

2. การบันทึกข้อความต่าง ๆ ที่ปรากฏอยู่บน RGUI ไว้เป็นแฟ้ม ซึ่ง **R** จะกำหนดให้ มีนามสกุลเป็น txt ในกรณีนี้เช่นเดียวกับ การ save วิธีแรก คือ ไม่แนะนำให้ใช้ การ save output ต่าง ๆ ที่ได้คำนวณไว้ ควรจะเก็บไว้เป็นแฟ้มคำสั่งดีกว่า ซึ่ง สามารถนำคำสั่งในแฟ้มมา run หลาย ๆ ครั้ง ได้ตามที่ต้องการ

โปรแกรม Crimson Editor

การทำงานกับแฟ้มคำสั่ง เพื่อให้ง่ายต่อการแก้ไข ควรใช้โปรแกรม Crimson Editor แก่ใจ คำสั่งก่อน จากนั้นจึง run คำสั่งในโปรแกรม **R** Crimson Editor เป็นโปรแกรม editor เช่นเดียวกับ Notepad แต่มีลักษณะพิเศษมากมายที่ช่วยในการเขียนโปรแกรมหรือสคริปต์ เช่น ทำงานกับหลายแฟ้มพร้อมกันได้ ให้สีสำหรับคำสงวนได้ ทำ undo/redo ได้หลายชั้น และค้นหา/แทนที่ได้

โปรแกรมนี้เป็น freeware ซึ่งจะสามารถ download ได้ฟรีจาก <http://www.crimsoneditor.com> หลังจากติดตั้งแล้ว เพื่อให้ Crimson รู้จัก syntax ของ **R** ให้คัดลอกแฟ้ม r.key ที่ให้ไว้แล้วในโฟลเดอร์ติดตั้งโปรแกรมในแผ่นซีดี ไปทับแฟ้มเดิมที่มีอยู่แล้วใน C:\Program Files\Crimson Editor\spec แฟ้มที่ให้ใหม่นี้ ได้ติดตั้งคำสงวนของ library **ice** ไว้แล้ว ท่านสามารถเปิดแฟ้มนี้ดูได้ด้วยโปรแกรม NotePad หรือแม้แต่ Crimson Editor ซึ่งส่วนที่เพิ่มเติมสำหรับ **ice** จะอยู่ตอนท้ายของแฟ้ม จะเห็นได้ว่าสามารถเพิ่มคำสงวนให้กับโปรแกรมนี้ได้ง่ายมาก

โปรแกรมสามารถตรวจสอบ syntax และขีดเส้นใต้คู่วงเล็บหรือวงเล็บซึ่งมีความสำคัญในการเขียนคำสั่งของ **R** ได้ แต่ก่อนที่โปรแกรมจะสามารถทำเช่นนั้นได้ ท่านต้องทำขั้นตอนต่อไป เพื่อให้โปรแกรมรู้จัก syntax ของ **R** ก่อน

1. เปิดโปรแกรม Crimson Editor แล้วเลือกเมนู “Tools -> Preferences...” แล้วเลือก Syntax Type ในรายการด้านซ้ายมือ
2. เลื่อนในช่อง Syntax Type ด้านขวามือไปด้านล่าง จนพบ –Empty- อันแรก แล้วคลิกเลือกทำแถบสี
3. ในช่อง “Description:” ด้านล่าง พิมพ์ “**R**” (**R** ตัวใหญ่)
4. คลิกที่ปุ่ม ... ด้านขวามือของช่อง “Lang Spec:” แล้วเลือก “r.spc” ที่อยู่ในรายการ
5. คลิกที่ปุ่ม ... ด้านขวามือของช่อง “Keywords:” แล้วเลือก “r.key” ที่อยู่ในรายการ
6. คลิกปุ่มขึ้นบนบนตรงด้านบนขวาเพื่อเลื่อน “R” ขึ้นไปให้อยู่ในตำแหน่งที่ถูกต้องตามลำดับตัวอักษร
7. คลิกปุ่ม “Apply” ที่ด้านล่างขวา
8. คลิกที่ “Filter” ในรายการด้านซ้ายมือ จะปรากฏรายการ File Type ขึ้นทางขวามือ
9. เลื่อนลงไปที่ตำแหน่ง –Empty- แรกที่พบ

10. พิมพ์ “R” ในช่อง “Description:” พิมพ์ “*.r” ในช่อง “Extension:” และ “r” ในช่อง “Default Ext:”
11. คลิกปุ่ม “Apply” ที่ด้านล่างขวา แล้วคลิก “OK”

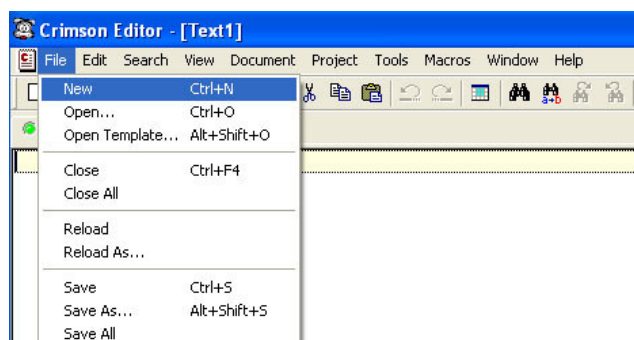
ถึงตรงนี้ **R** ควรปรากฏในรายการเมนู “Document -> Syntax Type” ต่อไปทุกครั้งที่ท่านทำงานกับแฟ้ม **R** ก็ให้แน่ใจว่า **R** ถูกเลือกใน “Document -> Syntax Type” ก็จะมั่นใจได้ว่าคำสั่งของ **R** จะเปลี่ยนสีอย่างถูกต้อง

ขั้นตอนต่อจากนี้ไป แนะนำให้แก้ไข และเพิ่มเติมคำสั่งต่าง ๆ โดยใช้โปรแกรม **Crimson Editor** ทั้งนี้เพื่อ

1. ให้โปรแกรม **Crimson Editor** ตรวจสอบความถูกต้องของคำสั่งก่อนการทำงาน
2. สามารถนำคำสั่งเดิมมาทำงานซ้ำ ๆ ได้โดยไม่ต้องพิมพ์ใหม่ ซึ่งการประหยัดเวลาในการพิมพ์คำสั่ง
3. บันทึกเป็นหลักฐานการทำงานต่าง ๆ ไว้เป็นเอกสาร ที่สามารถนำมาตรวจสอบแก้ไข เพิ่มเติมในภายหลังได้

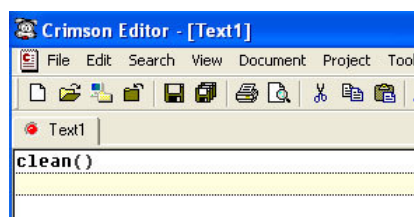
หลังจากติดตั้งโปรแกรม **Crimson Editor** และแก้ไขให้โปรแกรมรู้จักคำสั่งใน **ice** แล้ว ขั้นตอนการเปิดโปรแกรมขึ้นมาใช้ เป็นดังนี้

1. เปิดโปรแกรม **Crimson Editor** ขึ้นมา หากมีแฟ้มใดเปิดค้างอยู่ ให้ปิดเสียก่อน แล้วสร้างแฟ้มใหม่ด้วยตัวเลือกเมนู **File-->New** ดังตัวอย่าง



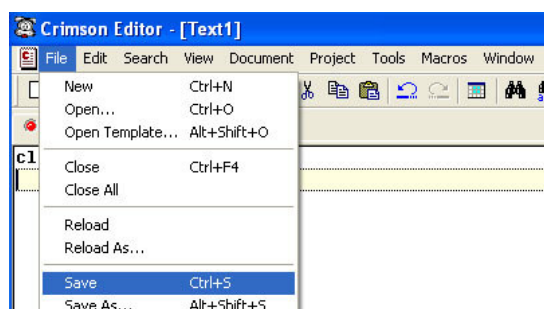
รูปที่ 4.4 จากเมนู **File** เลือก **New** ใน **Crimson Editor**

2. โดยทั่วไป ในการทำงานกับแฟ้มคำสั่งในการจัดการข้อมูลหรือการวิเคราะห์ทางสถิติ ในแต่ละครั้ง เราควรล้างหน่วยความจำที่ค้างมาจากเดิมเสียก่อน ด้วยการพิมพ์คำสั่ง `clean()` ในบรรทัดแรกสุด ดังนี้



รูปที่ 4.5 คำสั่ง *clean()*

3. จากนั้น save แฟ้มเสียก่อน ด้วยตัวเลือกเมนู File-->Save ก่อนที่จะพิมพ์คำสั่งต่อไป โดย save ใน folder ทำงานของ **R** หรือที่เดียวกับแฟ้มข้อมูลนั่นเอง ตั้งชื่อเป็น vctest.r ดังนี้ สังเกตว่าในที่นี้ใช้นามสกุลของแฟ้มเป็น **.R** เพื่อระบุว่าเป็นแฟ้มคำสั่ง



รูปที่ 4.6 เลือก Save จาก เมนู File

4.1.11 คำสั่ง `do()`

เป็นคำสั่งให้ **R** ทำงานตามคำสั่งที่เขียนไว้ภายในแฟ้มชุดคำสั่งที่มีนามสกุล **.R** อยู่ใน library **ice** มีรูปแบบการออกคำสั่งดังนี้

```
do(file, prompt.echo = ">>", from = 1, to = -1)
```

<code>file</code>	ชื่อแฟ้มชุดคำสั่ง เขียนไว้ภายใน "..."
<code>prompt.echo</code>	ตัวอักษรบอกลักษณะ <code>prompt</code> ที่จะใช้แสดง
<code>from</code>	เลขบรรทัดใน 'file' ที่เริ่มการทำงาน
<code>to</code>	เลขบรรทัดใน 'file' ที่หยุดการทำงาน ค่าปกติเป็น -1 หมายถึงทำงานจนหมดแฟ้มข้อมูล

ตัวอย่างการใช้คำสั่งดังนี้

```
> do("vcttest.r")
```

❶

```
> do("vcttest.r", from=10, to=20)
```

❷

คำอธิบาย

คำสั่ง ❶ เป็นการสั่งให้ **R** กระทำคำสั่งที่อยู่ในแฟ้ม `vcttest.r` ทั้งหมด ส่วนคำสั่งที่ ❷ เป็นการสั่งให้กระทำคำสั่งที่ 10 จนถึงคำสั่งที่ 20

4.1.12 คำสั่ง `logg()`

เป็นคำสั่งให้ **R** เก็บผลลัพธ์ที่ได้จากการคำนวณให้อยู่ในแฟ้ม คำสั่งนี้อยู่ใน library **ice** มีรูปแบบการออกคำสั่งดังนี้

```
logg(file, append=get.ec.option("append"))
```

<code>file</code>	ชื่อแฟ้มข้อมูลที่ต้องการบันทึก
<code>append</code>	ตัวเลือกที่ต้องการบันทึกต่อท้าย (ค่าปกติ <code>append=TRUE</code>) หรือเขียนทับหรือสร้างแฟ้มใหม่ (<code>append=FALSE</code>) ค่าปกติเป็นการเขียนทับ

ตัวอย่างการใช้คำสั่งนี้

```
> logg("vct.log")
```

❶

```
> summ(vct)
```

Variable	Obs.	NA	Mean	S.D.	Min.	Median	Max.
qid	315	0	161.34	93.32	1	162	323
a1	315	0	28.63	7.69	17	27	99
a2	315	0	3.17	1.31	1	3	9
a3	315	0	2.42	1.42	1	2	9
a4	315	0	30.03	81.36	0	10	999
a5	315	0	52.4	182.88	0	9	999
a6	315	0	18968.9	7502.88	0	7000	99999
a7	315	0	1.77	0.88	1	2	9
a8	218	97	1.8	0.97	1	2	9
a9	218	97	2.03	1	1	2	9
a10	315	0	2.1	0.95	1	2	9
a11	315	0	1.79	0.41	1	2	2
a12	249	66	1.44	1.09	1	1	9
a13	314	1	733.91	2046.06	30	200	9999

```
> ftab(vct, c(a2:a3))
```

a2

Label	Freq
1	5
2	108
3	100
4	58
5	31
6	8
8	1
9	4
Total	315

a3

Label	Freq
1	61
2	153
3	53
4	39
8	2
9	7
Total	315

```
> logg()
```

❷

คำอธิบาย

คำสั่ง ❶ เป็นการเปิดแฟ้มใหม่ขึ้นมาเพื่อบันทึกผลการวิเคราะห์ ซึ่งจะใช้นามสกุล .log เพื่อระบุว่าเป็นแฟ้มบันทึกผลการทำงาน ส่วนคำสั่งที่ ❷ เป็นการจบการบันทึกผล โดยไม่ใช้ชื่อแฟ้ม แนะนำให้ปิดชุดคำสั่งด้วย `logg()` ทันทีที่เปิดบันทึกผลการทำงาน มิฉะนั้นเมื่อกลับไปทำงานใน **R console** ผลการออกคำสั่งต่อไปหลังจากนี้จะถูกบันทึกลงแฟ้มทั้งหมด โดยไม่แสดงบนหน้าจอ โดยทั่วไป เรามักจะยังไม่บันทึกผลการทำงานลงแฟ้มตั้งแต่แรก แต่จะดูผลการทำงานใน **R console** ก่อน ก็สามารถทำได้โดยใช้เครื่องหมาย `#` ไว้หน้า `logg()` ก่อน เมื่อทดสอบการทำงานจนพอใจแล้ว และต้องการบันทึกผล จึงเอาเครื่องหมาย `#` ออก ดังนี้

4.2 การจัดการข้อมูล

4.2.1 คำสั่ง `as.na()`

คำสั่งนี้จะแทนที่ข้อมูลสูญหายที่กำหนดด้วยค่าตัวเลข ให้เป็นค่า NA หรือ เปลี่ยนค่าในเวกเตอร์หนึ่งหรือชุดของเวกเตอร์ในกรอบข้อมูลให้เป็น NA เป็นคำสั่งที่อยู่ใน library **ice** มีรูปแบบการออกคำสั่ง ดังนี้

```
as.na(data, x, code, append=get.ec.option("append"),
      var.names)
```

data	กรอบข้อมูล (อาจไม่ระบุก็ได้)
x	เวกเตอร์เดี่ยวหรือเวกเตอร์ของตัวแปรในกรอบข้อมูล (ถ้าระบุ data) ซึ่งอาจจะระบุเป็นชื่อตัวแปรในรูป <code>c(alcohol,smoke)</code> หรือเลขสคณภูมิในรูป <code>c(2:5)</code>
code	รหัสตัวเลขที่จะเปลี่ยนไปเป็น NA
append	ตัวเลือกจะสร้างตัวแปรใหม่ต่อท้ายกรอบข้อมูลหรือเขียนทับ ค่าปกติเป็น TRUE
var.names	เวกเตอร์ชื่อตัวแปรใหม่ ตัวเลือกนี้ใช้ได้เฉพาะเมื่อ <code>append=TRUE</code> ชื่อต้องใส่ไว้ในเครื่องหมายคำพูด

เนื่องจากในสถานะความเป็นจริงข้อมูลต่าง ๆ จะมีค่า missing เสมอ เช่น เก็บข้อมูลไม่ได้ ลืมสัมภาษณ์หรือเก็บข้อมูลในบางตัวแปร ในข้อมูล `vct` ตัวเลข 8 คือ ค่า missing จากการตอบที่ไม่เกี่ยวข้อง ส่วน ตัวเลข 9 คือค่า missing จากการไม่ตอบ ในการวิเคราะห์ต้องการตัดค่า missing เหล่านี้ทิ้ง ก่อนการตัดค่า missing ก็ควรทำการสำรวจข้อมูลด้วยคำสั่ง `summarize()` เพื่อดูค่าตัวเลขต่ำสุด และสูงสุด ดังตัวอย่าง

```
> summarize(vct)
```

Variable	Obs.	NA	Mean	S.D.	Min.	Median	Max.
qid	315	0	161.34	93.32	1	162	323
a1	315	0	28.63	7.69	17	27	99
a2	315	0	3.17	1.31	1	3	9
a3	315	0	2.42	1.42	1	2	9
a4	315	0	30.03	81.36	0	10	999
a5	315	0	52.4	182.88	0	9	999
a6	315	0	18968.9	7502.88	0	7000	99999
a7	315	0	1.77	0.88	1	2	9
a8	218	97	1.8	0.97	1	2	9
a9	218	97	2.03	1	1	2	9
a10	315	0	2.1	0.95	1	2	9
a11	315	0	1.79	0.41	1	2	2
a12	249	66	1.44	1.09	1	1	9
a13	314	1	733.91	2046.06	30	200	9999

ดูรายละเอียดรหัสต่าง ๆ ภายในตัวแปร ด้วยคำสั่ง **ftab()** ดังตัวอย่าง

```
> ftab(vct, c(a2:a3))
```

a2

Label	Freq
1	5
2	108
3	100
4	58
5	31
6	8
8	1
9	4
Total	315

a3

Label	Freq
1	61
2	153
3	53
4	39
8	2
9	7
Total	315

เมื่อทราบตัวเลขที่เป็นรหัสของค่า missing แล้วทำการแทนที่ค่าตัวเลขเหล่านี้ให้เป็น NA ด้วยคำสั่ง **as.na()** ดังตัวอย่าง

```
> vct <- as.na(vct, c(a2:a3,a7:a12),c(8,9))
> summ(vct)
```

Variable	Obs.	NA	Mean	S.D.	Min.	Median	Max.
qid	315	0	161.34	93.32	1	162	323
a1	315	0	28.63	7.69	17	27	99
a2	310	5	3.08	1.11	1	3	6
a3	306	9	2.23	0.91	1	2	4
a4	315	0	30.03	81.36	0	10	999
a5	315	0	52.4	182.88	0	9	999
a6	315	0	18968.9	7502.88	0	7000	99999
a7	312	3	1.7	0.52	1	2	4
a8	215	100	1.7	0.46	1	2	2
a9	214	101	1.9	0.3	1	2	2
a10	313	2	2.06	0.77	1	2	3
a11	315	0	1.79	0.41	1	2	2
a12	245	70	1.32	0.56	1	1	3
a13	314	1	733.91	2046.06	30	200	9999

สำหรับตัวแปรอื่น ๆ ทำการแทนที่ค่าตัวเลขให้เป็น NA เช่นเดียวกัน โดยอาจจะดูค่าตัวเลขในแต่ละตัวแปรด้วยคำสั่ง **sum()** หรือ **ftab()**

```
> vct <- as.na(vct, a1,99)
> summarize(vct)
```

คำอธิบาย

ผลจากการใช้คำสั่ง **summarize()** จะพบว่าตัวแปรที่มีค่าสูงสุดเป็นเลข 99 คือ a1 เลขสูงสุดเป็นเลข 9 คือ a2, a3 และ a7 ถึง a12 เป็นต้น ดังนั้นจากคำสั่ง **as.na()** ข้างต้นจึงเป็นการเปลี่ยนค่าตัวเลขที่เป็นรหัส missing คือ 8 และ 9 ให้เป็นค่า missing

4.2.2 คำสั่ง `recode.na()`

เปลี่ยนค่า NA กลับเป็นตัวเลขเหมือนเดิม หรือ เปลี่ยน NA ในเวกเตอร์หนึ่งหรือชุดของเวกเตอร์ในกรอบข้อมูลเป็นค่าตัวเลข คำสั่งนี้อยู่ใน library base มีรูปแบบการออกคำสั่งดังนี้

```
recode.na(data, x, code, append=get.ec.option("append"),
          var.names)
```

data	กรอบข้อมูล (อาจไม่ระบุก็ได้)
x	เวกเตอร์เดี่ยวหรือเวกเตอร์ของตัวแปรในกรอบข้อมูล (ถ้าระบุ data) ซึ่งอาจจะระบุเป็นชื่อตัวแปรในรูปแบบ c(alcohol,smoke) หรือเลขสครมภ์ในรูปแบบ c(2:5)
code	NA ที่จะเปลี่ยนไปเป็นตัวเลข
append	ตัวเลือกระบุว่าสร้างตัวแปรใหม่ต่อท้ายกรอบข้อมูล หรือเขียนทับ ค่าปกติเป็น TRUE
var.names	เวกเตอร์ชื่อตัวแปรใหม่ ตัวเลือกนี้ใช้ได้เฉพาะเมื่อ append=TRUE ชื่อต้องใส่ไว้ในเครื่องหมายคำพูด

ในบางครั้ง ผู้ใช้อาจต้องการคำนวณจำนวนที่มีค่าเป็น NA หรือค่าที่เป็น missing จึงต้องแทน NA ด้วยค่าตัวเลข ดังตัวอย่าง

```
> vct <- recode.na(vct, c(a2:a3, a7:a12),9)
```

คำอธิบาย

จากคำสั่งข้างต้นเปลี่ยนค่า NA ของตัวแปร a2, a3 และ a7 ถึง a12 กลับให้เป็นค่าตัวเลข คือ 9

เมื่อใช้คำสั่ง `summarize()` เพื่อดูค่าต่ำสุดและสูงสุดอีกครั้ง ก็จะพบว่าค่าตัวเลข 9 กลับมาอยู่ในข้อมูลอีกครั้งดังตัวอย่าง

```
> summarize(vct)
```

Variable	Obs.	NA	Mean	S.D.	Min.	Median	Max.
qid	315	0	161.34	93.32	1	162	323
a1	315	0	28.63	7.69	17	27	99
a2	315	0	3.17	1.31	1	3	9
a3	315	0	2.42	1.42	1	2	9
a4	315	0	30.03	81.36	0	10	999
a5	315	0	52.4	182.88	0	9	999
a6	315	0	18968.9	7502.88	0	7000	99999
a7	315	0	1.77	0.88	1	2	9
a8	315	0	4.02	3.43	1	2	9
a9	315	0	4.17	3.33	1	2	9
a10	315	0	2.1	0.95	1	2	9
a11	315	0	1.79	0.41	1	2	2
a12	315	0	3.02	3.23	1	1	9
a13	315	0	731.61	2043.2	9	200	9999

เพื่อการคำนวณในขั้นตอนต่อไปจึงตัดค่า missing ออกจากข้อมูลจึงต้องใช้คำสั่ง **as.na()** ให้เหมือนเดิมอีกครั้ง

```
> vct <- as.na(vct, c(a2:a3,a7:a13),9)
```

4.2.3 คำสั่ง **recode()**

เป็นการเปลี่ยนค่าตัวเลขเดิมให้เป็นค่าตัวเลขใหม่ คำสั่งนี้อยู่ใน library **ice** มีรูปแบบการออกคำสั่งดังนี้

```
recode(data, x, values, append=get.ec.option("append"),
       var.names)
```

data	กรอบข้อมูล อาจไม่ระบุก็ได้
x	เวกเตอร์หรือชุดของเวกเตอร์ อาจเรียกตัวแปรโดยชื่อหรือตัวเลขลำดับสดมภ์ก็ได้
values	เวกเตอร์กำหนดการเปลี่ยนรหัส
append	ตัวเลือกระบุว่าสร้างตัวแปรใหม่ต่อท้ายกรอบข้อมูลหรือเขียนทับ ค่าปกติเป็น TRUE
var.names	เวกเตอร์ของชื่อตัวแปรใหม่ ตัวเลือกนี้มีความหมายเมื่อ append=TRUE ชื่อต้องใส่ไว้ในเครื่องหมายคำพูด

ตัวอย่างดังนี้

```
> vct <- recode(vct,a7,c("1=0","2=1","3=1","4=1"), "hiv.test",
+ append=T)
```

คำอธิบาย

คำสั่งข้างต้นเป็นการแทนที่ค่าตัวเลขในตัวแปร a7 โดยให้ค่ารหัสเดิมจาก 1 เป็น= 0 และรหัส 2, 3, และ 4 มีค่าเป็น 1 แล้วเก็บค่าใหม่ไว้ในตัวแปรที่ชื่อ hiv.test

4.2.4 คำสั่ง *change()*

เป็นคำสั่งเปลี่ยนค่าภายในเวกเตอร์หรือชุดของเวกเตอร์ที่ระบุใน 'x' ด้วยเงื่อนไขที่ระบุใน 'condition' ให้เป็นค่าที่ระบุใน 'value' โดยการใช้คำสั่ง **change()** ที่อยู่ใน library **ice** มีรูปแบบการออกคำสั่งดังนี้

```
change(data, x, condition, value, append=get.ec.option
("append"), var.names)
```

data	กรอบข้อมูล อาจไม่ระบุก็ได้
x	เวกเตอร์หรือชุดของสคัมภ์ในกรอบข้อมูล
condition	ชุดของค่าหรือเงื่อนไขทางตรรกศาสตร์
value	ค่าใหม่ที่ต้องการ
append	ตัวเลือกระบุว่าสร้างตัวแปรใหม่ต่อท้ายกรอบข้อมูลหรือเขียนทับ ค่าปกติเป็น TRUE
var.names	เวกเตอร์ชื่อตัวแปรใหม่ ตัวเลือกนี้ใช้ได้เฉพาะเมื่อ append=TRUE ชื่อต้องใส่ไว้ในเครื่องหมายคำพูด

ดังตัวอย่าง

```
> vct <- change(vct,a10,a10!=1,2, "vct.blood", append=T)
```

คำอธิบาย

เป็นการเปลี่ยนค่าตัวเลขเดิมให้เป็นค่าตัวเลขใหม่ คือ ค่าในตัวแปร `a10` ที่ไม่เท่ากับ 1 ให้มีค่าเป็นเลข 2 ส่วนเลข 1 ยังคงเป็นค่าเดิมไว้ แล้วเก็บค่าใหม่ไว้ในตัวแปรใหม่ชื่อว่า `vct.blood`

4.2.5 คำสั่ง `transform()`

เป็นคำสั่งในการสร้างตัวแปรใหม่ใน **R** โดยเก็บค่าที่คำนวณได้ไว้ในตัวแปรใหม่ที่สร้างขึ้นมา คำสั่งนี้อยู่ใน `library base` มีรูปแบบการออกคำสั่ง ดังนี้

```
transform(x, ...)
```

`x`

ตัวแปรหรือชุดตัวแปรในกรอบข้อมูลที่ต้องการใช้ในการคำนวณ

`...`

ตัวเลือกอื่น ๆ หรือเงื่อนไขที่กำหนด

ดังตัวอย่างตัวแปร `a5` เป็นระยะเวลาของการทำอาชีพขายบริการทางเพศของหญิงบริการ โดยมีหน่วยเป็นเดือน ต้องการที่จะเปลี่ยนให้หน่วยระยะเวลาจากเดือนเป็นปี

```
> vct <- transform(vct, csw=a5/12)
```

คำอธิบาย

สร้างตัวแปรใหม่ชื่อว่า `csw` เพื่อเก็บค่าจากการคำนวณตัวแปร `a5` ที่อยู่ในกรอบข้อมูล `vct` หารด้วย 12 เพื่อเปลี่ยนหน่วยให้เป็นปี

หมายเหตุ

ผู้ใช้สามารถแทนที่ค่าที่คำนวณได้ลงไปในตัวแปรเดิมได้เลย โดยที่ไม่ต้องสร้างวัตถุใหม่ แต่วิธีนี้ไม่แนะนำให้ใช้ เพราะจะเสี่ยงต่อการเกิดข้อผิดพลาดได้ ควรเก็บค่าที่คำนวณได้ไว้ในตัวแปรใหม่จะปลอดภัยกว่า

4.2.6 คำสั่ง `label()`

เป็นคำสั่งในการให้คำอธิบายแก่ค่าตัวเลขแต่ละรหัสของตัวแปรชนิดที่เป็นแบบกลุ่ม เช่น เพศ (1=ชาย, 2 = หญิง) อยู่ใน library **ice** มีรูปแบบการออกคำสั่งดังนี้

```
label(data, x, labels, sep="", ordered=FALSE,
      append=get.ec.option("append"), var.names)
```

data	กรอบข้อมูล อาจไม่ใส่ก็ได้
x	เวกเตอร์หรือชุดของเวกเตอร์ในกรอบข้อมูลที่ต้องการกำหนดระดับ
labels	เวกเตอร์ระดับข้อมูล เขียนในรูป รหัส = คำอธิบาย ดังนี้ <code>c("0=no", "1=yes")</code>
sep	ตัวอักษรที่ใช้ล้อมชื่อระดับข้อมูล ค่าปกติไม่มี ถ้าใส่เพียงอักษรเดียว จะเป็นตัวอักษรที่วางหน้าชื่อระดับข้อมูล
ordered	ตัวเลือกเพื่อระบุว่าค่าในข้อมูลเรียงระดับหรือไม่ R จะตรวจสอบว่าเดิม vector นั้นถูกกำหนดให้เรียงระดับหรือไม่ และจะนำไปกำหนดเป็นค่าปกติของ factor ที่ส่งกลับ
append	ตัวเลือกระบุตำแหน่งที่จะวางตัวแปรใหม่ อาจวางต่อท้ายกรอบข้อมูล (ค่าปกติ <code>append=TRUE</code>) หรือแทนที่ตัวแปรเดิม (<code>append=FALSE</code>)
var.names	เวกเตอร์ของชื่อตัวแปรใหม่ ตัวเลือกนี้ใช้ได้ถ้า <code>append=TRUE</code> ชื่อต้องใส่ไว้ในเครื่องหมายคำพูด

ตัวอย่างเช่น การให้คำอธิบายแก่ค่าของตัวแปร `a2` ในกรอบข้อมูล `vct`

```
> class(a2)
[1] "numeric"
> vct <- label(vct, a2, c("1=illiterate", "2=primary", "3=secondary",
"4=high school", "5=vocation", "6=bachelor"), append=F)
> class(a2)
[1] "factor"
```

คำอธิบาย

ให้คำอธิบายค่าตัวเลข แก่ตัวแปร a2 คือ 1=ไม่ได้เรียน, 2=ประถม, 3= มัธยมต้น, 4=มัธยมปลาย, 5=ปวช/ปวส, 6=ปริญญาตรี

หากคำสั่งยาวเกิน 1 บรรทัด สามารถที่จะเกาะ Enter ขึ้นบรรทัดใหม่ได้ R จะปรากฏเครื่องหมายบวก (+) ขึ้นมา เพื่อบอกให้ทราบว่าการทำงานด้วยคำสั่งดังกล่าวยังไม่สมบูรณ์

คำสั่ง `label()` จะเปลี่ยนคลาสของตัวแปร a2 จากคลาส numeric เป็น factor โดยอัตโนมัติ ดังตัวอย่างข้างต้น

4.2.7 คำสั่ง `describe()`

เป็นคำสั่งสำหรับการสรุปกรอบข้อมูลอย่างย่อ หรือใส่คำอธิบายให้กับกรอบข้อมูล และ/หรือ ตัวแปรภายใน มีรูปแบบการออกคำสั่งดังนี้

```
describe(x, datalabel = NULL, var.labels = NULL)
```

x	กรอบข้อมูล
datalabel	ตัวเลือกเพื่อใส่คำอธิบายกรอบข้อมูล
var.labels	เวกเตอร์ตัวเลือกเพื่อใส่คำอธิบายตัวแปรในกรอบข้อมูล

ตัวอย่างการใช้คำสั่ง ดังต่อไปนี้

```
> describe(vct)
```

```
No. of observation = 315
```

	Variable	Description
1	qid	
2	a1	
3	a2	
4	a3	
5	a4	
6	a5	
7	a6	
8	a7	
9	a8	
10	a9	
11	a10	
12	a11	
13	a12	
14	a13	
15	hiv.test	
16	vct.blood	
17	csw	

จากคำสั่ง **describe()** ข้างต้น แสดงให้เห็นว่ากรอบข้อมูล `vct` ไม่มีคำอธิบายตัวแปร สามารถให้คำอธิบายตัวแปรดังกล่าวต่อไปนี้

```
> vct <- describe(vct, var=c("gestion id", "age", "education",
"province", "stay in Phuket","working", "earn money","hiv test",
"counselling", "wish to know", "know vct","vct","suitable time",
"highest cost", "hiv test", "know blood test", "duration of csw"))
> describe(vct)
No. of observation = 315

Variable   Description
1  qid      gestion id
2  a1       age
3  a2       education
4  a3       province
5  a4       stay in Phuket
6  a5       working
7  a6       earn money
8  a7       hiv test
9  a8       counselling
10 a9       wish to know
11 a10      know vct
12 a11      vct
13 a12      suitable time
14 a13      highest cost
15 hiv.test hiv test
16 vct.blood know blood test
17 csw      duration of csv
```

