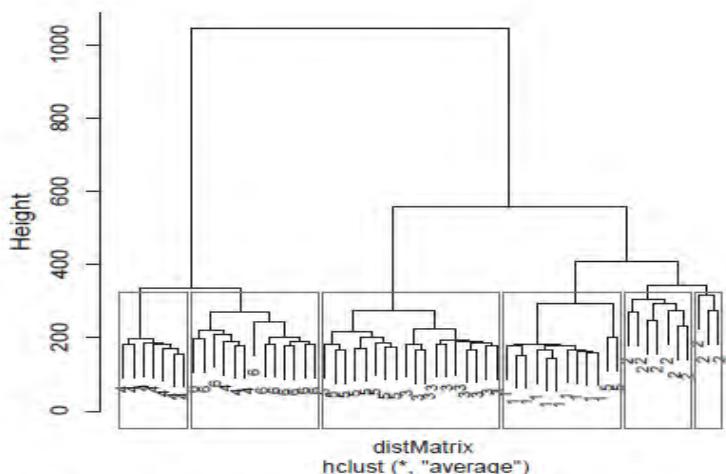


Figure 8.8 Hierarchical clustering with DTW distance.

```

> memb <- cutree(hc, k=6)

> table(observedLabels, memb)

      memb
observedLabels 1 2 3 4 5 6
1              10 0 0 0 0 0
2               0 7 3 0 0 0
3               0 0 0 10 0 0
4               0 0 0 0 7 3
5               2 0 0 8 0 0
6               0 0 0 0 0 10

```

เมื่อพิจารณาผลลัพธ์ที่ได้จากรูปที่ 8.8 และ 8.7 พบว่าวิธีวัดระยะแบบ DTW ดีกว่าวิธี Euclidean

8.5 การหาโมเดล classification จากข้อมูล Time Series (Time Series

Classification)

เป็นการสร้างโมเดลชนิด classification จากข้อมูล Time Series ที่มี class กำกับอยู่แล้ว เพื่อใช้พยากรณ์ข้อมูลใหม่ในอนาคตว่าจะเป็น class อะไร คุณลักษณะใหม่ ๆ ที่กรองออกมาจากข้อมูลอาจช่วยในการเพิ่มประสิทธิภาพโมเดลนั้น ๆ เทคนิคการกรองนี้คือ SVD (Single Value Decomposition), DFT (Discrete Fourier Transform), DWT (Discrete Wavelet Transform), PAA (Piecewise Aggregate Approximation), PIP (Perpetually Important Points, Piecewise Linear Representation, และ Symbolic Representation)

8.5.1 การหาโมเดล classification จากข้อมูลดิบ (classification with original data)

ใช้ฟังก์ชัน ctree() จาก package party โดยทำการแปลงชื่อ class ให้เป็นค่า categorical (ของเดิมเป็นจำนวนจริง) ก่อนส่งให้ ctree()

```
> classId <- rep(as.character(1:6), each=100)

> newSc <- data.frame(cbind(classId, sc))

> library(party)

> ct <- ctree(classId ~ ., data=newSc,
+               controls =ctree_control(minsplit=30, minbucket=10,
+                                         maxdepth=5))

> pClassId <- predict(ct)
> table(classId, pClassId)

    pClassId
classId 1   2   3   4   5   6
  1     97   0   0   0   0   3
  2      1   93   2   0   0   4
  3      0   0   96   0   4   0
  4      0   0   0   100  0   0
  5      4   0   10   0   86   0
  6      0   0   0   87   0   13

> # accuracy
> (sum(classId==pClassId)) / nrow(sc)
[1] 0.8083333

> plot(ct, ip_args=list(pval=FALSE), ep_args=list(digits=0))
```

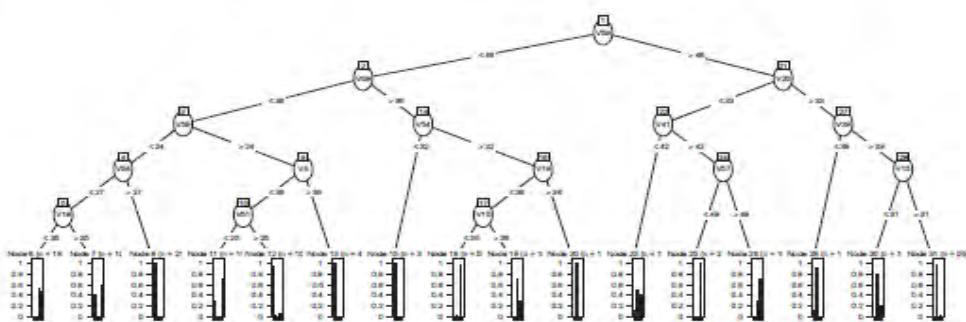


Figure 8.9 Decision tree.

8.5.2 การหาโมเดล classification จากการกรองข้อมูลดิบแล้ว (classification with extracted features)

เราจะใช้เทคนิค DWT เพื่อกรองข้อมูลดึงเดิมแล้วจึงนำไปสร้างโมเดล classification โดยเทคนิค wavelet ซึ่งมีการพิจารณาในรูปแบบ multi-resolution (ซึ่งมีเทคนิคที่คล้ายกันและได้รับความนิยมคือ DFT)

หลักการที่สำคัญของ DWT คือ การหาสัมประสิทธิ์ (ค่าคงที่) ของตัวแปร โดยการใช้ package wavelets และฟังก์ชัน dwt(x,filter,n.levels,...) เมื่อ x คือ ข้อมูลอนุกรมเวลา (univariate or multivariate time series), filter คือ ตัวกรองที่ใช้, n.levels คือ ระดับการแปลง (level of decomposition) ผลลัพธ์ที่ได้คือ class dwt ซึ่งค่า slot w(เรียกใช้@w) เป็น wavelet

```
> library(wavelets)
> wtData <- NULL
> for(i in 1:nrow(sc)) {
+   a <- t(sc[i,])
+   wt <- dwt(a, filter="haar", boundary="periodic")
+   wtData <- rbind(wtData, unlist(c(wt@W, wt@V[[wt@level]])))
+ }
> wtData <- as.data.frame(wtData)
> wtSc <- data.frame(cbind(classId, wtData))
> # build a decision tree with DWT coefficients
> ct <- ctree(classId ~ ., data=wtSc,
+               controls =ctree_control(minsplit=30, minbucket=10, maxdepth=5))
> pClassId <- predict(ct)
> table(classId, pClassId)

          pClassId
classId  1      2      3      4      5      6
  1       97     3     0     0     0     0
  2       1      99     0     0     0     0
  3       0      0    81     0    19     0
  4       0      0     0    63     0    37
  5       0      0    16     0    84     0
  6       0      0     0     1     0    99
```

coefficient และ slot v คือ scaling coefficient และอนุกรมเวลาสามารถถูกแปลงกลับเป็นข้อมูลด้วยเดิมด้วย idwt()

```
> (sum(classId==pClassId)) / nrow(wtSc)
[1] 0.8716667
> plot(ct, ip_args=list(pval=FALSE), ep_args=list(digits=0))
```

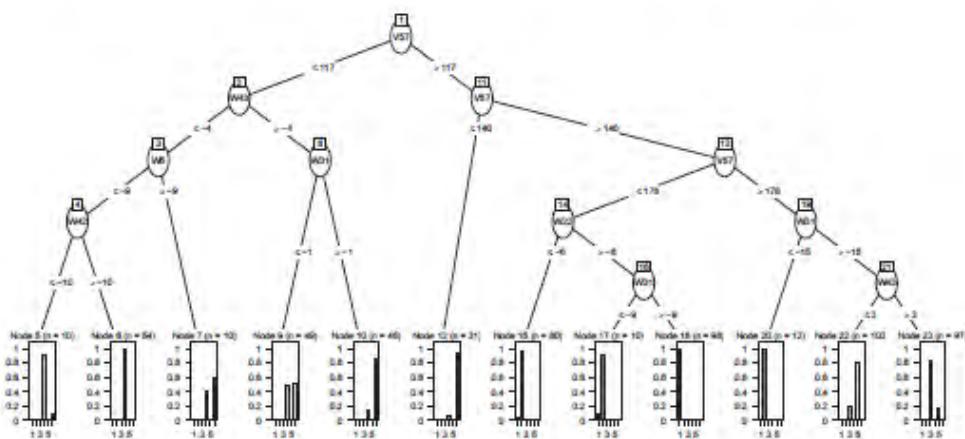


Figure 8.10 Decision tree with DWT.

8.5.3 การสร้างโมเดล k-NN classification

เราสามารถนำข้อมูล time series มาหาโมเดลชนิด k-NN classification โดยเมื่อนำข้อมูลใหม่ 1 ชุดมาแล้วต้องการพยากรณ์ว่าเป็น class อะไรก็นำไปเปรียบเทียบกับข้อมูลที่กล้วยๆจำนวน k ชุด แล้วดูว่าส่วนใหญ่แล้วเป็น class อะไร ข้อมูลใหม่นั้นก็จะเป็น class ส่วนใหญ่นั้น

เวลาที่ใช้ในโมเดล k-NN คือ $O(n^2)$ เมื่อ n คือขนาดข้อมูล ถ้าต้องการเพิ่มประสิทธิภาพจึงจำเป็นต้องใช้เทคนิคการสร้าง index เพิ่มขึ้น เราสามารถใช้ package RANN เพื่อทำ k-NN ที่ใช้เวลาเพียง $O(n \log n)$ ต่อไปนี้เป็นตัวอย่างการทำ k-NN classification แบบธรรมด้า (ไม่ใช้ index)

```

> k <- 20

> # create a new time series by adding noise to time series 501

> newTS <- sc[501,] + runif(100)*15

> distances <- dist(newTS, sc, method="DTW")

> s <- sort(as.vector(distances), index.return=TRUE)

> # class IDs of k nearest neighbors

> table(classId$six[1:k])

4   6

3 17

```

ผลลัพธ์ที่ได้ คือ ถ้าเรามีข้อมูลอนุกรมเวลาใหม่ 1 ชุด แล้วนำมาเทียบกับข้อมูล
ิกลีเดียง 20 ตัว ($k=20$) จะได้ 3 ตัวเป็น class 4, 17 ตัวเป็น class 6 ดังนั้น ข้อมูลใหม่นี้จะถูก
พยากรณ์ว่าจะเป็น class 6

8.6 บทสรุป

ในภาษา R มี package จำนวนมากที่ทำงานกับข้อมูลอนุกรมเวลาทั่ว ๆ ไป เช่น การแยก
องค์ประกอบ, การพยากรณ์ข้อมูลใหม่ แต่ยังไม่มีการทำ classification, clustering เนพาะกับ
ข้อมูลอนุกรมเวลา ซึ่งเป็นงานวิจัยที่ได้รับความสนใจสูง ในการทำ classification และ
clustering ทั่ว ๆ ไป สามารถทำได้โดย

1. กรองข้อมูลคอลัมน์และสร้างคอลัมน์ที่สำคัญเพิ่มเติม
2. นำเทคนิคทั่ว ๆ ไป เช่น โมเดล classification มาใช้กับข้อ 1

ถ้าต้องการหา โมเดล clustering เช่น k-means จะต้องกำหนดระยะทางวัดความเหมือน
(distance or similarity) นำมาใช้ให้เหมาะสมแล้วจึงใช้เทคนิค cluster ทั่ว ๆ ไปเพื่อหากลุ่มที่
เหมือนกัน

บทที่ 9 การหาโมเดล Association Rules

9.1 หลักการพื้นฐานของ association rules

- คือ กฎที่ใช้อธิบายความสัมพันธ์ระหว่างสิ่งที่เราสนใจว่าเกิดขึ้นพร้อมกันหรือไม่ อายุ
- รูปแบบของกฎนี้คือ $A \rightarrow B$ เมื่อ A และ B คือเซตที่ไม่มีสมาชิกร่วมกัน
- support (ค่าสนับสนุน) คือ ค่าเปอร์เซ็นต์ของกรณีที่เมื่อมี A แล้วจะมี B ด้วย
- lift คือ อัตราส่วนระหว่าง confidence กับ เปอร์เซ็นต์ที่มี B อยู่ สามารถแสดงได้ด้วย สูตร

$$\text{support}(A \Rightarrow B) = P(A \cup B), \quad (9.1)$$

$$\text{confidence}(A \Rightarrow B) = P(B|A), \quad (9.2)$$

$$= \frac{P(A \cup B)}{P(A)}, \quad (9.3)$$

$$\text{lift}(A \Rightarrow B) = \frac{\text{confidence}(A \Rightarrow B)}{P(B)}, \quad (9.4)$$

$$= \frac{P(A \cup B)}{P(A)P(B)}, \quad (9.5)$$

เมื่อ $P(A)$ คือเปอร์เซ็นต์ (หรือความน่าจะเป็น) ของกรณีที่มี A ปรากฏ

นอกเหนือจาก support, confident, lift ยังมีมาตรวัดอื่น ๆ อีกประมาณ 20 ตัว เช่น chi-square, conviction, gini, leverage สามารถศึกษาเพิ่มเติมจาก (Tan, 2002)

9.2 ข้อมูล Titanic

อยู่ใน package dataset ข้อมูล Titanic เป็นข้อมูลอยู่ในตารางมี 4 คอลัมน์ (variable) คือ social อายุ และรอดตายหรือไม่ (survival = yes, no) สามารถนำข้อมูล titanic.rdata มาใช้งาน class, sex, age, survival เป็นข้อมูลที่บันทึกรายละเอียดผู้โดยสารเรื่อไททานิกว่ามีรายละเอียดอย่างไรบ้าง ข้อมูลแต่ละแผลคือ ผู้โดยสารแต่ละคน และสามารถนำมาใช้หา association rules ได้

```

> str(Titanic)
table [1:4, 1:2, 1:2, 1:2] 0 0 35 0 0 0 17 0 118 154 ...
- attr(*, "dimnames")=List of 4
..$ Class    : chr [1:4] "1st" "2nd" "3rd" "Crew"
..$ Sex      : chr [1:2] "Male" "Female"
..$ Age      : chr [1:2] "Child" "Adult"
..$ Survived: chr [1:2] "No" "Yes"

> df <- as.data.frame(Titanic)

> head(df)

  Class   Sex   Age Survived Freq
1  1st   Male Child     No     0
2  2nd   Male Child     No     0
3  3rd   Male Child     No    35
4 Crew   Male Child     No     0
5  1st Female Child     No     0
6  2nd Female Child     No     0

> titanic.raw <- NULL

> for(i in 1:4) {
+   titanic.raw <- cbind(titanic.raw, rep(as.character(df[,i]),
+                                         df$Freq))
+ }

> titanic.raw <- as.data.frame(titanic.raw)
> names(titanic.raw) <- names(df)[1:4]

```

```

> dim(titanic.raw)
[1] 2201 4

> str(titanic.raw)

'data.frame' 2201 obs. of 4 variables:

$ Class    : Factor w/ 4 levels "1st","2nd","3rd",...: 3 3 3 3 3 3 3 3 3 ...
$ Sex      : Factor w/ 2 levels "Female","Male": 2 2 2 2 2 2 2 2 2 ...
$ Age      : Factor w/ 2 levels "Adult","Child": 2 2 2 2 2 2 2 2 2 ...
$ Survived: Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 ...

> head(titanic.raw)

  Class   Sex   Age   Survived
1 3rd   Male  Child  No
2 3rd   Male  Child  No
3 3rd   Male  Child  No
4 3rd   Male  Child  No
5 3rd   Male  Child  No
6 3rd   Male  Child  No

> summary(titanic.raw)

  Class        Sex          Age        Survived    
1st :325  Female:470  Adult:2092  No :1490    
2nd :285  Male   :1731  Child:109   Yes:711    
3rd :706                                          
Crew:885

```

The raw Titanic dataset can also be downloaded from <http://www.cs.toronto.edu/delve/data/titanic/desc.html>. The data is file “Dataset.data” in the compressed archive “titanic.tar.gz”. It can be read into *R* with the code below.

```

> # have a look at the 1st 5 lines
> readLines("./data/Dataset.data", n=5)
[1] "1st adult male yes" "1st adult male yes" "1st adult male
yes"
[4] "1st adult male yes" "1st adult male yes"

> # read it into R
> titanic <- read.table("./data/Dataset.data", header=F)
> names(titanic) <- c("Class", "Sex", "Age", "Survived")

```

9.3 Association Rule Mining

การหา Association Rule มีหลายวิธี แต่วิธีที่นิยมมากที่สุดคือ วิธี APRIORI มีลักษณะการทำงานที่คล้ายดับ ในแนววิธีที่ต้องการตัดสินใจไปข้างหน้า โดยการนับว่ามีสิ่งที่สนใจปรากฏบ่อยเพียงใด แล้วจึงนำมาจับคู่หางรูปที่เป็นไปได้ทั้งหมดตามค่า min support ที่กำหนดไว้

เราใช้ฟังก์ชัน apriori ใน package arules เพื่อหากฎต่าง ๆ นอกจากนี้ยังมีเทคนิค ECLAT เพื่อใช้หากฎในลักษณะ equivalence classes, depth-first search, และการหาความแตกต่างระหว่างเซต (แทนการนับ) โดยการใช้ eclat()

ตัวอย่างต่อไปนี้แสดงการหา association rules ด้วยวิธี apriori ซึ่งกำหนดให้ support = 0.1, confidence = 0.8, maxlen= 10 (ความยาวสูงสุดของกฎ)

```

> library(arules)
> # find association rules with default settings
> rules.all <- apriori(titanic.raw)

parameter specification:

confidence minval smax  arem aval  originalSupport support minlen maxlen target
0.8      0.1    1     none FALSE TRUE          0.1      1     10      rules

ext

FALSE

algorithmic control:

filter   tree   heap   memopt   load   sort   verbose
0.1      TRUE   TRUE   FALSE    TRUE   2      TRUE

apriori - find association rules with the apriori algorithm
version 4.21 (2004.05.09)      (c) 1996-2004 Christian Borgelt
set item appearances ... [0 item(s)] done [0.00s].
set transactions ... [10 item(s), 2201 transaction(s)] done [0.00s].
sorting and recoding items ... [9 item(s)] done [0.00s].
creating transaction tree ... done [0.00s].
checking subsets of size 1 2 3 4 done [0.00s].
writing ... [27 rule(s)] done [0.00s].
creating S4 object ... done [0.00s].
> rules.all
set of 27 rules
> inspect(rules.all)

```

	lhs	rhs	support	confidence	lift
1	{}	=> {Age=Adult}	0.9504771	0.9504771	1.0000000
2	{Class=2nd}	=> {Age=Adult}	0.1185825	0.9157895	0.9635051
3	{Class=1st}	=> {Age=Adult}	0.1449341	0.9815385	1.0326798
4	{Sex=Female}	=> {Age=Adult}	0.1930940	0.9042553	0.9513700
5	{Class=3rd}	=> {Age=Adult}	0.2848705	0.8881020	0.9343750
6	{Survived=Yes}	=> {Age=Adult}	0.2971377	0.9198312	0.9677574
7	{Class=Crew}	=> {Sex=Male}	0.3916402	0.9740113	1.2384742
8	{Class=Crew}	=> {Age=Adult}	0.4020900	1.0000000	1.0521033
9	{Survived=No}	=> {Sex=Male}	0.6197183	0.9154362	1.1639949
10	{Survived=No}	=> {Age=Adult}	0.6533394	0.9651007	1.0153856
11	{Sex=Male}	=> {Age=Adult}	0.7573830	0.9630272	1.0132040
12	{Sex=Female, Survived=Yes}	=> {Age=Adult}	0.1435711	0.9186047	0.9664669
13	{Class=3rd, Sex=Male}	=> {Survived=No}	0.1917310	0.8274510	1.2222950
14	{Class=3rd, Survived=No}	=> {Age=Adult}	0.2162653	0.9015152	0.9484870
15	{Class=3rd, Sex=Male}	=> {Age=Adult}	0.2099046	0.9058824	0.9530818
16	{Sex=Male, Survived=Yes}	=> {Age=Adult}	0.1535666	0.9209809	0.9689670
17	{Class=Crew, Survived=No}	=> {Sex=Male}	0.3044071	0.9955423	1.2658514
18	{Class=Crew, Survived=No}	=> {Age=Adult}	0.3057701	1.0000000	1.0521033
19	{Class=Crew, Sex=Male}	=> {Age=Adult}	0.3916402	1.0000000	1.0521033
20	{Class=Crew, Age=Adult}	=> {Sex=Male}	0.3916402	0.9740113	1.2384742
21	{Sex=Male, Survived=No}	=> {Age=Adult}	0.6038164	0.9743402	1.0251065

```

22 {Age=Adult,
     Survived=No} => {Sex=Male}      0.6038164 0.9242003 1.1751385

23 {Class=3rd,
     Sex=Male,
     Survived=No} => {Age=Adult}      0.1758292 0.9170616 0.9648435

24 {Class=3rd,
     Age=Adult,
     Survived=No} => {Sex=Male}      0.1758292 0.8130252 1.0337773

25 {Class=3rd,
     Sex=Male,
     Age=Adult}      => {Survived=No} 0.1758292 0.8376623 1.2373791

26 {Class=Crew,
     Sex=Male,
     Survived=No} => {Age=Adult}      0.3044071 1.0000000 1.0521033

27 {Class=Crew,
     Age=Adult,
     Survived=No} => {Sex=Male}      0.3044071 0.9955423 1.2658514

```

โดยปกติแล้วกฎที่ได้จาก association rule mining มักจะมีมากเกินไปและส่วนหนึ่งไม่น่าสนใจ เช่น ถ้าเราต้องการทราบเพียงกฎที่มีคำว่า “Survival = No”, “Survival = Yes” ปรากฏอยู่ทางขวา (rhs – right hand side) ส่วนทางด้านซ้าย (lhs) เป็นอื่น ๆ ที่เหลือจึงกำหนด default = “lhs”, และอาจจะเกิดกรณีที่ lhs เป็นค่า {} ซึ่งเราไม่ต้องการให้แสดงกฎนั้นจึงต้องทำการกำหนด minlen เป็น 2 และเราต้องการปิดการสนทนาโดยตอบ (เพื่อให้แสดงผลทันที) จึงกำหนด verbose = F, การแสดงกฎจะเรียงตามค่า lift จากมากไปน้อย

```

> # rules with rhs containing "Survived" only
> rules <- apriori(titanic.raw, control = list(verbose=F),
+   parameter = list(minlen=2, supp=0.005, conf=0.8),
+   appearance = list(rhs=c("Survived=No", "Survived=Yes"),
+     default="rhs"))
> quality(rules) <- round(quality(rules), digits=3)
> rules.sorted <- sort(rules, by="lift")
> inspect(rules.sorted)

      lhs                      rhs          support  confidence    lift
1 {Class=2nd,
  Age=Child} => {Survived=Yes} 0.011      1.000      3.096
2 {Class=2nd,
  Sex=Female,
  Age=Child} => {Survived=Yes} 0.006      1.000      3.096
3 {Class=1st,
  Sex=Female} => {Survived=Yes} 0.064      0.972      3.010
5 {Class=2nd,
  Sex=Female} => {Survived=Yes} 0.042      0.877      2.716
6 {Class=Crew,
  Sex=Female} => {Survived=Yes} 0.009      0.870      2.692
7 {Class=Crew,
  Sex=Female,
  Age=Adult} => {Survived=Yes} 0.009      0.870      2.692
8 {Class=2nd,
  Sex=Female,
  Age=Adult} => {Survived=Yes} 0.036      0.860      2.663
9 {Class=2nd,
  Sex=Male,
  Age=Adult} => {Survived=No} 0.070      0.917      1.354
10 {Class=2nd,
  Sex=Male} => {Survived=No} 0.070      0.860      1.271
11 {Class=3rd,
  Sex=Male,
  Age=Adult} => {Survived=No} 0.176      0.838      1.237
12 {Class=3rd,
  Sex=Male} => {Survived=No} 0.192      0.827      1.222

