ภาคผนวก ข.3

“คู่มือการใช้โปรแกรม R สำหรับการวิเคราะห์ข้อมูลทางสถิติ
และการเขียนฟังก์ชันด้วยโปรแกรม R”

เป็นส่วนหนึ่งของโครงการ

การพัฒนาการใช้โปรแกรม R ในการวิเคราะห์ข้อมูล

R Program Development for Research Utilization

โดย

หัชชา ศรีปลั่ง และคณะ

เดือน ปี ที่เสร็จโครงการตามสัญญา

ตุลาคม 2549

ต้นฉบับคู่มือ

“คู่มือการใช้โปรแกรม R สำหรับการวิเคราะห์ข้อมูลทางสถิติ และการเขียนฟังก์ชันด้วยโปรแกรม R”

ผู้แต่ง
หัชชา ศรีปลั่ง
อภิรดี แซ่ลิ่ม

คู่มือการใช้โปรแกรม R

สำหรับการวิเคราะห์ข้อมูลทางสถิติและ
การเขียนฟังก์ชันด้วยโปรแกรม R

หัชชา ศรีปลั่ง

อภิรดี แซ่ลิ่ม

เอ็ดเวิร์ด แม็คเนล

กิตติศักดิ์ ชูมาลี

นราทิพย์ จั่นสกุล

กัลยา นิติเรืองจรัส

คณะแพทยศาสตร์ และคณะวิทยาศาสตร์

มหาวิทยาลัยสงขลานครินทร์

คำนำ

โปรแกรม **R** เป็นโปรแกรมทางเลือกหนึ่งที่ใช้ในการวิเคราะห์ข้อมูลทางสถิติ ที่แจกจ่ายให้ฟรี โดยผู้ใช้สามารถ download ได้จาก web site ของ **R** ได้โดยตรง มหาวิทยาลัยเป็นสถาบันทางการศึกษาระดับสูง จึงควรเป็นแบบอย่างที่ดีแก่สังคม ในการใช้โปรแกรมทางคอมพิวเตอร์ที่ถูกกฎหมาย แต่ด้วยปัญหาที่มีงบประมาณจำกัด ทางเลือกหนึ่งคือ การหันไปใช้โปแกรมที่มีการแจกจ่ายให้ฟรี ไม่ต้องเสียค่าลิขสิทธิ์ ซึ่งได้มีการร่วมมือจากองค์กรต่าง ๆ ทั่วโลกที่สร้างโปรแกรมวิเคราะห์ข้อมูลทางสถิติ และแจกจ่ายแก่ผู้สนใจฟรี แต่โปรแกรมฟรีเหล่านี้ ค่อนข้างใช้ยาก ผู้ใช้ไม่คุ้นเคย จึงไม่ได้รับความนิยม ทั้ง ๆ ที่ประสิทธิภาพในการทำงานของโปรแกรมเหล่านี้ไม่ได้ด้อยไปกว่าโปรแกรมทางการค้าทั่วไป หรือบางครั้งมีประสิทธิภาพดีกว่าด้วยซ้ำ หน่วยระบาดวิทยา คณะแพทยศาสตร์ และภาควิชาคณิตศาสตร์ คณะวิทยาศาสตร์ มหาวิทยาลัยสงขานครินทร์ ได้เล็งเห็นถึงปัญหาเหล่านี้ จึงได้พัฒนาโปรแกรม **R** ให้ง่ายต่อการใช้งาน และส่งเสริมให้คนได้รู้จัก และหันมาใช้โปรแกรมเหล่านี้ในการวิเคราะห์ข้อมูลเพิ่มขึ้น

คณะผู้วิจัย

พฤศจิกายน 2547

สารบัญ

สารบัญรูป

บทที่ 1

โปรแกรม R

1.1 แนะนำโปรแกรม R

โปรแกรมสำเร็จรูป **R** เป็น Open Source Software ที่นำเสนอให้ฟรี สามารถดัดแปลงได้ตามต้องการอย่างอิสระ แจกจ่ายได้ แต่ห้ามจำหน่าย โดยหวังผลกำไร ผู้เริ่มต้นเขียนโปแกรมนี้ คือ Robert Gentleman และ Ross Ihaka จากภาควิชาสถิติ มหาวิทยาลัยโอ๊คแลนด์ ประเทศนิวซีแลนด์ จากนั้นตั้งแต่กลางปี พ.ศ. 2540 เป็นต้นมาจึงเกิดเป็นทีมผู้พัฒนาโปรแกรม **R** (The R Development Core Team) ซึ่งกลุ่มผู้พัฒนาโปรแกรมนี้เป็นกลุ่มเดียวกับผู้พัฒนาโปรแกรมสำเร็จรูป S Plus ที่ได้รับการยกย่องว่าเป็นโปรแกรมดีเด่น มีประสิทธิภาพในการคำนวณทางสถิติสูง แต่มีลิขสิทธ์และราคาแพง โปรแกรม **R** มีมูลนิธิ (Software foundation) เป็นผู้สนับสนุนด้านวิชาการทั้งจากสหรัฐอเมริกา, ออสเตรเลีย และญี่ปุ่น เป็นต้น โปรแกรมนี้จึงเป็นทางเลือกหนึ่งที่สามารถนำมาพัฒนาเพื่อใช้ในด้านการเรียนการสอนได้ นอกเหนือจากการเป็นซอฟต์แวร์ฟรี โปรแกรม **R** เป็นโอกาสอันดีสำหรับนักวิชาการในประเทศไทยที่จะนำไปใช้ในด้านต่าง ๆ เช่น **R** จะช่วยในการเรียนการสอนด้านสถิติให้เข้าใจเนื้อหาทางทฤษฎีได้ลึกซึ้ง เปิดโอกาสในนักวิจัยสามารถเขียนฟังก์ชันและกราฟใหม่ ๆ สำหรับใช้งานเฉพาะด้านของตน และที่สำคัญคือสามารถเข้าร่วมเรียนรู้กับนานาชาติ โดยการร่วมเขียน module ตามโจทย์และศักยภาพที่เรามีอยู่ วิสัยทัศน์ของการพัฒนาและพึ่งตนเองพร้อม ๆ กับประสานงานกับกลุ่มนานาชาติในการทำงานทางวิชาการที่ไม่ผูกติดกับผลประโยชน์ทางการค้า เพื่อพัฒนาสปิริตการสร้างสมบัติวิชาการที่เป็นสาธารณะของโลก ซึ่งประเทศไทยยังขาดการเรียนรู้ด้านนี้อยู่มาก จึงเป็นสิ่งที่ควรพัฒนาขึ้น

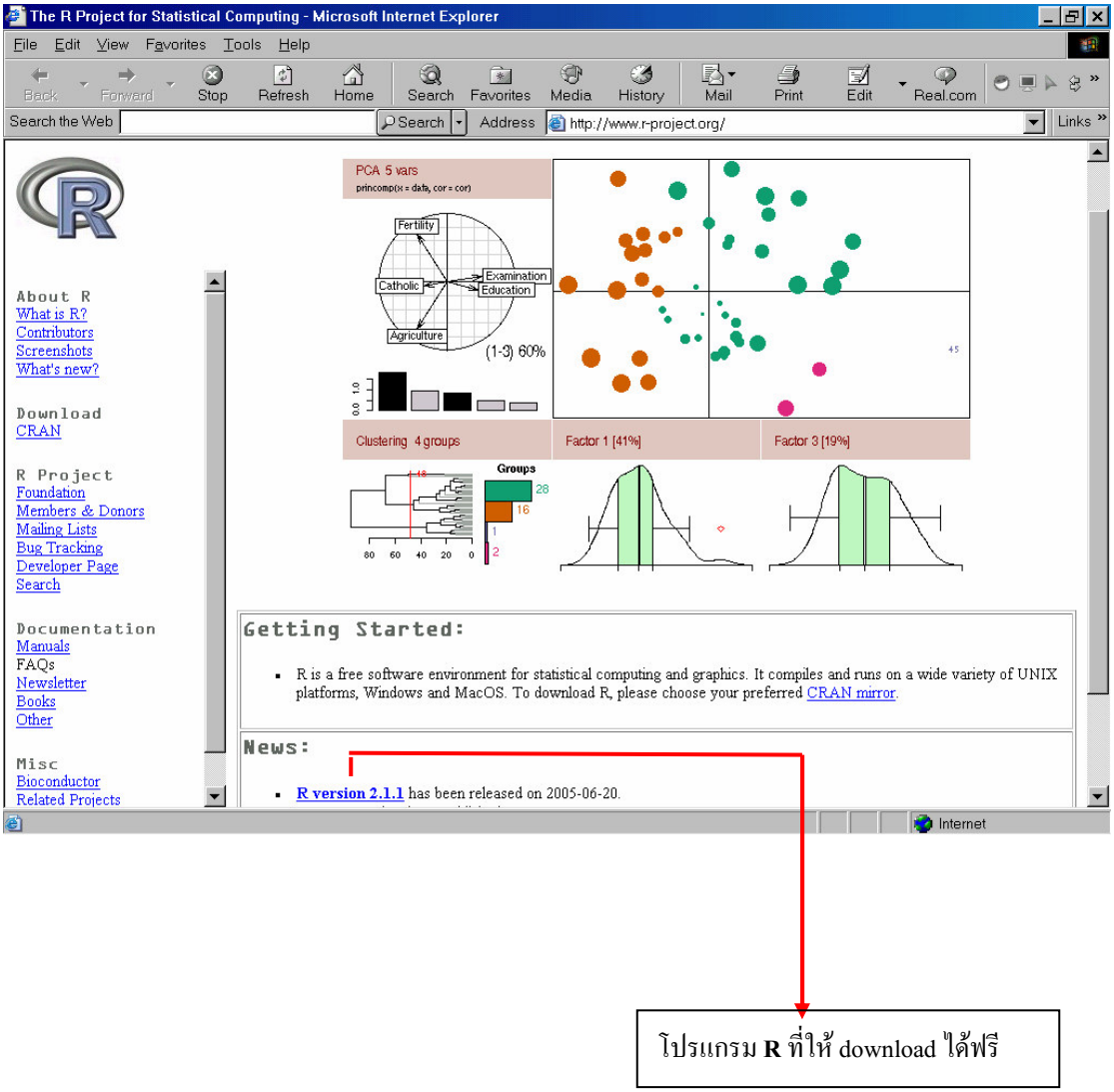
1.2 เหตุผลในการใช้โปรแกรม R

การประกาศนโยบายรับรองทรัพย์สินทางปัญญาเป็นนโยบายหนึ่งของรัฐบาล โดยเฉพาะอย่างยิ่งซอฟต์แวร์ทางคอมพิวเตอร์ โดยมีประกาศให้หน่วยราชการต่าง ๆ ใช้ซอฟต์แวร์ที่ถูกกฎหมายเท่านั้น อย่างไรก็ตาม ในทางปฏิบัติ หน่วยราชการต่าง ๆ มีงบประมาณในการจัดซื้อซอฟต์แวร์จำกัด หากจัดซื้อจริงคาดว่าค่าใช้จ่ายต่างๆ จะสูงขึ้นอย่างมาก โปรแกรมทางคอมพิวเตอร์ที่ใช้ในการวิเคราะห์ข้อมูลทางสถิติส่วนใหญ่เป็นโปรแกรมที่มีลิขสิทธิ์ ในปัจจุบันการวิเคราะห์ข้อมูลทางสถิติ ต้องอาศัยโปรแกรมสำเร็จรูป (software) ที่สามารถรองรับข้อมูลที่มีขนาดใหญ่และระเบียบวิธีทางสถิติที่สลับซับซ้อน อย่างไรก็ตามซอฟต์แวร์ที่มีประสิทธิภาพดังกล่าว ถูกผลิตขึ้นเพื่อการค้า และมีราคาแพง ประกอบกับหน่วยงานต่าง ๆ ไม่มีงบประมาณสำหรับซื้อซอฟต์แวร์

เหล่านั้น ทำให้ต้องใช้ซอฟต์แวร์อย่างผิดกฎหมาย มหาวิทยาลัยเป็นสถาบันการเรียนการสอน จึงควรเป็นแบบอย่างที่ดีแก่สังคมภายนอกในการใช้ซอฟต์แวร์อย่างถูกกฎหมาย ดังนั้นการพัฒนาการใช้ซอฟต์แวร์ที่ไม่ใช่เพื่อการค้าสำหรับวิเคราะห์ข้อมูลให้ชำนาญมากขึ้น จึงเป็นสิ่งจำเป็นในการส่งเสริมให้มีการใช้อย่างแพร่หลายขึ้น เพื่อให้หลุดพ้นจากปัญหาเชิงการค้าให้ได้มากที่สุด

1.3 การ download โปรแกรม R

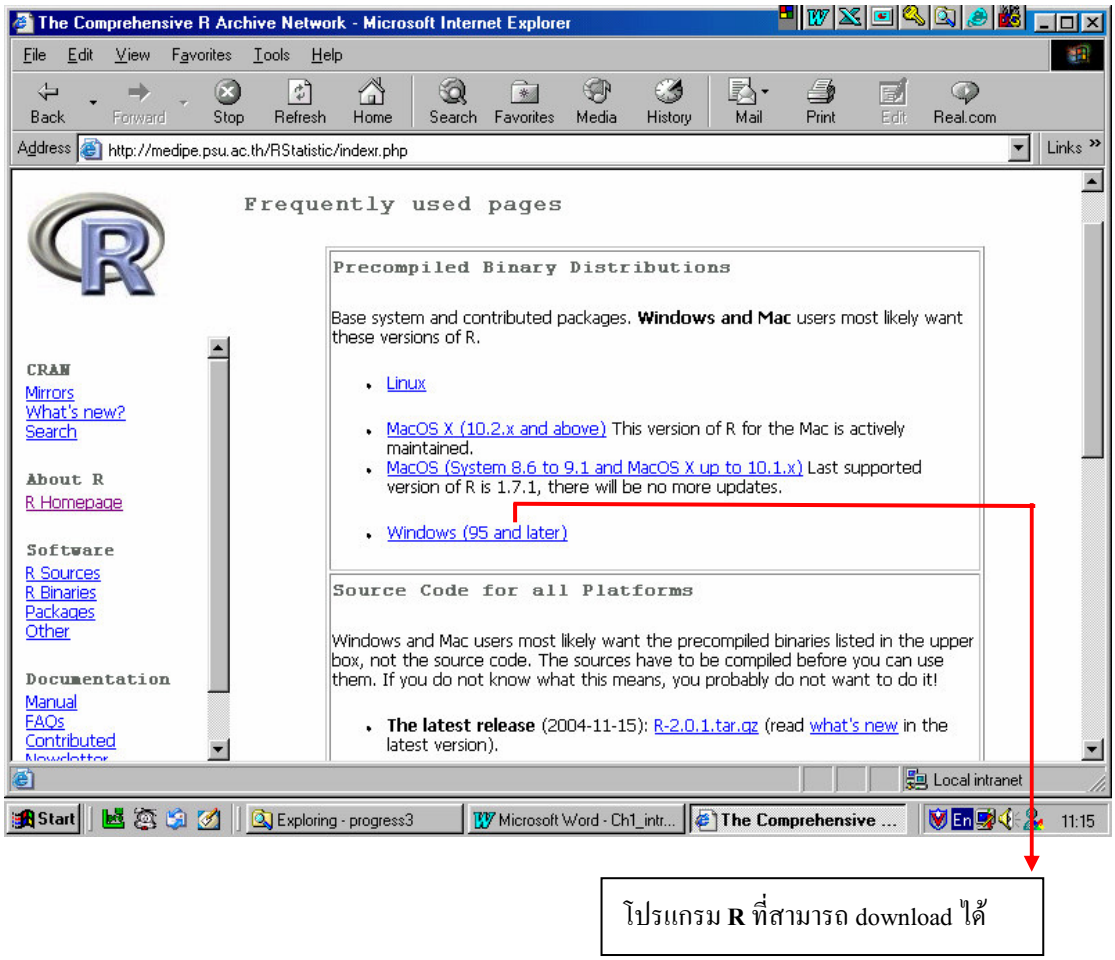
โปรแกรม **R** สามารถเข้าไป download ได้ที่ web site ของ **R** โดยตรงคือ http:// www.r-project.org ดังรูปที่ 1.1 ปัจจุบันโปรแกรม **R** ที่ download จะอยู่ใน version 2.4.0



รูปที่ 1.1 Web page R

หมายเหตุ โปรแกรม **R** version 2.4.0 ออกเมื่อเดือนตุลาคม 2549

นอกจากนี้ หน่วยระบาดวิทยาได้เก็บโปรแกรมดังกล่าวไว้ที่ web site ของหน่วยเอง คือ http://medipe.psu.ac.th/Rstatisitc/ndexr.php ดังรูปที่ 1.2



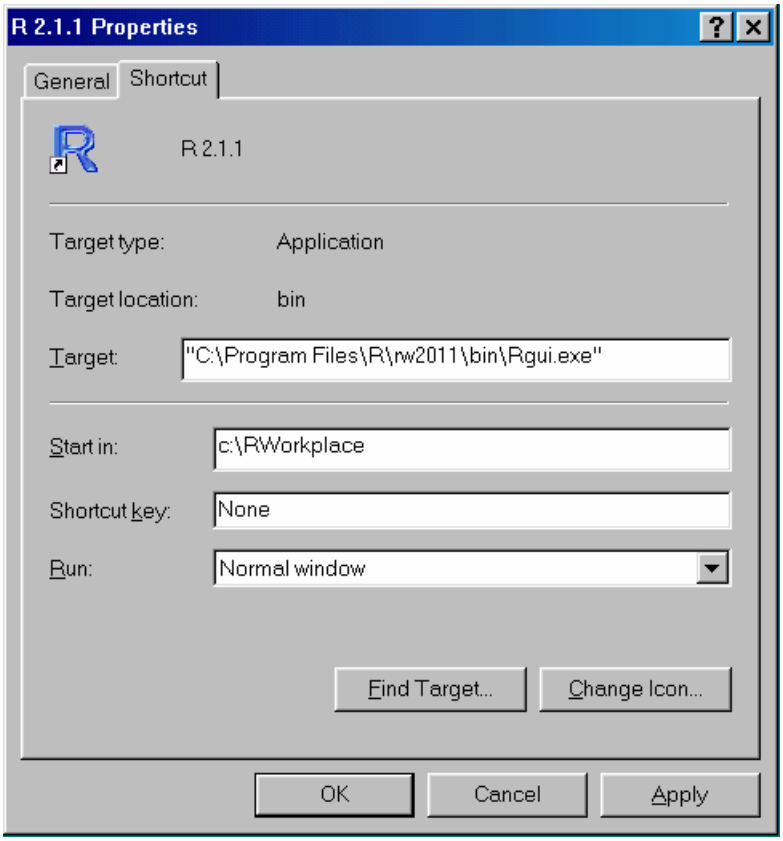
รูปที่ 1.2 Web page ของหน่วยระบาดวิทยาที่เก็บโปรแกรม R สำหรับให้ download

1.4 การติดตั้งโปรแกรม R

เมื่อ download โปรแกรมมาจาก web site ก็จะได้ไฟล์ชื่อ R-2.4.0-win32.exe ทำการติดตั้งโปรแกรม **R** ดังนี้

- Double click ที่ไฟล์ชื่อ R-2.4.0-win32.exe
- ทำตามขั้นตอน โดยการคลิก Next ต่อไปเรื่อย ๆ โปรแกรม ก็จะถูกติดตั้ง เมื่อติดตั้งเสร็จแล้ว ก็จะปรากฏ icon ของโปรแกรม ที่ desktop โดยเป็นสัญลักษณ์ตัวอักษร **R** ที่มีสีน้ำเงิน เมื่อดับเบิ้ลคลิกที่ไอคอน โปรแกรม **R** ก็จะถูกเปิดขึ้น

หลังจากการติดตั้ง โปรแกรม **R** จะถูกเก็บไว้ที่ C:\Program Files\R\r-2.4.0\ และ shortcut icon จะถูกสร้างไว้บน desktop โดยอัตโนมัติ แฟ้มข้อมูล ฟังก์ชัน ตัวแปร ผลลัพธ์ เป็นต้น ควรจะเก็บไว้ที่ working folder ที่ผู้ใช้ได้สร้างไว้ ซึ่งควรเก็บไว้ใน folder แยกต่างหากจะสะดวกกว่า เช่น เก็บไว้ที่ C:\RWorkplace โดยใช้ mouse ปุ่มขวา click ที่ไอคอน **R** แล้วเลือก Properties จากนั้น พิมพ์คำว่า C:\RWorkplace ใน 'Start in' ของ Properties ดังรูปที่ 1.3 การสร้าง working folder ควรจะสร้างโดยใช้ Windows Explorer สร้างขึ้นมาก่อน ก่อนที่จะกำหนด working folder ใน Start in ของ Properties ผู้ใช้สามารถสร้าง working folder ใหม่และใช้ชื่อตามที่ต้องการ ในที่นี้ใช้ชื่อ folder ว่า "RWorkplace"

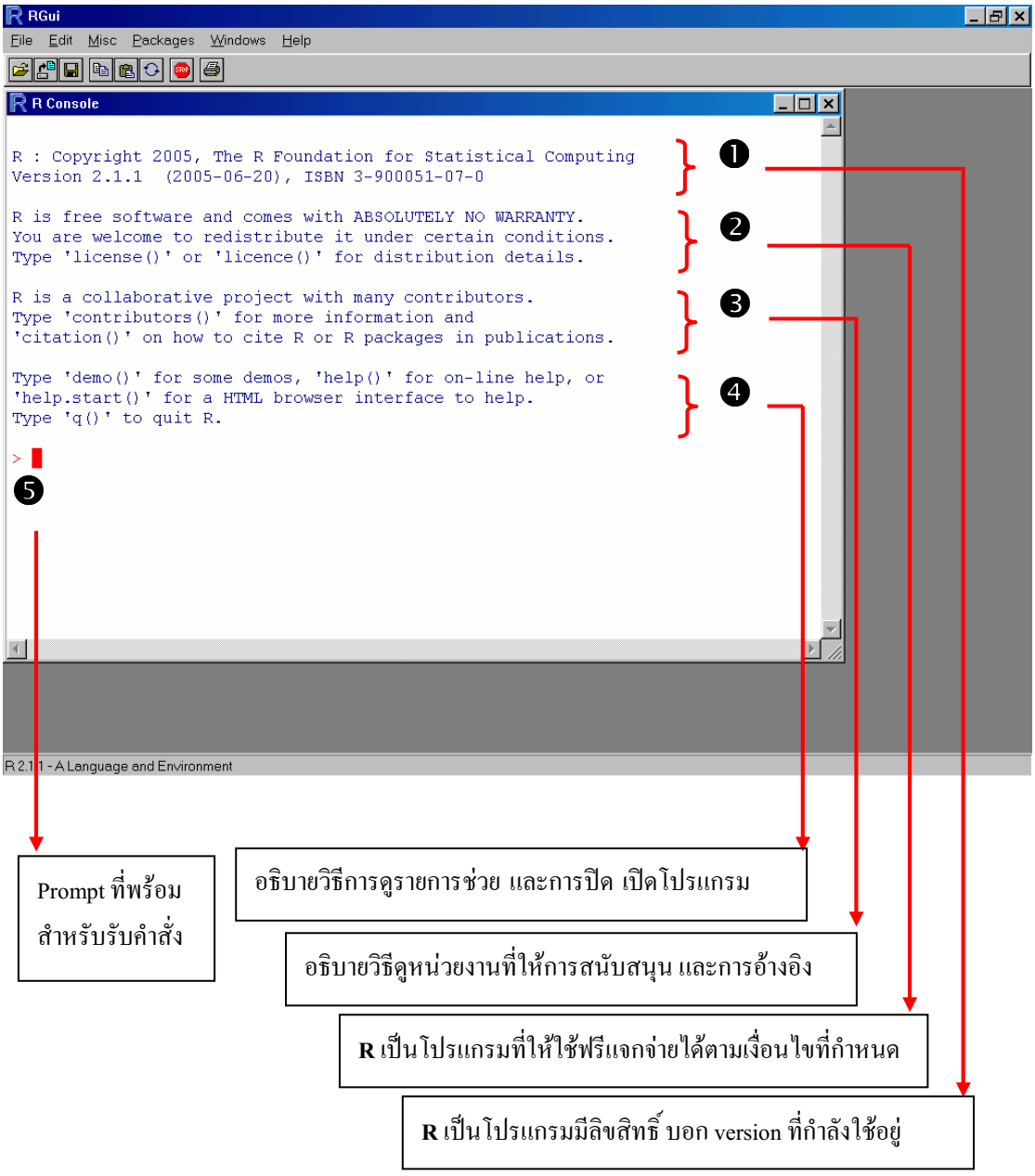


รูปที่ 1.3 การตั้งค่าใน Start in

1.5 การเริ่มทำงานกับโปรแกรม R

เมื่อเปิดโปรแกรม **R** ขึ้นมาโดยการดับเบิ้ลคลิกที่ไอคอน **R** ที่อยู่บน desktop ก็จะปรากฏ **R** Console Window ขึ้นดังรูปที่ 1.4 พร้อมกับ prompt ที่พร้อมสำหรับรับคำสั่ง ซึ่ง prompt ที่สามารถป้อนคำสั่งใช้สัญลักษณ์ > (เป็นสัญลักษณ์เครื่องหมายมากกว่า) พร้อมกับแถบ cursor อยู่หลัง

prompt ในเอกสารเล่มนี้จะแสดง prompt พร้อมกับคำสั่ง หากผู้ใช้ต้องการใช้คำสั่ง ไม่ต้องพิมพ์ prompt ที่เป็นเครื่องหมายมากกว่า เพราะเครื่องหมายนี้จะปรากฏอยู่แล้วใน **R** Console



รูปที่ 1.4 หน้าจอ Rgui (R Graphic User Interface)

เนื่องจากโปรแกรม **R** เป็นโปรแกรมที่ต้องป้อนคำสั่ง หากไม่ทราบอะไรเลย สิ่งหนึ่งที่จะช่วยได้ คือการเรียกดูรายการช่วย ดังอธิบายไว้แล้วในส่วนที่ ❹ คำสั่งที่จะขอดูรายการช่วย คือ

```
> help.start()
```

หากต้องการออกจากโปรแกรม ก็จะมีบอกไว้แล้วในส่วนที่ ❹ เช่นกัน นั่นคือ พิมพ์คำว่า

```
> q()
```

โปรแกรมจะถามว่า Save work space image? ให้เลือก No เพราะหากเลือก Yes โปรแกรมก็จะ save work space ทุกอย่างที่เราได้สร้างไว้ โดยให้นามสกุลของไฟล์เป็น ".Rdata" ที่บันทึกข้อมูลต่าง ๆ ที่ทำงานค้างไว้ ควรจะเลือก Yes ก็ต่อเมื่อต้องการบันทึกงานค้างที่ยังทำไม่เสร็จและอยาก
กลับมาทำต่อเนื่องในทันทีที่เปิด **R** มาใช้งาน

1.6 คำและความหมายในโปรแกรม R

สำหรับโปรแกรม **R** แล้ว จะมีคำต่าง ๆ ที่ผู้ใช้ไม่คุ้นเคยนัก แต่เพื่อให้ใช้โปรแกรมได้อย่างมีประสิทธิภาพ และมีความเข้าใจในโปรแกรม คำต่าง ๆ ที่ผู้ใช้จำเป็นต้องทราบคือ

1.6.1 วัตถุ (Object)

เป็นสิ่งที่ **R** สร้างขึ้น จากคำสั่งที่ผู้ใช้ได้พิมพ์ไว้ และถูกเก็บไว้ในหน่วยความจำ โปรแกรม **R** เป็นโปรแกรมชนิด object oriented ทุกสิ่งใน **R** จะมีสภาพเป็นวัตถุ มีคุณสมบัติที่สำคัญ คือ

1. มีที่อยู่ที่อ้างอิงได้ในหน่วยความจำ
2. การอ้างอิงทำได้โดยการ เรียกชื่อวัตถุนั้น และผู้ใช้สามารถสร้างและทำลายวัตถุในหน่วยความจำได้
3. วัตถุไม่ว่าจะเป็นชนิดใด เมื่อเรียกด้วยชื่อ จะมีระดับการอ้างถึงเท่ากันหมด

นอกจากนี้ยังมีคุณสมบัติอีกหลายประการที่จะไม่กล่าวถึงในที่นี้ ผลที่ได้ตามมาจากคุณสมบัติเหล่านี้คือ วัตถุอาจเป็นอะไรก็ได้ เช่นตัวแปรตัวเลข เช่น ตัวอย่าง เมื่อมีการตั้งชื่อของเวกเตอร์, กรอบข้อมูล หรือฟังก์ชัน ซ้ำเกิดขึ้น วัตถุเดิมจะถูกทำลาย (ถูกปลดจากการอ้างอิง แล้วถูก garbage collector ทำลายในภายหลัง) ดังนั้น พึงระวังในการตั้งชื่อซ้ำกัน เพราะจะไม่มีการเตือนแต่อย่างไร ตัวอย่างคำสั่งในการสร้างวัตถุขึ้นมาใหม่ เป็นดังต่อไปนี้

```
> x = 6 + 9      ❶
> x <- 6 + 9     ❷
```

คำอธิบาย

คำสั่ง ❶ และ ❷ มีความหมายเดียวกัน คือ การให้คำนวณ ค่า 6 บวกด้วย 9 และเก็บค่าที่ได้ไว้ในวัตถุที่ชื่อ x หากต้องการดูผลลัพธ์ จะต้องพิมพ์ชื่อวัตถุที่ได้สร้างขึ้น ดังเช่น

```
> x
[1] 15
```

[1] คือ ผลลัพธ์ที่ได้จากการคำนวณ

15 คือ ผลลัพธ์ลำดับที่ 1 ของ x

1.6.2 เวกเตอร์ (Vector)

โครงสร้างข้อมูลพื้นฐานชนิดหนึ่งใน **R** คือ เวกเตอร์ตัวเลข ซึ่งเป็นรูปแบบอย่างง่ายที่สุด ประกอบด้วยลำดับของตัวเลข การสร้างเวกเตอร์จะต้องอาศัย ฟังก์ชัน **c()** ดังตัวอย่าง

```
> c(10, 20, 30, 40, 50)
[1] 10 20 30 40 50
```

คำอธิบาย

ตัวอย่างข้างต้นยังไม่ได้กำหนดชื่อวัตถุที่เก็บผลลัพธ์ จึงมีการแสดงผลลัพธ์ในบรรทัดถัดมา แล้ววัตถุในหน่วยความจำที่ไม่มีชื่อ จะไม่สามารถอ้างอิงเรียกใช้ได้ และจะถูกทำลายโดยอัตโนมัติ หากต้องการเก็บผลลัพธ์ไว้ใช้ในอนาคต ควรกำหนดให้เก็บไว้ในวัตถุตามชื่อที่กำหนด เช่น

```
> x <- c(10, 20, 30, 40, 50)
> x
[1] 10 20 30 40 50
> (x <- c(10, 20, 30, 40, 50))
[1] 10 20 30 40 50
```

❶

❷

คำอธิบาย

จากตัวอย่างคำสั่งที่ ❶ เป็นการกำหนดชื่อวัตถุที่เก็บผลลัพธ์เป็นชื่อ x เมื่อ พิมพ์คำว่า x ผลลัพธ์ถูกแสดงในบรรทัดถัดมา คำสั่งที่ ❷ เป็นคำสั่งเช่นเดียวกับคำสั่งที่ ❶ แต่ใส่คำสั่ง

ทั้งหมดไว้ในวงเล็บ โดยย่อมาจากคำสั่ง `print(x <- c(10, 20, 30, 40, 50)` เพื่อให้โปรแกรมแสดงผลลัพธ์ออกมาทันที โดยไม่ต้องพิมพ์ชื่อวัตถุ ดังคำสั่งที่ ❶

1.6.3 ลิสต์ (List)

ลิสต์เป็นวัตถุที่องค์ประกอบภายในเรียงกันเป็นลำดับ ซึ่งองค์ประกอบภายในนั้น เรียกว่า component ไม่มีความจำเป็นที่องค์ประกอบของลิสต์จะต้องเป็นชนิดเดียวกันทั้งหมด ลิสต์สามารถที่จะรวมเอาเวกเตอร์ตัวเลข, ตัวแปรตัวเลข, เมตริกซ์, เวกเตอร์ซ้อน, อาเรย์, ตัวอักษร, ฟังก์ชัน และอื่น ๆ เข้าไว้ด้วยกันก็ได้ ตัวอย่างต่อไปนี้เป็นตัวอย่างอย่างง่ายสำหรับการสร้างลิสต์

```
> Lst <- list(name="Fred", wife="Mary", no.child=3, child.age=c(4,7,9))
> Lst
$name
[1] "Fred"

$wife
[1] "Mary"

$no.child
[1] 3

$child.age
[1] 4 7 9
```

แสดงผลลัพธ์ที่ได้จากการพิมพ์ชื่อวัตถุ Lst

คำอธิบาย

ทำการสร้างวัตถุชนิดลิสต์ขึ้นมา ให้ชื่อว่า `Lst` โดยภายในประกอบด้วยองค์ประกอบ ชื่อ `name`, `wife`, `no.child` และ `child.age` เมื่อพิมพ์เรียกชื่อวัตถุ `Lst` รายละเอียดต่าง ๆ ภายในลิสต์จะปรากฏขึ้น โดยมีเครื่องหมาย $ หน้าชื่อตัวแปรเป็นการบอกให้ทราบว่าชื่อที่อยู่หลังเครื่องหมายเป็นองค์ประกอบ (component) ที่อยู่ในลิสต์ องค์ประกอบเหล่านี้จะยังไม่เรียกว่าตัวแปร คำว่าตัวแปรจะใช้เมื่อเป็นองค์ประกอบในกรอบข้อมูล เพราะองค์ประกอบในลิสต์ อาจเป็นฟังก์ชัน หรือกรอบข้อมูล ก็ได้

ลองทำคำสั่งดังต่อไปนี้ และค่าที่ได้จากคำสั่งเหล่านี้คืออะไร

```
> Lst$child.age[1]
> Lst$child.age[2]
```

1.6.4 กรอบข้อมูล (Data frame)

คือลิสต์ที่อยู่ในคลาส “data.frame” มีข้อจำกัดของลิสต์ที่จะทำเป็นกรอบข้อมูลได้ คือ ต้องประกอบด้วย เวกเตอร์ (ตัวเลข, ตัวอักษร หรือ ตรรกะ), แฟกเตอร์, เมตริกซ์ตัวเลข หรือกรอบข้อมูล

กรอบข้อมูลใหม่จะได้ตัวแปรมาจากสดมภ์ของเมตริกซ์ องค์ประกอบของลิสต์ หรือตัวแปรในกรอบข้อมูลที่นำมารวมกันนั้น โครงสร้างของเวกเตอร์ ซึ่งก็คือ ตัวแปรในกรอบข้อมูลจะต้องมีขนาดความยาวเดียวกัน และโครงสร้างของเมตริกซ์ จะต้องมีความยาวของแถวเท่ากัน วัตถุประสงค์ของข้อจำกัดข้างต้น ก็คือ ต้องการทำให้ข้อมูลทั้งหมดบรรจุลงในสดมภ์ (ตัวแปร) ของกรอบข้อมูลได้พอดีด้วยฟังก์ชัน **`data.frame()`** ดังนี้

```
> x <- c(10, 5.6, -3, 6.4, 21.7)
> y <- c(5.1, 3, 0.5, 11, 2.5)
> z <- c("a1", "a2", "a3", "a4", "a5")
> data.dat <- data.frame(x,y,z)
> rm(x, y, z)
> data.dat
     x     y      z
1  10.0    5.1   a1
2   5.6    3.0   a2
3  -3.0    0.5   a3
4   6.4   11.0   a4
5  21.7    2.5   a5
> attributes(data.dat)
$names
[1] "x" "y" "z"

$row.names
[1] "1" "2" "3" "4" "5"

$class
[1] "data.frame"
```

❶ ❷ ❸ ความยาว (จำนวนองค์ประกอบ) ของ x, y และ z ต้องเท่ากัน

❹

❺

❻

แสดงรายละเอียดของข้อมูลที่อยู่ภายใน data.dat โดย x, y และ z จะถูกแสดงในแนวตั้ง คือสดมภ์

❼

แสดงชื่อตัวแปรที่มีอยู่ภายใน data.dat

แสดงจำนวนแถวที่มีอยู่ภายใน data.dat

แสดงคลาสของวัตถุ data.dat

คำอธิบาย

❶, ❷ และ ❸ เป็นการสร้างวัตถุ `x`, `y` และ `z` ที่ประกอบด้วยค่าตัวเลข และตัวอักษรต่าง ๆ โดยที่ `x` และ `y` เป็นวัตถุชนิดเวกเตอร์ตัวเลข ส่วน `z` เป็นวัตถุชนิดเวกเตอร์ตัวอักษร

❹ สร้างวัตถุ โดยให้ชื่อว่า `data.dat` ซึ่งก็คือ กรอบข้อมูล (data frame) ที่ประกอบด้วยองค์ประกอบ 3 ตัว ที่มีค่าเท่ากับ `x`, `y` และ `z`

❺ สังเกตว่าวัตถุ `x`, `y` และ `z` เดิมยังคงอยู่ จึงควรลบวัตถุ `x`, `y` และ `z` ออกจากหน่วยความจำ เพราะถ้าหากไม่ลบบอก การทำงานอาจเกิดการสับสนได้ เช่น เมื่อต้องเรียกใช้ `x` เป็นตัวแปรหนึ่งในกรอบข้อมูล `data.dat` ทำให้ไปเรียกใช้วัตถุ `x` ที่อยู่นอกกรอบข้อมูลได้

❻ เป็นการแสดงองค์ประกอบต่าง ๆ ที่อยู่ในกรอบข้อมูล `data.dat`

❼ เป็นการดูแอ็ททริบิวต์ (attribute) ของกรอบข้อมูล `data.dat` ซึ่งจะแสดงผลลัพธ์ คือ บอกชื่อตัวแปร (สดมภ์) ที่มีอยู่ในกรอบข้อมูล ถัดมาก็จะบอกถึงจำนวนแถวที่มีอยู่ภายในกรอบข้อมูล และสุดท้ายบอกประเภทของคลาส คือ เป็น กรอบข้อมูล (data.frame)