4.2.8 คำสั่ง **rename()**

เป็นคำสั่งในการเปลี่ยนชื่อตัวแปรในกรอบข้อมูล อยู่ใน library **ice** มีรูปแบบการออกคำสั่ง ดังนี้

```
rename(data, x, var.names)
```

<code>data</code>	กรอบข้อมูล
<code>x</code>	ตัวแปรหรือชุดตัวแปรในกรอบข้อมูลที่ต้องการเปลี่ยนชื่อ อาจเรียกตัวแปรโดยชื่อหรือเลขสคณภักก็ได้
<code>var.names</code>	เวกเตอร์ข้อความระบุชื่อใหม่

ดังตัวอย่าง

```
> vct <- rename(vct, c(a1, a2), c("age", "education"))
> describe(vct)

No. of observation = 315
```

	Variable	Description
1	qid	gestion id
2	age	age
3	education	education
4	a3	province
5	a4	stay in Phuket
6	a5	working
7	a6	earn money
8	a7	hiv test
9	a8	counselling
10	a9	wish to know
11	a10	know vct
12	a11	vct
13	a12	suitable time
14	a13	highest cost
15	hiv.test	hiv test
16	vct.blood	know blood test
17	csw	duration of csv

คำอธิบาย

เปลี่ยนชื่อตัวแปร a1 และ a2 เป็น age และ education เมื่อ describe ขึ้นมาคุณก็จะเห็นว่าชื่อตัวแปรถูกเปลี่ยนแล้ว

4.2.9 คำสั่ง `to.character()`

เป็นคำสั่ง เปลี่ยนชุดของแฟกเตอร์ในกรอบข้อมูลเป็นเวกเตอร์ข้อความ เป็นคำสั่งที่อยู่ใน library **ice** มีรูปแบบการออกคำสั่งดังนี้

```
to.character(data, x, append=get.ec.option("append"),
             var.names)
```

data	กรอบข้อมูล อาจไม่ระบุก็ได้
x	เวกเตอร์หรือชุดของแฟกเตอร์ที่ต้องการเปลี่ยน
append	ตัวเลือกระบุตำแหน่งที่จะวางตัวแปรใหม่ ซึ่งอาจเป็น ท้ายกรอบข้อมูล (ค่าปกติ append=TRUE) หรือเขียนทับ ตัวแปรเดิม (append=FALSE)
var.names	เวกเตอร์ของชื่อตัวแปรใหม่ จะใช้ได้เมื่อ append=TRUE ชื่อต้องใส่ไว้ในเครื่องหมายคำพูด

ในบางครั้งผู้ใช้อาจพบเจอข้อมูลที่เป็นตัวเลข แต่ถูกเก็บเป็นประเภทตัวอักษร ซึ่งในการคำนวณทางสถิติขั้นสูง โปรแกรมทางสถิติทุกโปรแกรม จะไม่สามารถนำตัวแปรที่เป็นประเภทตัวอักษรมาคำนวณได้ จึงจำเป็นที่จะต้องแปลงข้อมูลให้เป็นประเภทตัวเลขเสียก่อน

ดังตัวอย่างต่อไปนี้

```
> class(a3)
[1] "numeric"
> vct <- to.character(vct,a3)
> class(a3)
[1] "character"
```

❶

4.2.10 คำสั่ง *to.factor()*

เป็นคำสั่งเปลี่ยนชุดของเวกเตอร์ตัวเลขในกรอบข้อมูลเป็นแฟกเตอร์ เป็นคำสั่งที่อยู่ใน library **ice** มีรูปแบบการออกคำสั่งดังนี้

```
to.factor(data, x, labels, ordered=FALSE, append=get.ec.option
("append"), var.names)
```

data	กรอบข้อมูล อาจไม่ระบุก็ได้
x	เวกเตอร์เดี่ยวหรือชุดของเวกเตอร์ตัวเลขที่ต้องการเปลี่ยน labels เวกเตอร์ข้อความระบุคำอธิบายรหัส
ordered	ตัวเลือกตรรกะบอกว่าระดับควรถือว่าเรียงลำดับหรือไม่ (ตามลำดับที่ให้ไว้)
append	ตัวเลือกระบุตำแหน่งที่จะวางตัวแปรใหม่ ซึ่งอาจเป็นท้ายกรอบข้อมูล (ค่าปกติ append=TRUE) หรือเขียนทับตัวแปรเดิม (append=FALSE)
var.names	เวกเตอร์ของชื่อตัวแปรใหม่ จะใช้ได้เมื่อ append=TRUE ชื่อต้องใส่ไว้ในเครื่องหมายคำพูด

ตัวอย่างดังต่อไปนี้

```
> class(a4)
```

```
[1] "integer"
> vct <- to.factor(vct,a4)
> class(vct$a4)
[1] "factor"
```

②

4.2.11 คำสั่ง `to.vector()`

เปลี่ยนชุดของแฟกเตอร์ในกรอบข้อมูลเป็นเวกเตอร์ตัวเลข เป็นคำสั่งที่อยู่ใน library **ice** มีรูปแบบการออกคำสั่งดังนี้

```
to.vector(data, x, append=get.ec.option("append"), var.names)
```

data	กรอบข้อมูล อาจไม่ระบุก็ได้
x	แฟกเตอร์หรือชุดของแฟกเตอร์ที่ต้องการเปลี่ยน
append	ตัวเลือกระบุตำแหน่งที่จะวางตัวแปรใหม่ ซึ่งอาจเป็น ท้ายกรอบข้อมูล (ค่าปกติ append=TRUE) หรือเขียนทับ ตัวแปรเดิม (append=FALSE)
var.names	เวกเตอร์ของชื่อตัวแปรใหม่ จะใช้ได้เมื่อ append=TRUE ชื่อต้องใส่ไว้ในเครื่องหมายคำพูด

ตัวอย่างดังต่อไปนี้

```
> class(a4)
[1] "factor"
> vct <- to.vector(vct,a4)
> class(a4)
[1] "numeric"
```

③

4.2.12 คำสั่ง `to.numeric()`

เปลี่ยนชุดของแฟกเตอร์ในกรอบข้อมูลเป็นเวกเตอร์ตัวเลข เป็นคำสั่งที่อยู่ใน library **ice** มีรูปแบบการออกคำสั่งดังนี้

```
to.numeric(data, x, append=get.ec.option("append"), var.names)
```

data	กรอบข้อมูล อาจไม่ระบุก็ได้
x	แฟกเตอร์หรือชุดของแฟกเตอร์ที่ต้องการเปลี่ยน

append	ตัวเลือกระบุตำแหน่งที่จะวางตัวแปรใหม่ ซึ่งอาจเป็น ท้ายกรอบข้อมูล (ค่าปกติ append=TRUE) หรือเขียนทับ ตัวแปรเดิม (append=FALSE)
var.names	เวกเตอร์ของชื่อตัวแปรใหม่ จะใช้ได้เมื่อ append=TRUE ชื่อต้องใส่ไว้ในเครื่องหมายคำพูด

ตัวอย่างดังต่อไปนี้

```
> class(a3)
[1] "character"
> vct <- to.numeric(vct,a3)
> class(vct$a3)
[1] "numeric"
```

④

คำอธิบาย

- ① เปลี่ยนตัวแปรจากตัวเลขเป็นตัวอักษร เมื่อใช้คำสั่ง `class(a3)` R จะแสดงคำว่า "character"
- ② เป็นการเปลี่ยนจากตัวเลขเป็น factor เมื่อใช้คำสั่ง `class(a3)` ก็จะเห็นคำว่า "factor"
- ③ เปลี่ยนชุดของแฟกเตอร์ในกรอบข้อมูลเป็นเวกเตอร์ตัวเลข
เมื่อใช้คำสั่ง `class(a4)` ก็จะเห็นคำว่า "numeric"
- ④ เป็นการเปลี่ยนจากตัวอักษรเป็นตัวเลข เมื่อใช้คำสั่ง `class(a3)` ก็จะเห็นคำว่า "numeric"

คำเตือน

การเปลี่ยนประเภทตัวแปรดังกล่าวเป็นการเปลี่ยนชั่วคราว หากเมื่อไรก็ตามผู้ใช้เปิด data editor ด้วยคำสั่ง `fix()` ก็จะทำให้การเปลี่ยนประเภทตัวแปรดังกล่าวกลับสู่ตัวแปรประเภทเดิม

4.2.13 คำสั่ง `defactor()`

เป็นคำสั่งสำหรับสร้างเวกเตอร์ตัวเลขใหม่จากแฟกเตอร์เดิม โดยคงค่ารหัสเดิมไว้
เป็นคำสั่งที่อยู่ใน library `ice` มีรูปแบบการออกคำสั่งดังนี้

```
defactor(data, x, append=get.ec.option("append"), var.names)
```

data	แฟกเตอร์ที่ต้องการเปลี่ยน
x	ตัวเลือกว่าจะเติมตัวแปรใหม่ท้ายกรอบข้อมูล (ค่าปกติ append=TRUE) หรือแทนที่ตัวแปรเดิม
append	เวกเตอร์ของชื่อตัวแปรใหม่ ตัวเลือกนี้ใช้ได้ถ้า append=TRUE เท่านั้น ชื่อต้องใส่ไว้ในเครื่องหมายคำพูด
var.names	เวกเตอร์ชื่อตัวแปรใหม่ ตัวเลือกนี้ใช้ได้เฉพาะเมื่อ append=TRUE ชื่อต้องใส่ไว้ในเครื่องหมายคำพูด

ดังตัวอย่าง ให้คำอธิบายตัวแปร a2 (ในกรณีที่ยังไม่ label) จากนั้นตัวแปรก็จะเป็นประเภท factor จึงจะใช้คำสั่ง **defactor()** ได้

```
> class(education)
[1] "factor"
> vct <- defactor(vct, age)
> class(education)
[1] "integer"
```

4.2.14 คำสั่ง cut ()

เป็นคำสั่งแบ่งข้อมูลชนิดต่อเนื่องออกเป็นช่วงค่า และให้รหัสตัวเลขแทนแต่ละกลุ่มของช่วงค่านั้น ๆ เป็นคำสั่งที่อยู่ใน library base มีรูปแบบการออกคำสั่งดังนี้

```
cut(x, breaks, labels = NULL, include.lowest = FALSE,
    right = TRUE, dig.lab = 3, ...)
```

x	เวกเตอร์ตัวเลข หรือตัวแปรต่อเนื่องที่ต้องการตัดแบ่งเป็นกลุ่ม
breaks	เวกเตอร์ตัวเลขที่ระบุจุดตัด หรือตัวเลขที่ระบุจำนวนกลุ่มที่ต้องการ
labels	การให้คำอธิบายชื่อแต่ละกลุ่มที่ถูกตัดแบ่งแล้ว
include.lowest	กำหนดให้รวมค่าต่ำสุดที่อยู่ด้านซ้ายสุดให้อยู่ในกลุ่มแรก
right	กำหนดให้รวมค่าจุดตัดด้านขวาสุดให้อยู่ในกลุ่ม
dig.lab	กำหนดจำนวนตัวเลขที่มีได้สำหรับแต่ละจุดตัด

ในทางสถิติ เมื่อนำข้อมูลเชิงปริมาณทำการแบ่งกลุ่ม ข้อมูลบางอย่างอาจจะหายไป เช่น การแบ่งกลุ่มอายุเด็กเป็นช่วง ช่วงละ 6 เดือน ก็จะไม่ข้อมูลของเด็กอายุ 10 เดือน ที่นำมาใช้ใน

การคำนวณ แต่อายุดังกล่าวจะอยู่ในกลุ่มที่มีอายุ 7 – 12 เดือน อย่างไรก็ตามบางครั้งก็มีความจำเป็นที่จะต้องแบ่งข้อมูลต่อเนื่องเป็นข้อมูลแบบกลุ่มเพื่อความสะดวกในการดูจำนวนความถี่ในแต่ละกลุ่ม เช่น หากรายงานว่าร้อยละ 25 ของเด็กที่มีภาวะไตวายอยู่ในช่วงกลุ่มอายุ 0 – 3 เดือน ก็จะทำให้คนทั่วไปเข้าใจได้ง่ายกว่าที่จะบอกว่ามีอายุเฉลี่ย 4.7 ปี ด้วยส่วนเบี่ยงเบนมาตรฐาน 3.2 ปี

ตัวอย่างเช่น จัดกลุ่มอายุ ตัวแปร `age` เป็น 4 กลุ่มที่มีช่วงค่าเท่า ๆ กัน

```
> vct <- transform(vct, age.4 = cut(age, br = 4))
> tab(vct, age.4)
vct$age.4
-----
Label      | Freq
-----+-----
(17,26]    | 121
(26,35]    | 145
(35,44]    | 39
(44,53]    | 9
-----
```

หากต้องการเปลี่ยนช่วงค่าเป็น 10 ก็จะเริ่มจาก 15 ดังนี้

```
> vct <- transform(vct, age.4 = cut(age, br=c(15,25,35,45,55))) # หรือ
> vct <- transform(vct, age.4 = cut(age, br=10*c(1.5:5.5)))
> tab(vct, age.4)
vct$age.4
-----
Label      | Freq
-----+-----
(15,25]    | 121
(25,35]    | 145
(35,45]    | 43
(45,55]    | 5
-----
```

จำนวนความถี่จะไม่เหมือนกัน กลุ่มสุดท้ายมีเพียง 5 คน การตัดกลุ่มให้มีจำนวนความถี่ที่เท่า ๆ กัน สามารถทำได้ด้วยการใช้คำสั่ง `quantile()`

```
quantile(x, probs = seq(0, 1, 0.25), na.rm = FALSE,
         names = TRUE, type = 7, ...)
```

`x` เวกเตอร์ตัวเลข หรือตัวแปรต่อเนื่องที่ต้องการตัดแบ่งเป็นกลุ่ม

probs	เวกเตอร์ตัวเลขของความน่าจะเป็น ค่าอยู่ระหว่าง 0 และ 1
na.rm	ถ้าเป็น TRUE ค่า missing ก็จะถูกตัดออกจากการคำนวณ
names	ถ้าเป็น TRUE ค่า x ก็จะมี attribute ชื่อ names
type	เป็นวิธีการของการเลือกคอนโวลูททางพีชคณิต ซึ่งมี 9 วิธี (ดูรายละเอียดเพิ่มเติมจาก help)

ตัวอย่างคำสั่ง

```
> vct <- transform(vct, age.4=cut(age br=quantile(age, na.rm=TRUE)))
> tab(vct, age.4)
vct$age.4
-----
Label      | Freq
-----+-----
(17,24]    |    99
(24,27]    |    67
(27,32]    |    78
(32,53]    |    69
-----
```

จำนวนความถี่ในแต่ละกลุ่มมีจำนวนที่ใกล้เคียงกัน แต่ช่วงค่าในแต่ละกลุ่มไม่เท่ากัน สามารถเปลี่ยนช่วงค่าใหม่ และให้คำอธิบายกลุ่มได้ดังนี้

```
> br = c(15,25,30,35,55)
> labels = c("<25","25-30","30-35","35+")
> vct <- transform(vct,age.4=cut(a1, br = br, labels = labels))
> tab(vct, age.4)
vct$age.4
-----
Label      | Freq
-----+-----
<25        |   121
25-30      |    98
30-35      |    47
35+        |    48
-----
```

4.3 แบบฝึกหัด

จากข้อมูล `vct.dta` ให้ใช้โปรแกรม **R** ในการจัดการข้อมูลดังต่อไปนี้

1. จงจัดการข้อมูลสูญหายที่มีรหัสตัวเลขแทนค่า missing ต่อไปนี้ให้เป็น missing
 - a1 รหัส missing คือ 99
 - a4 และ a5 รหัส missing คือ 888 และ 999
 - a6 รหัส missing คือ 88888 และ 99999
 - a2, a3, a7, a8, a9, a10, a11 และ a12 รหัส missing คือ 8 และ 9
 - a13 รหัส missing คือ 8888 และ 9999
2. จงใส่คำอธิบายค่าตัวแปร a2, a7, a8, a9, a10, a11 และ a12
3. จงคำนวณอายุที่เริ่มทำอาชีพขายบริการของหญิงบริการ แล้วสร้างตัวแปรใหม่เพื่อเก็บค่าที่คำนวณได้ให้อยู่ในกรอบข้อมูล
4. จงจัดกลุ่มระดับการศึกษา (ตัวแปร a2) ใหม่เป็น 1 = น้อยกว่ามัธยมปลาย, 2 = มัธยมปลายขึ้นไป และเก็บเป็นตัวแปรใหม่ในกรอบข้อมูล
5. จงแบ่งกลุ่มค่าใช้จ่ายสูงสุดสำหรับการรับคำปรึกษาและตรวจเลือดที่คุ้นชินดีและสามารถจ่ายได้เป็น 2 กลุ่ม คือ 1= 100 บาทลงมา และ 2= สูงกว่า 100 บาท และสร้างเป็นตัวแปรใหม่ในกรอบข้อมูล
6. จงแบ่งกลุ่มรายได้ออกเป็น 3 กลุ่มที่มีความถี่ใกล้เคียงกัน

บทที่ 5

สถิติที่ใช้ในการวิเคราะห์ข้อมูลแบบกลุ่ม

การวิเคราะห์ข้อมูลแบบกลุ่มส่วนใหญ่ใช้ในการศึกษาวิจัยทางด้านระบาดวิทยา ซึ่งแบ่งออกเป็นหลายแบบ เช่น การศึกษาแบบพรรณนา กับการศึกษาความสัมพันธ์ในเชิงสาเหตุ ในการศึกษาหาความสัมพันธ์เชิงสาเหตุนั้น อาจแบ่งย่อยออกเป็น การศึกษาเชิงสังเกต กับการศึกษาเชิงทดลอง ในแต่ละการศึกษายังสามารถแบ่งย่อยออกไปได้อีกหลายแบบ และมีการศึกษาแบบพิเศษออกไปได้อีกหลายแบบ แต่ในที่นี้จะแสดงการแบ่งแบบง่ายๆ เพื่อให้เห็นภาพรวมของวิธีวิจัยทางระบาดวิทยาอย่างง่ายๆ ดังนี้

1. การศึกษาแบบพรรณนา
 1. การศึกษาเชิงสำรวจ (survey)
 2. การศึกษาจากข้อมูลที่มีอยู่แล้ว หรือจากเวชระเบียน (registry)
 3. การศึกษาแบบ diagnostic test
 4. อื่น ๆ
2. การศึกษาความสัมพันธ์
 1. การศึกษาเชิงสังเกต
 1. การศึกษาภาคตัดขวาง (cross-sectional)
 2. การศึกษาแบบ case-control
 3. การศึกษาแบบ cohort
 4. การศึกษาแบบ nested case-control
 5. การศึกษาแบบ repeated measures
 6. อื่น ๆ
 2. การศึกษาเชิงทดลอง
 1. การศึกษาแบบ randomized controlled trial
 2. การศึกษาแบบ cross-over clinical trial
 3. อื่น ๆ

5.1 การคำนวณทางสถิติในการศึกษาแบบพรรณนา

ในการศึกษาแบบพรรณนานั้น สถิติที่ใช้ก็จะเป็นสถิติในเชิงพรรณนา ได้แก่การแจกแจงความถี่ หรือสัดส่วน (proportion) หรือบอกเป็นค่าเฉลี่ยหรือมัธยฐาน หรือบอกเป็นค่าดัชนีสุขภาพ เช่น อุบัติการณ์ ความชุก อัตราตาย อัตราการอยู่รอด แม้จะดูเหมือนว่าค่าดัชนีสุขภาพเป็นเรื่องที่ต้องคำนวณ ซับซ้อน แท้จริง ๆ แล้วความหมายของดัชนีสุขภาพส่วนใหญ่จะคล้ายคลึงกับการแจกแจงความถี่ เช่น อัตราอุบัติการณ์ของการเกิดมะเร็งในประชากรในพื้นที่หนึ่งในช่วงเวลาหนึ่ง ก็คำนวณจากจำนวนคนที่ เป็นมะเร็งใหม่ในพื้นที่นั้นในช่วงเวลานั้นหารด้วยจำนวนประชากรทั้งหมดในพื้นที่นั้นที่กลางช่วงเวลานั้น ซึ่งก็就会有ความหมายเช่นเดียวกับการแจกแจงความถี่นั่นเอง แต่เนื่องจากค่าที่ได้มีค่าน้อย ก็จะนิยม ใช้หน่วยเป็น ต่อแสนประชากร แทนที่จะเป็นร้อยละ สำหรับดัชนีอย่างอื่นอาจใช้หน่วยเป็นต่อหนึ่ง พันหรือต่อหนึ่งหมื่น ก็ได้แล้วแต่ความนิยมของผู้ที่อยู่ในสาขานั้นๆ

ในกรณีเช่นนี้การแสดงความสำคัญทางสถิติจะนิยมบอกเป็นช่วงความเชื่อมั่นร้อยละ 95 ซึ่ง คำนวณจากการหาค่า standard error (SE) แม้ว่าการคำนวณค่า SE สำหรับดัชนีสุขภาพแต่ละอย่าง อาจ会有ความแตกต่างกันไปบ้าง แต่ความหมายก็จะเป็นเช่นเดียวกันทั้งหมด คือเป็นความเบี่ยงเบนที่อาจ เกิดขึ้นกับการประมาณค่าดัชนีนั้นจากการสุ่มตัวอย่างในประชากรหนึ่งในช่วงเวลาหนึ่งนั่นเอง ในที่นี้ จะกล่าวถึงเฉพาะการคำนวณค่า SE ของค่าสัดส่วนตามปกติ ซึ่งมีค่าดังนี้

$$SE = \sqrt{\frac{p(1-p)}{n}}$$

โดยที่ p คือค่า proportion นั้น N คือจำนวนตัวอย่างทั้งหมด

ดังตัวอย่าง การศึกษาหนึ่งพบว่าในช่วงปี 1995-1997 มีประชากรชายในจังหวัดสงขลาเป็น มะเร็งปอดทั้งหมด 278 คน เป็นชาย 192 คน และหญิง 86 คน

จะได้ว่า สัดส่วนของชายที่เป็นมะเร็งปอดในจังหวัดสงขลาในช่วง 1995-1997 เป็น $192/278 = 0.6906$ หรือ 69.1%

$$SE = \sqrt{\frac{0.6906(1-0.6906)}{278}}$$

แต่การบอกค่า SE โดยตรงเช่นนี้ไม่สื่อถึงความคลาดเคลื่อนที่เกิดขึ้นจากการศึกษาโดยตรง จะต้องนำไปคำนวณเป็นช่วงความเชื่อมั่นร้อยละ 95 (95% confidence interval) เสียก่อน ดังนี้

$$95\% CI = p \pm Z_{\alpha/2} SE$$

ดังนั้นในตัวอย่างข้างต้น จะได้

$$95\% \text{ CI} = 0.6906 \pm (1.96 \times 0.0277)$$

$$= 0.6362, 0.7449$$

ทำให้สรุปว่า สัดส่วนของชายที่เป็นมะเร็งปอดเป็นร้อยละ 69.1 โดยที่ค่าจริงอาจอยู่ในช่วงระหว่าง 63.6 ถึง 74.5 ด้วยความเชื่อมั่นร้อยละ 95 ซึ่งก็มีความหมายว่า ในความเป็นจริงแล้ว สัดส่วนระหว่างเพศชายที่เป็นมะเร็งปอดในพื้นที่ใกล้เคียงสงขลา หรือในปีอื่นๆ ที่ใกล้เคียงกันนี้ น่าจะเบี่ยงเบนอยู่ระหว่างร้อยละ 63.6 ถึง 74.5

อาจมีผู้ค้านว่าการศึกษาที่เป็นการนับจำนวนผู้ป่วยทั้งหมดในช่วงเวลานั้นแล้ว ก็ไม่มีความจำเป็นจะต้องคำนวณช่วงความเชื่อมั่นร้อยละ 95 แต่เพียงสังเกตว่าวัตถุประสงค์ของการศึกษาวิจัยใด ๆ นั้น เราต้องการหาค่าประมาณที่นำไปใช้ในกรณีอื่นๆ ด้วย ไม่ใช่รายงานแต่ค่าที่ได้ในการศึกษานั้น กรณีเช่นนี้จึงจำเป็นต้องคำนวณช่วงความเชื่อมั่นร้อยละ 95 เสมอเมื่อบอกค่าใดค่าหนึ่งออกไป เพื่อให้นำไปใช้กับประชากรอื่นหรือเวลาอื่นที่คล้ายคลึงกันได้ด้วย

ในการวิจัยแบบ diagnostic test นั้น เราต้องการหาความไว (sensitivity) และความจำเพาะ (specificity) ของการทดสอบหนึ่งๆ ในการหาช่วงความเชื่อมั่นร้อยละ 95 ของค่าความไวและความจำเพาะนั้น ก็ใช้การคำนวณในลักษณะเดียวกันนี้ โดยแทนค่า p ด้วยความไว หรือความจำเพาะที่ต้องการหา นั่น แต่เพียงระลึกไว้ว่าค่า N ในการหาช่วงความเชื่อมั่นของค่าความไวและความจำเพาะนั้น แตกต่างกัน

ความไว (sensitivity) คือสัดส่วน (proportion) ของคนที่ เป็นโรคจริงๆ ในประชากร ที่การตรวจก็ให้ผลว่าเป็นโรคเช่นกัน

ความจำเพาะ (specificity) คือสัดส่วน (proportion) ของคนที่ ไม่เป็นโรคจริงๆ ในประชากร ที่การตรวจก็ให้ผลว่าไม่เป็นโรค

เนื่องจากคำนิยามนี้ค่อนข้างทำความเข้าใจได้ยากหากไม่อธิบาย ขอให้พิจารณาจากตาราง 2×2 ต่อไปนี้

		disease	
		+	-
test	+	a	b
	-	c	d

ในที่นี้จำนวนคนที่ เป็นโรคในประชากรมีค่าเท่ากับ $a+c$ และในจำนวนนี้มี a คนที่ผลการตรวจก็ให้ผลว่าเป็นโรคด้วยเช่นกัน นั่นคือ

$$\text{ความไว(sensitivity)} = \frac{a}{a+c}$$

และจำนวนคนที่ไม่เป็นโรคในประชากรมีค่าเท่ากับ $b+d$ และในจำนวนนี้มี d คนที่ผลการตรวจก็ให้ผลว่าไม่เป็นโรคด้วยเช่นกัน นั่นคือ

$$\text{ความจำเพาะ(specificity)} = \frac{d}{b+d}$$

จะเห็นว่าในกรณีของความไว N มีค่าเท่ากับ $a+c$ และในกรณีของความจำเพาะ N มีค่าเท่ากับ $b+d$ ไม่ใช่ N รวมทั้งหมดของการวิจัย

5.2 การคำนวณทางสถิติในการวิจัยความสัมพันธ์

5.2.1 การทดสอบ Chi-squared และ Fisher's exact test

การทดสอบ Chi-squared และ Fisher's exact test เป็นการทดสอบเพื่อหาความสัมพันธ์ระหว่างตัวแปร โดยการทดสอบ Chi-squared จะต้องไม่มีความถี่คาดหวังเซลล์ใดเซลล์หนึ่งที่น้อยกว่า 5 เกิน 25% ของตาราง $n \times n$ หรือมีความถี่สังเกตน้อยกว่า 1 หากไม่เป็นไปตามกฎนี้ จึงควรต้องหันไปใช้ Fisher's exact test แทน การทดสอบ Chi-squared โดยมีสูตรในการคำนวณดังนี้ คือ

$$\chi^2 = \sum \frac{(O - E)^2}{E}$$

โดยที่

$$E = \frac{\text{ความถี่รวมในแถว} \times \text{ความถี่รวมในแนวสดมภ์}}{\text{จำนวนรวมทั้งหมด}}$$

$$df = (\text{จำนวนแถว} - 1)(\text{จำนวนสดมภ์} - 1)$$

O คือ ค่าความถี่จากการสังเกต

E คือ ค่าความถี่จากการคาดหวัง

		ตัวแปรที่ 1		
		+	-	
ตัวแปรที่ 2	+	a	b	r_1
	-	c	d	r_0
		c_1	c_0	N

สำหรับ Fisher's exact test มีสูตรในการคำนวณ คือ

$$\text{Fisher's exact test} = \frac{r_1! \times r_0! \times c_1 \times c_0}{N! \times a! \times b! \times c! \times d!}$$

5.2.1.1 คำสั่ง `tab()` และ `xtab()`

คำสั่งใน library stats สำหรับการทดสอบ Chi-squared และ Fisher's exact test ซึ่งจะสามารถเรียกใช้ได้ทันที ได้แก่ คำสั่ง `chisq.test()` และ `fisher.test()` แต่เนื่องจากคำสั่งทั้งสองนี้ไม่ได้แสดงตารางข้อมูลประกอบด้วย และรูปแบบการแสดงผลก็ดูค่อนข้างยาก ใน library `ice` จึงได้มีคำสั่งที่แสดงตารางข้อมูลประกอบกับการทดสอบทั้งสองให้เลย ได้แก่ `tab` และ `xtab` ซึ่งมีรูปแบบการออกคำสั่งดังนี้

```
tab(data, x, y, group, row=get.ec.option("row"), col=get.ec.option(
  "row"), test=get.ec.option("test"), frame=get.ec.option(
  "frame"), na.included=FALSE, factor.labels=TRUE)
```

data	กรอบข้อมูล (อาจไม่ใส่ก็ได้)
x, y	เวกเตอร์ที่จะนำมาสร้างตารางไขว้ ถ้าระบุแค่ 'x' เพียงค่าเดียว จะสร้างตารางความถี่
group	ตัวเลือกตัวแปรแบ่งกลุ่ม (อาจไม่มีก็ได้)
pct	แสดงเป็นร้อยละด้วย ค่าปกติเป็น FALSE
row	แสดงตารางร้อยละด้านแถว ค่าปกติเป็น FALSE
col	แสดงร้อยละด้านสดมภ์ ค่าปกติเป็น FALSE
test	ทดสอบ chi-squared และ Fisher's exact test ค่าปกติเป็น FALSE

<code>frame</code>	วาดกรอบให้กับตาราง ค่าปกติเป็น TRUE
<code>na.included</code>	แสดง NA เป็นอีกระดับหนึ่งหรือไม่ ค่าปกติคือ FALSE
<code>factor.labels</code>	แสดงแฟกเตอร์โดยคำอธิบายระดับหรือค่า ค่าปกติคือแสดงโดยคำอธิบาย (TRUE)

```
xtab(data, key, columns, row=get.ec.option("row"),
      colm=get.ec.option("colm"), tesst=get.ec.option("test"),
      position=get.ec.option("position"), frame=get.ec.option
("frame"), na.included=FALSE, factor.labels=TRUE)
```

<code>data</code>	กรอบข้อมูล
<code>key</code>	ตัวแปรหลักที่ใช้สร้างตารางความสัมพันธ์ไว้กับตัวแปรอื่นๆ สามารถระบุเป็นตัวเลขลำดับสดมภ์ในกรอบข้อมูลหรือระบุเป็นชื่อตัวแปรก็ได้
<code>columns</code>	สดมภ์ต่างที่จะนำมาสร้างตารางความถี่หรือที่จะนำมาสร้างตารางความสัมพันธ์ไว้กับตัวแปรหลัก <code>key</code> อาจระบุเป็นเวกเตอร์ของเลขลำดับสดมภ์หรือเวกเตอร์ของชื่อสดมภ์ก็ได้
<code>row</code>	แสดงร้อยละด้านแถว (แนวนอน) ค่าปกติเป็น FALSE
<code>colm</code>	แสดงร้อยละด้านสดมภ์ (แนวตั้ง) ค่าปกติเป็น FALSE
<code>test</code>	ทดสอบนัยสำคัญด้วยวิธี chi-squared และ Fisher's exact test ค่าปกติเป็น FALSE
<code>position</code>	ตำแหน่งที่จะวางสดมภ์ <code>key</code> ค่าปกติคือ "row" ถ้าต้องการให้อยู่ด้านสดมภ์ ให้ระบุเป็น "column" หรือเขียนอย่างย่อเป็น "col" หรือ "c" ก็ได้
<code>frame</code>	วาดกรอบให้ตาราง ค่าปกติเป็น TRUE
<code>na.included</code>	แสดง NA เป็นอีกระดับหนึ่งหรือไม่ ค่าปกติคือ FALSE
<code>factor.labels</code>	แสดงแฟกเตอร์โดยคำอธิบายระดับหรือค่า ค่าปกติคือแสดงโดยคำอธิบาย (TRUE)