```

จะพบว่าเมื่อกำหนด minimum support ให้ลดลงจะได้กฎเพิ่มขึ้น (เพราะมีการกำหนดเปอร์เซ็นต์การเกิดขึ้นพร้อม ๆ กันน้อยลงทำให้มีกรณีที่สอดคล้องเพิ่มขึ้น)

นอกจากเกณฑ์ support, confidence, lift ยังมีเกณฑ์ที่นิยมใช้คือ chi-square, conviction, gini, leverage โดยเรียกใช้ฟังก์ชัน interestMeasure() สามารถศึกษาเพิ่มเติมได้จาก (Tan, 2002)

9.4 การกำจัดกฎที่ปรากฏซ้ำ (Removing Redundancy)

อาจจะเกิดเหตุการณ์ที่มีกฎบางตัวซ้ำกับกฎอื่นหรือไม่ให้ข้อมูลเพิ่มเติมที่เป็นประโยชน์ต่อการนำไปใช้งาน เช่น กฎที่ 2 ไม่มีประโยชน์ เพราะกฎที่ 1 ระบุครอบคลุมไว้แล้ว เราจะเรียกว่า กฎที่ 2 ปรากฏซ้ำ (redundant)

```
> # find redundant rules
> subset.matrix <- is.subset(rules.sorted, rules.sorted)
> subset.matrix[lower.tri(subset.matrix, diag=T)] <- NA
> redundant <- colSums(subset.matrix, na.rm=T) >= 1
> which(redundant)
[1] 2 4 7 8
> # remove redundant rules
> rules.pruned <- rules.sorted[!redundant]
> inspect(rules.pruned)

      lhs          rhs      support   confidence    lift
1 {Class=2nd,
  Age=Child} => {Survived=Yes} 0.011     1.000    3.096
2 {Class=1st,
  Sex=Female} => {Survived=Yes} 0.064     0.972    3.010
3 {Class=2nd,
  Sex=Female} => {Survived=Yes} 0.042     0.877    2.716
4 {Class=Crew,
  Sex=Female} => {Survived=Yes} 0.009     0.870    2.692
```

ตัวอย่างอื่นเช่น กฎที่ปรากฏชั้นคือ 4, 7, 8 เมื่อเปรียบเทียบกับกฎ 3, 6, 5 เราสามารถกำจัดกฎที่ปรากฏชั้นดังนี้

5	{Class=2nd,					
	Sex=Male,					
	Age=Adult)	=> {Survived=No}	0.070	0.917		1.354
6	{Class=2nd,					
	Sex=Male)	=> {Survived=No}	0.070	0.860		1.271
7	{Class=3rd,					
	Sex=Male,					
	Age=Adult)	=> {Survived=No}	0.176	0.838	1.237	
8	{Class=3rd,					
	Sex=Male)	=> {Survived=No}	0.192	0.827	1.222	

ฟังก์ชัน `is.subset(r1,r2)` ทำการตรวจสอบว่า `r1` เป็นเซตย่อยของ `r2` หรือไม่, ฟังก์ชัน `lower.tri()` จะให้ค่า `TRUE` ในแนวนวยงมุม เมตริกซ์ด้านล่าง (lower triangle) ผลที่ได้จะกำจัดกฎที่ปรากฏชั้นออกไป (pruned rules)

9.5 การอธิบายกฏ (Interpreting Rules)

เราสามารถหากฏที่มีค่า lift สูงໄດ້ไม่ยาก แต่การแปลความกฏที่ได้จะยากกว่า เพราะเป็นไปได้ว่ากฏที่หมายໄດ້ไม่สามารถนำไปใช้ประโยชน์ได้จริง เช่น

“{Class=2nd, Age=Child} => {Survive=Yes}”

มีค่า `conf = 1, lift = 3` และ “ไม่มีกฏอื่นที่ระบุถึงเด็กที่อยู่ในชั้นหนึ่งและสามเกี้ยวกับชั้นถ้าเราแปลความกฏนี้ว่า “เด็กที่อยู่ second class จะมีโอกาสลดมากกว่าเด็กที่อยู่ class อื่น” (ผิด) เพราะกฏอธิบายเพียงว่า เด็กที่อยู่ second class ทึ่หมครอดตาย แต่ไม่ได้เปรียบเทียบกับเด็กที่อยู่ที่ class อื่น (class อื่นอาจจะไม่มีเด็กเลยก็ได้)

ถ้าต้องการเปรียบเทียบเด็กกับ class อื่น ๆ จะต้องกำหนด rhs คือ Survival = Yes และ lhs มี Class = 1st, Class = 2nd, Class = 3rd, Age = Child, Age = Adult เท่านั้น default = “none” (ไม่มี item อื่น ๆ) เราใช้ threshold (support กับ confidence) ที่ตั้งไว้เพื่อให้แสดงจำนวนกฎที่มากขึ้น

```
> rules <- apriori(titanic.raw,
+                     parameter = list(minlen=3, supp=0.002, conf=0.2),
+                     appearance = list(rhs=c("Survived=Yes"),
+                                       lhs=c("Class=1st", "Class=2nd",
+                                             "Class=3rd",
+                                             "Age=Child", "Age=Adult"),
+                                       default="none"),
+                     control = list(verbose=F))
> rules.sorted <- sort(rules, by="confidence")
> inspect(rules.sorted)

      lhs          rhs          support      confidence    lift
1 {Class=2nd,
  Age=Child} => {Survived=Yes} 0.010904134 1.0000000  3.0956399
2 {Class=1st,
  Age=Child} => {Survived=Yes} 0.002726034 1.0000000  3.0956399
3 {Class=1st,
  Age=Adult}  => {Survived=Yes} 0.089504771 0.6175549  1.9117275
4 {Class=2nd,
  Age=Adult}  => {Survived=Yes} 0.042707860 0.3601533  1.1149048
5 {Class=3rd,
  Age=Child}  => {Survived=Yes} 0.012267151 0.3417722  1.0580035
6 {Class=3rd,
  Age=Adult}  => {Survived=Yes} 0.068605179 0.2408293  0.7455209
```

ผลที่ได้คือ 2 กฎแรกแสดง

เด็กที่ first class จะมีอัตรากว่าการรอดเท่ากับเด็กที่ second class

กฎที่ 5 แสดงให้เห็นว่า

เด็กที่อยู่ third class จะมีอัตราการรอดตายเพียง 34% (ซึ่งน้อยกว่าผู้ใหญ่ที่อยู่ first class, จากกฎที่ 3)

9.6 การแสดงภาพ association rule (Visualizing Association Rules)

สามารถแสดงภาพในลักษณะ scatter plot, balloon plot, parallel coordinate plot

ซึ่งอยู่ใน package arulesViz

```
> library(arulesViz)
> plot(rules.all) (see Figure 9.1)
> plot(rules.all, method="grouped") (see Figure 9.2)
```

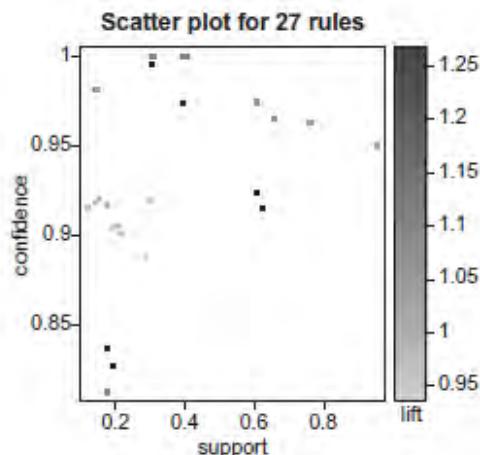


Figure 9.1 A scatter plot of association rules.

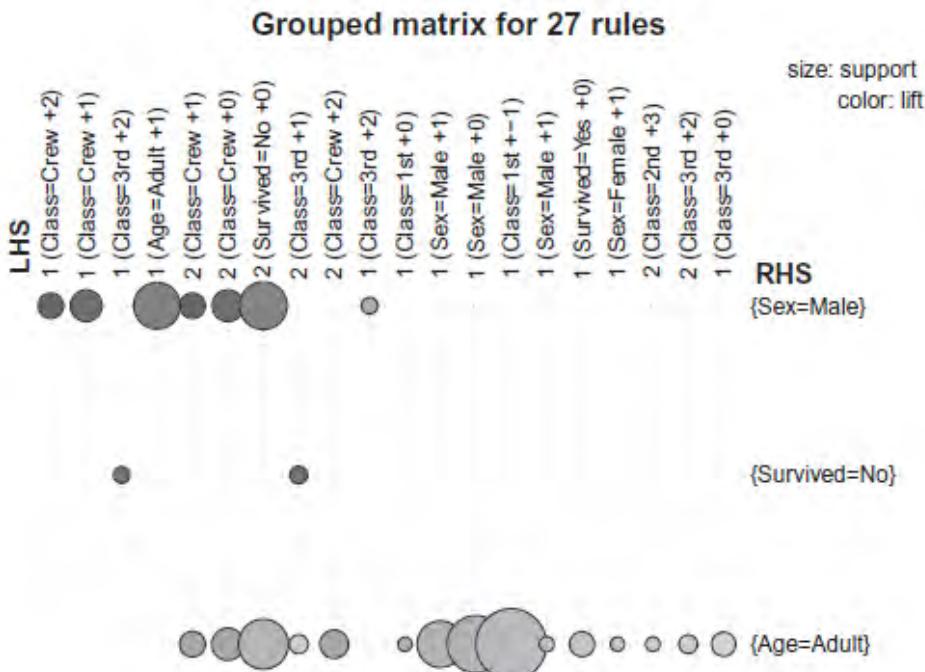


Figure 9.2 A balloon plot of association rules.

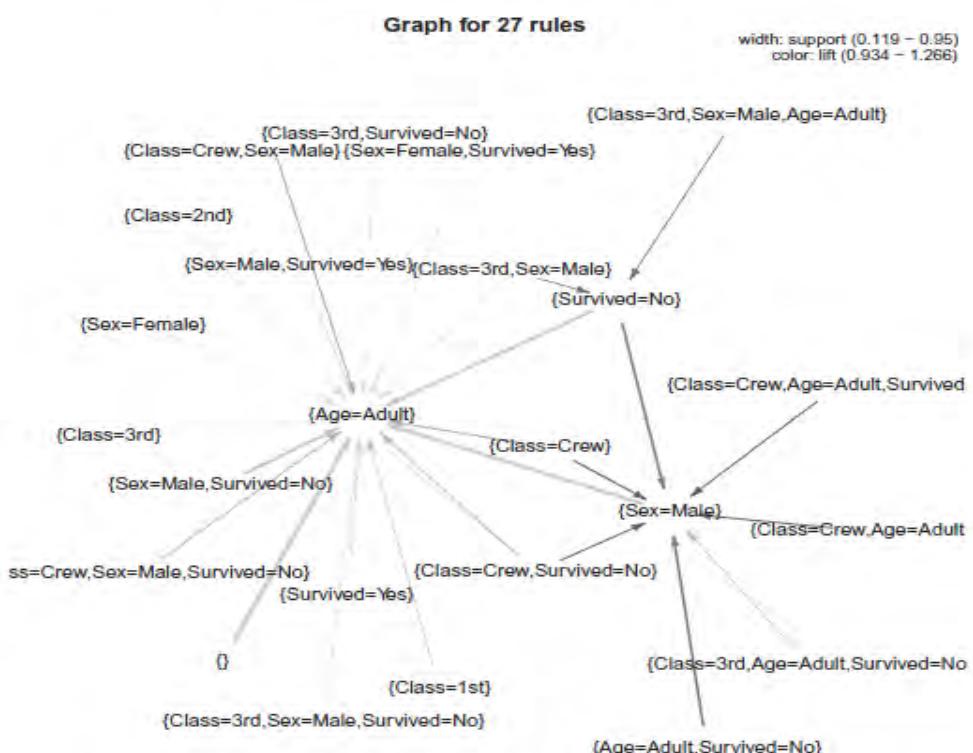


Figure 9.3 A graph of association rules.

```

> plot(rules.all, method="graph") (see Figure 9.3)
> plot(rules.all, method="graph", control=list(type="items")) (see
Figure 9.4)
> plot(rules.all, method="paracoord", control=list(reorder=TRUE)) (see
Figure 9.5)

```

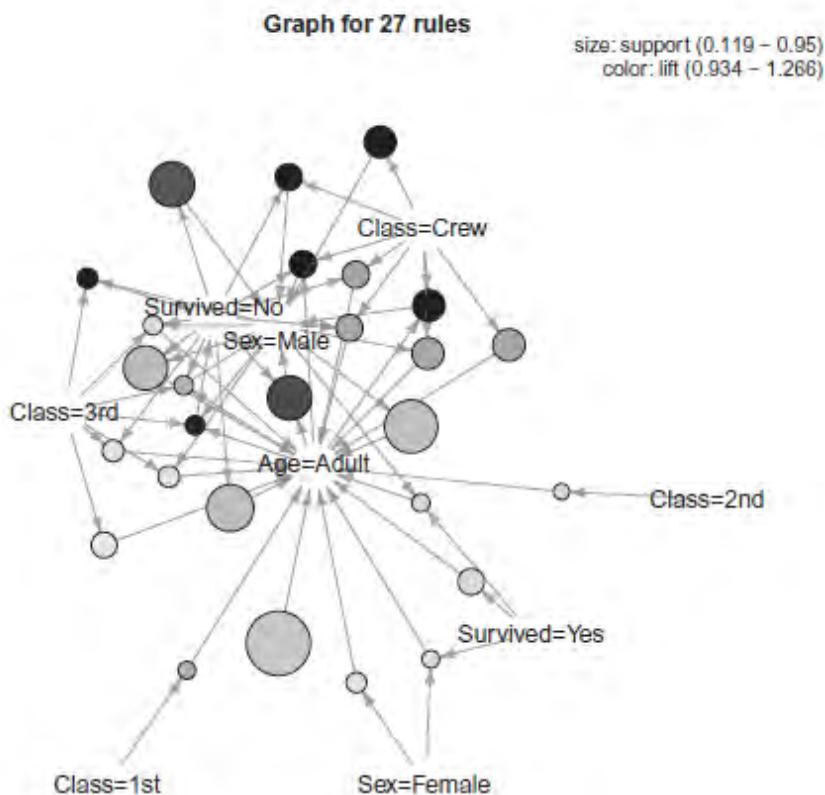


Figure 9.4 A graph of items.

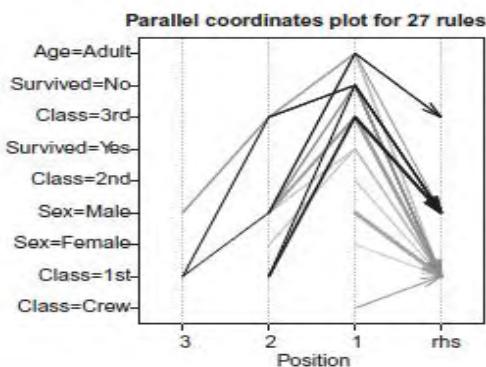


Figure 9.5 A parallel coordinates plot of association rules.

9.7 บทสรุป

บทนี้อธิบายการหากฎในลักษณะของ association rules จาก package arules ตัวอย่าง อื่น ๆ สามารถดูได้จาก Hashler 2011 นอกจากนี้ยังมี package อื่น ๆ เช่น arulesSequences และ arulesNBMiner

การพิจารณากฎภายในขั้นตอนการสร้าง (post-mining) มีหลายวิธี เช่น การ เสือกกฎที่น่าสนใจ, การแสดงกฎเป็นภาพ, การใช้ association rules สำหรับการทำ classification สามารถศึกษาเพิ่มเติมจาก (Zhao, 2009)

บทที่ 10 การทำเหมืองข้อมูล (Text Mining)

การทำเหมืองข้อมูล คือ การค้นหาความรู้จากข้อมูลจำนวนมาก ในที่นี่ใช้ข้อมูลจาก Twitter ที่ RDataMining

กระบวนการ Text Mining คือ

1. ดึงข้อมูลจากแหล่งข้อมูลนั้น
2. จัดข้อมูลให้อยู่ในรูปแมตริกซ์ (document-term matrix)
3. ค้นหาคำที่ปรากฏบ่อยและความสัมพันธ์ที่ปรากฏ (association)
4. แสดงคำที่ปรากฏบ่อยในรูปกลุ่มคำ (word cloud)
5. จัด cluster ของ word และ cluster ของ tweet (document)

จะมีการใช้ package twitter เพื่อเข้าจดข้อมูลจาก twitter

จะมีการใช้ package tm เพื่อทำ text mining

จะมีการใช้ package wordcloud เพื่อแสดงภาพกลุ่มคำ

10.1 การดึงข้อมูลจาก twitter

โดยการใช้ useTimeline() จาก Package twitter (ซึ่งจะต้องมีการเรียกใช้ package RCurl ก่อน) (อีกวิธีที่เรียกใช้ข้อมูลจาก twitter คือ ใช้ package XML) สำหรับผู้ที่ไม่สามารถเข้าใจบริการ twitter สามารถดาวน์โหลดข้อมูลเพื่อใช้ทดสอบได้ที่ <http://www.rdatamining.com/data> โดยเลือกไฟล์ rdmTweets.RData

```

> library(twitteR)

> # retrieve the first 200 tweets (or all tweets if fewer than
200) from the user timeline of @rdatamining

> rdmTweets <- userTimeline("rdatamining", n=200)

> (nDocs <- length(rdmTweets))

[1] 154

```

เราสามารถดู ข้อความที่ 11 ถึง 15 โดยใช้คำสั่ง

```
> rdmTweets [11:15]
```

ซึ่งจะแสดงข้อความติดต่อกันทำให้อ่านไม่สะดวก เราสามารถสั่งให้แสดงอย่างชัดเจนดังนี้

```

> for (i in 11:15) {
+   cat(paste ("[[", i, "]] ", sep=""))
+   writeLines(strwrap(rdmTweets[[i]]$getText(), width=73))
+ }

[[11]] Slides on massive data, shared and distributed
memory, and concurrent programming: bigmemory and foreach
http://t.co/a6bQzxj5

[[12]] The R Reference Card for Data Mining is updated with
functions & packages for handling big data & parallel computing.
http://t.co/FHoVZCyk

[[13]] Post-doc on Optimizing a Cloud for Data Mining primitives,
INRIA, France http://t.co/cA28STPO

[[14]] Chief Scientist - Data Intensive Analytics, Pacific
Northwest National Laboratory (PNNL), US http://t.co/0GdzqlNt
http://t.co/0GdzqlNt

[[15]] Top 10 in Data Mining http://t.co/7kAuNvuf

```

10.2 การแปลงข้อความ (Transforming Text)

ในการทำงานกับข้อความ โดยตรงจะไม่สะดวก เพราะ โครงสร้างข้อมูลที่ไม่ชัดเจน จึงต้องแปลงข้อความให้อยู่ในรูปที่ชัดเจน เพื่อนำไปประมวลผลได้ง่าย ซึ่งมี 2 ขั้นตอนดังนี้

1. แปลงให้อยู่ในโครงสร้าง data frame
2. แปลงข้อความจากข้อ 1 ให้อยู่ในรูป corpus (เนื้อความ)

```
> # convert tweets to a data frame  
  
> df <- do.call("rbind", lapply(rdmTweets, as.data.frame))  
  
> dim(df)  
  
[1] 154 10  
  
> library(tm)  
  
> # build a corpus, and specify the source to be character  
  vectors  
  
> myCorpus <- Corpus(VectorSource(df$text))
```

ขั้นตอนการแปลงให้อยู่ในรูป corpus จะต้องทำให้ละเอียดลงอีกคือ

- 2.1 แปลงให้เป็นตัวอักษรตัวเล็ก
- 2.2 ลบตัวคัน, ตัวเลข, คำหุ่ด (stop word) ตัวอย่างข้างล่างใช้ stop word มาตรฐานรวมกับคำ available, via และลบ r, big ออกจาก stop word (ใช้เป็นการแสดงให้เห็นการประยุกต์ใช้กับงานที่เราสามารถกำหนดเองได้)

```

> # convert to lower case
> myCorpus <- tm_map(myCorpus, tolower)
> # remove punctuation
> myCorpus <- tm_map(myCorpus, removePunctuation)
> # remove numbers
> myCorpus <- tm_map(myCorpus, removeNumbers)
> # remove URLs
> removeURL <- function(x) gsub("http[:alnum:]*", "", x)
> myCorpus <- tm_map(myCorpus, removeURL)
> # add two extra stop words: "available" and "via"
> myStopwords <- c(stopwords('english'), "available", "via")
> # remove "r" and "big" from stopwords
> myStopwords <- setdiff(myStopwords, c("r", "big"))
> # remove stopwords from corpus
> myCorpus <- tm_map(myCorpus, removeWords, myStopwords)

```

ฟังก์ชัน `tm_map()` ใช้ในการแปลงเป็น `corpus` นอกเหนือจากนี้เรายังสามารถต่อการแปลงแบบต่างๆ โดยใช้ `getTransformations()`

`removeURL()` ใช้ลบ hyperlink ซึ่งตรงกับรูปแบบ `http...`

10.3 การหาแก่นคำ (Stemming Words)

ในเอกสารจะปราศจากคำที่มีแก่นคำเดียวกัน แต่เปลี่ยนต่างกัน เช่น `walk`, `walking` จึงต้องลดรูปให้เป็นคำเดียวกัน วิธีการนี้เรียกว่า `stemming` ซึ่งมีฟังก์ชันให้เรียกใช้จาก package หลายตัว เช่น `snowball`, `RWeka`, `rJava`, `RWekajars`

หลังจากนั้นเราจะใช้ `stemCompletion()` เพื่อหา `stem` โดยการใช้คิดชั้นนารี `myCorpusCopy`

```

> # keep a copy of corpus to use later as a dictionary for stem
completion

> myCorpusCopy <- myCorpus

> # stem words

> myCorpus <- tm_map(myCorpus, stemDocument)

> # inspect documents (tweets) numbered 11 to 15

> # inspect(myCorpus[11:15])

> # The code below is used for to make text fit for paper width

> for (i in 11:15) {

+   cat(paste("[[", i, "]] ", sep=""))

+   writeLines(strwrap(myCorpus[[i]], width=73))

+ }

[[11]] slide massiv data share distribut memoryand concurr
program bigmemori foreach

[[12]] r refer card data mine updat function packag handl big
data parallel comput

[[13]] postdoc optim cloud data mine primit inria franc

[[14]] chief scientist data intens analyt pacif northwest nation
laboratori pnml

[[15]] top data mine
> # stem completion

> myCorpus <- tm_map(myCorpus, stemCompletion,
dictionary=myCorpusCopy)

```

เมื่อต้องการคุณลักษณะเอกสารที่ 11 ถึง 15 ทำดังนี้

```

> inspect(myCorpus[11:15])

[[11]] slides massive data share distributed memoryand concurrent
      programming foreach

[[12]] r reference card data miners updated functions package
      handling big data parallel computing

[[13]] postdoctoral optimizing cloud data miners primitives inria
      france

[[14]] chief scientist data intensive analytics pacific northwest
      national pnnl

[[15]] top data miners