1.6.5 ฟังก์ชัน (Function)

เป็นคำสั่งที่โปรแกรม **R** ใช้ในการทำงานตามที่ผู้ใช้เลือก ฟังก์ชันของโปรแกรม **R** มีมากมาย จึงยากต่อผู้ใช้ที่จะจดจำได้ นอกจากนี้แล้ว ผู้ใช้ในระดับสูง เช่น นักสถิติ สามารถสร้างฟังก์ชันขึ้นมาใหม่ได้ตามต้องการ และหากต้องการเผยแพร่ ก็จะส่งฟังก์ชันของตัวเองให้กับ CRAN ซึ่งเป็นองค์กรกลางของ **R** เมื่อตรวจสอบการทำงาน และความถูกต้องแล้ว ฟังก์ชันเหล่านี้ก็จะถูกนำมาใส่ไว้ใน **R** ในเวอร์ชันถัดไป **R** จึงมีการพัฒนาและเปลี่ยนแปลงเวอร์ชันบ่อย ๆ โดยมีการเปลี่ยนแปลงเวอร์ชันทุก ๆ 1 – 2 เดือน ซึ่งแสดงให้เห็นว่ามีผู้พัฒนา **R** อยู่ทั่วทุกมุมโลก เป็นโปรแกรมที่พัฒนาขึ้นมาใช้ตามวัตถุประสงค์ของตนเองได้ ฟังก์ชันมักจะตามด้วยอากิวเมนต์ (argument) ที่อยู่ในวงเล็บ ซึ่ง argument ในที่นี้อาจจะเป็น ค่าตัวเลข, สมการ, ชื่อของวัตถุ หรือ ชื่อตัวแปรก็ได้ ตัวอย่างฟังก์ชันใน **R** เช่น

ฟังก์ชัน	**ความหมาย**	**ตัวอย่าง**
`log()`	ลอกการิทึม	`log(100)`
`sqrt()`	รากที่สอง	`sqrt(16)`
`exp()`	ค่ายกกำลังของ e	`exp(5)`

c()	การนำข้อมูลมาต่อกันเป็นเวกเตอร์	c(10, 20, 30, 40, 50)
cbind()	การนำเวกเตอร์มาต่อกันแนวสดมภ์	cbind(x, y)
rbind()	การนำเวกเตอร์มาต่อกันตามแนวแถว	rbind(x,y)
attach()	การนำกรอบข้อมูลปะติดกับเส้นทางค้นหาของ R	attach(data.dat)
detach()	การยกเลิกการปะติดกรอบข้อมูล	detach(data.dat)

ฟังก์ชันพิเศษ attach() และ detach()

โดยปกติในการเรียกชื่อตัวแปรในกรอบข้อมูล เราจะเรียกชื่อกรอบข้อมูลตามด้วยเครื่องหมาย $ แล้วจึงตามด้วยชื่อตัวแปร (หรือสดมภ์) เช่น เมื่อมีกรอบข้อมูล ชื่อ data.dat ที่ประกอบด้วย x, y และ z ถ้าต้องการเรียกชื่อ ตัวแปร x ภายใน data.dat จะเรียกดังนี้

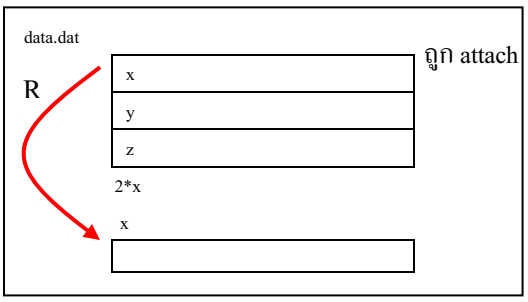
```
> data.dat$x
```

และหากต้องการบวกค่าของ x และ y ใน data.dat เข้าด้วยกัน จะต้องใช้คำสั่งดังนี้

```
> data.dat$x + data.dat$y
```

จะเห็นได้ว่าการเรียกชื่อเช่นนี้จะทำให้ต้องพิมพ์ชื่อยาวมาก ฟังก์ชัน **attach()** จะช่วยแก้ปัญหานี้ โดยเมื่อใส่ชื่อกรอบข้อมูลเป็นอากิวเมนต์ของฟังก์ชัน **attach()** เช่น **attach(data.dat)** ก็จะทำให้ **R** มองเห็นชื่อตัวแปรภายในกรอบข้อมูลที่ attach แล้วนั้นโดยไม่ต้องเรียกชื่อกรอบข้อมูลนำหน้า ดังนั้น การบวก x เข้ากับ y เข้าด้วยกัน จะทำได้ง่ายขึ้น ดังนี้

```
> attach(data.dat)
> x+y
```



ถึงจุดนี้พึงสังเกตว่าคำสั่งกำหนดค่าต่อไปนี้

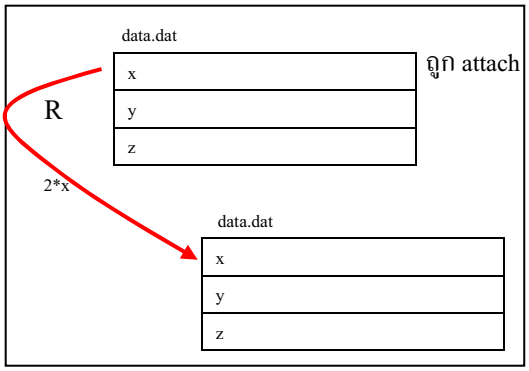
```
> x <- 2*x
```

จะไม่เป็นการเปลี่ยนค่า x ใน data.dat ให้เป็น 2*x แต่จะเป็นการสร้างตัวแปร x ขึ้นมาใหม่นอกกรอบข้อมูล data.dat โดยใช้ค่า x ในกรอบข้อมูล data.dat คูณด้วย 2 คือ มีความหมายเท่ากับ

```
> x <- 2*data.dat$x
```

ถ้าหากต้องการเปลี่ยนค่า x ในกรอบข้อมูล data.dat ให้มีค่าใหม่เป็น 2*x จะต้องใส่ชื่อเดิมของตัวแปร x นั้นไว้ด้านซ้ายมือของเครื่องหมาย <- ดังนี้

```
> data.dat$x <- 2*x
```



อย่างไรก็ตาม พึงจำไว้ว่าขณะนี้ **R** จะยังจำวัตถุกรอบข้อมูล data.dat อันเดิมที่ได้ attach ไว้ก่อนหน้านี้ ถ้าเรียกดู x โดยไม่ใส่ชื่อกรอบข้อมูลนำหน้าจะยังคงได้ค่า x เหมือนเดิม ไม่ใช่ค่าใหม่แต่อย่างใด การที่จะให้ **R** เลิกจำกรอบข้อมูลเดิม (ในหน่วยความจำ) จะใช้คำสั่ง **detach()** และเมื่อ attach กรอบข้อมูล data.dat ใหม่ ค่า x ที่เรียกได้จึงจะแสดงค่าใหม่ให้เห็น ดังนี้

```
> x
> detach(data.dat)
> attach(data.dat)
> x
```

ให้ค่า x เดิม

ให้ค่า x ที่คำนวณใหม่

หมายเหตุ detach กรอบข้อมูลที่ได้ attach ไว้ทุกครั้งที่ทำงานกับกรอบข้อมูลเสร็จแล้ว มิฉะนั้นจะทำงานกับข้อมูลผิดชุดได้ หากต้องการทำงานกับกรอบข้อมูลนั้นอีก ก็ให้ attach ใหม่อีกครั้ง

ฟังก์ชัน `search()` และ `searchpaths()`

ฟังก์ชัน search เป็นการค้นหาวัตถุที่มีอยู่ใน R console ในขณะที่กำลังเปิดใช้งาน คำสั่งนี้จะแสดงpackage หรือ library หรือวัตถุ ที่ถูก attach ใน R ดังตัวอย่างต่อไปนี้

```
> search()                                                    ❶
[1] ".GlobalEnv"        "package:methods"   "package:stats"
[4] "package:graphics"  "package:grDevices" "package:utils"
[7] "package:datasets"  "Autoloads"         "package:base"

> searchpaths()
[1] ".GlobalEnv"
[2] "C:/PROGRA~1/R/RW2001/library/methods"
[3] "C:/PROGRA~1/R/RW2001/library/stats"
[4] "C:/PROGRA~1/R/RW2001/library/graphics"
```

```
[5] "C:/PROGRA~1/R/RW2001/library/grDevices"
[6] "C:/PROGRA~1/R/RW2001/library/utils"
[7] "C:/PROGRA~1/R/RW2001/library/datasets"
[8] "Autoloads"
[9] "C:/PROGRA~1/R/RW2001/library/base"
```

คำอธิบาย

คำสั่ง ❶ แสดงชื่อ packages หรือ library ที่ถูก attach ใน **R** ส่วนคำสั่งที่ ❷ แสดงเส้นทางหรือที่อยู่ใน library นั้น ๆ ว่าอยู่ใน drive ไหน folder อะไร

1.6.6 คลาส (Class)

วัตถุทุกชนิดจะถูกจัดเป็นคลาสประเภทต่าง ๆ เช่น กรอบข้อมูล, ตัวเลข, อักษร, แฟกเตอร์, ลิสต์, เมตริกซ์ เป็นต้น และในคลาสใหญ่คลาสหนึ่งอาจมีคลาสย่อยต่าง ๆ ได้อีก เช่น แฟกเตอร์ เป็นคลาสย่อยของคลาสเวกเตอร์ตัวเลข

1.6.7 แฟกเตอร์ (Factor)

แฟกเตอร์ เป็นคลาสย่อยของเวกเตอร์ตัวเลขที่มีคุณสมบัติพิเศษ คือ แต่ละค่าจะเป็นรหัสแทนความหมายใด ๆ เช่น เพศ ให้ตัวเลข 1 และ 2 โดยที่เลขรหัส 1 คือ ชาย และ 2 คือ หญิง

1.6.8 อาเรย์ (Array)

อาเรย์ หมายถึงชุดของข้อมูลชนิดเดียวที่เรียงเป็นชั้นหลายมิติ **R** จะถือว่ามิติแรกเป็นแถว มิติที่ 2 เป็นสดมภ์ และมิติที่ 3 เป็นชั้น (stratum) แม้ว่าในความเป็นจริง อาเรย์อาจมีมากกว่า 3 มิติก็ได้ แต่ในทางสถิติมักจำกัด อาเรย์ไว้เพียง 3 มิติเท่านั้น

1.6.9 เมตริกซ์ (Matrix)

เมตริกซ์ เป็น เวกเตอร์หลาย ๆ เวกเตอร์มาประกอบกัน ซึ่งเวกเตอร์ จะมีเพียงแถวเพียงหนึ่งแถว ส่วนเมตริกซ์ เป็นการรวมเวกเตอร์ที่เป็นตัวเลขเข้าด้วยกัน จึงมีแถวหลายแถว ดังนั้นเมตริกซ์จึงเป็นอาเรย์ที่มี 2 มิติ คือ แถวกับสดมภ์ เมตริกซ์ทำให้เป็นกรอบข้อมูลได้ แต่กรอบข้อมูลที่ทุกตัวแปรเป็นชนิดเดียวกันเท่านั้นที่สามารถถูกเปลี่ยนให้เป็นเมตริกซ์ได้

1.6.10 แพคเกจ (Package)

หมายถึง ชุดฟังก์ชัน ซึ่งประกอบไปด้วยหลาย ๆ ฟังก์ชัน สร้างขึ้นเพื่อใช้ในการทำงานตามวัตถุประสงค์นั้น ๆ

1.6.11 ไลบรารี่ (Library)

หมายถึง โครงสร้างที่รวมของ Package, help, demo และ อื่น ๆ ที่เกี่ยวข้องกับ package นั้น ผู้ใช้ในระดับสูงสามารถสร้าง Library ของตัวเองขึ้นมาได้ตามที่ต้องการ เพื่อให้ง่ายต่อการใช้งาน ดังเช่น library **ice** เป็นต้น

1.7 สัญลักษณ์ที่ใช้ในโปรแกรม R

ในการเขียนคำสั่ง ในโปรแกรม **R** จะมีสัญลักษณ์พิเศษที่มักจะพบเจอในโปรแกรมนี้ ได้แก่

1.7.1 สัญลักษณ์ทั่ว ๆ ไป

สัญลักษณ์	ความหมาย
<−	Left assignment หมายถึง ให้นำค่าหรือผลจากการคำนวณหรือการทำงานของฟังก์ชันที่ปลายลูกศร ไปใส่ในวัตถุที่เขียนไว้ด้วยหัวลูกศร (ด้านซ้าย)
−>	Right assignment หมายถึง ให้นำค่าหรือผลจากการคำนวณหรือการทำงานของฟังก์ชันที่ปลายลูกศร ไปใส่ในวัตถุที่เขียนไว้ด้วยหัวลูกศร (ด้านขวา)
=	เป็นเครื่องหมาย assignment เช่นเดียวกับ <- แต่ไม่ควรใช้ เพราะจะสับสนกับเครื่องหมาย == ที่เป็นเครื่องหมายเท่ากับทางตรรกศาสตร์

1.7.2 สัญลักษณ์ทางคณิตศาสตร์

สัญลักษณ์	ความหมาย
+	บวก
-	ลบ
*	คูณ
/	หาร
^	ยกกำลัง

1.7.3 สัญลักษณ์ทางตรรกศาสตร์

สัญลักษณ์	ความหมาย
<	น้อยกว่า
>	มากกว่า
==	เท่ากับ
>=	มากกว่าหรือเท่ากับ
<=	น้อยกว่าหรือเท่ากับ
!=	ไม่เท่ากับ
&	และ
|	หรือ

1.8 ข้อควรทราบในการเขียนคำสั่งด้วย R

1.8.1 ประเภทของตัวพิมพ์ภาษาอังกฤษ

โปรแกรม **R** ถือว่าภาษาอังกฤษตัวพิมพ์เล็ก และตัวพิมพ์ใหญ่ แตกต่างกัน เช่น

```
> a <- 2 + 5
> A <- 6 + 7
> a==A
[1] FALSE
```

คำอธิบาย

วัตถุ a และ วัตถุ A ข้างต้น **R** จะถือว่าเป็นคนละตัวกัน

1.8.2 การแสดงผลลัพธ์

โดยปกติแล้วการคำนวณใน **R** โดยการให้เก็บผลลัพธ์เป็นชื่อวัตถุ **R** จะไม่แสดงผลลัพธ์ให้เห็น จนกว่าจะพิมพ์ชื่อวัตถุนั้น แต่หากต้องการให้แสดงผลลัพธ์ในทันที และเก็บผลลัพธ์นั้นไว้เป็นวัตถุ ให้พิมพ์วงเล็บคร่อมคำสั่งทั้งหมด ดังตัวอย่าง

```
> (a <- 2 + 5)
[1] 7
```

1.8.3 การกำหนดชื่อของวัตถุ

ชื่อของวัตถุประกอบด้วยตัวอักษรที่มีตั้งแต่หนึ่งตัวขึ้นไป สามารถมีได้ไม่จำกัด แต่ต้องไม่มีการเว้นวรรค หากชื่อมีข้อความต่อกันมาก ๆ สามารถคั่นด้วยจุด (.) หรือใช้ตัวพิมพ์ใหญ่ หรือใช้เครื่องหมาย "_" (underscore) ได้ ตัวอย่าง เช่น

```
> mean.blood.pressure<- (120+130)/2
> mean_blood_pressure <- (120+130)/2
> MeanBloodPressure <- (120+130)/2
```

1.8.4 การรวมคำสั่งไว้ในบรรทัดเดียวกัน

R สามารถที่จะรวมการพิมพ์คำสั่งหลาย ๆ คำสั่งให้อยู่ในบรรทัดเดียวกันได้ โดยการคั่นแต่ละคำสั่งด้วยเครื่องหมาย ; (semi colon)

```
> a <- 2 + 5 ; b <- 5 + 7
> a; b
[1] 7
[2] 12
```

1.8.5 การใช้คำอธิบาย

ผู้ใช้สามารถใส่คำอธิบายต่าง ๆ ในโปรแกรม **R** ได้ ซึ่งสามารถทำได้โดยใช้เครื่องหมาย # ข้อความใด ๆ ก็ตาม ที่ตามหลังเครื่องหมาย # **R** จะไม่นำไปทำงาน การใช้คำอธิบาย เหมาะสำหรับผู้ใช้ที่ต้องการใส่ข้อความหลังคำสั่งที่ใช้ใน **R** ว่าคำสั่งแต่ละคำสั่ง สร้างขึ้นเพื่อให้ทำอะไรบ้าง เช่น

```
> name <- c("Tom", "John", "Smith") # สร้างวัตถุชื่อ name ประกอบด้วยชื่อคน 3 ชื่อ
> age <- c(30, 40, 37)              # สร้างวัตถุชื่อ age ประกอบด้วยอายุ 3 ค่า
> wt <- c(70, 80, 75)               # สร้างวัตถุชื่อ wt ประกอบด้วยน้ำหนัก 3 ค่า
> ht <- c(180, 185, 179)            # สร้างวัตถุชื่อ ht ประกอบด้วยส่วนสูง 3 ค่า
> person.dat <- data.frame(I(name), age, wt, ht)
# สร้างกรอบข้อมูล person.dat มีตัวแปรชื่อ name, age, wt และ ht
# โดยกำหนดให้ตัวแปร name เป็นตัวแปรประเภทตัวอักษร ด้วยการใส่สัญลักษณ์ I นำหน้าตัวแปร
```

1.8.6 การเรียกใช้คำสั่งเดิม

หากต้องการเรียกใช้คำสั่งเดิมที่ได้ใช้ไปแล้ว สามารถเรียกใช้โดยไม่ต้องพิมพ์ ด้วยการกดลูกศรชี้ขึ้นบนคีย์บอร์ด **R** จะเรียกคำสั่งที่ใช้ไปก่อนหน้า ทีละ 1 คำสั่ง

1.8.7 การเรียกดูวัตถุที่ถูกสร้างไว้แล้วใน R console

การทำงานใน **R** ผู้ใช้มักจะสร้างวัตถุต่าง ๆ ขึ้นมา บางครั้งไม่สามารถจำชื่อได้หมด หรือชื่อวัตถุที่ถูกสร้างใหม่มีชื่อเหมือนกับชื่อตัวแปรในกรอบข้อมูล ในกรณีที่ชื่อวัตถุและชื่อตัวแปรในกรอบข้อมูลมีชื่อเหมือนกัน **R** จะเรียกใช้วัตถุที่อยู่นอกกรอบข้อมูลมาทำงานก่อนเสมอ ดังนั้นจึงต้องมีการตรวจสอบเสมอว่าได้สร้างวัตถุอะไรไปบ้าง วัตถุอะไรที่อยู่นอกกรอบข้อมูล ด้วยการใช้คำสั่ง **`ls()`**

```
> ls()
 [1] "a"                    "A"                    "age"
 [4] "b"                    "ht"                   "mean.blood.pressure"
 [7] "mean_blood_pressure""MeanBloodPressure"    "name"
[10] "person.dat"          "wt"
```

คำสั่ง **`ls()`** แสดงให้เห็นเพียงชื่อของวัตถุที่ถูกสร้างขึ้นเท่านั้น แต่ไม่แสดงชื่อ package หรือ library ให้เห็น

1.8.8 การเรียกดูแถวหรือสดมภ์ในกรอบข้อมูล

ลำดับการเรียกข้อมูลใน **R** กำหนดให้ตัวแรกสุดเป็นแถวก่อนเสมอ ตัวถัดมาจะเป็นสดมภ์ ดังคำสั่งต่อไปนี้

```
> person.dat[1,1]          # ให้แสดงข้อมูลในแถวที่ 1 สดมภ์ที่ 1
[1] "Tom"
```

หากต้องการเรียกทุกแถว หรือทุกสดมภ์ ทำได้โดยการพิมพ์เครื่องหมายคอมม่า “,”

```
> person.dat[1,]           # ให้แสดงข้อมูลในแถวที่ 1 ทุกสดมภ์
  name age wt  ht
1  Tom  30 70 180
> person.dat[,2]           # ให้แสดงข้อมูลทุกแถวของสดมภ์ที่ 2
[1] 30 40 37
```

หากต้องการเรียกชื่อตัวแปร ให้พิมพ์เครื่องหมาย “$” หลังชื่อกรอบข้อมูลสำหรับกรณีที่ไม่ได้ attach ข้อมูล

```
> person.dat[person.dat$name=="John",]
  name age wt  ht
2 John  40 80 185
```

1.8.9 การลบวัตถุออกจาก R console

วัตถุบางตัว ที่ไม่ต้องการใช้แล้ว หรือป้องกันการสับสนระหว่างชื่อวัตถุกับชื่อตัวแปร ผู้ใช้สามารถลบออกจาก **R** console ได้ ด้วยคำสั่ง **rm()**

```
> rm(age, name, wt, ht)
> ls()
[1] "a"                    "A"                    "b"
[4] "mean.blood.pressure" "mean_blood_pressure" "MeanBloodPressure"
[7] "person.dat"
```

เมื่อใช้คำสั่ง **ls()** ดูวัตถุที่อยู่ใน **R** console จะเห็นได้ว่า วัตถุที่ชื่อ age และ name ถูกลบไปแล้ว โดยในกรอบข้อมูล person.dat มีตัวแปรชื่อ age, name, wt และ ht เช่นกัน ดังนั้น การลบวัตถุ age, name, wt และ ht ออก ก็จะช่วยลดการสับสนของการเรียกใช้ตัวแปรในภายหลัง หากต้องการลบวัตถุทุกชนิดออกจาก **R** console ก็สามารถทำได้ด้วยการใช้คำสั่งดังนี้

```
> rm(list=ls())
```

1.8.10 การลบผลลัพธ์ออกจากหน้าจอ R console

ในบางครั้งเมื่อผู้ใช้ทำงานไปนาน ๆ ก็จะมีผลลัพธ์ปรากฏขึ้นหน้าจอเป็นจำนวนมาก หากไม่ต้องผลลัพธ์ดังกล่าว สามารถลบออกจากหน้าจอ **R** console ได้ โดยการไปเลือกที่เมนู Edit และเลือก Clear console หรือ กด Ctrl+L

1.8.11 การสลับไปมาระหว่างหน้าจอต่าง ๆ ใน RGui

หน้าจอในการพิมพ์คำสั่งต่าง ๆ ใน **R** คือ **R** Console แต่หากบางครั้งมีการสร้างกราฟ **R** ก็จะปรากฏหน้าจอ **R** Graphics มาให้ หรือการเรียก help ขึ้นมาดู หากต้องการทำงานสลับไปมาระหว่างหน้าจอต่าง ๆ เหล่านี้ ทำได้โดยการกด Alt ค้างไว้ ตามด้วยปุ่ม Tab เพื่อสลับไปยังหน้าจอที่ต้องการ

1.8.12 ความครบถ้วนของคำสั่ง

หากคำสั่งที่พิมพ์ไม่ครบถ้วน เมื่อกด Enter **R** จะไม่กระทำตามคำสั่ง แต่จะขึ้นบรรทัดใหม่ พร้อมด้วยเครื่องหมาย + นั่นหมายถึงคำสั่งยังไม่ครบถ้วน เช่น ขาดวงเล็บปิด ก็จะต้องพิมพ์วงเล็บปิด แล้วจึงกด Enter **R** จึงจะกระทำตามคำสั่งนั้น แต่ถ้าหากคำสั่งนั้น ๆ ตกวงเล็บหลายวงเล็บ **R** จะไม่ให้ผู้ใช้แก้ไขคำสั่งนั้น ในกรณีนี้ให้กด Esc เพื่อยกเลิกคำสั่ง

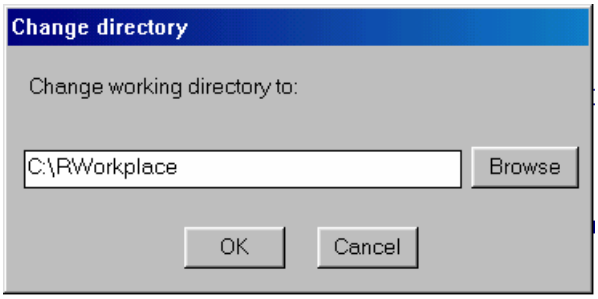
1.8.13 การกำหนดโฟลเดอร์สำหรับการทำงาน

หากไม่ได้กำหนดโฟลเดอร์ที่ต้องการทำงานด้วยการเลือก Properties จากไอคอน **R** สามารถพิมพ์คำสั่งใน **R** Console เพื่อกำหนดไดเรกตอรี่ที่ต้องการทำงานด้วยคำสั่ง **`setwd()`** ดังนี้

```
> setwd("c:/rworkplace")
```

R กำหนดให้ใช้สัญลักษณ์ **forward slash (/)** เป็นตัวแบ่งแยก sub-folder ต่าง ๆ เหมือนระบบ Linux ในขณะที่โปรแกรมทั่วไปบน Windows กำหนดให้ใช้สัญลักษณ์ **back slash (\)** เป็นตัวแบ่งแยก หรือสามารถกำหนดโฟลเดอร์สำหรับการทำงานได้โดยการเลือกเมนู File จากนั้นเลือก Change dir... ก็จะปรากฏ

ดังรูปที่ 1.5 ให้พิมพ์ C:\RWorkplace (ถ้าเป็นการเปลี่ยน Folder หรือ Directory ที่เลือกจากเมนู จะใช้เครื่องหมาย \ เหมือนกับโปรแกรมอื่นๆ)



รูปที่ 1.5 การเปลี่ยน working directory

1.9 การเรียกใช้การช่วยเหลือ

โปรแกรม **R** มีลักษณะพิเศษ คือ ผู้ใช้สามารถขอความช่วยเหลือในขณะที่ใช้ผ่าน online help การดูวิธีการเรียกใช้ help สามารถใช้คำสั่งดังต่อไปนี้

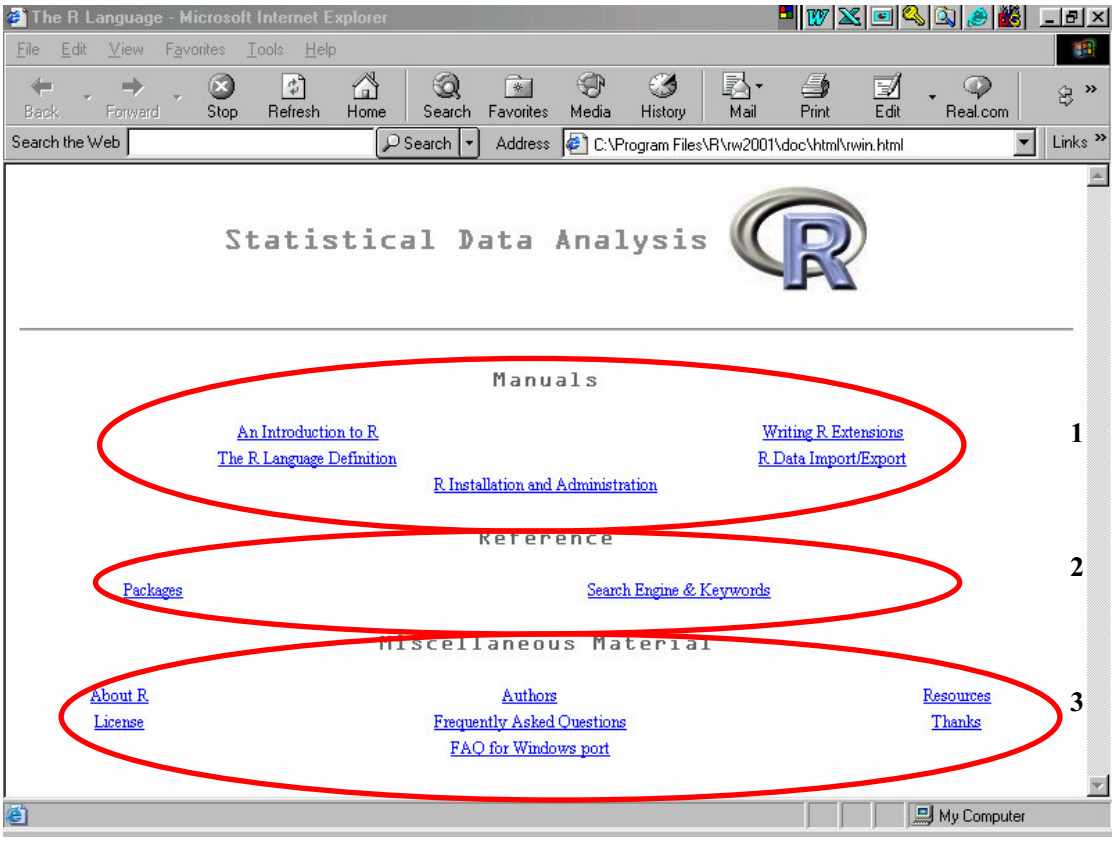
```
> help.start()          ❶
> help()                ❷ หรือ
> ?help                 ❸
```

คำอธิบาย

คำสั่งที่ ❶ ใช้ในกรณีที่ต้องการทราบรายละเอียดต่าง ๆ ใน **R** เมื่อพิมพ์คำสั่งนี้ หน้าจอ online help ดังรูปที่ 1.6 ก็จะปรากฏขึ้น ซึ่งจะมีให้เลือก 3 ส่วนด้วยกัน (ดังรูป 1.5) คือ

1. Manuals เป็นส่วนที่แสดงคู่มือการใช้โปรแกรม
2. Reference เป็นส่วนที่เก็บรวบรวม packages หรือ libraries ต่าง ๆ และมีส่วนของการค้นหาโดยใช้คำสำคัญ คือ Search Engine & Keywords
3. Miscellaneous Material เป็นส่วนอื่น ๆ ที่เกี่ยวกับโปรแกรม เช่น ประวัติความเป็นมาของการเขียนโปรแกรม ผู้เขียนโปรแกรม ลิขสิทธิ์ เป็นต้น

คำสั่งที่ ❷ และ ❸ ใช้เหมือนกัน โดยคำสั่งที่ ❸ ใช้เครื่องหมาย ? แทนการพิมพ์คำว่า help เมื่อพิมพ์คำสั่งดังกล่าว **R** จะแสดงหน้าจอ online help ขึ้นมาแสดงให้เห็นวิธีการใช้คำสั่ง help



รูปที่ 1.6 หน้าจอแสดงผลการใช้คำสั่ง `help.start()`

1.9.1 กรณีทราบคำสั่งที่มีอยู่แล้วใน R

ถ้าต้องการทราบไวยากรณ์ (Syntax) ของการใช้คำสั่งต่าง ๆ ที่มีอยู่แล้วใน **R** ให้พิมพ์ คำสั่ง **help()** ตามด้วยชื่อคำสั่งอยู่ในวงเล็บ หรือ สามารถพิมพ์ เครื่องหมายคำถาม คือ ? ซึ่งใช้แทนคำว่า help จากนั้นจึงตามด้วยคำสั่งที่ต้องการขอความช่วยเหลือ เช่น

```
> help(mean)          ❶
> ?mean               ❷
```

คำอธิบาย

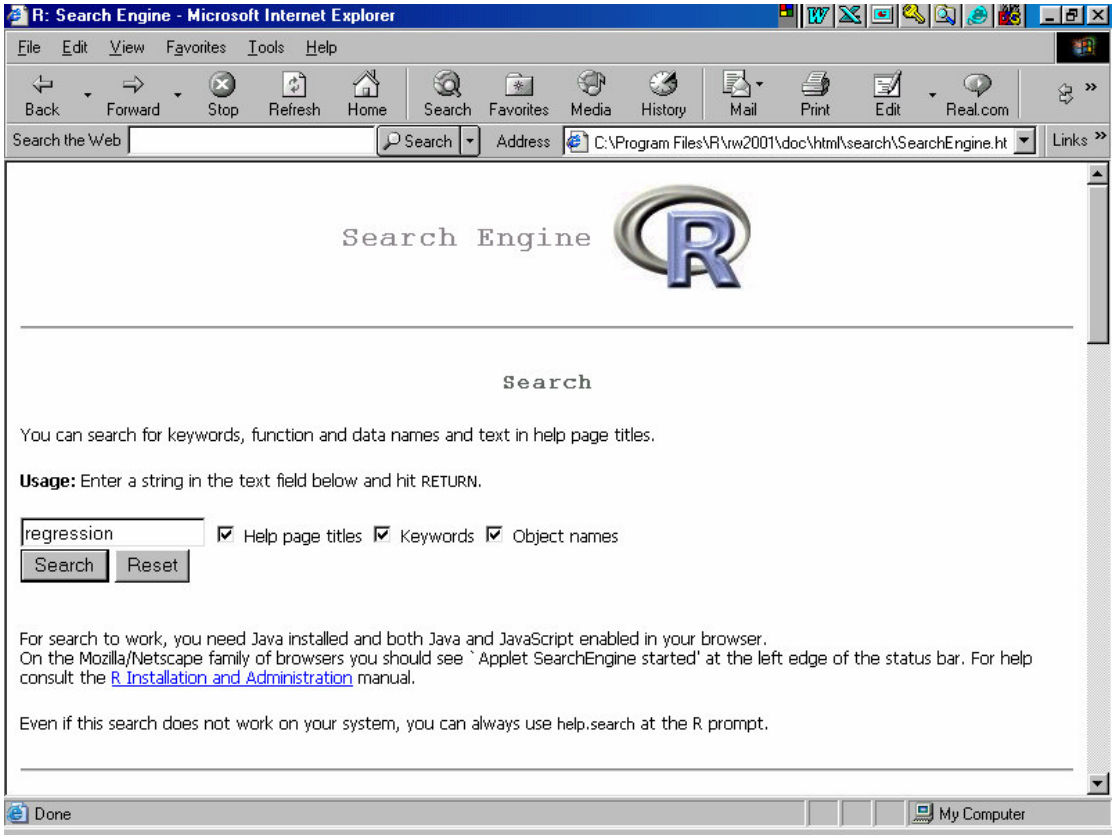
คำสั่งทั้งสองคำสั่งจะแสดงหน้าจอ online help ของการใช้คำสั่ง mean เหมือนกัน ฉะนั้นหากต้องการเรียกความช่วยเหลือที่สั้นกะทัดรัดก็ควรจะใช้คำสั่งที่ ❷

1.9.2 กรณีไม่ทราบคำสั่งที่มีอยู่ใน R

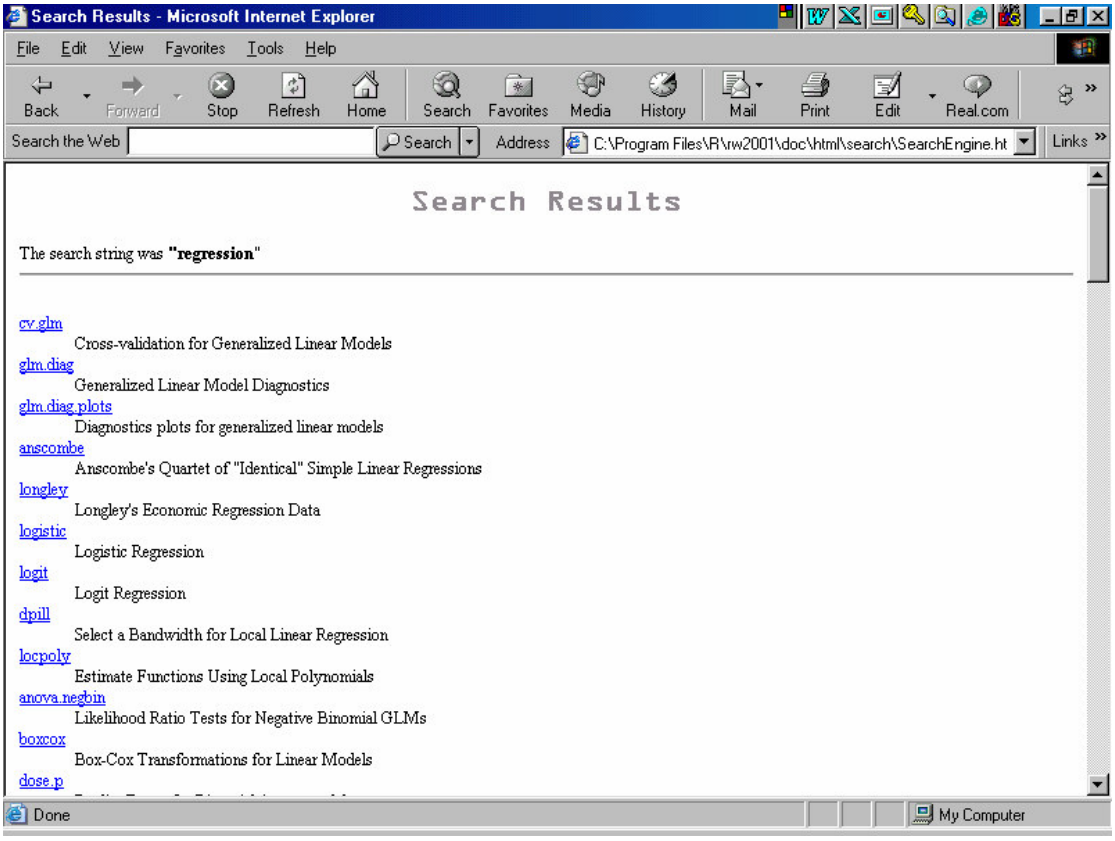
สำหรับกรณีนี้ หากทราบคำบางคำที่เกี่ยวข้องกับงานที่ต้องการใช้ ซึ่งอาจเป็นคำสำคัญ (keyword) เช่น คำว่า test การเรียกดูความช่วยเหลือจะต้องพิมพ์คำนั้นไว้ในเครื่องหมาย " " ดังเช่น

```
> help.search("test")
```

อย่างไรก็ตาม การพิมพ์ **help.start()** เพื่อเปิด online help ดังรูปที่ 1.6 แล้วเลือก Search Engine & Keywords จะเป็นวิธีที่ง่ายและมีตัวเลือกให้มากกว่า เช่นต้องการค้นคำว่า regression ก็ให้คลิกที่เมนูดังกล่าวก็จะปรากฏหน้าจอดังรูปที่ 1.7 ขึ้น และให้พิมพ์คำว่า regression ในช่องว่าง จากนั้นจึงกดปุ่ม Search ผลการค้นหาดังในรูปที่ 1.8