ตัวอย่างที่ 5.1 จากข้อมูลวิจัยมะเร็ง cholangiocarcinoma ในจังหวัดนครพนม (data(cca)) จงทดสอบความแตกต่างของตัวแปรต่างๆ เมื่อเทียบกับตัวแปร status

```
> data(cca)
> summ(cca)
> xtab(cca, status, c(2:4), test=TRUE)
row: status, column: ovab
```

	negative	positive	Total
control	118	11	129
case	62	65	127
Total	180	76	256

Chi-squared = 53.75, df = 1, p-value = 0
Fisher's exact test (2-sided) p-value = 0

```
row: status, column: alcohol
```

	never	occasional	ex-drinker	drinker	Total
control	48	60	7	13	128
case	31	40	17	40	128
Total	79	100	24	53	256

Chi-squared = 25.58, df = 3, p-value = 0
Fisher's exact test (2-sided) p-value = 0

```
row: status, column: smoke
```

	never	occasional	ex-smoker	smoker	Total
control	64	6	14	44	128
case	47	8	21	52	128
Total	111	14	35	96	256

Chi-squared = 4.956, df = 3, p-value = 0.175
Fisher's exact test (2-sided) p-value = 0.176

จากตัวอย่างข้างต้นตัวแปร status มีความสัมพันธ์กับตัวแปร ovab และ alcohol อย่างมีนัยสำคัญทางสถิติ ในขณะที่ไม่พบความสัมพันธ์ที่มีนัยสำคัญทางสถิติกับตัวแปร smoke

5.2.1.2 คำสั่ง `mosaicplot()`

เป็นคำสั่งที่สร้างแผนภาพ mosaic plot เพื่อแสดงขนาดจำนวนของแต่ละกลุ่มด้วยกราฟ คำสั่งนี้

อยู่ใน library graphics มีรูปแบบการออกคำสั่งดังนี้

```
mosaicplot(x, main = deparse(substitute(x)),
            sub = NULL, xlab = NULL, ylab = NULL,
            sort = NULL, off = NULL, dir = NULL,
            color = FALSE, shade = FALSE, margin = NULL,
            cex.axis = 0.66, las = par("las"),
            type = c("pearson", "deviance", "FT"), ...)
```

①

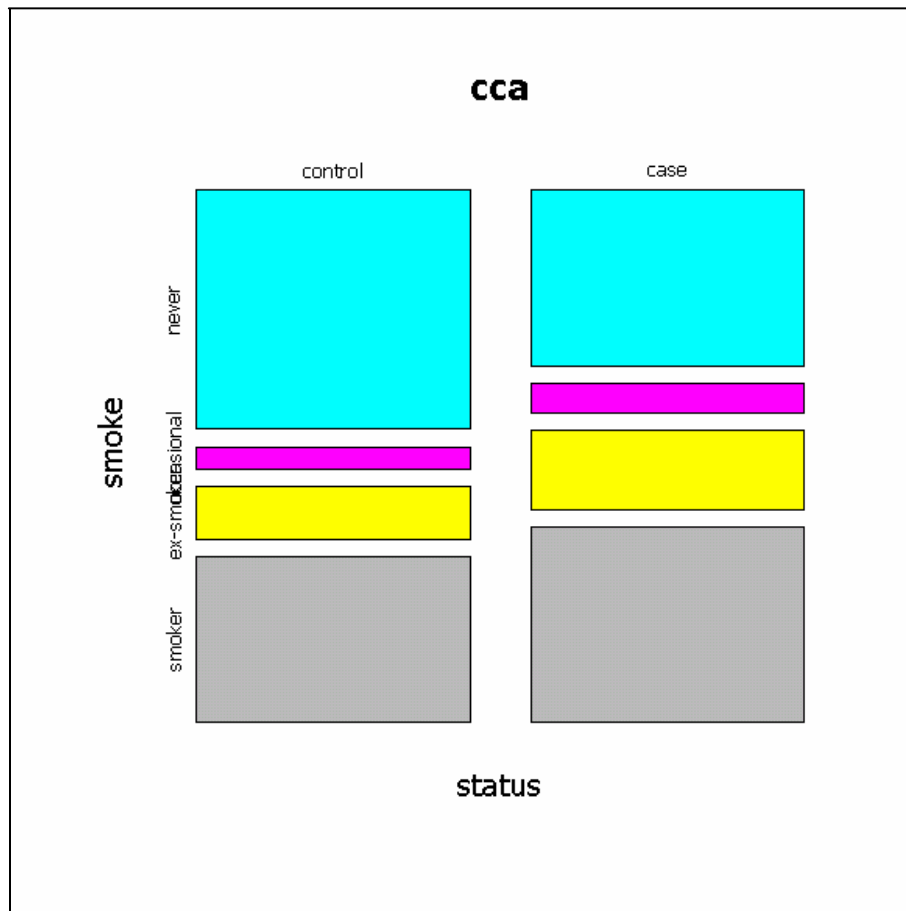
```
mosaicplot(formula, data = NULL, ...,
            main = deparse(substitute(data)), subset,
            na.action = stats::na.omit)
```

②

x	ตารางที่รวมจำนวนความถี่ของแต่ละกลุ่ม
main	คำอธิบายหัวเรื่องหรือชื่อกราฟ
sub	คำอธิบายหัวเรื่องย่อย (ด้านล่างกราฟ)
xlab, ylab	คำอธิบายแกน x และ แกน y
sort	การเรียงลำดับเวกเตอร์ของตัวแปร
off	ค่าที่บอกระยะห่างระหว่างแต่ละ mosaic ค่าที่เหมาะสมอยู่ระหว่าง 0 – 20
dir	เวกเตอร์ที่บอกทิศทาง เช่น v คือ แนวตั้ง h คือแนวนอน
color	การเลือกสีสำหรับแต่ละ mosaic
shade	การให้เฉดสีบอกจุดตัดของ residual
las	รูปแบบการให้คำอธิบายแกน
formula	เป็นสมการที่ต้องการ plot เช่น $\sim y+x$
data	ชื่อกรอบข้อมูล
subset	การเลือกข้อมูลย่อยในกรอบข้อมูล
na.action	ในกรณีของข้อมูลที่มีค่า missing ให้แสดงค่าเหล่านี้หรือไม่

ในที่นี้ใช้ตัวอย่างคำสั่งที่ ② ดังตัวอย่างการใช้คำสั่ง

```
> mosaicplot(~status+smoke, col=s:8, data=cca)
```



รูปที่ 5.1 กราฟ mosaic plot แสดงขนาดตามจำนวนความถี่ในแต่ละกลุ่ม

จากกราฟจะเห็นได้ว่ากลุ่ม control มีคนจำนวนคนที่ดื่ม หรือเคยดื่มแอลกอฮอล์น้อยกว่ากลุ่ม case อย่างชัดเจน ในขณะที่กลุ่มคนที่ไม่ดื่มแอลกอฮอล์ในกลุ่ม control มีสูงกว่ากลุ่ม case

5.2.1.3 คำสั่ง `assocplot()`

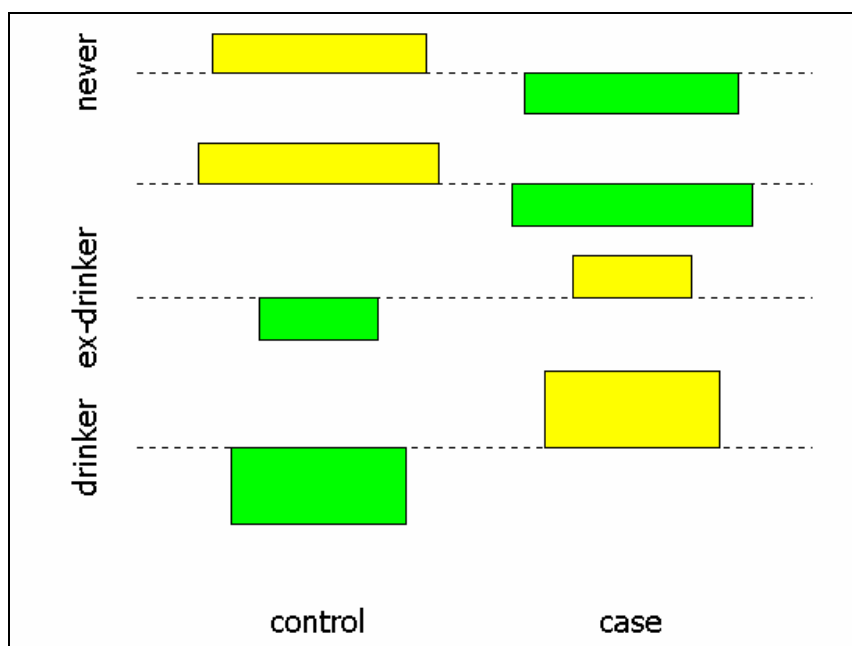
เป็นคำสั่งที่สร้างแผนภาพ Cohen-Friendly association plot เพื่อศึกษาความสัมพันธ์ในแต่ละกลุ่ม ความสูงของแต่ละแท่งแต่ละแท่งเป็นสัดส่วนตามค่า Chi-squared ในแต่ละเซลล์ ส่วนความกว้างเป็นสัดส่วนตามค่าความถี่คาดหวัง ดังนั้นพื้นที่ของแต่ละแท่งแสดงสัดส่วนของความแตกต่างระหว่างค่าความถี่จากการสังเกต กับความถี่คาดหวัง แท่งในแต่ละแถว ถ้ามีค่าความถี่สังเกตสูงกว่าความถี่คาดหวัง แท่งก็จะอยู่บนเส้นประ แต่ถ้าน้อยกว่าแท่งก็จะอยู่ใต้เส้นประ คำสั่งนี้อยู่ใน library graphics มีรูปแบบการออกคำสั่งดังนี้

```
assocplot(x, col = c("black", "red"), space = 0.3,
          main = NULL, xlab = NULL, ylab = NULL)
```

x	ตารางที่รวมจำนวนความถี่ของแต่ละกลุ่ม
col	การเลือกสีสำหรับแต่ละกลุ่ม
space	กำหนดระยะห่างระหว่างแผนภูมิแต่ละแท่ง
main	คำอธิบายหัวเรื่องหรือชื่อกราฟ
xlab, ylab	คำอธิบายแกน x และ แกน y

ดังตัวอย่างการใช้คำสั่ง

```
> a <- tab(cca, status, alcohol)
> assocplot(a, col=c("yellow", "green"))
```



รูปที่ 5.2 กราฟ Cohen-Friendly association plot

5.2.2 การคำนวณเพื่อหาขนาดความเสี่ยง

รูปแบบการวิจัยที่ใช้ในการหาปัจจัยเสี่ยงของโรค หรือปัจจัยป้องกันโรคนั้น อาจเรียกรวมกันว่า เป็นการวิจัยเชิงวิเคราะห์ (analytical studies) แบ่งได้เป็นสองกลุ่มหลัก คือการวิจัยเชิงสังเกต (observational) และการวิจัยเชิงทดลอง (experimental) ในการวิจัยเชิงสังเกตนั้น ไม่ได้มีการให้ปัจจัยที่ต้องการศึกษาแก่คนที่ผู้วิจัยสังเกต เพียงแต่แบ่งกลุ่มผู้ถูกวิจัยเป็นกลุ่มๆ ตามปัจจัย และ/หรือโรคที่คนเหล่านั้นได้รับเองหรือเป็นโรคอยู่ ส่วนการวิจัยเชิงทดลองนั้น ผู้วิจัยได้ให้ปัจจัยเสี่ยง (ที่ไม่เป็นอันตรายถึงแก่ชีวิตหรือพิการ) หรือโดยส่วนใหญ่แล้ว จะเป็นยาหรือสิ่งที่ใช้ป้องกันโรค แล้วตามดูผลที่จะเกิดขึ้นในการป้องกันโรคหรือภาวะแทรกซ้อนอันตราย หรือแม้กระทั่งการเสียชีวิต

ในกลุ่มการวิจัยเชิงสังเกตนั้น มีวิธีการวิจัยหลักอยู่ 2 แบบคือ การศึกษาแบบ case-control และการศึกษาแบบ cohort ซึ่งการศึกษาแบบ case-control นั้น เป็นการศึกษาย้อนหลัง โดยเลือกผู้ที่เป็นโรคแล้ว กับผู้ที่ไม่เป็นโรคจากประชากรเดียวกัน มาศึกษาย้อนหลังว่าได้รับปัจจัยเสี่ยง (exposure) ที่สนใจบ้างหรือไม่บ้าง มากน้อยเพียงใด แล้วนำมาเปรียบเทียบกัน ส่วนการศึกษาแบบ cohort นั้น จะแบ่งกลุ่มผู้เข้าร่วมการศึกษาตามปัจจัยเสี่ยงที่ได้รับตามปกติที่เขาเป็นอยู่ โดยผู้วิจัยไม่ได้ตั้งใจให้ปัจจัยนั้นแก่บุคคลเหล่านั้น แล้วติดตามในระยะยาวว่ากลุ่มคนเหล่านั้นต่อมาเป็นโรคอะไรบ้าง แตกต่างกันระหว่างกลุ่มที่มี และไม่มีปัจจัยเสี่ยงอย่างไร

ในกลุ่มการวิจัยเชิงทดลองนั้น วิธีการหลักที่นิยมกันคือ การศึกษาแบบ randomized controlled

trial (RCT) ซึ่งแบ่งกลุ่มผู้เข้าร่วมวิจัยเป็นกลุ่มๆ อย่าง random แล้วจึงให้ปัจจัยที่ต้องการศึกษาแก่กลุ่มเหล่านั้นแตกต่างกันทั้งกลุ่ม แล้วดูผลที่เกิดขึ้นต่อกลุ่มเหล่านั้นทั้งกลุ่ม มีการวิจัยแบบอื่นๆ อีก เช่น การศึกษาแบบ cross-over ซึ่งจะแบ่งกลุ่มผู้เข้าร่วมวิจัยเช่นเดียวกัน แต่เมื่อให้ปัจจัยหนึ่งไปแล้วระยะหนึ่ง วัดผลที่เกิดขึ้นแล้ว ก็จะหยุดให้จนหมดฤทธิ์ของปัจจัยนั้น แล้วสลับให้ปัจจัยอีกอย่างหนึ่งแก่กลุ่มที่ไม่ได้รับตั้งแต่ครั้งแรก ในกรณีนี้การเปรียบเทียบจะทำในคนเดียวกันนั่นเอง ผลที่ได้รับปัจจัยหนึ่งเทียบกับผลที่เกิดขึ้นจากอีกปัจจัยหนึ่ง ซึ่งจะกำจัดการแปรปรวน (variation) ที่อาจเกิดจากการแบ่งกลุ่มผู้วิจัยไม่สมดุลกัน

ในการวิจัยกลุ่มต่างๆ ที่กล่าวมาแล้วนี้ แต่ละอย่างยังมีรายละเอียดปลีกย่อยที่แตกต่างกันได้อีก ทำให้เรียกชื่อวิธีการวิจัยแตกต่างกันออกไปอีก ซึ่งจะไม่กล่าวถึงในที่นี้ แต่จะกล่าวถึงเฉพาะหลักการทั่วไปของการวิเคราะห์ปัจจัยเสี่ยงและปัจจัยป้องกัน

เพื่อให้ง่ายต่อการทำความเข้าใจ ในเบื้องต้นขอให้พิจารณาจากกรณีของตัวแปร แบบ dichotomous คือเป็นไปได้สองทางดังต่อไปนี้

ปกติในประชากรกลุ่มใดกลุ่มหนึ่งย่อมมีทั้งผู้ที่เป็นโรคและผู้ที่ไม่เป็นโรคอยู่ปะปนกัน และมีทั้งผู้ที่มีและไม่มีปัจจัยที่สนใจว่าจะเป็นปัจจัยเสี่ยง ซึ่งอาจแสดงด้วยตาราง 2×2 ดังนี้

		การเกิดโรค D		
		+	-	
การได้รับปัจจัยเสี่ยง E	+	a	b	n_1
	-	c	d	n_0
		m_1	m_0	N

ดังที่ทราบอยู่แล้วว่าในกระบวนการเกิดโรคนั้นการได้รับปัจจัยเสี่ยงย่อมเกิดขึ้นก่อนการเป็นโรค ดังนั้นเมื่อคำนึงถึงการได้รับปัจจัยเสี่ยง โอกาสที่จะเกิดโรค D เมื่อได้รับปัจจัยเสี่ยง E มีค่าเท่ากับ a/n_1 และโอกาสเกิดโรค D เมื่อไม่ได้รับปัจจัยเสี่ยง E มีค่าเท่ากับ c/n_0 ขนาดของความเสี่ยงสัมพัทธ์ที่จะเกิดโรคถ้าได้รับปัจจัยเสี่ยง เมื่อเทียบกับการไม่ได้รับปัจจัยเสี่ยง (Relative risk หรือ Risk ratio เรียกย่อว่า RR) จะเป็น

$$\text{Relative risk (RR)} = \frac{a/n_1}{c/n_0} = \frac{a/(a+b)}{c/(c+d)}$$

ยังมีอีกวิธีหนึ่งที่ใช้ในการบอกขนาดของความเสี่ยงที่จะเกิดโรคโดยใช้ค่าอัตราส่วนระหว่าง a กับ b แทนที่จะใช้ค่าสัดส่วน a/n_1 ดังข้างต้น และใช้ค่าอัตราส่วนระหว่าง c กับ d แทนที่จะใช้ค่าสัดส่วน

c/n_0 ค่าที่ได้นี้เรียกว่า Odds ratio (OR) ซึ่งมีค่าดังนี้

$$\text{Odds ratio (RR)} = \frac{a/b}{c/d} = \frac{ad}{bc}$$

ส่วนค่า variance ของ $\ln(\text{OR})$ หาได้โดยถือเสียว่าการกระจายของ $\ln(\text{OR})$ เป็นแบบปกติ มีสูตรการคำนวณดังนี้

$$\text{SE}(\ln(\text{OR})) = \sqrt{\frac{1}{a} + \frac{1}{b} + \frac{1}{c} + \frac{1}{d}}$$

ดังนั้นช่วงความเชื่อมั่นของ $\ln(\text{OR})$ จะมีค่า

$$95\% \text{ CI} = \ln(\text{OR}) \pm Z_{\alpha/2} \text{SE}$$

โดยปกติแล้ว เราต้องการหาค่า RR เพื่อจะบอกความเสี่ยงสัมพัทธ์ของการเกิดโรคถ้าได้รับปัจจัยเสี่ยงเทียบกับการไม่ได้รับปัจจัยเสี่ยง ขณะที่ค่า OR นั้นไม่ได้ให้ความหมายเช่นนี้โดยตรง ซึ่งทำให้เข้าใจได้ยากกว่า แต่เราจะพบว่าเมื่อ RR มีค่าสูง ค่า OR ก็จะมีค่าสูงด้วยเช่นกัน และเมื่อ RR มีค่าต่ำ ค่า OR ก็จะมีค่าต่ำด้วยเช่นกัน

และนอกจากนี้ หากในประชากรนั้นๆ มีความชุกของโรคน้อย นั่นคือว่า b มีค่ามากกว่า a มาก จนทำให้ค่า $a+b$ มีค่าไม่ต่างจาก a และ d มีค่ามากกว่า c มาก จนทำให้ค่า $c+d$ มีค่าเท่ากับ d เราจะพบว่า relative risk จะมีค่าเท่ากับ odds ratio ดังนี้

$$\text{Relative risk (RR)} = \frac{a/(a+b)}{c/(c+d)} = \frac{a/b}{c/d} = \frac{ad}{bc} = \text{Odds ratio}$$

และหาก relative risk มีค่าเท่ากับ 1 แล้ว เราจะได้ว่า

$$a/(a+b) = c/(c+d)$$

$$a(c+d) = c(a+b)$$

$$ac + ad = ac + bc$$

$$ad = bc$$

นั่นคือว่ากรณีนั้น odds ratio จะมีค่าเท่ากับ 1 ด้วย ทำให้อาจสรุปได้ว่า เมื่อค่า RR มีค่าใกล้เคียง 1 (เช่น ประมาณ ไม่เกิน 3 หรือ ไม่น้อยกว่า 1/3) ค่าของ OR ก็จะมีค่าใกล้เคียงกับ RR

ข้อสรุปเช่นนี้แม้จะดูพินิจ แต่มีความสำคัญมากต่อการวิจัยทางระบาดวิทยา เนื่องจากในบางกรณี เช่น ในการศึกษาแบบ case-control นั้น เราเริ่มต้นการวิจัยด้วยการหาผู้ที่เป็นโรคและหาผู้ที่ไม่เป็น

โรค แล้วจึงหาประวัติการได้รับหรือไม่ได้รับปัจจัยที่สนใจในภายหลัง ในกรณีเช่นนี้ ค่า $a+b$ และค่า $c+d$ จะไม่มีความหมายใดๆ เนื่องจากเราอาจเลือกสัดส่วนของผู้ที่เป็นโรคต่อผู้ที่ไม่เป็นโรคเป็นเท่าใดก็ได้ตามแต่ความพอใจ ค่าสัดส่วนนี้ไม่ใช่ค่าจริงในประชากร ดังนั้นในการศึกษาแบบ case-control จะไม่สามารถหาค่า RR ได้ แต่จะหาค่า OR ได้ เราจึงจำเป็นต้องอนุมานค่า RR จากค่า OR นั้น และค่า RR ที่อนุมานมาจาก OR นั้น จะมีค่าใกล้เคียง RR จริงๆ ในกรณีที่โรคนั้นมีความชุกหรืออุบัติการณ์น้อยในประชากร และค่า OR ไม่ห่างจาก 1 มากนัก

ในการวิจัยเชิงวิเคราะห์ทางระบาดวิทยานิยมคำนวณหา OR มากกว่า RR แล้วอนุมานเอาว่า RR มีค่าใกล้เคียงกับ OR ทั้งนี้เนื่องจากค่า OR เป็นค่าที่หาได้ไม่ว่าจะเป็นรูปแบบการวิจัยแบบ cross-sectional, case-control หรือ cohort ก็ตาม และนอกจากนี้ โมเดลทางสถิติแบบ logistic regression ส่วนใหญ่จะให้ค่า OR แทนที่จะให้ค่า RR ในรูปของ exponential ของค่าสัมประสิทธิ์ของตัวแปรที่สนใจ ดังนี้

$$OR = \exp(\beta)$$

ดังนั้นจึงมีความจำเป็นที่จะต้องอนุมานค่า RR จากค่า OR ถึงแม้ว่าในการศึกษานั้นจะเก็บข้อมูลการได้รับปัจจัยที่สนใจก่อนที่จะเป็นโรคก็ตาม ทั้งนี้เพราะไม่สามารถหาค่า RR จากโมเดลสถิติได้โดยตรงนั่นเอง ดังนั้นพึงระลึกไว้เสมอว่า หากโรคที่สนใจนั้นเป็นโรคที่พบบ่อยในประชากร หรือค่า OR มีค่าค่อนข้างสูง ค่า RR ที่อนุมานโดยวิธีนี้จะคลาดเคลื่อนจากความเป็นจริงได้มาก ในกรณีเช่นนี้ควรบอกว่าเป็นค่า OR โดยตรงจะดีกว่าที่จะอนุมานว่าเป็นค่าของ RR

ในการวิจัยทางระบาดวิทยาจริงๆ แล้ว เรามักไม่คำนวณหาค่า RR หรือ OR โดยวิธีง่ายๆ ดังที่กล่าวข้างต้น เนื่องจากต้องคำนึงถึงปัจจัยกวนอื่นๆ ด้วย แต่จะใช้การคำนวณโดยใช้โมเดลทางสถิติ ซึ่งจะได้กล่าวถึงในบทต่อไป

การกวน (Confounding)

การกวนเป็นการเบี่ยงเบนของความสัมพันธ์ระหว่างโรคกับปัจจัยที่ได้รับโดยปัจจัยอื่นๆ ซึ่งมีความสัมพันธ์กับทั้งโรคและทั้งปัจจัยที่สนใจนั้น และความสัมพันธ์ระหว่างตัวกวนกับโรคยังเป็นความสัมพันธ์ในเชิงสาเหตุอีกด้วย ปัจจัยที่กวนนั้นเรียกว่า ตัวกวน (confounder หรือ confounding factor) ซึ่งอาจแสดงเป็นแผนภูมิได้ดังนี้



ปัจจัยเสี่ยง E กับโรค D ดังนี้

ปัจจัยกวน C +				ปัจจัยกวน C -					
D				D					
		+	-			+	-		
E	+	a ₁	b ₁	n ₁₁		+	a ₂	b ₂	n ₁₂
	-	c ₁	d ₁	n ₀₁		-	c ₂	d ₂	n ₀₂
		m ₁₁	m ₀₁	N			m ₁₂	m ₀₂	N
		OR ₁					OR ₂		

ในกรณีนี้ ความสัมพันธ์ระหว่างปัจจัยเสี่ยง E กับโรค D ในคราวนี้จะไม่ขึ้นกับปัจจัยกวน C อีกต่อไป ทั้งนี้เพราะในแต่ละตาราง ทุกรายได้รับปัจจัยกวนเหมือนกันทั้งหมดในตารางด้านซ้าย และไม่ได้รับปัจจัยกวนเหมือนกันทั้งหมดในตารางด้านขวา

เป็นไปได้ที่ $OR_1 = OR_2 = OR$ (ซึ่งไม่เท่ากับ OR_p ข้างต้น) ซึ่งหมายความว่าความสัมพันธ์ระหว่างปัจจัยเสี่ยง E กับโรค D มีค่าเท่ากัน ไม่ว่าจะในกลุ่มประชากรที่มีปัจจัยกวน C ด้วยหรือไม่ และก็เป็นไปได้เช่นกันที่ OR_1 ไม่เท่ากับ OR_2 ในเบื้องต้นนี้ขอคิดในกรณีที่ OR_1 และ OR_2 มีค่าเท่ากัน และเป็นค่าความเสี่ยงจริงในประชากรด้วย ไม่ใช่แค่ความเสี่ยงในกลุ่มศึกษาเท่านั้น

ในกรณีนี้ OR เป็นความสัมพันธ์ระหว่างปัจจัยเสี่ยง E กับโรค D หลังจากกำจัดปัจจัยกวน C ไปแล้ว นั่นการกวนเกิดขึ้นก็ต่อเมื่อ

1. C และ E สัมพันธ์กันในกลุ่มที่ไม่เป็นโรค (และเนื่องจาก $OR_1 = OR_2$ ด้วย ดังนั้นจึงสัมพันธ์กันในกลุ่มที่เป็นโรคด้วย)
2. ปัจจัย C สัมพันธ์กับโรค ถ้าแบ่งชั้นตามปัจจัย E

นั่นคือ เราอาจพูดอย่างกว้างๆ ว่า ปัจจัย C สัมพันธ์กับทั้งปัจจัยเสี่ยงและกับโรค และความสัมพันธ์ระหว่าง C กับ E ต้องพิจารณาแยกกันในกลุ่มที่เป็นโรคกับกลุ่มที่ไม่เป็นโรค และเช่นเดียวกัน ความสัมพันธ์ระหว่าง C กับโรค ก็ต้องพิจารณาแยกกันในกลุ่มที่ได้รับปัจจัย E กับกลุ่มที่ไม่ได้รับปัจจัย E ความสัมพันธ์กันเองเช่นนี้จะเข้าใจได้ง่ายด้วยแผนภูมิข้างต้นนั่นเอง

การกวนทำให้ผลของความสัมพันธ์เมื่อคำนวณโดยใช้ข้อมูลทั้งหมด (OR_p) เบี่ยงเบนไปจากความเป็นจริง ($OR_1 = OR_2 = OR$) โดยที่อาจทำให้ห่างจาก unity มากขึ้น หรือทำให้เข้าใกล้มากขึ้น unity ก็ได้ และในบางกรณีที่พบไม่บ่อย อาจทำให้ค่าที่ได้อยู่คนละด้านของ unity เลยก็เป็นได้

อัตราส่วน OR_p/OR เรียกว่า confounding risk ratio เป็นวิธีวัดขนาดของการกวนได้ จำไว้ว่า

ความจริงแล้วเราไม่ได้ต้องการวัดขนาดของการกวน แต่ต้องการกำจัดอิทธิพลของการกวนในการคำนวณขนาดของความสัมพันธ์ระหว่างปัจจัยเสี่ยงกับโรคต่างหาก เราพบว่า confounding risk ratio จะมีค่ามากกว่า 1 เมื่อ

ก. C มีความสัมพันธ์เชิงบวกกับทั้ง E และ โรค ($OR_c > 1$ และ $p_1 > p_2$; โดยที่ OR_c คือ OR ของ C กับโรค p_1 คือความถี่ของการพบ C ในกลุ่มที่มีปัจจัยเสี่ยง และ p_2 คือความถี่ของการพบ C ในกลุ่มที่ไม่มีปัจจัยเสี่ยง)

ข. C มีความสัมพันธ์ในทางลบกับทั้ง E และ โรค ($OR_c < 1$ และ $p_1 < p_2$)

ทั้งปัจจัย E และ C อาจเป็นค่าที่แบ่งเป็นหลายระดับ ไม่ใช่แค่สองระดับก็ได้ ซึ่งแนวคิดของการกวนก็ยังเป็นเช่นเดียวกับตัวแปรที่แบ่งเป็นสองระดับ ส่วนรายละเอียดต่างๆ จะไม่กล่าวในที่นี้

5.2.2.1 คำสั่งที่ใช้ในการคำนวณขนาดความเสี่ยง

ใน library **epid** ซึ่งมีฟังก์ชันที่ใช้งานทางระบาดวิทยานั้น มีคำสั่งที่ใช้ในการคำนวณขนาดความเสี่ยงหลายคำสั่ง คือ **cc()**, **cs()**, **ir()**, **mcc()** และ **mhor()** ซึ่งมีรูปแบบการออกคำสั่งดังนี้

```
cc(case, expose)
cc(a, b, c, d)
cs(case, expose)
cs(a, b, c, d)
ir(case, expose, time)
ir(a, b, n1, n2)
mcc(case, control)
mcc(a, b, c, d)
mhor(case, expose, group)
```

อย่างไรก็ตาม จะกล่าวในที่นี้เพียงบางคำสั่งเท่านั้น ได้แก่ **cc()** และ **mhor()**

ดังที่กล่าวมาแล้วข้างต้น ว่าการศึกษาแบบ case-control นั้น เราคาดหวังว่าจะได้ค่า odds ratio (OR) เป็นผลของการศึกษา ซึ่งจะนำไปใช้เป็นค่าประมาณ (estimate) ของ relative risk (RR) ใน library **epid** มีคำสั่ง **cc()** ไว้ให้ คำสั่งนี้สร้างขึ้นคล้ายคำสั่ง **cc** ในโปรแกรม **Stata** โดยมีรูปแบบการออกคำสั่งดังนี้

```
cc(table, outcome=get.ec.option("outcome"), frame=get.ec.option
("frame"))
cc(data, case, expose, outcome=get.ec.option("outcome"),
frame=get.ec.option("frame"))
```

case, expose	เป็นเวกเตอร์ของ case และ expose ซึ่งต้องมีค่าที่เป็นไปได้ เพียงสองค่าเท่านั้น คือระบุว่าเป็น control หรือ case กับ unexposed หรือ exposed
table	วัตถุตาราง
data	กรอบข้อมูล (อาจไม่ระบุก็ได้)
case, expose	เวกเตอร์ของ case และ exposure ทั้งสองต้องมีค่าเป็นไปได้ สองทางเท่านั้น คือ case หรือ control กับ exposed หรือ unexposed

```
cc(a, b, c, d, outcome=get.ec.option("outcome"), frame=
get.ec.option("frame"))
```

a, b, c, d	ตัวเลขจำนวนผู้ที่อยู่ในกลุ่มต่างๆ เรียงดังนี้ case-exposed, case-unexposed, control-exposed, control-unexposed
outcome	ตำแหน่งการวาง case/outcome ในตาราง ค่าปกติเป็น "column"
frame	วาดตารางมีกรอบ ค่าปกติเป็น TRUE

ตัวอย่างที่ 5.2 จากข้อมูลวิจัยมะเร็ง cholangiocarcinoma ในจังหวัดนครพนม ลองหา odds ratio ของความสัมพันธ์ระหว่างการเป็น cholangiocarcinoma กับ antibody to *Opisthorchis viverrini* (ovab), alcohol drinking behaviour (alcohol) และ smoking behaviour (smoke)