```

จากผลลัพธ์จะพบว่ามีบางสิ่งหนึ่งอ่อนแอกลางความคาดหมายเกี่ยวกับการทำ stemming คือ

1. จาก corpus จะได้ “memoryand” ซึ่งได้มาจากการแยกอักขระที่ 11 คือ “...memory, and...”
2. จาก corpus จะได้ “bigmemori” ซึ่งได้มาจากการแยกอักขระที่ 11 คือ “bigmemory”
3. ข้อความที่ 12, 13, 15 จากคำว่า “mining” จะถูกทำ stemming ได้ “mine” และทำให้เสร็จสมบูรณ์เป็น “miners”
4. ข้อความที่ 14 จากคำว่า “Laboratory” จะถูกทำ stemming ได้ “laboratori” และทำให้เสร็จสมบูรณ์ข้อความนี้จะหายไป

สาเหตุแต่ละข้อคือ

- ข้อ 1. เพราะไม่มีช่องว่างตรงเครื่องหมาย, แก้ไขโดยเติมช่องว่างหลัง,
- ข้อ 2. และ 4 ไม่ทราบແนรัชด้วนว่าเกิดจากสาเหตุใด แต่โดยปกติที่คำเหล่านี้ไม่มีความสำคัญ
- ข้อ 3. อาจจะแก้ไขโดยการปรับปรุงคิกชันนารี แต่อาจจะเสียเวลาในการหาวิธีแก้ไข เราสามารถทำง่ายได้โดยการแทนค่า miners ด้วยคำสั่งดังนี้

```

> # count frequency of "mining"
> miningCases <- tm_map(myCorpusCopy, grep, pattern="\\\<mining")
> sum(unlist(miningCases))
[1] 47

> # count frequency of "miners"
> minerCases <- tm_map(myCorpusCopy, grep, pattern="\\\<miners")
> sum(unlist(minerCases))
[1] 2

> # replace "miners" with "mining"
> myCorpus <- tm_map(myCorpus, gsub, pattern="miners",
replacement="mining")

```

คำสั่ง `tm_map()` จะทำการเรียก `grep()` เพื่อทำงานกับเอกสารด้วยการใช้พารามิเตอร์ “`pattern = "\\\<mining"` เป็นการระบุแพทเทิร์นให้หาคำว่า `mining` ที่คั่นหน้าด้วยช่องว่าง เพื่อให้แน่ใจว่า “`rdatamining`” จะไม่ถูกนำมาพิจารณา

10.4 การสร้างเมต्रิกซ์แสดงความสัมพันธ์ระหว่างคำกับเอกสาร (Building a Term-Document Matrix)

Term-Document Matrix คือ เมตริกซ์ที่เก็บความสัมพันธ์ระหว่างคำกับเอกสาร คือ ระบุว่าเอกสารแต่ละตัวมีคำอะไรปรากฏ โดยแต่ละແลขอจะเป็นคำและคอลัมน์เป็นเอกสาร คำนี้มาจาก corpus สามารถสร้างเมต릭ซ์ด้วย `TermDocumentMatrix()` การกำหนด default parameter จะไม่พิจารณาคำที่มีความยาวน้อยกว่า 3 ถ้าต้องการระบุเอง จะต้องกำหนดที่ `wordLengths`

```

> myTdm <- TermDocumentMatrix(myCorpus,
  control=list(wordLengths=c(1, Inf)))
> myTdm
A term-document matrix (444 terms, 154 documents)
Non-/sparse entries : 1085/67291
Sparsity             : 98%
Maximal term length : 27
Weighting            : term frequency (tf)

```

ผลที่ได้คือเมตริกซ์ประกอบด้วย 444 คำมากจาก 154 เอกสาร มีลักษณะกระจายมาก โดย 98% มีค่าศูนย์ เราสามารถดู 6 คำแรกที่ขึ้นต้นด้วย r จากข้อความที่ 101 ถึง 110 ดังนี้

```

> idx <- which(dimnames(myTdm)$Terms == "r")
> inspect(myTdm[idx+(0:5),101:110])
A term-document matrix (6 terms, 10 documents)
Non-/sparse entries : 9/51
Sparsity             : 85%
Maximal term length : 12
Weighting            : term frequency (tf)

```

Terms	Docs									
	101	102	103	104	105	106	107	108	109	110
r	1	1	0	0	2	0	0	1	1	1
ramachandran	0	0	0	0	0	0	0	0	0	0
random	0	0	0	0	0	0	0	0	0	0
ranked	0	0	0	0	0	0	0	0	1	0
rapidminer	1	0	0	0	0	0	0	0	0	0
rdatamining	0	0	0	0	0	0	0	1	0	0

เราสามารถระบุความยาวของคำโดยใช้ minWordLength, และสามารถถูคำโดยใช้ rownames(myTdm) เราสามารถนำคำเหล่านี้ไปใช้ในขั้นตอน data mining ทั่วไป เช่น classification, clustering ในบางครั้งการที่มีคำมากเกินไป ไม่เหมาะสมในการนำไปใช้งานจึงอาจจะลดคำที่ไม่สำคัญออกด้วยเทคนิคต่าง ๆ เช่น การกำหนด minimum number of document, การกรองคำด้วย TF-IDF (Term frequency incuse document frequency)

10.5 คำที่ปรากฏบ่อย และการทำ Association (Frequent Terms and Association)

สามารถหาคำที่ปรากฏบ่อยดังนี้

```
> # inspect frequent words  
> findFreqTerms(myTdm, lowfreq=10)  
  
[1] "analysis" "computing" "data"      "examples" "introduction"  
[6] "mining"    "network"   "package"   "positions" "postdoctoral"  
[11] "r"         "research"  "slides"    "social"    "tutorial"  
[16] "users"
```

ฟังก์ชัน findFreqTerms() จะหาคำที่ปรากฏบ่อยไม่ต่ำกว่า 10 ครั้ง และแสดงเรียงลำดับตามตัวอักษร และถ้าต้องการให้แสดง bar plot เพื่อแสดงคำเหล่านี้ว่ามีจำนวนคำเท่าไร โดยใช้ rowsum() เพื่อหาผลรวมของคำนั้น และ ggplot2 เพื่อแสดง bar plot โดยมีพารามิเตอร์ geom = “bar” เป็นการระบุแสดงชนิดของ bar, coord_flip() เป็นการสลับแกน x กับแกน y ดังต่อไปนี้

```

> termFrequency <- rowSums(as.matrix(myTdm))

> termFrequency <- subset(termFrequency, termFrequency >= 10)

> library(ggplot2)

> qplot(names(termFrequency), termFrequency, geom="bar",
xlab="Terms") + coord_flip()

```

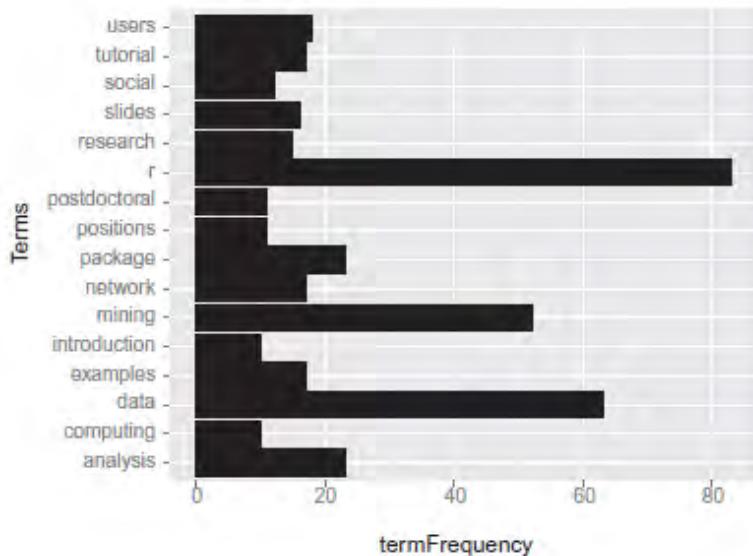


Figure 10.1 Frequent terms.

นอกจากนี้เราสามารถสร้าง bar plot โดยใช้ barplot() เมื่อ las = 2 เป็นการกำหนดแสดงในแนวตั้ง

```
> barplot(termFrequency, las = 2)
```

ถ้าต้องการทำ association เพื่อคุณว่าคำที่เราสนใจมีความสัมพันธ์กับคำอื่นอย่างไรทำได้โดยใช้ findAssocs()

```

> # which words are associated with "r"?
> findAssocs(myTdm, 'r', 0.25)
      r     users   canberra    cran   list examples
1.00  0.32    0.26       0.26  0.26  0.25

> # which words are associated with "mining"?
> findAssocs(myTdm, 'mining', 0.25)
      mining     data   mahout recommendation   sets
1.00        0.55     0.39         0.39        0.39
      supports frequent itemset      card functions
0.39        0.35     0.34         0.29        0.29
      reference   text
0.29        0.26

```

10.6 กลุ่มคำ (Word Cloud)

Word Cloud คือกลุ่มคำหรือกลุ่มก้อนเมฆของคำ ใช้ในการหมายที่มีคำเป็นจำนวนมากและต้องการให้แสดงบางสิ่งที่สำคัญภายในนั้น สามารถสร้างได้โดย package wordcloud ซึ่งมี wordcloud() ใช้แสดงภาพ การกำหนด pandom.order = F เป็นการกำหนดให้คำที่ปรากฏบ่อยแสดงที่กลางภาพและใช้ rainbow() เพื่อให้แสดงสีอย่างสวยงาม

```

> library(wordcloud)
> m <- as.matrix(myTdm)
> # calculate the frequency of words and sort it descendingly by
  frequency
> wordFreq <- sort(rowSums(m), decreasing=TRUE)
> # word cloud
> set.seed(375) # to make it reproducible
> grayLevels <- gray((wordFreq+10) / (max(wordFreq)+10))
> wordcloud(words=names(wordFreq), freq=wordFreq, min.freq=3,
  random.order=F, colors=grayLevels)

```

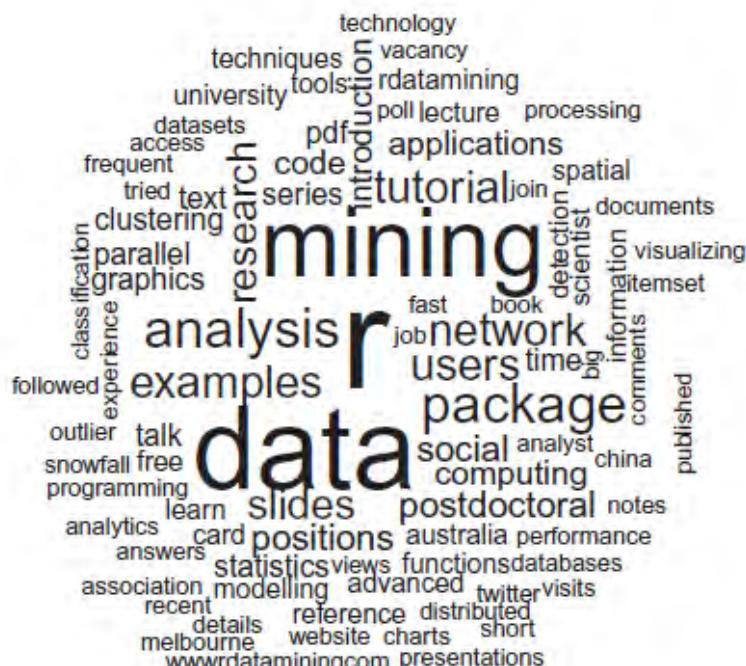


Figure 10.2 Word cloud.

ภาพที่ได้แสดงให้เห็นว่าคำ r, data, mining เป็นคำยอดนิยม 3 คำแรก ที่มีอยู่ในเอกสารเหล่านี้

10.7 การจัดกลุ่มคำ (Clustering Words)

เราสามารถจัดกลุ่มคำด้วยเทคนิค hierarchical clustering (จัดกลุ่มจากล่างขึ้นบน) โดยคำที่มีการกระจายมาก ๆ จะถูกกลุ่มทั้ง และระยะทางระหว่างคำจะถูกคำนวณด้วย dist() และวิธีที่ใช้ในการรวมกลุ่มคือ variance โดยกำหนดให้แสดงเพียง 10 กลุ่ม

```
> # remove sparse terms  
> myTdm2 <- removeSparseTerms(myTdm, sparse=0.95)  
> m2 <- as.matrix(myTdm2)  
> # cluster terms  
> distMatrix <- dist(scale(m2))  
> fit <- hclust(distMatrix, method="ward")  
> plot(fit)  
> # cut tree into 10 clusters  
> rect.hclust(fit, k=10)
```

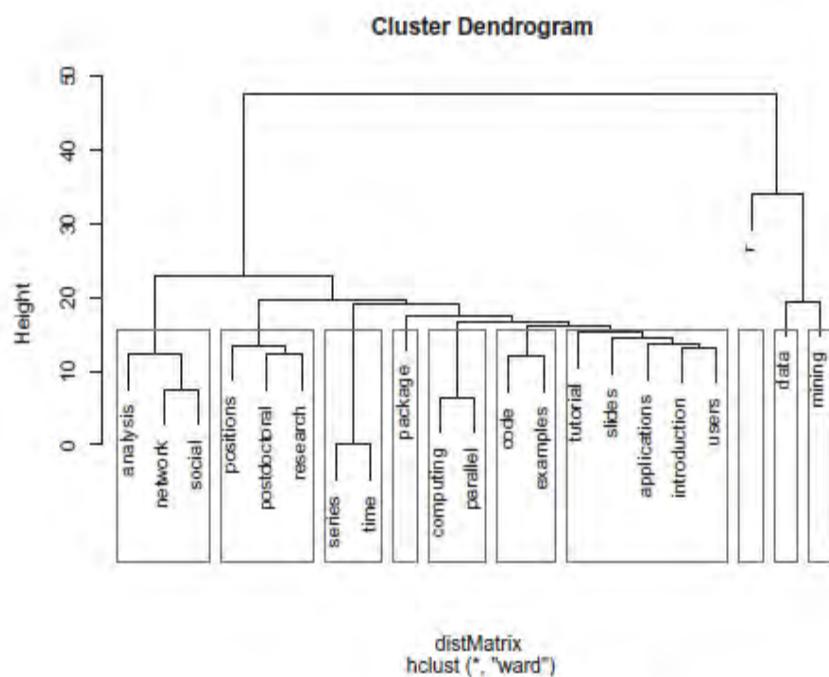


Figure 10.3 Clustering of words.

```

> (groups <- cutree(fit, k=10))

analysis      applications   code      computing   data      examples
1             2                 3          4          5          3
introduction  mining       network    package    parallel  positions
2             6                 1          7          4          8
postdoctoral r           research  series    slides    social
8             9                 8          10         2          1
time          tutorial     users
10            2                 2

```

10.8 การจัดกลุ่มเอกสาร (Clustering Tweets)

สามารถจัดกลุ่มโดยวิธีทั่วๆไปคือ k-mean และ k-medoids

10.8.1 โดยใช้วิธี k-means มีขั้นตอนดังนี้

1. โดยการ transpose = สลับ列-หลัก) เมตริกซ์ Term-Document เป็นเมตริกซ์ Document-Term
2. เรียกใช้ kmean() โดยกำหนดให้ $k = 8$
3. หาคำที่ปรากฏบ่อยมากที่สุด 3 คำแรกภายในกลุ่ม และหาจุดกึ่งกลางกลุ่ม
4. กำหนดค่าสุ่มคงที่เพื่อจะได้ตรวจสอบผลลัพธ์หลาย ๆ ครั้งแล้วให้ผลเหมือนเดิม ด้วยฟังก์ชัน set.seed() ก่อนเรียกใช้ kmeans()

```

> # transpose the matrix to cluster documents (tweets)
> m3 <- t(m2)

> # set a fixed random seed
> set.seed(122)

> # k-means clustering of tweets

> k <- 8

> kmeansResult <- kmeans(m3, k)

> # cluster centers

> round(kmeansResult$centers, digits=3)

  analysis applications code computing data examples introduction mining
  1 0.040    0.040      0.240 0.000    0.040 0.320    0.040    0.120
  2 0.000    0.158      0.053 0.053    1.526 0.105    0.053    1.158
  3 0.857    0.000      0.000 0.000    0.000 0.071    0.143    0.071
  4 0.000    0.000      0.000 1.000    0.000 0.000    0.000    0.000
  5 0.037    0.074      0.019 0.019    0.426 0.037    0.093    0.407
  6 0.000    0.000      0.000 0.000    0.000 0.100    0.000    0.000
  7 0.533    0.000      0.067 0.000    0.333 0.200    0.067    0.200
  8 0.000    0.111      0.000 0.000    0.556 0.000    0.000    0.111

  network package parallel positions postdoctoral r      research series slides
  1 0.080    0.080      0.000 0.000    0.000      1.320 0.000    0.040 0.000
  2 0.000    0.368      0.053 0.000    0.000      0.947 0.053    0.000 0.053
  3 1.000    0.071      0.000 0.143    0.143      0.214 0.071    0.000 0.071
  4 0.000    0.125      0.750 0.000    0.000      1.000 0.000    0.000 0.125
  5 0.000    0.000      0.000 0.093    0.093      0.000 0.000    0.019 0.074
  6 0.000    1.200      0.100 0.000    0.000      0.600 0.100    0.000 0.100
  7 0.067    0.000      0.000 0.000    0.000      1.000 0.000    0.400 0.533
  8 0.000    0.000      0.000 0.444    0.444      0.000 1.333    0.000 0.000

```

	social	time	tutorial	users
1	0.000	0.040	0.200	0.160
2	0.000	0.000	0.000	0.158
3	0.786	0.000	0.286	0.071
4	0.000	0.000	0.125	0.250
5	0.000	0.019	0.111	0.019
6	0.000	0.000	0.100	0.100
7	0.000	0.400	0.000	0.400
8	0.111	0.000	0.000	0.000

ตัวอย่างต่อไปนี้เป็นการหาคำที่ปรากฏบ่อย 3 คำแรกภายในแต่ละ cluster

ผลลัพธ์ที่ได้พบว่า กลุ่มที่ 1 จะปรากฏคำที่พบบ่อย (พูดเกี่ยวกับ) r, example, code

```
> for (i in 1:k) {
+ cat(paste("cluster ", i, ":", sep=""))
+ s <- sort(kmeansResult$centers[i,], decreasing=T)
+ cat(names(s)[1:3], "\n")
+ # print the tweets of every cluster
+ # print(rdmTweets[which(kmeansResult$cluster==i)])
+ }
cluster 1: r examples code
cluster 2: data mining r
cluster 3: network analysis social
cluster 4: computing r parallel
cluster 5: data mining tutorial
cluster 6: package r examples
cluster 7: r analysis slides
cluster 8: research data positions
```

10.8.2 การจัดกลุ่มโดยใช้วิธี k-medoids

คล้ายกับวิธี k-means สิ่งที่แตกต่าง ก็คือตัวแทนกลุ่ม (จุดกึ่งกลาง) จะต้องมีอยู่จริงภายในกลุ่มนั้น โดยใช้วิธี PAM (Partitioning Around Medoids) วิธีนี้จะทนทานต่อข้อมูลรบกวน (noise) และข้อมูลที่อยู่แตกต่างออกไปมาก ๆ (outlier) ในตัวอย่างต่อไปนี้จะแสดงชิลูอेट (silhouette) ซึ่งเป็นมาตรวัดแสดงคุณภาพของ cluster

ฟังก์ชัน pamk() จาก package fpc ใช้ทำ k-medoids ซึ่งจะคำนวณค่า k (จำนวนกลุ่ม) ที่เหมาะสมอัตโนมัติ (จะมีการเรียกใช้ pam() ภายใน)

```
> library(fpc)

> # partitioning around medoids with estimation of number of
  clusters

> pamResult <- pamk(m3, metric="manhattan")

> # number of clusters identified

> (k <- pamResult$nc)
[1] 9

> pamResult <- pamResult$pamobject

> # print cluster medoids

> for (i in 1:k) {
+   cat(paste("cluster", i, ": "))
+   cat(colnames(pamResult$medoids)
+       [which(pamResult$medoids[i,]==1)], "\n")
+   # print tweets in cluster i
+   # print(rdmTweets[pamResult$clustering==i])
+ }

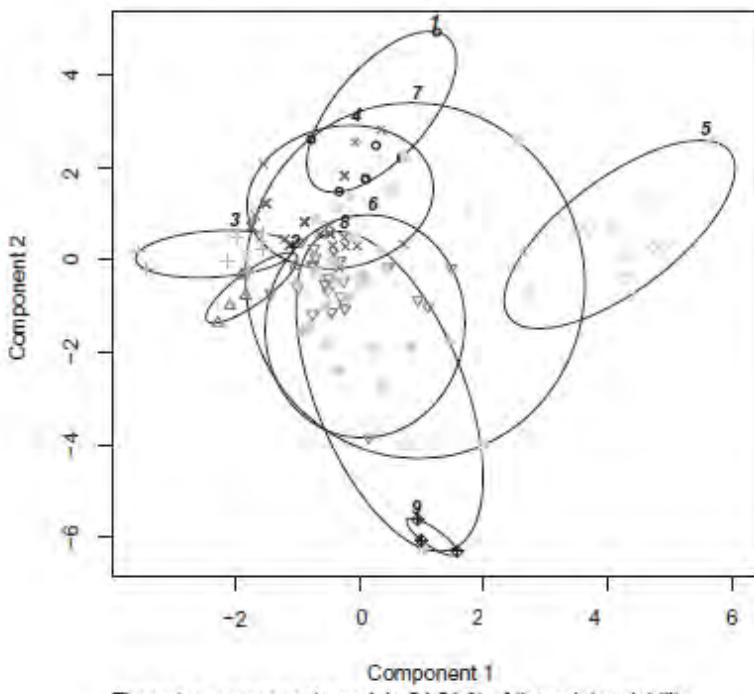
cluster 1:  data positions research
cluster 2:  computing parallel r
cluster 3:  mining package r
cluster 4:  data mining
cluster 5:  analysis network social tutorial
cluster 6:  r
cluster 7:
cluster 8:  examples r
cluster 9:  analysis mining series time users
```

```

> # plot clustering result
> layout(matrix(c(1,2),2,1)) # set to two graphs per page
> plot(pamResult, color=F, labels=4, lines=0, cex=.8, col.clus=1,
+       col.p=pamResult$clustering)
> layout(matrix(1)) # change back to one graph per page

```

```
clusplot(pam(x = sdata, k = k, metric = "manhattan"))
```



These two components explain 24.81 % of the point variability.

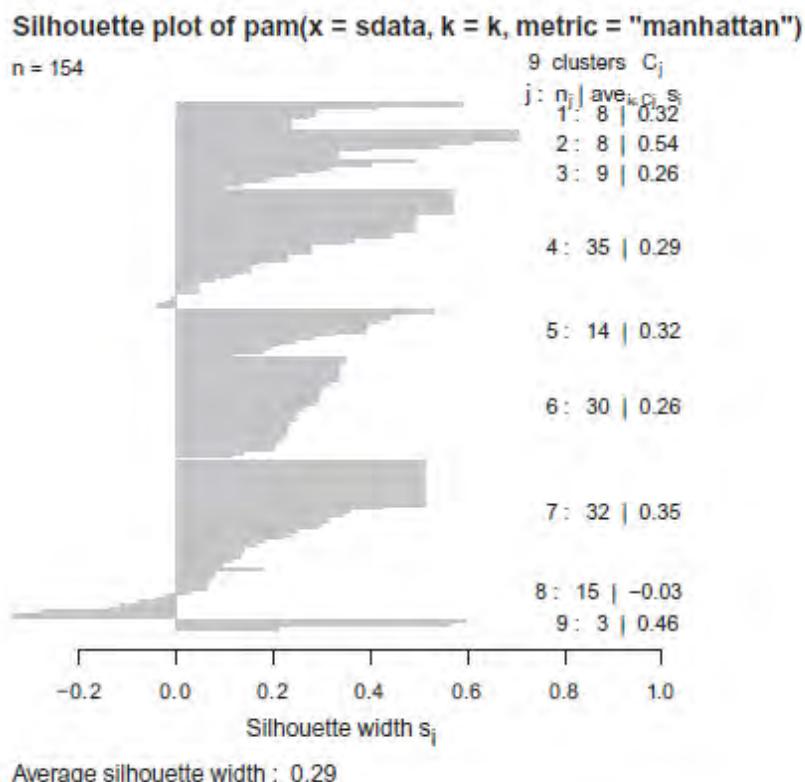


Figure 10.4 Clusters of tweets.