รูปที่ 1.7 หน้าจอ Search Engine ใน R



รูปที่ 1.8 ผลการ Search จากคำสำคัญ regression

1.9.3 กรณีคำสั่งที่ต้องการใช้ไม่ได้อยู่ในไลบรารี่ที่เปิดมาพร้อมกับ R console

โดยปกติแล้วหากเปิดโปรแกรม **R** ขึ้นมาใช้งาน library ที่ถูกเปิดขึ้นโดยอัตโนมัติอยู่แล้ว คือ base stats graphics grDevices utils methods และ datasets การขอดูความช่วยเหลือคำสั่งที่อยู่ใน library เหล่านี้ก็จะเรียกได้ทันที แต่หากคำสั่งที่ต้องการใช้อยู่ใน library อื่น จะต้องเปิด library นั้นขึ้นมาก่อน จึงจะสามารถเรียกดูความช่วยเหลือคำสั่งที่อยู่ใน library ที่ต้องการได้ ดังเช่น การดูความช่วยเหลือของคำสั่ง negative.binomial ซึ่งอยู่ใน library MASS จะต้องเปิด library นี้ขึ้นมาก่อน ดังนี้

```
> library(MASS)
> help(negative.binomial)
```

1.9.4 การเรียกตัวอย่างในแฟ้มความช่วยเหลือมาทำงานบน R console

เมื่อต้องการเรียกตัวอย่างคำสั่งมาทำงาน อาจใช้วิธี copy ข้อความมา paste บน **R** console ก็ได้ แต่มีอีกวิธีหนึ่งที่สะดวกคือใช้คำสั่ง **`example()`** ตามด้วยชื่อคำสั่งที่อยู่ในวงเล็บ ดังเช่น ต้องการดูตัวอย่างการใช้คำสั่ง **`summary()`** ซึ่งจะได้ผลลัพธ์บน **R** console ดังนี้

```
> example(summary)

summry> summary(attenu, digits = 4)
     event            mag           station         dist
 Min.   : 1.00   Min.   :5.000   117    :  5   Min.   :  0.50
 1st Qu.: 9.00   1st Qu.:5.300   1028   :  4   1st Qu.: 11.32
 Median :18.00   Median :6.100   113    :  4   Median : 23.40
 Mean   :14.74   Mean   :6.084   112    :  3   Mean   : 45.60
 3rd Qu.:20.00   3rd Qu.:6.600   135    :  3   3rd Qu.: 47.55
 Max.   :23.00   Max.   :7.700   (Other):147   Max.   :370.00
                                 NA's   : 16
     accel
 Min.   :0.00300
 1st Qu.:0.04425
 Median :0.11300
 Mean   :0.15422
 3rd Qu.:0.21925
 Max.   :0.81000

summry> summary(attenu$station, maxsum = 20)
  117   1028     113    112    135    475   1030   1083   1093   1095
    5      4       4      3      3      3      2      2      2      2
  111    116    1219   1299    130   1308   1377   1383(Other)   NA's
    2      2       2      2      2      2      2      2    120     16
```

```
summry> lst <- unclass(attenu$station) > 20

summry> summary(lst)
   Mode   FALSE    TRUE    NA's
logical      28     138      16

summry> summary(as.factor(lst))
FALSE  TRUE  NA's
   28   138    16
```

ในการใช้คำสั่ง **example()** จะทำให้ผลที่ได้จากการทำงานของตัวอย่างมี prompt เป็นชื่อของคำสั่งนั้น ตามด้วยเครื่องหมาย '>' เช่นเป็น 'summary>' ดังตัวอย่างข้างต้น

1.9.5 รูปแบบการแสดงหน้าจอความช่วยเหลือ

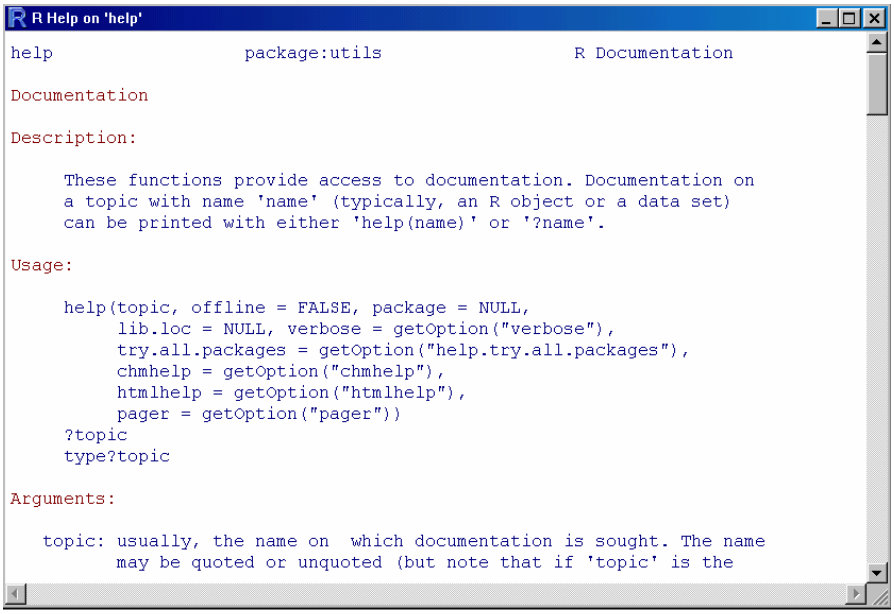
R แสดงหน้าจอ help เป็น 3 รูปแบบด้วยกัน คือ

1. ความช่วยเหลือที่ปรากฏด้วยขึ้นมาภายใต้หน้าจอของ RGUI
2. ความช่วยเหลือที่ปรากฏด้วย Web browser ซึ่งอยู่ในรูปแบบ html
3. ความช่วยเหลือที่ปรากฏด้วย Windows ที่เป็น Complied html ขึ้นมา

โดย **R** จะแสดงหน้าจอ online help ตามรูปแบบที่ผู้ใช้เลือก ตัวอย่างการใช้คำสั่ง help ดังนี้

```
> help(help)
```

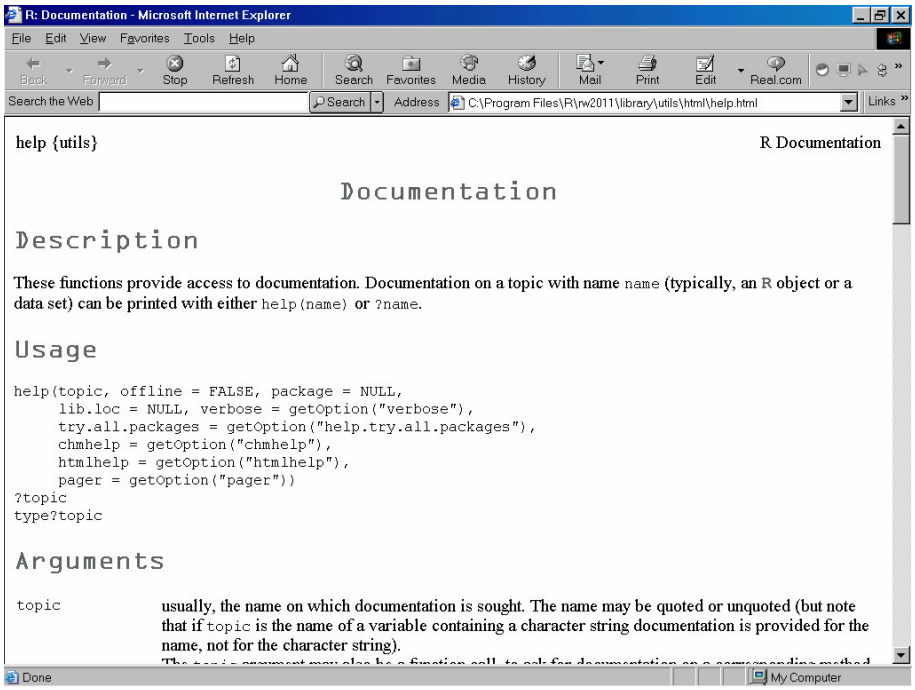
จากคำสั่งนี้ **R** จะแสดงหน้าจอ help ดังรูปที่ **1.9**



รูปที่ 1.9 หน้าจอ help ภายใต้ RGUI

```
> help(help, html=T)
```

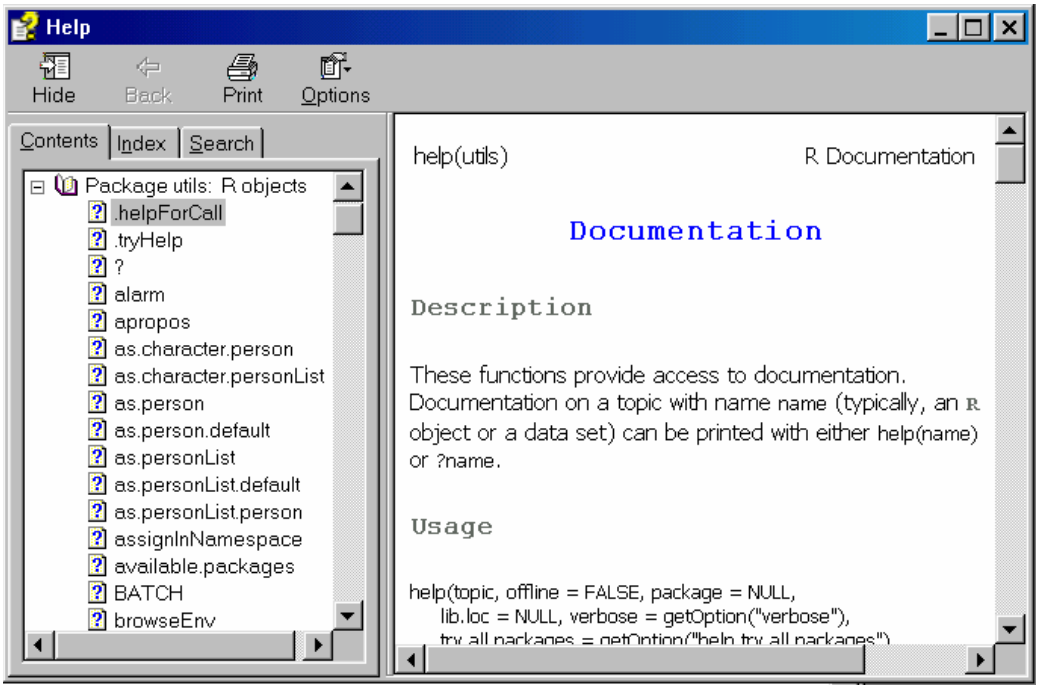
คำสั่งข้างต้นจะแสดงหน้าจอ help ด้วย web browser ดังรูปที่ 1.10



รูปที่ 1.10 หน้าจอ help จาก web browser

```
> help(help, chm=T)
```

จากคำสั่งนี้ **R** จะแสดงหน้าจอ help ดังรูปที่ 1.11



รูปที่ 1.11 หน้าจอ help จาก Complied html

1.10 แบบฝึกหัด

จงใช้โปรแกรม **R** คำนวณค่าต่าง ๆ ต่อไปนี้

1. ค่ารากที่สองของ 876 มีค่าเท่าใด
2. ค่าลอกกาลิทึมของ 0.0001 มีค่าเท่าใด
3. จากผลการศึกษาอัตราการเกิดโรคชนิดหนึ่ง เป็นดังนี้

	เป็นโรค	ไม่เป็นโรค	รวม
ชาย	50	120	170
หญิง	75	205	280
รวม	125	325	450

- จงคำนวณหาอัตราการเป็นโรคในชาย และหญิง
- จงคำนวณค่า Odds ของการเป็นโรคในเพศชาย และหญิง โดยสูตรการคำนวณ คือ $Odds_{ชาย} = a/b$ และ $Odds_{หญิง} = c/d$
- จงคำนวณหาค่า Odds Ratio ของการเป็นโรคดังกล่าวโดยสูตรการคำนวณ คือ $OR = ad/bc$

บทที่ 2

Library ice

2.1 Library และ package ใน R

R เป็นสภาพการทำงานคล้าย shell ในระบบ UNIX หรือ LINUX ซึ่งมีลักษณะพิเศษคือมีคำสั่ง (function) ที่สนับสนุนการทำงานทางสถิติและกราฟิกอย่างมากมาย จึงถูกนำมาใช้เป็นเครื่องมือในการคำนวณทางสถิติดังเช่นที่จะได้แนะนำในที่นี้ คำสั่งต่างๆ ถูกบรรจุใน package หรือถ้าหากจะมองเป็นกลุ่มคำสั่งที่บรรจุรวมกัน รวมทั้งตัวอย่าง ความช่วยเหลือ และอื่นๆ รวมกันทั้งหมดแล้ว ก็จะเรียกเป็น library ซึ่ง package หลักที่จะถูกเรียกทุกครั้งเมื่อเรียกใช้ R สามารถเรียกดูได้ด้วยคำสั่ง `search()` ดังนี้

```
> search()
 [1] ".GlobalEnv"        "package:methods"   "package:stats"
 [4] "package:graphics"  "package:grDevices" "package:utils"
 [7] "package:datasets"  "package:base"      "Autoloads"
```

มี 6 package ที่ถูกเรียกเมื่อ **R** เริ่มทำงาน คือ

```
package:methods
package:stats
package:graphics
package:grDevices
package:utils
package:datasets
package:base
```

คำสั่งในการทำงานส่วนใหญ่จะอยู่ใน package base การทดสอบทางสถิติอยู่ใน package stats ส่วน package อื่นๆ ทำหน้าที่เฉพาะต่างๆ ตามชื่อของ package เหล่านั้น พึงสังเกตว่าหาก package เหล่านี้ไม่ถูกเรียกใช้งานตอนเริ่มต้นการใช้งาน **R** หรือถูกลบไปจากหน่วยความจำขณะทำงานไม่ว่าจะด้วยความตั้งใจหรือไม่ก็ตาม **R** ก็จะทำงานไม่ถูกต้องหรือไม่สามารถทำงานได้ package จึงเป็นส่วนสำคัญยิ่งในการทำงานของ **R**

library อยู่ใน `"C:\Program Files\R\r-2.4.0\library"` ลองเข้าไปดูในโฟลเดอร์นี้จะพบ library จำนวนมาก มากกว่า 6 package ที่ถูกเรียกใช้งานตั้งแต่เริ่มเรียกใช้ **R** ดังที่กล่าวแล้ว ในแต่ละ library มีแฟ้มสำคัญดังนี้

CONTENTS	
DESCRIPTION	คำอธิบาย library
INDEX	

นอกจากนี้ยังอาจมีแฟ้มอื่นได้แก่

NAMESPACE	คำสั่งที่อนุญาติให้เรียกใช้ได้ใน library
CITATION	การอ้างอิง

โฟลเดอร์ย่อยดังนี้ (ในแต่ละ library อาจมีแฟ้มไม่ครบในทุกโฟลเดอร์ก็ได้)

data	บรรจุแฟ้มข้อมูลตัวอย่าง
demo	บรรจุแฟ้มสาธิตการทำงาน
help	บรรจุแฟ้มข้อความช่วยเหลือในรูปแบบ text เป็นภาษาอังกฤษ
html	บรรจุแฟ้มข้อความช่วยเหลือที่เปิดได้โดย web browser
latex	บรรจุแฟ้มข้อความช่วยเหลือในรูปแบบ Latex
man	บรรจุแฟ้มข้อความช่วยเหลือในรูปแบบ Rd
R	บรรจุแฟ้มคำสั่ง
R-ex	บรรจุแฟ้มตัวอย่างการเรียกใช้งาน

จะเห็นได้ว่า library มีสิ่งอื่นๆ อีกมากมายนอกจากชุดคำสั่งในการทำงาน ที่เรียกว่า package แต่โดยทั่วไปก็มักเรียกรวมกันไปหมดว่า library โดยเป็นที่เข้าใจว่าเป็นทั้งชุดคำสั่ง และอื่นๆ ที่เกี่ยวข้องนั่นเอง

พึงเข้าใจว่ารูปแบบของ library ใน **R** รุ่นก่อน 2.0.0 นั้นแตกต่างไปพอสมควรจากรุ่น 2.0.0 จนใช้งานกันไม่ได้ ดังนั้น library **ice** ที่จะกล่าวต่อไปนี้ซึ่งได้สร้างขึ้นเพื่อใช้กับ **R** 2.0.0 เป็นต้นไป จึงใช้กับ **R** รุ่นก่อนหน้านี้ไม่ได้

2.2 Library ice

library **ice** ถูกสร้างขึ้นใช้ในหน่วยระบาดวิทยา มหาวิทยาลัยสงขลานครินทร์ โดยได้รับการสนับสนุนจากสำนักงานกองทุนสนับสนุนการวิจัย (สกว) ผ่านทางมูลนิธิสาธารณสุขแห่งชาติ (มสช) โดยมีวัตถุประสงค์เพื่อสร้างคำสั่งใน **R** เพิ่มเติมให้ใช้งานได้สะดวกกับการวิเคราะห์งานวิจัยทางระบาดวิทยาและชีวสถิติ โดยยึดถือหลักการพื้นฐานคือให้ใช้งานได้ง่าย และไม่จำเป็นต้องรู้กระบวนการขั้นตอนการทำงานของ **R** มากนักก็ยังสามารถทำงานได้

ดังนั้น ในการใช้งาน library **ice** จึงขอให้ผู้ใช้ทำตามกรอบการทำงานบางอย่างดังต่อไปนี้ ก็จะสามารถใช้งานได้อย่างเป็นขั้นตอน ตรวจสอบได้ ทำซ้ำได้ และไม่สับสน

ทำงานกับกรอบข้อมูลทีละหนึ่งกรอบข้อมูล แม้ว่า **R** จะไม่มีข้อจำกัดว่าจะต้องทำงานกับกรอบข้อมูลทีละหนึ่งกรอบข้อมูล แต่ผู้ใช้มักจะสับสนเมื่อเปิดใช้งานกรอบข้อมูลหลายกรอบข้อมูลในคราวเดียวกัน และลืมไปว่าขณะนี้กำลังทำงานกับกรอบข้อมูลไหนอยู่ ทั้งนี้เนื่องจากหลายครั้ง ชื่อตัวแปรในกรอบข้อมูลต่างกันก็มีชื่อเดียวกัน จึงทำให้สับสนได้ง่ายมาก

อย่าสร้างตัวแปรใดนอกกรอบข้อมูลที่กำลังทำงาน ในการสร้างตัวแปรใหม่ทุกครั้ง ขอให้สร้างโดยใช้คำสั่ง **`transform()`** ซึ่งจะทำให้ตัวแปรที่สร้างขึ้นใหม่มีจำนวน record เท่ากับ record ที่มีอยู่แล้วในกรอบข้อมูลเดิม และนำมาคำนวณร่วมกับตัวแปรอื่นๆ ในกรอบข้อมูลเดิมได้ทันที หรือหากจะสร้างด้วยคำสั่งอื่นที่ทำงานกับตัวแปรโดยตรง ให้ระบุชื่อตัวแปรที่รับผลโดยใส่ชื่อกรอบข้อมูลที่ต้องการบรรจุตัวแปรใหม่นั้นไว้ด้วย ในรูปแบบ `dataframe$variable <-`

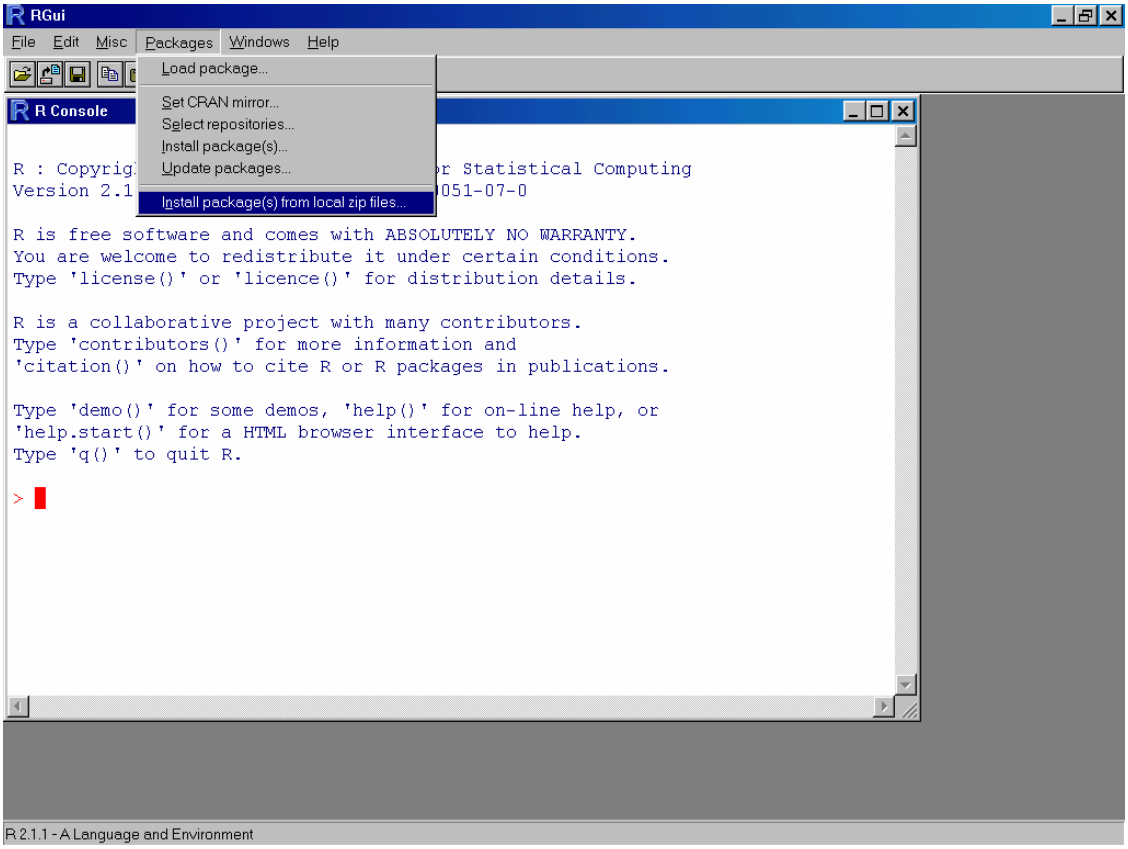
หากต้องการเปลี่ยนแปลงใดๆ กับข้อมูลเดิม ให้สร้างเป็นตัวแปรใหม่ อย่าตั้งชื่อซ้ำกับตัวแปรเดิม ทั้งนี้จะได้ประโยชน์คือ จะรักษาข้อมูลเดิมไว้ได้ ซึ่งหากจะยกเลิกการเปลี่ยนแปลงนั้นๆ ก็เพียงลบตัวแปรที่สร้างขึ้นจากตัวแปรเดิมนั้น แล้วสร้างใหม่ด้วยเงื่อนไขใหม่ เช่นหากต้องการแบ่งช่วงค่าตัวแปร ก็สร้างตัวแปรใหม่ขึ้นมาเพื่อกำหนดการแบ่งช่วง และหากต้องการยกเลิกการแบ่งช่วงเดิมไปใช้การแบ่งช่วงใหม่ ก็ทำได้ง่ายโดยการทำซ้ำจากตัวแปรเดิมที่ยังคงอยู่นั่นเอง

หากต้องการแก้ไขข้อมูลดิบใดๆ ให้แก้ไขให้เสร็จก่อนอ่านข้อมูลดิบเข้ามาวิเคราะห์ใน **R** ข้อมูลที่อ่านเข้ามาใน **R** แล้ว ห้ามบันทึกกลับไปทับแฟ้มเดิม ทั้งนี้เพื่อรักษาหลักฐานของข้อมูลเดิมไว้ตามหลักการของการเก็บข้อมูลวิจัยที่ดี ซึ่งหากยึดตามหลักนี้ การวิเคราะห์ทางสถิติจะเป็นการทำงานทางเดียว คือ อ่านข้อมูลเข้ามา --> ตรวจสอบข้อมูล --> ดัดแปลงข้อมูลให้เหมาะสมกับการวิเคราะห์ --> วิเคราะห์ --> ผลการวิเคราะห์

ทุกครั้งที่ทำการวิเคราะห์ข้อมูล ให้สร้างและบันทึกแฟ้มชุดคำสั่ง **R** script ที่ใช้วิเคราะห์ข้อมูลเก็บไว้ ซึ่งจะนำแฟ้มนั้นมาตรวจสอบความถูกต้อง (ด้วยตัวเองหรือให้ผู้รู้ตรวจสอบได้) หรือทำซ้ำใหม่ได้ตามต้องการ

2.3 การติดตั้ง library ice

ในแผ่น CD ชื่อ R-2.4.0-ICE จะมีแฟ้ม R-2.4.0-win32.exe เพื่อติดตั้ง **R** รุ่น 2.4.0 สำหรับ Windows และแฟ้ม ice_1.0-4.zip เพื่อติดตั้ง library **ice** ที่มีข้อความช่วยเหลือเป็นภาษาไทย ในการติดตั้ง library **ice** ให้ดับเบิ้ลคลิกไอคอน R เพื่อเปิดโปรแกรมขึ้นมา จากนั้น เลือกเมนู Packages -> Install package(s) from local zip files ดังรูปที่ 2.1 จากนั้นเลือกไฟล์ ice_1.0-4.zip จากแผ่น CD ใน Folder ../R-ICE/Thai/ice_1.0-4/



รูปที่ 2.1 การเลือกเมนูสำหรับการ Install library ice

2.4 คำสั่งที่มีอยู่ใน library ice รุ่น 1.0-4

ใน library **ice** 1.0-4 มีคำสั่งอยู่จำนวนมาก แบ่งเป็นหมวดหมู่ได้ดังนี้

```
GENERAL
   clean()
   exit()
```

```
   li(...)
   pause()
FILE MANAGEMENT
   do(file, prompt.echo = ">>", from = 1, to = -1)
   fread(file, ...)
   fwrite(file, ...)
   logg(file, append=FALSE)
   output(source, file, append = FALSE, prompt = ">>")
DATA MANIPULATION
   as.na(data, x, code, append=get(".ec.append",.GlobalEnv), var.names)
   change(data, x, condition, value, append = get(".ec.append",
     .GlobalEnv), var.names)
   cjoin(data, x, sep = "", var.names)
   create(...)
   defactor(data, x, append = get(".ec.append",.GlobalEnv), var.names)
   expand(x, column, include = FALSE, retain = TRUE)
   label(data, x, labels, sep="", ordered = FALSE, append =
     get(".ec.append",.GlobalEnv), var.names)
   label(data, x, values, labels, sep="", ordered = FALSE, append =
     get(".ec.append",.GlobalEnv), var.names)
   label(data, x, label.table, sep = "", ordered = FALSE, append =
     get(".ec.append",.GlobalEnv), var.names)
   pack(data, select, FUN)
   recode(data, x, values, append = get(".ec.append",.GlobalEnv),
     var.names)
   recode(data, x, old, new, append = get(".ec.append",.GlobalEnv),
     var.names)
   recode(data, x, conv.table, append = get(".ec.append",.GlobalEnv),
     var.names)
   recode.na(data, x, code, append = get(".ec.append",.GlobalEnv),
     var.names)
   relabel(data, x, ref, level.order, append = get(".ec.append",
     .GlobalEnv), var.names)
   rename(data, x, names)
   shift(data, x, direction = "down", n, value = NA, append =
     get(".ec.append",env = .GlobalEnv), var.names)
   sort.data(data, ..., decreasing = FALSE)
   to.character(data, x, append=get(".ec.append",.GlobalEnv), var.names)
   to.factor(data, x, labels, ordered = FALSE, append = get
     (".ec.append", .GlobalEnv), var.names)
   to.numeric(data, x, append = get(".ec.append",.GlobalEnv), var.names)
   to.vector(data, x, append = get(".ec.append",.GlobalEnv), var.names)
DATA SUMMARY
   describe(data, datalabel = NULL, var.labels = NULL)
   display(data, subset, select, mode = "console")
   ftab(data, columns, pct = get(".ec.pct",.GlobalEnv),
     frame = get(".ec.frame",.GlobalEnv), na.included = FALSE,
     factor.labels = TRUE)
   list.rep(data, x, n = 2)
   summarize(object, subset, select, group, pct = get(".ec.pct",
     .GlobalEnv), frame = get(".ec.frame",.GlobalEnv))
   tab(x, freq = get(".ec.freq",.GlobalEnv), na.included=FALSE,
      factor.labels=TRUE)
   tab(x, y, group, row = get(".ec.row",.GlobalEnv), colm =
     get(".ec.colm",.GlobalEnv), test = get(".ec.test",.GlobalEnv),
     frame = get(".ec.frame",.GlobalEnv), na.included = FALSE,
      factor.labels = TRUE)
   xtab(data, key, columns, row = get(".ec.row",.GlobalEnv),
```

```
     colm = get(".ec.colm",.GlobalEnv), test = get(".ec.test",
     .GlobalEnv), position = get(".ec.position",.GlobalEnv),
     frame = get(".ec.frame",.GlobalEnv), na.included = FALSE,
     factor.labels = TRUE)
STATISTICAL TESTS
   anova.test(formula, data, subset, frame = get(".ec.frame",
     .GlobalEnv))
   ci(data, x, subset, group, level = 0.95, frame = get(".ec.frame",
     .GlobalEnv))
   ci(n, m, sd, level = 0.95, frame = get(".ec.frame",.GlobalEnv))
   lr.test(model1, model2)
   shapiro.wilk.test(x)
   shapiro.wilk.test(data, select, subset, group)
   st.test(data, x, y = NULL, mu = 0, paired = FALSE, var.equal = FALSE,
     conf.level = 0.95)
   st.test(formula, data, subset, frame = get(".ec.frame",.GlobalEnv))
   wilcoxon.test(data, x, y = NULL, mu = 0, paired=FALSE, exact = NULL,
     correct = TRUE, conf.int=FALSE, conf.level=0.95,
     frame = get(".ec.frame",.GlobalEnv))
   wilcoxon.test(formula, data, subset, frame = get(".ec.frame",
     .GlobalEnv))
STATISTICAL MODELS
   logistic(formula, data, subset, frame = get(".ec.frame",.GlobalEnv))
   logit(formula, data, subset, or = TRUE, frame = get(".ec.frame",
     .GlobalEnv))
MISCELLANEOUS
   spower(x, y, sd1, sd2, n1, test="twosided", sample = "two",
     alpha = 0.05, ratio = 1)
   ssize(x, y, sd1, sd2, test = "twosided", sample = "two",
     alpha = 0.05, power = 0.8, ratio = 1)
```

หมวดคำสั่งต่างๆ ของ library ice มีความหมายดังนี้

GENERAL เป็นกลุ่มคำสั่งที่ทำงานทั่วไป

FILE MANAGEMENT เป็นกลุ่มคำสั่งที่ทำงานเกี่ยวกับแฟ้มข้อมูล เช่น การอ่านและเขียนแฟ้มข้อมูล การเรียกใช้งานแฟ้ม **R** script และสร้างแฟ้ม log

DATA MANIPULATION เป็นกลุ่มคำสั่งที่ใช้ในการเปลี่ยนแปลงกรอบข้อมูล ตัวแปร และข้อมูลภายในนั้น

DATA SUMMARY เป็นกลุ่มคำสั่งที่ใช้ในการสรุปกรอบข้อมูล ตัวแปร ทำตาราง

STATISTICAL TESTS เป็นกลุ่มคำสั่งที่ใช้ในการทดสอบทางสถิติในรูปแบบต่างๆ

STATISTICAL MODEL เป็นกลุ่มคำสั่งที่ทำงานกับโมเดลทางสถิติ

MISCELLANEOUS เป็นคำสั่งอื่นๆ ที่ไม่เข้าหมวดข้างต้น

บทที่ 3

การสำรวจข้อมูล (Data exploration)

การพินิจข้อมูล หรือการสำรวจข้อมูล มีวัตถุประสงค์เพื่อการตรวจสอบดูลักษณะ และการแจกแจงของข้อมูล เพื่อตรวจสอบความถูกต้องของข้อมูลก่อนการนำไปวิเคราะห์ทางสถิติ คำสั่งที่ใช้สำหรับการสำรวจข้อมูลที่จะกล่าวถึงมี 2 ลักษณะคือ การสำรวจจข้อมูลจากการดูรายละเอียดตัวแปรและการดูจากตาราง และการสำรวจข้อมูลจากการใช้กราฟ

การสำรวจข้อมูลจากรายละเอียดตัวแปรและตาราง คำสั่งต่าง ๆ ที่ใช้คือ

`describe()`	การสรุปกรอบข้อมูลอย่างง่าย
`list.rep()`	การหาการซ้ำกันของข้อมูล
`display()`	การแสดงรายละเอียดข้อมูล
`sort.data()`	การเรียงลำดับข้อมูล
`summarize()`	การคำนวณค่าเฉลี่ย มัธยฐาน ส่วนเบี่ยงเบนมาตรฐาน
`ftab()`	การคำนวณความถี่ทางเดียว
`tab()`	การคำนวณความถี่สองทาง
`xtab()`	การคำนวณความถี่สองทางหลายตัวแปร

การสำรวจข้อมูลจากกราฟ คำสั่งที่ใช้คือ

`hist()`	การสร้างฮีสโตแกรม
`boxplot()`	การสร้าง boxplot
`plot()`	การสร้าง scatter plot
`stem()`	การสร้าง stem and leaf plot
`pairs()`	การสร้าง scatter plot matrix

ตัวอย่างข้อมูลที่ใช้ในบทนี้ เป็นข้อมูลจากการสำรวจความคิดเห็นต่อการเข้ารับการให้คำปรึกษาในการตรวจเลือดเพื่อหาเชื้อ HIV ด้วยความสมัครใจ (VCT) ในกลุ่มหญิงบริการในจังหวัดภูเก็ต กลุ่มตัวอย่างคือหญิงอาชีพขายบริการ ที่ทำงานอยู่ในบาร์เบียร์ อาบอบนวดทั้งแผนโบราณ และแผนปัจจุบัน ทั้งหมดจำนวน 315 คน ตัวอย่างแบบสอบถามเป็นดังนี้

แบบสำรวจความต้องการและความเต็มใจที่จะจ่ายสำหรับบริการให้คำปรึกษาและตรวจเลือดด้วยความสมัครใจ (วีซีที)

<u>คำแนะนำ: กรุณากากบาท (X) ในช่อง [ ] หน้าคำตอบที่คุณต้องการเลือก</u>

ส่วนที่ 1: *ข้อมูลทั่วไปของผู้ตอบแบบสอบถาม* ID[ ][ ][ ]

คำถาม	**สำหรับเจ้าหน้าที่กรอก**
1.ปัจจุบันคุณอายุ…….. ปี	A1 [ ] [ ]
2.คุณเรียนจบชั้นใด [ ] 1. ไม่ได้เรียน แต่อ่านออกเขียนได้ [ ] 2.ประถมศึกษา (ป.1– ป. 6) [ ] 3. มัธยมศึกษาตอนต้น (ม.1-ม.3) [ ] 4.มัธยมศึกษาตอนปลาย (ม.4-ม.6) [ ] 5. อาชีวศึกษาและระดับอนุปริญญา [ ] 6.ปริญญาตรี หรือสูงกว่า	A2 [ ]
3. ภูมิลำเนาเดิมคุณมาจากจังหวัด…….……....	A3 [ ] [ ]
4. คุณมาอยู่ที่ภูเก็ตนาน……....…… เดือน	A4 [ ] [ ] [ ]
5. คุณทำอาชีพขายบริการทางเพศมานาน…… …… เดือน	A5 [ ] [ ][ ]
6. รายได้ของคุณเดือนละประมาณ……..…… บาท	A6 [ ] [ ] [ ] [ ] [ ]

ส่วนที่ 2: *ความรู้ ประสบการณ์ และความต้องการเกี่ยวกับการรับคำปรึกษาและตรวจเลือดโดยความสมัครใจ (วีซีที)*

คำถาม	**สำหรับเจ้าหน้าที่กรอก**
7. คุณเคยไปรับการเจาะเลือดหาเชื้อเอชไอวีหรือไม่ [ ] 1. ไม่เคย **(ถ้าไม่เคยโปรดข้ามไปตอบข้อ 10)** [ ] 2. เคยและรู้ผล [ ] 3. เคยแต่ไม่มีใครบอกผล [ ] 4. เคยแต่ไม่ต้องการรู้ผล	A7 [ ]
8. คุณได้รับคำแนะนำจากเจ้าหน้าที่เกี่ยวกับการป้องกันโรคเอดส์ ในวันที่ไปเจาะเลือดหรือไม่ [ ] 1. ไม่ได้รับ [ ] 2. ได้รับ	A8 [ ]
9. ถ้าไม่มีใครบังคับ คุณยังต้องการเจาะเลือดเพื่อหาเชื้อเอชไอวีหรือไม่ [ ] 1. ไม่ต้องการ [ ] 2. ต้องการ	A9 [ ]
10. คุณเคยรู้ว่ามีบริการให้คำปรึกษาพร้อมกับตรวจเลือดหรือไม่ [ ] 1. ไม่เคยรู้จัก [ ] 2. รู้จัก แต่ไม่เคยไปใช้บริการ [ ] 3. รู้จัก และเคยไปใช้บริการ	A10 [ ]
11. คุณต้องการไปรับ *บริการให้คำปรึกษาและตรวจเลือดโดยความสมัครใจ (วีซีที)* หรือไม่ [ ] 1. ไม่ต้องการ **(ข้ามไปตอบข้อ 13)** [ ] 2. ต้องการ	A11 [ ]
12. ถ้าต้องการคุณคิดว่าเวลาที่เหมาะสมสำหรับการไปรับ *บริการให้คำปรึกษาและตรวจเลือดโดยความสมัครใจ (วีซีที)* คือ [ ] 1. 8.00–16.00 น. [ ] 2. 16.00-20.00 น. [ ] 3. 20.00-24.00 น.	A12 [ ]

ส่วนที่ 3: *ความเต็มใจที่จะจ่ายสำหรับบริการคำปรึกษาและตรวจเลือดด้วยความสมัครใจ (วีซีที)*