หมายเหตุ 1 สังเกตว่าคำสั่ง `cc()` จะไม่ทำงานกับ alcohol และ smoke เนื่องจากทั้งสองมีค่า 4 ระดับ ลองใช้คำสั่ง `summary(cca)` เพื่อดูว่า alcohol และ smoke มีระดับอะไรอยู่บ้าง ลองสร้างตัวแปรใหม่ให้ alcohol และ smoke มีค่าเป็น dichotomous โดยให้ never กับ occasional รวมเป็นชั้นเดียวกัน และที่เหลือเป็นอีกชั้นหนึ่ง ก็จะสามารถใช้คำสั่ง `cc()` เพื่อหา odds ratio ของตัวแปรใหม่นี้ได้ ดังนี้

```
> cca <- transform(cca, alc2=ifelse(alcohol=="never" |
alcohol=="occasional", 0, 1))
> cca <- transform(cca, smo2=ifelse(smoke=="never" |
smoke=="occasional", 0, 1))
```

หมายเหตุ 2 คำสั่ง `cc()` จะถือว่าระดับชั้นที่มีค่าต่ำสุดเป็น control เสมอ โดยไม่สนใจว่ารหัสชั้นมีความหมายอย่างไร ดังนี้ในกรณีของ alc2 และ smo2 ที่เราใช้เลข 0 กับเลข 1 เป็นรหัสชั้น `cc()` จะตีความได้ถูกต้อง แต่ถ้าไปกำหนดให้ alc2 มีค่าเป็น "never" กับ "drink" R จะเห็นว่า "drink" มีค่าน้อยกว่า "never" (ตัวอักษร d มีค่าน้อยกว่า n) ในกรณีนี้ค่า odds ratio ที่ได้จะมีค่าเป็น 1/OR ที่ต้องการ ปัญหาในเรื่องการ label factor จะเป็นปัญหาที่เกิดขึ้นเป็นประจำใน R สำหรับผู้ใช้ใหม่ โดยทั่วไปแล้วไม่ควรใช้ label ถ้าไม่จำเป็น ให้ใช้ตัวเลขรหัสในการทำงานเสมอ

```
> cc(cca, status, ovab)
Case/Outcome: cca$status, Exposure: cca$ovab
-----
              |      Cases | Controls
-----+-----+-----
Exposed       |         65 |         11
Unexposed     |         62 |        118
-----+-----+-----

Odds ratio = 11.13, 95% CI = 5.349 - 25.16
Chi-squared = 53.754, df = 1, p-value = 0.0000
Fisher's exact test (2-sided) p-value = 0.0000

> cc(cca, status, alc2)
Case/Outcome: cca$status, Exposure: cca$alc2
-----
```

	Cases	Controls
Exposed	57	20
Unexposed	71	108

Odds ratio = 4.309, 95% CI = 2.321 - 8.26
 Chi-squared = 24.071, df = 1, p-value = 0.0000
 Fisher's exact test (2-sided) p-value = 0.0000

```
> cc(cca, status, smo2)
```

Case/Outcome: cca\$status, Exposure: cca\$smo2

	Cases	Controls
Exposed	73	58
Unexposed	55	70

Odds ratio = 1.599, 95% CI = 0.949 - 2.705
 Chi-squared = 3.064, df = 1, p-value = 0.08
 Fisher's exact test (2-sided) p-value = 0.0798

5.2.2.2 คำสั่งที่ใช้ในการคำนวณขนาดความเสี่ยงในกรณีที่มีตัวกวน

หากความสัมพันธ์ระหว่าง outcome กับ exposure มีการกวนด้วยปัจจัยอื่นอีกหนึ่งปัจจัย การวิเคราะห์ความสัมพันธ์ระหว่าง outcome กับ exposure โดยไม่คำนึงถึงการกวนนั้นจะได้ผลไม่ถูกต้อง ดังได้กล่าวแล้วในเรื่องก่อนหน้านี้ ใน library **Epid** มีคำสั่ง mhor (Mantel-Haenzel odds ratio) เพื่อคำนวณหาความสัมพันธ์ระหว่าง outcome และ exposure โดยปรับอิทธิพลของการกวนเสียก่อน ดังมีรูปแบบการออกคำสั่งดังนี้

```
mhor(data, case, expose, group, frame=get.ec.option("frame"))
```

data	กรอบข้อมูล (อาจไม่ระบุก็ได้)
case, expose	เวกเตอร์ case และ expose ทั้งสองต้องมีสองระดับ นั่นคือ case หรือ control ในกรณีของ case และ exposed หรือ unexposed ในกรณีของ expose
group	เวกเตอร์ที่ใช้แบ่งชั้น ซึ่งต้องเป็นตัวแปรชนิดจำนวนนับ ซึ่งอาจมีสองระดับขึ้นหรือมากกว่าก็ได้
frame	วาดตารางมีกรอบ ค่าปกติเป็น TRUE

ตัวอย่างที่ 5.3 จากข้อมูลวิจัยมะเร็ง cholangiocarcinoma (CCA) ในจังหวัดนครพนม จะเห็นว่าทั้งการมี

antibody ต่อ OV และการดื่ม alcohol มีความสัมพันธ์กับการเป็นมะเร็ง CCA ลองหาความสัมพันธ์ระหว่าง antibody ต่อ OV กับการดื่ม alcohol (ใช้ตัวแปร alc2) และ หา OR ของการดื่ม alcohol เมื่อปรับอิทธิพลการกวนจากการติดเชื้อ OV แล้ว และคิดว่าจะตีความ OR ที่ได้จากการใช้คำสั่ง mhor อย่างไร

```
> xtab(cca, ovab, alc2, test=T)
```

```
row: ovab, column: alc2
```

	0	1	Total
negative	131	45	176
positive	43	31	74
Total	174	76	250

Chi-squared = 5.812, df = 1, p-value = 0.0159

Fisher's exact test (2-sided) p-value = 0.0155

```
> mhor(cca, status, alc2, ovab)
```

Stratified analysis by group

	Odds ratio	Lower Lim.	Upper Lim.	p-value
group.negative	4.279	1.977	9.313	0.000
group.positive	3.838	0.700	38.695	0.106
M-H combined	4.187	2.172	8.071	0.000

M-H Chi-squared = 19.752, df = 1, p-value = 0.0000

Homogeneity test: Chi-squared = 0.015, df = 1, p-value = 0.9032

5.2.2.3 คำสั่งที่ใช้ในการคำนวณช่วงความเชื่อมั่นร้อยละ 95

ในการคำนวณช่วงความเชื่อมั่นร้อยละ 95 สามารถทำได้เป็นขั้นตอนโดยใช้คำสั่งพื้นฐานใน R ก็ได้

$$\text{SE (for proportion)} = \sqrt{\frac{p(1-p)}{n}}$$

$$95\% \text{ CI (for proportion)} = p \pm Z_{\alpha/2} \text{SE}$$

$$\text{SE (for mean)} = \frac{\text{SD}}{\sqrt{n}}$$

$$95\% \text{ CI (for mean)} = \bar{X} \pm Z_{\alpha/2} \text{SE}$$

แต่เนื่องจากจะเป็นการยุ่งยากเกินไป จึงได้สร้างคำสั่งสำเร็จรูปไว้ให้แล้วใน library **ci** คือ คำสั่ง **ci** ซึ่งมีรูปแบบการใช้งานดังนี้

กรณีที่ 1 ส่งผ่านตัวเลขให้กับคำสั่ง **ci**

```
ci(n, m, sd, level=0.95, frame=get.ec.option("frame"))
```

n	จำนวนตัวอย่างในการศึกษา
m	จำนวนที่เกิดเหตุการณ์ หรือค่าเฉลี่ย
sd	ค่าเบี่ยงเบนมาตรฐาน ถ้า m เป็นจำนวนการเกิดเหตุการณ์ ไม่ต้องใส่ค่า sd
level	ระดับของค่าช่วงความเชื่อมั่น
frame	วาดตารางมีกรอบ ค่าปกติเป็น TRUE

ตัวอย่างที่ 5.4 จากข้อมูลวิชัยมะเร็ง cholangiocarcinoma ในจังหวัดนครพนม ในกลุ่มผู้ป่วย มีผู้ดื่มเหล้าหรือเคยดื่มเหล้าประจำอยู่ 57 คน จากผู้ป่วยทั้งหมด 128 คน และมีผู้ดื่มเหล้าหรือเคยดื่มเหล้าประจำอยู่ 20 คน จากกลุ่มควบคุมทั้งหมด 128 คน จงหาสัดส่วนของการดื่มเหล้าและช่วงความเชื่อมั่นร้อยละ 95 ในทั้งสองกลุ่มนี้

```
> ci(128,57)
```

```
-----
| Obs. | Prob. | S.E. | Lower | Upper
--+-+--+-+--+-+--+-+--+-+--+-+--+-+--+-+--
| 128 | 0.445 | 0.044 | 0.359 | 0.531
-----
```

```
> ci(128,20)
```

```
-----
| Obs. | Prob. | S.E. | Lower | Upper
---+---+---+---+---
| 128 | 0.156 | 0.032 | 0.094 | 0.219
-----
```

ตัวอย่างที่ 5.5 จากข้อมูลวิจัยมะเร็ง cholangiocarcinoma ในจังหวัดนครพนม ในกลุ่มผู้ป่วย 128 คน มีอายุเฉลี่ย 58.76 ปี ค่าเบี่ยงเบนมาตรฐานเป็น 12.19 ปี และในกลุ่มควบคุม 128 คน มีอายุเฉลี่ย 59.22 ปี ค่าเบี่ยงเบนมาตรฐานเป็น 12.14 ปี จงหาช่วงความเชื่อมั่นร้อยละ 95 ของอายุเฉลี่ยในกลุ่มทั้งสองนี้

```
> ci(128,58.76,12.19)
```

```
-----
| Obs. | Mean | S.E. | Lower | Upper
---+---+---+---+---
| 128 | 58.76 | 1.077 | 56.628 | 60.892
-----
```

```
> ci(128,59.22,12.14)
```

```
-----
| Obs. | Mean | S.E. | Lower | Upper
---+---+---+---+---
| 128 | 59.22 | 1.073 | 57.097 | 61.343
-----
```

กรณีที่ 2 ส่งผ่านเวกเตอร์ให้กับคำสั่ง *ci*

```
ci(data, x, subset, group, level=0.95, frame=get.ec.option
    ("frame"))
```

data	กรอบข้อมูล (อาจไม่ระบุก็ได้)
x	เวกเตอร์ที่ต้องการหาช่วงความเชื่อมั่น
subset	เงื่อนไขระบุ subset ของข้อมูล
group	เวกเตอร์ระบุการแยกกลุ่มของข้อมูล
level	ระดับของค่าช่วงความเชื่อมั่น
frame	วาดตารางมีกรอบ ค่าปกติเป็น TRUE

ตัวอย่างที่ 5.6 จากข้อมูลวิจัยมะเร็ง cholangiocarcinoma ในจังหวัดนครพนม (data(cca)) จงหาสัดส่วน

ของการดื่มเหล้าและช่วงความเชื่อมั่นร้อยละ 95 ในกลุ่มผู้ป่วยและกลุ่มควบคุม

```
> data(cca)
> ci(cca, alcohol)

-----
      | Obs. | Prop. | S.E. | Lower | Upper
-----+-----+-----+-----+-----+-----
never   |   79 | 0.309 | 0.001 | 0.254 | 0.367
occasional |  100 | 0.391 | 0.001 | 0.332 | 0.449
ex-drinker |   24 | 0.094 |    0 | 0.059 | 0.133
drinker  |   53 | 0.207 | 0.001 | 0.16  | 0.258
-----

> cca <- transform(cca,alc2=ifelse(alcohol=="never" |
alcohol=="occasional",0,1))
> attach(cca)
> ci(cca,alc2,status=="case")
alc2
-----
      | Obs. | Mean | S.E. | Lower | Upper
-----+-----+-----+-----+-----
      |  128 | 0.445 | 0.044 | 0.359 | 0.531
-----

> ci(cca,alc2,status=="control")
alc2
-----
      | Obs. | Mean | S.E. | Lower | Upper
-----+-----+-----+-----+-----
      |  128 | 0.156 | 0.032 | 0.093 | 0.219
-----

> detach(cca)
```

ตัวอย่างที่ 5.7 จากข้อมูลวิจัยมะเร็ง cholangiocarcinoma ในจังหวัดนครพนม (data(cca)) จงหาช่วงความเชื่อมั่นร้อยละ 95 ของอายุเฉลี่ยในกลุ่มผู้ป่วยและกลุ่มควบคุม

```
> data(cca)
> ci(cca$age)

-----
      | Obs. | Mean | S.E. | Lower | Upper
-----+-----+-----+-----+-----
      |  262 | 58.989 | 0.75 | 57.519 | 60.459
-----

> ci(cca,age,status=="case")
age
```

	Obs.	Mean	S.E.	Lower	Upper
	131	59.221	1.061	57.141	61.301

```
> ci(cca, age, status=="control")
```

```
age
```

	Obs.	Mean	S.E.	Lower	Upper
	131	58.756	1.065	56.669	60.843

5.2.3 การใช้โมเดลทางสถิติเพื่อวิเคราะห์ปัจจัยเสี่ยง

การใช้คำสั่ง `mhor` เพื่อหา Mantel-Haenszel odds ratio นั้น มีข้อจำกัดสำคัญตรงที่จะใส่ตัวทวนเข้าไปปรับค่า odds ratio ได้เพียงตัวเดียวเท่านั้น หากในงานวิจัยหนึ่ง มีตัวแปรที่เป็นตัวทวนของความสัมพันธ์ระหว่าง exposure กับ outcome ที่สนใจมากกว่าหนึ่งตัวแล้ว การใช้คำสั่ง `mhor` ก็จะไม่สามารถหาค่า odds ratio ที่ปรับอิทธิพลของตัวทวนมากกว่าหนึ่งตัวในคราวเดียวกันได้ เพื่อจะทำเช่นนี้จะต้องใช้โมเดลแบบ linear logistic เพื่อแก้ปัญหา ดังสูตรการคำนวณต่อไปนี้

ถ้า p เป็นโอกาส (หรือความเสี่ยง) ที่จะเป็นโรค และ y เป็น logit function ของ p

$$y = \text{logit } p = \ln\left(\frac{p}{(1-p)}\right)$$

ซึ่งอาจเขียนในอีกรูปแบบหนึ่งได้ว่า

$$p = \frac{e^y}{1+e^y}$$

เนื่องจาก $\frac{p}{1-p}$ คือ odds ของการเกิดโรค ดังนั้น $\text{logit } p$ จึงเป็น $\log \text{ odds}$

รูปแบบสมการทั่วไปของ logit เป็น

$$\text{logit } p_i(x) = \alpha_i + \beta_k x_k$$

และเราจะได้ว่า e^{β_k} จะเท่ากับ odds ratio ของตัวแปร x_k นั้น

แม้ว่าในทางทฤษฎีอาจดูยุ่งยาก แต่ในทางปฏิบัติแล้ว การคำนวณเพื่อแก้ logistic model นั้นกลับไม่ยุ่งยากแต่อย่างใด ลองพิจารณาจากตัวอย่าง ANC ข้างต้น คำสั่งใน library `ice` ของโปรแกรม R

ในการเรียกใช้ linear logistic model คือคำสั่ง `logistic` ซึ่งมีการส่งสมการทางสถิติในรูปของ outcome ~ predictor(s) โดยที่ตัวแปร outcome จะต้องมามีค่า 0 หรือ 1 เท่านั้น หากมีตัวแปร predictor มากกว่า 1 ตัว จะใช้เครื่องหมาย + คั่นระหว่างตัวแปร ดังตัวอย่าง model3

หมายเหตุ คำสั่ง `logistic()` เป็นคำสั่งที่พัฒนาขึ้นในหน่วยระบาดวิทยา คำสั่งนี้เรียกใช้คำสั่งพื้นฐาน `glm()` โดยกำหนด family เป็น binomial และแสดงผลเป็น odds ratio แทนที่จะเป็น coefficient ตามปกติในการแสดงผลของคำสั่ง `glm()`

5.2.3.1 คำสั่ง `logistic()` และคำสั่ง `logit()`

```
logistic(formula, data, subset, frame=get.ec.option("frame"))
logis(formula, data, subset, frame=get.ec.option("frame"))
```

formula	โมเดลสถิติที่ต้องการหาค่า ในรูปของ y~x. y ต้องเป็นตัวแปรที่มีค่าได้เพียงสองค่า คือ 0 หรือ 1
data	ตัวเลือกรอบข้อมูลที่มีตัวแปรในโมเดลอยู่ หากตัวแปรเหล่านั้นอยู่ในกรอบข้อมูลเดียวกันทั้งหมด
subset	ตัวเลือกระบุกลุ่มย่อยของข้อมูลที่จะใช้ในการ fit
frame	วาดตารางมีกรอบ ค่าปกติเป็น TRUE

โดยปกติแล้ว หากใช้คำสั่ง `logit()` จะแสดงผลลัพธ์ออกมาเป็น coefficient ไม่ได้แสดงเป็น odds ratio ซึ่งค่า exponential ของ coefficient นี้ก็เท่ากับ odds ratio นั้นเอง แต่ในทางระบาดวิทยาแล้ว ผู้ใช้มักต้องการค่า odds ratio มากกว่า ดังนั้น คำสั่ง `logistic()` จึงมีความหมายเช่นเดียวกันคำสั่ง `logit()` ที่ระบุตัวเลือก `or=TRUE` นั้นเอง

การออกคำสั่ง `logistic()` หรือ `logit()` นั้น อาจทำได้โดยส่งค่าให้วัตถุในคลาส `glm` หรือไม่ก็ได้ แต่ถ้าต้องการทำการทดสอบ likelihood ratio หรือต้องการนำไปใช้งานอื่นๆ ต่อ ก็ควรนำค่าที่ได้ไปเก็บไว้ในวัตถุ เช่น

```
> data(ancdata)
> logistic(death~anc, data=ancdata)
```

	Odds ratio	lower lim.	upper lim.	Pr(> z)
ancold	1.9485	1.1288	3.3636	0.0166

Log-likelihood = -220.82183, No. of observations = 755
AIC value = 445.6437

```
> model1 <- logistic(death~anc,data=ancdata)
```

	Odds ratio	lower lim.	upper lim.	Pr(> z)
ancold	1.9485	1.1288	3.3636	0.0166

Log-likelihood = -220.82183, No. of observations = 755
AIC value = 445.6437

สังเกตว่าหากไม่ส่งผลลัพธ์ออกไปเก็บในวัตถุใด คำสั่งนี้จะสรุปโมเดลที่เรียกคำสั่ง glm มาทำงานในตอนท้ายให้ดูอีกครั้ง และให้ค่า coefficient ทั้งส่วนของ intercept และ ของตัวแปรให้ดูด้วย แต่หากส่งผลลัพธ์ไปเก็บไว้ในวัตถุแล้ว คำสั่งจะแสดงเพียงส่วนของ odds ratio, 95% confidence interval และค่า p-value แล้วสรุปค่า log-likelihood และ Akaike Information Criterion (AIC) ให้ คำ AIC มีความสัมพันธ์กับค่า log-likelihood คือ $AIC = -2 * \text{log-likelihood} + k * \text{npar}$ โดยที่ $k = 2$ และ npar คือจำนวน parameter ในโมเดล

แม้ว่า AIC จะมีวิธีการคำนวณที่ซับซ้อน แต่วิธีใช้กลับไม่ยุ่งยาก ลองเปรียบเทียบค่า AIC ของ model1 และ model2 ข้างล่างนี้

```
> model1 <- logistic(death~anc,data=ancdata)
```

	Odds ratio	lower lim.	upper lim.	Pr(> z)
ancold	1.9485	1.1288	3.3636	0.0166

Log-likelihood = -220.82183, No. of observations = 755
AIC value = 445.6437

```
> model2 <- logistic(death~clinic,data=ancdata)
```

	Odds ratio	lower lim.	upper lim.	Pr(> z)
clinicB	2.8932	1.7309	4.8361	1e-04

Log-likelihood = -215.57206, No. of observations = 755
AIC value = 435.1441

model2 ให้ค่า AIC ต่ำกว่า model1 และจะเห็นด้วยว่า model2 ให้ค่า p-value ที่ต่ำกว่า model1 โดยทั่วไปแล้ว โมเดลที่อธิบายความสัมพันธ์ระหว่าง outcome กับ exposure ในชุดข้อมูลได้ดีกว่า จะให้ค่า AIC และค่า p-value ที่ต่ำกว่า โดยไม่ขึ้นกับค่า odds ratio แต่อย่างใด

จากโมเดลทั้งสอง เราพบว่า death สัมพันธ์กับทั้ง anc และ clinic (อย่าลืมว่าวัตถุประสงค์หลักของการวิจัยคือการหาความสัมพันธ์ระหว่าง death กับ anc) แต่เมื่อเรารู้อยู่แล้วว่า anc มีความสัมพันธ์กับ clinic อยู่ตามที่กล่าวไว้ในบทก่อน ค่า odds ratio ของ anc ในสมการ death ~ anc จึงยังไม่ได้คำนึงถึงอิทธิพลของ clinic แต่อย่างใด หากต้องการใส่อิทธิพลของ clinic เข้าไปในสมการด้วย เพื่อปรับให้ odds ratio ของความสัมพันธ์ระหว่าง death กับ anc มีค่าที่ถูกต้อง (ปรับอิทธิพลของ clinic แล้ว) เราจะระบุ formula ดังนี้

```
death ~ anc + clinic
```

และได้ผลดัง model3 ข้างล่าง

```
> model1 <- logistic(death~anc,data=ancdata)
```

	Odds ratio	lower lim.	upper lim.	Pr(> z)
ancold	1.9485	1.1288	3.3636	0.0166

Log-likelihood = -220.82183, No. of observations = 755
AIC value = 445.6437

```
> model2 <- logistic(death~clinic,data=ancdata)
```

	Odds ratio	lower lim.	upper lim.	Pr(> z)
clinicB	2.8932	1.7309	4.8361	1e-04

Log-likelihood = -215.57206, No. of observations = 755
AIC value = 435.1441

```
> model3 <- logistic(death~anc+clinic,data=ancdata)
```

	Odds ratio	lower lim.	upper lim.	Pr(> z)
ancold	1.1622	0.6085	2.2198	0.6489
clinicB	2.6812	1.4634	4.9124	0.0014

Log-likelihood = -215.4681, No. of observations = 755
AIC value = 436.9362

ตัวอย่างที่ 5.8 ศึกษาผลลัพธ์ของ model3 โดยเปรียบเทียบค่า p-value ของ anc ระหว่าง model1 และ model3 และค่า p-value ของ clinic ระหว่าง model2 และ model3 และเปรียบเทียบค่า AIC ของโมเดลทั้งสาม แล้วตอบว่าโมเดลไหนอธิบายความสัมพันธ์ระหว่าง death กับ anc ได้ดีที่สุด

และนอกจากนี้ หากต้องการดู interaction ระหว่างตัวแปรก็สามารถทำได้โดยใช้เครื่องหมาย * ใส่ระหว่างตัวแปรคู่ที่ต้องการจะดู interaction นั้น ดังตัวอย่าง

```
> model4 <- logistic(death~anc*clinic,data=ancdata)
```

	Odds ratio	lower lim.	upper lim.	Pr(> z)
ancold	1.2486	0.5773	2.7005	0.5727
clinicB	3.1848	0.9835	10.3125	0.0533
ancold:clinicB	0.7948	0.2036	3.1029	0.7410

Log-likelihood = -215.41464, No. of observations = 755
AIC value = 438.8293

เมื่อใส่เครื่องหมาย * ระหว่าง anc กับ clinic ผลลัพธ์ที่ได้จะแสดง odds ratio ให้ 3 บรรทัด คือ

```
anc
clinic
anc:clinic
```

นั่นคือ

anc * clinic = anc + clinic + anc:clinic

ซึ่งลองดูผลของ model5 ต่อไปนี้

```
> model5 <- logistic(death~anc+clinic+anc:clinic,data=ancdata)
```

	Odds ratio	lower lim.	upper lim.	Pr(> z)
ancold	1.2486	0.5773	2.7005	0.5727

clinicB		3.1848		0.9835		10.3125		0.0533
ancold:clinicB		0.7948		0.2036		3.1029		0.7410

Log-likelihood = -215.41464, No. of observations = 755
AIC value = 438.8293

จะเห็นว่าได้ผลเหมือนกันนั่นเอง

สรุปว่าสัญลักษณ์ที่แสดง interaction ระหว่างตัวแปรสองตัวคือเครื่องหมาย : นั่นเอง แต่เราไม่อาจเขียนสมการที่ระบุความสัมพันธ์ระหว่าง Y กับ A:B ลอยๆ โดยที่ Y ไม่มีความสัมพันธ์กับ A และ B ได้ ดังนั้น ในการเขียน formula จึงใช้เครื่องหมาย * โดยที่ R จะตีความ $A*B = A + B + A:B$ นั่นเอง แต่การตีความ interaction ทางสถิตินั้นค่อนข้างจะซับซ้อน จึงจะไม่กล่าวในที่นี้ เพียงแต่จะแสดงให้เห็นว่ากระบวนการหาความสัมพันธ์นั้น นอกจากจะต้องปรับค่าความสัมพันธ์ระหว่าง exposure ที่สนใจกับ outcome ด้วยตัวกวนแล้ว ยังอาจต้องดู interaction ระหว่างตัวแปรในสมการด้วย

5.2.3.2 คำสั่ง `lr.test()`

```
lr.test(model1, model2)
```

model1, model2 วัตถุในคลาส glm ที่ต้องการทดสอบ likelihood ratio

ดังที่กล่าวแล้วข้างต้นว่าเราอาจบอกได้ว่าโมเดลทางสถิติไหนเข้าได้ดีกับชุดข้อมูลมากกว่ากัน โดยดูจากค่า AIC แต่ยังมีอีกวิธีหนึ่งในการทดสอบคือ การทดสอบ likelihood ratio ซึ่งใน library **ice** ใช้คำสั่ง `lr.test()`

หากต้องการทดสอบว่า model1 หรือ model3 อธิบายชุดข้อมูลได้ดีกว่ากัน ก็ออกคำสั่งง่ายๆ ดังนี้

```
> model1 <- logistic(death~anc,data=ancdata)
```

	Odds ratio	lower lim.	upper lim.	Pr(> z)
ancold	1.9485	1.1288	3.3636	0.0166

```
Log-likelihood = -220.82183, No. of observations = 755
AIC value = 445.6437
```

```
> model3 <- logistic(death~anc+clinic,data=ancdata)
```

	Odds ratio	lower lim.	upper lim.	Pr(> z)
ancold	1.1622	0.6085	2.2198	0.6489
clinicB	2.6812	1.4634	4.9124	0.0014

```
Log-likelihood = -215.4681, No. of observations = 755
AIC value = 436.9362
```

```
> lr.test(model1,model3)
```

```
Likelihood ratio test: Chi-squared df(1) = 10.7075, p-value = 0.0011
```

เราได้ว่า p-value ของ chi-squared ของการทดสอบ likelihood ratio มีค่า 0.0011 นั่นคือน้อยกว่า 0.05 ซึ่งตีความว่าโมเดลทั้งสอง มีความแตกต่างกันอย่างมีนัยสำคัญทางสถิติ ส่วนการจะบอกว่าโมเดลไหนดีกว่ากันนั้น ก็ดูจากโมเดลที่ให้ค่า log-likelihood สูงกว่า (AIC ต่ำกว่า) กัน สังเกตว่าค่า log-likelihood ratio มีค่าติดลบ ค่าที่ดีกว่าคือค่าที่ติดลบน้อยกว่า หรือมีค่า log-likelihood สูงกว่านั่นเอง

ในชุดข้อมูลที่มีตัวแปรหลายตัวที่ต้องการหาความสัมพันธ์กับ outcome ตัวเดียว ดังเช่นในชุดข้อมูล cca นั้น เราต้องการหาความเสี่ยงที่จะเกิด cholangiocarcinoma หาก ovab positive, คีโมเหลว, สูบบุหรี่, GSTM1 และอื่นๆ (ตัดตัวแปรมาไว้ในที่นี้เพียงแค่นี้) ในทางปฏิบัติเราอาจมีกระบวนการหาโมเดลที่อธิบายข้อมูลได้ดีที่สุด โดยเริ่มจากตัวแปรสำคัญที่สุดที่เป็นวัตถุประสงค์หลักของงานวิจัยก่อนแล้วจึงค่อยๆ เพิ่มตัวแปรอื่นเข้าไปในโมเดลทีละตัว (forward inclusion) หรือใส่เข้าไปให้มากที่สุดแล้วตัดออกทีละตัว (แต่ต้องเก็บตัวแปรหลักของการวิจัยไว้เสมอ – backward elimination) ดังนี้

```
cca <- fread("cca.tab")
model1 <- logistic(status~ovab,data=cca)
model2 <- logistic(status~ovab+alcohol,data=cca)
model3 <- logistic(status~ovab+alcohol+smoke,data=cca)
model4 <- logistic(status~ovab+alcohol+smoke+gstml1,data=cca)
```

โดยที่แต่ละครั้งที่เพิ่มตัวแปรเข้าไป ก็ตรวจสอบว่าการเพิ่มตัวแปรนั้นเข้าไปในโมเดล ทำให้ AIC เพิ่มขึ้นหรือลดลง หาก AIC ลดลง ก็เพิ่มตัวแปรตัวต่อไปเข้าไปอีก แต่หาก AIC เพิ่มขึ้น เราจะพบว่าค่า p-value ของตัวแปรนั้นก็จะสูงกว่า 0.05 ด้วย เราก็อาจตัดตัวแปรนั้นออกจากโมเดลไปเลยได้ แต่หากว่าค่า AIC ลดลง แต่ค่า p-value ของตัวแปรนั้นสูงกว่า 0.05 มักจะหมายความว่า การใส่ตัวแปรนั้นไว้จะอธิบายชุดข้อมูลได้ดีกว่าเดิม แต่อาจเป็นไปได้ว่าขนาดตัวอย่างไม่พอที่จะทำให้ค่า p-value ของตัวแปรนั้นมีนัยสำคัญได้ เราก็จะเป็นต้องเก็บตัวแปรนั้นไว้ในโมเดล

ส่วนการเริ่มต้นจาก full model แล้วตัดตัวแปรออกทีละตัวเป็นดังนี้

```
cca <- fread("cca.tab")
model1 <- logistic(status~ovab+alcohol+smoke+gstml,data=cca)
model2 <- logistic(status~ovab+alcohol+gstml,data=cca)
model3 <- logistic(status~ovab+alcohol,data=cca)
model4 <- logistic(status~ovab,data=cca)
```

ด้วยวิธีนี้ การจะตัดตัวแปรใดออกก่อน ให้เลือกเอาตัวแปรที่ให้ค่า p-value สูงสุดออกก่อน (ยกเว้นตัวแปรนั้นเป็นตัวแปรหลักของการวิจัยให้เก็บไว้ห้ามตัดออก) และเราจะไม่ตัดตัวแปรนั้นออก เมื่อพบว่าการตัดออกทำให้ค่า AIC ของ regression สูงขึ้น แม้ว่าค่า p-value ของตัวแปรตัวนั้นจะมากกว่า

0.05 ก็ตาม

5.3 แบบฝึกหัด

1. ใช้ข้อมูล cca.tab เพื่อหาความสัมพันธ์ของตัวแปรต่างๆ กับการเป็น cholangiocarcinoma ด้วยวิธี forward inclusion และ backward elimination ลองใช้ข้อมูล cca.tab เพื่อหาความสัมพันธ์ของตัวแปรต่างๆ กับการเป็น cholangiocarcinoma ด้วยวิธี forward inclusion และ backward elimination
2. ใช้ข้อมูล cca9.tab ซึ่งเป็นข้อมูลที่ไม่ได้ label และมีรหัส 9 เป็นรหัสสำหรับค่า missing ติดอยู่ แทนที่จะใช้ cca.tab ในการคำนวณ แล้วดูว่าผลที่ต่างกันเกิดจากอะไร และจะแก้ปัญหานั้นได้อย่างไร (ในการทำงานจริง เรามักได้ชุดข้อมูลในลักษณะนี้ เราต้องเปลี่ยนรหัส 9 ให้เป็น NA แล้ว label ข้อมูลรหัสต่างๆ ก่อนจะวิเคราะห์ข้อมูล)