รูปที่ 10.4 ภาพแรก แสดง clusplot และภาพที่สอง แสดงค่า silhouette มีความหมายว่า ถ้ามีค่ามาก ๆ (เข้าใกล้ 1) จะได้ว่ากกลุ่มถูกจัดอย่างดี แต่ถ้าติดลบ แสดงว่า จัดกลุ่มผิด ถ้าค่าเป็น 0 แสดงว่า ถูกจัดอยู่ระหว่าง 2 กลุ่ม (ถูกและผิด) จากผลที่ได้มีค่า = 0.29 แสดงว่า การจัดกลุ่มนี้ไม่ค่อยดี

การตีความผลที่ได้ คือ จะได้กลุ่มจำนวน 9 กลุ่ม (หาโดยอัตโนมัติได้ $k = 9$) กลุ่มที่ 1, 2, 3, 5, 9 จัดได้ดี (เน้นไปที่เนื้อหาเฉพาะ) กลุ่ม 7 คือ เอกสารไม่เข้ากับกลุ่มอื่น และซ้อนทับกับกลุ่มอื่น กลุ่ม 6 กับ 8 มีการซ้อนทับกันมาก และในกลุ่ม 8 ค่า silhouette เป็นค่าติดลบ แสดงว่าข้อมูลบางตัวไม่ควรอยู่ในกลุ่ม 8 (จัดกลุ่มได้ไม่ดี)

วิธีการปรับปรุงให้มีการจัดกลุ่มคี้น (ต้องการให้ silhouette มีค่ามากขึ้น) โดยการกำหนดช่วงค่า k ให้อยู่ระหว่าง 2 ถึง 8 คือ krangle = 2 : 8 และวิธีการใช้ pamk() จะได้ค่าในกลุ่ม 8 ส่วนหนึ่งจะถูกจัดไปที่กลุ่ม 8

```
> pamResult2 <- pamk(m3, krangle=2:8, metric="manhattan")
```

10.9 บทสรุป

ในบทนี้แสดงการหาคำที่ปรากฏบ่อยในเอกสาร (twitter) และนำໄไปทำ association, clustering นอกจากนี้ยังแสดงผลการทำงานอยู่ในรูปกราฟแบบต่าง ๆ และยังสามารถแสดงในรูปโครงข่ายซึ่งจะอธิบายต่อในบทที่ 11 Social Network Analysis

Package อื่น ๆ เกี่ยวกับ Text mining คือ

1. tm ประกอบด้วย framework สำหรับการทำ text mining
2. tm.plugin.mail เป็นส่วนเสริม (plugin) ที่ทำกับ Text mining ของ E-mail
3. texteal ทำกับ n-Gram Text mining
4. Lda เกี่ยวข้องกับ LDA (Latent Dirichlet Allocation)
5. Topicmodels เกี่ยวข้องกับ LDA และ CTM (correlated topics model)

บทที่ 11 การวิเคราะห์เครือข่ายสังคม (Social Network Analysis)

ในการวิเคราะห์ Social Network จะคล้ายกับ Text mining ของข้อมูลการติดต่อสื่อสาร (Tweet) คือทำการวิเคราะห์ความสัมพันธ์ของกลุ่มคำ ระหว่างกลุ่มว่ามีความเกี่ยวข้องกันหรือไม่ มีหลักการดังนี้

1. สร้างเครือข่ายของคำว่ามีความเชื่อมโยงภายในเอกสารอย่างไร
2. สร้างเครือข่ายของเอกสารว่ามีความเชื่อมโยงคำภายในเอกสารนั้น ๆ อย่างไร
3. สร้างเครือข่ายของทั้งเอกสารและคำ

11.1 Network of Term

เป็นการแสดงความสัมพันธ์ของคำภายในเอกสาร โดยแสดงเป็นเส้นเชื่อมมีขึ้นตอน คือ

1. โหลด term-document เมตริกซ์ ชื่อ termDocMatrix (คือ m2 จากบทที่ 10)
2. แปลงเป็น term-term adjacency matrix (เมตริกซ์เก็บความสัมพันธ์ term-term ในแนวคอลัมน์ และแถว โดยค่าเพียงแนวสามเหลี่ยมบนเท่านั้น)
3. สร้างกราฟแสดงคำที่ปรากฏบ่อย ๆ

```

> # load termDocMatrix
> load("./data/termDocMatrix.rdata")
> # inspect part of the matrix
> termDocMatrix[5:10, 1:20]

          Docs
Terms      1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
data       1 1 0 0 2 0 0 0 0 0 1 2 1 1 1 0 1 0 0 0 0
Examples   0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
introduction 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1
mining     0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 0 0 0 0 0 0
network    0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 1 1 1
package    0 0 0 1 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0

> # change it to a Boolean matrix
> termDocMatrix[termDocMatrix>=1] <- 1
> # transform into a term-term adjacency matrix
> termMatrix <- termDocMatrix %*% t(termDocMatrix)
> # inspect terms numbered 5 to 10
> termMatrix[5:10, 5:10]

          Terms
Terms      data examples introduction mining network package
data       53      5           2        34      0       7
Examples   5       17          2         5      2       2
introduction 2       2          10        2      2       0
mining     34      5           2        47      1       5
network    0       2           2         1      17      1
package    7       2           0         5      1      21

```

จากคำสั่งที่ผ่านมา เครื่องหมาย %*% คือการคูณเมตริกซ์ และ t() คือการทำทรานสโพสเมตริกซ์ เมื่อเราคือคำ, คอลัมน์คือ คำ และค่าในเมตริกซ์คือ จำนวนที่ปรากฏพร้อม ๆ กัน ของทั้ง 2 คำ (ยิ่งมากยิ่งดี) ขึ้นต่อไปคือสร้างกราฟด้วย graph.adjacency() ซึ่งอยู่ใน package igraph

```

> library(igraph)

> # build a graph from the above matrix

> g <- graph.adjacency(termMatrix, weighted=T, mode="undirected")

> # remove loops

> g <- simplify(g)

> # set labels and degrees of vertices

> V(g)$label <- V(g)$name

> V(g)$degree <- degree(g)

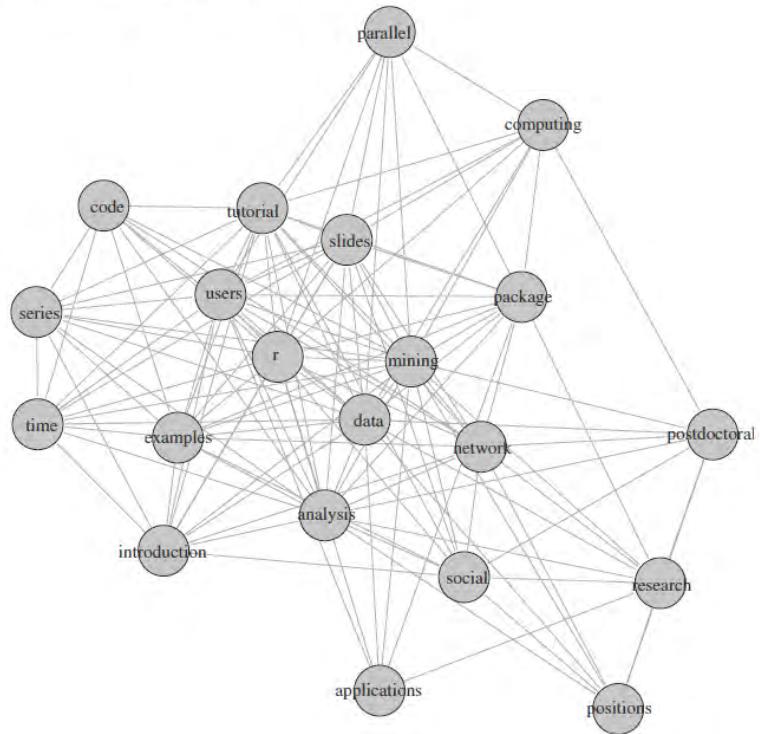
```

ต่อจากนั้นจึงสร้างกราฟ network ด้วย layout.fruchterman.reingold

```

> # set seed to make the layout reproducible
> set.seed(3952)
> layout1 <- layout.fruchterman.reingold(g)
> plot(g, layout=layout1)

```



รูปที่ 11.1 A network of terms-I

รูปข้างบนจะถูกเก็บไว้ที่ layout1 และเราสามารถจัดการเพิ่มเติมได้ในภายหลัง นอกจากนี้เราสามารถกำหนดรูปแบบ (layout) ที่แตกต่างออกໄປได้ด้วยคำสั่งบรรทัดที่ 1, ส่วนบรรทัดที่ 2 เป็นการระบุให้สามารถจัดการในภายหลังด้วยมือได้ (interactive plot) ถ้าต้องการดูรายละเอียด

```

> plot(g, layout=layout.kamada.kawai)
> tkplot(g, layout=layout.kamada.kawai)

```

เราสามารถ save เป็น pdf ดังนี้

```

> pdf("term-network.pdf")
> plot(g, layout=layout.fruchterman.reingold)
> dev.off()

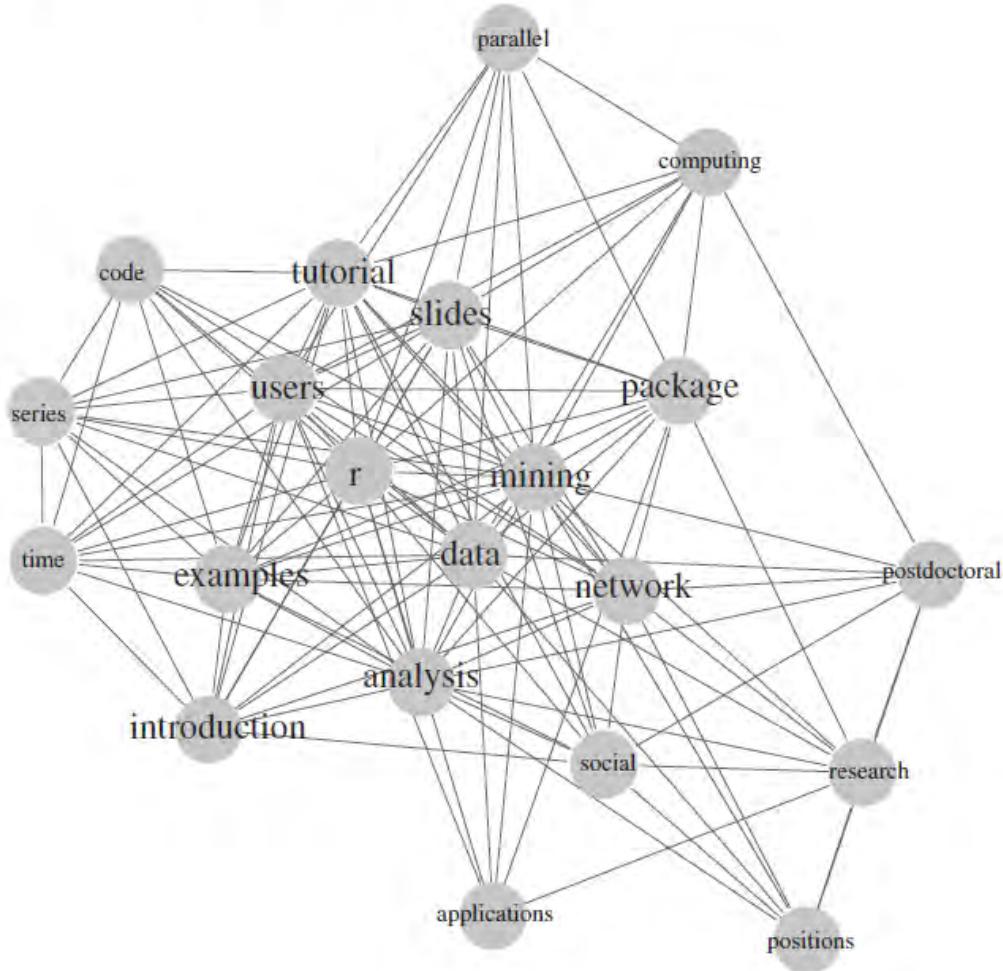
```

ดัดໄປเป็นการกำหนด font ของคำถ้าปรากฏบอย font จะใหญ่กว่าปกติ และให้ปรับแสง เพื่อฉาดเจ้ายืน โดยที่ v() คือโหนด (Vertex) และ E() คือเส้นเชื่อม (Edge), พื้นที่ชัน rgb (red, green, blue, alpha) แสดงสี และ alpha คือระดับความโปร่งใส

```

> V(g)$label.cex <- 2.2 * V(g)$degree / max(V(g)$degree) + .2
> V(g)$label.color <- rgb(0, 0, .2, .8)
> V(g)$frame.color <- NA
> egam <- (log(E(g)$weight) + .4) / max(log(E(g)$weight) + .4)
> E(g)$color <- rgb(.5, .5, 0, egam)
> E(g)$width <- egam
> # plot the graph in layout1
> plot(g, layout=layout1)

```



รูปที่ 11.2

11.2 Network of Tweets

กีอการสร้างเส้นเชื่อมระหว่างเอกสารที่สัมพันธ์กัน (โดยมีคำเดียวกันปรากฏอยู่ในทั้ง 2 เอกสาร) จากการศึกษาที่ผ่านมา (ในข้อมูล tweet ของบทที่ 10) พบว่าเอกสารส่วนใหญ่ จะมีคำ r, data, mining ทำให้เส้นเชื่อมของคำทั้ง 3 จะปรากฏเยอะมากจนยุ่งเหยิง เพื่อความสวยงาม เราจะไม่นำคำทั้ง 3 มาทำงาน แล้วหาเส้นเชื่อมของคำที่ปรากฏบ่อยที่เหลือ

```

> # remove "r", "data" and "mining"

> idx <- which(dimnames(termDocMatrix)$Terms %in% c("r", "data",
  "mining"))

> M <- termDocMatrix[-idx,]

> # build a tweet-tweet adjacency matrix

> tweetMatrix <- t(M) %*% M

> library(igraph)

> g <- graph.adjacency(tweetMatrix, weighted=T,
  mode="undirected")

> V(g)$degree <- degree(g)

> g <- simplify(g)

> # set labels of vertices to tweet IDs

> V(g)$label <- V(g)$name

> V(g)$label.cex <- 1

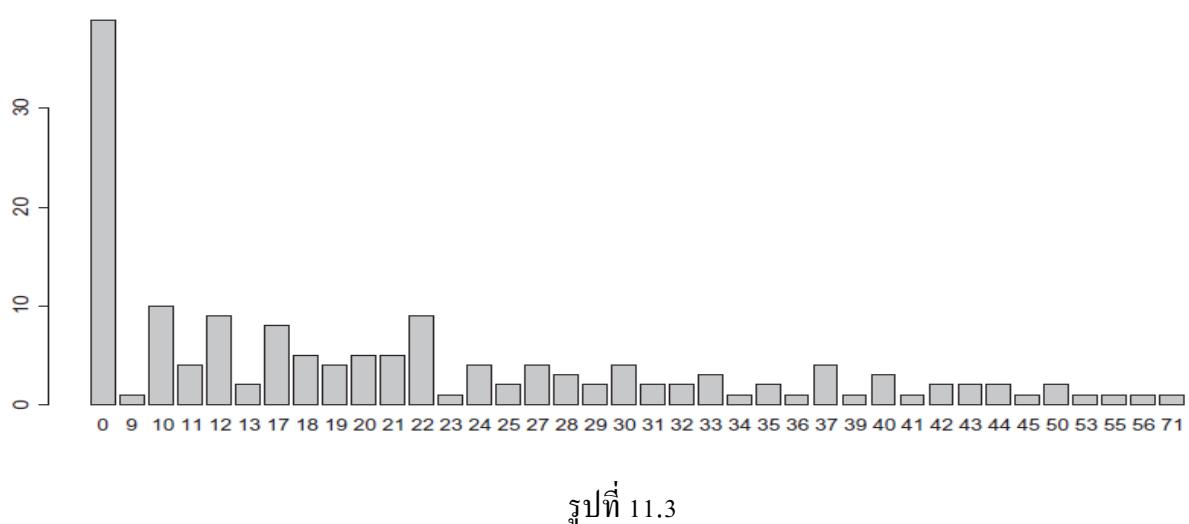
> V(g)$label.color <- rgb(.4, 0, 0, .7)

> V(g)$size <- 2

> V(g)$frame.color <- NA

```

จากรูปจะพบว่ามีประมวล 40 เอกสารซึ่งมีค่าเป็น 0 เพราะมาจากการที่ไม่นำ r, data, mining มาคิด



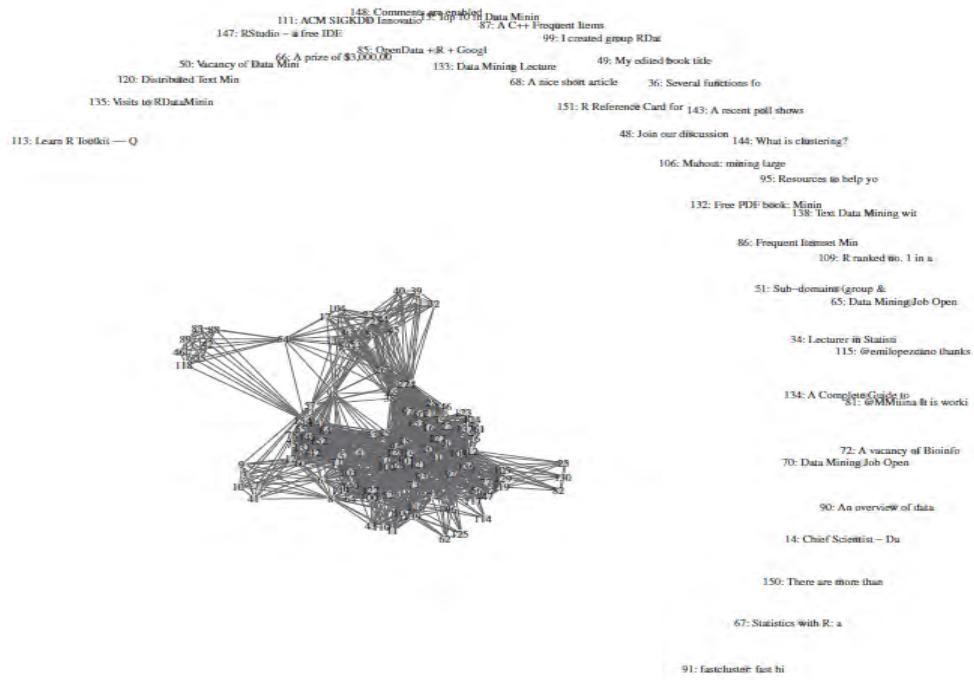
รูปที่ 11.3

จากคำสั่งต่อไปนี้เรากำหนดให้สิ่งเอกสารแทนคำที่ปรากฏ และเอกสารจะถูกระบุคัด้วยหมายเลขทำให้รูปที่ได้ไม่แน่นเกินไป และกำหนด ความกว้างและสิ่งของเส้นเชื่อม ตามระดับความสัมพันธ์ ดังนี้

```

> idx <- V(g)$degree == 0
> V(g)$label.color[idx] <- rgb(0, 0, .3, .7)
> # load twitter text
> library(twitteR)
> load(file = "data/rdmTweets.RData")
> # convert tweets to a data frame
> df <- do.call("rbind", lapply(rdmTweets, as.data.frame))
> # set labels to the IDs and the first 20 characters of tweets
> V(g)$label[idx] <- paste(V(g)$name[idx], substr(df$text[idx], 1, 20), sep=": ")
> egam <- (log(E(g)$weight) + .2) / max(log(E(g)$weight) + .2)
> E(g)$color <- rgb(.5, .5, 0, egam)
> E(g)$width <- egam
> set.seed(3152)
> layout2 <- layout.fruchterman.reingold(g)

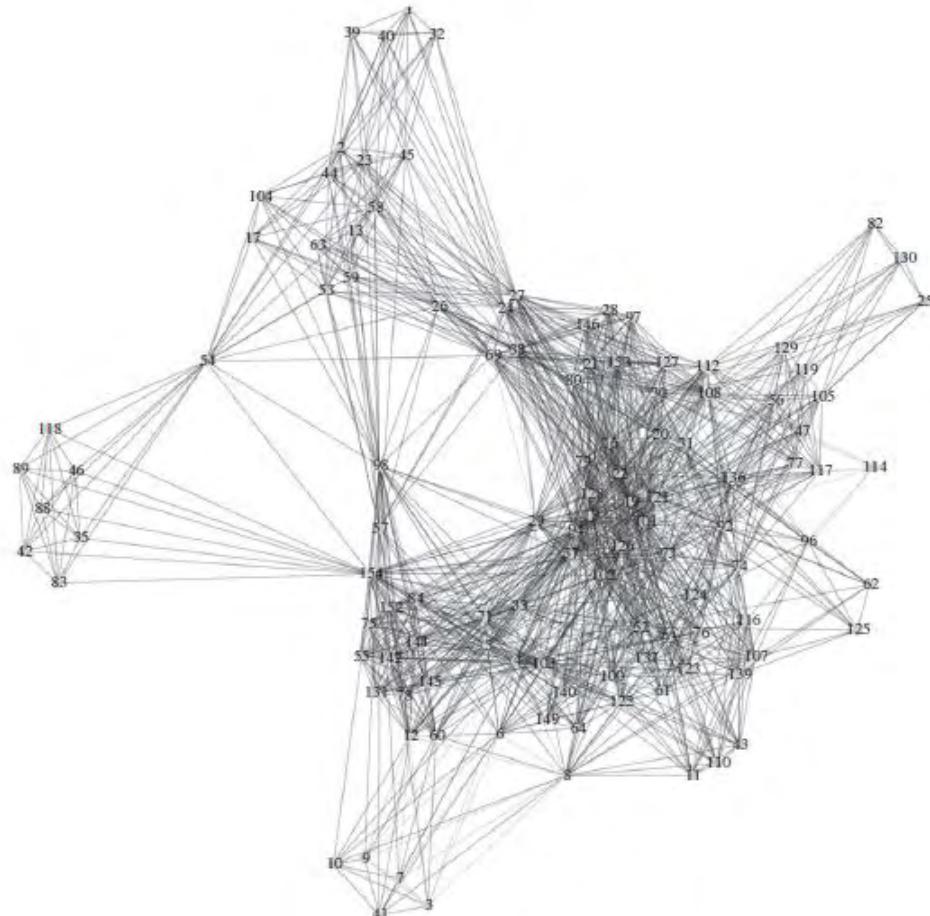
```