คำถาม	**สำหรับเจ้าหน้าที่กรอก**
13. ค่าใช้จ่ายสูงสุดสำหรับการรับคำปรึกษาและตรวจเลือดที่คุณยินดีและสามารถจ่ายได้คือ …….. บาท	A13 [ ] [ ] [ ] [ ]

3.1 การสำรวจข้อมูลจากรายละเอียดตัวแปรและตาราง

3.1.1 คำสั่ง ***describe()***

เป็นคำสั่งสำหรับการสรุปกรอบข้อมูลอย่างย่อ หรือใส่คำอธิบายให้กับกรอบข้อมูล และ/หรือ ตัวแปรภายใน มีรูปแบบการออกคำสั่งดังนี้

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

`data`	กรอบข้อมูล
`datalabel`	ตัวเลือกเพื่อใส่คำอธิบายกรอบข้อมูล
`var.labels`	เวกเตอร์ตัวเลือกเพื่อใส่คำอธิบายตัวแปรในกรอบข้อมูล

ถ้าออกคำสั่ง **`describe()`** โดยไม่ส่งค่ากลับสู่วัตถุใด และไม่ระบุตัวเลือก `datalabel` หรือ `var.labels` จะเป็นคำสั่งเพื่อดูตัวแปรที่อยู่ภายในกรอบข้อมูล `x` บางครั้งเมื่อเราทำงานกับกรอบข้อมูลหนึ่งมาระยะหนึ่งแล้ว อาจลืมไปว่าชื่อตัวแปรภายในนั้นเป็นอย่างไร คำสั่ง **`summarize()`** หรือ **`summary()`** หรือคำสั่งอื่นๆ จะแสดงผลมากเกินความจำเป็น หรือในการทำงานกับคำสั่ง **`ftab()`** และ **`xtab()`** ซึ่งจะกล่าวต่อไป เราอาจต้องการดูเลขลำดับของตัวแปรภายในกรอบข้อมูล คำสั่งนี้ **`describe()`** จะให้ผลสั้นๆ เท่าที่ต้องการ ดังตัวอย่าง

```
> clean()
> library(ice)
> vct <- fread("vcttest.dta")
> describe(vct)
No. of observation = 318
   Variable   Description
1  qid        qid  question number
2  a1         a1    how old are you?
3  a2         a2    education
4  a3         a3    province
5  a4         a4    how long in phuket?
6  a5         a5    how long worked ?
7  a6         a6    how much earn each month?
8  a7         a7    ever had hiv tested?
9  a8         a8    received any counselling?
10 a9         a9    if unforced, wish to know?
11 a10        a10   know about vct before?
12 a11        a1ı   after vdo,want to go vct?
13 a12        a1ı   the suitable time of day?
14 a13        a13   highest cost
```

3.1.2 คำสั่ง ***list.rep()***

เป็นคำสั่งสำหรับการหาการซ้ำกันของข้อมูลในเวกเตอร์ มีรูปแบบการออกคำสั่งดังนี้

```
list.rep(data, x, n=2)
```

`data`	กรอบข้อมูล
`x`	เวกเตอร์หรือตัวแปร
`n`	ตัวเลือกระบุจำนวนการซ้ำที่ต้องการให้หา ปกติตั้งไว้ที่มากกว่าหรือเท่ากับ 2

ปกติในข้อมูลวิจัยชุดหนึ่ง ๆ จะมีตัวแปรที่ใช้ระบุหมายเลขวิจัยที่ไม่ซ้ำกันเลย แต่บางครั้ง ในกระบวนการกรอกข้อมูล อาจมีการกรอกข้อมูลซ้ำ หรือในการให้หมายเลขวิจัย อาจมีการให้หมายเลขซ้ำได้เช่นเดียวกัน ในโปรแกรมที่ใช้กรอกข้อมูลที่ดีจะมีการป้องกันข้อผิดพลาดนี้อยู่แล้ว แต่ทั้งนี้ก็ขึ้นกับผู้ออกแบบกระบวนการกรอกข้อมูล ซึ่งอาจไม่ได้ป้องกันการกรอกข้อมูลซ้ำก็ได้ ใน library **ice** จึงได้สร้างคำสั่ง **`list.rep()`** ไว้ให้ตรวจสอบหมายเลขซ้ำไว้ด้วย ดังตัวอย่างการใช้งานดังนี้

```
> list.rep(vct,qid)
    rep
100   2
120   2
197   2
```

ผลจากคำสั่งดังกล่าว จะเห็นได้ว่าในชุดข้อมูล `vct` มีตัวแปร `qid` ระบุหมายเลขวิจัยที่ซ้ำกัน 2 ครั้ง จำนวน 3 ค่า คือ หมายเลข 100 120 และหมายเลข 197 ดังนั้นจึงต้องตรวจสอบข้อมูลเพื่อให้แน่ใจว่าเป็นข้อมูลที่ซ้ำกันจริงหรือไม่ หากซ้ำกันจริงจึงต้องตัดข้อมูลที่ป้อนซ้ำออกก่อนการนำไปวิเคราะห์ทางสถิติในขั้นถัดไป คำสั่งนี้จึงมีประโยชน์เป็นอย่างมาก ทำให้มั่นใจได้ว่าชุดข้อมูลที่มีอยู่นั้น สามารถนำไปวิเคราะห์ต่อได้ และน่าจะให้ผลการวิเคราะห์ถูกต้องตามที่ได้ออกแบบการวิจัยไว้

3.1.3 คำสั่ง ***display()***

เป็นคำสั่งที่ใช้สำหรับเรียกดูข้อมูลในกรอบข้อมูล มีรูปแบบการออกคำสั่งดังนี้

```
display(data, ..., select, mode = "console")
```

`data`	กรอบข้อมูล
`...`	เงื่อนไขในการเลือกแสดงข้อมูล
`select`	เวกเตอร์ระบุตัวแปรที่จะแสดง
`mode`	เงื่อนไขระบบให้แสดงบนจอ “console” หรือบน “editor”

คำสั่งนี้ทำงานคล้ายคำสั่ง **`subset()`** ซึ่งจะกล่าวถึงในบทถัดไป แต่จุดต่างที่สำคัญคือคำสั่ง **`display()`** สามารถรับลำดับตัวแปรเป็นตัวเลขหรือชื่อก็ได้ เช่น `c(1:5)` หรือ `c(qid, a1)` และสามารถเรียกใช้ editor ได้จากคำสั่งในทันที หากเลือก `mode = "editor"`

จากข้อมูล `vct` หากต้องการดูรายละเอียดข้อมูลในแต่ละชุด เพื่อการตรวจสอบว่าซ้ำกันหรือไม่ โดยการพิมพ์คำสั่งต่อไปนี้

```
> display(vct, subset = c(qid==100 | qid==120 | qid==197))
    qid a1 a2 a3 a4 a5    a6 a7 a8 a9 a10 a11 a12 a13
99  100 26  2  3 29 29  5000  2  1  2   3   2   2 500
118 120 27  2  2  1  1 15000  1 NA NA   1   2   1 500
193 197 25  2  2  2  2  6000  1 NA NA   1   2   1 100
316 100 26  2  3 29 29  5000  2  1  2   3   2   2 500
317 120 27  2  2  1  1 15000  1 NA NA   1   2   1 500
318 197 25  2  2  2  2  6000  1 NA NA   1   2   1 100
```

คำสั่งโดยการเรียกตัวแปรให้อยู่ในวงเล็บสี่เหลี่ยม จะแสดงผลเช่นเดียวกับคำสั่ง **`display()`** ดังตัวอย่างต่อไปนี้

```
> vct[qid==100 | qid==120 | qid==197,]
    qid a1 a2 a3 a4 a5    a6 a7 a8 a9 a10 a11 a12 a13
99  100 26  2  3 29 29  5000  2  1  2   3   2   2 500
118 120 27  2  2  1  1 15000  1 NA NA   1   2   1 500
193 197 25  2  2  2  2  6000  1 NA NA   1   2   1 100
316 100 26  2  3 29 29  5000  2  1  2   3   2   2 500
317 120 27  2  2  1  1 15000  1 NA NA   1   2   1 500
318 197 25  2  2  2  2  6000  1 NA NA   1   2   1 100
```

3.1.4 คำสั่ง `sort.data()`

เป็นคำสั่งสำหรับการเรียงลำดับของข้อมูล เป็นคำสั่งที่อยู่ใน library **ice** มีรูปแบบการออกคำสั่งดังนี้

```
sort.data(data, ..., decreasing = FALSE)
```

`data`	กรอบข้อมูล
`...`	ตัวแปรในกรอบข้อมูลที่ต้องการให้ทุกตัวแปรเรียงลำดับตามตัวแปรนั้นๆ
`decreasing`	ค่าตรรกะ เรียงข้อมูลเพิ่มขึ้น หรือลดลง ค่าปกติคือเพิ่มขึ้น (decreasing = FALSE)

ส่วนใหญ่แล้วจะเรียงลำดับโดยใช้ ID แต่ในบางครั้งการทำงานผู้ใช้อาจต้องการเรียงลำดับข้อมูลตามตัวแปรต่าง ๆ

จากข้อมูล `vct` ที่ได้ใช้คำสั่ง **`display()`** แล้วปรากฏว่าตัวแปร `id` ไม่ได้เรียงลำดับ หากต้องการเรียงลำดับใหม่ เพื่อให้ง่ายต่อการพิจารณาข้อมูล สามารถทำได้โดยการใช้คำสั่ง **sort.data()** ดังต่อไปนี้

```
> vct2 <- sort.data(vct, qid)
> vct2[qid==100 | qid==120 | qid==197,]
    qid a1 a2 a3 a4 a5    a6 a7 a8 a9 a10 a11 a12 a13
99  100 26  2  3 29 29  5000  2  1  2   3   2   2 500
100 100 26  2  3 29 29  5000  2  1  2   3   2   2 500
119 120 27  2  2  1  1 15000  1 NA NA   1   2   1 500
120 120 27  2  2  1  1 15000  1 NA NA   1   2   1 500
195 197 25  2  2  2  2  6000  1 NA NA   1   2   1 100
196 197 25  2  2  2  2  6000  1 NA NA   1   2   1 100
```

จากคำสั่งข้างต้น เรียงลำดับข้อมูลตามตัวแปร `qid` แล้วเก็บไว้ในกรอบข้อมูลชื่อใหม่ คือ `vct2` ทั้งนี้เพื่อไม่ต้องการให้มีการเปลี่ยนแปลงในกรอบข้อมูล `vct` เดิม และหากทำงานผิดพลาด ก็จะสามารถกลับไปเรียกกรอบข้อมูลเดิมมาใช้ใหม่ได้ จะเห็นได้ว่ารหัส 100 120 และ 197 ถูกป้อนซ้ำกันจริง จึงควรตัดตัวที่ซ้ำออก เช่นต้องการลบ `qid` ที่อยู่ในแถวที่ 100, 120 และ 196 ด้วยการใส่เครื่องหมายลบ (-) หน้าหมายเลขแถวที่ต้องการลบทิ้ง ดังคำสั่งต่อไปนี้

```
> vct2 <- vct2[-c(100,120,196),]
```

3.1.5 คำสั่ง `summarize()` หรือ `summ()`

เป็นคำสั่งสำหรับการสรุปข้อมูลอย่างสั้น ๆ ซึ่งมีรูปแบบการออกคำสั่งดังนี้

```
summarize(object, subset, select, group, pctile =
   get.ec.option("pctile"), frame = get.ec.option("frame"))
```

`object`	กรอบข้อมูล เวกเตอร์ แฟกเตอร์ หรือวัตถุชนิดอื่น
`subset`	กลุ่มย่อยของตัวแปร ถ้า 'object' เป็นกรอบข้อมูล
`select`	ตัวเลือกเงื่อนไข `group` เวกเตอร์หรือแฟกเตอร์กำหนดการแบ่งกลุ่ม
`pctile`	แสดงเป็น percentile ค่าปกติเป็น FALSE ซึ่งเป็นการแสดงอย่างย่อ
`frame`	วาดกรอบให้กับตาราง ค่าปกติเป็น TRUE

ตัวอย่างการใช้งานแสดงผลของคำสั่ง **`summarize()`** กรอบข้อมูล ซึ่งจะต่างจากผลของ **`summary()`** ถ้าวัตถุที่ส่งให้กับคำสั่ง **`summarize()`** เป็นกรอบข้อมูลเช่นนี้ ตัวเลือก `strata` และ `pct` จะไม่ทำงาน ดังนี้

```
> summarize(vct2)
```

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

```
> summary(vct2)
      qid              a1              a2              a3
 Min.   :  1.0   Min.   :17.00   Min.   :1.000   Min.   :1.000
 1st Qu.: 80.5   1st Qu.:24.00   1st Qu.:2.000   1st Qu.:2.000
 Median :162.0   Median :27.00   Median :3.000   Median :2.000
 Mean   :161.3   Mean   :28.63   Mean   :3.175   Mean   :2.416
 3rd Qu.:242.5   3rd Qu.:32.00   3rd Qu.:4.000   3rd Qu.:3.000
 Max.   :323.0   Max.   :99.00   Max.   :9.000   Max.   :9.000

       a4               a5              a6              a7
 Min.   :  0.00   Min.   :  0.0   Min.   :     0   Min.   :1.000
 1st Qu.:  3.00   1st Qu.:  2.0   1st Qu.:  5000   1st Qu.:1.000
 Median : 10.00   Median :  9.0   Median :  7000   Median :2.000
 Mean   : 30.03   Mean   : 52.4   Mean   : 18969   Mean   :1.768
 3rd Qu.: 33.50   3rd Qu.: 24.0   3rd Qu.: 15000   3rd Qu.:2.000
 Max.   :999.00   Max.   :999.0   Max.   : 99999   Max.   :9.000

       a8              a9              a10             a11
 Min.   : 1.000   Min.   : 1.000   Min.   :1.000   Min.   :1.000
 1st Qu.: 1.000   1st Qu.: 2.000   1st Qu.:1.000   1st Qu.:2.000
 Median : 2.000   Median : 2.000   Median :2.000   Median :2.000
 Mean   : 1.798   Mean   : 2.028   Mean   :2.102   Mean   :1.787
 3rd Qu.: 2.000   3rd Qu.: 2.000   3rd Qu.:3.000   3rd Qu.:2.000
 Max.   : 9.000   Max.   : 9.000   Max.   :9.000   Max.   :2.000
 NA's   :97.000   NA's   :97.000
      a12             a13
 Min.   : 1.000   Min.   :  30.0
 1st Qu.: 1.000   1st Qu.: 100.0
 Median : 1.000   Median : 200.0
 Mean   : 1.438   Mean   : 733.9
 3rd Qu.: 2.000   3rd Qu.: 300.0
 Max.   : 9.000   Max.   :9999.0
 NA's   :66.000   NA's   :   1.0
```

จะเห็นว่าคำสั่ง **summarize()** จะให้ผลอย่างสั้น ๆ และอ่านง่าย ในขั้นแรกเมื่อนำเข้าข้อมูลมาจากภายนอกจะใช้คำสั่งนี้เพื่อดูช่วงค่าของข้อมูลภายในตัวแปรแต่ละตัวก่อน ในที่นี้จะเห็นว่าค่าสูงสุดของ `a1`, `a2`, `a3`, `a4`, `a5`, `a6`, `a7`, `a8`, `a9`, `a10`, `a12`, และ `a13` เป็น 99, 9, 9, 999, 999, 99999, 9, 9, 9, 9 และ 9999 ตามลำดับ ซึ่งเป็นรหัสของ missing ไม่ใช่ค่าจริง ในการคำนวณทางสถิติเรามักจะต้องเปลี่ยนค่า missing เหล่านี้เป็น NA เสียก่อน โดยจะกล่าวถึงในบทที่ 4

เราอาจสรุปตัวแปรแต่ละตัวด้วยคำสั่ง **summarize()** และในกรณีเช่นนี้ เราอาจแบ่งสรุปตัวแปรตามกลุ่มของข้อมูลอื่นที่เป็นระดับชั้นได้ ลองดูการกระจายของข้อมูล `a1` แบ่งตามระดับของ `a2` ดังนี้

```
> summarize(a1)
 ----------------------------------------------------------------------
   |   Obs. |     NA |   Mean |   S.D. |   Min. | Median |   Max.
 ---+--------+--------+--------+--------+--------+--------+--------
 a1 |    315 |      0 |  28.61 |   7.66 |     17 |     27 |     99
 ----------------------------------------------------------------------
> summarize(a1, group=a2)
```

```
a1 stratified by a2
 ------------------------------------------------------------------
   |   Obs. |     NA |   Mean |   S.D. |   Min. | Median |   Max.
 --+--------+--------+--------+--------+--------+--------+--------
 1 |      5 |      0 |   26.6 |   5.55 |     20 |     26 |     34
 2 |    108 |      0 |  32.73 |   9.78 |     18 |   30.5 |     99
 3 |    100 |      0 |  26.25 |   5.33 |     17 |     25 |     41
 4 |     58 |      0 |  26.83 |    5.1 |     19 |     26 |     40
 5 |     31 |      0 |  26.58 |   5.49 |     20 |     26 |     40
 6 |      8 |      0 |     27 |   3.12 |     24 |     26 |     32
 8 |      1 |      0 |     36 |     NA |     36 |     36 |     36
 9 |      4 |      0 |   23.5 |    4.2 |     19 |     23 |     29
 ------------------------------------------------------------------
```

ในบางครั้ง เราอาจต้องการดูการกระจายของข้อมูลอย่างเร็วๆ เช่น เพื่อจะหาจุดตัดข้อมูลที่เหมาะสมในการแปลงตัวแปรต่อเนื่องให้เป็นตัวแปรแบบ ordinal เรามักตัดช่วงตัวแปรตามค่า quartile หรือ percentile นั่นเอง ตัวเลือก `pct` ในที่นี้จะช่วยให้ตัดสินใจได้เร็วขึ้น ดังนี้

```
> summarize(a1, pct=T, frame=F)
   Obs. NA Mean   S.D.  Min. 1pct  2pct 5pct 10pct 25pct Median 75pct 90pct
a1 315  0  28.632 7.693 17   18.14 19   20   21    24    27     32    38
   98pct 99pct Max.
a1 45    50.72 99
```

อย่างไรก็ตามการตัดตัวแปรต่อเนื่องให้เป็นตัวแปรกลุ่มนั้น ต้องจัดการกับค่า missing ก่อนเสมอ ดังจะกล่าวในบทที่ 4

3.1.6 คำสั่ง `ftab()`

เป็นคำสั่งสำหรับการทำตารางความถี่ของตัวแปรภายในกรอบข้อมูลได้ทีละหลายตัวแปรพร้อมกัน ซึ่งมีรูปแบบการออกคำสั่งดังนี้

```
ftab(data, columns, pct = get.ec.option("pct"), frame = get.ec.option
      ("frame"), na.included = FALSE, factor.labels = TRUE)
```

`data`	กรอบข้อมูล
`columns`	สดมภ์ต่างที่จะนำมาสร้างตารางความถี่หรือที่จะนำมาสร้างตารางความสัมพันธ์ไขว้กับตัวแปรหลัก key อาจระบุเป็นเวกเตอร์ของเลขลำดับสดมภ์หรือเวกเตอร์ของชื่อสดมภ์ก็ได้
`pct`	แสดงตารางร้อยละด้วย ค่าปกติเป็น FALSE

`frame`	วาดกรอบให้กับตาราง ค่าปกติเป็น TRUE
`na.included`	แสดง NA เป็นอีกระดับหนึ่งหรือไม่ ค่าปกติคือ FALSE
`factor.labels`	แสดงแฟกเตอร์โดยคำอธิบายระดับหรือค่า ค่าปกติคือแสดงโดยคำอธิบาย (TRUE)

ในการทำงานกับกรอบข้อมูลที่มีตัวแปรแบบกลุ่มจำนวนมาก ถ้าจะใช้คำสั่ง **`tab()`** กับตัวแปรเหล่านั้นทีละตัวจะเป็นการเสียเวลาในการพิมพ์คำสั่งมาก ใน library **ice** จึงได้มีคำสั่ง **`ftab()`** ไว้ให้ออกคำสั่งเพียงคำสั่งเดียวกับตัวแปรจำนวนมาก และในกรณีนี้ เราสามารถเรียกชื่อตัวแปรในกรอบข้อมูลได้โดยไม่ต้อง **`attach()`** แต่อย่างใด (เพราะบ่อยครั้งผู้ใช้จะลืม **`detach()`** กรอบข้อมูลนั้น แล้วเปลี่ยนไปทำงานกับกรอบข้อมูลอื่น หากมีตัวแปรชื่อซ้ำหรือคล้ายกันอยู่ อาจไปเรียกตัวแปรผิดมาทำงานได้) ดังตัวอย่าง

```
> summarize(vct2)
```

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(vct2, c(a2:a3, a7:a12), pct=T)
a2
 ----------------------------
 Label    |    Freq | Percent
 ---------+---------+---------
 1        |       5 |    1.59
 2        |     108 |   34.29
 3        |     100 |   31.75
 4        |      58 |   18.41
 5        |      31 |    9.84
 6        |       8 |    2.54
 8        |       1 |    0.32
 9        |       4 |    1.27
 ---------+---------+---------
 Total    |     315 |  100.01
 ----------------------------

xa3
 ----------------------------
 Label    |    Freq | Percent
 ---------+---------+---------
 1        |      61 |   19.37
 2        |     153 |   48.57
 3        |      53 |   16.83
 4        |      39 |   12.38
 8        |       2 |    0.63
 9        |       7 |    2.22
 ---------+---------+---------
 Total    |     315 |  100.00
 ----------------------------

a7
 ----------------------------
 Label    |    Freq | Percent
 ---------+---------+---------
 1        |     101 |   32.06
 2        |     206 |   65.40
 3        |       3 |    0.95
 4        |       2 |    0.63
 9        |       3 |    0.95
 ---------+---------+---------
 Total    |     315 |   99.99
 ----------------------------

a8
 ----------------------------
 Label    |    Freq | Percent
 ---------+---------+---------
 1        |      65 |   29.82
 2        |     150 |   68.81
 9        |       3 |    1.38
 ---------+---------+---------
 Total    |     218 |  100.01
 ----------------------------
```

```
a9
 ----------------------------
 Label    |    Freq | Percent
 ---------+---------+---------
 1        |      22 |   10.09
 2        |     192 |   88.07
 9        |       4 |    1.83
 ---------+---------+---------
 Total    |     218 |   99.99
 ----------------------------

a10
 ----------------------------
 Label    |    Freq | Percent
 ---------+---------+---------
 1        |      85 |   26.98
 2        |     125 |   39.68
 3        |     103 |   32.70
 9        |       2 |    0.63
 ---------+---------+---------
 Total    |     315 |   99.99
 ----------------------------

a11
 ----------------------------
 Label    |    Freq | Percent
 ---------+---------+---------
 1        |      67 |   21.27
 2        |     248 |   78.73
 ---------+---------+---------
 Total    |     315 |  100.00
 ----------------------------

a12
 ----------------------------
 Label    |    Freq | Percent
 ---------+---------+---------
 1        |     179 |   71.89
 2        |      54 |   21.69
 3        |      12 |    4.82
 8        |       1 |    0.40
 9        |       3 |    1.20
 ---------+---------+---------
 Total    |     249 |  100.00
 ----------------------------
```

หมายเหตุ

ขอให้สังเกตว่าเราสามาถเรียกตัวแปรเป็นช่วง โดยใช้เครื่องหมาย : (colon) ได้ด้วย ทำให้สะดวกในการทำงานอย่างมาก โดยที่เราใช้คำสั่ง **`describe()`** หรือ **`summarize()`** เพื่อดูลำดับของตัวแปรในกรอบข้อมูลเสียก่อน ในการเรียกตัวแปรนั้น อาจเรียกด้วยชื่อ เช่น `c(a7:a12)` หรือเรียกด้วยเลขลำดับตัวแปรก็ได้ เช่น `c(8:13)`

3.1.7 คำสั่ง `tab()`

เป็นคำสั่งสำหรับสร้างตารางพร้อมด้วยตารางร้อยละด้านแถวและสดมภ์และการทดสอบนัยสำคัญทางสถิติ ซึ่งมีรูปแบบการออกคำสั่งดังนี้

```
tab(data, x, y, group, row = get.ec.option("row"), colm =
      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 ถ้าระบุแต่ 'x' เพียงค่าเดียวจะสร้างตารางความถี่
`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
`frame`	วาดกรอบให้กับตาราง ค่าปกติเป็น TRUE
`na.included`	แสดง NA เป็นอีกระดับหนึ่งหรือไม่ ค่าปกติคือ FALSE
`factor.labels`	แสดงแฟกเตอร์โดยคำอธิบายระดับหรือค่า ค่าปกติคือแสดงโดยคำอธิบาย (TRUE)

คำสั่งนี้หากไม่ระบุตัวเลือกใด ๆ จะให้ผลคล้ายคำสั่ง table นั่นเอง แต่จะรับตัวแปรที่จะนำมาทำตารางเพียงไม่เกินสองตัวแปรเท่านั้น แต่ถ้าระบุตัวเลือกจะเห็นประโยชน์ของการสรุปข้อมูลอย่างรวดเร็วในการดูร้อยละด้านแถวหรือสดมภ์ รวมทั้งให้ผลการทดสอบนัยสำคัญด้วยวิธี chi-squared และ Fisher's exact test ได้ในคำสั่งเดียว ดังนี้

```
> tab(a2,a11,row=TRUE,test=TRUE)
```

```
row: a2, column: a11
 ------------
   |  1 |  2
 --+----+----
 1 |  1 |  4
 2 | 23 | 85
 3 | 17 | 83
 4 | 10 | 48
 5 | 12 | 19
 6 |  3 |  5
 8 |  0 |  1
 9 |  1 |  3
 ------------
<row percent>
 --------------------------------
   |      X1 |      X2 |   Total
 --+---------+---------+---------
 1 |   20.00 |   80.00 |  100.00
 2 |   21.30 |   78.70 |  100.00
 3 |   17.00 |   83.00 |  100.00
 4 |   17.24 |   82.76 |  100.00
 5 |   38.71 |   61.29 |  100.00
 6 |   37.50 |   62.50 |  100.00
 8 |    0.00 |  100.00 |  100.00
 9 |   25.00 |   75.00 |  100.00
 --------------------------------
Chi-squared = 8.848, df = 7, p-value = 0.2638
```

3.1.8 คำสั่ง `xtab()`

เป็นคำสั่งสำหรับการทำตารางความสัมพันธ์ไขว้ของตัวแปรภายในกรอบข้อมูลระหว่างตัวแปรหลักกับตัวแปรอื่นๆ ได้ทีละหลายตัวแปรพร้อมกัน ซึ่งมีรูปแบบการออกคำสั่งดังนี้

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

`data`	กรอบข้อมูล
`key`	ตัวแปรหลักที่ใช้สร้างตารางความสัมพันธ์ไขว้กับตัวแปรอื่นๆ สามารถระบุเป็นตัวเลขลำดับสดมภ์ในกรอบข้อมูลหรือระบุเป็นชื่อตัวแปรก็ได้
`columns`	สดมภ์ต่างที่จะนำมาสร้างตารางความถี่หรือที่จะนำมาสร้างตารางความสัมพันธ์ไขว้กับตัวแปรหลัก
key	อาจระบุเป็นเวกเตอร์ของเลขลำดับสดมภ์หรือเวกเตอร์ของชื่อสดมภ์ก็ได้

`row`	แสดงร้อยละด้านแถว (แนวนอน) ค่าปกติเป็น FALSE
`colm`	แสดงร้อยละด้านสดมภ์ (แนวตั้ง) ค่าปกติเป็น FALSE
`test`	ทดสอบนัยสำคัญด้วยวิธี chi-squared และ Fisher's exact test ค่าปกติเป็น FALSE
`position`	ตำแหน่งที่จะวางสดมภ์ key ค่าปกติคือ "row" ถ้าต้องการให้อยู่ด้านสดมภ์ ให้ระบุเป็น "column" หรือเขียนอย่างย่อเป็น "col" หรือ "c" ก็ได้
`frame`	วาดกรอบให้ตาราง ค่าปกติเป็น TRUE
`na.included`	แสดง NA เป็นอีกระดับหนึ่งหรือไม่ ค่าปกติคือ FALSE
`factor.labels`	แสดงแฟกเตอร์โดยคำอธิบายระดับหรือค่า ค่าปกติคือแสดงโดยคำอธิบาย (TRUE)

ในการวิเคราะห์ข้อมูลในบางกรณี เช่น การวิเคราะห์ข้อมูลของการวิจัยแบบ case-control เรามักต้องการทำตารางความสัมพันธ์ไขว้ระหว่างตัวแปรผลตัวเดียว กับตัวแปร exposure หลาย ๆ ตัว หากใช้คำสั่ง **`table()`** หรือ **`tab()`** จับคู่ตัวแปรทีละคู่จะเสียเวลาในการพิมพ์คำสั่งมาก library **ice** จึงได้มีคำสั่งให้สำหรับงานเช่นนี้โดยเฉพาะ ดังตัวอย่างต่อจากที่กล่าวในคำสั่ง **`ftab()`** ข้างบน

```
> xtab(vct2, a11, c(8:10), test=T)
row: a11, column: a7
 ------------------------------------------------------
       |     1 |     2 |     3 |     4 |     9 | Total
 ------+-------+-------+-------+-------+-------+-------
 1     |    33 |    32 |     0 |     0 |     2 |    67
 2     |    68 |   174 |     3 |     2 |     1 |   248
 ------+-------+-------+-------+-------+-------+-------
 Total |   101 |   206 |     3 |     2 |     3 |   315
 ------------------------------------------------------
Chi-squared = 16.93, df = 4, p-value = 0.002
Fisher's exact test (2-sided) p-value = 0.0015

row: a11, column: a8
 ------------------------------------
       |     1 |     2 |     9 | Total
 ------+-------+-------+-------+-------
 1     |    10 |    22 |     2 |    34
 2     |    55 |   128 |     1 |   184
 ------+-------+-------+-------+-------
 Total |    65 |   150 |     3 |   218
 ------------------------------------
Chi-squared = 6.045, df = 2, p-value = 0.0487
Fisher's exact test (2-sided) p-value = 0.1093
```

```
row: a11, column: a9
 ------------------------------------
       |     1 |     2 |     9 | Total
 ------+-------+-------+-------+-------
 1     |    16 |    16 |     2 |    34
 2     |     6 |   176 |     2 |   184
 ------+-------+-------+-------+-------
 Total |    22 |   192 |     4 |   218
 ------------------------------------

Chi-squared = 65.84, df = 2, p-value = 0
Fisher's exact test (2-sided) p-value = 0
```

เพื่อให้ดูตารางได้ง่ายขึ้น จึงควรมีการให้คำอธิบายค่าตัวเลขของตัวแปรแต่ละตัว ซึ่งจะแสดงวิธีการให้คำอธิบายในบทที่ 4 ต่อไป

จากคำสั่งต่าง ๆ ข้างต้น สามารถตรวจสอบความถูกต้องของข้อมูลได้ เช่น ตัวเลขที่พิมพ์ผิดหรือไม่มีในรหัสที่ได้กำหนดไว้ สามารถตรวจสอบได้จากคำสั่ง **`ftab()`** หรือการตอบที่ไม่ตรงกัน สามารถตรวจสอบด้วยการสร้างตารางไขว้ **`tab()`** หรือ **`xtab()`**

3.2 การสำรวจข้อมูลจากรายละเอียดตัวแปรและตาราง

โดยปกติ การเขียนกราฟโดย **R** จะมีข้อพิเศษแตกต่างจากโปรแกรมอื่น คือ การเขียนกราฟใน **R** จะมี 2 ระดับ คือ

1. High level plotting ที่คำสั่งในการสร้างกราฟสมบูรณ์โดยตัวมันเอง คำสั่งหลัก เช่น **`plot()`**, **`boxplot()`**, **`hist()`** เป็นต้น
2. Low level plot ที่คำสั่งไม่ได้สร้างกราฟโดยตัวมันเอง แต่จะเป็นตัวช่วยเติมค่าต่าง ๆ หรือคำอธิบายให้กราฟที่สร้างด้วย High level plotting มีความสมบูรณ์ขึ้น เช่น **`points()`**, **`lines()`**, **`segments()`**, **`text()`**, **`title()`**, **`mtext()`**, **`legend()`** เป็นต้น

3.2.1 คำสั่ง `hist()`

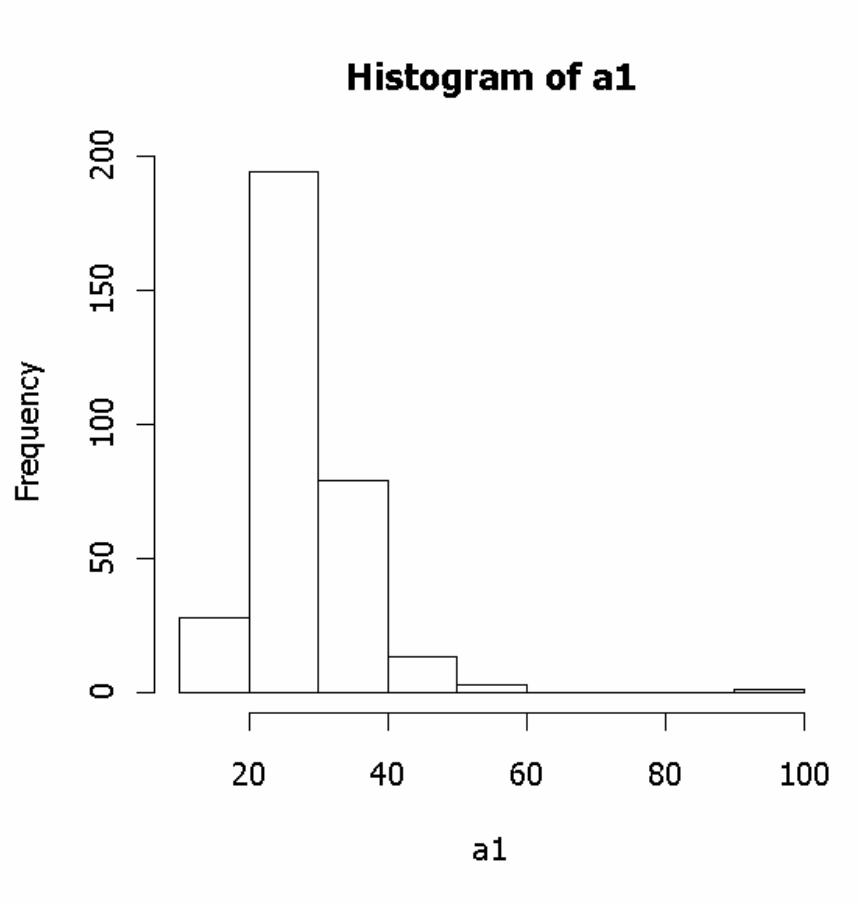
เป็นคำสั่งสำหรับการสร้างฮีสโตแกรมด้วยการ plot ความถี่ของข้อมูลภายในแต่ละอัตรภาคชั้นที่ถูกแบ่งด้วยค่า breaks ความสูงของแท่งจะเป็นสัดส่วนกับจำนวนข้อมูลที่ตกอยู่ในอันตรภาคชั้นนั้น ๆ เป็นคำสั่งที่อยู่ใน library Base ซึ่งมีรูปแบบการออกคำสั่งดังนี้

```
hist(x, breaks = "Sturges", freq = NULL, probability = !freq,
    include.lowest = TRUE, right = TRUE, density = NULL, angle = 45,
    col = NULL, border = NULL, main = paste("Histogram of" , xname),
    xlim = range(breaks), ylim = NULL, xlab = xname, ylab, axes =
    TRUE, plot = TRUE, labels = FALSE, nclass = NULL, ...)
```

x	เวกเตอร์หรือตัวแปรที่ต้องการนำมาสร้างฮีสโตแกรม
breaks	ตัวกำหนดจุดตัดของแท่งฮีสโตแกรม
freq	ความสูงของแท่งจะใช้เป็นความถี่ หากมีค่าเป็น TRUE จะใช้เป็นความน่าจะเป็น ถ้ามีค่าเป็น FALSE
include.lowest	เลือกนับข้อมูลตัวที่มีค่าเท่ากับขีดจำกัดล่าง ถ้ามีค่าเป็น TRUE
right	เลือกนับข้อมูลตัวที่มีค่าเท่ากับขีดจำกัดบน ถ้ามีค่าเป็น TRUE
density	กำหนดให้มีการแรเงาแท่งกราฟด้วยเส้นทะแยง
angle	กำหนดองศาของมุมของเส้นทะแยงที่แรเงาแท่งกราฟ
col	ระบุสีของแท่ง default คือไม่มีสี
plot	ตัวเลือกที่ให้แสดงกราฟบนหน้าจอหรือไม่ default=TRUE
labels	ตัวเลือกที่ให้แสดงคำอธิบายบนแท่งกราฟหรือไม่
...	ตัวเลือก เกี่ยวกับชื่อ และแกนของการสร้างกราฟ

ตัวอย่างการใช้คำสั่งดังนี้

```
> hist(a1)
```



รูปที่ 3.1 ฮีสโตแกรมข้อมูลอายุ

จากฮีสโตแกรมจะเห็นได้ว่าข้อมูลอายุของหญิงบริการทางเพศมีข้อมูลบางข้อมูลที่มีอายุประมาณ 99 ปี ซึ่งข้อมูลดังกล่าวก็คือข้อมูล missing นั่นเอง

3.2.2 คำสั่ง `boxplot()`

เป็นคำสั่งสำหรับการสร้าง boxplot โดยการแบ่งข้อมูลออกเป็น 4 ส่วนเท่า ๆ กัน ครึ่งหนึ่งของข้อมูลจะอยู่ใน box หรือที่เรียกว่า inter-quartile range เส้นกึ่งกลาง box เป็นเส้นที่บ่งบอกถึงกึ่งกลางของข้อมูล คือ ค่ามัธยฐาน (median) เป็นคำสั่งที่อยู่ใน library base ซึ่งมีรูปแบบการออกคำสั่งดังนี้

```
boxplot(formula, data = NULL, ..., subset, na.action = NULL)
```