บทที่ 6

สถิติที่ใช้ในการวิเคราะห์ข้อมูลชนิดต่อเนื่อง

6.1 การทดสอบ t-test

จากข้อมูล cca ตัวแปร age เป็นตัวแปรที่มีค่าต่อเนื่อง (continuous) ไม่ได้เป็นรหัสเหมือนตัวแปรอื่น ในการทดสอบความแตกต่างของตัวแปรชนิดนี้ระหว่าง 2 กลุ่ม ต้องใช้การทดสอบเพื่อหาค่า mean ของกลุ่มเหล่านั้นแตกต่างกันหรือไม่ มีวิธีทดสอบมาตรฐานคือ t-test ซึ่งวิธีนี้มีเงื่อนไขที่ต้องดูก่อนอยู่ 2 อย่าง คือ 1. การกระจายของตัวแปรในทั้งสองกลุ่มที่จะนำมาเปรียบเทียบกับกันนั้น ต้องมีการกระจายคล้ายการกระจายแบบปกติ และ 2. ค่า variance (หรือค่า standard deviation) ของตัวแปรในทั้งสองกลุ่มที่จะนำมาเทียบ ต้องมีค่าใกล้เคียงกัน

ใน package stats มีคำสั่ง t.test ให้ใช้ทดสอบดังกล่าว แต่เนื่องจาก output ดูไม่ค่อยสวยงาม และเข้าใจยาก ใน package ice จึงได้สร้างคำสั่ง st.test ซึ่งจะทำงานเช่นเดียวกัน แต่ให้ output ง่ายกว่า คำสั่งนี้มีรูปแบบการออกคำสั่งได้ 2 รูปแบบ แต่ในที่นี้จะแนะนำเพียงรูปแบบเดียวก่อน ดังนี้

```
st.test(data, x, y=NULL, mu=0, paired=FALSE, var.equal=FALSE,
        conf.level=0.95, frame=get.ec.option("frame"))
st.test(formula, data, subset, frame=get.ec.option("frame"))
```

data ตัวเลือกกรอบข้อมูลที่มีตัวแปร 'x' และ 'y' (รูปแบบที่ 1) หรือ
กรอบข้อมูลที่มีตัวแปรที่ระบุใน 'formula' (รูปแบบที่ 2)

x เวกเตอร์ตัวเลขที่มีข้อมูลที่ต้องการทดสอบ

y เวกเตอร์ตัวเลขที่มีข้อมูลอีกชุดหนึ่ง ใช้เมื่อต้องการทดสอบ

แบบ paired (paired = TRUE)

mu ตัวเลขระบุค่าจริงของค่าเฉลี่ย (หรือความต่างของค่าเฉลี่ย ถ้า
กำลังทดสอบแบบสองตัวอย่าง)

paired ค่าตรรกะระบุว่าการทดสอบ paired t-test หรือไม่

var.equal ค่าตรรกะระบุว่าการให้คำสั่งคือเสมือนว่า variance ของ
สองกลุ่มเท่ากันหรือไม่ ถ้าระบุเป็น TRUE ค่า variance รวม

จะถูกนำมาใช้ มิฉะนั้นจะใช้ค่าประมาณ Welch (หรือ

Satterthwaite) ที่ degree of freedom นั้น

conf.level ระดับความเชื่อมั่น

formula สมการสถิติในรูป lhs ~ rhs โดยที่ lhs is เป็นตัวแปรตัวเลขที่

มีค่าข้อมูล และ rhs เป็นแฟกเตอร์ที่มีค่าสองระดับของกลุ่ม

ตัวอย่างสองกลุ่มที่ต้องการทดสอบ

subset ตัวเลือกกรอบข้อมูลที่มีตัวแปรที่ระบุใน formula

frame วาดตารางมีกรอบ ค่าปกติเป็น TRUE

ในการดูว่าการกระจายของตัวแปรในสองกลุ่มใกล้เคียงกับการกระจายแบบปกติหรือไม่ และ variance เท่ากันหรือไม่ อาจดูได้ง่ายๆ โดยใช้คำสั่ง **summarize()** โดยระบุตัวเลือก group เป็นชื่อกลุ่มที่ต้องการเทียบกัน ดังนี้

```
> data(cca)
> summ(cca, select=age, group=status, frame=TRUE)
group: status = control
```

Variable	Obs.	NA	Mean	S.D.	Min.	Median	Max.
age	131	0	58.76	12.19	31	57	87

```
group: status = case
```

Variable	Obs.	NA	Mean	S.D.	Min.	Median	Max.
age	131	0	59.22	12.14	28	58	83

จะเห็นว่า mean และ median ในทั้งสองกลุ่มมีค่าใกล้เคียงกัน min และ max ก็ใกล้เคียงกัน ทำให้พอจะคิดได้ว่า การกระจายของ age ในทั้งสองกลุ่มมีค่าใกล้เคียงกับการกระจายแบบปกติ อย่างไรก็ตาม มีการทดสอบการกระจายว่าเป็นแบบปกติหรือไม่ ซึ่งสามารถดูค่า p-value ได้ทันที คือคำสั่ง shapiro.test ใน package stats ดังนี้

```
shapiro.wilk.test(data, select, subset, group)
x                    เวกเตอร์ตัวเลขที่มีค่าข้อมูล ตัวเลขต้องมีค่าระหว่าง 3 ถึง
5000 อาจมีค่าว่าง (NA) ก็ได้
```

data กรอบข้อมูล (อาจไม่ระบุก็ได้)

select เวกเตอร์ตัวเลขหรือชุดของเวกเตอร์ตัวเลขในกรอบข้อมูล

อาจเรียกตัวแปรโดยชื่อหรือเลขสคริปต์ก็ได้

subset เงื่อนไขเซตย่อย

group เวกเตอร์ระบุกลุ่มหรือชั้น

```
> attach(cca)
> shapiro.test(age[status=="control"])
```

Shapiro-Wilk normality test

```
data: age[status == "control"]
W = 0.9843, p-value = 0.1357

> shapiro.test(age[status=="case"])

Shapiro-Wilk normality test

data: age[status == "case"]
W = 0.9853, p-value = 0.1726
> detach(cca)
```

จะเห็นว่า ค่า p-value ของทั้งสอง มีค่ามากกว่า 0.05 แสดงว่าการกระจายของตัวแปร age ในกลุ่ม control และ case ไม่ต่างจากการกระจายแบบปกติ สังเกตว่าใน R นั้น เราสามารถระบุ subset ของข้อมูลที่ต้องการทำงานด้วยได้ไม่ยาก ด้วยการระบุไว้ใน [...] ดังตัวอย่างข้างต้น

ส่วนค่า variance นั้นไม่ต้องถือเป็นจริงเป็นจังมากนัก เพราะคำสั่งนี้จะถือว่า variance มีค่าไม่เท่ากันไว้ก่อน แต่หากพบว่า variance มีค่าเท่าๆ กัน การระบุให้ var.equal=TRUE จะทำให้การทดสอบมีค่าแม่นยำมากขึ้น ทีนี้ลองทดสอบ Student's t test ดังนี้

```
> st.test(age~status, data=cca)
```

ตัวอย่างที่ 6.1 ในงานวิจัยมะเร็ง cholangiocarcinoma ในจังหวัดนครพนม มีผู้ป่วยกลุ่มหนึ่งได้รับ active hexose correlated compound (AHCC) ซึ่งเป็นสารสกัดจากเห็ดชนิดหนึ่ง ซึ่งคาดว่าจะช่วยเพิ่มภูมิคุ้มกัน ทำให้ผู้ป่วยมีความสบายขึ้นได้ระยะหนึ่งก่อนเสียชีวิต ในงานวิจัยนี้ได้วัด c-reactive protein (CRP) ด้วยโดยวัดครั้งแรกก่อนให้อาหารเสริม AHCC ครั้งที่ 2 หลังให้ไปแล้ว 3 เดือน และครั้งที่ 3 หลังให้ไป 6 เดือน ดังข้อมูลใน CRP.tab จงทดสอบความแตกต่างของค่า CRP หลังให้ AHCC เมื่อเทียบกับระดับก่อนให้ AHCC ในคนเดียวกัน

หมายเหตุ ดังที่กล่าวแล้วว่าคำสั่ง st.test มีการออกคำสั่งได้ 2 รูปแบบ แบบฝึกหัดนี้จะใช้การออกคำสั่งอีกรูปแบบหนึ่ง ซึ่งไม่ได้กล่าวในที่นี้ แต่ศึกษาเองได้ใน help

6.2 การทดสอบ ANOVA

หากต้องการเปรียบเทียบทีละตัว 3 ค่าพร้อมกัน จะต้องใช้การทดสอบ analysis of variance (ANOVA) ใช้คำสั่ง anova.test ซึ่งมีรูปแบบการออกคำสั่งคือ

```
anova.test(formula, data, subset, frame=get.ec.option("frame"))
```

formula	สมการสถิติที่ต้องการทดสอบ
data	กรอบข้อมูลที่ต้องการทำงานด้วย
subset	เงื่อนไขเซตย่อย
frame	วาดตารางมีกรอบ ค่าปกติเป็น TRUE

ในข้อมูลชุด CRP จะเห็นว่ารูปแบบข้อมูลแยกการตรวจของแต่ละครั้งในคนเดียวกันเป็น 3 ตัวแปร โดยให้อยู่ใน record เดียวกัน ข้อมูลรูปแบบนี้จะใช้รูปแบบการออกคำสั่งแบบ formula ไม่ได้ หากจะทำ จำเป็นต้องเปลี่ยนรูปแบบข้อมูลเสียก่อน โดยให้มีตัวแปรหนึ่งที่ใช้บอกว่าเป็นค่าจากการตรวจครั้งใด และรวมผลการตรวจทุกครั้งไว้ในตัวแปรเดียวกัน ในที่นี้จะแสดงการรวมแบบง่ายๆ โดยใช้โปรแกรม spreadsheet ก่อน ดังนี้

เนื่องจากข้อมูลเป็น tab delimited จึงใช้โปรแกรม Excel เปิดขึ้นมาได้เลย

	A	B	C	D
1	case	crp1	crp2	crp3
2	A2	0.01	0.173	0.048
3	A3	0.274	0.138	0.068
4	A4	0.068	0.109	0.087
5	A5	0.424	2.49	1.818
6	A6	0.276	0.144	0.211
7	A8	0.129	0.108	0.024
8	A10	0.748	0.333	0.042
9	A11	0.088	0.036	0.599
10	A13	0.144	0.071	0.048
11	A15	0.298	0.247	0.198
12	A18	0.061	0.069	0.435

รูปที่ 6.1 ข้อมูล CRP ในแนวกว้าง

สร้างแฟ้มข้อมูลเปล่าขึ้นมาใหม่ แล้ว copy ชุดข้อมูล case ไป paste สามครั้งต่อกันในสดมภ์ A ตั้งชื่อว่า case

ในสดมภ์ B ใส่เลข 1 ตรงกับชุดข้อมูล case ชุดแรก ใส่ 2 ตรงกับชุดข้อมูล case ชุดที่สอง และ 3 ตรงกับชุดข้อมูล case ชุดที่ 3 ตั้งชื่อว่า time

ในสดมภ์ที่ 3 copy ข้อมูล crp1, crp2 และ crp3 มาใส่โดยเรียงต่อกันไปในสดมภ์เดียวกัน ตั้งชื่อสดมภ์ว่า crp

save แฟ้มให้เป็น tab delimited ตั้งชื่อเป็น CRP2.tab ใส่ใน R workplace ที่กำลังทำงานอยู่ (ที่เดียวกับ folder ของ CRP.tab เดิมนั่นเอง)

ได้ข้อมูลดังนี้

	A	B	C
1	case	time	crp
2	A2	1	0.01
3	A3	1	0.274
4	A4	1	0.068
5	A5	1	0.424
6	A6	1	0.276
7	A8	1	0.129
8	A10	1	0.748
9	A11	1	0.088
10	A13	1	0.144
11	A15	1	0.298
12	A18	1	0.061
13	A20	1	2.45
14	A22	1	0.938
15	A23	1	0.578
16	A24	1	0.191

รูปที่ 6.2 ข้อมูล CRP ในแนวยาว

อ่านข้อมูลเข้ามาด้วยคำสั่ง **fread()** ดังนี้

```
> crp2 <- fread("crp2.tab")
```

อีกวิธีหนึ่งในการเปลี่ยนรูปแบบข้อมูล คือการใช้คำสั่งใน **R** ทำได้โดยการอ่านแฟ้มข้อมูล **CRP.tab** เข้ามาในกรอบข้อมูล **crpdata** ดังนี้

```
> crpdata <- fread("crp.tab")
```

จากนั้นสร้างกรอบข้อมูลใหม่จากกรอบข้อมูลนี้ โดยให้กรอบข้อมูลใหม่มีตัวแปรคือ **case**, **time** และ **crp** คล้ายกับที่กล่าวมาข้างต้น ในขั้นแรก สร้างกรอบข้อมูล **crpdata2** ให้มีแค่เฉพาะตัวแปร **case** ที่มี record เหมือนในกรอบข้อมูลเดิม แต่ซ้ำชุดเดิม 3 ครั้ง ดังนี้

```
> case = rep(crpdata$case, 3)
> time = c(rep(1, 26), rep(2, 26), rep(3, 26))
> crp = c(crpdata$crp1, crpdata$crp2, crpdata$crp3)
> crpdata2 <- data.frame(case, time, crp)
> rm(case, time, crp)
```

สร้างตัวแปรชื่อ **case**, **time** และ **crp** โดยที่ตัวแปร **case** สร้างจากการทำซ้ำ (**rep()**) ตัวแปร **crpdata\$case** 3 ครั้ง

แล้วสร้างตัวแปร **time** โดยทำซ้ำเลข 1 จำนวน 26 ครั้ง (เท่าจำนวน record ของ **crpdata**) เลข 2 จำนวน 26 ครั้ง และเลข 3 จำนวน 26 ครั้ง แล้วรวมทั้งหมดเข้าด้วยกันด้วยคำสั่ง **c()**

แล้วสร้างตัวแปร `crp` โดยสร้างจากการนำเอาตัวแปร `crpdata$crp1`, `crpdata$crp2` และ `crpdata$crp3` มาเรียงต่อกัน ด้วยคำสั่ง `c()`

จากนั้นสร้าง data frame ชื่อ `crpdata2` จากการนำ (ชื่อและค่าใน) ตัวแปร `case`, `time` และ `crp` มารวมกัน

เนื่องจากในขั้นตอนที่ 4 เป็นการสร้าง data frame ใหม่ ที่มีตัวแปรที่มีชื่อ `case`, `time` และ `crp` อยู่ภายใน แต่ตัวแปรที่สร้างขึ้นแต่เดิมยังอยู่นอก data frame นี้ จึงควรลบออกด้วยคำสั่ง `rm()` เพื่อจะได้ไม่สับสนเวลาเรียกใช้งาน

หมายเหตุ ในการให้ค่าแก่ตัวแปร อาจใช้เครื่องหมาย `=` หรือ `<-` ก็ได้ ดังตัวอย่างข้างบน

สำหรับผู้ที่ไม่ชำนาญในการใช้คำสั่งของ **R** จะรู้สึก่วาวิธีแรกจะสะดวกและเข้าใจง่ายกว่ามาก แต่หากชำนาญการใช้คำสั่งของ **R** แล้ว ก็จะมีวิธีที่ง่ายกว่าวิธีหลังที่ทำได้ง่ายเช่นกัน หากต้องการบันทึกกรอบข้อมูล `crpdata2` เป็นแฟ้มข้อมูล ก็ทำได้ไม่ยาก ด้วยการใช้คำสั่ง `fwrite()` ดังนี้

```
> fwrite(crpdata2, "crpdata2.tab")
```

ตัวอย่างที่ 6.2 จากข้อมูล CRP ก่อนและหลังการให้ AHCC ในผู้ป่วย cholangiocarcinoma จงเปรียบเทียบว่าค่า CRP ทั้งสาม ครั้งต่างกันหรือไม่

หมายเหตุ หากต้องการเปรียบเทียบทั้ง 3 ครั้งพร้อมกันในคราวเดียว ให้ใช้การทดสอบ ANOVA แต่วิธีนี้จะไม่เห็นว่าการเพิ่มขึ้นหรือลดลง หรือค่าของการวัดครั้งไหนที่ต่างไปจากกลุ่มอื่น จึงควรทำ t-test คู่ทีละคู่ด้วย

การทดสอบ ANOVA นั้น ไม่มีข้อกำหนดที่เคร่งครัดว่าการกระจายของข้อมูลแต่ละกลุ่มต้องเป็นแบบปกติหรือ variance ของแต่ละกลุ่มต้องใกล้เคียงกัน แต่สำหรับการทดสอบ t-test นั้น จะเน้นเรื่องการกระจายของข้อมูลในแต่ละกลุ่มต้องคล้ายการกระจายแบบปกติเป็นอย่างมาก หากข้อมูลที่มีอยู่มีการกระจายไม่เป็นแบบปกติแล้ว มีทางออกอยู่สองวิธีใหญ่คือ

หากข้อมูลมีการกระจายแบบเบ้ซ้ายหรือขวา ลอง transform ข้อมูลด้วยการทำให้เป็น logarithm ดู อาจพบว่าการกระจายของ logarithm ของข้อมูลชุดเดิมนั้นเปลี่ยนเป็นการกระจายปกติได้ ในกรณีนี้ ก็จะสามารถใช้ t-test กับข้อมูลที่เป็น log นั้นได้ แต่การสรุปผลพึงระวังว่าการทดสอบนั้นทำกับ log ของข้อมูลเดิมนั้น ไม่ใช่ตัวข้อมูลเดิมแท้ๆ

หากการกระจายเบ้มาก จนไม่สามารถ transform ได้ หรือการกระจายมีลักษณะอื่นๆ ที่ไม่สามารถ tranform ได้ เช่น เป็นแบบ bimodal คือกราฟการกระจายมีสองยอด ก็ยังมีอีกวิธีหนึ่งคือใช้ non-

parametric test ที่คล้ายคลึงกับ t-test คือ signed rank (Wilcoxon) test หรือ rank sum (Mann-Whitney) test เป็นการนำอันดับ (rank) ของข้อมูลแทนค่าของข้อมูลนั้นในการคำนวณในลักษณะเดียวกับ t-test ใน R นั้น จะรวมวิธีการทั้งสองเป็นคำสั่งเดียวกัน คือ `wilcox.test()` ซึ่งมีรูปแบบการออกคำสั่งเหมือนคำสั่ง `t.test()` และ `st.test()` คือใช้ได้ทั้งวิธีส่งผ่าน formula หรือส่งผ่านตัวแปร หรือ คู่ของตัวแปร เช่นกัน

6.3 การวิเคราะห์การถดถอย

เป็นการวิเคราะห์สมการถดถอยที่ศึกษาระดับและทิศทางของความสัมพันธ์ระหว่างตัวแปรตามและตัวแปรอิสระตั้งแต่ 1 ตัวขึ้นไป เช่นต้องการหาปัจจัยที่ส่งผลต่อน้ำหนักแรกคลอดของเด็กทารก ปัจจัยที่มีผลต่อน้ำหนักดังกล่าว เช่น อายุครรภ์ จำนวนครั้งของการตั้งครรภ์ น้ำหนักของมารดา และภาวะโภชนาการ เป็นต้น ด้วยการใช้คำสั่ง `lm()` ซึ่งอยู่ใน library stats มีรูปแบบคำสั่ง คือ

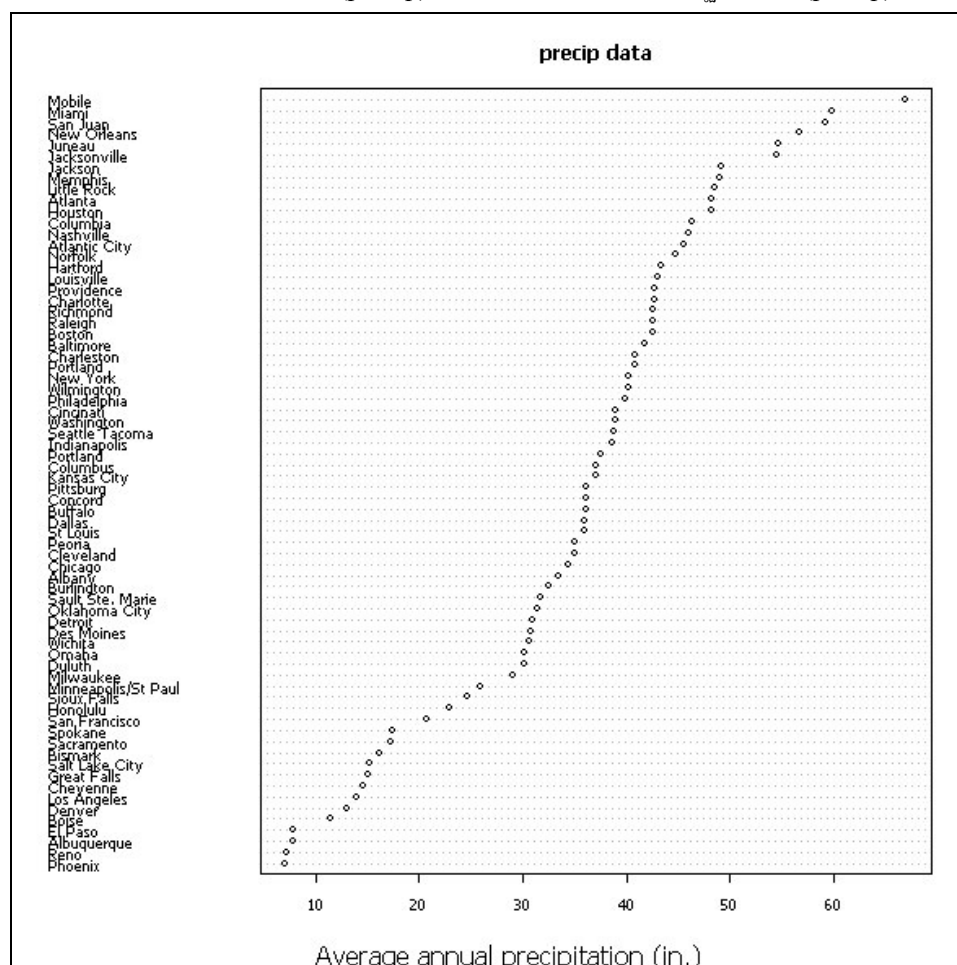
```
lm(formula, data, subset, weights, na.action,
   method = "qr", model = TRUE, x = FALSE, y = FALSE, qr = TRUE,
   singular.ok = TRUE, contrasts = NULL, offset, ...)
```

formula	สมการสถิติที่ต้องการทดสอบ ประกอบด้วยตัวแปรตาม และตัวแปรต้นอยู่ในรูปแบบ <code>yr~x1+x2</code>
data	กรอบข้อมูล
subset	การระบุการเลือกข้อมูล
weights	การถ่วงน้ำหนักในกระบวนการ fit model
na.action	ตัวเลือกในการทำงานกับข้อมูล missing โดยค่า default คือ <code>na.omit</code>
method	วิธีการที่เลือก
model, x, y, qr	ถ้ากำหนดให้เป็น TRUE ก็จะทำให้ค่าขององค์ประกอบเหล่านี้ออกมา

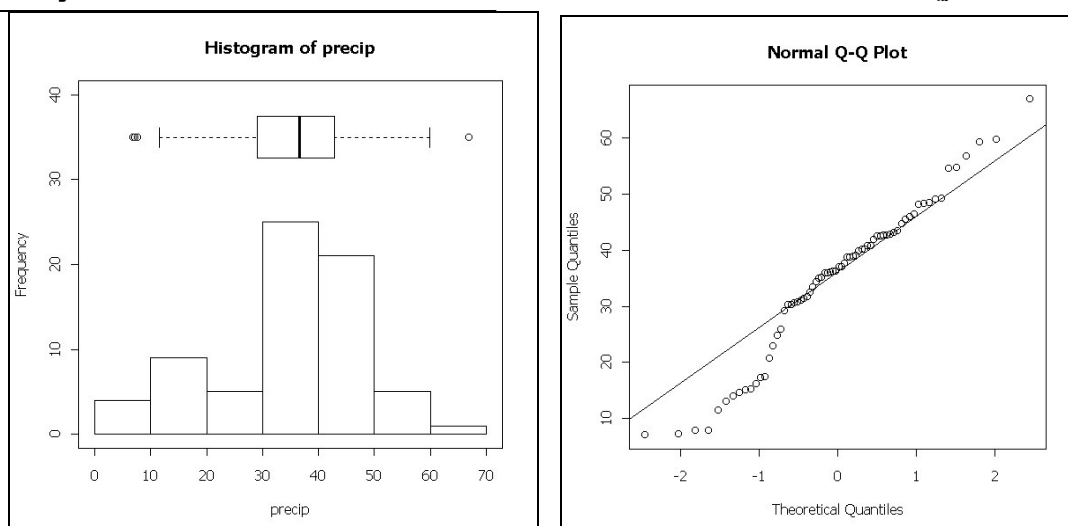
ตัวอย่างที่ 6.3 ชุดข้อมูล 'precip' บันทึกปริมาณเฉลี่ยของน้ำฝนหน่วยเป็นนิ้ว ใน 70 เมืองของประเทศสหรัฐอเมริกาและประเทศเปอโตริโก (Puerto Rico) จงสร้างกราฟเพื่อดูการแจกแจงของข้อมูลชุดนี้ และข้อมูลมีการแจกแจงปกติหรือไม่

เพื่อให้เห็นการแจกแจงของข้อมูล เราสามารถใช้คำสั่ง `plot` หรือการใช้คำสั่ง `boxplot` และ `hist` ซึ่งจะสามารถแสดงให้เห็นการแจกแจงของข้อมูล (รูปที่ 6.4) การใช้คำสั่ง `qqnorm` และ `qqline` จะช่วยให้เห็นการแจกแจงของค่าที่อยู่ใต้เส้นอันเนื่องจากการแจกแจงไม่ปกติ เพราะว่าค่า *Quantiles* จากตัวอย่างและค่า *Quantiles* จากในทฤษฎี แตกต่างกันมากดังในมุมด้านล่างของกราฟ (รูปที่ 6.5) การใช้คำสั่ง `hist()` และ `density()` เพื่อประมาณฟังก์ชันการแจกแจงที่อยู่ใต้เส้น จากรูปที่ 6.6 เส้นทึบแสดง

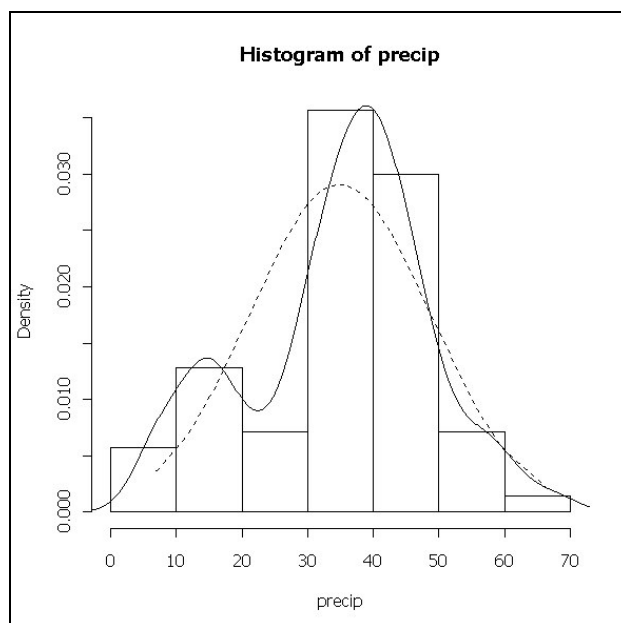
ค่าที่ประมาณจากฟังก์ชันการแจกแจง ในขณะที่เส้นประแสดงค่าที่ประมาณจากฟังก์ชันการแจกแจงที่มีการแจกแจงปกติ ด้วยค่าเฉลี่ย = $mean(precip)$ และส่วนเบี่ยงเบนมาตรฐาน = $sd(precip)$.