Σύμπλοκα 11.4

ปรับปรุงให้ node ที่มีค่าเส้นเชื่อม 0 หายไปโดยใช้คำสั่ง

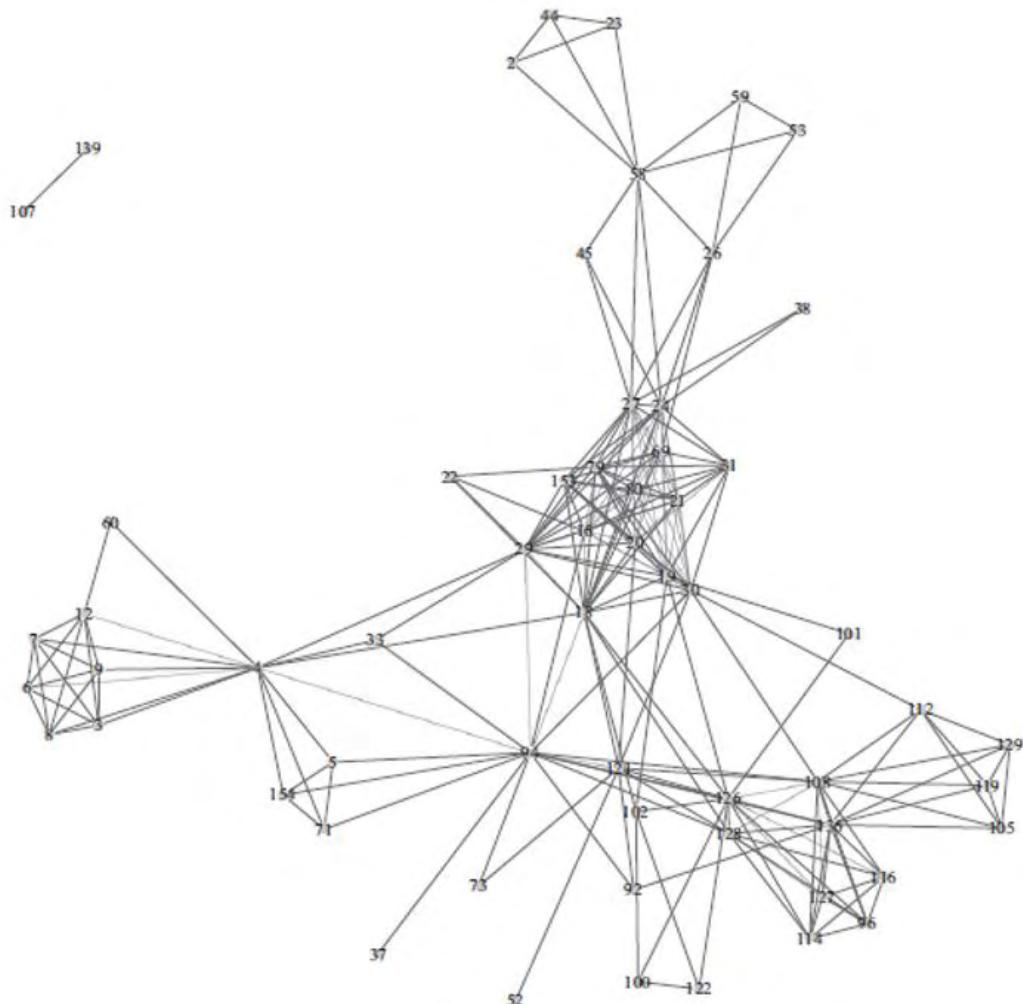
```
> g2 <- delete.vertices(g, V(g) [degree(g)==0] )  
> plot(g2, layout=layout.fruchterman.reingold)
```



จูปที่ 11.5

ปรับปรุงให้ node ที่มีค่าเส้นเชื่อมน้อย (≤ 1) หายไปโดยใช้ delete.edges() และวิจัยลบโหนดนั้น

```
> g3 <- delete.edges(g, E(g) [E(g)$weight <= 1] )  
> g3 <- delete.vertices(g3, V(g3) [degree(g3) == 0] )  
> plot(g3, layout=layout.fruchterman.reingold)
```



รูปที่ 11.6

จากรูป 11.6 จะมีกลุ่มเล็ก ๆ ทางซ้ายคือหมายเลข 7, 12, 6, 9, 8, 3, 4 สมมุติว่าเราสนใจ สามารถดูรายละเอียดต่อไป

```
> df$text[c(7, 12, 6, 9, 8, 3, 4)]  
[7] State of the Art in Parallel Computing with R  
http://t.co/zmC1glqi  
[12] The R Reference Card for Data Mining is updated with  
functions & packages for handling big data & parallel computing.  
http://t.co/FHoVZCYk  
[6] Parallel Computing with R using snow and snowfall  
http://t.co/nxp8EZpv  
[9] R with High Performance Computing: Parallel processing and  
large memory http://t.co/XZ3ZZBRF  
[8] Slides on Parallel Computing in R http://t.co/AdDVxb0Y  
[3] Easier Parallel Computing in R with snowfall and sfCluster  
http://t.co/BPcinvkK  
[4] Tutorial: Parallel computing using R package snowfall  
http://t.co/CHBCvyr76
```

11.3 Two-Mode Network

ເພີ້ມເຕີນ

บทที่ 12 กรณีศึกษา 1 : การวิเคราะห์และพยากรณ์ ดัชนีราคาที่พักอาศัย

HPI (House Price Index) คือดัชนีราคาที่พักอาศัย จะแสดงค่า (ราคาม้าน) ตามเวลาถ้าเวลาผ่านไปนาน แนวโน้มจะสูงขึ้น (หรือต่ำลง) ขึ้นอยู่กับปัจจัยหลายอย่าง แต่จุดประสงค์ของการวิเคราะห์ในที่นี้คือ เมื่อนำราคาม้าน ในอดีตมาพิจารณาจะพยากรณ์ราคาในอนาคตได้อย่างไร

ข้อมูลที่ใช้ทดลองศึกษาคือ ข้อมูลบ้านในรัฐแคนเบอร์拉 ประเทศออสเตรเลียซึ่งได้มาจากการ Residex ในรูป csv file

12.1 การนำข้อมูล HPI มาใช้งาน

เป็นข้อมูลแสดงราคาบ้านทุกสิ้นเดือน จากเดือน January 1990 ถึง January 2011 มีลักษณะข้อมูลดังนี้

31-Jan-90, 1.00763

28-Feb-90, 1.01469

31-Mar-90, 1.02241

30-Apr-90, 1.03062

ขั้นตอนการนำข้อมูลเข้ามาใช้งานใน R คือ

1. ใช้ `read.csv()` เพื่ออ่านจาก csv file
2. กำหนดชื่อให้ทุก colum
3. แปลงวันที่ให้เป็น ปี และ เดือน

```

> # import data
> filepath <- "./data/"
> filename <- "House-index-canberra.csv"
> houseIndex <- read.csv(paste(filepath, filename, sep=""),
  header=FALSE)
> names(houseIndex) <- c("date", "index")
> n <- nrow(houseIndex)
> # check start date and end date
> cat(paste("HPI from", houseIndex$date[1], "to",
  houseIndex$date[n], "\n"))
HPI from 31-Jan-90 to 31-Jan-11
> # extract year and month
> dates <- strptime(houseIndex$date, format="%d-%b-%y")
> houseIndex$year <- dates$year + 1900
> houseIndex$month <- dates$mon + 1

```

การแปลงวันที่อาจจะใช้คำสั่งต่อไปนี้ก็ได้

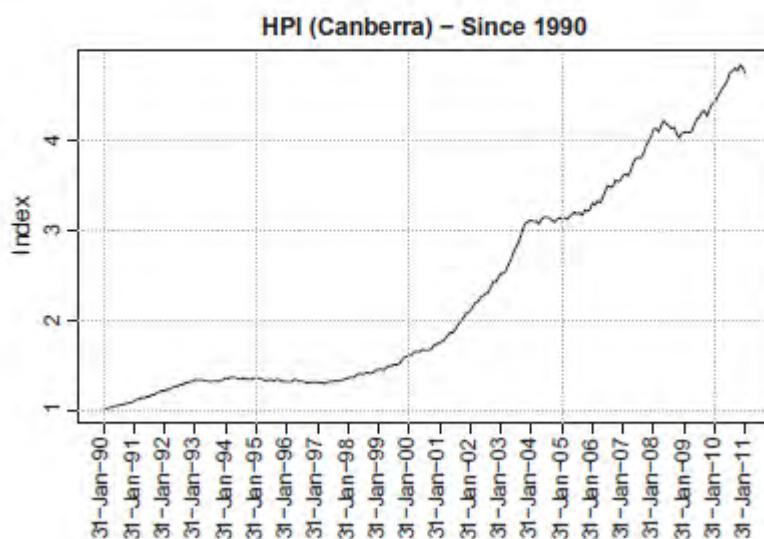
```

> dates <- as.Date(houseIndex$date, format="%d-%b-%y")
> houseIndex$year <- as.numeric(format(dates, "%y"))
> houseIndex$month <- as.numeric(format(dates, "%m"))

```

12.2 การพิจารณาข้อมูล HPI เพื่อคุ้ว่าข้อมูลนี้มีลักษณะทั่วๆ ไปเป็นอย่างไร ดังต่อไปนี้

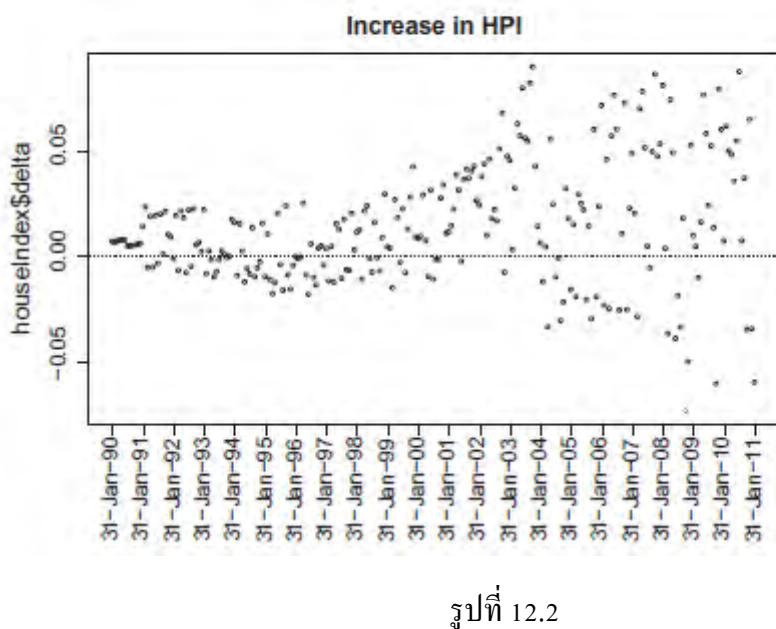
```
> plot(houseIndex$index, pty=1, type="l", lty="solid", xaxt="n",
+       xlab="", ylab="Index",
+       main=paste("HPI(Canberra) - Since ", fromYear, sep=""))
> # draw tick-marks at 31 Jan of every year
> nYear <- ceiling(n/12)
> posEveryYear <- 12 * (1:nYear) - 11
> axis(1, labels=houseIndex$date[posEveryYear], las=3,
+       at=posEveryYear)
> # add horizontal reference lines
> abline(h=1:4, col="gray", lty="dotted")
> # draw a vertical reference line every five years
> posEvery5years <- 12 * (5 * 1:ceiling(nYear/5) - 4) - 11
> abline(v=posEvery5years, col="gray", lty="dotted")
```



รูปที่ 12.1

ต้องการดูว่า ค่าที่เพิ่มขึ้นแต่ละปี มีลักษณะอย่างไร (คำนวณจากค่า $\$delta$)

```
> houseIndex$delta <- houseIndex$index - c(1, houseIndex$index[-n])  
  
> plot(houseIndex$delta, main="Increase in HPI", xaxt="n",  
       xlab="")  
  
> axis(1, labels=houseIndex$date[posEveryYear], las=3,  
       at=posEveryYear)  
  
> # add a reference line  
  
> abline(h=0, lty="dotted")
```



จากรูปที่ 12.2 พบว่าหลังจากปี 2003 จะมีแนวโน้มการเปลี่ยนแปลงราคาในลักษณะกระแทบมากกว่าก่อนหน้าและเมื่อถึงปี 2011 จะมีการกระจายประมาณ 5 เท่าของปี 1990

ในการพิจารณาแบบอื่น ๆ เช่น คุณภาพขึ้น-ลง ของ HPI ในแต่ละเดือน เครื่องหมาย + และคงเพิ่มขึ้น, 0

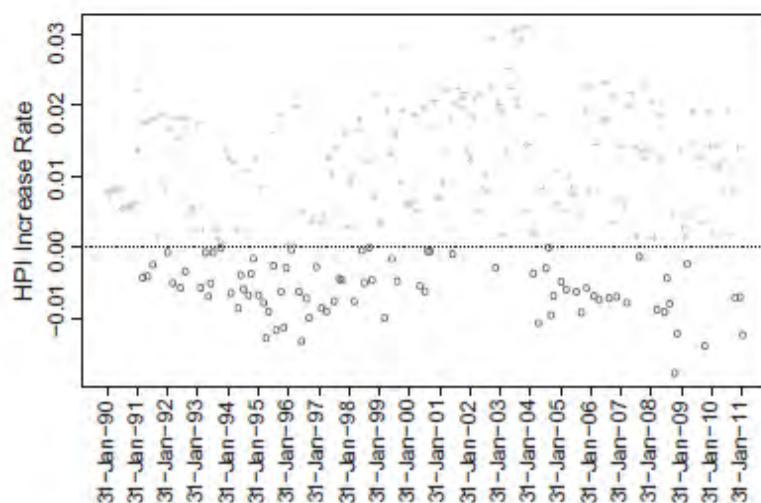
แสดงผลดัง

```

> # increase ratio in every month
> houseIndex$rate<-houseIndex$index/c(1, houseIndex$index[-n])-1
> # percentage of months having positive increases in HPI
> 100 * sum(houseIndex$rate>0)/n
[1] 67.58893

> # use ifelse() to set positive values to green and negative ones to red
> plot(houseIndex$rate, xaxt="n", xlab="", ylab="HPI Increase Rate",
+       col=ifelse(houseIndex$rate>0, "green", "red"),
> axis(1, labels=houseIndex$date[posEveryYear], las=3,
+       at=posEveryYear)
> abline(h=0, lty="dotted")

```



รูปที่ 12.3

รูปที่ 12.3 พบว่า

- มีแนวโน้มเพิ่มขึ้นมากกว่าลดลง
- อัตราการเพิ่มขึ้นประมาณ 0-2 % มากกว่าอัตราการลดลงประมาณ 0-1 %
- มี 2 ช่วงเวลา ที่ลดลงอย่างมากคือ 1995-1996 และ 2008-2009 และ 1 ช่วงเวลาที่เพิ่มขึ้นอย่างมากคือช่วงเวลา 2002-2003

เราสามารถแสดงตารางการเพิ่มขึ้นและ bar chart (แสดงแต่ละก้อน) โดย colum แสดง juxtaposed bar โดยการกำหนด “beside=TRUE” และระยะห่างระหว่างก้อนกำหนดโดย space = c(0,2)

```

> rateMatrix <- xtabs(rate ~ month + year, data=houseIndex)
> # show the first four years, rounded to 4 decimal places
> round(rateMatrix[,1:4], digits=4)

      year
month 1990    1991    1992    1993
  1   0.0076   0.0134 -0.0007   0.0172
  2   0.0070   0.0219   0.0164 -0.0057
  3   0.0076  -0.0043 -0.0050   0.0023
  4   0.0080   0.0174   0.0180 -0.0007
  5   0.0082  -0.0041   0.0151 -0.0069
  6   0.0079   0.0176 -0.0057 -0.0051
  7   0.0052  -0.0025   0.0178 -0.0008
  8   0.0053   0.0179 -0.0034   0.0023
  9   0.0053   0.0013   0.0180   0.0010
 10   0.0055   0.0186   0.0046 -0.0001
 11   0.0058   0.0091   0.0055   0.0004
 12   0.0061   0.0081   0.0021   0.0136

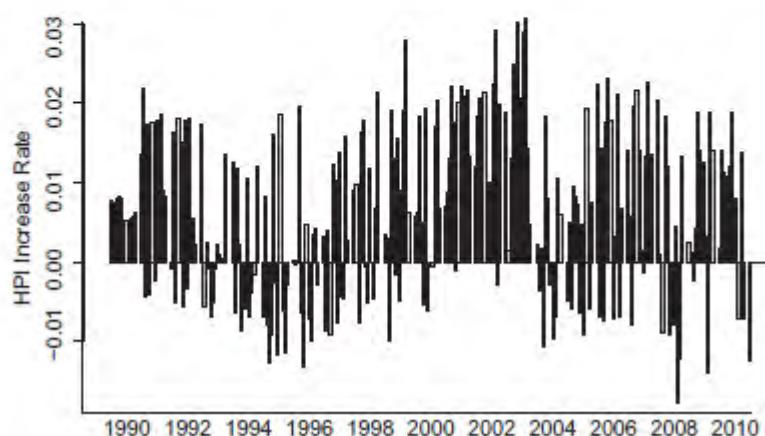
```

> # plot a grouped barchart:

```

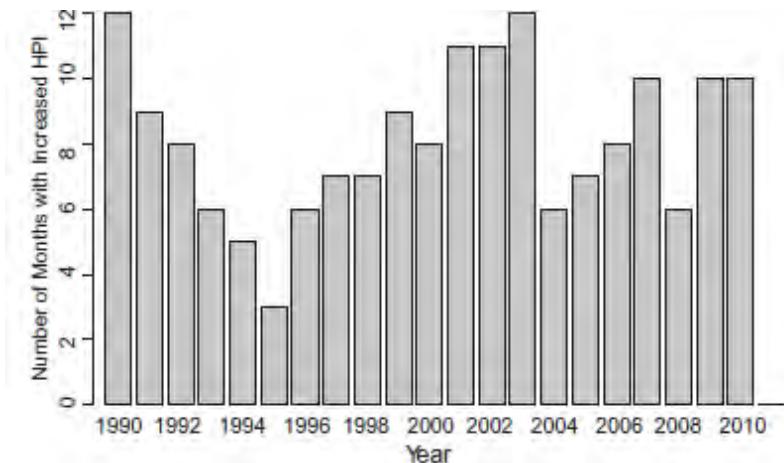
> barplot(rateMatrix, beside=TRUE, space=c(0,2),
+           col=ifelse(rateMatrix>0, "lightgreen", "lightpink"),
+           ylab="HPI Increase Rate", cex.names=1.2)

```

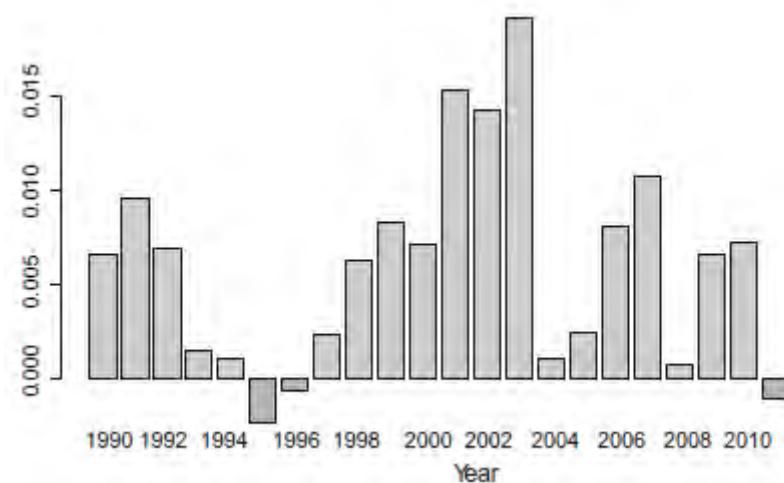


รูปที่ 12.4

รูปที่ 12.5, 12.6, 12.7 แสดงจำนวนเดือนที่มี HPI เพิ่มขึ้น และอัตราการเพิ่มขึ้นเฉลี่ยรายเดือน, รายปี พังก์ชัน colSums(), colMeans(), rowMeans() ใช้คำนวณผลรวมและค่าเฉลี่ยของเมตริกซ์ rateMatrix



รูปที่ 12.5

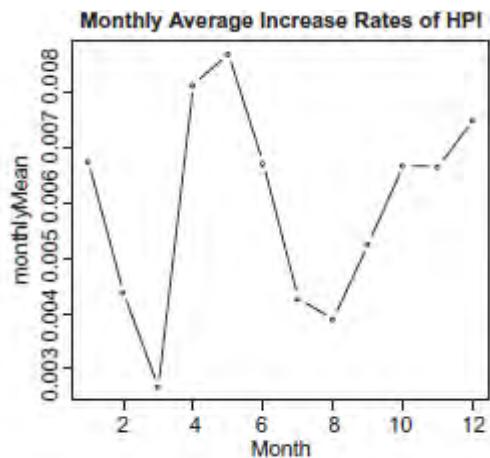


รูปที่ 12.6

```
> plot(names(monthlyMean), monthlyMean, type="b", xlab="Month",
+      main="Monthly Average Increase Rates of HPI")
```

ต่อไปต้องการดูการกระจายของอัตราการเพิ่มขึ้น (distribution of increase rate) โดยใช้ `summary()` และ `boxplot()` เพื่อแสดงกราฟสี่เหลี่ยมที่เรียกว่า box-and-whisker plot ที่มีค่า median, ควาไกล์ที่ 1 และ 3 ซึ่งจะระบะห่างเรียกว่า IQR (Inter Quartile Range)

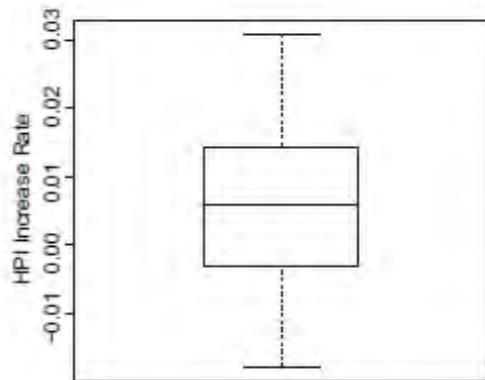
```
> summary(houseIndex$rate)
```



รูปที่ 12.7

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
-0.017710	-0.002896	0.005840	0.006222	0.014210	0.030700

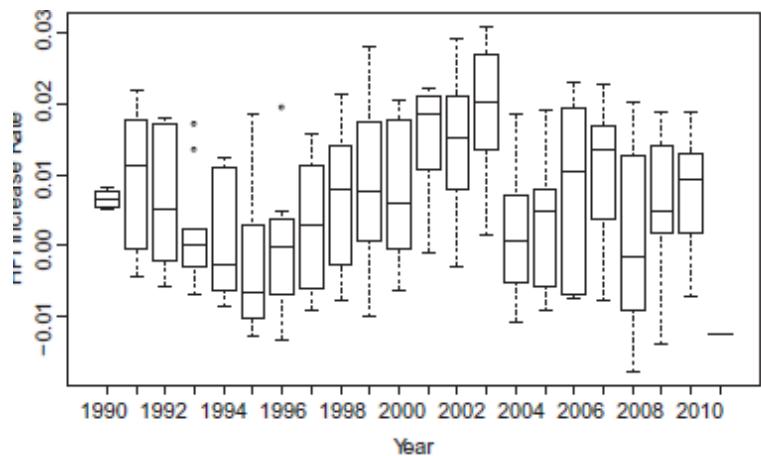
```
> boxplot(houseIndex$rate, ylab="HPT Increase Rate")
```