หรือ

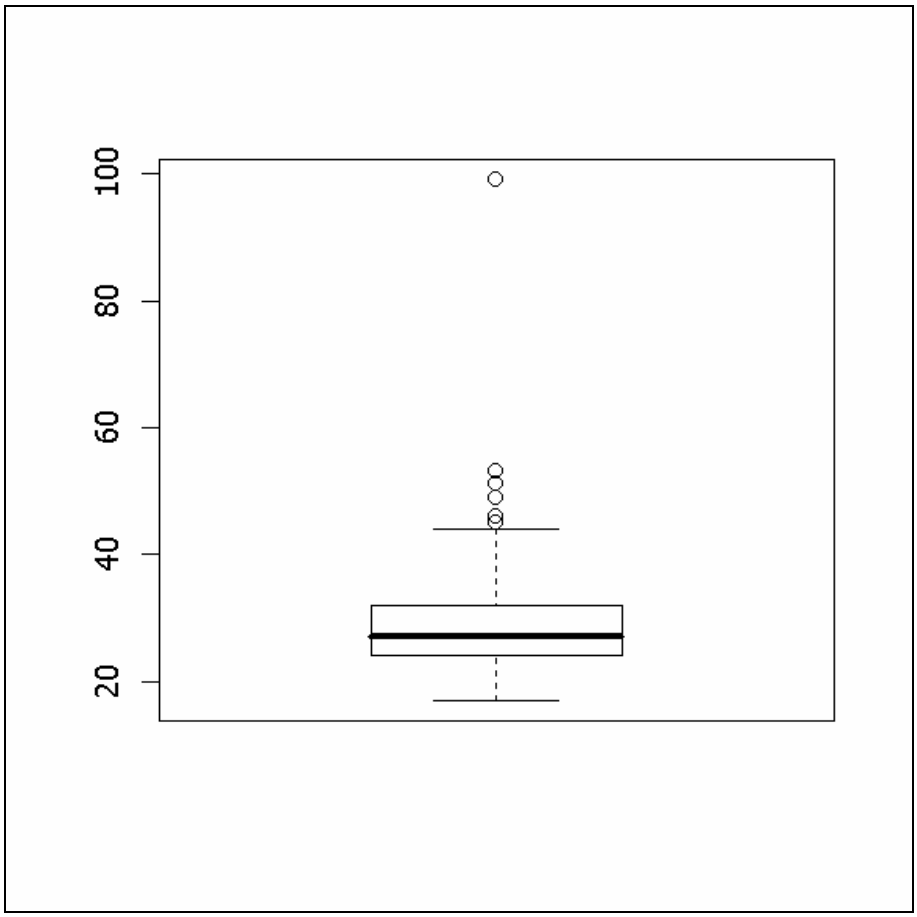
```
boxplot(x, ..., range = 1.5, width = NULL, varwidth = FALSE,
      notch = FALSE, outline = TRUE, names, plot = TRUE,
      border = par("fg"), col = NULL, log = "",
      pars = list(boxwex = 0.8, staplewex = 0.5, outwex = 0.5),
```

```
horizontal = FALSE, add = FALSE, at = NULL)
```

x	เวกเตอร์หรือตัวแปรที่ต้องการนำมาสร้าง boxplot
...	ตัวเลือก เกี่ยวกับชื่อ และแกนของการสร้างกราฟ
formula	สูตรที่ใช้ เช่น ‘y~x’ เมื่อ y เป็นข้อมูลเชิงปริมาณที่นำมาสร้าง boxplot จำแนกตามกลุ่มของ x
data	กรอบข้อมูล
subset	การเลือกข้อมูลเพียงบางส่วนมาสร้าง boxplot
range	กำหนดค่าที่ให้เส้น whiskers ของ boxplot ว่ามีความยาวเท่าใด หากมีค่าเป็นบวก เส้นก็จะมีความยาวที่สุดเท่าที่ค่าสุดโต่งของข้อมูลมีค่าไม่เกินค่าในช่วง interquartile จาก box
width	เป็นเวกเตอร์ที่กำหนดค่าความก้าวของ box
notch	ถ้าค่าเป็น TRUE แต่ละ box ก็จะมีการคอดเข้าที่ตรงกลางของ box
outline	การให้แสดง outlier หรือไม่แสดง
names	การให้แสดงคำอธิบายภายใต้แต่ละ boxplot
boxwex	การให้แสดงสเกลให้แต่ละ box
staplewex	การขยายเส้นปลายขีดจำกัดล่างหรือขีดจำกัดบนตามสัดส่วนความกว้างของ box
outwex	การขยายเส้น outlier ตามสัดส่วนความกว้างของ box
plot	ตัวเลือกที่ให้แสดงกราฟบนหน้าจอหรือไม่ default=TRUE
border	ตัวเลือกที่กำหนดสีให้กับกรอบของ box
col	ตัวเลือกที่กำหนดสีให้ภายใน box
log	กำหนดแกน x หรือ y ที่ต้องการให้แสดงเป็น log scale
par	ตัวเลือกค่าพารามิเตอร์ของกราฟ
horizontal	การกำหนดให้ box plot อยู่ในแกนนอนหรือแกนตั้ง default=FALSE
add	ถ้า เป็น TRUE ก็จะสามารถเพิ่ม box plot ใหม่ ใน box plot ที่มีอยู่เดิมได้
at	เวกเตอร์ตัวเลขที่ให้ตำแหน่ง boxplot ควรจะอยู่ที่ใด

ตัวอย่างการใช้คำสั่งดังนี้

```
> boxplot(a1)
```



รูปที่ 3.2 Boxplot ข้อมูลอายุ

จาก boxplot ทำให้ทราบได้ว่ามี outlier 1 ค่า ซึ่งจะต้องกลับไปตรวจดูว่าเป็นข้อมูลที่อยู่ในชุดใด ดังคำสั่งต่อไปนี้

```
> vct[a1>90,]
    qid a1 a2 a3 a4 a5   a6 a7 a8 a9 a10 a11 a12 a13
183 187 99  2  9 48 48 7000  2  2  2   2   2   1 100
```

อายุใน `qid` ที่ 187 มีค่าเป็น 99 ปี ซึ่งก็คือรหัสของข้อมูลอายุที่เป็น missing

3.2.3 คำสั่ง `plot()`

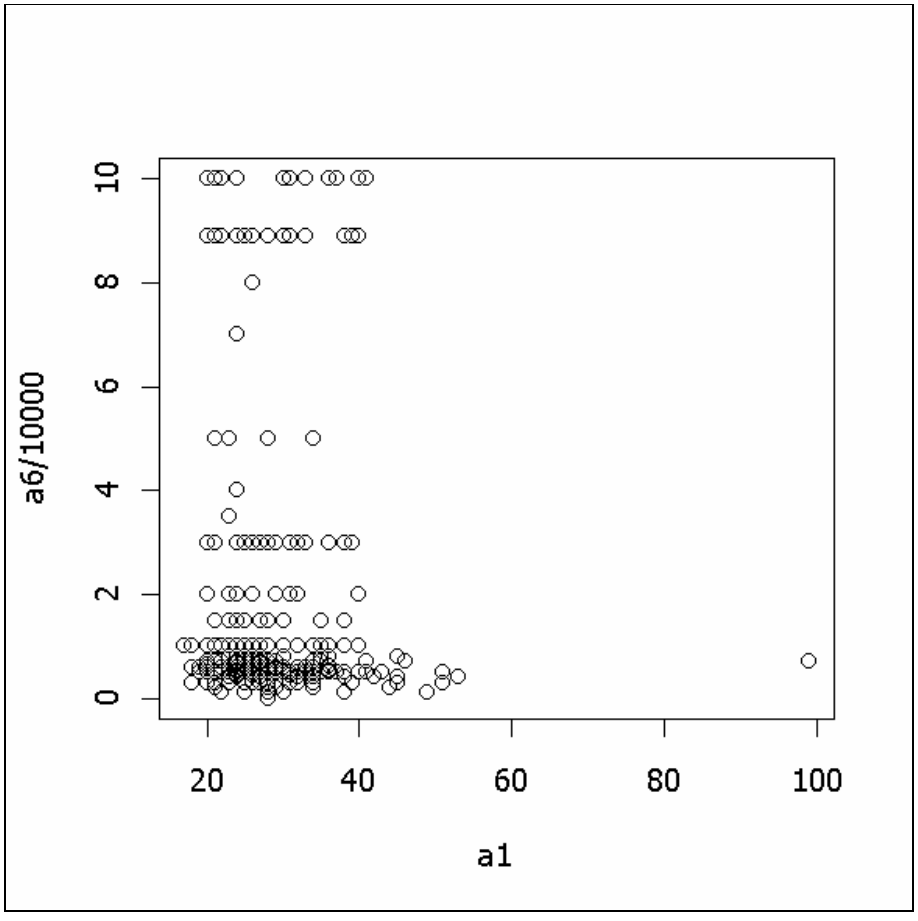
เป็นคำสั่งสำหรับการสร้าง scatter plot เป็นคำสั่งที่อยู่ใน library base ซึ่งมีรูปแบบการออกคำสั่งดังนี้

```
plot(x, y, ...)
```

`x`	ค่าตัวเลขหรือตัวแปรในแกน x
`y`	ค่าตัวเลขหรือตัวแปรในแกน y
`...`	ตัวเลือก เกี่ยวกับชื่อ และแกนของการสร้างกราฟ

ตัวอย่างการใช้คำสั่งดังนี้

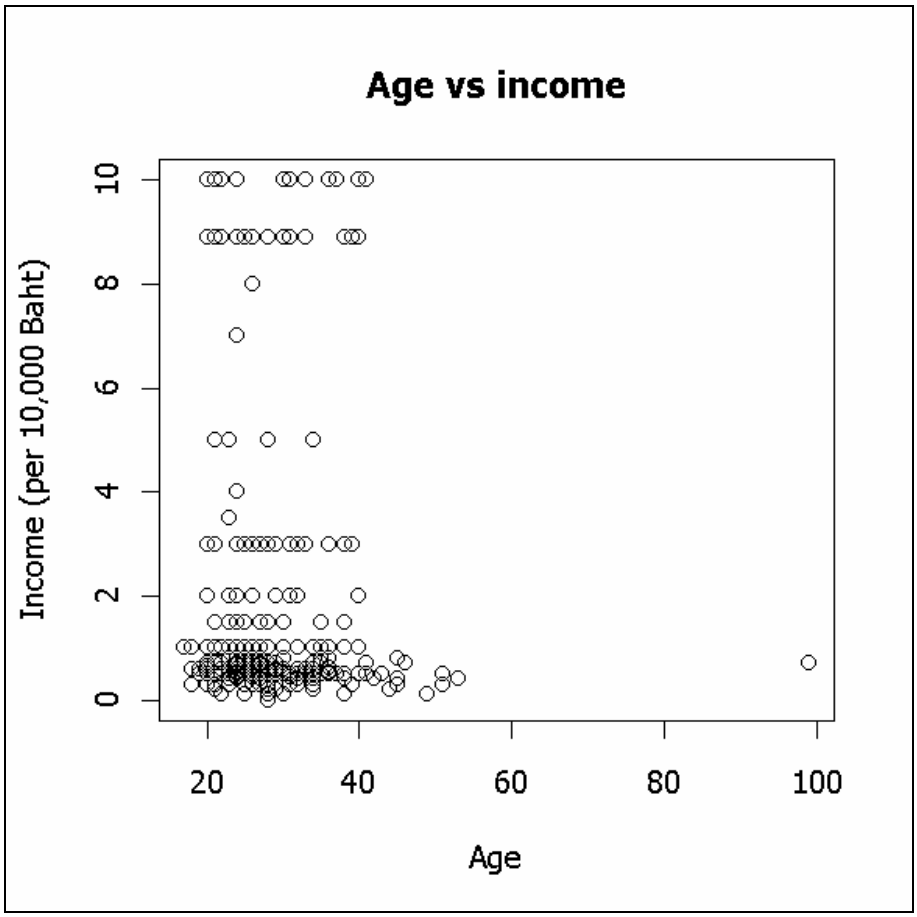
```
> plot(a1, a6)
```



รูปที่ 3.3 Scatter plot ระหว่างอายุและรายได้

ในกรณีที่ต้องการใส่คำอธิบาย สามารถเพิ่มคำอธิบายลงในกราฟได้ ดังตัวอย่างการใช้คำสั่งดังนี้

```
> plot(a1, a6/10000, main="Age vs income", xlab="Age", ylab="Income (per
10,000 Baht)")
```



รูปที่ 3.4 Scatter plot ระหว่างอายุและรายได้โดยการให้คำอธิบายแกน x และ y

3.2.4 คำสั่ง `pairs()`

เป็นคำสั่งสำหรับการสร้าง scatter plot เป็นคำสั่งที่อยู่ใน library base ซึ่งมีรูปแบบการออกคำสั่งดังนี้

```
pairs(formula, data = NULL, ..., subset, na.action = na.pass)
```

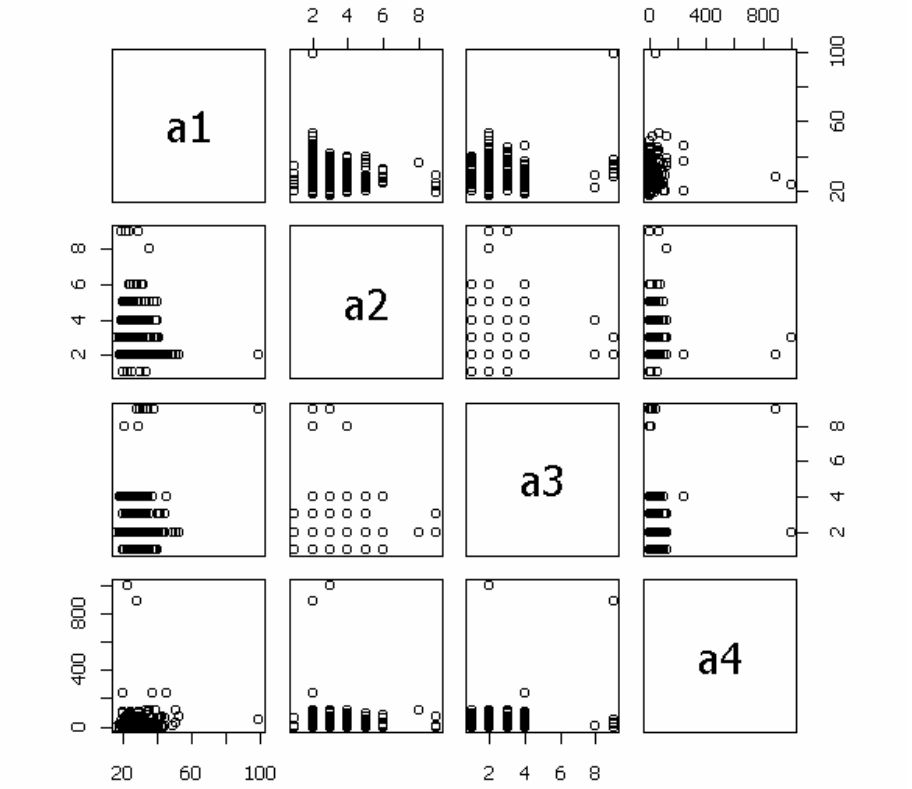
หรือ

```
pairs(x, labels, panel = points, ...,
      lower.panel = panel, upper.panel = panel,
      diag.panel = NULL, text.panel = textPanel,
      label.pos = 0.5 + has.diag/3,
      cex.labels = NULL, font.labels = 1,
      row1attop = TRUE, gap = 1)
```

`x`	เวกเตอร์หรือตัวแปรที่ต้องการนำมาสร้าง scatter plot
`formula`	เป็นสมการที่ต้องการ plot เช่น ~x+y+z แต่ละตัวแปรจะ plot กราฟแยกเป็นแต่ละคู่ ตัวแปรจึงควรต้องเป็นลักษณะเวกเตอร์ตัวเลข
`data`	กรอบข้อมูล
`subset`	การเลือกข้อมูล
`na.action`	ค่า default เป็น na.pass คือให้แสดงค่า missing แต่ถ้าไม่ต้องการให้แสดง ให้เลือกเป็น na.omit
`label`	การให้คำอธิบายชื่อตัวแปร
`panel`	เป็นฟังก์ชันค่า x และ y ที่ต้องการ plot ค่า
`upper.panel, upper.panel`	เป็นฟังก์ชั่น x และ y ที่ใช้สำหรับเป็นส่วนที่อยู่ด้านบนเส้นทะแยงมุมหรือส่วนที่อยู่ด้านล่างเส้นทะแยงมุม
`diagonal.panel`	ฟังก์ชั่น x และ y ที่ใช้สำหรับแนวทะแยงมุม
`text.panel`	ตัวเลือกฟังก์ชั่น x และ y ที่ใช้สำหรับแนวทะแยงมุม
`label.pos`	ตำแหน่งy ที่จะให้คำอธิบาย
`cex.labels,font.labels`	ตัวเลือกทางด้านกราฟฟิก
`row1attop`	ตัวเลือกในการสร้าง scatter plot matrix ที่จะให้ชื่อตัวแปรแรกสุดอยู่ที่มุมซ้ายด้านบน หรือ มุมซ้ายด้านล่าง

ตัวอย่างคำสั่ง

```
> pairs(~a1+a2+a3+a4, data=vct2)
```



รูปที่ 3.5 Scatter plot matrix ระหว่างตัวแปรต่าง ๆ

คำสั่ง **`pairs()`** เป็นคำสั่งหนึ่งที่มีประโยชน์อย่างมากต่อการสำรวจการกระจายของข้อมูล โดยสามารถทำให้เห็นข้อมูลที่มีค่าผิดปกติทันที

3.3 แบบฝึกหัด

1. จากข้อมูล birthwt.tab จงสำรวจข้อมูล มีการป้อนซ้ำหรือไม่ มีค่าที่ป้อนผิดในตัวแปรใดบ้าง และ มีรหัส missing คือค่าใดในตัวแปรต่าง ๆ
2. จงสำรวจข้อมูลด้วยคำสั่งกราฟต่าง ๆ

บทที่ 4

การจัดการข้อมูล (Data manipulation)

หลังจากเสร็จสิ้นกระบวนการนำข้อมูลเข้าสู่คอมพิวเตอร์แล้ว ยังมีกระบวนการต่างๆ อีกมากที่จะต้องทำก่อนการวิเคราะห์ข้อมูลทางสถิติ หนึ่งในกระบวนการดังกล่าวที่จะกล่าวถึง คือ การจัดการกับข้อมูล ทั้งนี้เนื่องจากในกระบวนการวิเคราะห์ข้อมูล บางครั้งไม่สามารถนำตัวแปรที่มีอยู่ในข้อมูลมาวิเคราะห์ได้โดยตรง จำเป็นต้องผ่านกระบวนการจัดการข้อมูลที่มีอยู่ให้ตรงตามวัตถุประสงค์ มีความถูกต้อง จากนั้นจึงจะสามารถนำไปวิเคราะห์ได้ กระบวนการต่าง ๆ ที่ใช้ในการจัดการข้อมูล มีหลากหลายรูปแบบ การจัดการข้อมูลเป็นขั้นตอนที่รวมถึงการจัดการแฟ้มข้อมูล และข้อมูลที่อยู่ในแฟ้ม รายละเอียดการใช้คำสั่งในบทนี้ โดยส่วนใหญ่จะเป็นคำสั่งใน library **ice** ที่ทางกลุ่มผู้วิจัยได้พัฒนาขึ้นมา เพื่อให้ง่ายต่อการใช้งาน เหมาะสำหรับผู้ใช้ที่ไม่ต้องการจำคำสั่งที่ยุ่งยากซับซ้อน ที่อยู่ใน library base ฉะนั้นเมื่อเปิดโปรแกรม **R** ขึ้นมาใช้งานขั้นตอนแรกสุด จะต้องเรียกใช้ library **ice** ก่อน เพื่อที่จะสามารถใช้ฟังก์ชัน หรือคำสั่งต่าง ๆ ที่อยู่ใน library นี้ได้ ด้วยคำสั่งดังนี้

```
> library(ice)
```

คำสั่งต่าง ๆ ที่จะกล่าวถึงสำหรับการจัดการข้อมูล มีดังนี้

1. การจัดการเกี่ยวกับแฟ้ม เช่น

clean()	การลบข้อมูลออกจากหน่วยความจำ
create()	การสร้างกรอบข้อมูลใหม่
fread()	การเปิดอ่านแฟ้ม
fix()	การเปิดกรอบข้อมูลขึ้นมาแสดงในหน้าต่าง Data Editor
subset()	การเลือกเพียงบางส่วน
fwrite()	การบันทึกเป็นข้อมูลใหม่
rbind()	การเชื่อมต่อแฟ้มข้อมูลแฟ้มที่ 2 ต่อต่อท้ายแฟ้มข้อมูลแรก
merge()	การเชื่อมต่อแฟ้มข้อมูลแฟ้มที่ 2 ต่อต่อท้ายแฟ้มข้อมูลแรก
cbind()	การเชื่อมต่อแฟ้มข้อมูล 2 มาต่อด้านข้างแฟ้มข้อมูลแรก
savehistory()	การบันทึกคำสั่งต่าง ๆ ที่ได้ใช้แล้วเก็บไว้เป็นแฟ้ม
do()	การทำงานกับแฟ้มคำสั่ง
logg()	การเก็บผลลัพธ์เป็นแฟ้ม

2. การจัดการเกี่ยวกับข้อมูล

`as.na()`	การจัดการกับข้อมูล missing
`recode.na()`	การเปลี่ยนค่า missing ให้เป็นค่าตัวเลข
`recode()`	การเปลี่ยนค่าและแทนที่ค่าตัวเลข
`change()`	การเปลี่ยนค่าตัวเลข
`transform()`	การสร้างตัวแปรใหม่
`label()`	การให้คำอธิบายค่าตัวเลขของตัวแปร
`rename()`	การให้คำอธิบายชื่อตัวแปรและการเปลี่ยนชื่อตัวแปร
`to.character()`	การเปลี่ยนประเภทตัวแปรอื่น ๆ ให้เป็นประเภทตัวอักษร
`to.factor()`	การเปลี่ยนประเภทตัวแปรอื่น ๆ ให้เป็นประเภท factor
`to.numeric()`	การเปลี่ยนประเภทตัวแปรอื่น ๆ ให้เป็นประเภทตัวเลข
`to.vector`	การเปลี่ยนประเภทตัวแปรอื่น ๆ ให้เป็นประเภทเวกเตอร์
`defactor()`	สร้างเวกเตอร์ตัวเลขใหม่จากแฟกเตอร์เดิม โดยคงค่ารหัสเดิมไว้
`cut()`	การตัดแบ่งข้อมูลต่อเนื่องให้เป็นช่วงค่า

4.1 การจัดการแฟ้ม

4.1.1 คำสั่ง `clean()`

เป็นการลบข้อมูล และวัตถุต่าง ๆ ที่ ได้สร้างขึ้นมาให้ออกจากหน่วยความจำ หรือการปิดแฟ้มที่กำลังเปิดอยู่

```
clean()
```

แนะนำให้ทำทุกครั้งเมื่อต้องการทำงานกับข้อมูลชุดใหม่ เพื่อป้องกันการสับสน การทำงานใน **R** นั้น จะมีวัตถุที่ถูกสร้างขึ้นที่อยู่นอกกรอบข้อมูลจำนวนมาก หากผู้ใช้ไม่มีการลบออกจากหน่วยความจำ ก็จะทำให้เกิดการเรียกใช้ผิดได้ คำสั่งที่ใช้ในการลบวัตถุต่างๆ ทั้งหมด ยกเว้น package ออกจากหน่วยความจำนี้ เป็นคำสั่งที่อยู่ใน library **ice**

4.1.2 คำสั่ง `create()`

เป็นการสร้างกรอบข้อมูลใหม่ มีรูปแบบการออกคำสั่งดังนี้

```
create(...)
```

`...` ชุดของเวกเตอร์ตัวแปรที่จะใส่ในกรอบข้อมูล ถ้าไม่ระบุ เวกเตอร์ใดๆ จะเป็นการสร้างกรอบข้อมูลเปล่า

คำสั่งนี้ใช้เพื่อสร้างกรอบข้อมูลจากเวกเตอร์ที่ระบุในข้อมูลนำเข้า ถ้าเวกเตอร์ที่ใส่ไม่มีชื่อ เช่น `c(1,2,3)` ข้อความเวกเตอร์นั้นจะถูกนำไปใช้เป็นชื่อตัวแปร เพื่อป้องกันปัญหานี้ ควรระบุชื่อเวกเตอร์เสมอ ถ้าใส่เวกเตอร์หลายเวกเตอร์เป็นข้อมูลนำเข้า ทุกเวกเตอร์ต้องมีความยาวเท่ากัน หลังจากเรียกคำสั่งนี้ กรอบข้อมูลนั้นจะถูกแสดงเพื่อแก้ไขเพิ่มเติมได้ เป็นคำสั่งที่อยู่ใน library **ice**

ดังตัวอย่างต่อไปนี้

```
> x <- create()
```

คำสั่งนี้จะสร้างกรอบข้อมูลเปล่าขึ้นมาดังรูปที่ 4.1

รูปที่ 4.1 กรอบข้อมูลเปล่าที่เปิดขึ้นมาจากคำสั่ง *create()*

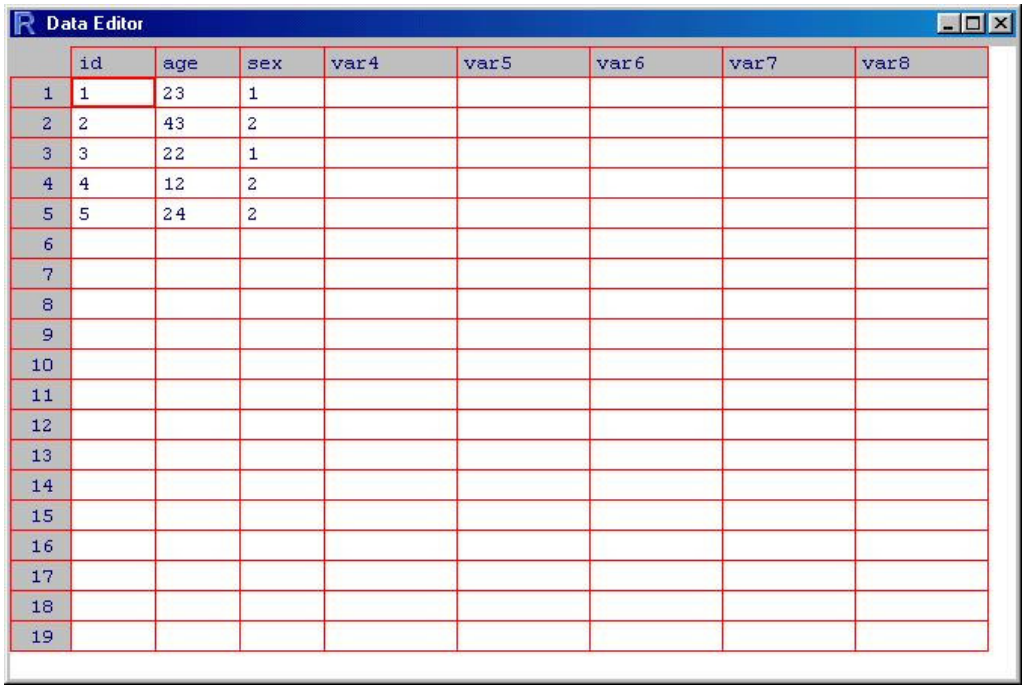
แถบบนสุดของ Data Editor จะถูกกำหนดให้เป็นขอบเขตสำหรับการกำหนดชื่อตัวแปร เมื่อใช้เม้าส์คลิกบนแถบ var1 ก็จะปรากฏ Variable editor ออกมาเพื่อที่จะให้พิมพ์ชื่อตัวแปร ดังรูปที่ 4.2

รูปที่ 4.2 Variable editor ในโปรแกรม R

เช่น ต้องการสร้างชื่อตัวแปร `id` ให้พิมพ์ `id` แทน var1 ลงในช่อง variable name และกำหนดเป็น type ที่เป็น numeric จากนั้นก็จะสามารถป้อนค่าตัวเลขของแต่ละ `id` ได้ ในส่วนของแถว จะมีหมายเลขแถวขึ้นมาให้เห็น

คำสั่งในการสร้าง data frame และป้อนค่าตัวเลขสำหรับแต่ละตัวแปรลงในคำสั่งเลย โดยที่ไม่ต้องป้อนลงใน Data Editor ดังตัวอย่างคำสั่งต่อไปนี้ ซึ่งเมื่อเคาะ Enter หน้าจอ Data Editor ก็จะถูกเปิดขึ้นโดยอัตโนมัติดังรูปที่ 4.3รูปที่ 4.3

```
>x <- create(id=c(1,2,3,4,5),age=c(23,43,22,12,24),sex=c(1,2,1,2,2))
```



รูปที่ 4.3 แสดงค่าตัวแปรจากการพิมพ์ด้วยคำสั่ง *create()*

4.1.3 คำสั่ง fread()

คำสั่งที่สร้างขึ้นเพื่อรวมสำหรับอ่านกรอบข้อมูลจากแฟ้มข้อมูลชนิดต่างๆ เข้าสู่หน่วยความจำ แฟ้มข้อมูลอาจเป็นได้ทั้งชนิด .dta ของ Stata, .rec จาก EpiInfo หรือ EpiData, ชนิด .sav จากโปรแกรม SPSS, ชนิด .mtp จากโปรแกรม Minitab, หรือจะเป็นแฟ้มข้อมูล text ที่แยกข้อมูลด้วย tab (นามสกุล .txt หรือ .tab) หรือ comma (นามสกุล .csv) ก็ได้ ก็คือคำสั่ง **`fread()`** โดยคำสั่งนี้อยู่ใน library **ice** **`fread()`** จะเลือกวิธีอ่านที่เหมาะสมสำหรับแฟ้มข้อมูลชนิดต่างๆ จากนามสกุลของชื่อแฟ้มนั้นๆ ให้โดยอัตโนมัติ ดังนั้นจึงต้องระบุนามสกุลของแฟ้มข้อมูลให้ถูกต้อง มีรูปแบบการออกคำสั่งดังนี้

```
fread("file", ...)
```

`file` ชื่อแฟ้มข้อมูลที่ต้องการอ่านต้องมีเครื่องหมาย “ ” คร่อม

`...` ตัวเลือกสำหรับแฟ้มข้อมูลแต่ละชนิด

ดังตัวอย่างคำสั่งต่อไปนี้

```
> clean()
> setwd("c:/workplace")
> vct <- fread("vct.dta")
```

คำอธิบาย

เป็นการอ่านแฟ้มที่ชื่อว่า `vct.dta` ซึ่งเป็นแฟ้มในรูปแบบของ STATA ที่อยู่ใน folder `c:\RWorkplace` เข้ามาเก็บไว้ในวัตถุที่ชื่อว่า `vct` ข้อควรรู้คือ `vct` จะถูกจัดให้อยู่ในคลาสกรอบข้อมูล `data.frame` โดยอัตโนมัติ ในส่วนที่อยู่ในวงเล็บซึ่งเป็นชื่อแฟ้ม จะต้องมีเครื่องหมาย " ทั้งเปิด และปิด คร่อมไว้เสมอ

4.1.4 คำสั่ง `fix()`

เป็นการเปิดหน้าต่าง Data Editor ขึ้นมาเพื่อดูข้อมูลที่อยู่ภายในวัตถุหรือกรอบข้อมูลที่ต้องการเปิด คำสั่งนี้อยู่ใน library utils มีรูปแบบการออกคำสั่งดังนี้

`fix(x, ...)`

`file` ชื่อวัตถุหรือกรอบข้อมูลที่ต้องการเลือก

`...` ตัวเลือกอื่น ๆ หรือเงื่อนไขที่กำหนด

ตัวอย่างคำสั่งดังต่อไปนี้

```
> fix(vct)
```

4.1.5 คำสั่ง `subset()`

เป็นคำสั่งสำหรับการเลือกข้อมูลเพียงบางส่วนมาใช้งาน หรือเลือกข้อมูลตามเงื่อนไขที่กำหนด เป็นคำสั่งที่อยู่ใน library base มีรูปแบบการออกคำสั่งดังนี้

`subset(x, ...)`

`x` ชื่อวัตถุหรือกรอบข้อมูลที่ต้องการเลือก

`...` ตัวเลือกอื่น ๆ หรือเงื่อนไขที่กำหนด

ตัวอย่างคำสั่ง ดังต่อไปนี้

```
> vct.first <- subset(vct, select=c(qid:a7))                    ❶
> vct.second <- subset(vct, select=c(qid, a8:a13))              ❷
> vct.99 <- subset(vct, qid < 100)                              ❸
```

คำอธิบาย

❶ เป็นการเลือกข้อมูลตั้งแต่ตัวแปร `qid` จนถึงตัวแปร `a7` ตัวแปรที่อยู่ระหว่าง `qid` และ `a7` คือ `a1, a2, a3, a4, a5` และ `a6` จะถูกเลือกด้วย ทั้งหมดนี้ถูกเก็บในวัตถุที่ตั้งชื่อว่า `vct.first`

❷ เป็นการเลือกข้อมูลตัวแปร `qid` และตัวแปร `a8` จนถึงตัวแปร `a13` ทั้งหมดนี้ถูกเก็บในวัตถุที่ตั้งชื่อว่า `vct.second`

❸ เป็นการเลือกข้อมูลที่มี `qid` น้อยกว่า `100` มาใช้ในการวิเคราะห์ ฉะนั้นข้อมูลที่ `qid` มากกว่าหรือเท่ากับ `100` จะถูกตัดทิ้งไป ข้อมูลที่ถูกเลือกจะนำมาเก็บไว้ในวัตถุที่ชื่อว่า `vct.99`

4.1.6 คำสั่ง `fwrite()`

เป็นคำสั่งในการบันทึกข้อมูลที่กำลังทำงานอยู่ให้เป็นแฟ้มใหม่ เป็นคำสั่งที่อยู่ใน library **ice** การบันทึกข้อมูลใน **R** จะบันทึกข้อมูลเป็น ASCII แฟ้ม คือ เป็น text สามารถให้นามสกุลเป็น txt หรือ tab ที่แยกข้อมูลด้วย tab หรือ นามสกุลเป็น csv ที่แยกข้อมูลด้วย comma มีรูปแบบการออกคำสั่งดังนี้

```
fwrite(x, file, ...)
```

`x`	กรอบข้อมูล
`file`	ชื่อแฟ้มข้อมูลที่ต้องการบันทึก ใส่ไว้ภายใน "..."
`...`	ตัวเลือกสำหรับแฟ้มข้อมูลชนิดต่างๆ ดูเพิ่มเติมใน write.dta และ write.table แล้วแต่ชนิดของแฟ้มข้อมูลที่ต้องการเขียน

ตัวอย่างคำสั่งดังต่อไปนี้

```
> fwrite(vct.first,"vct1.tab")
> fwrite(vct.second,"vct2.tab")
```

คำอธิบาย

เป็นการบันทึกข้อมูลที่ถูกเก็บอยู่ในวัตถุที่ชื่อ `vct.first` ให้เป็นแฟ้มชื่อว่า `vct1.tab` และวัตถุที่ชื่อ `vct.second` ให้เป็นแฟ้มชื่อว่า `vct2.tab`

หมายเหตุ

ผู้ใช้สามารถให้นามสกุลเป็น .txt หรือ .csv ก็ได้ ซึ่งแฟ้มเหล่านี้จะสามารถเปิดดูได้ด้วยโปรแกรม Excel จึงสะดวกในการใช้งาน

4.1.7 คำสั่ง `rbind()`

เป็นการเชื่อมกรอบข้อมูลข้อมูลหรือวัตถุตั้งแต่ 2 ชุดขึ้นไปเข้าด้วยกัน โดยเอาข้อมูลชุดที่ 2 มาต่อท้ายข้อมูลชุดแรก ซึ่งอยู่ใน library base มีรูปแบบการออกคำสั่งดังนี้

```
rbind(..., deparse.level = 1)
```

`...`	ตัวเลือกอื่น ๆ หรือเงื่อนไขที่กำหนด
`deparse.level`	เป็นตัวเลขที่ใช้ในการควบคุมการสร้าง `label` ปกติกำหนดให้มีค่าเป็น 1

คำสั่งนี้มีข้อกำหนดคือ ข้อมูลที่นำมาเชื่อมกันในแนวตั้งจะต้องมีจำนวนสดมภ์ หรือจำนวนตัวแปร เท่ากัน มีชื่อตัวแปรเหมือนกัน ถ้าหากทำการเชื่อมข้อมูล 2 ชุด ที่มีตัวแปรไม่เท่ากัน ผลที่ได้จากการเชื่อม จำนวนสดมภ์จะเท่ากับจำนวน สดมภ์ของแฟ้มที่มีจำนวนสดมภ์น้อยกว่า (จำนวนสดมภ์ของแฟ้มที่มีมากกว่าจะถูกตัดทิ้งไป)

ตัวอย่างข้อมูล `vct` ที่ทำการสัมภาษณ์หญิงบริการในจังหวัดภูเก็ต ซึ่งมีผู้ทำการป้อนข้อมูล 4 คน คือ Jianping ป้อนข้อมูลเก็บไว้ในแฟ้ม ชื่อ jianping.rec, Yongxia ป้อนข้อมูลเก็บไว้ในแฟ้มชื่อ yongxia.rec, Caile ป้อนข้อมูลเก็บไว้ในแฟ้มชื่อ caile.rec และ Udom ป้อนข้อมูลเก็บไว้ในแฟ้มชื่อ udom.rec ต้องการที่จะรวมข้อมูลจากที่ทั้ง 4 คนได้ป้อนไว้เป็นแฟ้มเดียวกัน สามารถทำได้ดังนี้

```
> Jianping <- fread("jianping.rec", lower.case.names=T)      ❶
> Yongxia <- fread("yongxia.rec", lower.case.names=T)        ❷
> Caile <- fread("caile.rec", lower.case.names=T)            ❸
> Udom <- fread("udom.rec", lower.case.names=T)              ❹
```

```
> vct.ap <- rbind(Jianping, Yongxia, Caile, Udom)                    ❺
```