รูปที่ 6.3 Dotplot ของปริมาณน้ำฝนรายปีจากจำนวน 70 เมืองในประเทศสหรัฐอเมริกา



รูปที่ 6.4 ฮิสโตแกรมพร้อมด้วย boxplot รูปที่ 6.5 Q-Q plot สำหรับการประเมินการแจกแจง



รูปที่ 6.6 ฮิสโตแกรมพร้อมด้วยเส้นแสดงการแจกแจง

ตัวอย่างคำสั่ง ดังต่อไปนี้

```
> library(MASS)
> data(precip)
> dotchart(precip[order(precip)], main = "precip data", cex=0.6)
> title(sub = "Average annual precipitation (in.)")

> hist(precip, ylim=c(0,40))
> boxplot(precip, horizontal=T, at = 35, add=T, boxwex=10)

> qqnorm(precip)
> qqline(precip)

> hist(precip, freq=FALSE)
> lines(density(precip), lty=1)
> lines(sort(precip), dnorm(sort(precip), mean(precip), sd(precip)),
  lty = 2)
```

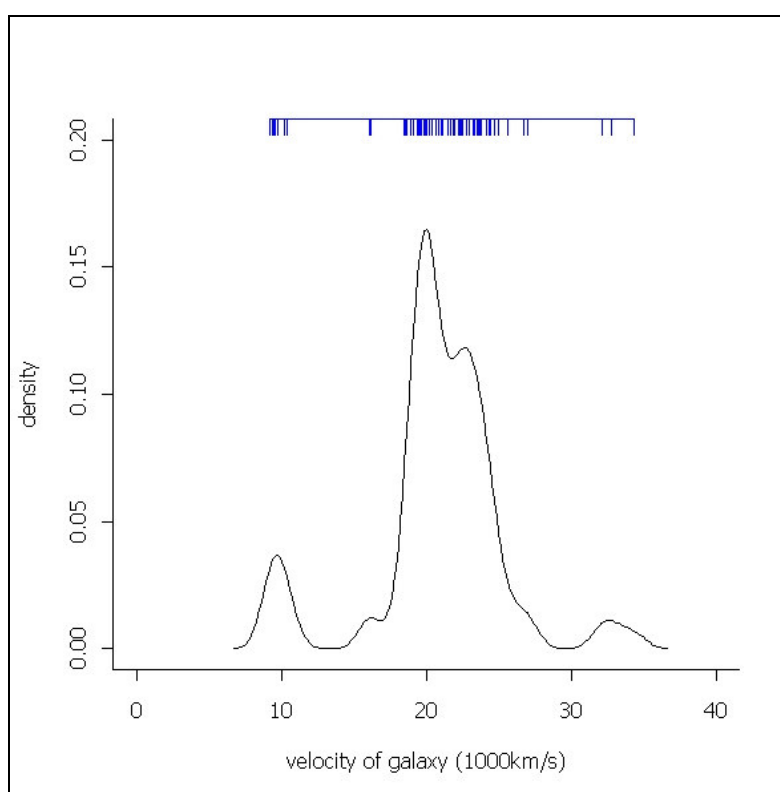
ตัวอย่างที่ 6.4 พิจารณาชุดข้อมูล 'galaxies' จาก library 'MASS' ชุดข้อมูลนี้เป็นข้อมูลเวกเตอร์ตัวเลขที่แสดงความเร็วของหมู่ดาวในหน่วยกิโลเมตรต่อวินาที จำนวน 82 กลุ่ม ยอดกราฟหลายยอดที่ได้จากการสำรวจแต่ละครั้ง จะเป็นตัวบ่งบอกว่าหมู่ดาวไม่มีการเคลื่อนที่ หรือมีการเคลื่อนที่เป็นเป็นลักษณะกลุ่มก้อนที่เกิดขึ้นในระบบจักรวาล จงพิจารณาว่ามีการแจกแจงที่ไม่ปกติเป็นจำนวนกี่กลุ่ม

รูปที่ 6.7 แสดงกราฟตัวอย่างด้วยคำสั่ง `example(galaxies)` การแจกแจงแสดงให้เห็นยอดจำนวน 3 ยอด ผลการสังเกตที่น่าสนใจคือ ค่าเฉลี่ย ค่ามัธยฐานของข้อมูลมีค่าเท่ากัน

ตัวอย่างคำสั่งดังต่อไปนี้

```
> library(MASS)
> data(galaxies)
> gal <- galaxies/1000
> plot(x = c(0, 40), y = c(0, 0.2), type = "n", bty = "l", ylab =
      "density", xlab = "velocity of galaxy (1000km/s)")
> rug(jitter(gal, amount=0.01), side=3, col="blue")
> lines(density(gal, width = 3.25, n = 200), lty = 1)

> summary(galaxies)
Min. 1st Qu. Median Mean 3rd Qu. Max.
9172 19530 20830 20830 23130 34280
```



รูปที่ 6. 7 Density plot ของข้อมูลหมู่ดาว

ตัวอย่างที่ 6.5 ชุดข้อมูล 'USArrests' ประกอบด้วยค่าสถิติของการจับกุมต่อแสนคนอันเนื่องมาจากการทำร้ายร่างกาย การฆาตกรรม และการข่มขืนในมลรัฐจำนวน 50 มลรัฐในประเทศสหรัฐอเมริกาในปี ค.ศ. 1973 รวมถึงข้อมูลเปอร์เซ็นต์จำนวนประชากรที่อาศัยอยู่ในเขตเมือง สถิติความถี่ของการถูกจับกุมจะสูงในรัฐที่มีเปอร์เซ็นต์ประชากรในเขตเมืองสูงหรือไม่

สามารถใช้คำสั่ง *pairs* เพื่อตรวจสอบความสัมพันธ์ระหว่างตัวแปร ความสัมพันธ์นี้แสดงด้วยการใช้ตัวเลือก *panel* ในคำสั่ง **pairs()** รูปที่ 6.8 แสดง scatter plot matrix ของชุดข้อมูลนี้ พบว่า จำนวนการข่มขืนและการทำร้ายร่างกายสูงขึ้นในรัฐที่มีเปอร์เซ็นต์ของประชากรในเขตเมืองสูง ในขณะที่ความถี่ของจำนวนการฆาตกรรมไม่ได้ขึ้นกับเปอร์เซ็นต์ของประชากรในเขตเมือง จำนวนการฆาตกรรมสูงขึ้นเมื่อจำนวนการทำร้ายร่างกายสูงขึ้น จากการสร้างโมเดลเส้นตรงจากข้อมูลชุดนี้ ก็จะเห็นว่าความถี่ของการจับกุมเพิ่มขึ้นในรัฐที่มีเปอร์เซ็นต์ของประชากรในเมืองสูง

ตัวอย่างคำสั่งดังต่อไปนี้

```
> data(USArrests)
> attach(USArrests)
> pairs(USArrests, panel=panel.smooth)

> Arrests <- Murder+Assault+Rape
> fm <- lm(Arrests~UrbanPop)

> summary(fm)

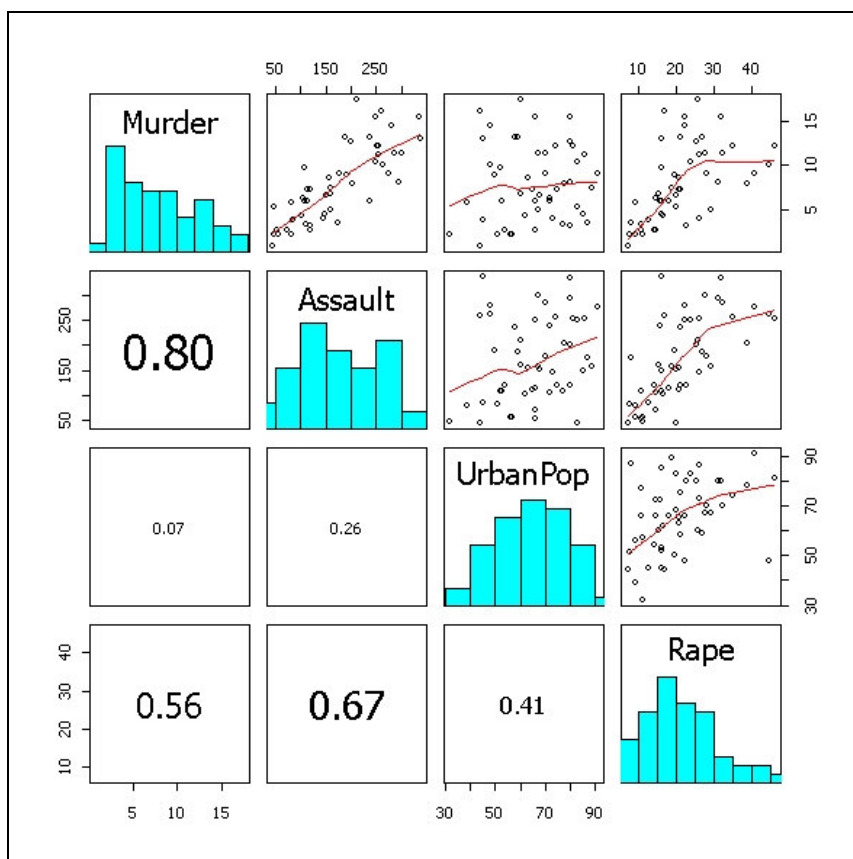
Call:
lm(formula = Arrests ~ UrbanPop)

Residuals:
    Min       1Q   Median       3Q      Max
-159.3  -67.1  -18.3   76.5  202.8

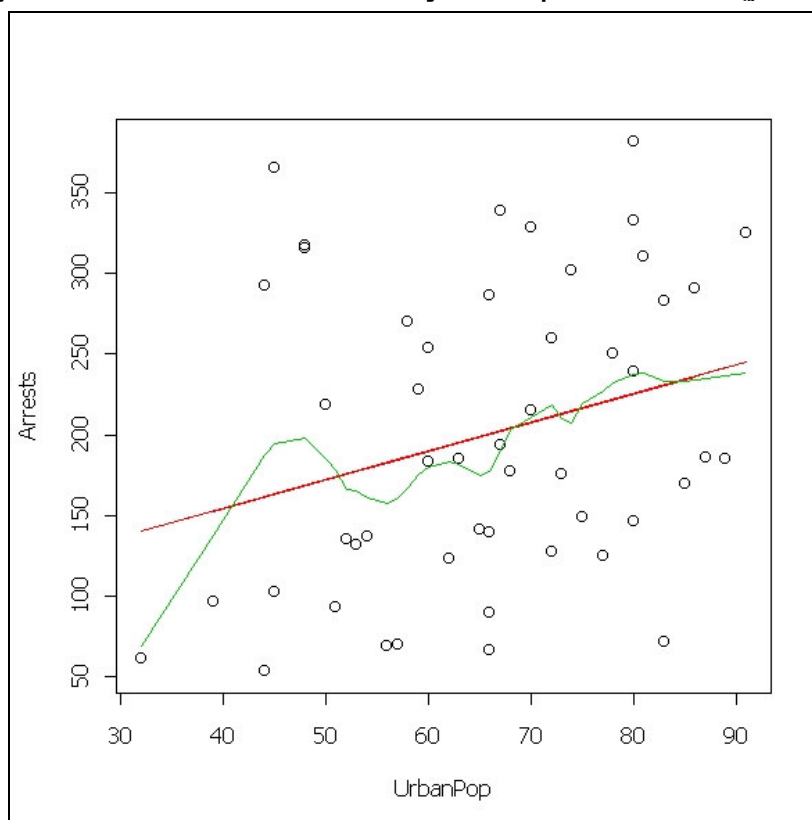
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   83.280     60.046   1.39   0.172
UrbanPop       1.778      0.895   1.99   0.053

Residual standard error: 90.7 on 48 degrees of freedom
Multiple R-Squared: 0.0759,    Adjusted R-squared: 0.0567
F-statistic: 3.94 on 1 and 48 DF, p-value: 0.0528

plot(Arrests~UrbanPop)
lines(UrbanPop, fitted(fm), col=2)
lines(lowess(UrbanPop, Arrests, f=1/3), col=3)
```



รูปที่ 6.8 Scatter plot matrix ของข้อมูลการจับกุมในประเทศสหรัฐอเมริกา



รูปที่ 6.9 Scatterplot พร้อมด้วยเส้น predict สำหรับข้อมูลการจับกุมในประเทศสหรัฐอเมริกา

ตัวอย่างที่ 6.6 ชุดข้อมูล ‘GAGurine’ ใน library ‘MASS’ ประกอบด้วยความเข้มข้นทางเคมีของ GAG ในปัสสาวะเด็กจำนวน 314 คน ช่วงอายุ 0 – 7 ปี วิเคราะห์ข้อมูล และสร้างแผนภูมิเพื่อช่วยกุมารแพทย์ ในการประเมินความเข้มข้นของค่า GAG ว่ามีการแจกแจงปกติหรือไม่

จากรูปที่ 6.10 จะพบว่าเด็กอายุน้อย มีความเข้มข้นของค่า GAG ในปัสสาวะสูง เนื่องจากการแจกแจง เป็นแบบ *exponential* โมเดล *log-linear* แสดงดังรูปที่ 6.11 จากรูปที่ 6.13 จะพบว่า *cubic model* จะเป็นโมเดลที่ดีที่สุด ซึ่งแสดงพร้อมเส้นการประมาณช่วงค่า (prediction interval)

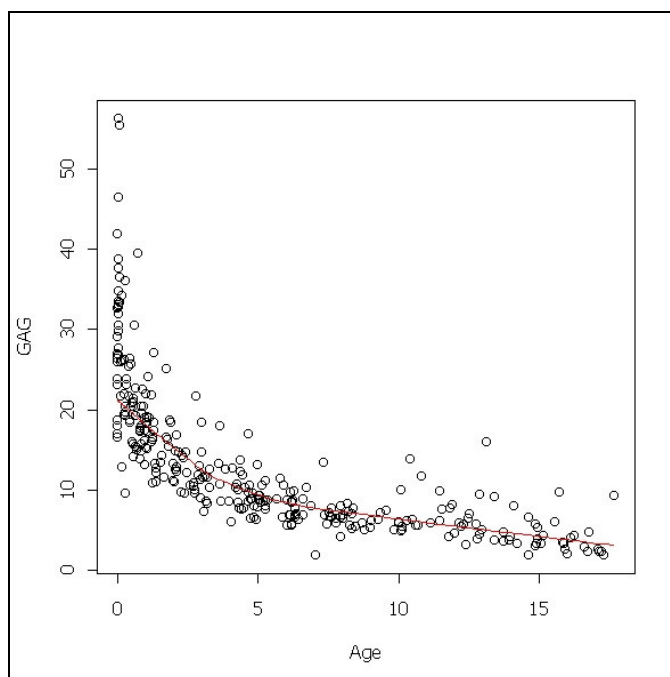
ตัวอย่างคำสั่งดังต่อไปนี้

```
> library(MASS)
> data(GAGurine)
> plot(GAGurine)
> lines(lowess(GAGurine), col=2)

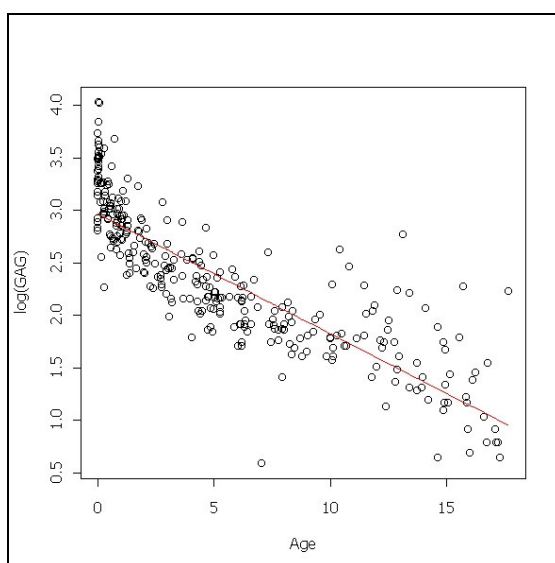
> fm1 <- lm(log(GAG)~Age, data=GAGurine)
> plot(log(GAG)~Age, data=GAGurine)
> lines(Age, fitted(fm1), col=2)
> plot(resid(fm1)~fitted(fm1))

> fm2 <- lm(log(GAG)~Age+I(Age^2)+I(Age^3), data=GAGurine)
> pr <- predict(fm2, GAGurine, interval="prediction")

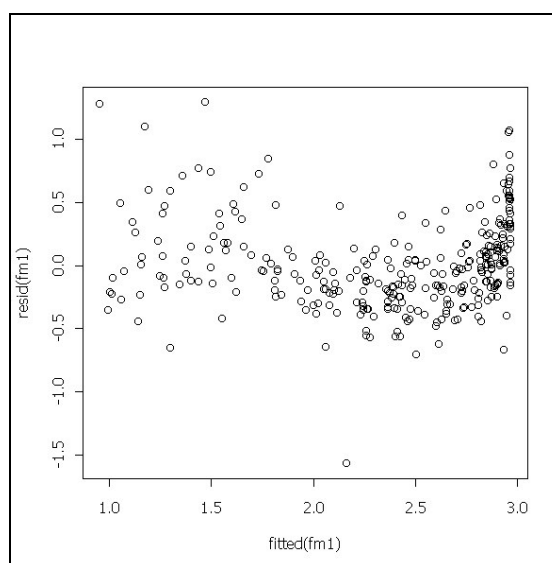
> plot(log(GAG)~Age, data=GAGurine)
> lines(Age, fitted(fm2))
> lines(Age, pr[,3], lty=2)
> lines(Age, pr[,2], lty=2)
```



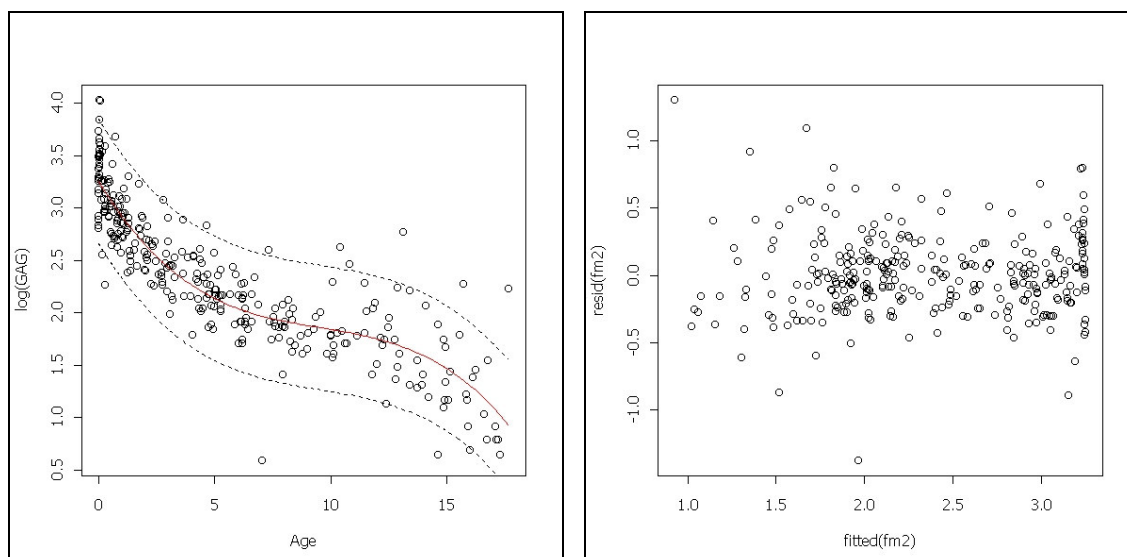
รูปที่ 6.10 Scatterplot สำหรับข้อมูล GAG



รูปที่ 6.11 การสร้างโมเดล log-linear

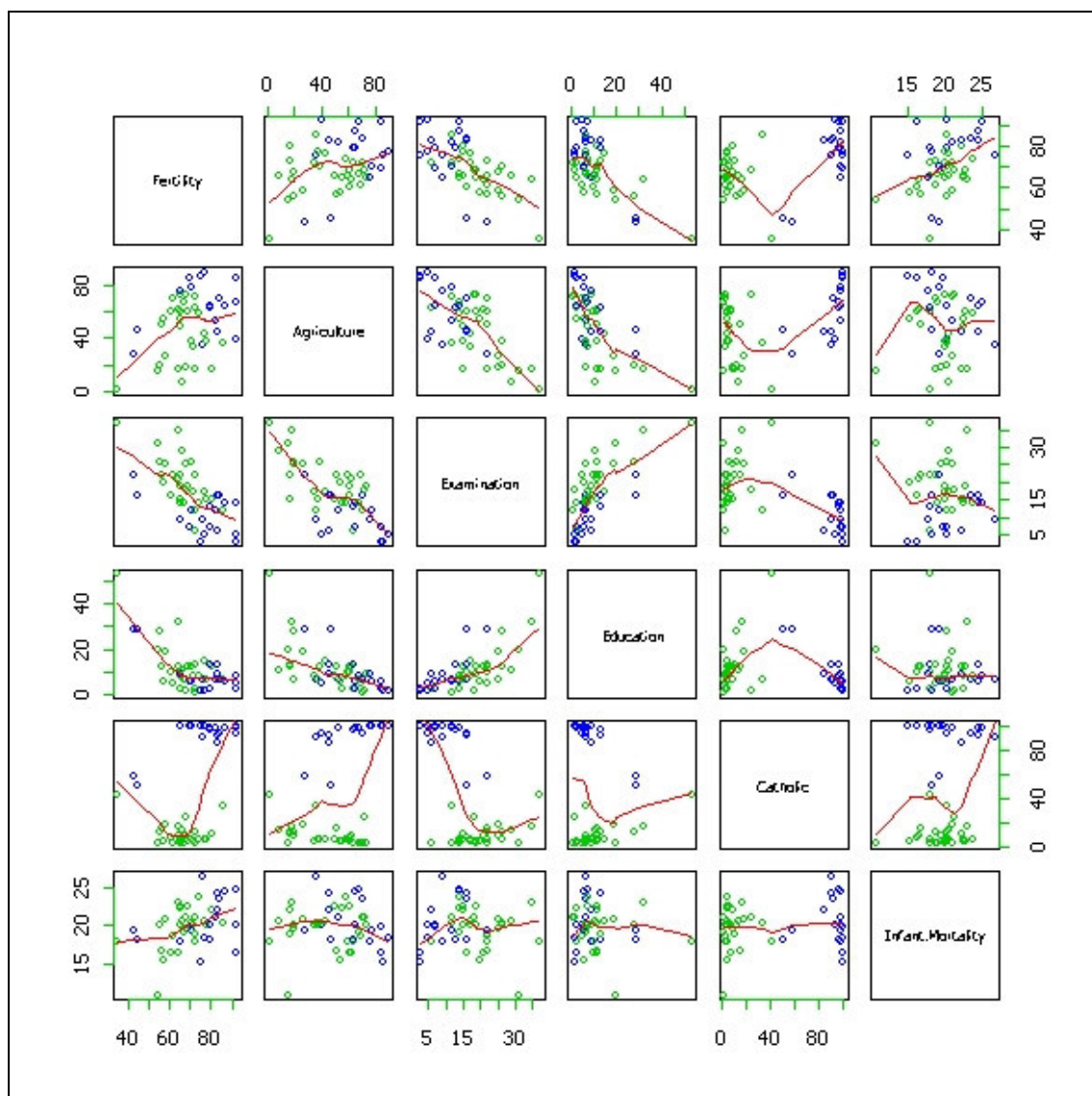


รูปที่ 6.12 Residuals vs fitted values

รูปที่ 6.13 การสร้าง โมเดล *log-linear cubic*รูปที่ 6.14 *Residuals vs fitted values*

ตัวอย่างที่ 6.7 ชุดข้อมูล 'swiss' เป็นข้อมูลการวัดภาวะการเจริญพันธุ์ และตัวชี้วัดทางเศรษฐกิจทางสังคมจำนวน 47 จังหวัดที่พูดภาษาฝรั่งเศสในประเทศสวิตเซอร์แลนด์ในปี ค.ศ. 1888 การสร้าง *linear model* สำหรับข้อมูลการเจริญพันธุ์กับตัวแปรอื่น ๆ มีความเหมาะสมหรือไม่

พิจารณาจาก *scatter plot matrix* ทำให้เห็นถึงความแตกต่างระหว่างแคทอริก และโปแตสแตนต์ ดังนั้นการสร้าง *scatter plot matrix* จึงมีประโยชน์ *linear model* แสดงให้เห็นถึงความสัมพันธ์ระหว่างภาวะเจริญพันธุ์ และระดับการศึกษาอย่างมีนัยสำคัญทางสถิติ พิจารณา *linear model* ที่ดีที่สุดทำได้ด้วยการใช้คำสั่ง `step()` จะเห็นว่าตัวแปร 'Examination' สามารถละออกจากโมเดลได้ ในขณะที่ตัวแปรอื่น ๆ มีความสัมพันธ์กับภาวะเจริญพันธุ์อย่างมีนัยสำคัญทางสถิติ



รูปที่ 6.15 Scatter plot matrix สำหรับข้อมูล swiss

ตัวอย่างคำสั่งดังต่อไปนี้

```
> data(swiss)
> pairs(swiss, panel=panel.smooth, col=3+(swiss$Catholic>50))

> fm <- lm(Fertility~., data=swiss)
> summary(fm)
```

Call:
lm(formula = Fertility ~ ., data = swiss)

Residuals:

Min	1Q	Median	3Q	Max
-15.274	-5.262	0.503	4.120	15.321

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	66.9152	10.7060	6.25	1.9e-07
Agriculture	-0.1721	0.0703	-2.45	0.0187
Examination	-0.2580	0.2539	-1.02	0.3155
Education	-0.8709	0.1830	-4.76	2.4e-05
Catholic	0.1041	0.0353	2.95	0.0052
Infant.Mortality	1.0770	0.3817	2.82	0.0073

Residual standard error: 7.17 on 41 degrees of freedom
 Multiple R-Squared: 0.707, Adjusted R-squared: 0.671
 F-statistic: 19.8 on 5 and 41 DF, p-value: 5.59e-10

> step(fm)

Start: AIC= 190.69

Fertility ~ Agriculture + Examination + Education + Catholic +
 Infant.Mortality

	Df	Sum of Sq	RSS	AIC
- Examination	1	53	2158	190
<none>			2105	191
- Agriculture	1	308	2413	195
- Infant.Mortality	1	409	2514	197
- Catholic	1	448	2553	198
- Education	1	1163	3268	209

Step: AIC= 189.86

Fertility ~ Agriculture + Education + Catholic + Infant.Mortality

	Df	Sum of Sq	RSS	AIC
<none>			2158	190
- Agriculture	1	264	2422	193
- Infant.Mortality	1	410	2568	196
- Catholic	1	957	3115	205
- Education	1	2250	4408	221

Call:

lm(formula = Fertility ~ Agriculture + Education + Catholic +
 Infant.Mortality, data = swiss)

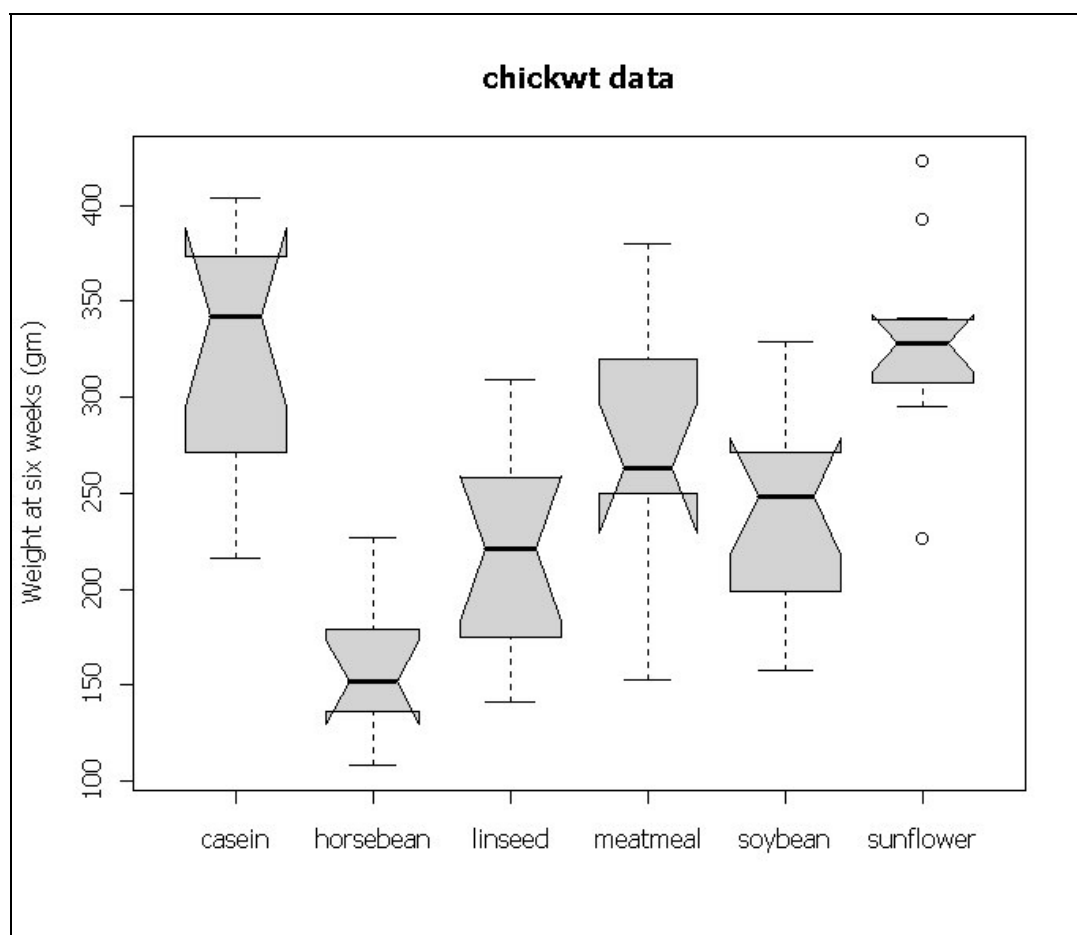
Coefficients:

(Intercept)	Agriculture	Education	Catholic	Infant.Mortality
62.101	-0.155	-0.980	0.125	1.078

ตัวอย่างที่ 6.8 ชุดข้อมูล 'chickwts' ประกอบด้วยผลจากการทดลองเพื่อวัดและเปรียบเทียบประสิทธิภาพของอาหารเสริมชนิดต่าง ๆ ต่ออัตราการเจริญเติบโตของไก่ ลูกไก่ที่ถูกฟักออกมาจะถูกสุ่มแบ่งเป็น 6 กลุ่ม แต่ละกลุ่มจะได้รับอาหารเสริมแตกต่างกัน ซึ่งน้ำหนักไก่หลังจากการให้อาหารนี้เป็นระยะเวลา 6 สัปดาห์ สร้างกราฟด้วยคำสั่ง `boxplot0` และวิเคราะห์ความสัมพันธ์ระหว่างน้ำหนักไก่หลังจากเลี้ยงด้วยอาหารเสริมเป็นระยะเวลา 6 เดือน ทดสอบว่าน้ำหนักไก่มีความแตกต่างกันหรือไม่ในแต่ละกลุ่ม

จากรูปที่ 6. 16 จะพบว่าอาหารเสริมด้วย *casein* และ *sunflower* มีผลต่ออัตราการเจริญเติบโตของไก่ ในขณะที่ *hoersebean* มีผลต่ำที่สุด จากกราฟจะเห็นว่าอาหารเสริม *sunflower* มีค่า outlier จำนวนหนึ่ง

แม้ว่าข้อมูลจะมีการแจกแจงปกติ แต่ค่าความแปรปรวนไม่เท่ากัน ดังนั้นจึงไม่สามารถทดสอบด้วย *ANOVA* การทดสอบแบบ *Kruskal Wallis* สำหรับน้ำหนักไก่แต่ละกลุ่มมีความแตกต่างกันอย่างมีนัยสำคัญทางสถิติ การพิจารณาความแตกต่างระหว่างน้ำหนักไก่ที่เลี้ยงด้วยอาหารเสริมในแต่ละชนิดโดยใช้ *linear model* ผลจาก *linear model* พบว่าการเลี้ยงไก่ด้วยอาหารเสริมมีผลต่ออัตราการเพิ่มของน้ำหนักไก่ ยกเว้นอาหารเสริมชนิด *sunflower*



รูปที่ 6.16 Boxplots สำหรับข้อมูล chickwts

ตัวอย่างคำสั่งดังต่อไปนี้

```
> data(chickwts)
> boxplot(weight ~ feed, data = chickwts, col = "lightgray",
  varwidth = TRUE, notch = TRUE, main = "chickwt data",
  ylab = "Weight at six weeks (gm)")
```

```
> summ(weight, group=feed)
weight stratified by feed
```

	Obs.	NA	Mean	S.D.	Min.	Median	Max.
casein	12	0	323.58	64.43	216	342	404
horsebean	10	0	160.2	38.63	108	151.5	227
linseed	12	0	218.75	52.24	141	221	309
meatmeal	11	0	276.91	64.9	153	263	380
soybean	14	0	246.43	54.13	158	248	329
sunflower	12	0	328.92	48.84	226	328	423

```
> kruskal.test(weight~feed, data=chickwts)
```

Kruskal-Wallis rank sum test

data: weight by feed

Kruskal-Wallis chi-squared = 37.343, df = 5, p-value = 5.113e-07

```
> fm <- lm(weight~feed, data=chickwts)
```

```
> summary(fm)
```

Call:

```
lm(formula = weight ~ feed, data = chickwts)
```

Residuals:

Min	1Q	Median	3Q	Max
-123.91	-34.41	1.57	38.17	103.09

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	323.58	15.83	20.44	< 2e-16
feedhorsebean	-163.38	23.49	-6.96	2.1e-09
feedlinseed	-104.83	22.39	-4.68	1.5e-05
feedmeatmeal	-46.67	22.90	-2.04	0.04557
feedsoybean	-77.15	21.58	-3.58	0.00067
feedsunflower	5.33	22.39	0.24	0.81249

Residual standard error: 54.9 on 65 degrees of freedom

Multiple R-Squared: 0.542, Adjusted R-squared: 0.506

F-statistic: 15.4 on 5 and 65 DF, p-value: 5.94e-10

แบบฝึกหัด

จากข้อมูล `birthwt3.tab` เป็นข้อมูลปัจจัยที่มีผลต่อน้ำหนักแรกคลอดของเด็กทารก ประกอบด้วย ตัวแปร

<code>bwt</code>	น้ำหนักแรกคลอดของเด็กทารก
<code>gestation</code>	อายุครรภ์ขณะคลอด
<code>parity</code>	จำนวนครั้งของการตั้งครรภ์
<code>age</code>	อายุของมารดา
<code>height</code>	น้ำหนักมารดา
<code>weight</code>	ส่วนสูงมารดา
<code>smoke</code>	การสูบบุหรี่ของมารดา (0=ไม่สูบ, 1=สูบ)

จงสร้างกราฟด้วยคำสั่ง `pairs()` เพื่อสำรวจความสัมพันธ์ระหว่างตัวแปร
 จงวิเคราะห์หาปัจจัยที่มีผลต่อน้ำหนักแรกคลอดของเด็กทารก

บทที่ 7

การเขียนฟังก์ชัน

7.1 การเขียน function ใน R

function หรือ คำสั่ง ใน R คือวัตถุชนิดหนึ่งที่ใช้สำหรับการทำงานต่างๆ เช่นคำสั่ง `help` เป็นต้น ในการออกคำสั่งให้ function ทำงาน ทำได้โดยการเขียนชื่อ function นั้น ตามด้วยวงเล็บ ซึ่งอาจจะมี argument หรือไม่มี argument อยู่ภายในก็ได้ เช่น `help.start()` ทำงานได้โดยไม่ต้องใส่ argument ใดๆ แต่ `help()` จะต้องมีชื่อของคำสั่งที่ต้องการดูความช่วยเหลือเป็น argument ด้วย เช่น `help(pack)`

การเขียน function ใ้เองใน R นั้นสามารถทำได้ง่าย เพียงแค่เรียนรู้สิ่งที่เกี่ยวข้องกับ function เพิ่มเติมอีกเล็กน้อยเท่านั้น ลองศึกษาจากขั้นตอนต่อไปนี้

7.1.1 *function เป็นวัตถุ*

จึงต้องมีชื่อ และถูกสร้างด้วยคำสั่ง `function()` ตามด้วยวงเล็บหนึ่งคู่ และวงเล็บปีกกาอีกหนึ่งคู่ ดังตัวอย่างการสร้าง `myfun` ดังนี้

```
> myfun <- function () {}
```

ตอนนี้เราจะได้ function ชื่อ `myfun` ที่ไม่ได้ทำงานอะไร และสามารถเรียกใช้ได้โดยเรียก `myfun()` บน R console ซึ่งจะได้ผลลัพธ์เป็น NULL คือไม่มีอะไรเป็นผลลัพธ์ ดังนี้

```
> myfun()
NULL
```