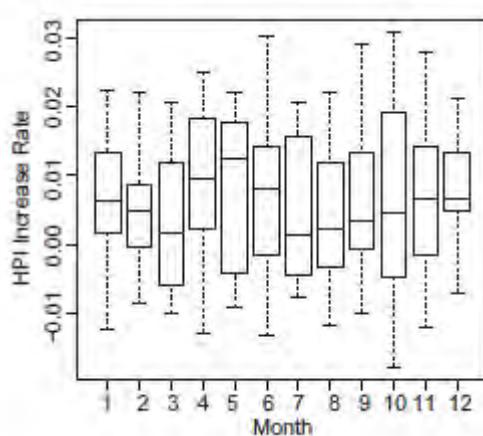
รูปที่ 12.8

ต่อไปต้องการดูการกระจายของอัตราการเพิ่มขึ้น ทุกปี และทุกเดือน ด้วย boxplot

```
> boxplot(rate ~ year, data=houseIndex, xlab="year", ylab="HPI  
Increase Rate")  
  
> boxplot(rate ~ month, data=houseIndex, xlab="month", ylab="HPI  
Increase Rate")
```



รูปที่ 12.9



รูปที่ 12.10

รูปที่ 12.10 แสดงให้เห็นว่า เดือน April และ May จะมีอัตราการเพิ่มอย่างรวดเร็ว เดือนอื่น ๆ ที่มีการเพิ่มขึ้นสูง คือ June, Jan, Dec ส่วนเดือนที่มีอัตราการเพิ่มลดลงคือ เดือน March, July, August

12.3 แนวโน้มและฤดูกาล (Trend and Seasonal Components of HPI)

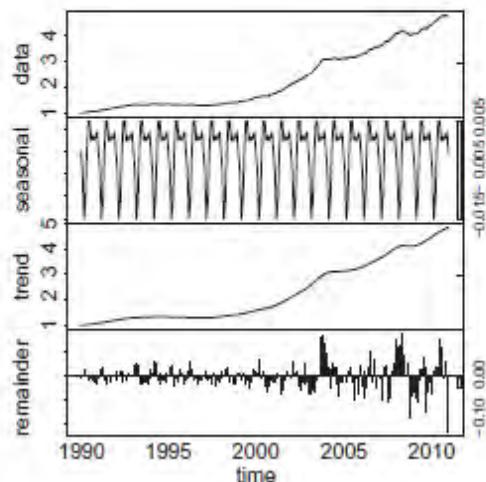
เมื่อพิจารณาลักษณะข้อมูลคร่าว ๆ แล้ว สิ่งที่สำคัญคือการวิเคราะห์องค์ประกอบที่ซ่อนอยู่ภายในข้อมูล คือ การหา Trend และ Seasonal factor แล้วจึงนำไปพยากรณ์อนาคต ด้วยขั้นตอนต่อไปนี้

1. แปลง index ไปเป็น object ชนิด time series ด้วยฟังก์ชัน `ts()`
2. หาองค์ประกอบภายในด้วย `stl()`

```

> hpi <- ts(houseIndex$index, start=c(1990,1), frequency=12)
> f <- stl(hpi, "per")
> plot(f)

```



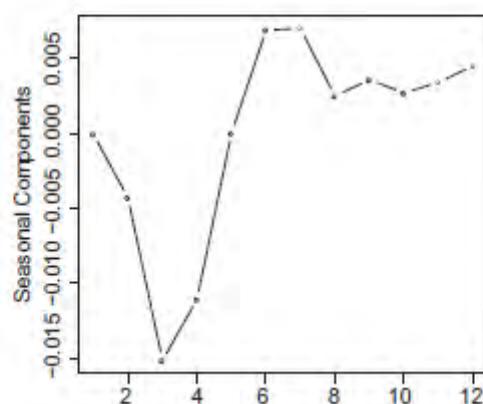
รูปที่ 12.11

3. แสดงองค์ประกอบฤดูกาล ซึ่งคล้ายกับกราฟแสดงอัตราการเพิ่มขึ้นรายเดือน

```

> # plot seasonal components
> plot(f$time.series[1:12, "seasonal"], type='b', xlab="Month",
+       ylab="Seasonal Components")

```



รูปที่ 12.12

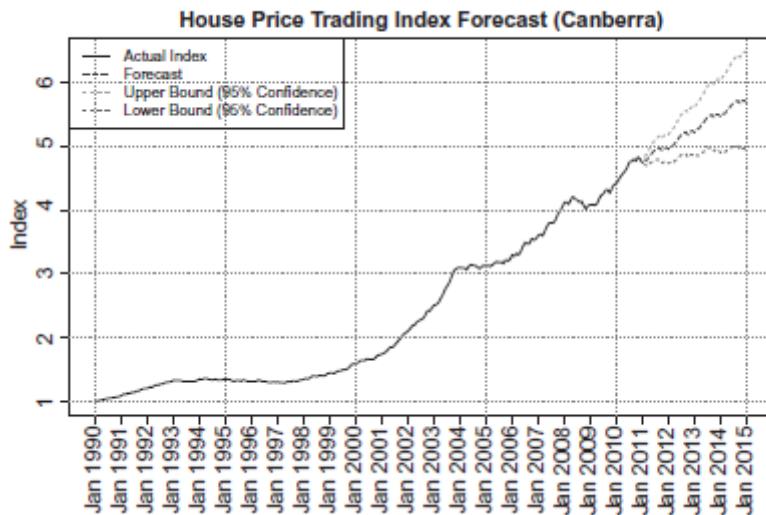
อาจจะใช้ฟังก์ชัน decompose() ในการหาองค์ประกอบได้

```
> # an alternative decomposition function  
> f2 <- decompose(hpi)  
  
> plot(f2)  
  
> # plot seasonal components  
  
> plot(f2$figure, type="b", xlab="Month", ylab="Seasonal Components")
```

12.4 HPI Forcasting

เราจะใช้โมเดล ARIMA (Auto Regressive Integrated Moving Average) เพื่อชิบายข้อมูล HPI และใช้พยากรณ์อนาคตในอีก 4 ปี ข้างหน้า

```
> startYear <- 1990  
  
> endYear <- 2010  
  
> # to forecast HPIs in the next four years  
  
> nYearAhead <- 4  
  
> fit <- arima(hpi, order=c(2,0,1), seasonal=list(order=c(2,1,0),  
+ period=12))  
  
> fore <- predict(fit, n.ahead=12*nYearAhead)  
  
> # error bounds at 95% confidence level  
  
> U <- fore$pred + 2 * fore$se  
  
> L <- fore$pred - 2 * fore$se  
  
> # plot original and predicted data, as well as error bounds  
  
> ts.plot(hpi, fore$pred, U, L, col=c("black",  
+ "blue", "green", "red"),  
+ lty=c(1,5,2,2), gpars=list(xaxt="n", xlab=""),  
+ ylab="Index", main="House Price Trading Index Forecast (Canberra)")  
  
> # add labels, reference grid and legend  
  
> years <- startYear:(endYear + nYearAhead+1)  
  
> axis(1, labels=paste("Jan ", years, sep=""), las=3, at=years)  
> grid()  
  
> legend("topleft", col=c("black", "blue", "green", "red"),  
+ lty=c(1,5,2,2), c("Actual Index", "Forecast", "Upper Bound(95% Confidence)", "Lower Bound(95% Confidence)"))
```



รูปที่ 12.13

ต้องการคูณย่างละเอียดว่าค่าพยากรณ์ตั้งแต่ปี 2011 มีลักษณะอย่างไร ทำได้ดังนี้

```
> ts.plot(fore$pred, U, L, col=c("blue", "green", "red"),
  lty=c(5,2,2), gpars=list(xaxt="n", xlab=""), ylab="Index",
  main="House Price Trading Index Forecast (Canberra)")

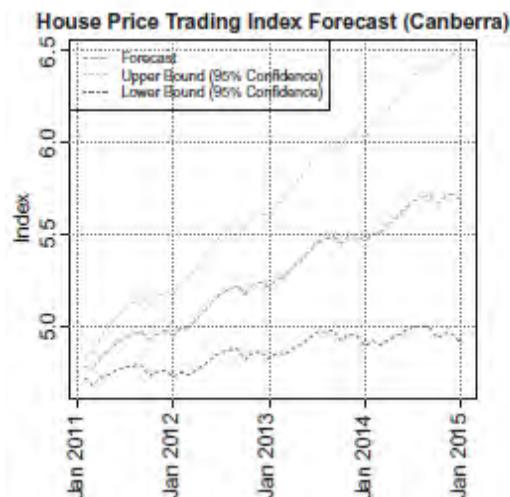
> years <- endYear +(1:(nYearAhead+1))

> axis(1, labels=paste("Jan ", years, sep=""), las=3, at=years)

> grid(col = "gray", lty = "dotted")

> legend("topleft", col=c("blue", "green", "red"), lty=c(5,2,2),
+        c("Forecast", "Upper Bound (95% Confidence)",
+          "Lower Bound (95% Confidence)"))


```



รูปที่ 12.14

12.5 การประมาณค่าบ้าน

สมมุติว่าบ้านหลังหนึ่งมีราคา 535,000 ดอลลาร์ เมื่อ September 2001 และต้องการทราบว่าอีก 2 ปี ต่อมาจะมีราคาเท่าไร ทำได้ดังนี้

```
> newHpi <- ts(c(hpi, fore$pred), start=c(1990,1), frequency=12)
> (startDate <- start(newHpi) )
[1] 1990 1
> startYear <- startDate[1]
> m <- 9 + (2009-startYear)*12
> n <- 9 + (2011-startYear)*12
> # percentage of increase
> 100 * (newHpi[n]/ newHpi[m] - 1)
[1] 15.15576
> round(535000 * newHpi[n]/ newHpi[m])
[1] 616083
```

12.6 บทสรุป

ในบทนี้เป็นการวิเคราะห์ Time Series ของข้อมูลราคาบ้านของเมืองหนึ่ง สิ่งที่น่าสนใจคือ การเปรียบเทียบกับเมืองอื่น ๆ ผลที่ได้อาจจะมีรูปแบบที่คล้าย ๆ กันในแต่ละช่วงเวลา, อาจจะมีความสัมพันธ์ของราคาระหว่างเมือง, อาจจะมีรูปแบบเหมือนกันเมื่อเวลาหนึ่ง (lag) นอกจากนี้อาจจะมีตัวแปรอื่นที่เข้ามาเกี่ยวข้องเช่น สภาพแวดล้อมทางเศรษฐกิจ, การเปลี่ยนแปลงประชากร, นโยบายรัฐบาล

บทที่ 13 กรณีศึกษา 2 : พยากรณ์การตอบสนองของลูกค้าและผลประโยชน์ที่สูงที่สุด

(Customer Response Prediction and Profit Optimization)

บทนำ

เราจะใช้ข้อมูลจาก KDD Cup 1998 เพื่อใช้พยากรณ์การตอบสนองของลูกค้าเพื่อให้มีกำไรสูงที่สุด โดยใช้ decision tree และได้มีการนำไปใช้งานจริงในทางธุรกิจ รายละเอียดคือ ต้องการพยากรณ์ว่า จะมีคนตอบจดหมายจำนวนเท่าไร จากการส่งข้อมูลรับบริจาคออไป โดยข้อมูลของบุคคลคือ demographic และประวัติการซื้อ เงินเดือน ถ้าพยากรณ์ได้ใกล้เคียงกับความจริงจะสามารถคำนวณยอดเงินบริจาคได้ดีที่สุด

13.1 ลักษณะข้อมูล

ข้อมูลหลักอยู่ในไฟล์ cup98LRN.txt มี 95,412 เรคคอร์ด และ 481 คอลัมน์ ข้อมูลจะมี comma คั่นและมี cup98VAL.txt เป็นกลุ่มข้อมูลทดสอบ (test, validate) ประกอบด้วย 96,367 เรคคอร์ด และ 479 คอลัมน์ ข้อมูลหลักมี column ที่สำคัญคือ

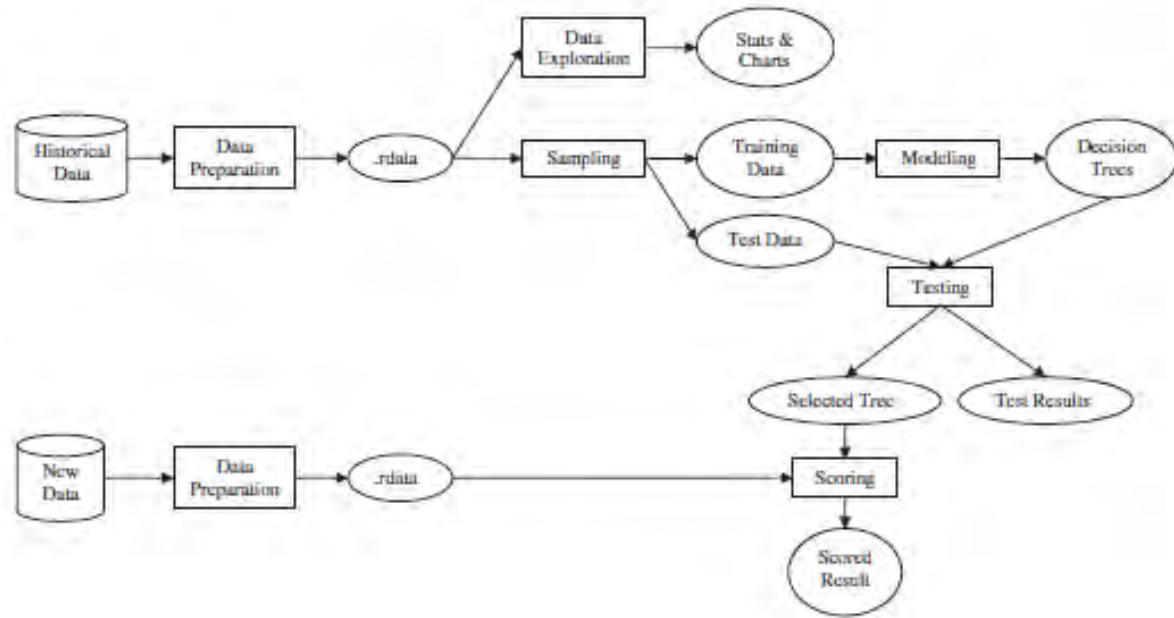
CONTROLN เป็นค่า unique ID

TARGET_B มี 2 ค่าคือ 0, 1 แสดงสถานะว่ามีการตอบจดหมายหรือไม่

TARGET_D เป็นค่าเงินบริจาค

ข้อมูลทดสอบ จะไม่มี TARGET_B และ TARGET_D

รูปที่ 13.1 แสดง กระบวนการทำ data mining



คำสั่งต่อไปนี้เป็นการโหลดข้อมูลและพิจารณาข้อมูลทั่วๆไป

```
> cup98 <- read.csv("./data/KDDCup1998/cup98LRN.txt")
> dim(cup98)
[1] 95412 481
> # have a look at the first 30 variables
> str(cup98[,1:30])
'data.frame': 95412 obs. of 30 variables:

$ ODATEDW: int 8901 9401 9001 8701 8601 9401 8701 9401 8801 9401 ...
$ OSOURCE: Factor w/ 896 levels " ","AAA","AAD",...: 343 122 50 128
1 220 255 613 487 549 ...
$ TCODE : int 0 1 1 0 0 0 0 0 1 1 ...
$ STATE : Factor w/ 57 levels "AA","AE","AK",...: 20 9 33 9 14 4
21 24 18 48 ...
$ ZIP : Factor w/ 19938 levels "00801","00802",...: 9940 16858 336
18629 2937 3841 5897 12146 7439 4251 ...
$ MAILCODE: Factor w/ 2 levels " ","B": 1 1 1 1 1 1 1 1 1 1 ...
$ PVASTATE: Factor w/ 3 levels " ","E","P": 1 1 1 1 1 1 1 1 1 1 ...
$ DOB : int 3712 5202 0 2801 2001 0 6001 0 0 3211 ...
$ NOEXCH : Factor w/ 4 levels " ","0","1","X": 2 2 2 2 2 2 2 2 2 2
2 ...
$ RECINHSE: Factor w/ 2 levels " ","X": 1 1 1 1 2 1 1 1 1 1 ...
$ RECP3 : Factor w/ 2 levels " ","X": 1 1 1 1 2 1 1 1 1 1 ...
$ RECPGVG: Factor w/ 2 levels " ","X": 1 1 1 1 1 1 1 1 1 1 ...
$ RECSWEEP: Factor w/ 2 levels " ","X": 1 1 1 1 1 1 1 1 1 1 ...
$ MDMAUD : Factor w/ 28 levels "C1CM","C1LM",...: 28 28 28 28 28
28 28 28 28 ...
$ DOMAIN : Factor w/ 17 levels " ","C1","C2",...: 12 8 6 6 9 12 12
12 6 11 ...
$ CLUSTER: int 36 14 43 44 16 40 40 39 45 35 ...
$ AGE     : int 60 46 NA 70 78 NA 38 NA NA 65 ...
$ AGEFLAG: Factor w/ 3 levels " ","E","I": 1 2 1 2 2 1 2 1 1 3 ...
$ HOMEOWNR: Factor w/ 3 levels " ","H","U": 1 2 3 3 2 1 2 3 3 1 ...
$ CHILD03: Factor w/ 4 levels " ","B","F","M": 1 1 1 1 1 1 1 1 1
1 ...
$ CHILD07: Factor w/ 4 levels " ","B","F","M": 1 1 1 1 1 1 1 1 1
1 ...
$ CHILD12: Factor w/ 4 levels " ","B","F","M": 1 1 1 1 1 1 3 1 1
1 ...
$ CHILD18: Factor w/ 4 levels " ","B","F","M": 1 4 1 1 1 1 1 1 1
1 ...
$ NUMCHLD: int NA 1 NA NA 1 NA 1 NA NA NA ...
```

```

$ INCOME : int NA 6 3 1 3 NA 4 2 3 NA ...
$ GENDER : Factor w/ 7 levels " ","A","C","F",...: 4 6 6 4 4 1 4 4
6 6 ...
$ WEALTH1: int NA 9 1 4 2 NA 6 9 2 NA ...
$ HIT     : int 0 16 2 2 60 0 0 1 0 0 ...
$ MBCRAFT: int NA 0 0 0 1 NA NA 0 NA NA ...
$ MBGARDEN: int NA 0 0 0 0 NA NA 0 NA NA ...
> head(cup98[,1:30])

```

	ODATEDW	OSOURCE	TCODE	STATE	ZIP	MAILCODE	PVASTATE	DOB	NOEXCH	RECINHSE
1	8901	GRI	0	IL	61081			3712	0	
2	9401	BOA	1	CA	91326			5202	0	
3	9001	AMH	1	NC	27017			0	0	
4	8701	BRY	0	CA	95953			2801	0	
5	8601		0	FL	33176			2001	0	X
6	9401	CWR	0	AL	35603			0	0	

	RECP3	RECPVG	RECSWEEP	MDMAUD	DOMAIN	CLUSTER	AGE	AGEFLAG	HOMENR	CHILD03
1				XXXX	T2	36	60			
2				XXXX	S1	14	46	E	H	
3				XXXX	R2	43	NA		U	
4				XXXX	R2	44	70	E	U	
5	X			XXXX	S2	16	78	E	H	
6				XXXX	T2	40	NA			

	CHILD07	CHILD12	CHILD18	NUMCHLD	INCOME	GENDER	WEALTH1	HIT	MBCRAFT	MBGARDEN
1				NA	NA	F	NA	0	NA	NA
2				M	1	6	M	9 16	0	0
3				NA	3	M	1	2	0	0
4				NA	1	F	4	2	0	0
5				1	3	F	2	60	1	0
6				NA	NA		NA	0	NA	NA

```
> # a summary of the first 10 variables
```

```
> summary(cup98[,1:10])
```

ODATEDW	OSOURCE	TCODE	STATE
Min.	:8306	MBC	:4539
1st Qu.	:8801	SYN	:3563
Median	:9201	AML	:3430
Mean	:9141	BHG	:3324
3rd Qu.	:9501	IMP	:2986
Max.	:9701	ARG	:2409
		Min.	: 0.00
		1st Qu.	: 0.00
		Median	: 1.00
		Mean	: 54.22
		3rd Qu.	: 2.00
		Max.	: 72002.00
			NC :4160

ZIP	MAILCODE	PVASTATE	DOB	NOEXCH	RECINHSE
85351	:61	:94013	:93954	Min. : 0	: 7 : 88709
92653	:59	B	: 1399	E : 5 1st Qu. : 201	0: 95085 X: 6703
85710	:54		P : 1453	Median : 2610	1: 285
95608	:50			Mean : 2724	X: 35
60619	:45			3rd Qu. : 4601	
89117	:45			Max. : 9710	
(Other)	:95098				

ในการดูตัวแปร (คอลัมน์) ทั้งหมด ทำดังนี้

```
> library(Hmisc)

> describe(cup98[,1:28]) # demographics

> describe(cup98[,29:42]) # number of times responded to other
  types of mail order offers

> describe(cup98[,43:55]) # overlay data

> describe(cup98[,56:74]) # donor interests

> describe(cup98[,75]) # PEP star RFA status

> describe(cup98[,76:361]) # characteristics of the donors
  neighborhood

> describe(cup98[,362:407])# promotion history

> describe(cup98[,408:412])# summary variables of promotion
  history

> describe(cup98[,413:456])# giving history

> describe(cup98[,457:469])# summary variables of giving history

> describe(cup98[,470:473])# ID & targets

> describe(cup98[,474:479])# RFA (Recency/Frequency/Donation
  Amount)

> describe(cup98[,480:481])# CLUSTER & GEOCODE
```

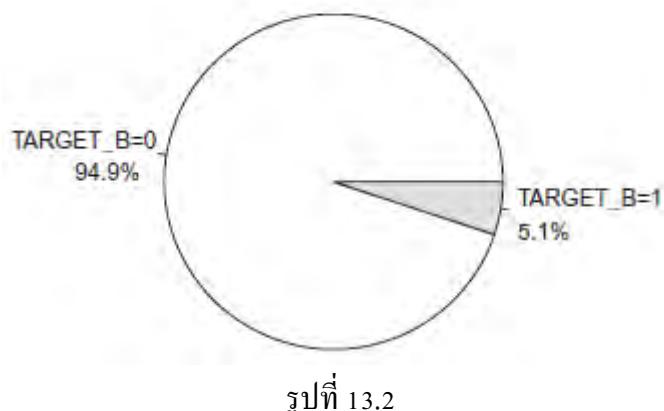
ดูกราฟ pie ของข้อมูล TARGET_B และ TARGET_D ดังรูปที่ 13.2

```
> (response.percentage <- round(100 * prop.table(table(cup98$TARGET_B)), digits=1))

  0      1
94.9   5.1

> mylabels <- paste("TARGET_B=", names(response.percentage), "\n",
+                   response.percentage, "%", sep=" ")

> pie(response.percentage, labels=mylabels)
```