คำอธิบาย

❶, ❷, ❸ และ ❹ เป็นการอ่านแฟ้มข้อมูลที่แต่ละคนป้อนไปเก็บไว้ในวัตถุ ที่ชื่อ `Jianping, Yongxia, Caile` และ `Udom` ตามลำดับ เนื่องจากข้อมูลที่เก็บจากภาคสนาม จะมีบางตัวแปรที่เป็น missing คือไม่มีข้อมูล ดังนั้นในแฟ้มข้อมูลที่บางตัวแปรไม่มีข้อมูล เมื่อทำการอ่านแฟ้ม จะมีข้อความเตือนขึ้น

```
Warning message:
NAs introduced by coercion
```

ในที่นี้ **R** จะบอกว่า missing จะถูกแทนด้วยสัญลักษณ์ `NA (not available)` ส่วนคำสั่ง `lower.case.names=T` คือ อ่านตัวแปรเข้ามาใน **R** ให้เป็นตัวพิมพ์เล็ก

❺ ทำการเชื่อมวัตถุทั้ง 4 เข้าด้วยกันตามแนวตั้ง ด้วยการใช้คำสั่ง **`rbind()`** และเก็บข้อมูลไว้ในวัตถุชื่อ `vct.ap`

4.1.8 คำสั่ง `merge()`

เป็นการเชื่อมกันในแนวนอน ซึ่งเชื่อมข้อมูลตั้งแต่ 2 ชุดขึ้นไปเข้าด้วยกัน โดยการเอาข้อมูลชุดที่ 2 มาต่อด้านข้างของข้อมูลชุดแรก ซึ่งคำสั่งนี้อยู่ใน library base มีรูปแบบการออกคำสั่งดังนี้

```
merge(x, y, by = intersect(names(x), names(y)), by.x = by,
      by.y = by, all = FALSE, all.x = all, all.y = all,
      sort = TRUE, suffixes = c(".x",".y"), ...)
```

`x, y`	กรอบข้อมูลชุดแรก และชุดที่สอง
`by = intersect(names(x), names(y))`	เชื่อมกันโดยใช้ชื่อที่เหมือนกัน
`by.x = by`	เชื่อมกันโดยใช้ชื่อตัวแปรจากแฟ้มชุดแรก
`by.y=by`	เชื่อมกันโดยใช้ชื่อตัวแปรจากแฟ้มชุดที่สอง
`all=FALSE`	ถ้าหาก เป็น `TRUE` คือ ในแต่ละแถวของ `x` ที่ไม่ match กับ `y` ก็จะมีค่าเป็น NA หรือ แต่ละแถวของ

	y ที่ไม่ match กับ x มีค่าเป็น NA
all.x=all	แต่ละแถวของ x ที่ไม่ match กับ y ก็จะมีค่าเป็น NA
all.y=all	แต่ละแถวของ y ที่ไม่ match กับ x มีค่าเป็น NA
sort = TRUE	เรียงข้อมูลตามสดมภ์
suffixes	ตัวอักษรที่ระบุท้ายตัวแปรที่จะใช้ในการ merge เพื่อบอกว่ามาจากแฟ้มใด
...	ตัวเลือกอื่นๆ หรือเงื่อนไขที่กำหนด

4.1.9 คำสั่ง cbind()

คำสั่ง **cbind()** ที่อยู่ใน library base ก็จะสามารถเชื่อมแฟ้มเช่นเดียวกับคำสั่ง **merge()** แต่คำสั่ง **cbind()** มีเงื่อนไขว่าข้อมูลทั้งสองชุดจะต้องมีความยาวเท่ากัน ถ้าหากมีความยาวไม่เท่ากัน **R** จะไม่เชื่อมข้อมูลดังกล่าวให้ มีรูปแบบการออกคำสั่งดังนี้

```
cbind(..., deparse.level = 1)
```

...	ตัวเลือกอื่น ๆ หรือเงื่อนไขที่กำหนด
deparse.level	เป็นตัวเลขที่ใช้ในการควบคุมการสร้าง label ปกติกำหนดให้มีค่าเป็น 1

ตัวอย่างการใช้คำสั่ง ดังต่อไปนี้

```
> vct1 <- fread("vct1.tab")                          ❶
> vct2 <- fread("vct2.tab")                          ❷
> vct.mg <- merge(vct1, vct2, by="qid")              ❸
> vct.mg2 <- cbind(vct1, vct2)                       ❹
```

คำอธิบาย

❶ และ ❷ เปิดอ่านข้อมูล vct1.tab และ vct2.tab แล้วเก็บไว้ในวัตถุชื่อว่า vct1 และ vct2 ตามลำดับ โดยที่ vct1 และ vct2 ถูกจัดอยู่ในคลาสกรอบข้อมูลโดยอัตโนมัติ

❸ ทำการเชื่อมวัตถุทั้งสองเข้าด้วยกันตามแนวนอน ด้วยการใช้คำสั่ง **merge()** และเก็บข้อมูลไว้ในวัตถุชื่อ vct.mg เช่นเดียวกับคำสั่ง ❹ ที่ใช้คำสั่ง **cbind()**

4.1.10 คำสั่ง savehistory()

เป็นคำสั่งที่บันทึกคำสั่งต่าง ๆ ที่ได้สั่งไปแล้วย้อนหลัง แล้วเก็บไว้เป็นแฟ้ม มีรูปแบบการออกคำสั่งดังนี้

```
loadhistory(file = ".Rhistory")
savehistory(file = ".Rhistory")
history(max.show = 25, reverse = FALSE)
```

`file`	สำหรับใส่ชื่อแฟ้มที่ต้องการบันทึก
`max.show`	จำนวนคำสั่งสูงสุดที่ต้องการให้แสดง
`reverse`	เป็นการเรียงลำดับคำสั่งล่าสุดขึ้นก่อน

ตัวอย่างดังคำสั่งต่อไปนี้

```
> savehistory(file="vcttest.r")      ❶
> loadhistory(file="vcttest2.r")     ❷
> history(max.show=50)               ❸
```

คำอธิบาย

คำสั่งสองคำสั่งแรกเป็นการบันทึกคำสั่งเป็นแฟ้ม ส่วนคำสั่งสุดท้ายเป็นการแสดงคำสั่งที่ได้สั่งไปแล้ว

การบันทึกคำสั่งต่าง ๆ ไว้เป็นแฟ้ม ก็เพื่อที่จะนำคำสั่งเหล่านั้นกลับมาทำงานซ้ำอีกครั้งโดยที่ไม่ต้องเสียเวลาพิมพ์ใหม่อีกครั้ง นอกจากคำสั่งต่าง ๆ ข้างต้นแล้ว **R** ยังสามารถบันทึกคำสั่งในอดีตที่ได้พิมพ์ไปแล้วย้อนกลับไปได้ทั้งหมด ด้วยการเลือกเมนู File แล้วเลือก Save History ... จากนั้นก็จะปรากฏ Window ขึ้นมาให้ใส่ชื่อแฟ้มสำหรับเก็บรวบรวมคำสั่งที่ได้ใช้ไปแล้วในอดีตทั้งหมด **R** จะปรากฏนามสกุลของแฟ้ม Rhistory มาให้โดยอัตโนมัติ ในการบันทึกคำสั่งให้เปลี่ยนนามสกุลเป็น .r อย่างเดียว แล้วเลือกบันทึกไว้ใน Working folder ที่กำลังทำงานอยู่ คือ RWorkplace เช่น บันทึกเป็นแฟ้มที่ให้ชื่อว่า vcttest.r จากนั้นสามารถเปิดแฟ้มนี้ด้วยการใช้ Notepad หรือ Metapad หรือ Crimson Editor เพื่อแก้ไขคำสั่ง หรือลบคำสั่งที่พิมพ์ผิดออกไป วิธีนี้เป็นวิธีที่นักวิเคราะห์ข้อมูลเลือกใช้ เพราะสามารถประหยัดเวลาในการทำงาน และสามารถตรวจสอบความถูกต้อง และรายละเอียดคำสั่งต่าง ๆ ที่ได้ใช้ไปแล้ว

นอกจากนี้ยังเก็บเป็นเอกสารไว้ใช้ในภายหลังได้ รายละเอียดในแฟ้มที่ได้บันทึกไว้ประกอบด้วยคำสั่งต่าง ๆ ที่ได้ใช้ไปแล้ว ดังตัวอย่างต่อไปนี้

```
library(ice)
clean()
create()
x <- create(id=c(1,2,3,4,5),age=c(23,43,22,12,24),sex=c(1,2,1,2,2))
clean()
setwd("c:/rworkplace")
vct <- fread("vct.dta")
fix(vct)
vct.first <- subset(vct, select=c(qid:a7))
vct.second <- subset(vct, select=c(qid, a9:a13))
vct.99 <- subset(vct, qid < 100)
fwrite(vct.first,"vct1.tab")
fwrite(vct.second,"vct2.tab")
Jianping <- fread("jianping.rec")
Yongxia <- fread("yongxia.rec")
Caile <- fread("caile.rec")
Udom <- fread("udom.rec")
vct.ap <- rbind(Jianping, Yongxia, Caile, Udom)
vct1 <- fread("vct1.tab")
vct2 <- fread("vct2.tab")
vct.mg <- merge(vct1, vct2, by="qid")
vct.mg2 <- cbind(vct1, vct2)
savehistory(file="vcttest.r")
```

นอกจากการ savehistory แล้ว ใน **R** ยังมีวิธีการ save อีก 2 วิธี คือ

1. การ Save Workspace จากเมนู File ซึ่งจะเป็นการบันทึกวัตถุทุกอย่างที่สร้างไว้ใน **R** เก็บไว้ในแฟ้มที่มีนามสกุล .Rdata หรือพิมพ์คำสั่ง ดังต่อไปนี้

```
save.image("file")
```

เหมาะสำหรับการทำงานบางอย่างค้างเอาไว้ และต้องการกลับมาทำงานต่อโดยไม่ต้องทำงานเดิมซ้ำอีกครั้ง แต่ในที่นี้ไม่แนะนำให้ใช้ ควรที่จะเก็บเป็นแฟ้มคำสั่งไว้จะดีกว่า เพราะไม่ทำให้เปลืองเนื้อที่ในการจัดเก็บแฟ้ม

2. การบันทึกข้อความต่าง ๆ ที่ปรากฏอยู่บน RGUI ไว้เป็นแฟ้ม ซึ่ง **R** จะกำหนดให้มีนามสกุลเป็น txt ในกรณีนี้เช่นเดียวกับ การ save วิธีแรก คือ ไม่แนะนำให้ใช้การ save output ต่าง ๆ ที่ได้คำนวณไว้ ควรจะเก็บไว้เป็นแฟ้มคำสั่งดีกว่า ซึ่งสามารถนำคำสั่งในแฟ้มมา run หลาย ๆ ครั้ง ได้ตามที่ต้องการ

โปรแกรม Crimson Editor

การทำงานกับแฟ้มคำสั่ง เพื่อให้ง่ายต่อการใช้ ควรใช้โปรแกรม Crimson Editor แก้ไขคำสั่งก่อน 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 ท่านต้องทำขั้นตอนต่อไปนี้ 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. คลิกปุ่มชี้ขึ้นบนตรงด้านบนขวาเพื่อเลื่อน “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”

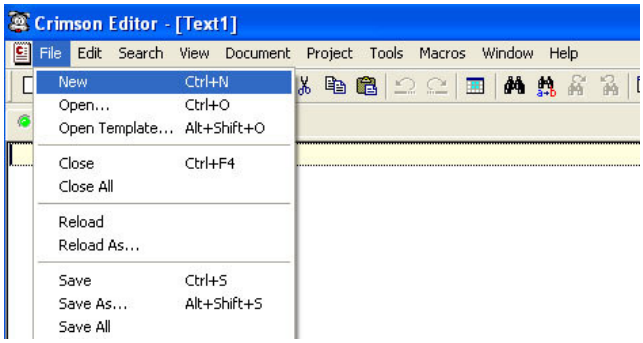
ถึงตรงนี้ **R** ควรปรากฏในรายการเมนู “Document -> Syntax Type” ต่อไปทุกครั้งที่ท่านทำงานกับแฟ้ม **R** ก็ให้แน่ใจว่า **R** ถูกเลือกใน “Document -> Syntax Type” ก็จะมั่นใจได้ว่าคำสงวนของ **R** จะเปลี่ยนสีอย่างถูกต้อง

ขั้นตอนต่อจากนี้ไป แนะนำให้แก้ไข และเพิ่มเติมคำสั่งต่าง ๆ โดยใช้โปรแกรม Crimson Editor ทั้งนี้เพื่อ

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

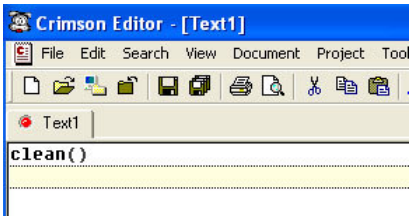
หลังจากติดตั้งโปรแกรม Crimson Editor และแก้ไขให้โปแกรมรู้จักคำสั่งใน **ice** แล้ว ขั้นตอนการเปิดโปรแกรมขึ้นมาใช้ เป็นดังนี้

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



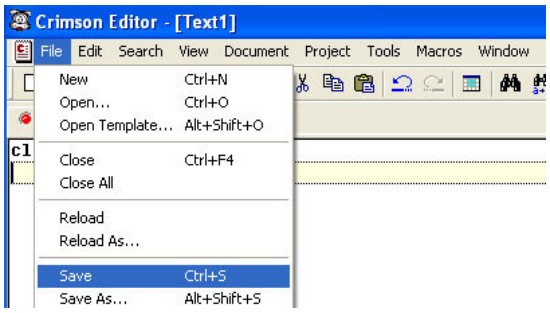
รูปที่ 4.4 จากเมนู File เลือก New ใน Crimson Editor

2. โดยทั่วไป ในการทำงานกับแฟ้มคำสั่งในการจัดการข้อมูลหรือการวิเคราะห์ทางสถิติในแต่ละครั้ง 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 vcttest.r ดังนี้ สังเกตว่าในที่นี้ใช้นามสุกลของแฟ้มเป็น **.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)
```

`file`	ชื่อแฟ้มชุดคำสั่ง เขียนไว้ภายใน "..."
`prompt.echo`	ตัวอักษรบอกลักษณะ prompt ที่จะใช้แสดง
`from`	เลขบรรทัดใน 'file' ที่เริ่มการทำงาน
`to`	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"))
```

`file`	ชื่อแฟ้มข้อมูลที่ต้องการบันทึก
`append`	ตัวเลือกว่าต้องการบันทึกต่อท้าย (ค่าปกติ append=TRUE) หรือเขียนทับหรือสร้างแฟ้มใหม่ (append=FALSE) ค่าปกติเป็นการเขียนทับ

ตัวอย่างการใช้คำสั่งดังนี้

```
> 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 โดยไม่ใช่ชื่อแฟ้ม แนะนำให้ปิดชุดคำสั่งด้วย **`logg()`** ทันทีที่เปิดบันทึกผลการทำงาน มิฉะนั้นเมื่อกลับไปทำงานใน **R** console ผลการออกคำสั่งต่อๆ ไปหลังจากนี้จะถูกบันทึกลงแฟ้มทั้งหมด โดยไม่แสดงบนหน้าจอ โดยทั่วไป เรามักจะยังไม่บันทึกผลการทำงานลงแฟ้มตั้งแต่แรก แต่จะดูผลการทำงานใน **R** console ก่อน ก็สามารถทำได้โดยใส่เครื่องหมาย # ไว้หน้า **`logg()`** ก่อน เมื่อทดสอบการทำงานจนพอใจแล้ว และต้องการบันทึกผล จึงเอาเครื่องหมาย # ออก ดังนี้

4.2 การจัดการข้อมูล

4.2.1 คำสั่ง `as.na()`

คำสั่งนี้จะแทนที่ข้อมูลสูญหายที่กำหนดด้วยค่าตัวเลข ให้เป็นค่า NA หรือ เปลี่ยนค่าในเวก-เตอร์หนึ่งหรือชุดของเวกเตอร์ในกรอบข้อมูลให้เป็น 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) ซึ่งอาจระบุเป็นชื่อตัวแปรในรูป c(alcohol,smoke) หรือเลขสดมภ์ในรูป c(2:5)
`code`	รหัสตัวเลขที่จะเปลี่ยนไปเป็น NA
`append`	ตัวเลือกระบุว่าจะสร้างตัวแปรใหม่ต่อท้ายกรอบข้อมูลหรือเขียนทับ ค่าปกติเป็น TRUE
`var.names`	เวกเตอร์ชื่อตัวแปรใหม่ ตัวเลือกนี้ใช้ได้เฉพาะเมื่อ append=TRUE ชื่อต้องใส่ไว้ในเครื่องหมายคำพูด

เนื่องจากในสภาวะความเป็นจริงข้อมูลต่าง 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 ทิ้งควรทำการสำรวจข้อมูลด้วยคำสั่ง **`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) ซึ่งอาจระบุเป็นชื่อตัวแปรในรูป c(alcohol,smoke) หรือเลขสดมภ์ในรูป c(2:5)
`code`	NA ที่จะเปลี่ยนไปเป็นตัวเลข
`append`	ตัวเลือกระบุว่าจะสร้างตัวแปรใหม่ต่อท้ายกรอบข้อมูลหรือเขียนทับ 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`	ตัวเลือกระบุว่าจะสร้างตัวแปรใหม่ต่อท้ายกรอบข้อมูลหรือเขียนทับ 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`	เวกเตอร์หรือชุดของสดมภ์ในกรอบข้อมูล
`condition`	ชุดของค่าหรือเงื่อนไขทางตรรกศาสตร์
`value`	ค่าใหม่ที่ต้องการ
`append`	ตัวเลือกระบุว่าจะสร้างตัวแปรใหม่ต่อท้ายกรอบข้อมูลหรือเขียนทับ 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()`

เป็นคำสั่งในกาสร้างตัวแปรใหม่ใน **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`	เวกเตอร์ระบุระดับข้อมูล เขียนในรูป รหัส = คำอธิบาย ดังนี้ c("0=no", "1=yes")
`sep`	ตัวอักษรที่ใช้ล้อมชื่อระดับข้อมูล ค่าปกติไม่มี ถ้าใส่เพียงอักษรเดียว จะเป็นตัวอักษรที่วางหน้าชื่อระดับข้อมูล
`ordered`	ตัวเลือกเพื่อระบุว่าค่าในข้อมูลเรียงระดับหรือไม่ **R** จะตรวจสอบว่าเดิม vector นั้นถูกกำหนดให้เรียงระดับหรือไม่ และจะนำไปกำหนดเป็นค่าปกติของ factor ที่ส่งกลับ
`append`	ตัวเลือกระบุตำแหน่งที่จะวางตัวแปรใหม่ อาจวางต่อท้ายกรอบข้อมูล (ค่าปกติ `append=TRUE`) หรือแทนที่ตัวแปรเดิม `(append=FALSE)`
`var.names`	เวกเตอร์ของชื่อตัวแปรใหม่ ตัวเลือกนี้ใช้ได้ถ้า `append=TRUE` 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 สามารถที่จะเคาะ 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 สามารถให้คำอธิบายตัวแปรดังคำสั่งต่อไปนี้

```
> vct <- describe(vct, var=c("qestion 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        qestion 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)
```

data	กรอบข้อมูล
x	ตัวแปรหรือชุดตัวแปรในกรอบข้อมูลที่ต้องการเปลี่ยนชื่อ อาจเรียกตัวแปรโดยชื่อหรือเลขสดมภ์ก็ได้
var.names	เวกเตอร์ข้อความระบุชื่อใหม่

ดังตัวอย่าง

```
> vct <- rename(vct, c(a1, a2), c("age", "education"))
> describe(vct)
No. of observation = 315
```

```
   Variable   Description
1  qid        qestion 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 ขึ้นมาดูก็จะเห็นว่าชื่อตัวแปรถูกเปลี่ยนแล้ว

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`	เวกเตอร์หรือชุดของเวกเตอร์ที่ต้องการเปลี่ยน
`append`	ตัวเลือกระบุตำแหน่งที่จะวางตัวแปรใหม่ ซึ่งอาจเป็นท้ายกรอบข้อมูล (ค่าปกติ append=TRUE) หรือเขียนทับตัวแปรเดิม (append=FALSE)
`var.names`	เวกเตอร์ของชื่อตัวแปรใหม่ จะใช้ได้เมื่อ append=TRUE ชื่อต้องใส่ไว้ในเครื่องหมายคำพูด

ในบางครั้งผู้ใช้อาจพบเจอข้อมูลที่เป็นตัวเลข แต่ถูกเก็บเป็นประเภทตัวอักษร ซึ่งในการคำนวณทางสถิติขั้นสูง โปรแกรมทางสถิติทุกโปแกรม จะไม่สามารถนำตัวแปรที่เป็นประเภทตัวอักษรมาคำนวณได้ จึงจำเป็นที่จะต้องแปลงข้อมูลให้เป็นประเภทตัวเลขเสียก่อน

ดังตัวอย่างต่อไปนี้

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

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`	เวกเตอร์เดี่ยวหรือชุดของเวกเตอร์ตัวเลขที่ต้องการ
	เปลี่ยน `labels` เวกเตอร์ข้อความระบุคำอธิบายรหัส
	ordered ตัวเลือกตรรกะบอกว่าระดับควรถือว่าเรียงลำดับ
	หรือไม่ (ตามลำดับที่ให้ไว้)
`append`	ตัวเลือกระบุตำแหน่งที่จะวางตัวแปรใหม่ ซึ่งอาจเป็น
	ท้ายกรอบข้อมูล (ค่าปกติ append=TRUE) หรือเขียนทับ
	ตัวแปรเดิม (append=FALSE)
`var.names`	เวกเตอร์ของชื่อตัวแปรใหม่ จะใช้ได้เมื่อappend=TRUE
	ชื่อต้องใส่ไว้ในเครื่องหมายคำพูด

ตัวอย่างดังต่อไปนี้

```
> class(a4)
```

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

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 ซึ่งอาจเป็นท้ายกรอบข้อมูล (ค่าปกติ append=TRUE) หรือเขียนทับตัวแปรเดิม (append=FALSE)
`var.names`	เวกเตอร์ของชื่อตัวแปรใหม่ จะใช้ได้เมื่อ append=TRUE ชื่อต้องใส่ไว้ในเครื่องหมายคำพูด

ตัวอย่างดังต่อไปนี้

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

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 ซึ่งอาจเป็นท้ายกรอบข้อมูล (ค่าปกติ append=TRUE) หรือเขียนทับตัวแปรเดิม (append=FALSE)
var.names	เวกเตอร์ของชื่อตัวแปรใหม่ จะใช้ได้เมื่อ append=TRUE ชื่อต้องใส่ไว้ในเครื่องหมายคำพูด

ตัวอย่างดังต่อไปนี้

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

คำอธิบาย

❶ เปลี่ยนตัวแปรจากตัวเลขเป็นตัวอักษร เมื่อใช้คำสั่ง **`class(a3)`** **R** จะแสดงคำว่า `"character"`

❷ เป็นการเปลี่ยนจากตัวเลขเป็น factor เมื่อใช้คำสั่ง **`class(a3)`** ก็จะเห็นคำว่า "factor"

❸ เปลี่ยนชุดของแฟกเตอร์ในกรอบข้อมูลเป็นเวกเตอร์ตัวเลข

เมื่อใช้คำสั่ง **`class(a4)`** ก็จะเห็นคำว่า `"numeric"`

❹ 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 ตัวเลือกนี้ใช้ได้ถ้าappend=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 ก็จะไม่มีข้อมูลของเด็กอายุ 10 เดือน ที่นำมาใช้ใน

การคำนวณ แต่อายุดังกล่าวจะอยู่ในกลุ่มที่มีอายุ 7 - 12 เดือน อย่างไรก็ตามบางครั้งก็มีความจำเป็นที่จะต้องแบ่งข้อมูลต่อเนื่องเป็นข้อมูลแบบกลุ่มเพื่อความสะดวกในการดูจำนวนความถี่ในแต่ละกลุ่ม เช่น หากรายงานว่า ร้อยละ 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	เป็นวิธีการของการเลือกควอนไทล์ทางพีชคณิต 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 แล 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. จงแบ่งกลุ่มค่าใช้จ่ายสูงสุดสำหรับการรับคำปรึกษาและตรวจเลือดที่คุณยินดีและสามารถจ่ายได้เป็น 2 กลุ่ม คือ 1= 100 บาทลงมา และ 2= สูงกว่า 100 บาท และสร้างเป็นตัวแปรใหม่ในกรอบข้อมูล
6. จงแบ่งกลุ่มรายได้ออกเป็น 3 กลุ่มที่มีความถี่ใกล้เคียงกัน

บทที่ 5

สถิติที่ใช้ในการวิเคราะห์ข้อมูลแบบกลุ่ม

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

1. การศึกษาแบบพรรณา
    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 การคำนวณทางสถิติในการศึกษาแบบพรรณา

ในการศึกษาแบบพรรณานั้น สถิติที่ใช้ก็จะเป็นสถิติในเชิงพรรณา ได้แก่การแจกแจงความถี่หรือสัดส่วน (proportion) หรือบอกเป็นค่าเฉลี่ยหรือมัธยฐาน หรือบอกเป็นค่าดัชนีสุขภาพ เช่น อุบัติการณ์ ความชุก อัตราตาย อัตราการอยู่รอด แม้จะดูเหมือนว่าค่าดัชนีสุขภาพเป็นเรื่องที่ต้องคำนวณซับซ้อน แต่จริงๆ แล้วความหมายของดัชนีสุขภาพส่วนใหญ่จะคล้ายคลึงกับการแจกแจงความถี่ เช่น อัตราอุบัติการณ์ของการเกิดมะเร็งในประชากรในพื้นที่หนึ่งในช่วงเวลาหนึ่ง ก็คำนวณจากจำนวนคนที่เป็นมะเร็งใหม่ในพื้นที่นั้นในช่วงเวลานั้น หารด้วยจำนวนประชากรทั้งหมดในพื้นที่นั้นที่กลางช่วงเวลานั้น ซึ่งก็จะมีความหมายเช่นเดียวกับการแจกแจงความถี่นั่นเอง แต่เนื่องจากค่าที่ได้มีค่าน้อย ก็จะนิยมใช้หน่วยเป็น ต่อแสนประชากร แทนที่จะเป็นร้อยละ สำหรับดัชนีอย่างอื่นอาจใช้หน่วยเป็นต่อหนึ่งพันหรือต่อหนึ่งหมื่น ก็แล้วแต่ความนิยมของผู้ที่อยู่ในสาขานั้นๆ

ในกรณีเช่นนี้การแสดงนัยสำคัญทางสถิติจะนิยมบอกเป็นช่วงความเชื่อมั่นร้อยละ 95 ซึ่งคำนวณจากการหาค่า standard error (SE) แม้ว่าการคำนวณค่า SE สำหรับดัชนีสุขภาพแต่ละอย่างอาจจะมีความแตกต่างกันไปบ้าง แต่ความหมายก็เป็นเช่นเดียวกันทั้งหมด คือเป็นความเบี่ยงเบนที่อาจเกิดขึ้นกับการประมาณค่าดัชนีนั้นจากการสุ่มตัวอย่างในประชากรหนึ่งในช่วงเวลาหนึ่งนั่นเอง ในที่นี้จะกล่าวถึงเฉพาะการคำนวณค่า SE ของค่าสัดส่วนตามปกติ ซึ่งมีค่าดังนี้

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

โดยที่ p คือค่า proportion นั้น N คือจำนวนตัวอย่างทั้งหมด

ดังตัวอย่าง การศึกษาหนึ่งพบว่าในช่วงปี 1995-1997 มีประชากรชายในจังหวัดสงขลาเป็นมะเร็งปอดทั้งหมด 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

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

$$= 0.6362,\quad 0.7449$$

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 แต่พึงสังเกตว่าวัตถุประสงค์ของการศึกษาวิจัยใด 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 หรือความจำเพาะที่ต้องการหานั้น แต่พึงระลึกไว้ว่าค่า N ในการหาช่วงความเชื่อมั่นของค่าความไวและความจำเพาะนั้นแตกต่างกัน

ความไว (sensitivity) *คือสัดส่วน (*proportion*) ของคนที่เป็นโรคจริงๆ ในประชากร ที่การตรวจก็ให้ผลว่าเป็นโรคเช่นกัน*

ความจำเพาะ (specificity) *คือสัดส่วน (*proportion*) ของคนที่ไม่เป็นโรคจริงๆ ในประชากร ที่การตรวจก็ให้ผลว่าไม่เป็นโรค*

เนื่องจากคำนิยามนี้ค่อนข้างทำความเข้าใจได้ยากหากไม่อธิบาย ขอให้พิจารณาจากตาราง 2×2 ต่อไปนี้

		disease +	disease -
test	+	a	b
	-	c	d

ในที่นี้จำนวนคนที่เป็นโรคในประชากรมีค่าเท่ากับ a+c และในจำนวนนี้มี a คนที่ผลการตรวจก็ให้ผลว่าเป็นโรคด้วยเช่นกัน นั่นคือ

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

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 จะต้องไม่มีความถี่คาดหวังเซลใดเซลหนึ่งที่น้อยกว่า 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

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

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

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 ได้แก่คำสั่ง **`chisq.test`**() และ **`fisher.test()`** แต่เนื่องจากคำสั่งทั้งสองนี้ไม่ได้แสดงตารางข้อมูลประกอบด้วย และรูปแบบการแสดงผลก็ดูค่อนข้างยาก ใน library **ice** จึงได้มีคำสั่งที่แสดงตารางข้อมูลประกอบกับการทดสอบทั้งสองให้เลย ได้แก่ tab และxtab 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 ถ้าระบุแต่ 'x' เพียงค่าเดียวจะสร้างตารางความถี่
`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

`frame`	วาดกรอบให้กับตาราง ค่าปกติเป็น TRUE
`na.included`	แสดง NA เป็นอีกระดับหนึ่งหรือไม่ ค่าปกติคือ FALSE
`factor.labels`	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)
```

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

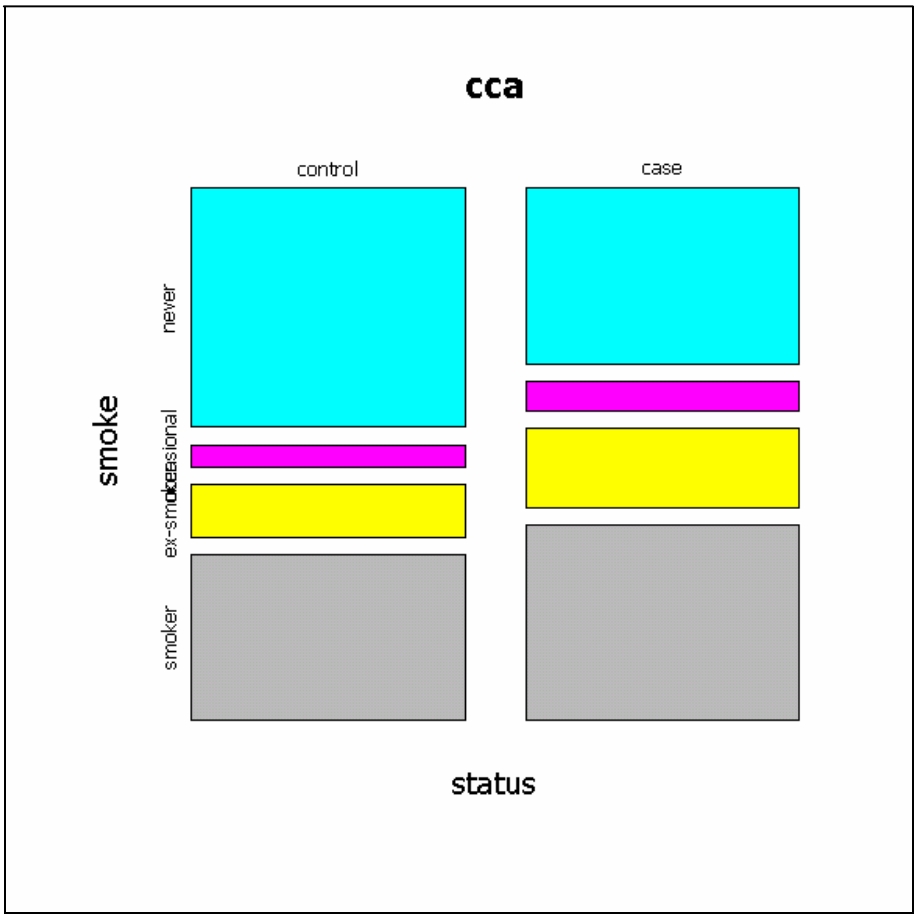
อยู่ใน library graphics มีรูปแบบการออกคำสั่งดังนี้

```
mosaicplot(x, main = deparse(substitute(x)),                    ❶
      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"), ...)