7.1.2 *เขียนขั้นตอนการทำงานของ function ไว้ในวงเล็บปีกกา*

ดังตัวอย่างข้างต้น ต้องการให้ `myfun` พิมพ์ประโยคว่า This is my first function. ก็เขียนกำหนด `myfun` ใหม่ดังนี้

```
> myfun <- function () {
+   cat("This is my first function.\n")
+ }
```

สังเกตว่าเมื่อแยกเขียนวงเล็บปีกกาเปิดไว้ แล้วเคาะ Enter เพื่อจะเขียนข้อความบรรทัดต่อไป prompt จะเปลี่ยนเป็นเครื่องหมาย + ในบรรทัดที่ 2 เขียนประโยค `cat("This is my first function.\n")` เพื่อเป็นการบอกให้พิมพ์ประโยค This is my first function. บน R console สังเกตว่าในที่นี้ปิดท้ายประโยคด้วย `'\n'` หมายความว่าให้ขึ้นบรรทัดใหม่หลังจากแสดงข้อความเสร็จแล้ว จากนั้นเคาะ Enter อีกครั้ง แล้วปิดด้วยวงเล็บปีกกาปิด คราวนี้เมื่อเรียกใช้งาน `myfun` จะได้ผลลัพธ์ดังนี้

```
> myfun()
This is my first function.
```

7.1.3 เขียนคำสั่งใน script file

ดีกว่าเขียนตรงๆ บน R console เนื่องจากหากพิมพ์ผิดพลาด การกลับไปแก้ไขยุ่งยาก ดังนั้นสร้างแฟ้ม `myfun.R` ไว้ใน working directory ของ R ที่กำลังทำงานอยู่ แล้วพิมพ์ส่วนของ function ไว้ในแฟ้มนั้น ดังนี้

```
> myfun <- function () {
  cat("This is my first function.\n")
}
```

เมื่อจะเรียกใช้งาน function ก็เรียกโดยการใช้คำสั่ง `source` โดยพิมพ์ใน R console ดังตัวอย่างข้างล่าง `myfun` จะถูกเรียกเข้ามาในหน่วยความจำของ R โดยไม่แสดงแต่อย่างใด แต่หลังจากถูกเรียกเข้ามาแล้ว ก็สามารถเรียกใช้ `myfun()` ได้ ดังตัวอย่าง

```
> source("myfun.R")
> myfun()
This is my first function.
```

ในที่นี้ ใช้รูปแบบการย่อหน้าที่นิยมในการเขียนภาษา C ซึ่งเป็นส่วนหนึ่งของที่มาหรือต้นกำเนิดของภาษา R คือเมื่อจะเขียนเครื่องหมายวงเล็บปีกกาเปิดและปิดแยกกัน ให้เขียนวงเล็บปีกกาเปิดไว้ท้ายประโยค หรือขึ้นบรรทัดใหม่ที่ตำแหน่งเดียวกันกับตัวอักษรแรกของชื่อ function และปิดวงเล็บปีกกาโดยขึ้นบรรทัดใหม่เปล่า โดยวางไว้ที่ตำแหน่งตรงกับตัวอักษรแรกของชื่อ function

```
myfun <- function () {
  cat("This is my first function.\n")
}
```

หรือ

```
myfun <- function ()
{
  cat("This is my first function.\n")
}
```

ที่นิยมกันเช่นนี้เนื่องจากในการเขียน function ที่ทำงานซับซ้อน เรามักจำเป็นต้องใช้เครื่องหมาย { } ซ้อนกันหลายครั้ง การเขียนในลักษณะเช่นนี้จะทำให้ติดตามได้ว่าวงเล็บปีกกาปิดอันไหนคู่กับวงเล็บปีกกาเปิดอันไหน จะได้ไม่สับสน จำไว้ว่าขอมเสียบรรทัดมากขึ้นดีกว่าสับสนว่าปิดวงเล็บครบทุกอันหรือยัง

ความนิยมอีกอย่างหนึ่งคือ ข้อความบรรทัดต่างๆ ภายในวงเล็บปีกกา จะย่อหน้าเข้าไปจากตำแหน่งของวงเล็บปีกกา 1 tab (3 ถึง 5 ตัวอักษร แล้วแต่จะตั้ง) และหากมีการใช้ { } ซ้อนอีก ก็ย่อหน้าเข้าไปอีก ดังตัวอย่างข้างล่าง ซึ่งจะทำให้การเขียนคำสั่งสวยงาม และติดตาม block ของคำสั่งได้ง่าย

```
do <- function (file, prompt.echo = ">>", from = 1, to = -1)
{
  script.version <- function(lns) {
    if (substr(lns[1], 1, 1) != "#")
      result <- try(eval(parse(text = lns[1]), envir = .GlobalEnv),
        silent = TRUE)
    if ((substr(lns[1], 1, 5) == "clean" | substr(lns[1],
      1, 2) == "rm") & substr(lns[2], 1, 1) != "#")
      result <- try(eval(parse(text = lns[2]), envir = .GlobalEnv),
        silent = TRUE)
    if (exists(".ec.version", env = .GlobalEnv)) {
      version <- get(".ec.version", env = .GlobalEnv)
    }
    return(version)
  }
}
```

7.1.4 ส่งผ่าน argument ในวงเล็บ

ซึ่งหากมี argument หลายตัวก็แยกด้วยจุลภาค ',' ดังตัวอย่างข้างต้น เราอาจเปลี่ยน myfun ให้รับข้อความไปแสดงออกบน R console แล้วขึ้นบรรทัดใหม่ทันทีก็ได้ โดยแก้ไข myfun ในแฟ้ม myfun.R ดังนี้

```
myfun <- function (x) {
  cat(x, "\n")
}
```

จากนั้นเรียก source("myfun.R") ใหม่อีกครั้ง myfun เดิมจะถูกแทนที่ด้วยคำสั่งที่แก้ไขใหม่แล้วนั้น จากนั้นเราสามารถส่งผ่านข้อความใน myfun ได้ดังนี้

```
> source("myfun.R")
> myfun("Here it is.")
Here it is.
>
```

7.1.5 *function* ใน R สามารถรับ *argument* ชนิดต่างๆ กันได้ โดยไม่จำกัด

และไม่ต้องระบุชนิดเหมือนภาษาโปรแกรมอื่นบางภาษา ลองดูผลการส่งผ่านวัตถุชนิดต่างๆ เป็น *argument* เช่น ตัวเลข หรือเวกเตอร์ ดูผลลัพธ์ดังข้างล่าง

```
> myfun(2*5)
10
> x <- c(1,2,3,4,5,6)
> myfun(x)
1 2 3 4 5 6
```

7.1.6 การระบุการทำงานของ *function* ให้แตกต่างกันตามคลาสของ *argument* ตัวแรก

ในข้อ 7.1.5 ข้างต้น *function* ทำงานโดยไม่พิจารณาถึงความแตกต่างของคลาสของ *argument* แต่เราสามารถทำให้ *function* ทำงานต่างกันตามคลาสของ *argument* ตัวแรกได้ คุณสมบัตินี้เรียกว่า **polymorphism** ของ *function* โดยมีขั้นตอนในการทำต่อไปนี้

ขั้นแรก ระบุวิธีการส่งผ่านร่วม โดยเขียนคำสั่ง `UseMethod` โดยระบุ `string` ของชื่อ *function* ร่วมกัน (เขียนไว้ในเครื่องหมายคำพูด) ทำยบรรทัดกำหนด *function* หรือจะขึ้นบรรทัดใหม่ก็ได้ จากนั้นเขียน *function* ใหม่โดยใช้ชื่อ *function* เดิมนั้น ตามด้วย `'.'` และชื่อคลาสของ *argument* โดยเขียนวิธีการทำงานให้เหมาะสมกับคลาสของ *argument* นั้นๆ และบางครั้ง หากการจัดการกับ *argument* ในคลาสนั้นนอกเหนือจากนั้นมีการทำงานที่เฉพาะออกไป ก็จำเป็นต้องมี *function* ที่เป็น `'default'` เพื่อจัดการด้วย ดังตัวอย่าง

```
myfun <- function (x)
UseMethod("myfun")

myfun.character <- function (x)
{
  cat("The STRING is:", x, "\n")
}

myfun.numeric <- function (x)
{
  cat("The NUMBER is:", x, "\n")
}

myfun.default <- function (x)
{
  cat("The OBJECT is:", x, "\n")
}
```

จะเห็นได้ว่าเมื่อเรียก `source("myfun.R")` ใหม่อีกครั้ง `myfun` จะทำงานต่างกันตามแต่ argument ที่ส่งผ่านเข้าไป ดังนี้

```
> myfun("Here it is.")
The STRING is: Here it is.
> myfun(3*4)
The NUMBER is: 12
```

7.1.7 การระบุชื่อ *function* หลายชื่อให้ทำงานอย่างเดียวกัน

สิ่งนี้มีประโยชน์ในการเรียก *function* ด้วยชื่อย่อ เพื่อให้พิมพ์คำสั่งเพียงสั้นๆ การทำงานทำได้โดยใช้ `UseMethod` อ้างไปถึง *function* เดียวกัน ดังตัวอย่าง

```
myfun <- function (x)
  UseMethod("myfun")

myf <- function (x)
  UseMethod("myfun")

my <- function (x)
  UseMethod("myfun")

myfun.character <- function (x)
{
  cat("The STRING is:", x, "\n")
}
...
```

ซึ่งมีความหมายว่าเรากำหนดให้ชื่อ `myfun`, `myf` และ `my` เป็น *function* ที่ไปเรียกชุดวิธีการทำงานเดียวกัน คือ `myfun.character`, `myfun.numeric` และ `myfun.default` ดังนั้นเราสามารถเรียกคำสั่ง `myfun` ได้ด้วยชื่อย่อได้อีกเป็น `myf` หรือ `my` ตามที่กำหนดนั่นเอง

7.2 การนำชื่อกรอบข้อมูลและตัวแปรมาใช้

พึงสังเกตว่าการนำชื่อวัตถุใดๆ มาใช้นั้น เราไม่สามารถเรียกได้จากฟังก์ชัน `names()` เพราะชื่อวัตถุใดๆ ไม่ได้เป็น attribute ดังนั้นในการนำชื่อวัตถุออกมา จึงต้องใช้วิธีที่อ้อมเล็กน้อย โดยใช้ฟังก์ชัน `substitute()` ซึ่งทำหน้าที่ค้นหาชื่อของวัตถุนั้นในเส้นทางค้นหาในสิ่งแวดล้อมขณะนั้น

7.2.1 การนำชื่อกรอบข้อมูลมาใช้ เมื่อส่งผ่านกรอบข้อมูลเป็น *argument*

เมื่อต้องการนำชื่อ *data frame* มาใช้ เช่นในกรณีที่ส่งผ่าน *data* เป็น *argument* ของฟังก์ชัน `data.name()` เราสามารถใช้ฟังก์ชัน `substitute()` เพื่อนำชื่อ *data* ออกมาได้ดังนี้

```
library(ice)
data(cca)
data.name <- function (data, ...)
{
  data.name <- substitute(data)
  cat(data.name, "\n")
}
data.name(cca)
```

เนื่องจากฟังก์ชัน `substitute()` ทำหน้าที่ค้นหาชื่อของวัตถุนั้นในเส้นทางค้นหาในสิ่งแวดล้อมขณะนั้น ดังนั้นเมื่อออกคำสั่ง `substitute(cca)` จะได้ชื่อ `cca` กลับออกมาดังนี้

```
> substitute(cca)
cca
```

นั่นคือเราสามารถใช้ฟังก์ชัน `substitute()` เพื่อหาชื่อของวัตถุนั้นได้ ผลลัพธ์ที่ได้จากการเรียกฟังก์ชัน `data.name()` จะเป็นดังนี้

```
> data.name(cca)
cca
```

7.2.2 การนำชื่อตัวแปรในกรอบข้อมูลมาใช้ เมื่อส่งผ่านกรอบข้อมูลและชื่อตัวแปรเป็น *argument*

ในกรณีที่ส่งผ่าน `variable` เป็น `argument` ในลักษณะที่เป็น `vector` เราสามารถนำชื่อ `variable` ใน `data frame` ออกมาได้ด้วยการใช้ฟังก์ชัน `substitute` เช่นกัน แต่ควรใช้ร่วมกับฟังก์ชัน `deparse()` ด้วย ดังในตัวอย่างฟังก์ชัน `var.name()` ดังนี้

```
var.name <- function (data, var)
{
  data.name <- substitute(data)
  var.name <- deparse(substitute(var))
  cat(data.name, var.name, "\n")
}
var.name(cca, status)
```

ฟังก์ชัน `deparse()` ทำหน้าที่จับข้อความที่ใส่ใน `argument` ออกมาเป็น `string` ข้อดีของการใช้ `deparse()` ร่วมด้วยก็คือ หากมีการ `subset variable` นั้นด้วย ก็จะได้ข้อความที่ถูกต้องกลับมา มิฉะนั้นการใช้ `substitute()` เฉยๆ จะได้ผลลัพธ์เป็นข้อความว่าง ผลลัพธ์ที่ได้จากการเรียกฟังก์ชัน `data.name()` จะเป็นดังนี้

```
> var.name(cca, status)
cca status
```

อีกวิธีหนึ่งในการหาชื่อของ variable ใน data frame มาใช้ ในกรณีที่รู้ลำดับของ variable นั้นใน data frame คือการใช้ฟังก์ชัน `names` เนื่องจากในขณะนี้ variable อยู่ภายใน data frame จึงเป็น attribute หนึ่งใน data frame นั้น เราจึงใช้ฟังก์ชัน `names` เพื่อหาชื่อ variable ภายใน data frame ได้ดังนี้

```
var.name <- function (data, var.order)
{
  data.name <- substitute(data)
  var.name <- names(data)[var.order] #<--
  cat(data.name, var.name, "\n")
}
var.name(cca, 1)
var.name(cca, 2)
```

ผลที่ได้เป็นดังนี้

```
> var.name(cca, 1)
cca status
> var.name(cca, 2)
cca ovab
```

แต่สังเกตว่าในกรณีนี้เราต้องรู้ลำดับตัวแปรใน data frame จึงจะหาชื่อ variable ได้ เราสามารถรวมสองวิธีนี้เข้าเป็นฟังก์ชันเดียวกันได้ โดยการตรวจสอบว่า argument ลำดับที่สองที่ส่งเข้ามาเป็น vector หรือเป็นตัวเลขเดี่ยวๆ ซึ่งทำได้อย่างน้อยสองวิธี วิธีหนึ่งคือ ตรวจสอบความยาว (length) ของ argument นั้น ถ้าความยาวเป็น 1 ก็ (คาดได้ว่า) เป็นตัวเลขลำดับ การตรวจสอบความยาวเช่นนี้จะผิดพลาดได้ถ้า data frame นั้นมี variable เพียงตัวเดียว แต่คงเป็นไปได้ที่จะมี data frame ที่มีตัวแปรเดียวเช่นนั้น ถ้าสร้าง data frame อย่างถูกต้องตามหลักการของฐานข้อมูลที่ดี คือจะต้องมี primary key เป็น identifier อยู่ด้วย ซึ่งจะทำให้ data frame ต้องมี variable อย่างน้อยสองตัวเสมอ การตรวจสอบอีกวิธีหนึ่งที่ไม่น่าจะเกิดความผิดพลาดไม่ว่ากรณีใดๆ คือใช้ฟังก์ชัน `is.vector()` ซึ่งถึงแม้ variable นั้นจะเป็น factor ก็จะทำให้ผลเป็น TRUE เช่นกัน แต่ถ้าเป็นตัวเลขเดี่ยวๆ จะให้ผลเป็น FALSE ในที่นี้จะไม่แสดงให้ดู ลองทำเองเป็นการทดสอบความรู้ได้

7.2.3 การนำชื่อกรอข้อมูลกับชื่อตัวแปรมาต่อกันด้วย "\$"

สามารถทำได้อย่างง่ายด้วยการใช้ฟังก์ชัน `paste()` แล้วใช้ `sep = "$"` ดังตัวอย่างฟังก์ชัน `var.name()` ในข้อ 2 สามารถเขียนใหม่ได้ดังนี้

```
var.name <- function (data, var)
{
  data.name <- substitute(data)
```

```

var.name <- deparse(substitute(var))
data.and.var <- paste(data.name, var.name, sep="$")
cat(data.and.var, "\n")
}

```

ผลลัพธ์ที่ได้จากการเรียกฟังก์ชัน `var.name()` จะเป็นดังนี้

```

> var.name(cca, status)
cca$status

```

ลองส่งค่ากลับออกมาให้ `var.name()` แล้วเรียกดู type และ class ของผลลัพธ์

```

var.name <- function (data, var)
{
  data.name <- substitute(data)
  var.name <- deparse(substitute(var))
  data.and.var <- paste(data.name, var.name, sep="$")
  cat(data.and.var, "\n")
  return(var.name)
}
x <- var.name(cca, status)
typeof(x)
class(x)

```

จะได้ผลที่น่าพอใจดังนี้

```

> x <- var.name(cca, status)
cca$status
> typeof(x)
[1] "symbol"
> class(x)
[1] "name"

```

สังเกตว่าในขณะนี้เราใช้ฟังก์ชัน `paste()` เพื่อนำผลลัพธ์มาเก็บไว้ในวัตถุหนึ่ง ค่า default ของตัวแยก string ของ `paste` เป็นช่องว่าง 1 ช่อง แต่ในที่นี้เราจะแทนที่ด้วย "\$" ก็จะทำให้ข้อทั้งสองต่อกันทันทีด้วยเครื่องหมาย \$

เราอาจสั่งให้พิมพ์ชื่อออกมาบน console โดยตรงโดยใช้ `cat` เลยก็ได้ แต่ต้องระวังว่าด้วยค่า default ของ `cat` นั้น string จะถูกแยกด้วยช่องว่าง 1 ช่องเช่นกัน จึงต้องระบุ `sep=""` ดังนี้

```

var.name <- function (data, var)
{
  data.name <- substitute(data)
  var.name <- deparse(substitute(var))
  cat(data.name, "$", var.name, "\n", sep="")
}

```

หรืออาจใช้ `sep="$"` ได้เช่นกัน ผลลัพธ์จะยังไม่ได้ดังที่ต้องการ คือมี "\$" ติดมาตอนท้ายด้วย ดังนี้

```
#1 cat(data.name,var.name,"\n", sep="$")
> var.name(cca,status)
cca$status$
```

ที่เป็นเช่นนี้เพราะ cat จะมองว่า "\n" เป็น string อีกอันหนึ่ง ที่ต้องคั่นด้วย "\$" เช่นกัน จึงต้องแยก "\n" ออกมาต่างหาก ก็จะได้ผลอย่างที่ต้องการ ดังนี้

```
#2 cat(data.name,var.name, sep="$"); cat("\n")
> var.name(cca,status)
cca$status
```

7.3 การสร้าง package ใน R

Library กับ package ใน R มีความหมายเช่นเดียวกันนั่นเอง เป็นการนำ function และ/หรือ data มารวมกันเป็นชุด ชื่อนี้จะนำไปนี้เป็นสิ่งที่จำเป็นอย่างน้อยที่สุดในการสร้าง package ให้ทำงานได้ หากต้องการให้ package ทำงานหรือมีคุณสมบัติอย่างอื่นๆ เพิ่มเติมมากกว่านี้ ควรอ่านรายละเอียดในคู่มือ Writing R Extensions

ในบทนี้จะกล่าวถึงเฉพาะการสร้างแฟ้มและไฟล์เดอร์ต่างๆ เท่านั้น ส่วนการ build package นั้นจะได้กล่าวในบทต่างหากต่อไป เนื่องจากมีขั้นตอนซับซ้อนพอสมควร และยังจะต้องติดตั้งโปรแกรมที่เกี่ยวข้องอีกมากมาย

ในการสร้าง package หนึ่งๆ นั้น เริ่มจากการสร้างไฟล์เดอร์ในชื่อของ package ที่ต้องการ แล้วสร้างไฟล์เดอร์ย่อยต่อไป (เป็นอย่างน้อย) แฟ้มและไฟล์เดอร์เหล่านี้ใช้ได้เหมือนกันในการ build package ในทุก platform ไม่ว่าจะเป็น Windows, Unix, หรือ Macintosh ก็ตาม ไฟล์เดอร์และแฟ้มเหล่านี้ได้แก่

1. R บรรจุแฟ้มของ function ต่างๆ
2. data บรรจุแฟ้มข้อมูลที่เรียกใช้ในตัวอย่าง ไม่จำเป็นต้องมีไฟล์เดอร์นี้ก็ได้
3. man บรรจุแฟ้มช่วยเหลือและตัวอย่างการใช้งานสำหรับ function ต่างๆ ใน package

แม้ว่าไฟล์เดอร์ย่อย man นี้จะไม่จำเป็น แต่ควรมีไว้ เพื่ออธิบายวิธีการใช้งาน function ต่างๆ เช่น argument ที่สามารถใช้ได้ ลำดับของ argument ข้อจำกัดของ function และอื่นๆ แม้ว่าสิ่งเหล่านี้ อาจเขียนเป็นเอกสารคู่มือได้ แต่ผู้ใช้จะสะดวกมากกว่าหากเรียกดูข้อความช่วยเหลือได้ทันทีหากลืมวิธีใช้หรือต้องการดูตัวอย่างวิธีใช้งานจาก R console หรือในสภาพแวดล้อมของการทำงานในขณะนั้น

นอกจากไฟล์เดอร์ย่อยทั้งสามแล้ว สิ่งที่เป็นมากอีกอย่างหนึ่งคือแฟ้มที่ชื่อ DESCRIPTION แฟ้มนี้ต้องเขียนด้วยอักษรตัวใหญ่ และไม่มีนามสกุลใดๆ ในบางครั้งโปรแกรม

editor บางตัวที่ใช้อาจใส่ নামสกุลให้โดยอัตโนมัติ ถ้าพบว่ามี การเติมนามสกุลให้เอง ต้องลบออกเสีย

ตัวอย่างเช่น การสร้าง package ชื่อ mypack ให้สร้างไฟล์เดอร์ชื่อ mypack ที่บรรจุไฟล์เดอร์ย่อยและแฟ้มดังนี้

ไฟล์เดอร์หลัก	แฟ้มในไฟล์เดอร์หลัก	ไฟล์เดอร์ย่อย	แฟ้มในไฟล์เดอร์ย่อย
mypack	DESCRIPTION		
		R	mypack.R
		data	FILELIST examdata.R ...
		man	myfun.Rd ...

ขอย้ำอีกครั้งหนึ่งว่าโครงสร้างนี้เป็นโครงสร้างอย่างน้อยที่สุดของ package ที่ควรมี ในคู่มือ Writing R Extensions จะแสดงให้เห็นว่ายังมีแฟ้มและไฟล์เดอร์ย่อยที่อาจเพิ่มเติมได้อีกมาก โดยที่แฟ้มหรือไฟล์เดอร์เหล่านั้นใช้สำหรับการเพิ่มการทำงานที่พิเศษหรือเฉพาะด้าน แต่ผู้ใช้งานใหญ่มักไม่ได้ใช้

7.3.1 แฟ้ม DESCRIPTION

เป็นแฟ้มที่บรรจุคำอธิบาย package นั้น ประกอบด้วยหัวข้อย่อยๆ ดังตัวอย่างแฟ้ม DESCRIPTION ของ package epican ดังนี้

```
Package: epican
Version: 1.3.3
Date: 2005-05-08
Title: Functions for Epidemiology
Depends: R (>= 2.0.0)
Maintainer: Epidemiology Unit, Prince of Songkla University
Author: Epidemiology Unit Faculty of Medicine and Department of
Statistics, Faculty of Science
Description: Common Statistical Functions for Epidemiology
License: GPL version 2 or newer
```

Package	ชื่อของ package
Version	รุ่นของ package
Date	วันที่พัฒนา package รุ่นนั้น (หากเปิดดูแฟ้มจาก package ที่พร้อมใช้งานใน ../R/library แล้วจะพบว่าบรรทัดสุดท้ายจะมีวันเวลาอีกครั้งหนึ่ง ส่วนนั้นไม่ต้องเขียน แต่ R จะสร้างให้เองในเวลาที build package นั้นให้เองโดยอัตโนมัติ ขั้นตอนนี้จะได้กล่าวต่อไป)

Title	หัวข้อที่จะแสดงเมื่อเรียกใช้งาน เช่นเมื่อเรียกดูข้อความช่วยเหลือ
Depends	รุ่นของ R ที่ package นั้นสามารถใช้งานได้ (เมื่อเปลี่ยนรุ่นของ R อาจมีบาง function ทำงานเปลี่ยนไปจากเดิม หรือยกเลิกไป หรือเพิ่มขึ้นมาใหม่ ทำให้ function ที่เรียกใช้ function เหล่านั้นอีกต่อหนึ่งทำงานไม่ถูกต้องหรือทำงานไม่ได้ไปเลยก็มี)
Maintainer	ผู้ดูแลรับผิดชอบ package
Author	ผู้เขียน package
Description	คำอธิบาย package อย่างสั้นๆ
License	ลักษณะของลิขสิทธิ์ของ package นั้น โดยปกติก็จะเป็น GPL ดังในตัวอย่าง

ไม่จำเป็นต้องระบุทุกหัวข้อก็ได้

7.3.2 โฟลเดอร์ย่อย R

เป็นโฟลเดอร์ที่บรรจุเพิ่ม function ต่างๆ ที่ต้องการรวมให้เป็น package เดียวกัน มักเป็นโฟลเดอร์ที่สำคัญที่สุดที่ต้องมีอยู่ใน package โดยทั่วไปหากมี function ไม่มากนัก ก็รวมอยู่ในแฟ้มเดียวกันได้ และมักนิยมตั้งชื่อให้ตรงกับชื่อ package นั้น และมีนามสกุลเป็น .R แต่หากมี function จำนวนมากก็อาจแบ่งเป็นหลายแฟ้มก็ได้ ในขณะที่ build นั้น R จะรวม function ในแฟ้มต่างๆ เข้าเป็น package เดียวกันให้เอง

ภายในแฟ้มนอกจากจะบรรจุ function แล้ว ยังอาจกำหนดค่าให้ตัวแปร global หรือตัวแปรเฉพาะของ package ด้วยก็ได้ สามารถใส่ comment ต่างๆ ได้เช่นเดียวกับการเขียนชุดคำสั่ง R โดยทั่วไปนั่นเอง

ตัวอย่างเช่น ใน package epican (ชื่อเดิมของ ice) รุ่นก่อนหน้า 1.3.3 มีแฟ้มเพียงแฟ้มเดียวคือ epican.R แต่ในรุ่น 1.3.3 ประกอบด้วยแฟ้มชื่อ data.man.R, data.summ.R, file.man.R, general.R, miscellaneous.R, stat.model.R, และ stat.test.R เป็นการแตกแฟ้มเดิมที่มีขนาดใหญ่ให้เป็นแฟ้มย่อยๆ เป็นกลุ่มๆ ของ function นั้นเอง เนื่องจากในแฟ้มขนาดใหญ่ นั้น หากพบข้อผิดพลาดขณะ build package จะหาตำแหน่งที่ผิดพลาดได้ยาก เพราะเป็นไปได้มากกว่าตำแหน่งบรรทัดที่ R แจ้งว่ามีข้อผิดพลาดจะไม่ใช่ตำแหน่งที่ผิดพลาดจริง คือเกิดความผิดพลาดมาก่อนหน้านั้นหลายบรรทัดแล้ว แต่ยังตรวจสอบไม่พบ เช่นการปิดวงเล็บปีกกา การตรวจสอบจะบอกไม่ได้ว่าควรปิดที่ตรงไหน แต่จะตรวจพบได้เมื่อพยายามหาวงเล็บปิดแล้วไม่พบ เป็นต้น (แพคเกจ ice ปัจจุบันมีถึง 51 แฟ้ม)

ขอให้สังเกตว่าในการตั้งชื่อแฟ้มนั้น เราอาจใส่จุดค้นชื่อหลายๆ จุดก็ได้ หรืออาจใช้เครื่องหมาย – หรือ _ เพื่อค้นคำที่มาประกอบกันเป็นชื่อก็ได้

7.3.3 โฟลเดอร์ย่อย *data*

เป็นโฟลเดอร์ที่บรรจุแฟ้มข้อมูลที่ใช้ในการแสดงตัวอย่างในแฟ้มช่วยเหลือหรือแฟ้มสาธิต (demo) แฟ้มข้อมูลนี้อาจสร้างได้สองวิธีคือ การสร้างโดยคำสั่ง `structure` แล้วกำหนด `class` ให้เป็น `"data.frame"` ดังตัวอย่างการสร้าง dataset ที่ชื่อ `evans` ดังนี้

```
"evans" <- structure(list(
  chd = structure(factor(c(rep(1,538),rep(2,71))),
    levels=1:2),
  .Label=c("no ", "yes")),

  chl = structure(factor(c(rep(1,447),rep(2,91),rep(1,57),rep(2,14))),
    levels=1:2),
  .Label=c("low ", "high"))),

row.names = paste(1:609),
class = "data.frame")
```

โดยบรรจุชุดคำสั่งทั้งหมดนี้ในแฟ้มชื่อ `evans.R` หรืออาจใช้คำสั่ง `read` อ่านข้อมูลเข้ามาก็ได้ ดังตัวอย่างคำสั่งในแฟ้มที่ชื่อ `ancdata.R` มีคำสั่งบรรทัดเดียวคือ

```
ancdata <- read.delim("ancdata.txt")
```

กรณีนี้ก็ต้องมีแฟ้มข้อมูลชื่อ `ancdata.txt` อยู่ในโฟลเดอร์ `data` นี้ด้วย และสุดท้ายควรสร้างแฟ้มที่ชื่อ `FILELIST` เขียนด้วยอักษรตัวใหญ่ทั้งหมดและไม่มีนามสกุล เพื่อบอกว่าแฟ้มข้อมูลทั้งหมดมีแฟ้มอะไรอยู่บ้าง เช่นใน package `epican` นั้น ชื่อความในแฟ้ม `FILELIST` มีดังนี้

```
cca.R
cca9.R
mccdata.R
irdata.R
ahcc.R
ccan.R
ancdata.R
evans.R
vct.R
```

7.3.4 โฟลเดอร์ย่อย *man*

เป็นโฟลเดอร์ที่บรรจุแฟ้มความช่วยเหลือ ที่เขียนในรูปแบบ R document (Rd) ดูรายละเอียดการเขียนแฟ้มชนิดนี้ใน `Writing R Extensions` และจะนำมากล่าวเป็นหัวข้อต่างหากในบทต่อไป

จากเพิ่มรูปแบบ Rd นี้ ในขณะที่ build package เพิ่มช่วยเหลือชนิดอื่นๆ จะถูกสร้างขึ้นมา เช่น ชนิด html, chhtml, dvi, และ text ทั้งนี้ขึ้นกับ platform ของคอมพิวเตอร์ที่ใช้ build นั้นเอง ส่วนเพิ่มชนิด pdf ที่อ่านได้ด้วยโปรแกรม Acrobat reader สามารถสร้างได้จากเพิ่ม dvi อีกต่อหนึ่ง

เราสามารถสร้างเพิ่มความช่วยเหลือเป็นภาษาใดก็ได้ (ไทย อังกฤษ หรือภาษาอื่น) แต่หัวข้อจะถูกสร้างเป็นภาษาอังกฤษโดยอัตโนมัติด้วยเงื่อนไขคำสั่งในรูปแบบ Rd นั้นเอง แต่การจัดการกับเพิ่มช่วยเหลือในภาษาอื่นมีรายละเอียดมากพอสมควร จะได้นำมากล่าวเป็นบทต่างหากต่อไป

ดูตัวอย่างเพิ่มความช่วยเหลือสำหรับคำสั่ง `clean()` ดังข้างล่างนี้

```
\name{clean}
\alias{clean}
\title{Clean R Memory}
\description{
  Clean R memory.
}

\usage{
  clean()
}

\details{
  Remove all objects from R memory.
}

\seealso{
\code{\link{rm}}
}

\examples{
\dontrun{clean()}
}
```