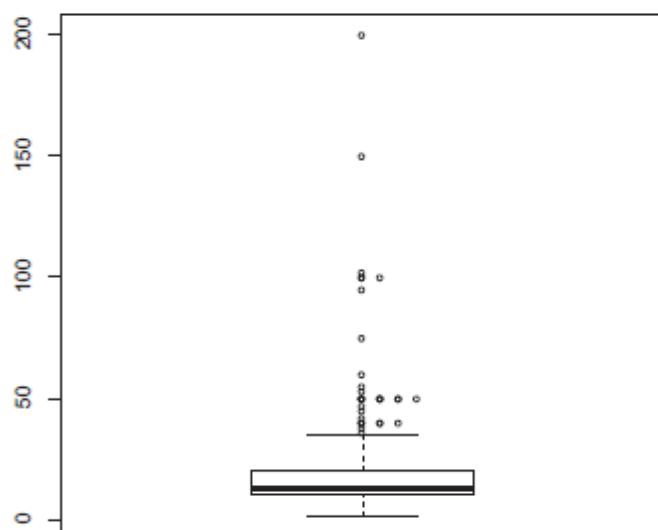


คุณลักษณะอิยคการบวิจักโดยกำหนด TARGET_D มากกว่า 0

```
> # data with positive donations
> cup98pos <- cup98[cup98$TARGET_D >0, ]
> targetPos <- cup98pos$TARGET_D
> summary(targetPos)
```

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
1.00	10.00	13.00	15.62	20.00	200.00

```
> boxplot(targetPos)
```



รูปที่ 13.3

จากการแสดงกราฟยอดเงินบวิจัก พบร่วงส่วนใหญ่ บวิจักไม่เกิน \$25 และเป็นทวีคูณของ 5

```

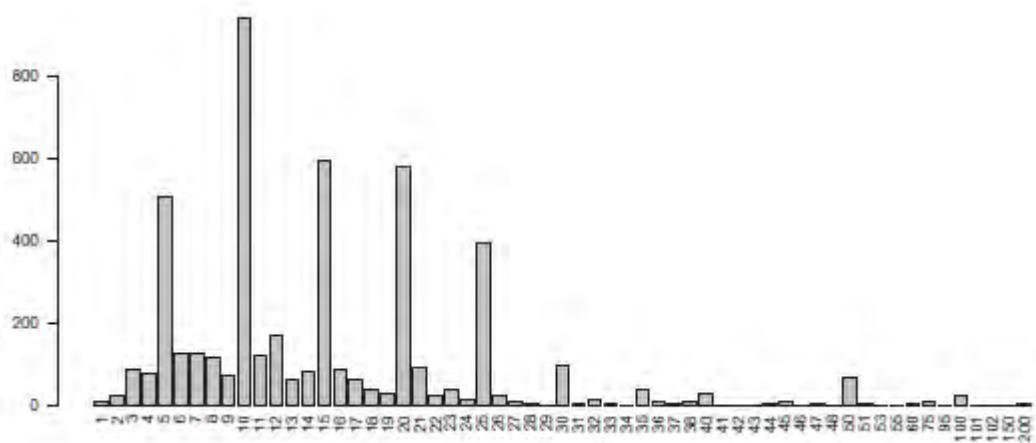
> # number of positive donations
> length(targetPos)
[1] 4843

> # number of positive donations not in whole dollars
> sum(!(targetPos %in% 1:200))
[1] 21

> targetPos <- round(targetPos)

> barplot(table(targetPos), las=2)

```



รูปที่ 13.4

จากกราฟ bar เราจะกำหนดช่วง (discretize) TARGET_D เพื่อมีขอบเขตชัดเจน จะได้ค่า TARGET_D2 โดย cut() ใช้ทำ discretize และ right = F คือระบุช่วงเปิดด้านขวา (ปิดด้านซ้าย)

```

> cup98$TARGET_D2 <- cut(cup98$TARGET_D, right=F,
  breaks=c(0, 0.1, 10, 15, 20, 25, 30, 50, max(cup98$TARGET_D)))
> table(cup98$TARGET_D2)
  [0,0.1)  [0.1,10)  [10,15)  [15,20)  [20,25)  [25,30)  [30,50)  [50,200)
  90569     1132     1378      806      745      435      233      110
> cup98pos$TARGET_D2 <- cut(cup98pos$TARGET_D, right=F,
  breaks=c(0, 0.1, 10, 15, 20, 25, 30, 50, max(cup98pos$TARGET_D)))

```

(เรียนหน้า 175-176)

13.3 Data Exploration-13.8

(กำลังดำเนินการ)

บทที่ 14 กรณีศึกษา 3 : การสร้างโมเดลเพื่อใช้พยากรณ์ข้อมูลขนาดใหญ่ที่ใช้หน่วยความจำน้อย

(Predictive Modeling of Big Data with Limited Memory)

เมื่อข้อมูล Training มีขนาดใหญ่ ไม่สามารถสร้างโมเดลได้ง่ายพระหน่วยความจำมีจำกัดวิธีที่นิยมใช้คือ สุ่มข้อมูลมาหลาย ๆ ชุด แต่ละชุดนำมาสร้างโมเดล (เช่น Decision tree) และวิจัยนำ Attribute ที่ปรากฏในโมเดลไปใช้เลือกคอลัมน์จากข้อมูลตั้งต้น ซึ่งเป็นการทำ feature selection ในขั้นการทดสอบโมเดลก็ใช้วิธีทำงานองเดียวกันคือ แบ่งออกเป็นข้อมูลหลาย ๆ ชุด โดยวิธีสุ่ม ซึ่งจำเป็นต่อการใช้หน่วยความจำที่มีจำกัด ในบทนี้ จะมีการสร้างกฎเกณฑ์ที่ได้จากโมเดลแสดงเป็นประโยชน์ภาษาอังกฤษเพื่อสะดวกในการนำไปใช้งาน

14.1 บทนำ

ข้อมูลที่ใช้ในการศึกษากือ KDD Cup 1998 ซึ่งเป็นข้อมูลชุดเดียวกับที่ใช้ในบทที่ 13 แต่ในบทนี้จะเน้นวิธีที่ทำในลักษณะเป็นข้อมูลขนาดใหญ่และใช้หน่วยความจำขนาดจำกัด ข้อมูลคือ

1. Cup98LRN.txt มี 95,412 เรคคอร์ด และ 481 คอลัมน์
2. Cup98VAL.txt มี 96,367 เรคคอร์ด และ 479 คอลัมน์

ข้อมูลมีทั้งชนิด numeric และ nominal

วัตถุประสงค์ในบทนี้คือการพยากรณ์ว่าคนจะบริจาครหัสไป (คอลัมน์ TARGET_B) ซึ่งมีค่า 0,1 วัตถุประสงค์นี้คล้ายกับการพยากรณ์ความน่าจะเป็น เครื่องมือที่ใช้ทดสอบคือเครื่อง PC ใช้ Windows XP มี CPU กือ Intel dual core i5 3.1 GHz และหน่วยความจำ 4GB

14.2 วิธีดำเนินการ

คอลัมน์เป้าหมาย (target class) คุณระบุด้วย 0 หรือ 1 คล้ายกับการวิเคราะห์โมเดลความเสี่ยง (risk modeling)

ในบทนี้จะมีการใช้ decision tree เพราะง่ายในการทำความเข้าใจ และการจัดการ (เมื่อเทียบกับ SVM หรือ Neural Network) ซึ่งสามารถจัดการได้ทั้งค่า numeric และ nominal นอกจากนี้ยังสามารถจัดการกับข้อมูลสูญหาย (Missing data) หลักการสร้างโมเดล กือ

1. สุ่มข้อมูลจาก Training Data จำนวน 20 ชุด
2. สร้างโมเดล decision tree จากข้อมูล ข้อ 1 จะได้ 20 trees
3. คอลัมน์ที่ปรากฏอยู่ใน decision tree จากข้อ 2 กือ คอลัมน์ที่ต้องการ
4. นำข้อมูล Training ดึงเดิมมาและตัดคอลัมน์อื่น ๆ ทิ้ง ยกเว้นคอลัมน์ที่ต้องการ
5. นำข้อมูล ข้อ 4 มาสร้างโมเดลซึ่งเป็นโมเดลที่นำໄไปใช้จริง

14.3 Data and Variables

ต่อไปนี้เป็นการทดลองด้วยภาษา R โดยการโหลดข้อมูล training และทำการเลือกคอลัมน์ที่ต้องการ

```
> cup98 <- read.csv("./data/KDDCup1998/cup98LRN.txt")  
  
> dim(cup98)  
  
[1] 95412 481  
  
> n.missing <- rowSums(is.na(cup98))  
  
> sum(n.missing > 0)  
  
[1] 95412  
  
> varSet <- c(  
  
+ # demographics  
  
+ "ODATEDW", "OSOURCE", "STATE", "ZIP", "PVASTATE", "DOB",  
"RECINHSE", "MDMAUD", "DOMAIN", "CLUSTER", "AGE", "HOMEOWNR",  
"CHILD03", "CHILD07", "CHILD12", "CHILD18", "NUMCHLD",  
"INCOME", "GENDER", "WEALTH1", "HIT",  
  
+ # donor interests  
  
+ "COLLECT1", "VETERANS", "BIBLE", "CATLG", "HOMEE", "PETS",  
"CDPLAY", "STEREO", "PCOWNERS", "PHOTO", "CRAFTS", "FISHER",  
"GARDENIN", "BOATS", "WALKER", "KIDSTUFF", "CARDS", "PLATES",  
  
+ # PEP star RFA status  
  
+ "PEPSTRFL",  
  
+ # summary variables of promotion history  
  
+ "CARDPROM", "MAXADATE", "NUMPROM", "CARDPM12", "NUMPRM12",  
  
+ # summary variables of giving history  
  
+ "RAMNTALL", "NGIFTALL", "CARDGIFT", "MINRAMNT", "MAXRAMNT",  
"LASTGIFT", "LASTDATE", "FISTDATE", "TIMELAG", "AVGGIFT",  
  
+ # ID & targets  
  
+ "CONTROLN", "TARGET_B", "TARGET_D", "HPHONE_D",  
  
+ # RFA (Recency/Frequency/Donation Amount)  
  
+ "RFA_2F", "RFA_2A", "MDMAUD_R", "MDMAUD_F", "MDMAUD_A",  
  
+ # others  
  
+ "CLUSTER2", "GEOCODE2")  
  
> # remove ID & TARGET_D  
  
> vars <- setdiff(varSet, c("CONTROLN", "TARGET_D"))  
  
> cup98 <- cup98[, vars]
```

14.4 Random Forest (กลุ่มต้นไม้)

คือการสร้างกลุ่มต้นไม้ตัดสินใจเพื่อนำผลลัพธ์ที่ได้บางอย่างไปใช้งาน (ในที่นี้จะใช้คอลัมน์ที่ปรากฏไปใช้งาน) มี 2 packages ที่ทำงานเกี่ยวกับ random forest คือ randomForest และ Party

randomForest ไม่สามารถจัดการ missing value หรือคอลัมน์ที่เป็น categorical มีค่าแตกต่างกันมากกว่า 32 ค่า (32 leads) วิธีแก้ไขคือถ้ามากกว่า 32 ค่า ให้ จัดกลุ่มรวมกันก่อนแล้วค่อยทำ

```
> library(randomForest)  
> rf <- randomForest(TARGET_B ~ ., data=cup98)
```

ต่อไปนี้เป็นการตรวจสอบ missing value และ categorical variable

```
> # check missing values  
> n.missing <- rowSums(is.na(cup98))  
> (tab.missing <- table(n.missing))  
  
n.missing  
0           1           2           3           4           5           6           7  
6782      36864     23841     13684     11716     2483      41       1  
  
> # percentage of records without missing values  
> round(tab.missing["0"] / nrow(cup98), digits=2)  
  
0  
0.07  
  
> # check levels of categorical variables  
> idx.cat <- which(sapply(cup98, is.factor))  
> all.levels <- sapply(names(idx.cat), function(x)  
  nlevels(cup98[,x]))  
  
> all.levels[all.levels > 10]  
  
OSOURCE   STATE    ZIP      MDMAUD  DOMAIN  
896       57      19938    28        17
```

ต่อไปนี้เป็นการแบ่งข้อมูลออกเป็น training และ test set

```

> trainPercentage <- 80
> testPercentage <- 20
> ind <- sample(2, nrow(cup98), replace=TRUE,
+                 prob=c(trainPercentage, testPercentage))
> trainData <- cup98[ind==1,]
> testData <- cup98[ind==2,]

```

แล้วจึงเรียกใช้ cforest() ซึ่งเป็นฟังก์ชันสร้างไมเดล random forest จากข้อมูล training (อยู่ใน package party)

```

> # cforest
> library(party)
> cf <- cforest(TARGET_B ~ ., data=trainData,
+                 control = cforest_unbiased(mtry = 2, ntree = 50))
> (time2 <- Sys.time())
> time2 - time1
> print(object.size(cf), units = "Mb")
> myPrediction <- predict(cf, newdata=testData)
> (time3 <- Sys.time())
> time3 - time2

```

14.5 เกี่ยวกับหน่วยความจำ

ต่อไปนี้ เราจะสร้าง decision tree ด้วย ฟังก์ชัน ctree() จาก package party

```

> memory.limit(4095)
[1] 4095
> library(party)
> ct <- ctree(TARGET_B ~ ., data=trainData)

```

ฟังก์ชัน memory.limit() เป็นการกำหนดหน่วยความจำที่ใช้

ฟังก์ชัน memory.size() เป็นการกำหนดขนาดของความจำที่ใช้

ฟังก์ชัน memory.profile() เป็นการกำหนดหน่วยความจำที่ใช้โดยทั่วไป

เมื่อ run โปรแกรมอาจจะเกิดข้อผิดพลาดแจ้งเตือนดังนี้

Error : cannot allocate vector of size 652 Mb

จึงต้องพยายามลดขนาดข้อมูลในแนวเร็คอร์ดหรือแนวคอลัมน์แต่การทำงานจะยุ่งยากขึ้น

14.6 การสร้างโมเดลจากข้อมูลสุ่ม

จุดประสงค์คือต้องการเลือกเฉพาะ columน์ (variable) ที่สำคัญมาใช้กับข้อมูลดังเดิม ในที่นี้จะสุ่มข้อมูลจำนวน 10 ชุด แล้วสร้าง decision tree 10 ตัว แล้วพิจารณาว่ามี columน์อะไรปรากฏในทรีจึงนำเฉพาะ columน์เหล่านั้นมาใช้กับข้อมูลดังเดิมในการหา decision tree ที่ต้องการจริง ๆ

วิธีการแบ่งข้อมูลคือ สุ่มออกเป็น 3 ส่วน

1. Training data = 30 %
2. Test data = 20 %
3. ที่เหลือ = 50 %

เหตุผลที่ training + test data มีเพียง 50 % เพราะจะได้ไม่ลื้นหน่วยความจำ

```
> library(party) # for ctree  
> trainPercentage <- 30  
> testPercentage <- 20  
> restPercentage <- 100 - trainPercentage - testPercentage  
> fileName <- paste("cup98-ctree", trainPercentage,  
testPercentage, sep="-")  
> vars <- setdiff(varSet, c("TARGET_D", "CONTROLN", "ZIP",  
"OSOURCE"))  
> # partition the data into training and test datasets  
> ind <- sample(3, nrow(cup98), replace=T,  
+ prob=c(trainPercentage, testPercentage, restPercentage))  
> trainData <- cup98[ind==1, vars]  
> testData <- cup98[ind==2, vars]
```

ต่อไปเป็นการตรวจสอบการกระจายของ target class ในทั้ง training data และ test data ว่ามีการกระจายเหมือนข้อมูลดังเดิมหรือไม่ ถ้าไม่อาจจะต้องทำการสุ่มใหม่ โดยใช้วิธีสุ่มแบบ stratified sampling (แบบคำนับชั้น)

```

> # check the percentage of classes
> round(prop.table(table(cup98$TARGET_B)), digits=3)
  0      1
0.949  0.051
> round(prop.table(table(trainData$TARGET_B)), digits=3)
  0      1
0.949  0.052
> round(prop.table(table(testData$TARGET_B)), digits=3)
  0      1
0.949  0.051
> # remove raw data to save memory
> rm(cup98, ind)
> gc()
      used      (Mb)   gc trigger      (Mb)   max used      (Mb)
Ncells  536865    28.7     818163    43.7    741108    39.6
Vcells 2454644    18.8    20467682   156.2   78071440   595.7
> memory.size()
[1] 57.95

```

ต่อไปเป็นการใช้ ctree() เพื่อสร้างโมเดลที่ ด้วย training data และฟังก์ชัน object.size() จะให้ค่าขนาดของ object

```

> # build ctree
> myCtree <- NULL
> startTime <- Sys.time()
> myCtree <- ctree(TARGET_B~, data=trainData)
> Sys.time() - startTime
Time difference of 8.802615 s
> print(object.size(myCtree), units = "Mb")
10.1 Mb
> #print(myCtree)
> memory.size()
[1] 370.7
> # plot the tree and save it in a .PDF file
> pdf(paste(fileName, ".pdf", sep=""), width=12, height=9,
+      paper="a4r", pointsize=6)
> plot(myCtree, type="simple", ip_args=list(pval=F),
+       ep_args=list(digits=0), main=fileName)
> graphics.off()

```

14.7 การสร้างโมเดลโดยใช้เฉพาะคอลัมน์ที่ต้องการ

หลังจากสร้าง decision tree จำนวน 10 ตัว แล้วทุกคอลัมน์ที่ปรากฏจะนำมาใช้กับข้อมูลดังเดิมอีกรอบ หนึ่งในการสร้าง decision tree ที่ใช้งานจริง

```
> vars.selected <- c("CARDS", "CARDGIFT", "CARDPM12", "CHILD12",
+ "CLUSTER2", "DOMAIN", "GENDER", "GEOCODE2", "HIT", "HOMEOWNR",
+ "INCOME", "LASTDATE", "MINRAMNT", "NGIFTALL", "PEPSTRFL",
+ "RECINHSE", "RFA_2A", "RFA_2F", "STATE", "WALKER")

> trainPercentage <- 80
> testPercentage <- 20
> fileName <- paste("cup98-ctree", trainPercentage,
+ testPercentage, sep="-")
> vars <- c("TARGET_B", vars.selected)
> # partition the data into training and test subsets
> ind <- sample(2, nrow(cup98), replace=T, prob=c(trainPercentage,
+ testPercentage))

> trainData <- cup98[ind==1, vars]
> testData <- cup98[ind==2, vars]
> # build a decision tree
> myCtree <- ctree(TARGET_B~., data=trainData)
> print(object.size(myCtree), units = "Mb")
39.7 Mb
> memory.size()
[1] 1010.44
> print(myCtree)
      Conditional inference tree with 21 terminal nodes
Response: TARGET_B
Inputs: CARDS, CARDGIFT, CARDPM12, CHILD12, CLUSTER2, DOMAIN,
        GENDER, GEOCODE2, HIT, HOMEOWNR, INCOME, LASTDATE, MINRAMNT,
        NGIFTALL, PEPSTRFL, RECINHSE, RFA 2A, RFA 2F, STATE, WALKER
Number of observations: 76450

1) RFA_2A == {D, E}; criterion = 1, statistic = 428.147
  2) LASTDATE <= 9606; criterion = 1, statistic = 93.226
    3) RFA_2F <= 2; criterion = 1, statistic = 87.376
      4) INCOME <= 1; criterion = 0.985, statistic = 77.333
        5)* weights = 903
      4) INCOME > 1
        6)* weights = 6543
    3) RFA_2F > 2
      7) CARDPM12 <= 4; criterion = 1, statistic = 54.972
        8)* weights = 1408
    7) CARDPM12 > 4
      9) PEPSTRFL == {X}; criterion = 1, statistic = 47.597
        10) WALKER == {Y}; criterion = 1, statistic = 40.911
          11)* weights = 1152
        10) WALKER == {}
          12)* weights = 8479
      9) PEPSTRFL == {}
        13)* weights = 3804
  2) LASTDATE > 9606
    14)* weights = 1000
```

```

1) RFA_2A == {F, G}
15) PEPSTRFL == {X}; criterion = 1, statistic = 102.032
16) LASTDATE <= 9607; criterion = 1, statistic = 48.418
17) MINRAMNT <= 12.5; criterion = 1, statistic = 45.804
18) RFA_2F <= 1; criterion = 1, statistic = 51.858
19)* weights = 8121
18) RFA_2F > 1
20) GENDER == { , A, J, M, U}; criterion = 0.998,
statistic = 46.458
21) GENDER == {A, J}; criterion = 0.998,
statistic = 37.321
22)* weights = 38
21) GENDER == { , M, U}
23)* weights = 3591
20) GENDER == {F}
24)* weights = 4428
17) MINRAMNT > 12.5
25) CARDPM12 <= 4; criterion = 0.983,
statistic = 37.436
26) NGIFTALL <= 2; criterion = 0.986,
statistic = 11.874
27)* weights = 9
26) NGIFTALL > 2
28)* weights = 39
25) CARDPM12 > 4
29)* weights = 605

16) LASTDATE > 9607
30) CARDPM12 <= 10; criterion = 1, statistic = 31.728
31)* weights = 881
30) CARDPM12 > 10
32)* weights = 113
15) PEPSTRFL == {}
33) CARDGIFT <= 5; criterion = 1, statistic = 90.915
34) CLUSTER2 <= 34; criterion = 1, statistic = 91.259
35)* weights = 19613
34) CLUSTER2 > 34
36) RFA_2A == {F}; criterion = 0.966,
statistic = 58.501
37)* weights = 10712
36) RFA_2A == {G}
38)* weights = 3843
33) CARDGIFT > 5
39) RFA_2F <= 2; criterion = 0.974, statistic = 39.703
40)* weights = 951
39) RFA_2F > 2
41)* weights = 217

```

ต่อไปจะเป็นการบันทึก decision tree ที่ได้เป็น Rdata file และกราฟถูกบันทึกลง PDF file ถ้าหากมีขนาดใหญ่ควรกำหนดให้กระดานมีขนาดใหญ่และตัวหนังสือมีขนาดเด็ก (width, height, pointsize)

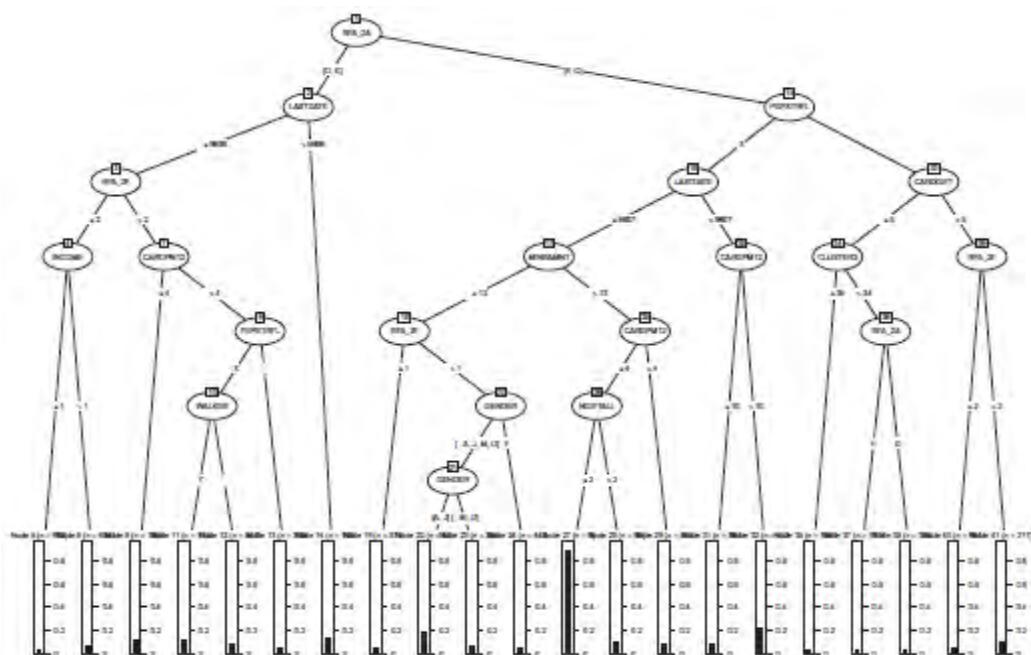
```
> save(myCtree, file = paste(fileName, ".Rdata", sep=""))

> pdf(paste(fileName, ".pdf", sep=""), width=12, height=9,
+ paper="a4r", pointsize=6)

> plot(myCtree, type="simple", ip_args=list(pval=F), ep_args=list
(digits=0), main=fileName)

> plot(myCtree, terminal_panel=node_barplot(myCtree),
ip_args=list(pval=F), ep_args=list(digits=0), main=fileName)

> graphics.off()
```