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

`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 ~y+x
`data`	ชื่อกรอบข้อมูล
`subset`	การเลือกข้อมูลย่อยในกรอบข้อมูล
`na.action`	ในกรณีของข้อมูลที่มีค่า missing ให้แสดงค่าเหล่านี้หรือไม่

ในที่นี้ใช้ตัวอย่างคำสั่งที่ ❷ ดังตัวอย่างการใช้คำสั่ง

```
> mosaicplot(~status+smoke, col=5:8, 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()

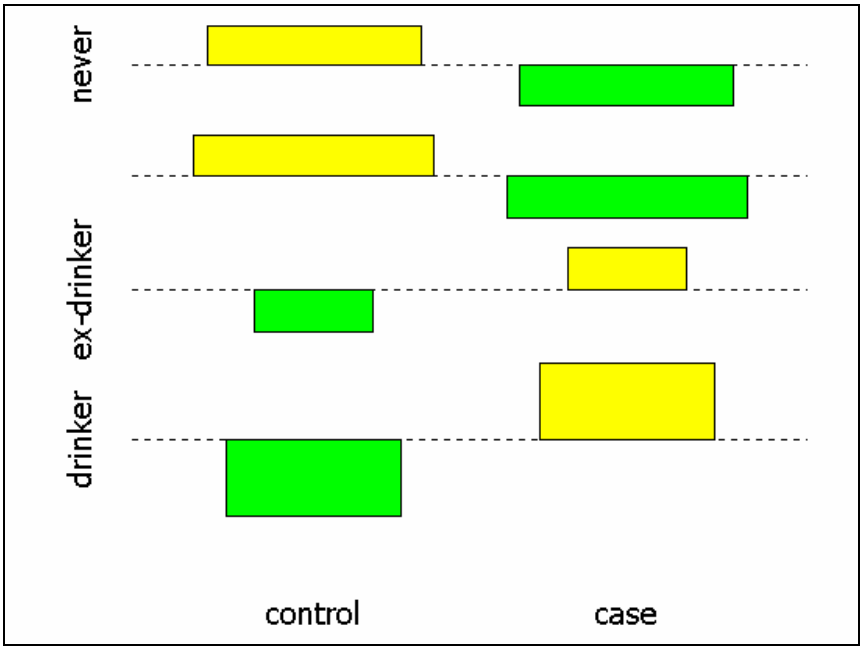
เป็นคำสั่งที่สร้างแผนภาพ Cohen-Friendly association plot เพื่อทิศทางความสัมพันธ์ในแต่ละกลุ่ม ความสูงของแท่งแต่ละแทงเป็นสัดส่วนตามค่า Chi-squared ในแต่ละเซล ส่วนความกว้างเป็นสัดส่วนตามค่าความถี่คาดหวัง ดั้งนั้นพื้นที่ของแต่ละแท่งแสดงสัดส่วนของความแตกต่างระหว่างค่าความถี่จากการสังเกต กับความถี่คาดหวัง 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`	กำหนดระยะห่างระหว่างแผนภูมแต่ละแท่ง
`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 อาจเรียกรวมกันว่าเป็นการวิจัยเชิงวิเคราะห์ (analytical studies) แบ่งได้เป็นสองกลุ่มหลัก คือการวิจัยเชิงสังเกต (observational) และการวิจัยเชิงทดลอง (experimental) ในการวิจัยเชิงสังเกตนั้น ไม่ได้มีการให้ปัจจัยที่ต้องการศึกษาแก่คนที่ผู้วิจัยสังเกต เพียงแต่แบ่งกลุ่มผู้ถูกวิจัยเป็นกลุ่มๆ ตามปัจจัย และ/หรือโรคที่คนเหล่านั้นได้รับเองหรือเป็นโรคอยู่ ส่วนการวิจัยเชิงทดลองนั้น ผู้วิจัยได้ให้ปัจจัยเสี่ยง (ที่ไม่เป็นอันตรายถึงแก่ชีวิตหรือพิการ) หรือโดยส่วนใหญ่แล้ว จะเป็นยารักษาหรือสิ่งที่ใช้ป้องกันโรค แล้วตามดูผลที่จะเกิดขึ้นในการป้องกันโรคหรือภาวะแทรกซ้อนอันตราย หรือแม้กระทั่งการเสียชีวิต

ในกลุ่มการวิจัยเชิงสังเกตนั้น มีวิธีการวิจัยหลักอยู่ 2 แบบคือ การศึกษาแบบ case-control และการศึกษาแบบ cohort ซึ่งการศึกษาแบบ case-control นั้น เป็นการศึกษาย้อนหลัง โดยเลือกผู้ที่เป็นโรคแล้ว กับผู้ที่ไม่เป็นโรคจากประชากรเดียวกัน มาศึกษาย้อนหลังว่าได้รับปัจจัยเสี่ยง (exposure) ที่สนใจบ้างหรือเปล่า มากน้อยเพียงใด แล้วนำมาเปรียบเทียบกัน ส่วนการศึกษาแบบ cohort นั้น จะแบ่งกลุ่มผู้เข้าร่วมการศึกษาตามปัจจัยเสี่ยงที่ได้รับตามปกติที่เขาเป็นอยู่ โดยผู้วิจัยไม่ได้จงใจให้ปัจจัยนั้นแก่บุคคลเหล่านั้น 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 ในกรณีนี้การเปรียบเทียบจะทำในคนเดียวกันนั้นเอง ดูผลที่ได้รับปัจจัยหนึ่งเทียบกับผลที่เกิดขึ้นจากอีกปัจจัยหนึ่ง ซึ่งจะกำจัดการแปรปรวน (variation) ที่อาจเกิดจากการแบ่งกลุ่มผู้วิจัยไม่สมดุลกัน

ในการวิจัยกลุ่มต่างๆ ที่กล่าวมาแล้วนี้ แต่ละอย่างยังมีรายละเอียดปลีกย่อยที่แตกต่างกันได้อีก ทำให้เรียกชื่อวิธีการวิจัยแตกต่างกันออกไปอีก ซึ่งจะไม่กล่าวถึงในที่นี้ แต่จะกล่าวถึงเฉพาะหลักการทั่วไปของการวิเคราะห์ปัจจัยเสี่ยงและปัจจัยป้องกัน

เพื่อให้ง่ายต่อการทำความเข้าใจ ในเบื้องต้นขอให้พิจารณาจากกรณีของตัวแปร แบบ dichotomous คือเป็นไปได้สองทางดังต่อไปนี้

ปกติในประชากรกลุ่มใดกลุ่มหนึ่งย่อมมีทั้งผู้ที่เป็นโรคและผู้ที่ไม่เป็นโรคอยู่ปะปนกัน และมีทั้งผู้ที่มีและไม่มีปัจจัยที่สนใจว่าจะเป็นปัจจัยเสี่ยง ซึ่งอาจแสดงด้วยตาราง 2 × 2 ดังนี้

		การเกิดโรค D: +	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(OR) หาได้โดยถือเสียว่าการกระจายของ ln(OR) เป็นแบบปกติ มีสูตรการคำนวณดังนี้

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

ดังนั้นช่วงความเชื่อมั่นของ ln(OR) จะมีค่า

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

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

และนอกจากนี้ หากในประชากรนั้นๆ มีความชุกของโรคน้อย นั่นคือว่า b มีค่ามากกว่า a มาก จนทำให้ค่า a+b มีค่าไม่ต่างจาก a และ d มีค่ามากกว่า c มาก จนทำให้ค่า c+d มีค่าเท่าๆ กับ d เราจะพบว่า relative risk จะมีค่าเท่าๆ กับ 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

ข้อสรุปเช่นนี้แม้จะดูพื้นๆ 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 ที่อนุมานเอาจาก 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 ที่อนุมานโดยวิธีนี้ก็จะคลาดเคลื่อนจากความเป็นจริงได้มาก ในกรณีเช่นนี้ควรบอกว่าเป็นค่า OR โดยตรงจะดีกว่าที่จะอนุมานว่าเป็นค่าของ RR

ในการวิจัยทางระบาดวิทยาจริงๆ แล้ว เรามักไม่คำนวณหาค่า RR หรือ OR โดยวิธีง่ายๆ ดังที่กล่าวข้างต้น เนื่องจากต้องคำนึงถึงปัจจัยกวนอื่นๆ ด้วย แต่จะใช้การคำนวณโดยใช้โมเดลทางสถิติ ซึ่งจะได้กล่าวถึงในบทต่อๆ ไป

การกวน (Confounding)

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

C D C D

โดยที่ E คือปัจจัยที่ได้รับ (exposure) D คือโรค (disease) และ C คือตัวกวน (confounder) นั่นคือทั้ง E และ C มีความสัมพันธ์ในเชิงสาเหตุกับโรค D ทั้งคู่ และ C กับ E ยังมีความสัมพันธ์กันเองอีกด้วย ในลักษณะที่ว่ามักพบร่วมกัน เช่น ผู้ที่ดื่มเหล้ามักสูบบุหรี่ หรือหากพบอย่างหนึ่งก็จะไม่ค่อยพบอีกอย่างหนึ่งก็ได้

หากไม่มีการปรับอิทธิพลของตัวกวนแล้ว ความสัมพันธ์ระหว่างปัจจัยที่ได้รับกับโรค อาจมีค่ามากหรือน้อยกว่าความเป็นจริงก็ได้ทั้งสองทางแล้วแต่กรณี เช่นในกรณีการเกิดมะเร็งปอดกับอาชีพผู้ใช้แรงงาน ส่วนใหญ่ผู้ใช้แรงงานจะสูบบุหรี่มากกว่าอาชีพที่ใช้ความชำนาญหรือผู้บริหาร และการสูบบุหรี่ก็เป็นปัจจัยเสี่ยงของมะเร็งปอด ดังนั้น หากไม่คำนึงถึงอิทธิพลของการสูบบุหรี่เข้าด้วยแล้ว เราก็จะพบว่าผู้ที่มีอาชีพใช้แรงงานมักเป็นมะเร็งปอดด้วยความสัมพันธ์ที่มากเกินจริง

ในกรณีของมะเร็งเต้านมกับการใช้ยาคุมกำเนิน estrogen เพื่อรักษาอาการในวัยหมดประจำเดือน การกวนเกิดจากอายุที่หมดประจำเดือน ผู้ที่หมดประจำเดือนเร็วมีความเสี่ยงต่อการเกิดมะเร็งเต้านมน้อย แต่ก็จะมีความจำเป็นในการได้รับ estrogen มากและเร็วขึ้น ในกรณีเช่นนี้ การไม่คำนึงถึงอิทธิพลของตัวกวนคืออายุที่หมดประจำเดือน ก็อาจส่งผลให้ความสัมพันธ์ระหว่างยาคุมกำเนิด estrogen กับมะเร็งเต้านมที่ต้องการทราบนั้นมีน้อยกว่าความเป็นจริง

เพื่อจะให้เข้าใจหลักการของการกวน ในที่นี้ขอเริ่มต้นอย่างง่ายๆ จากการพิจารณากรณีของตัวแปรแบบมีสองระดับ (dichotomous) คือเป็นไปได้สองกรณี โดยที่ทั้ง C, E และ D เป็นสองระดับทั้งสิ้นดังที่ได้กล่าวไว้ก่อนแล้วในหัวข้อที่ 2

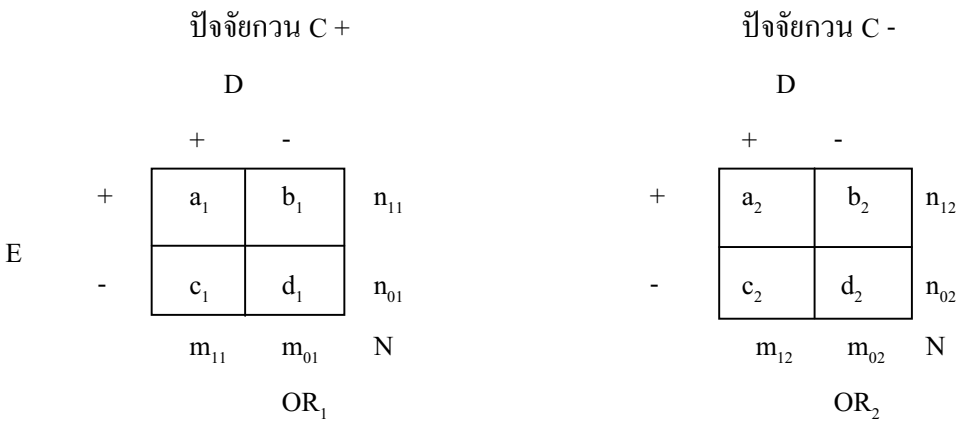
การเกิดโรค D

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

ซึ่งจะมีค่า $OR_p = ad/bc$ โดยที่ OR_p ในที่นี้หมายถึง OR ที่คำนวณจากข้อมูลทั้งหมด (pooled data)

แต่ถ้าคิดว่าความสัมพันธ์ระหว่างการได้รับปัจจัยเสี่ยง E กับโรค D อาจมีส่วนหนึ่งเกิดจากความสัมพันธ์ระหว่างปัจจัยกวน C กับทั้งปัจจัยเสี่ยง E และกับโรค D จะได้ความสัมพันธ์ระหว่าง

ปัจจัยเสี่ยง E กับโรค D ดังนี้



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 แต่ต้องการกำจัดอิ ทธิพลของการกวนในการคำนวณขนาดของความสัมพันธ์ระหว่างปัจจัยเสี่ยงกับโรคต่างหาก เราพบว่า confounding risk ratio จะมีค่ามากกว่า 1 เมื่อ

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

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

ทั้งปัจจัย E และ C อาจเป็นค่าที่แบ่งเป็นหลายระดับ ไม่ใช่แค่สองระดับก็ได้ ซึ่งแนวคิดของการกวนก็ยังคงเป็นเช่นเดียวกับตัวแปรที่แบ่งเป็นสองระดับ 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 ซึ่งต้องมีค่าที่เป็นไปได้เพียงสองค่า เท่านั้น คือระบุว่าเป็น control หรือ case กับ unexposed หรือ exposed
`table`	วัตถุตาราง
`data`	กรอบข้อมูล (อาจไม่ระบุก็ได้)
`case, expose`	เวกเตอร์ของ case และ exposure ทั้งสองต้องมีค่าเป็นไปได้สองทางเท่านั้น 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 ของความสัมพันธ์ระหว่าง การเป็น cholangiocarcinoma กับ antibody to *Opistorchis 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 และ ที่เหลือเป็นอีกชั้นหนึ่ง 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 โดยไม่สนใจว่ารหัสนั้นมีความหมายอย่างไร ดังนี้ในกรณีของ alc2 และ smo2 ที่เราใช้เลข 0 กับเลข 1 เป็นรหัสนั้น **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 exposure ทั้งสองต้องมีสองระดับ นั่นคือ case หรือ control ในกรณีของ case และ exposed หรือ unexposed ในกรณีของ expose
`group`	เวกเตอร์ที่ใช้แบ่งชั้น ซึ่งต้องเป็นตัวแปรชนิดจำนวนนับ ซึ่งอาจมีสองระดับชั้นหรือมากกว่าก็ได้
`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

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

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

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

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

แต่เนื่องจากจะเป็นการยุ่งยากเกินไป จึงได้สร้างคำสั่งสำเร็จรูปไว้ให้แล้วใน library **ice** 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 จากข้อมูลวิจัยมะเร็ง 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 มีข้อจำกัดสำคัญตรงที่จะใส่ตัวกวนเข้าไปปรับค่า odds ratio ได้เพียงตัวเดียวเท่านั้น หากในงานวิจัยหนึ่ง มีตัวแปรที่เป็นตัวกวนของความสัมพันธ์ระหว่าง exposure กับ outcome ที่สนใจมากกว่าหนึ่งตัวแล้ว การใช้คำสั่ง mhor ก็จะไม่สามารถหาค่า odds ratio ที่ปรับอิทธิพลของตัวกวนมากกว่าหนึ่งตัวในคราวเดียวกันได้ 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 logit p จึงเป็น log 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 จะต้องมีค่า 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`	ตัวเลือกกรอบข้อมูลที่มีตัวแปรในโมเดลอยู่ หากตัวแปรเหล่านั้นอยู่ในกรอบข้อมูลเดียวกันทั้งหมด
`subset`	ตัวเลือกระบุกลุ่มย่อยของข้อมูลที่จะใช้ในการ fit
`frame`	วาดตารางมีกรอบ ค่าปกติเป็น TRUE

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

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 Akaiki Information Criterion (AIC) ให้ ค่า AIC มีความสัมพันธ์กับค่า log-likelihood คือ AIC = -2*log-likelihood + k*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 สัมพัทธ์กับทั้ง anc และ clinic (อย่าลืมว่าวัตถุประสงค์หลักของการวิจัยคือการหาความสัมพันธ์ระหว่าง death กับ anc) แต่เมื่อเรารู้อยู่แล้วว่า anc มีความสัมพันธ์กับ clinic อยู่ตามที่กล่าวแล้วในบทก่อน 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 ตีความผลลัพธ์ของ `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 เพียงแต่จะแสดงให้ทราบไว้ว่ากระบวนการหาความสัมพันธ์นั้น 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, ดื่มเหล้า, สูบบุหรี่, GSTM1 และอื่นๆ (ตัดตัวแปรมาให้ในที่นี้เพียงแค่นี้) 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+gstm1,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 มักจะหมายความว่าการใส่ตัวแปรนั้นไว้จะอธิบายชุดข้อมูลได้ดีกว่าเดิม แต่อาจเป็นไปได้ว่าขนาดตัวอย่างไม่พอที่จะทำให้ค่า p-value ของตัวแปรนั้นมีนัยสำคัญได้ เราก็จะเป็นต้องเก็บตัวแปรนั้นไว้ในโมเดล

ส่วนการเริ่มต้นจาก full model แล้วตัดตัวแปรออกทีละตัวเป็นดังนี้

```
cca <- fread("cca.tab")
model1 <- logistic(status~ovab+alcohol+smoke+gstm1,data=cca)
model2 <- logistic(status~ovab+alcohol+gstm1,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 ต้องใช้การทดสอบเพื่อหาว่าค่า mean ของกลุ่มเหล่านั้นแตกต่างกันหรือไม่ มีวิธีทดสอบมาตรฐานคือ t-test ซึ่งวิธีนี้มีเงื่อนไขที่ต้องดูก่อนอยู่ 2 อย่าง คือ 1. การกระจายของตัวแปรในทั้งสองกลุ่มที่จะนำมาเปรียบเทียบกันนั้น 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 ดูง่ายกว่า 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) หรือกรอบข้อมูลที่มีตัวแปรที่ระบุใน 'formula' (รูปแบบที่ 2)

`x` เวกเตอร์ตัวเลขที่มีข้อมูลที่ต้องการทดสอบ

`y` เวกเตอร์ตัวเลขที่มีข้อมูลอีกชุดหนึ่ง ใช้เมื่อต้องการทดสอบแบบ paired (paired = TRUE)

`mu` ตัวเลขระบุค่าจริงของค่าเฉลี่ย (หรือความต่างของค่าเฉลี่ย ถ้ากำลังทดสอบแบบสองตัวอย่าง)

`paired` ค่าตรรกะระบุว่าต้องการทดสอบ paired t-test หรือไม่

`var.equal` ค่าตรรกะระบุว่าต้องการให้คำสั่งถือเสมือนว่า variance ของสองกลุ่มเท่ากันหรือไม่ ถ้าระบุเป็น TRUE ค่า variance รวมจะถูกนำมาใช้ 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 ก็ใกล้กัน ทำให้พอจะคิดได้ว่า การกระจายของ age ในทั้งสองกลุ่มมีค่าใกล้เคียงกับการกระจายแบบปกติ อย่างไรก็ตามมีการทดสอบการกระจายว่าเป็นแบบปกติหรือไม่ 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 อาจเรียกตัวแปรโดยชื่อหรือเลขสดมภ์ก็ได้

`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) ด้วย โดยวัดครั้งแรกก่อนให้อาหารเสริม 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

หากต้องการเปรียบเทียบทีเดียว 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`	เงื่อนไขเซ็ตย่อย
`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 delimeted ตั้งชื่อเป็น `CRP2.tab` ใส่ใน R workplace ที่กำลังทำงานอยู่ (ที่เดียวกับ folder ของ `CRP.tab` เดิมนั้นเอง)

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` ให้มีแต่เฉพาะตัวแปร `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** จะรู้สึกว่าวิธีแรกจะสะดวกและเข้าใจง่ายกว่ามาก แต่หากชำนาญการใช้คำสั่งของ **R** แล้ว ก็จะรู้สึกว่าวิธีหลังก็ทำได้ง่ายเช่นกัน 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 แต่วิธีนี้จะไม่เห็นว่าค่าเพิ่มขึ้นหรือลดลง หรือค่าของการวัดครั้งไหนที่ต่างไปกว่ากลุ่มอื่น จึงควรทำ t-test ดูทีละคู่ด้วย

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 เป็นการใช้ลำดับ (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 yr~x1+x2

`data` กรอบข้อมูล

`subset` การระบุการเลือกข้อมูล

`weights` การถ่วงน้ำหนักในกระบวนการ fit model

`na.action` ตัวเลือกในการทำงานกับข้อมูล missing โดยค่า default คือ na.omit

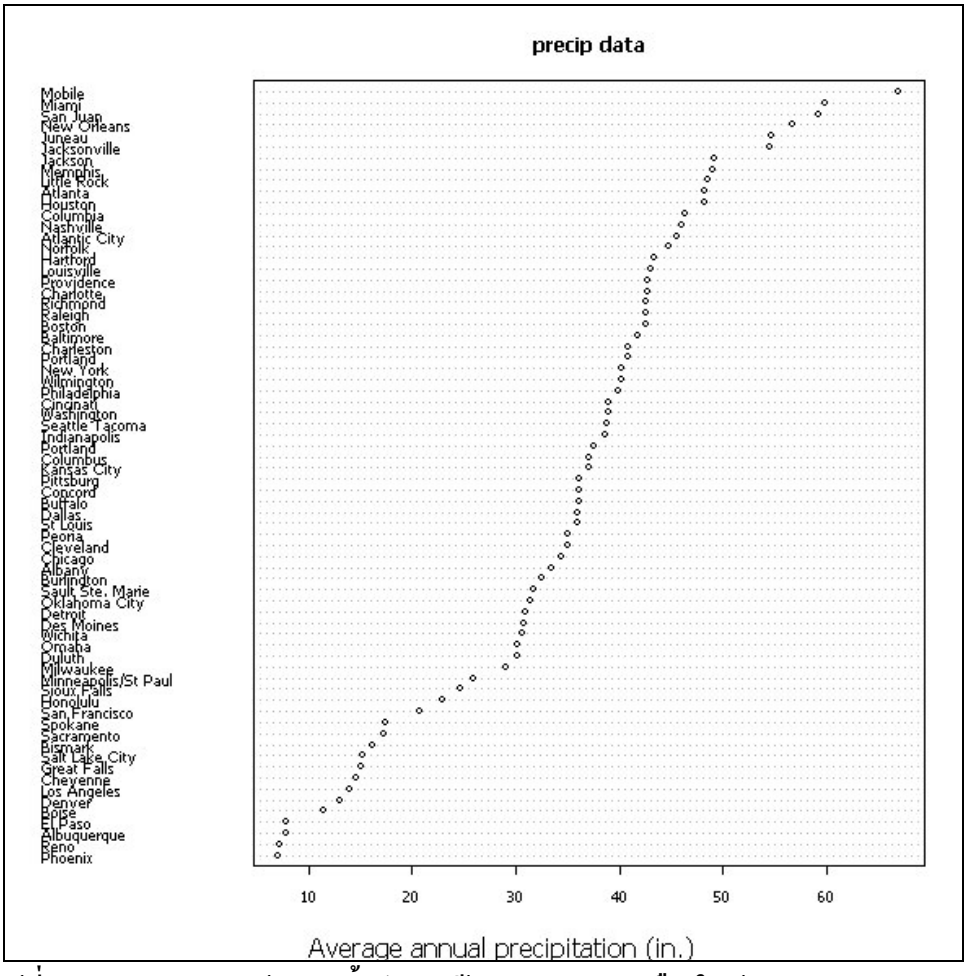
`method` วิธีการที่เลือก

`model,x,y,gr` ถ้ากำหนดให้เป็น TRUE ก็จะให้ค่าขององค์ประกอบเหล่านี้ออกมา

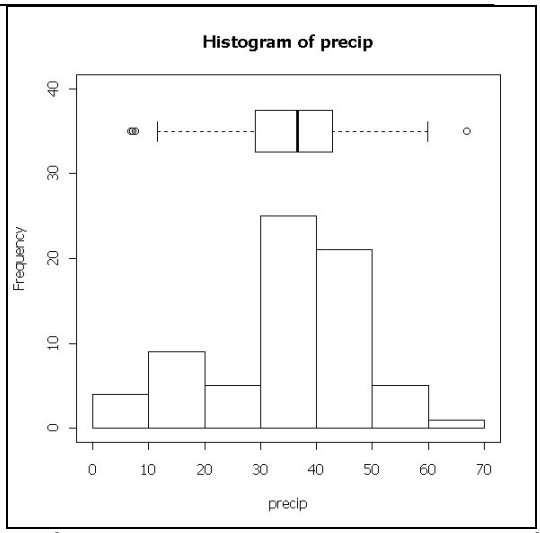
ตัวอย่างที่ 6.3 ชุดข้อมูล 'precip' บันทึกปริมาณเฉลี่ยของน้ำฝนหน่วยเป็นนิ้ว ใน 70 เมืองของประเทศสหรัฐอเมริกาและประเทศเปอโตริโก *(Puerto Rico)* จงสร้างกราฟเพื่อดูการแจกแจงของข้อมูลชุดนี้และข้อมูลมีการแจกแจงปกติหรือไม่

เพื่อให้เห็นการแจกแจงของข้อมูล เราสามารถที่จะใช้คำสั่ง *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

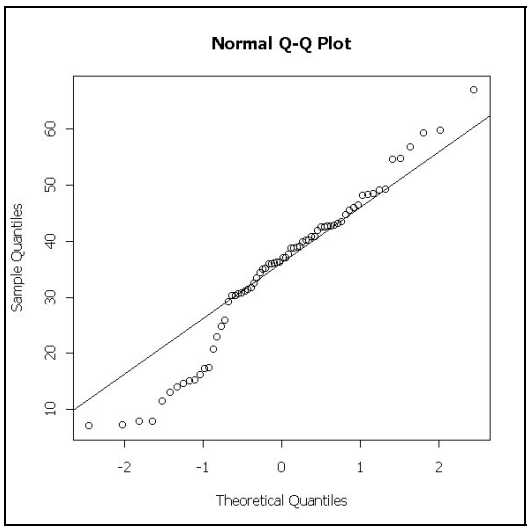
ค่าที่ประมาณจากฟังก์ชันการแจกแจง ในขณะที่เส้นประแสดงค่าที่ประมาณจากฟังก์ชันการแจกแจงที่มีการแจกแจงปกติ ด้วยค่าเฉลี่ย = *mean(precip)* และส่วนเบี่ยงเบนมาตรฐาน = *sd(precip)*.



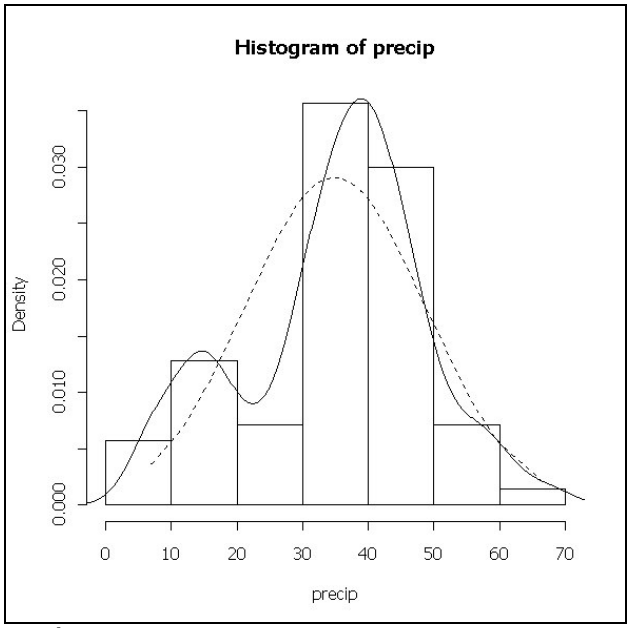
รูปที่ 6. 3 Dotplot ของปริมาณน้ำฝนรายปีจากจำนวน 70 เมืองในประเทศสหรัฐอเมริกา



รูปที่ 6. 4 ฮีสโตแกรมพร้อมด้วย boxplot



รูปที่ 6. 5 Q-Q plot สำหรับการประเมินการแจกแจง



รูปที่ 6. 6 ฮีสโตแกรมพร้อมด้วยเส้นแสดงการแจกแจง

ตัวอย่างคำสั่ง ดังต่อไปนี้

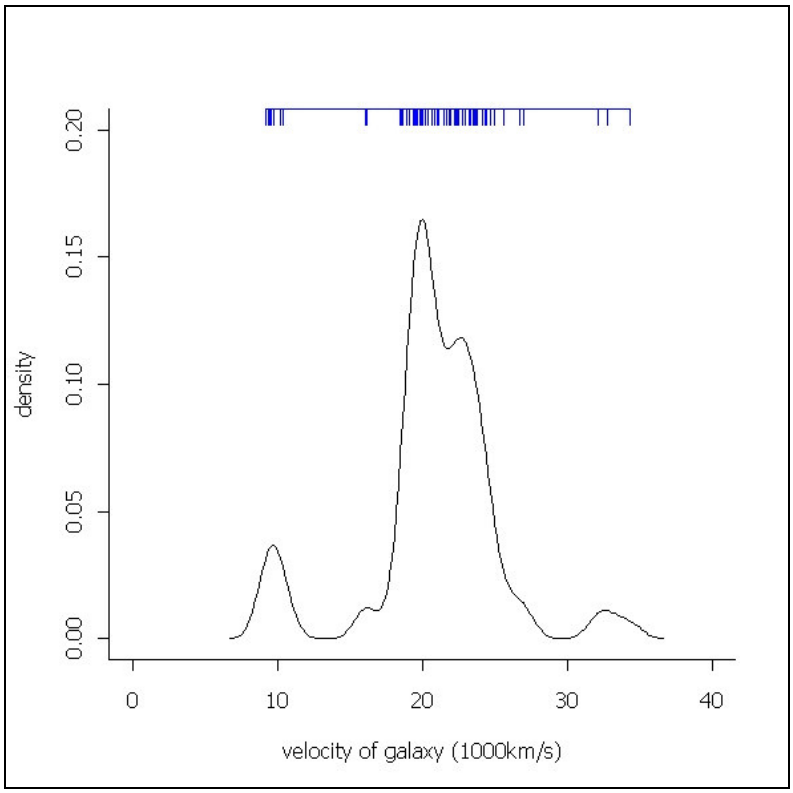
```
> library(MASS)
> data(precip)
> dotchart(precip[order(precip)], main = "precip data", cex=0.6)
> title(sub = "Average annual precipitation (in.)")

> 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
```



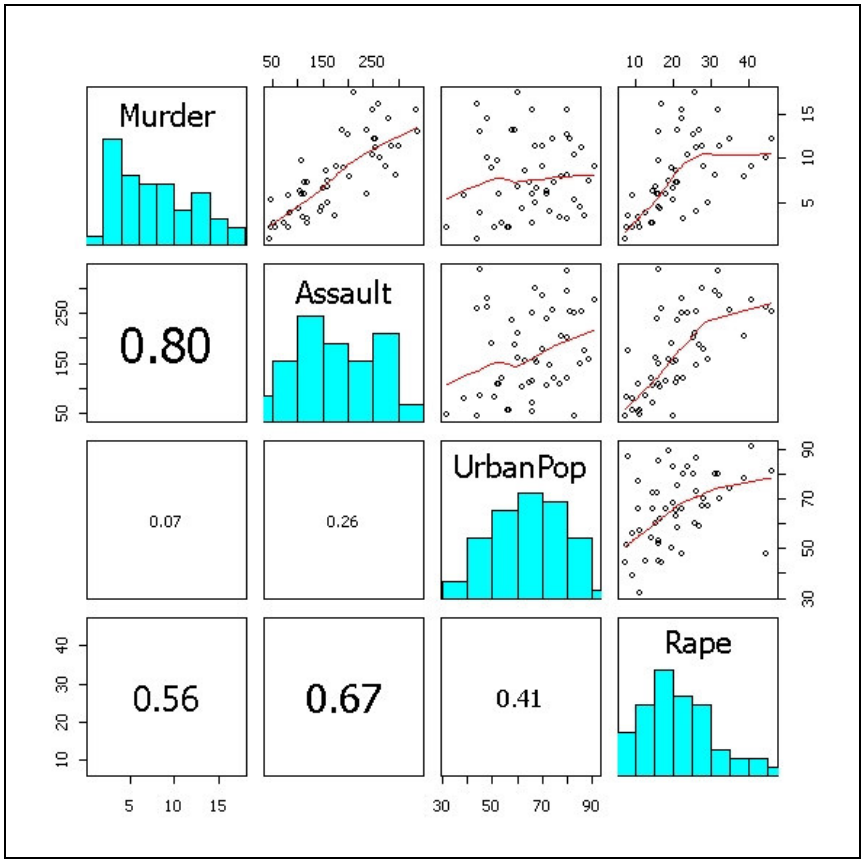
รูปที่ 6. 7 Density plot ของข้อมูลหมู่ดาว

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

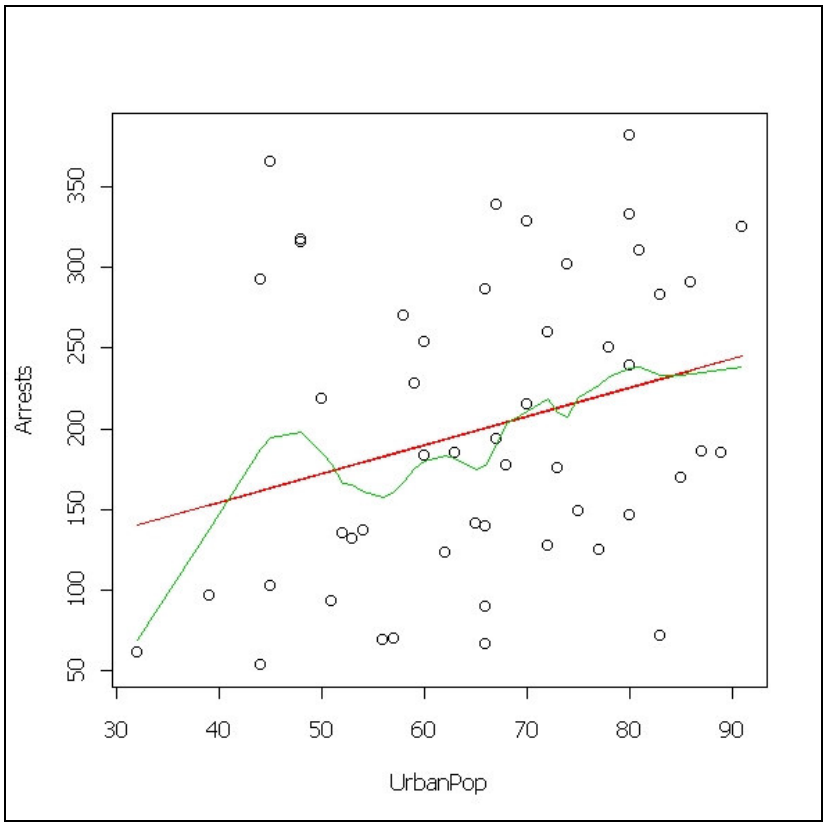
สามารถใช้คำสั่ง *pairs* เพื่อตรวจสอบความสัมพันธ์ระหว่างตัวแปร ความสัมพันธ์นี้แสดงด้วยการใช้ตัวเลือก `panel` ในคำสั่ง **`pairs()`** รูปที่ 6.8 แสดง scatter plot matrix ของชุดข้อมูลนี้ พบว่า จำนวนการข่มขืนและการทำร้ายร่างกายสูงขึ้นในรัฐที่มีเปอร์เซ็นต์ของประชากรในเขตเมืองสูง ในขณะที่ความถี่ของจำนวนการฆาตกรรมไม่ได้ขึ้นกับเปอร์เซ็นต์ของประชากรในเขตเมือง จำนวนการฆาตกรรมสูงขึ้นเมื่อจำนวนการทำร้ายร่างการสูงขึ้น 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



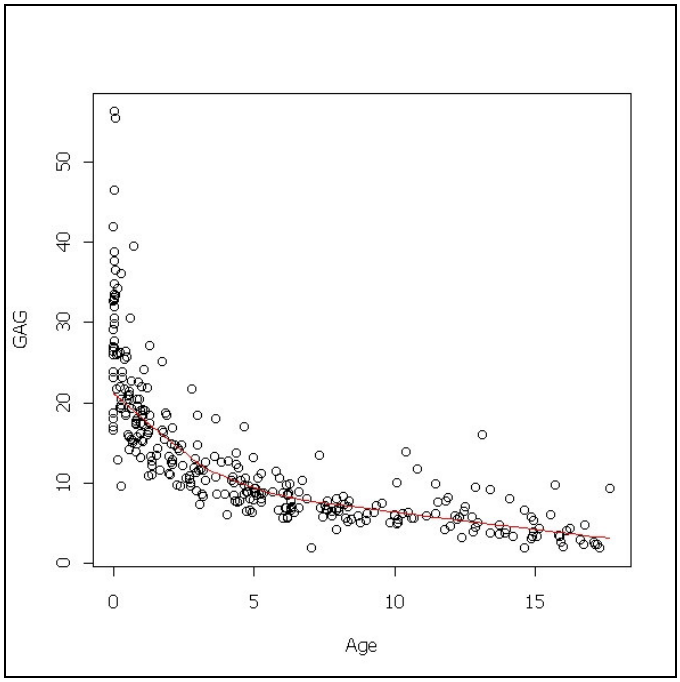
รูปที่ 6.9 Scatterplot พร้อมด้วยเส้น predict สำหรับข้อมูลการจับกุมในประเทศสหรัฐอเมริกา

ตัวอย่างที่ 6.6 ชุดข้อมูล '*GAGurine*' ใน library 'MASS' ประกอบด้วยความเข้มข้นทางเคมีของ *GAG* ในปัสสาวะเด็กจำนวน 314 คน ช่วงอายุ 0 – 7 ปี วิเคราะห์ข้อมูล และสร้างแผนภูมิเพื่อช่วยกุมารแพทย์ในการประเมินความเข้มข้นของค่า *GAG* ว่ามีการแจกแจงปกติหรือไม่

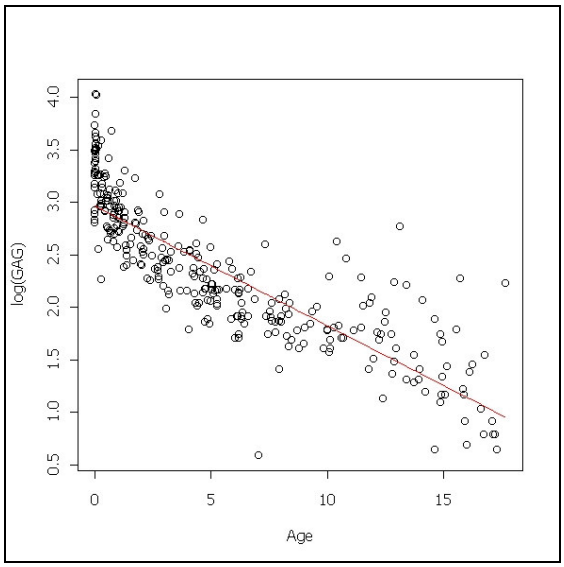
จากรูปที่ 6.10 จะพบว่าเด็กอายุน้อย มีความเข้มข้นของค่า *GAG* ในปัสสาวะสูง เนื่องจากการแจกแจงเป็นแบบ *exponential* โมเดล *log-linear* แสดงดังรูปที่ 6.11 จากรูปที่ 6.13 จะพบว่า *cubic model* จะเป็นโมเดลที่ดีที่สุด ซึ่งแสดงพร้อมเส้นการประมาณช่วงค่า (prediction interval)

ตัวอย่างคำสั่งดังต่อไปนี้

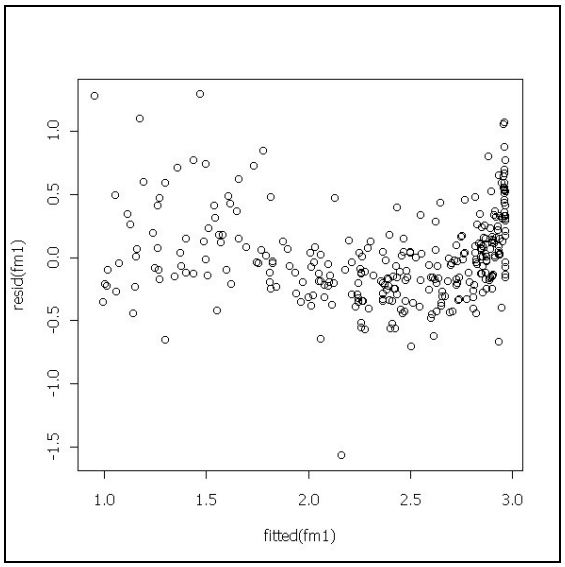
```
> library(MASS)
> data(GAGurine)
> plot(GAGurine)
> lines(lowess(GAGurine), col=2)

> 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)
```



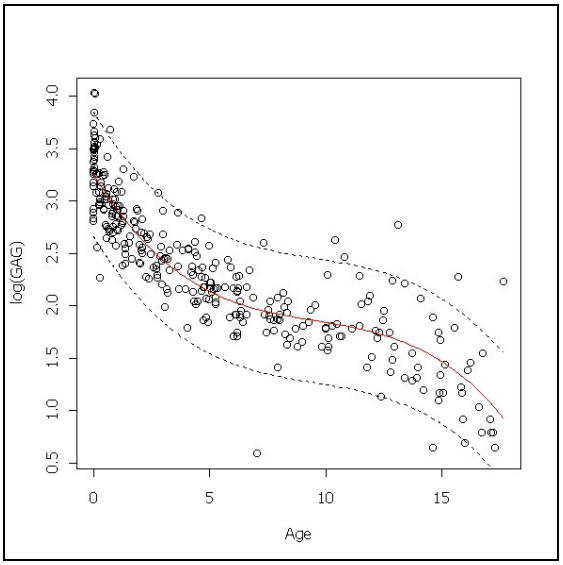
รูปที่ 6.10 Scatterplot สำหรับข้อมูล GAG