7.4 การ build package ใน R

โดยปกติการใช้งาน package ใน R สำหรับคอมพิวเตอร์ Linux และ Macintosh นั้น ผู้ใช้สามารถ install package จาก source file ที่บีบอัดข้อมูลในรูปแบบ tar.gz ได้โดยตรง นั่นคือผู้สร้าง package ไม่ต้อง build ให้ แต่ผู้ใช้เป็นคน build package จาก source file เองได้ แต่การทำงานบน Windows นั้นจะต่างออกไป คือผู้สร้างจะต้อง build ให้ แล้ว zip package นั้นเพื่อให้ผู้ใช้ unzip เอาไปใช้อีกต่อหนึ่ง ทั้งนี้เพราะกระบวนการ build บน Windows นั้นค่อนข้างยุ่งยาก เนื่องจากจะต้องทำงานเลียนแบบการทำงานบน Unix platform (Linux และ Macintosh เป็น Unix อยู่แล้ว จึงทำได้ทันที) ซึ่งจะกล่าวต่อไปในบทนี้

7.4.1 เครื่องมือที่ต้องใช้

มี web site ที่สนับสนุนการทำงานนี้โดยเฉพาะ คือ <http://www.murdoch-sutherland.com/Rtools/> ซึ่งมีแฟ้มเครื่องมือสำหรับการ build บริการให้ ผู้สนใจอาจเข้าไปศึกษา รายละเอียดเพิ่มเติมของการ build package บน Windows ที่ไม่ได้อธิบายไว้ในบทนี้ได้ที่ web site นี้ แต่ในขั้นต้นขอให้ download โปรแกรมเครื่องมือมาก่อนจาก <http://www.murdoch-sutherland.com/Rtools/tools.zip> จากนั้น unzip แฟ้มนี้ไปใส่ในโฟลเดอร์ ../R/rw20xx/bin

นอกจากแฟ้มเครื่องเหล่านี้แล้ว ยังต้อง download โปรแกรม ActivePerl จาก <http://www.activestate.com/Products/ActivePerl/Download.html> ซึ่งเป็นโปรแกรมที่มีอยู่เป็นปกติ บนเครื่อง Linux และ Macintosh อยู่แล้ว แต่ไม่มีอยู่บน Windows ActivePerl เป็นโปรแกรมที่ใช้ในการแปลคำสั่งแฟ้ม Rd เพื่อสร้างแฟ้มข้อความช่วยเหลือนั่นเอง เมื่อได้มาแล้ว ให้ติดตั้งตาม ขั้นตอนซึ่งจะใช้เวลาในการติดตั้งนานสักเล็กน้อย จากนั้น restart เครื่องคอมพิวเตอร์ โปรแกรม ActivePerl จะถูกเรียกทำงานอยู่เบื้องหลังทุกครั้งเมื่อเปิดเครื่องคอมพิวเตอร์

จากนั้นต้องตั้งค่าตัวแปร environment ที่ชื่อ TMPDIR ให้เป็นโฟลเดอร์ที่มีอยู่จริง เช่น C:\TEMP หรือ C:\WINDOWS\TEMP (หากยังไม่มีก็สร้างขึ้นใหม่) ทั้งนี้จะเป็นโฟลเดอร์ที่ ActivePerl จะใช้สำหรับใส่แฟ้มชั่วคราวที่สร้างขึ้นระหว่างการทำงานต่างๆ เพื่อจะได้ไม่ไปปะปน อยู่กับ โฟลเดอร์ทำงานตามปกติ สามารถทำได้โดยไปที่ Control Panel แล้วเลือก System กดปุ่ม Advanced ที่ด้านบน แล้วกดปุ่ม Environment Variables (ที่ด้านล่างเหนือปุ่ม OK) เมื่อปรากฏ หน้าต่างขึ้น ให้ดูในช่อง System variables ด้านล่าง แล้วเลื่อนลงไปจนพบชื่อ TMPDIR หากพบก็ ไม่ต้องทำอะไร แต่หากไม่พบให้กดปุ่ม New จะเกิดหน้าต่างเล็กๆ ขึ้น ช่อง Variable name ให้ใส่ TMPDIR และช่องด้านล่างให้ใส่ C:\TEMP หรือ C:\WINDOWS\TEMP ตามที่สร้างไว้หรือมีอยู่ แล้วคลิกตกลงข้างต้น

นอกจากนี้ยังมีอีกโปรแกรมหนึ่งที่ต้อง download มาคือ htmlhelp.exe ทั้งนี้เพื่อใช้สร้าง แฟ้มช่วยเหลือชนิด chhtml นั่นเอง โดย download จาก link ที่ให้ไว้ที่ <http://www.murdoch-sutherland.com/Rtools/> หรือ download ได้โดยตรงที่ <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/htmlhelp/html/hwmicrosofthtmlhelpdownloads.asp> หรือที่ <http://office.microsoft.com/en-gb/assistance/HA011362801033.aspx> ก็ได้ (link เหล่านี้อาจเปลี่ยนแปลงได้ ถ้าไม่สามารถเข้าได้ ให้ตรวจสอบ link ใหม่ที่ web site ของ Murdoch หรือใช้ search engine เช่น google ค้นหา link ใหม่โดยใช้คำหลักว่า htmlhelp.exe) ติดตั้งโปรแกรมตามขั้นตอน

7.4.2 วิธี *build R package*

สร้างโฟลเดอร์ package ตามที่ได้อธิบายไว้แล้วในเรื่อง “การสร้าง package ใน R” หรือตามรายละเอียดที่บรรยายใน “Writing R Extensions” คัดลอกโฟลเดอร์นี้ทั้งหมดไปใส่ในโฟลเดอร์ `../R/rw20xx/bin` ซึ่งจะเป็นโฟลเดอร์ทำงานในการ build บนเครื่อง Windows

ในที่นี้ขอแทรกเรื่องการ build package บน Linux และ Macintosh สักเล็กน้อย บน Linux นั้น หากจำ build หรือ install package ใดๆ ต้อง login เป็น root แล้วคัดลอกโฟลเดอร์ที่ต้องการทำงานไปไว้ที่ root home จากนั้นเรียก terminal ขึ้นมา หากต้องการ install ก็ติดตั้ง package เข้าไปใน R ทำโดยพิมพ์คำสั่งที่ prompt ดังนี้

```
R CMD INSTALL package_name
```

การทำงานนี้ใช้เมื่อแน่ใจว่าทุกอย่างที่สร้างขึ้นถูกต้องหมดแล้วทั้งสิ้น แต่ในการพัฒนา package จริงๆ นั้น จะต้องตรวจสอบก่อนว่าทุกอย่างที่เขียนใน package และเพิ่มความช่วยเหลือนั้นสามารถทำงานได้ถูกต้องหรือไม่ ซึ่ง R ได้มีคำสั่งที่ใช้ทดสอบการทำงานไว้ให้แล้ว โดยการใช้คำสั่ง

```
R CMD check package_name
```

เมื่อออกคำสั่งนี้ จะเกิดโฟลเดอร์ชื่อ `package_name.Rcheck` ขึ้น ภายในนั้นจะมีโฟลเดอร์ที่มีชื่อเดียวกับ package นั้นอยู่ หากไม่เกิดปัญหาใดๆ ในการ build เมื่อเข้าไปดูจะเห็นแฟ้มและโฟลเดอร์ย่อยต่างๆ คล้าย package ที่เป็นต้นกำเนิด แต่หากดูให้ลึกถึงข้อความในแฟ้มจะเป็นเปลี่ยนไป และมีแฟ้มเพิ่มขึ้น โดยจะมีแฟ้มชื่อ DESCRIPTION ซึ่งจะมีข้อความต่างไปจากต้นฉบับเดิมเล็กน้อย, CONTENTS, INDEX, และ NAMESPACE ถ้าในโฟลเดอร์เดิมมี NAMESPACE อยู่ด้วย และมีโฟลเดอร์ต่อไปนี้ data, help, html, latex, man, Meta, R, และ R-ex ในที่นี้จะไม่กล่าวถึงสิ่งที่บรรจุในโฟลเดอร์เหล่านี้ แต่โดยสรุปคือเป็นโครงสร้างและข้อมูลที่ใช้ทำงานจริงใน library folder ของ R นั้นเอง (ซึ่งแม้ว่าคำสั่ง R CMD INSTALL จะไม่แสดงแฟ้มและโฟลเดอร์เหล่านี้ให้เห็นที่โฟลเดอร์ root แต่คำสั่งจะติดตั้งแฟ้มและโฟลเดอร์เหล่านี้ลงใน library folder ทำงานของ R ทันที เพราะระบบปฏิบัติการ Linux ไม่ต้องการให้ผู้ใช้เข้าไปจัดการในส่วนนี้ แต่ระบบจะจัดการให้เอง)

บนระบบปฏิบัติการ Macintosh ก็เช่นเดียวกัน ให้คัดลอกโฟลเดอร์ package ไปไว้ที่ root (ชื่อ user ของผู้ใช้ ไม่ใช่ที่ไดเรกทอรีหลักของ hard disk เช่นผู้ใช้ชื่อ user ก็คัดลอกไปที่โฟลเดอร์ชื่อ user ไม่ใช่ที่ hard disk) จากนั้นเรียกโปรแกรม terminal จากโฟลเดอร์ Utilities ในโฟลเดอร์ Application นั่นคือขณะนี้ได้เข้า terminal mode เช่นเดียวกับ Linux แล้ว เมื่อพิมพ์ ls ที่ prompt ใน

terminal จะเห็นไฟล์เดอร์ package จากนั้นใช้คำสั่งเช่นเดียวกับที่กล่าวมาแล้วข้างต้นบน Linux นั่นเอง

ในการพิมพ์คำสั่งบน Linux หรือ Macintosh ขอให้พิมพ์ตัวพิมพ์เล็กหรือใหญ่ตามที่เขียนไว้ข้างต้นนี้ เพราะบนระบบปฏิบัติการตระกูล Unix นั้น อักษรตัวใหญ่ต่างจากอักษรตัวเล็ก

ที่กล่าวถึงระบบ Linux ก่อนเช่นนี้ก็เนื่องจากการทำงานบน Windows จะทำงานเลียนแบบระบบ Linux นั่นเอง ซึ่งในขั้นแรกจะต้องเข้า DOS mode โดยที่ Start Menu ให้เลือก All Programs → Accessories → Command Prompt แล้วเปลี่ยน directory โดยใช้คำสั่ง cd ไปจนถึง ../R/rw20xx/bin ที่ได้คัดลอกไฟล์เดอร์ package ไปไว้ก่อนหน้านี้แล้ว

จากนั้นที่ prompt พิมพ์คำสั่ง

```
Rcmd check package_name
```

ซึ่งเป็นการทำงานดังที่ได้กล่าวไว้ข้างต้นเช่นเดียวกับบนระบบ Linux นั่นเอง แต่สังเกตว่า R CMD คราวนี้จะพิมพ์ติดกัน และสามารถพิมพ์ตัวเล็กหรือตัวใหญ่ก็ได้

ในขั้นตอนการ build ด้วย check นั้น จะเกิดไฟล์เดอร์ชื่อ *package_name.Rcheck* เช่นเดียวกับที่กล่าวข้างต้น โดยไฟล์เดอร์นี้จะอยู่ที่ ../R/rw20xx/bin นั่นเอง แต่จะพบว่ามีการรายงานความผิดพลาดเกิดขึ้น ในขณะที่คำสั่งได้สร้างไฟล์เดอร์ชื่อ chm ในไฟล์เดอร์ package อันเก่าที่เป็นต้นฉบับ ในไฟล์เดอร์นี้จะเห็นแฟ้มความช่วยเหลือที่มีนามสกุล html และจะมีหนึ่งแฟ้มที่มีชื่อ *package_name.hhp*

เมื่อถึงขั้นตอนนี้ ให้เปิดโปรแกรม HTML Help Workshop จาก desktop alias หรือจาก Start Menu ก็ได้ แล้วเลือกที่ File → Open แล้วเปิดแฟ้มที่มีชื่อ *package_name.hhp* นั้น จากนั้นกลับไป File → Compile... เพื่อสร้างแฟ้มความช่วยเหลือที่มีชื่อเป็น *package_name.chm* ไว้ในไฟล์เดอร์ chm นี้ เมื่อเสร็จแล้วปิดโปรแกรม HTML Help Workshop ได้

ที่นี้ก็กลับมา build package อีกครั้งด้วยคำสั่ง

```
Rcmd check package_name
```

ถ้าคำสั่งและแฟ้มความช่วยเหลือทั้งหมดถูกต้อง คราวนี้คำสั่งจะ build ทุกอย่างได้โดยไม่มีข้อผิดพลาด และจะมีไฟล์เดอร์ย่อยชื่อ chhtml เพิ่มขึ้นมาจากการ build บน Linux หรือ Macintosh

ในระหว่างการ build หากมีข้อผิดพลาดอย่างใดเกิดขึ้น ขอให้อ่านข้อความแจ้งความผิดพลาดให้ละเอียด เพื่อกลับไปแก้ไขข้อผิดพลาดนั้น แล้วกลับมา build ใหม่อีกครั้ง แต่อย่าลืมว่าคราวนี้มีไฟล์เดอร์ package ถึงสามไฟล์เดอร์ คือไฟล์เดอร์ดั้งเดิมที่สร้างขึ้นในการพัฒนาคำสั่งไฟล์เดอร์ package ที่คัดลอกมาไว้ที่ /bin และไฟล์เดอร์ package ใน *package_name.Rcheck* ไม่ต้องไปสนใจไฟล์เดอร์ package ใน .Rcheck แต่อย่างใด แต่ทุกครั้งที่ได้แก้ไขต้นฉบับคำสั่งหรือแฟ้มความช่วยเหลือ ควรแก้ไขใน /bin และเมื่อทุกอย่างถูกต้องคือ build ผ่านหมดแล้ว ให้สำเนา

ไฟล์เดอรันี้ทั้งหมดกลับไปไว้ที่เดิมที่พัฒนา package นั้นด้วย มิฉะนั้นจะสับสนในภายหลังได้ เพราะอาจมีบางคำสั่งใน package ดันฉบับกับที่ build แล้วแตกต่างกันได้ และนอกจากนี้ยังแนะนำให้สำเนาไฟล์เดอร `package_name.Rcheck` เก็บไว้ทั้งไฟล์เดอร อย่าสำเนาเก็บแต่เฉพาะไฟล์เดอร package ภายในมาเก็บไว้ เพราะอย่างที่บอกแล้วว่าไฟล์เดอร package ที่สร้างขึ้นใน `.Rcheck` นั้นไม่เหมือนกับ package ดันฉบับ อาจทำให้สับสนได้ว่า package ที่ build แล้วนี้เป็นดันฉบับได้ แล้วไปลบดันฉบับจริงทิ้งเสีย การฟื้นดันฉบับทั้งหมดกลับมานั้นทำได้ยากมาก

อีกประการหนึ่ง หากมีการแก้ไขเพิ่มความช่วยเหลือชนิด `.Rd` ต้องลบไฟล์เดอร `chm` ในไฟล์เดอร package ที่เป็นดันฉบับใน `/bin` ด้วยเสมอ ที่จริงเพิ่มที่สำคัญที่สุดในส่วนนั้นคือ `package_name.chm` ซึ่งหากในระหว่างการ build นั้น โปรแกรม `Rcmd` พบเพิ่มนี้อยู่ในไฟล์เดอร `chm` คำสั่งจะห้ามการแก้ไขทั้งเพิ่มชนิด `.htm` และ `.chm` ไปเลย ซึ่งจะทำให้ได้เพิ่มความช่วยเหลือเท่าที่ยังไม่ได้แก้ไขแต่อย่างใด ดังนั้น หากมีการแก้ไขเพิ่มความช่วยเหลือชนิด `.Rd` ให้ลบไฟล์เดอร `chm` ทิ้งเสียด้วย และจะต้องเรียกคำสั่งในการ build สองครั้ง และต้องเรียกโปรแกรม `HTML Help Workshop` ใหม่ทุกครั้ง แม้ว่าจะเป็นเวลาเสียเวลา แต่ถ้าหากไม่ทำ เพิ่มความช่วยเหลือที่ได้จะไม่ถูกต้อง

เมื่อการ build เรียบร้อยแล้ว ให้เข้าไปในไฟล์เดอร `package_name.Rcheck` แล้ว zip ไฟล์เดอร package ในนั้นด้วยโปรแกรม `winzip` หรือ `7-Zip` ก็ได้ แล้วเพิ่ม `package_name.zip` ที่ได้นี้จะเป็นเพิ่มที่สามารถแจกจ่ายให้ผู้ใช้งาน R บนระบบ Windows ได้นั่นเอง (แต่ผู้ใช้บนระบบ Linux และ Macintosh จะใช้งานเพิ่มนี้ไม่ได้)

สำหรับการ build บน Linux และ Macintosh นั้น เมื่อ check ผ่านหมดทุกขั้นตอนดังนี้แล้ว จะต้องเรียกคำสั่งต่อไปในที่ prompt

```
R CMD build package_name
```

เป็นการสร้างเพิ่มชื่อ `package_name.tar.gz` ขึ้นนั่นเอง บน Windows นั้นก็สามารถสร้างเพิ่มนี้ได้โดยออกคำสั่งต่อไปในที่ DOS prompt

```
Rcmd build package_name
```

ในการแจกจ่ายเพิ่ม package สำหรับระบบ Linux และ Macintosh นั้น จะแจกจ่ายเป็นเพิ่ม `package_name.tar.gz` เมื่อผู้ใช้ต้องการจะติดตั้ง ให้ไปที่ terminal แล้วเรียกคำสั่ง `R CMD INSTALL` เพื่อติดตั้ง

ส่วนบนระบบ Windows ให้แจกจ่าย package เป็นเพิ่ม `package_name.zip` ผู้ใช้จะติดตั้งโดยเข้า R console ก่อน แล้วเลือกเมนู Packages → Install package(s) from

7.5 การทำเพิ่มความช่วยเหลือเป็นภาษาไทย

ในการทำเพิ่มความช่วยเหลือนั้น ดังที่ได้กล่าวไว้ในเรื่องการ build package แล้วว่าจะต้องเขียนในรูปแบบ R document (Rd) การที่จะทำเพิ่มความช่วยเหลือเป็นภาษาไทยก็เช่นเดียวกัน ต้องเริ่มต้นด้วยการสร้างเพิ่มความช่วยเหลือในรูปแบบ Rd นั้นเอง ซึ่งมี tag คำสั่งต่างๆ ดังได้อธิบายไว้ในเรื่อง “การเขียนเพิ่มความช่วยเหลือในรูปแบบ R document” แล้ว

ในการเขียนเพิ่มความช่วยเหลือในรูปแบบ Rd ให้เป็นภาษาไทยนั้นก็เขียนเช่นเดียวกับที่กล่าวไว้แล้วในบทที่กล่าวถึงข้างต้น โดยยังคง tag คำสั่งไว้เช่นเดียวกับที่ได้อธิบายไว้แล้ว เพียงแต่เขียนข้อความภายใน tag คำสั่ง บางอันเป็นภาษาไทยเท่านั้น ขอทบทวน tag คำสั่งที่สำคัญดังนี้

```
\name{name}
\alias{topic}
\title{Title}
\description{...}
\usage{fun(arg1, arg2, ...)}
\arguments{...}
\details{...}
\value{...}
\references{...}
\note{...}
\author{...}
\seealso{...}
\examples{...}
\keyword{key}
```

tag คำสั่ง \name และ \alias นั้นให้เขียนเป็นภาษาอังกฤษเสมอ เนื่องจากจะเป็นส่วนของชื่อคำสั่งนั้นๆ ที่ R ใช้อ้างอิงและค้นหาคำสั่งในระบบ tag คำสั่ง \title และ \description สามารถเขียนเป็นภาษาไทยได้ ในส่วนของ \usage นั้นจะต้องเขียนเป็นภาษาอังกฤษ เพราะเป็นส่วนที่แสดงวิธีการเรียกใช้ฟังก์ชัน ในส่วนของ \arguments นั้น ในส่วนของชื่อ argument จะต้องเป็นภาษาอังกฤษตามที่ระบุไว้ใน \usage นั้นเอง แต่ในส่วนของคำอธิบายสามารถเขียนเป็นภาษาไทยได้

ส่วน tag คำสั่ง \details, \value, \references, \note, และ \author สามารถเขียนเป็นภาษาไทยได้ตามความเหมาะสม ส่วน tag คำสั่ง \seealso, \examples, และ \keyword จะต้องเขียนเป็นภาษาอังกฤษ ยกเว้นในส่วนของ comment ใน \examples อาจเขียนเป็นภาษาไทยได้

ส่วนที่สำคัญที่จะกล่าวโดยละเอียดในบทนี้คือขั้นตอนการ build ให้ได้เพิ่มความช่วยเหลือภาษาไทยในรูปแบบ html และ chtml ได้แสดงเป็นภาษาไทยได้อย่างถูกต้อง ซึ่งหากทำไม่ถูกต้องขั้นตอนแล้วอาจทำให้การแสดงผลภาษาไทยไม่ถูกต้องได้

ดังที่ได้กล่าวในบท “การสร้าง package ใน R” แล้วนั้น โฟลเดอร์ที่บรรจุเพิ่มความช่วยเหลือในรูปแบบ Rd คือโฟลเดอร์ man นั้นเอง หากจะมีบางกรณีที่ต้องการเรียกความช่วยเหลือในรูปแบบ text ด้วย ให้เขียนเพิ่มความช่วยเหลือเป็นสองโฟลเดอร์ โดยโฟลเดอร์หนึ่งเขียนเป็น

ภาษาอังกฤษ และอีกโพลเดอร์หนึ่งเขียนเป็นภาษาไทย ตามคำแนะนำข้างต้น อย่างไรก็ตามเพื่อความสะดวกอาจเขียนเป็นภาษาไทยอย่างเดียวก็ได้ แล้วกำหนดค่าปกติของการเรียกความช่วยเหลือให้เรียกรูปแบบ html หรือ chtml (เฉพาะบน Windows) ซึ่งบนระบบปฏิบัติการ Macintosh นั้นจะไม่มีปัญหาแต่อย่างใด เนื่องจาก R บน Macintosh จะเรียกความช่วยเหลือในรูปแบบ html เสมออยู่แล้ว ส่วนบน Linux นั้นอาจเรียกคำสั่ง help ด้วย option `html=TRUE` เสมอ และบน Windows นั้น สามารถแก้ไขเพิ่ม Rprofile ในโพลเดอร์ `../R/rw20xx/etc` โดยเอาเครื่องหมาย # หน้าบรรทัด `options(chmhelp=TRUE)` หรือ `options(htmlhelp=TRUE)` ออกเสีย ส่วนจะเลือกเอาเครื่องหมายหน้าบรรทัดไหนออกนั้น ก็ขึ้นอยู่กับว่าต้องการให้แสดงความช่วยเหลือเป็นแบบ html หรือ chtml นั่นเอง

ในขั้นตอนของการ build package นั้น ขอแยกกล่าวเป็นสองกรณี คือการ build บน Macintosh บน Linux และการ build บน Windows

บน Macintosh นั้น ถ้าต้องการความช่วยเหลือภาษาไทย ก็สร้างเพิ่มความช่วยเหลือในรูปแบบ Rd เป็นภาษาไทยอย่างเดียวได้เลย แล้ว build package ตามที่อธิบายไว้ในบท “การ build package ใน R” ได้เลย

บน Linux นั้น เนื่องจากการ build เป็นภาระของผู้ใช้เอง ดังนั้นอาจต้องเตรียมเพิ่ม .tar.gz ไว้ทั้งที่มีเพิ่มความช่วยเหลือภาษาไทยและอังกฤษ ให้ผู้ใช้เลือกที่จะ build แบบใดแบบหนึ่งต่อไปนี้ด้วยตัวเอง

หากไม่ต้องการใช้ความช่วยเหลือแบบ text ก็สามารถสร้างเพิ่มความช่วยเหลือรูปแบบ Rd เป็นภาษาไทยอย่างเดียวได้เลย แต่หากเรียกคำสั่ง help โดยไม่ระบุตัวเลือก `html=TRUE` แล้ว จะได้ข้อความช่วยเหลือเป็น text ที่อ่านไม่รู้เรื่อง เนื่องจากความช่วยเหลือแบบ text นั้นจะแสดงบน terminal console ซึ่งแสดงภาษาไทยไม่ได้

ดังนั้นถ้าต้องการให้ข้อความช่วยเหลือรูปแบบ text ที่อ่านออกเป็นภาษาอังกฤษด้วย ก็จะต้อง build package สองครั้ง โดยครั้งแรกแนะนำให้ build โดยใช้เพิ่ม .tar.gz ที่มีเพิ่มความช่วยเหลือเป็นภาษาไทยก่อน โดยเมื่อขยายเพิ่มออกมาแล้ว ให้ build ด้วยคำสั่ง

```
R CMD INSTALL package_name
```

จากนั้นจึงขยายเพิ่ม .tar.gz ที่มีเพิ่มความช่วยเหลือภาษาอังกฤษออกมา แล้ว build ด้วยคำสั่ง

```
R CMD check package_name
```

อีกครั้ง คราวนี้จะได้โพลเดอร์ `package_name.Rcheck` ขึ้นที่ root (ในการ build package จะต้อง login เป็น root เสมอ) จากนั้นเข้าไปในโพลเดอร์ `package_name.Rcheck/package_name` จะเห็นโพลเดอร์ชื่อ help อยู่ ให้คัดลอกโพลเดอร์นี้ไปไว้ในโพลเดอร์ `R_HOME/library/package_name`

หับโฟลเดอร์ help ที่มีอยู่เดิม โฟลเดอร์ *R_HOME* อาจแตกต่างกันตามแต่ละระบบ Linux ที่ใช้ ทางที่ดี ให้ใช้วิธีค้นหาชื่อของ package ในโฟลเดอร์ / (ไม่ใช่ /root) จะเห็นว่าการทำค่อนข้างยุ่งยากพอสมควร

ส่วนบนระบบ Windows นั้น ผู้พัฒนา package จะต้องแจกจ่ายเพิ่ม package ในรูปแบบ zip จึงต้อง build package เสียก่อน นั่นคือผู้พัฒนาสามารถทำเพิ่ม *package_name.zip* ที่มีโฟลเดอร์ความช่วยเหลือ html และ chhtml เป็นภาษาไทย ขณะที่เพิ่มในโฟลเดอร์ help เป็นภาษาอังกฤษได้เลย

ขั้นตอนการ build ให้ build package ด้วยเพิ่มความช่วยเหลือภาษาอังกฤษ โดยใช้คำสั่ง

```
Rcmd check package_name
```

ดังที่ได้กล่าวไว้ในบท “การ build package ใน R” ซึ่งจะต้องออกคำสั่งนี้สองครั้งดังได้กล่าวในบทความ เมื่อเสร็จแล้วจะได้โฟลเดอร์ *package_name.Rcheck* ซึ่งภายในมีโฟลเดอร์ *package_name* อยู่ ให้ zip โฟลเดอร์นี้เป็น *package_name.zip* ที่มีเพิ่มความช่วยเหลือเป็นภาษาอังกฤษล้วน พร้อมแจกจ่ายให้กับชาวต่างประเทศได้ จากนั้นเปลี่ยนชื่อโฟลเดอร์ *package_name.Rcheck* นี้ให้เป็น *package_name.Rcheck.en* เพื่อให้รู้ว่าเป็นโฟลเดอร์ที่มีเพิ่มความช่วยเหลือเป็นภาษาอังกฤษ

จากนั้น build package ที่มีเพิ่มความช่วยเหลือเป็นภาษาไทย ในการออกคำสั่ง

```
Rcmd check package_name
```

ในรอบแรกนั้น ก่อนเรียกใช้โปรแกรม HTML Help Workshop จะต้องแก้ไขข้อความในแฟ้มทุกแฟ้มในโฟลเดอร์ chm ที่คำสั่งได้สร้างขึ้นใหม่เสียก่อน

เมื่อเปิดโฟลเดอร์ chm จะเห็นแฟ้มต่างๆ ที่มีนามสกุลเป็น .html โดยจะต้องมีแฟ้มแรกเป็น 00Index.html เสมอ ส่วนแฟ้มอื่นๆ จะมีชื่อตามแฟ้ม Rd ที่สร้างไว้เดิมนั่นเอง ตามด้วยนามสกุล .html ให้เปิดแฟ้ม 00Index.html ด้วยโปรแกรม text editor ที่มีอยู่ เช่น Notepad หรือ Notepad++ หรือ Crimson Editor หรือโปรแกรมอื่นก็ได้ ไม่ควรเปิดด้วย Word เพราะอาจเผลอบันทึกแฟ้มเป็น .doc ซึ่งจะทำให้การทำงานต่อไปไม่ถูกต้องได้

ในแฟ้ม 00Index.html นั้น สามบรรทัดแรกจะเป็นดังนี้

```
<html><head><title>Functions for Epidemiology</title>
<link rel="stylesheet" type="text/css" href="Rchm.css">
</head><body>
```

ข้อความใน tag <title>...</title> จะเปลี่ยนไปตามที่ระบุคำอธิบาย package นั้นๆ

ให้แทรกบรรทัดต่อไปนี้เป็นบรรทัดที่สองหรือสามก็ได้

```
<meta http-equiv="Content-Type" content="text/html; charset=tis-620">
```

จะได้ดังนี้

```
<html><head><title>Functions for Epidemiology</title>
<link rel="stylesheet" type="text/css" href="Rchm.css">
<meta http-equiv="Content-Type" content="text/html; charset=tis-620">
</head><body>
```

ทั้งนี้เพื่อระบุให้การ compile เพิ่มความช่วยเหลือที่เป็น chtml นั้นให้ใช้ภาษาไทยชุดอักษร tis-620 แก่เซ่นนี้กับทุกแฟ้มในโฟลเดอร์ chm นี้

จากนั้นเรียกโปรแกรม HTML Help Workshop เพื่อ compile สร้างเพิ่มความช่วยเหลือ chtml แล้ว build package ใหม่อีกครั้ง คราวนี้จะได้โฟลเดอร์ *package_name.Rcheck* ที่มีเพิ่มความช่วยเหลือเป็นภาษาไทย แต่สามบรรทัดแรกในแฟ้มความช่วยเหลือรูปแบบ html จะกลับไปเป็นเหมือนเดิมอีก และเพิ่มความช่วยเหลือในรูปแบบ text ที่ยังเป็นตัวอักษรไทยที่อ่านใน R console ไม่ได้

ขั้นตอนต่อไปเป็นการแก้ไขให้แฟ้ม html บังคับให้โปรแกรม browser ใช้ชุดตัวอักษรภาษาไทย tis-620 ทำได้โดยเข้าไปในโฟลเดอร์ html ในโฟลเดอร์ *package_name.Rcheck* แล้วเปิดทุกแฟ้มมาแก้ส่วน head เช่นเดียวกับที่กล่าวข้างต้นอีกครั้งหนึ่ง เช่นในแฟ้ม 00Index.html จะมีสามบรรทัดแรกเป็น

```
<html><head><title>R: Functions for Epidemiology</title>
<link rel="stylesheet" type="text/css" href="../../../Rth.css">
</head><body>
```

ก็แทรกบรรทัดเช่นเดิม จะได้ใหม่เป็นดังนี้

```
<html><head><title>R: Functions for Epidemiology</title>
<link rel="stylesheet" type="text/css" href="../../../Rth.css">
<meta http-equiv="Content-Type" content="text/html; charset=tis-620">
</head><body>
```

หมายเหตุ

ขอให้สังเกตว่าบรรทัดที่ 2 นั้นไม่เหมือนเดิมทีเดียว คือจะเรียกแฟ้ม cascade style sheet ที่ต่างกัน

เมื่อแก้ไขเสร็จแล้ว ให้กลับไปโฟลเดอร์ *package_name.Rcheck.en* ที่ได้สร้างไว้ก่อนแล้ว คัดลอกโฟลเดอร์ help ที่เป็นความช่วยเหลือภาษาอังกฤษนี้มาทับโฟลเดอร์ help ในโฟลเดอร์ *package_name* ที่อยู่ภายในโฟลเดอร์ *package_name.Rcheck* ที่เป็นภาษาไทยเสีย เพื่อให้เพิ่ม

ช่วยเหลือในรูปแบบ text อ่านได้เป็นภาษาอังกฤษ (อาจลบไฟล์เดอร์นี้ไปเลยก็ได้ แต่ต้องแจ้งให้ผู้ใช้ทราบว่าในการเรียกความช่วยเหลือสำหรับ package นี้ ต้องเรียกเป็น chhtml หรือ html เท่านั้น)

จากนั้น zip ไฟล์เดอร์ package_name นั้นดังที่ได้อธิบายไว้ข้างต้น จะได้เพิ่ม *package_name.zip* ที่มีเพิ่มความช่วยเหลือเป็นภาษาไทยพร้อมแจกจ่ายแก่ผู้ใช้ได้