รูปที่ 14.1

ต่อไปเป็นการนำ test data มาทดสอบกับ decision tree เพื่อดูว่า tree นี้มีประสิทธิภาพดีเพียงใด

```
> save(myCtree, file = paste(fileName, ".Rdata", sep=""))

> pdf(paste(fileName, ".pdf", sep=""), width=12, height=9,
+ paper="a4r", pointsize=6)

> plot(myCtree, type="simple", ip_args=list(pval=F), ep_args=list
(digits=0), main=fileName)

> plot(myCtree, terminal_panel=node_barplot(myCtree),
ip_args=list(pval=F), ep_args=list(digits=0), main=fileName)

> graphics.off()
```

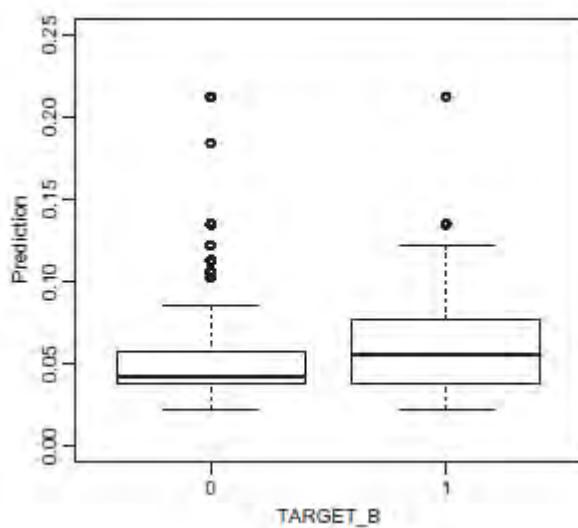
```

> rm(trainData)
> myPrediction <- predict(myCtree, newdata=testData)
> # check predicted results
> testResult <- table(myPrediction, testData$TARGET_B)
> percentageOfOne <- round(100 * testResult[,2] / (testResult[,1]
+ testResult[,2]), digits=1)
> testResult <- cbind(testResult, percentageOfOne)
> print(testResult)

          0      1   percentageOfOne
0.0223783502472027 884    23     2.5
0.0310077519379845 260     8     3.0
0.0323935772964899 2541    82     3.1
0.0377810635802784 4665   214     4.4
0.0426055904445265 2007    75     3.6
0.0525762355415352 208    10     4.6
0.0535230352303523 1046    54     4.9
0.0557308096740273 841    50     5.6
0.0570074889194559 1587    90     5.4
0.0573656363130047 845    55     6.1
0.0743801652892562 160     9     5.3
0.0764241066163463 1895   154     7.5
0.0851305334846765 204    10     4.7
0.102564102564103 15     2    11.8
0.105990783410138 45     6    11.8
0.1128472222222222 232    28    10.8
0.122159090909091 309    42    12.0
0.135     240    26    9.8
0.184210526315789 8     0    0.0
0.212389380530973 22     8    26.7
0.888888888888889 2     0    0.0

> boxplot(myPrediction ~ testData$TARGET_B, xlab="TARGET_B",
ylab="Prediction", ylim=c(0,0.25))

```



กู๊ด 14.2

```

> s1 <- sort(myPrediction, decreasing=TRUE, method = "quick",
  index.return=TRUE)

> testSize <- nrow(testData)

> TotalNumOfTarget <- sum(testData$TARGET_B)

> NumOfTarget <- rep(0, testSize)

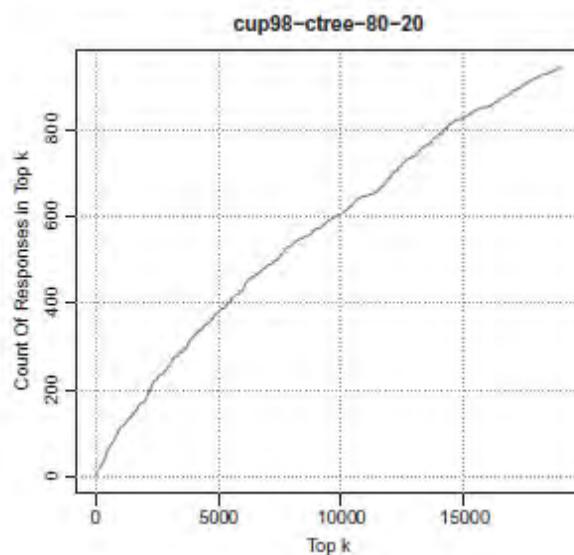
> NumOfTarget[1] <- (testData$TARGET_B)[s1$ix[1]]

> for (i in 2:testSize) {
+
+   NumOfTarget[i] <- NumOfTarget[i-1] + testData$TARGET_B[s1$ix[i]]
+
+ }

> plot(1:testSize, NumOfTarget, pty=". ", type="l", lty="solid",
  col="red", ylab="Count Of Responses in Top k", xlab="Top k",
  main=fileName)

> grid(col = "gray", lty = "dotted")

```



กู๊ด 14.3

```

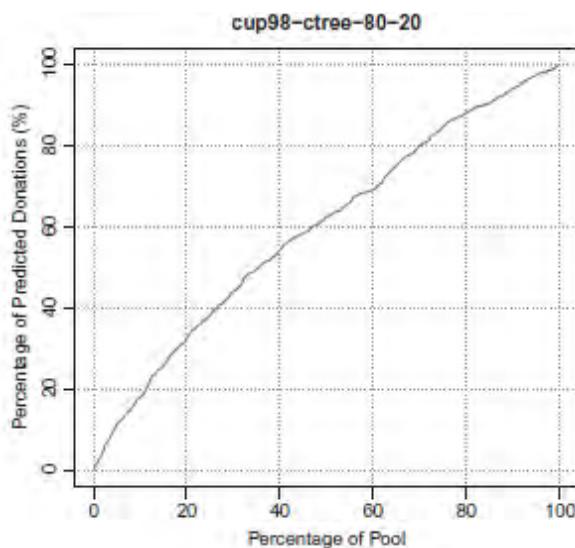
> percentile <- 100 * (1:testSize)/ testSize

> percentileTarget <- 100 * NumOfTarget/ TotalNumOfTarget

> plot(percentile, percentileTarget, pty=". ", type="l",
  lty="solid", col="red", ylab="Percentage of Predicted Donations
(%)", xlab="Percentage of Pool", main=fileName)

> grid(col = "gray", lty = "dotted")

```



รูปที่ 14.4

14.8 การวัดคะแนน (Scoring)

ในการวัดคะแนนทรีบินาดใหญ่และใช้ข้อมูลขนาดใหญ่ อาจจะทำให้ลื้นหน่วยความจำ (ไม่สามารถทำงานได้) เราสามารถใช้วิธีแบ่ง score data ออกเป็นกลุ่มย่อยหลาย ๆ กลุ่ม และนำไปทดสอบทรีเมื่อได้ผลออกมา จึงนำมารวมกันเป็นผลสรุป

```
> memory.limit(4095)
> # read scoring data and training data
> cup98val <- read.csv("./data/KDDCup1998/cup98VAL.txt")
> cup98 <- read.csv("./data/KDDCup1998/cup98LRN.txt")
> library(party) # for ctree
> treeFileName <- "cup98-ctree-80-20"
> splitNum <- 10
```

ก่อนทำ Scoring เราต้องตรวจสอบว่า categorical variable มีจำนวน level เท่ากับใน train data หรือไม่ ถ้าไม่เราต้องกำหนด factor level ใน scoreData ให้เท่า trainData จึงจะใช้ได้ในฟังก์ชัน predict() ถ้ามีค่า Missing ให้กำหนดเป็น NA

```

> # check and set levels of categorical variables
> trainData <- cup98[,vars]
> vars2 <- setdiff(c(vars,"CONTROLN"), "TARGET_B")
> scoreData <- cup98val[,vars2]
> rm(cup98, cup98val)
> trainNames <- names(trainData)
> scoreNames <- names(scoreData)
> #cat("\n checking and setting variable values \n")
> newScoreData <- scoreData
> variableList <- intersect(trainNames, scoreNames)
> for (i in 1:length(variableList)) {
+   varname <- variableList[i]
+   trainLevels <- levels(trainData[,varname])
+   scoreLevels <- levels(newScoreData[,varname])
+   if (is.factor(trainData[,varname]) & setequal(trainLevels,
+     scoreLevels)==F) {
+     cat("Warning: new values found in score data, and they
+       will be changed to NA!\n")
+     cat(varname, "\n")
+     cat("train: ", length(trainLevels), ", ", trainLevels,
+       "\n")
+     cat("score: ", length(scoreLevels), ", ", scoreLevels,
+       "\n\n")
+     newScoreData[,varname] <- factor(newScoreData[,varname],
+       levels=trainLevels)
+   }
+ }
Warning: new values found in score data, and they will be changed
to NA!

```

```

GENDER
train: 7, A C F J M U
score: 5, F J M U

Warning: new values found in score data, and they will be changed
to NA!

STATE
train: 57, AA AE AK AL AP AR AS AZ CA CO CT DC DE FL GA GU HI
      IA ID IL IN KS KY LA MA MD ME MI MN MO MS MT NC ND
      NE NH NJ NM NV NY OH OK OR PA RI SC SD TN TX UT VA
      VI VT WA WI WV WY
score: 59, AA AE AK AL AP AR ASV AZ CA CO CT DC DE FL GA GU
      HI IA ID IL IN KS KY LA MA MD ME MI MN MO MS MT NC
      ND NE NH NJ NM NV NY OH OK OR PA PR PW RI SC SD TN
      TX UT VA VI VT WA WI WV WY

```

After checking the new data, we then load the model and check memory usage. We also remove some objects which will no longer be used and do a garbage collection with function `gc()`.

```

> # loading model
> load(paste(treeFileName, ".Rdata", sep=""))
> print(object.size(trainData), units = "Mb")
8 Mb
> print(object.size(scoreData), units = "Mb")
8.1 Mb
> print(object.size(newScoreData), units = "Mb")
8.1 Mb
> print(object.size(myCtree), units = "Mb")
39.7 Mb
> gc()
      used      (Mb)  gc trigger  (Mb)  max used      (Mb)
Ncells  670228     35.8   1073225     57.4   1073225     57.4
Vcells  60433162    461.1  130805779    998.0  130557146    996.1

> memory.size()
[1] 516.73

> rm(trainNames, scoreNames)
> rm(variableList)
> rm(trainLevels, scoreLevels)
> rm(trainData, scoreData)
> gc()

      used      (Mb)  gc trigger  (Mb)  max used      (Mb)
Ncells  670071     35.8   1073225     57.4   1073225     57.4
Vcells  58323258    445.0  130805779    998.0  130557146    996.1

> memory.size()
[1] 500.23

```

หลังจากตรวจสอบข้อมูลแล้วเราจะโหลดโมเดลและตรวจสอบการใช้หน่วยความจำเพื่อให้มีหน่วยความจำว่างเพิ่มขึ้น เราอาจจะทำการลบ object บางตัวและทำการลบขยะ (garbage collection) ด้วย gc()

```
> nScore <- dim(newScoreData)[1]  
> (splitSize <- round(nScore/splitNum) )  
[1] 9637  
  
> myPred <- NULL  
  
> for (i in 1:splitNum) {  
+   startPos <- 1 + (i-1)*splitSize  
+   if (i==splitNum) {  
+     endPos <- nScore  
+   }  
+   else {  
+     endPos <- i * splitSize  
+   }  
+   print(paste("Predicting:", startPos, "-", endPos))  
+   # make prediction  
+   tmpPred <- predict(myCtree, newdata = newScoreData  
+                       [startPos:endPos,])  
+   myPred <- c(myPred, tmpPred)  
+ }
```

```
[1] "Predicting: 1 - 9637"
[1] "Predicting: 9638 - 19274"
[1] "Predicting: 19275 - 28911"
[1] "Predicting: 28912 - 38548"
[1] "Predicting: 38549 - 48185"
[1] "Predicting: 48186 - 57822"
[1] "Predicting: 57823 - 67459"
[1] "Predicting: 67460 - 77096"
[1] "Predicting: 77097 - 86733"
[1] "Predicting: 86734 - 96367"
> # cumulative count and percentage
> length(myPred)
[1] 96367
> rankedLevels <- table(round(myPred, digits=4))
> # put highest rank first by reversing the vector
> rankedLevels <- rankedLevels[length(rankedLevels):1]
> levelNum <- length(rankedLevels)
> cumCnt <- rep(0, levelNum)
> cumCnt[1] <- rankedLevels[1]
> for (i in 2:levelNum) {
+   cumCnt[i] <- cumCnt[i-1] + rankedLevels[i]
+}
> cumPercent <- 100 * cumCnt / nScore
> cumPercent <- round(cumPercent, digits=1)
```

```

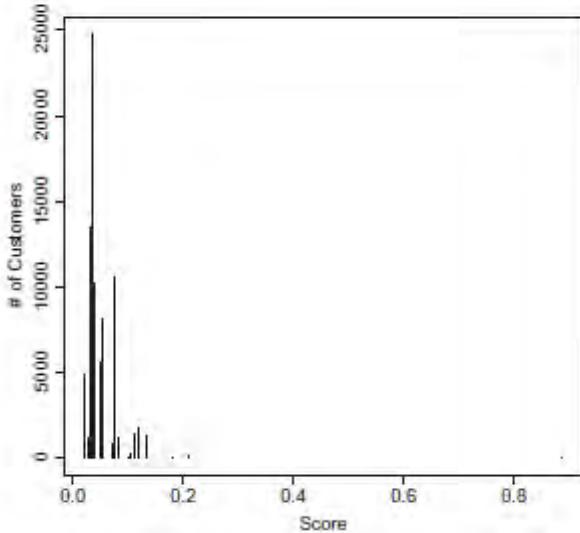
> percent <- 100 * rankedLevels / nScore
> percent <- round(percent, digits=1)
> cumRanking <- data.frame(rankedLevels, cumCnt, percent,
+ cumPercent)
> names(cumRanking) <- c("Frequency", "CumFrequency",
+ "Percentage", "CumPercentage")
> print(cumRanking)

   Frequency CumFrequency Percentage CumPercentage
0.8889    9          9      0.0       0.0
0.2124   141        150      0.1       0.2
0.1842    68        218      0.1       0.2
0.135     1342      1560      1.4       1.6
0.1222    1779      3339      1.8       3.5
0.1128    1369      4708      1.4       4.9
0.106     278       4986      0.3       5.2
0.1026    56        5042      0.1       5.2
0.0851    1138      6180      1.2       6.4
0.0764    10603     16783     11.0      17.4
0.0744    800       17583     0.8       18.2
0.0574    4611      22194     4.8       23.0
0.057     8179      30373     8.5       31.5
0.0557    4759      35132     4.9       36.5
0.0535    5558      40690     5.8       42.2
0.0526    1178      41868     1.2       43.4
0.0426    10191     52059     10.6      54.0
0.0378    24757     76816     25.7      79.7
0.0324    13475     90291     14.0      93.7
0.031     1189      91480     1.2       94.9
0.0224    4887      96367     5.1      100.0

> write.csv(cumRanking, "cup98-cumulative-ranking.csv",
+ row.names=T)

> pdf(paste("cup98-score-distribution.pdf", sep=""))
> plot(rankedLevels, x=names(rankedLevels), type="h",
+ xlab="Score", ylab="# of Customers")
> graphics.off()

```



กูน 14.5

ต่อไปเป็นการแบ่ง scoring data เป็นหลายกลุ่มย่อย เพื่อไปตรวจสอบกับโมเดลที่ และมีการแสดงการกระจายของคะแนน

```
> s1 <- sort(myPred, decreasing=TRUE, method = "quick",
  index.return=T)

> varToOutput <- c("CONTROLN")

> score <- round(myPred[s1$ix], digits=4)

> table(score, useNA="ifany")

score
0.0224 0.031 0.0324 0.0378 0.0426 0.0526 0.0535 0.0557 0.057 0.0574 0.0744
4887 1189 13475 24757 10191 1178 5558 4759 8179 4611 800
0.0764 0.0851 0.1026 0.106 0.1128 0.1222 0.135 0.1842 0.2124 0.8889
10603 1138 56 278 1369 1779 1342 68 141 9

> result <- data.frame(cbind(newScoreData[s1$ix], varToOutput),
  score)

> names(result) <- c(varToOutput, "score")

> write.csv(result, "cup98-predicted-score.csv", row.names=F)
```

ต่อไปเราจะใช้ score ในการจัดระดับลูกค้าและบันทึกลงใน .csv file

```
> # output as an EXCEL file

> library(RODBC)

> xlsFile <- odbcConnectExcel("cup98-predicted-score.xls",
  readOnly=F)

> sqlSave(xlsFile, result, rownames=F)

> odbcCloseAll()
```

14.9 แสดงกฎ (Print Rules)

เป็นการแสดงกฎตามลำดับ score ในลักษณะข้อความภาษาอังกฤษ

14.9.1 แสดงกฎ

โดยปรับปรุงฟังก์ชัน TerminalNode, SplittingNode, ordered Split, nominalSplit ที่อยู่ในไฟล์ Print.R ใน package party

```

> # functions for printing rules from ctree
> # based on "Print.R" from package party
> print.TerminalNode <- function(x, rule = NULL, ...) {
+   n.rules <- n.rules + 1
+   node.ids <- c(node.ids, x$nodeID)
+   n.records <- c(n.records, sum(x$weights))
+   scores <- c(scores, x$prediction)
+   ruleset <- c(ruleset, rule)
+
+ }
> print.SplittingNode <- function(x, rule = NULL, ...) {
+   if (!is.null(rule)) {
+     rule <- paste(rule, "\n")
+
+     rule2 <- print(x$psplit, left = TRUE, rule=rule)
+     print(x$left, rule=rule2)
+
+     rule3 <- print(x$psplit, left = FALSE, rule=rule)
+     print(x$right, rule=rule3)
+
+   }
}

```

```

> print.orderedSplit <- function(x, left = TRUE, rule = NULL, ...) {
+   if (!is.null (attr (x$splitpoint, "levels"))) {
+     sp <- attr (x$splitpoint, "levels") [x$splitpoint]
+   } else {
+     sp <- x$splitpoint
+   }
+   n.pad <- 20 - nchar (x$variableName)
+   pad <- paste(rep(" ", n.pad), collapse="")
+   if (!is.null(x$toleft)) {
+     left <- as.logical(x$toleft) == left
+   }
+   if (left) {
+     rule2 <- paste(rule, x$variableName, pad, "<= ", sp, sep = "
+   } else {
+     rule2 <- paste (rule, x$variableName, pad, "> ", sp, sep = "
+   }
+   rule2
+ }

> print.nominalSplit <- function(x, left = TRUE, rule = NULL, ...) {
+   levels <- attr(x$splitpoint, "levels")
+   ### is > 0 for levels available in this node
+   tab <- x$table

+   if (left) {
+     lev <- levels[as.logical(x$splitpoint) & (tab > 0)]
+   } else {
+     lev <- levels[!as.logical(x$splitpoint) & (tab > 0)]
+   }
+   txt <- paste("", paste(lev, collapse="", ''), "", sep="")
+   n.pad <- 20 - nchar(x$variableName)
+   pad <- paste(rep(" ", n.pad), collapse="")
+   rule2 <- paste(rule, x$variableName, pad, txt, sep = "")
+   rule2
+ }

```

ผลลัพธ์จากการใช้ `print(myCtree@tree)` จะได้ global variable คือ

- `n.rules`: the number of rules;
- `node.ids`: the IDs of leaf nodes;
- `n.records`: the number of records falling in every leaf node;
- `scores`: the score of every leaf node; and
- `ruleset`: a set of rules corresponding to every leaf node.

```
> library(party) # for ctree  
> # loading model  
> load(paste(treeFileName, ".Rdata", sep=""))  
> # extract rules from tree  
> n.rules <- 0  
> node.ids <- NULL  
> n.records <- NULL  
> scores <- NULL  
> ruleset <- NULL  
> print(myCtree@tree)  
> n.rules
```

ต่อไปนี้ใช้ฟังก์ชัน cumsum() เพื่อคำนวณผลรวมสะสมของคอลัมน์ที่เป็นตัวเลข

```
> # sort by score descendingly

> s1 <- sort(scores, decreasing=T, method="quick",
  index.return=T)

> percentage <- 100 * n.records[s1$ix] / sum(myCtree@weights)

> cumPercentage <- round(cumsum(percentage), digits=1)

> percentage <- round(percentage, digits=1)

> # print all rules

> for (i in 1:n.rules) {

+   cat("Rule", i, "\n")

+   cat("Node:", node.ids[s1$ix[i]])

+   cat(", score:", scores[s1$ix[i]])

+   cat(", Percentage: ", percentage[i], "%", sep="")

+   cat(", Cumulative Percentage: %", cumPercentage[i], "%",
  sep="")

+   cat(ruleset[s1$ix[i]], "\n\n")

+ }

Rule 1

Node: 27, score: 0.8888889, Percentage: 0%, Cumulative Percentage:
0%

RFA_2A      'F', 'G'
PEPSTRFL    'X'
LASTDATE    <= 9607
MINRAMNT   > 12.5
CARDPM12    <= 4
NGIFTALL    <= 2
Rule 2

Node: 32, score: 0.2123894, Percentage: 0.1%, Cumulative
Percentage: 0.2%

RFA_2A      'F', 'G'
PEPSTRFL    'X'
LASTDATE    > 9607
CARDPM12    > 10

Rule 3

Node: 22, score: 0.1842105, Percentage: 0%, Cumulative Percentage:
0.2%

RFA_2A      'F', 'G'
PEPSTRFL    'X'
LASTDATE    <= 9607
MINRAMNT   <= 12.5
RFA_2F      > 1
GENDER      ' ', 'A', 'J', 'M', 'U'
GENDER      'A', 'J'
```

Rule 4

```
Node: 14, score: 0.135, Percentage: 1.3%, Cumulative Percentage:  
1.5%
```

```
RFA_2A      'D', 'E'  
LASTDATE   > 9606
```

Rule 5

```
Node: 8, score: 0.1221591, Percentage: 1.8%, Cumulative  
Percentage: 3.4%
```

```
RFA_2A      'D', 'E'  
LASTDATE   <= 9606  
RFA_2F      > 2  
CARDPM12   <= 4
```

14.10 บทสรุป

บทนี้อธิบายการสร้างโมเดลและการนำไปใช้งาน กับชุดข้อมูลใหญ่ เมื่อมีหน่วยความจำขนาดจำกัด โดยการสร้างกลุ่มข้อมูลขนาดเล็กเพื่อนำไปสร้างและทดสอบ โมเดล มีวิธีการอื่นที่น่าสนใจคือ การสุ่มคอตั้นเน็ต การสุ่มเรคคอร์ด และใช้ขั้นตอนท่านองเดียวกันในการสร้าง โมเดล ซึ่งจะใช้หน่วยความจำน้อยลงอย่างมาก