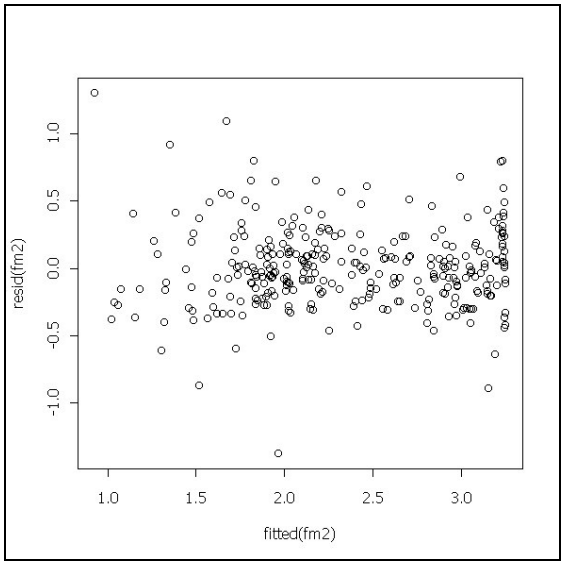
รูปที่ 6. 11 การสร้างโมเดล log-linear



รูปที่ 6. 12 Residuals vs fitted values



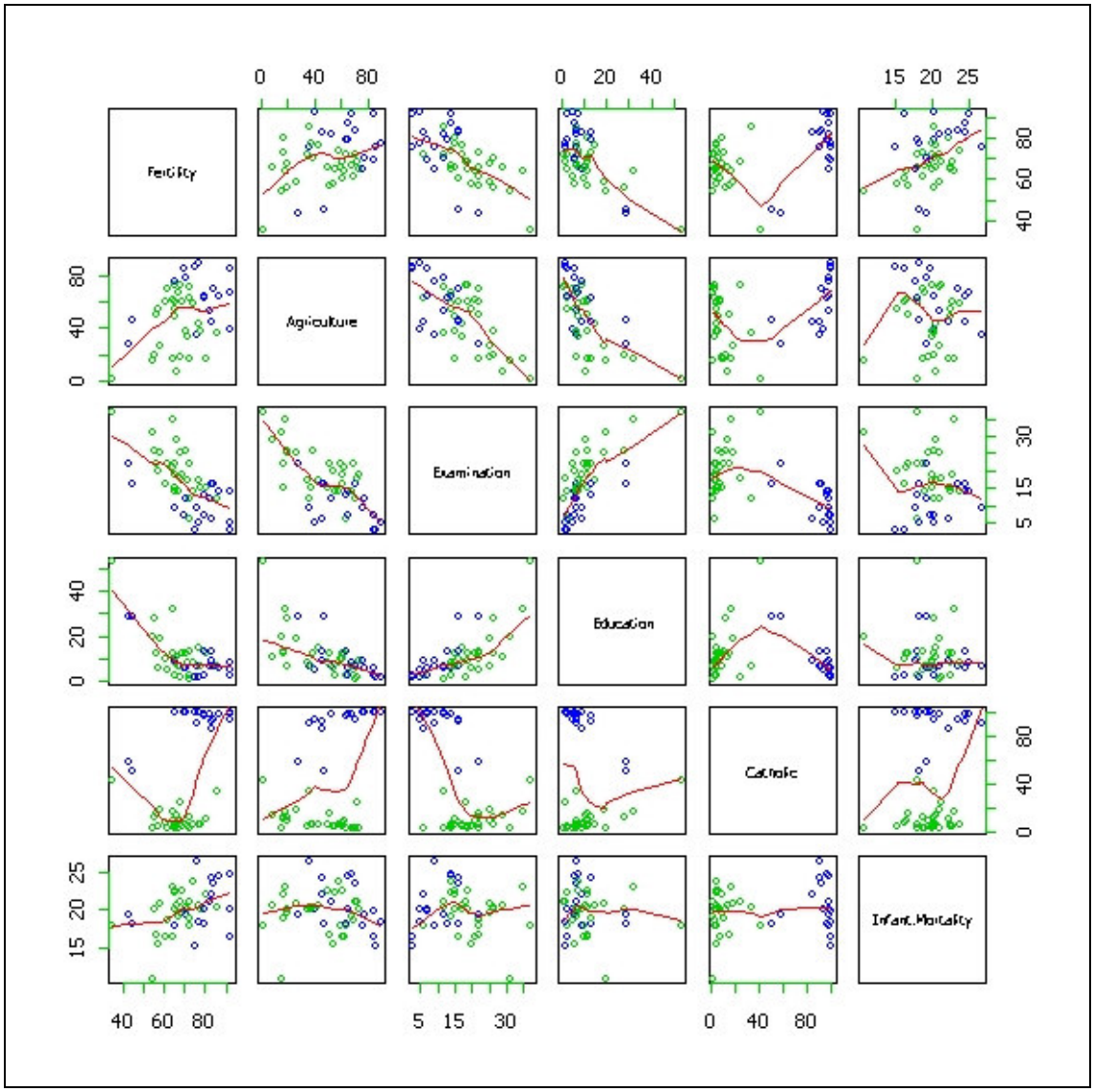
รูปที่ 6. 13 *การสร้างโมเดล log-linear cubic*



รูปที่ 6. 14 *Residuals vs fitted values*

ตัวอย่างที่ 6.7 ชุดข้อมูล *'swiss'* เป็นข้อมูลการวัดภาวะการเจริญพันธ์ และตัวชี้วัดทางเศรษฐานะทางสังคมจำนวน 47 จังหวัดที่พูดภาษาฝรั่งเศษในประเทศสวิตเซอร์แลนด์ในปี ค.ศ. 1888 การสร้าง *linear model* สำหรับข้อมูลการเจริญพันธ์กับตัวแปรอื่น ๆ มีความเหมาะสมหรือไม่

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



รูปที่ 6. 15 Scatter plot matrix สำหรับข้อมูล swiss

ตัวอย่างคำสั่งดังต่อไปนี้

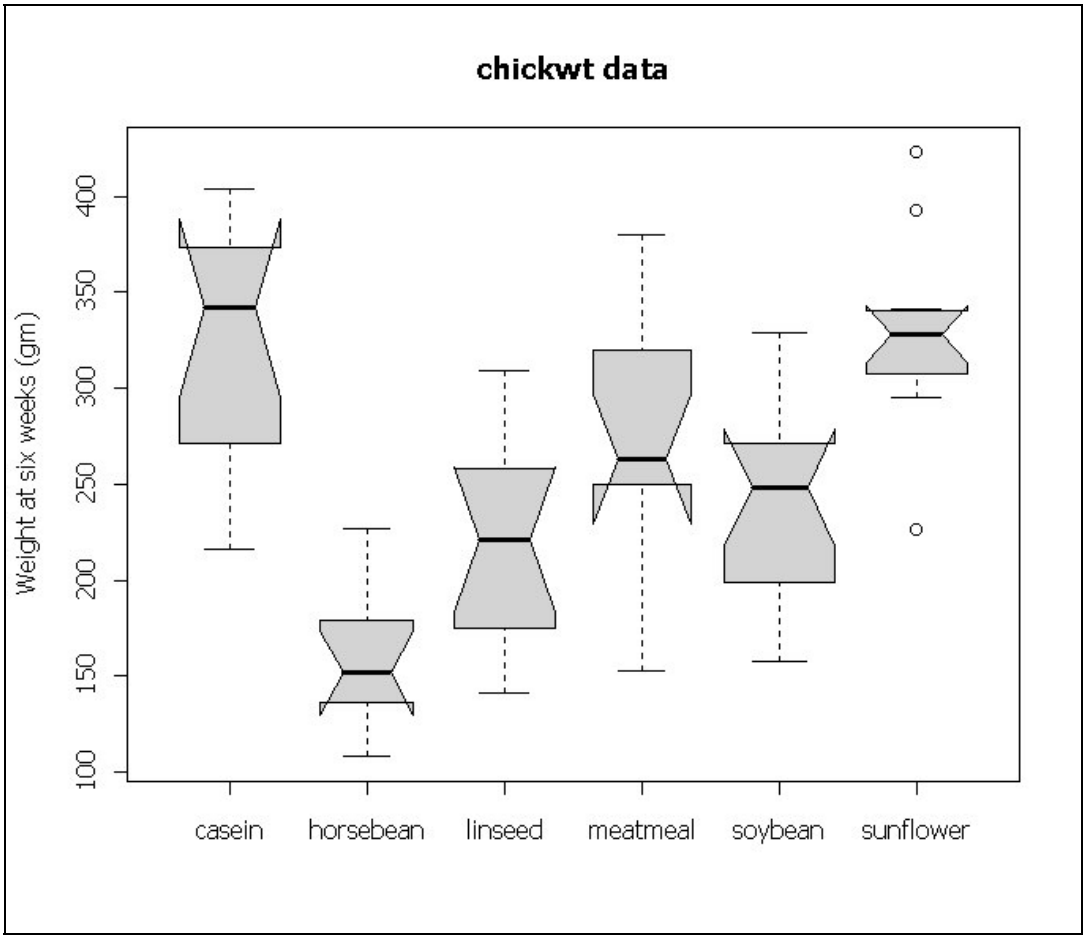
```
> data(swiss)
> pairs(swiss, panel=panel.smooth, col=3+(swiss$Catholic>50))

> 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 ชั่งน้ำหนักไก่หลังจากการให้อาหารนี้เป็นระยะเวลา 6 สัปดาห์ สร้างกราฟด้วยคำสั่ง **boxplot**() และวิเคราะห์ความสัมพันธ์ระหว่างน้ำหนักไก่หลังจากเลี้ยงด้วยอาหารเสริมเป็นระยะเวลา 6 เดือน ทดสอบว่าน้ำหนักไก่มีความแตกต่างกันหรือไม่ในแต่ละกลุ่ม

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

แม้ว่าข้อมูลจะมีการแจกแจงปกติ แต่ค่าความแปรปรวนไม่เท่ากัน ดังนั้นจึงไม่สามารถทดสอบด้วย *ANOVA* การทดสอบแบบ *Kruskal Wallis* สำหรับน้ำหนักไก่แต่ละกลุ่มมีความแตกต่างกันอย่างมีนัยสำคัญทางสถิติ การพิจารณความแตกต่างระหว่างน้ำหนักไก่ที่เลี้ยงด้วยอาหารเสริมในแต่ละชนิด โดยการใช้ *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 เป็นข้อมูลปัจจัยที่มีผลต่ำน้ำหนักแรกคลอดของเด็กทารก ประกอบด้วย ตัวแปร

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

จงสร้างกราฟด้วยคำสั่ง **`pairs()`** เพื่อสำรวจดูความสัมพันธ์ระหว่างตัวแปร
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 ใช้เองใน R นั้นสามารถทำได้ง่าย เพียงแต่เรียนรู้สิ่งที่เกี่ยวข้องกับ 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 การกลับไปแก้จะยุ่งยาก 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 จำไว้ว่ายอมเสียบรรทัดมากขึ้นดีกว่าสับสนว่าปิดวงเล็บครบทุกอันหรือยัง

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 ร่วมนั้น (เขียนไว้ในเครื่องหมายคำพูด) ท้ายบรรทัดกำหนด function หรือจะขึ้นบรรทัดใหม่ก็ได้ จากนั้นเขียน function ใหม่โดยใช้ชื่อ function เดิมนั้น ตามด้วย '.' และชื่อคลาสของ argument โดยเขียนวิธีการทำงานให้เหมาะสมกับคลาสของ argument นั้นๆ และบางครั้ง หากการจัดการกับ argument ในคลาสอื่นนอกเหนือจากนั้นมีการทำงานที่เฉพาะออกไป 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 ได้ด้วยตัวย่อได้อีกเป็น 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 ผลลัพธ์ที่ได้จากการเรียกฟังก์ชั่น `data.name()` จะเป็นดังนี้

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

7.2.2 การนำชื่อตัวแรในกรอบข้อมูลมาใช้ 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 ผลลัพธ์ที่ได้จากการเรียกฟังก์ชั่น `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 ตรวจความยาว (length) ของ argument นั้น ถ้าความยาวเป็น 1 ก็ (คาดได้ว่า) เป็นตัวเลขลำดับ การตรวจสอบความยาวเช่นนี้จะผิดพลาดได้ถ้า data frame นั้นมี variable เพียงตัวเดียว แต่คงเป็นไปไม่ได้ที่จะมี 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 ก็จะให้ผลเป็น TRUE เช่นกัน แต่ถ้าเป็นตัวเลขเดี่ยวๆ จะให้ผลเป็น FALSE ในที่นี้จะไม่แสดงให้ดู ลองทำเองเป็นการทดสอบความรู้ได้

7.2.3 การนำชื่อกรอบข้อมูลกับชื่อตัวแปรมาต่อกันด้วย "$"

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")
}
```

ผลลัพธ์ที่ได้จากการเรียกฟังก์ชั่น `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 `"$"` ก็จะทำให้ชื่อทั้งสองต่อกันทันทีด้วยเครื่องหมาย $

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 ข้อแนะนำต่อไปนี้เป็นสิ่งที่จำเป็นอย่างน้อยที่สุดในการสร้าง package ให้ทำงานได้ หากต้องการให้ package ทำงานหรือมีคุณสมบัติอย่างอื่นๆ เพิ่มเติมมากกว่านี้ ควรอ่านรายละเอียดในคู่มือ Writing R Extensions

ในบทนี้จะกล่าวถึงเฉพาะการสร้างแฟ้มและโฟลเดอร์ต่างๆ เท่านั้น ส่วนการ build package นั้นจะได้กล่าวในบทต่างหากต่อไป เนื่องจากมีขั้นตอนซับซ้อนพอสมควร และยังจะต้องติดตั้งโปรแกรมที่เกี่ยวข้องอีกมากมาย

ในการสร้าง package หนึ่งๆ นั้น เริ่มจากการสร้างโฟลเดอร์ในชื่อของ package ที่ต้องการแล้วสร้างโฟลเดอร์ย่อยต่อไปนี้ (เป็นอย่างน้อย) แฟ้มและโฟลเดอร์เหล่านี้ใช้ได้เหมือนกันในการ build package ในทุก platform ไม่ว่าจะเป็น Windows, Unix, หรือ Macintosh ก็ตาม โฟลเดอร์และแฟ้มเหล่านี้ได้แก่

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

แม้ว่าโฟลเดอร์ย่อย man นี้จะไม่จำเป็น แต่ควรมีไว้ เพื่ออธิบายวิธีการใช้งาน function ต่างๆ เช่น argument ที่สามารถใช้ได้ ลำดับของ argument ข้อจำกัดของ function และอื่นๆ แม้ว่าสิ่งเหล่านี้อาจเขียนเป็นเอกสารคู่มือได้ แต่ผู้ใช้จะสะดวกมากกว่าหากเรียกดูข้อความช่วยเหลือได้ทันทีหากลืมวิธีใช้หรือต้องการดูตัวอย่างวิธีใช้งานจาก R console หรือในสภาพแวดล้อมของการทำงานในขณะนั้น

นอกจากโฟลเดอร์ย่อยทั้งสามแล้ว สิ่งที่จำเป็นมากอีกอย่างหนึ่งคือแฟ้มที่ชื่อ DESCRIPTION แฟ้มนี้ต้องเขียนด้วยอักษรตัวใหญ่ และไม่มีนามสกุลใดๆ ในบางครั้งโปรแกรม

editor บางตัวที่ใช้อาจใส่นามสกุลให้โดยอัตโนมัติ ถ้าพบว่ามีการเติมนามสกุลให้เอง ต้องลบออกเสีย

ตัวอย่างเช่น การสร้าง package ชื่อ mypack ให้สร้างโฟลเดอร์ชื่อ mypack ที่บรรจุโฟลเดอร์ย่อยและแฟ้มดังนี้

โฟลเดอร์หลัก	**แฟ้มในโฟลเดอร์หลัก**	**โฟลเดอร์ย่อย**	**แฟ้มในโฟลเดอร์ย่อย**
mypack	DESCRIPTION		
		R	mypack.R
		data	FILELIST
			examdata.R
			…
		man	myfun.Rd
			…

ขอย้ำอีกครั้งหนึ่งว่าโครงสร้างนี้เป็นโครงสร้างอย่างน้อยที่สุดของ package ที่ควรมี ในคู่มือ Writing R Extensions จะแสดงให้เห็นว่ายังมีแฟ้มและโฟลเดอร์ย่อยที่อาจเพิ่มเติมได้อีกมาก โดยที่แฟ้มหรือโฟลเดอร์เหล่านั้นใช้สำหรับเพิ่มการทำงานที่พิเศษหรือเฉพาะด้าน แต่ผู้ใช้ส่วนใหญ่มักไม่ได้ใช้

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 จะสร้างให้เองในเวลาที่ 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

เป็นโฟลเดอร์ที่บรรจุแฟ้ม function ต่างๆ ที่ต้องการรวมให้เป็น package เดียวกัน มักเป็นโฟลเดอร์ที่สำคัญที่สุดที่ต้องมีอยู่ใน package โดยทั่วไปหากมี function ไม่มากนัก ก็รวมอยู่ในแฟ้มเดียวกันได้ และมักนิยมตั้งชื่อให้ตรงกับชื่อ package นั้น และมีนามสกุลเป็น .R แต่หากมี function จำนวนมากก็อาจแบ่งเป็นหลายแฟ้มก็ได้ ในขณะ 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 นั่นเอง เนื่องจากในแฟ้มขนาดใหญ่นั้น หากพบข้อผิดพลาดขณะ build package จะหาตำแหน่งที่ผิดพลาดได้ยาก เพราะเป็นไปได้มากว่าตำแหน่งบรรทัดที่ R แจ้งว่ามีข้อผิดพลาดจะไม่ใช่ตำแหน่งที่ผิดพลาดจริง คือเกิดความผิดพลาดมาก่อนหน้านั้นหลายบรรทัดแล้ว แต่ยังตรวจสอบไม่พบ เช่นการปิดวงเล็บปีกกา การตรวจสอบจะบอกไม่ได้ว่าควรปิดที่ตรงไหน แต่จะตรวจพบได้เมื่อพยายามหาวงเล็บปิดแล้วไม่พบ เป็นต้น (แพคเกจ ice ปัจจุบันมีถึง 51 แฟ้ม)

ขอให้สังเกตว่าในการตั้งชื่อแฟ้มนั้น เราอาจใส่จุดคั่นชื่อหลายๆ จุดก็ได้ หรืออาจใช้เครื่องหมาย – หรือ _ เพื่อคั่นคำที่มาประกอบกันเป็นชื่อก็ได้

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")
```

กรณีนี้ก็จะต้องมีแฟ้มข้อมูลชื่อ ancdata.txt อยู่ในโฟลเดอร์ data นี้ด้วย และสุดท้ายควรสร้างแฟ้มที่ชื่อ FILELIST เขียนด้วยอักษรตัวใหญ่ทั้งหมดและไม่มีนามสกุล เพื่อบอกว่าแฟ้มข้อมูลทั้งหมดมีแฟ้มอะไรอยู่บ้าง เช่นใน package epican นั้น ข้อความในแฟ้ม 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 และจะนำมากล่าวเป็นหัวข้อต่างหากในบทอื่นต่อไป

จากแฟ้มรูปแบบ Rd นี้ ในขณะ build package แฟ้มช่วยเหลือชนิดอื่นๆ จะถูกสร้างขึ้นมา เช่น ชนิด html, chtml, dvi, และ text ทั้งนี้ขึ้นกับ platform ของคอมพิวเตอร์ที่ใช้ build นั้นเอง ส่วนแฟ้มชนิด pdf ที่อ่านได้ด้วยโปรแกรม Acrobat reader สามารถสร้างได้จากแฟ้ม dvi อีกต่อหนึ่ง

เราสามารถสร้างแฟ้มความช่วยเหลือเป็นภาษาใดก็ได้ (ไทย อังกฤษ หรือภาษาอื่น) แต่หัวข้อจะถูกสร้างเป็นภาษาอังกฤษโดยอัตโนมัติด้วยเงื่อนไขคำสั่งในรูปแบบ Rd นั้นเอง แต่การจัดการกับแฟ้มช่วยเหลือในภาษาอื่นมีรายละเอียดมากพอสมควร จะได้นำมากล่าวเป็นบทต่างหากต่อไป

ดูตัวอย่างแฟ้มความช่วยเหลือสำหรับคำสั่ง `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 ผู้สนใจอาจเข้าไปศึกษารายละเอียดเพิ่มเติมของการ 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 ซึ่งเป็นโปรแกรมที่มีอยู่เป็นปกติบนเครื่อง Linux และ Macintosh อยู่แล้ว แต่ไม่มีอยู่บน Windows ActivePerl เป็นโปรแกรมที่ใช้ในการแปลคำสั่งแฟ้ม Rd เพื่อสร้างแฟ้มข้อความช่วยเหลือนั่นเอง เมื่อได้มาแล้ว ให้ติดตั้งตามขั้นตอนซึ่งจะใช้เวลาในการติดตั้งนานสักเล็กน้อย จากนั้น restart เครื่องคอมพิวเตอร์ โปรแกรม ActivePerl จะถูกเรียกทำงานอยู่เบื้องหลังทุกครั้งเมื่อเปิดเครื่องคอมพิวเตอร์

จากนั้นต้องตั้งค่าตัวแปร environment ที่ชื่อ TMPDIR ให้เป็นโฟลเดอร์ที่มีอยู่จริง เช่น C:\TEMP หรือ C:\WINDOWS\TEMP (หากยังไม่มีก็สร้างขึ้นใหม่) ทั้งนี้จะเป็นโฟลเดอร์ที่ ActivePerl จะใช้สำหรับใส่แฟ้มชั่วคราวที่สร้างขึ้นระหว่างการทำงานต่างๆ เพื่อจะได้ไม่ไปปะปนอยู่กับ โฟลเดอร์ทำงานตามปกติ สามารถทำได้โดยไปที่ Control Panel แล้วเลือก System กดปุ่ม Advanced ที่ด้านบน แล้วกดปุ่ม Environment Variables (ที่ด้านล่างเหนือปุ่ม OK) เมื่อปรากฏหน้าต่างขึ้น ให้ดูในช่อง System variables ด้านล่าง แล้วเลื่อนลงไปจนพบชื่อ TMPDIR หากพบก็ไม่ต้องทำอะไร แต่หากไม่พบให้กดปุ่ม New จะเกิดหน้าต่างเล็กๆ ขึ้น ช่อง Variable name ให้ใส่ TMPDIR และช่องด้านล่างให้ใส่ C:\TEMP หรือ C:\WINDOWS\TEMP ตามที่สร้างไว้หรือมีอยู่แล้วดังกล่าวข้างต้น

นอกจากนี้ยังมีอีกโปรแกรมหนึ่งที่ต้อง download มาคือ htmlhelp.exe ทั้งนี้เพื่อใช้สร้างแฟ้มช่วยเหลือชนิด chtml นั้นเอง โดย download จาก link ที่ให้ไว้ที่ http://www.murdoch-sutherland.com/Rtools/หรือ downloadได้โดยตรงที่ 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 ถ้าไม่สามารถเข้าได้ให้ตรวจสอบ 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 คือติดตั้ง package เข้าไปใน R ทำโดยพิมพ์คำสั่งที่ prompt ดังนี้

```
R CMD INSTALL package_name
```

การทำงานนี้ใช้เมื่อแน่ใจว่าทุกอย่างที่สร้างขึ้นถูกต้องหมดแล้วทั้งสิ้น แต่ในการพัฒนา package จริงๆ นั้น จะต้องตรวจสอบก่อนว่าทุกอย่างที่เขียนใน package และแฟ้มความช่วยเหลือนั้นสามารถทำงานได้ถูกต้องหรือไม่ 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 นั่นเอง (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 ไม่ใช่ที่ไดเรกตอรีหลักของ 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 จะเห็นโฟลเดอร์ package จากนั้นใช้คำสั่งเช่นเดียวกับที่กล่าวมาแล้วข้างต้นบน Linux นั่นเอง

ในการพิมพ์คำสั่งบน Linux หรือ Macintosh ขอให้พิมพ์ตัวพิมพ์เล็กหรือใหญ่ตามที่เขียนไว้ข้างต้นนี้ เพราะบนระบบปฏิบัติการตระกูล Unix นั้น อักษรตัวใหญ่ต่างจากอักษรตัวเล็ก

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

จากนั้นที่ prompt พิมพ์คำสั่ง

```
Rcmd check package_name
```

ซึ่งเป็นการทำงานดังที่ได้กล่าวไว้ข้างต้นเช่นเดียวกับบนระบบ Linux นั่นเอง แต่สังเกตว่า R CMD คราวนี้จะพิมพ์ติดกัน และสามารถพิมพ์ตัวเล็กหรือตัวใหญ่ก็ได้

ในขั้นตอนการ build ด้วย check นั้น จะเกิดโฟลเดอร์ชื่อ *package_name*.Rcheck เช่นเดียวกับที่กล่าวข้างต้น โดยโฟลเดอร์นี้จะอยู่ที่ ../R/rw20xx/bin นั่นเอง แต่จะพบว่ามีการรายงานความผิดพลาดเกิดขึ้น ในขณะที่คำสั่งได้สร้างโฟลเดอร์ชื่อ chm ในโฟลเดอร์ package อันเก่าที่เป็นต้นฉบับ ในโฟลเดอร์นี้จะเห็นแฟ้มความช่วยเหลือที่มีนามสกุล html และจะมีหนึ่งแฟ้มที่มีชื่อ *package_name*.hhp

เมื่อถึงขั้นตอนนี้ ให้เปิดโปรแกรม HTML Help Workshop จาก desktop alias หรือจาก Start Menu ก็ได้ แล้วเลือกที่ File → Open แล้วเปิดแฟ้มที่ชื่อ *package_name*.hhp นั้น จากนั้นกลับไปที่ File → Compile… เพื่อสร้างแฟ้มความช่วยเหลือที่มีชื่อเป็น *package_name*.chm ไว้ในโฟลเดอร์ chm นี้ เมื่อเสร็จแล้วปิดโปรแกรม HTML Help Workshop ได้

ทีนี้กลับมา build package อีกครั้งด้วยคำสั่ง

```
Rcmd check package_name
```

ถ้าคำสั่งและแฟ้มความช่วยเหลือทั้งหมดถูกต้อง คราวนี้คำสั่งจะ build ทุกอย่างได้โดยไม่มีข้อผิดพลาด และจะมีโฟลเดอร์ย่อยชื่อ chtml เพิ่มขึ้นมาจากการ build บน Linux หรือ Macintosh

ในระหว่างการ build หากมีข้อผิดพลาดอย่างใดเกิดขึ้น ขอให้อ่านข้อความแจ้งความผิดพลาดให้ละเอียด เพื่อกลับไปแก้ไขข้อผิดพลาดนั้น แล้วกลับมา build ใหม่อีกครั้ง แต่อย่าลืมว่าคราวนี้มีโฟลเดอร์ package ถึงสามโฟลเดอร์ คือโฟลเดอร์ดั้งเดิมที่สร้างขึ้นในการพัฒนาคำสั่ง โฟลเดอร์ package ที่คัดลอกมาไว้ที่ /bin และโฟลเดอร์ package ใน *package_name*.Rcheck ไม่ต้องไปสนใจโฟลเดอร์ package ใน .Rcheck แต่อย่างใด แต่ทุกครั้งที่แก้ไขต้นฉบับคำสั่งหรือแฟ้มความช่วยเหลือ ควรแก้ไขใน /bin และเมื่อทุกอย่างถูกต้องคือ build ผ่านหมดแล้ว ให้สำเนา

โฟลเดอร์นี้ทั้งหมดกลับไปไว้ที่เดิมที่พัฒนา package นั้นด้วย มิฉะนั้นจะสับสนในภายหลังได้ เพราะอาจมีบางคำสั่งใน package ต้นฉบับกับที่ build แล้วแตกต่างกันได้ และนอกจากนี้ยังแนะนำให้สำเนาโฟลเดอร์ *package_name*.Rcheck เก็บไว้ทั้งโฟลเดอร์ อย่าสำเนาเก็บแต่เฉพาะโฟลเดอร์ package ภายในมาเก็บไว้ เพราะอย่างที่บอกแล้วว่าโฟลเดอร์ package ที่สร้างขึ้นใน .Rcheck นั้นไม่เหมือนกับ package ต้นฉบับ อาจทำให้สับสนได้ว่า package ที่ build แล้วนี้เป็นต้นฉบับได้ แล้วไปลบต้นฉบับจริงทิ้งเสีย การฟื้นต้นฉบับทั้งหมดกลับมานั้นทำได้ยากมาก

อีกประการหนึ่ง หากมีการแก้ไขแฟ้มความช่วยเหลือชนิด .Rd ต้องลบโฟลเดอร์ chm ในโฟลเดอร์ package ที่เป็นต้นฉบับใน /bin ด้วยเสมอ ที่จริงแฟ้มที่สำคัญที่สุดในนั้นคือ *package_name*.chm ซึ่งหากในระหว่างการ build นั้น โปรแกรม Rcmd พบแฟ้มนี้อยู่ในโฟลเดอร์ chm คำสั่งจะข้ามการแก้ไขทั้งแฟ้มชนิด .htm และ .chm ไปเลย ซึ่งจะทำให้ได้แฟ้มความช่วยเหลือเก่าที่ยังไม่ได้แก้ไขแต่อย่างใด ดังนั้น หากมีการแก้ไขแฟ้มความช่วยเหลือชนิด .Rd ให้ลบโฟลเดอร์ chm ทิ้งเสียด้วย และจะต้องเรียกคำสั่งในการ build สองครั้ง และต้องเรียกโปรแกรม HTML Help Workshop ใหม่ทุกครั้ง แม้ว่าจะเป็นการเสียเวลา แต่ถ้าหากไม่ทำ แฟ้มความช่วยเหลือที่ได้จะไม่ถูกต้อง

เมื่อการ build เรียบร้อยแล้ว ให้เข้าไปในโฟลเดอร์ package_name.Rcheck แล้ว zip โฟลเดอร์ package ในนั้นด้วยโปรแกรม winzip หรือ 7-Zip ก็ได้ แล้วแฟ้ม package_name.zip ที่ได้นี้จะเป็นแฟ้มที่สามารถแจกจ่ายให้ผู้ใช้งาน R บนระบบ Windows ได้นั่นเอง (แต่ผู้ใช้บนระบบ Linux และ Macintosh จะใช้งานแฟ้มนี้ไม่ได้)

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

```
R CMD build package_name
```

เป็นการสร้างแฟ้มชื่อ package_name.tar.gz ขึ้นนั่นเอง บน Windows นั้นก็สามารถสร้างแฟ้มนี้ได้โดยออกคำสั่งต่อไปนี้ที่ DOS prompt

```
Rcmd build package_name
```

ในการแจกจ่ายแฟ้ม package สำหรับระบบ Linux และ Macintosh นั้น จะแจกจ่ายเป็นแฟ้ม *package_name*.tar.gz เมื่อผู้ใช้ต้องการจะติดตั้ง ให้ไปที่ terminal แล้วเรียกคำสั่ง R CMD INSTALL เพื่อติดตั้ง

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

7.5 การทำแฟ้มความช่วยเหลือเป็นภาษาไทย

ในการทำแฟ้มความช่วยเหลือนั้น ดังที่ได้กล่าวไว้ในเรื่องการ build package แล้วว่าจะต้องเขียนในรูปแบบ R document (Rd) การที่จะทำแฟ้มความช่วยเหลือเป็นภาษาไทยก็เช่นเดียวกัน ต้องเริ่มต้นด้วยการสร้างแฟ้มความช่วยเหลือในรูปแบบ Rd นั่นเอง ซึ่งมี tag คำสั่งต่างๆ ดังได้อธิบายไว้ในเรื่อง “การเขียนแฟ้มความช่วยเหลือในรูปแบบ R document” แล้ว

ในการเขียนแฟ้มความช่วยเหลือในรูปแบบ Rd ให้เป็นภาษาไทยนั้นก็เขียนเช่นเดียวกับที่กล่าวไว้แล้วในบทที่กล่าวถึงข้างต้น โดยยังคง tag คำสั่งไว้เช่นเดียวกับที่ได้อธิบายไว้นั้น 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 นั่นเอง 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 ให้ได้แฟ้มความช่วยเหลือภาษาไทยในรูปแบบ html และ chtml ได้แสดงเป็นภาษาไทยได้อย่างถูกต้อง ซึ่งหากทำไม่ถูกขั้นตอนแล้วอาจทำให้การแสดงภาษาไทยไม่ถูกต้องได้

ดังได้กล่าวในบท “การสร้าง package ใน R” แล้วนั้น โฟลเดอร์ที่บรรจุแฟ้มความช่วยเหลือในรูปแบบ Rd คือโฟลเดอร์ man นั่นเอง หากจะมีบางกรณีที่ต้องการเรียกความช่วยเหลือในรูปแบบ text ด้วย ให้เขียนแฟ้มความช่วยเหลือเป็นสองโฟลเดอร์ โดยโฟลเดอร์หนึ่งเขียนเป็น

ภาษาอังกฤษ และอีกโฟลเดอร์หนึ่งเขียนเป็นภาษาไทย 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 สามารถแก้ไขแฟ้ม Rprofile ในโฟลเดอร์ ../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 ก็สร้างแฟ้มความช่วยเหลือในรูปแบบ Rd เป็นภาษาไทยอย่างเดียวได้เลย แล้ว build package ตามที่อธิบายไว้ในบท “การ build package ใน R” ได้เลย

บน Linux นั้น เนื่องจากการ build เป็นภาระของผู้ใช้เอง ดังนั้นอาจต้องเตรียมแฟ้ม .tar.gz ไว้ทั้งที่มีแฟ้มความช่วยเหลือภาษาไทยและอังกฤษ ให้ผู้ใช้เลือกที่จะ build แบบใดแบบหนึ่งต่อไปนี้ด้วยตัวเอง

หากไม่ต้องการใช้ความช่วยเหลือแบบ text ก็สามารถสร้างแฟ้มความช่วยเหลือรูปแบบ 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 โดยใช้แฟ้ม .tar.gz ที่มีแฟ้มความช่วยเหลือเป็นภาษาไทยก่อน โดยเมื่อขยายแฟ้มออกมาแล้ว ให้ build ด้วยคำสั่ง

```
R CMD INSTALL package_name
```

จากนั้นจึงขยายแฟ้ม .tar.gz ที่มีแฟ้มความช่วยเหลือภาษาอังกฤษออกมา แล้ว build ด้วยคำสั่ง

```
R CMD check package_name
```

อีกครั้ง คราวนี้จะได้โฟลเดอร์ *package_name*.Rcheck ขึ้นที่ root (ในการ build package จะต้อง login เป็น root เสมอ) จากนั้นเข้าไปในโฟลเดอร์ *package_name*.Rcheck/*package_name* จะเห็นโฟลเดอร์ชื่อ help อยู่ ให้คัดลอกโฟลเดอร์นี้ไปไว้ในโฟลเดอร์ *R_HOME*/library/*package_name*

ทับโฟลเดอร์ help ที่มีอยู่เดิม โฟลเดอร์ *R_HOME* อาจแตกต่างกันตามแต่ระบบ Linux ที่ใช้ ทางที่ดีให้ใช้วิธีค้นหาชื่อของ package ในโฟลเดอร์ / (ไม่ใช่ /root) จะเห็นว่าการทำค่อนข้างยุ่งยากพอสมควร

ส่วนบนระบบ Windows นั้น ผู้พัฒนา package จะต้องแจกจ่ายแฟ้ม package ในรูปแบบ zip จึงต้อง build package เสียก่อน นั่นคือผู้พัฒนาสามารถทำแฟ้ม *package_name*.zip ที่มีโฟลเดอร์ความช่วยเหลือ html และ chtml เป็นภาษาไทย ขณะที่แฟ้มในโฟลเดอร์ help เป็นภาษาอังกฤษได้เลย

ขั้นตอนการ build ให้ build package ด้วยแฟ้มความช่วยเหลือภาษาอังกฤษ โดยใช้คำสั่ง

```
Rcmd check package_name
```

ดังที่ได้กล่าวแล้วในบท “การ build package ใน R” ซึ่งจะต้องออกคำสั่งนี้สองครั้งดังได้กล่าวในบทความ เมื่อเสร็จแล้วจะได้โฟลเดอร์ package_name.Rcheck ซึ่งภายในมีโฟลเดอร์ *package_name* อยู่ ให้ zip โฟลเดอร์นี้เป็น *package_name*.zip ที่มีแฟ้มความช่วยเหลือเป็นภาษาอังกฤษล้วน พร้อมแจกจ่ายให้กับชาวต่างประเทศได้ จากนั้นเปลี่ยนชื่อโฟลเดอร์ *package_name*.Rcheck นี้ให้เป็น *package_name*.Rcheck.en เพื่อให้รู้ว่าเป็นโฟลเดอร์ที่มีแฟ้มความช่วยเหลือเป็นภาษาอังกฤษ

จากนั้น build package ที่มีแฟ้มความช่วยเหลือเป็นภาษาไทย 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 แฟ้มความช่วยเหลือที่เป็น chtml นั้นให้ใช้ภาษาไทยชุดอักษร tis-620 แก้เช่นนี้กับทุกแฟ้มในโฟลเดอร์ chm นี้

จากนั้นเรียกโปรแกรม HTML Help Workshop เพื่อ compile สร้างแฟ้มความช่วยเหลือ chtml แล้ว build package ใหม่อีกครั้ง คราวนี้จะได้โฟลเดอร์ *package_name*.Rcheck ที่มีแฟ้มความช่วยเหลือเป็นภาษาไทย แต่สามบรรทัดแรกในแฟ้มความช่วยเหลือรูปแบบ html จะกลับไปเป็นเหมือนเดิมอีก และแฟ้มความช่วยเหลือในรูปแบบ text ที่ยังเป็นตัวอักษรไทยที่อ่านใน R console ไม่ได้อยู่

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

เมื่อแก้เสร็จแล้ว ให้กลับไปที่โฟลเดอร์ *package_name*.Rcheck.en ที่ได้สร้างไว้ก่อนแล้ว คัดลอกโฟลเดอร์ help ที่เป็นความช่วยเหลือภาษาอังกฤษนี้มาทับโฟลเดอร์ help ในโฟลเดอร์ *package_name* ที่อยู่ภายในโฟลเดอร์ *package_name*.Rcheck ที่เป็นภาษาไทยเสีย เพื่อให้แฟ้มความ

ช่วยเหลือในรูปแบบ text อ่านได้เป็นภาษาอังกฤษ (อาจลบโฟลเดอร์นี้ไปเลยก็ได้ แต่ต้องแจ้งให้ผู้ใช้ทราบว่าในการเรียกความช่วยเหลือสำหรับ package นี้ ต้องเรียกเป็น chtml หรือ html เท่านั้น)

จากนั้น zip โฟลเดอร์ package_name นั้นดังที่ได้อธิบายไว้ข้างต้น จะได้แฟ้ม *package_name*.zip ที่มีแฟ้มความช่วยเหลือเป็นภาษาไทยพร้อมแจกจ่ายแก่ผู้ใช้ได้