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

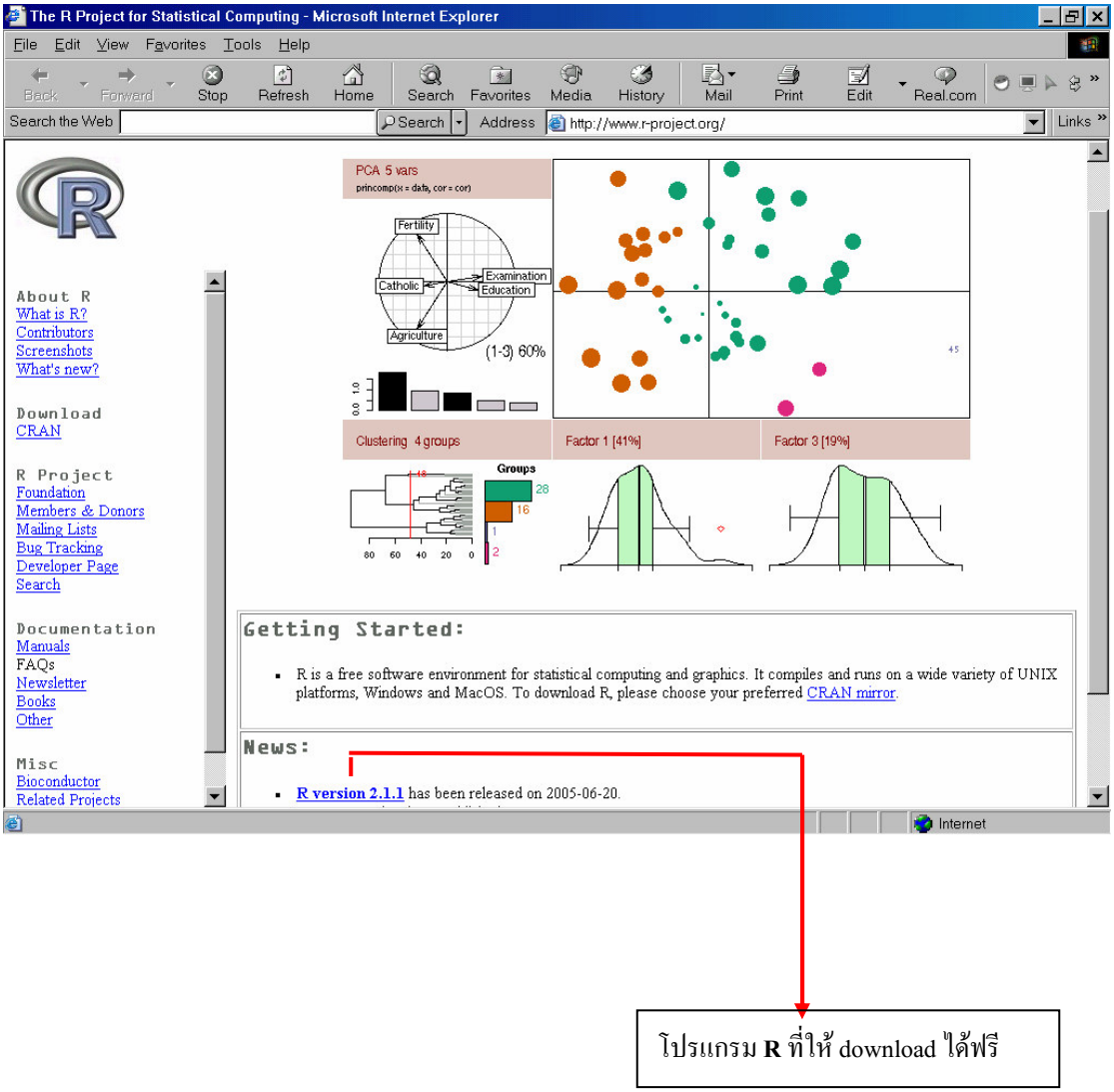
1.2 เหตุผลในการใช้โปรแกรม R

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

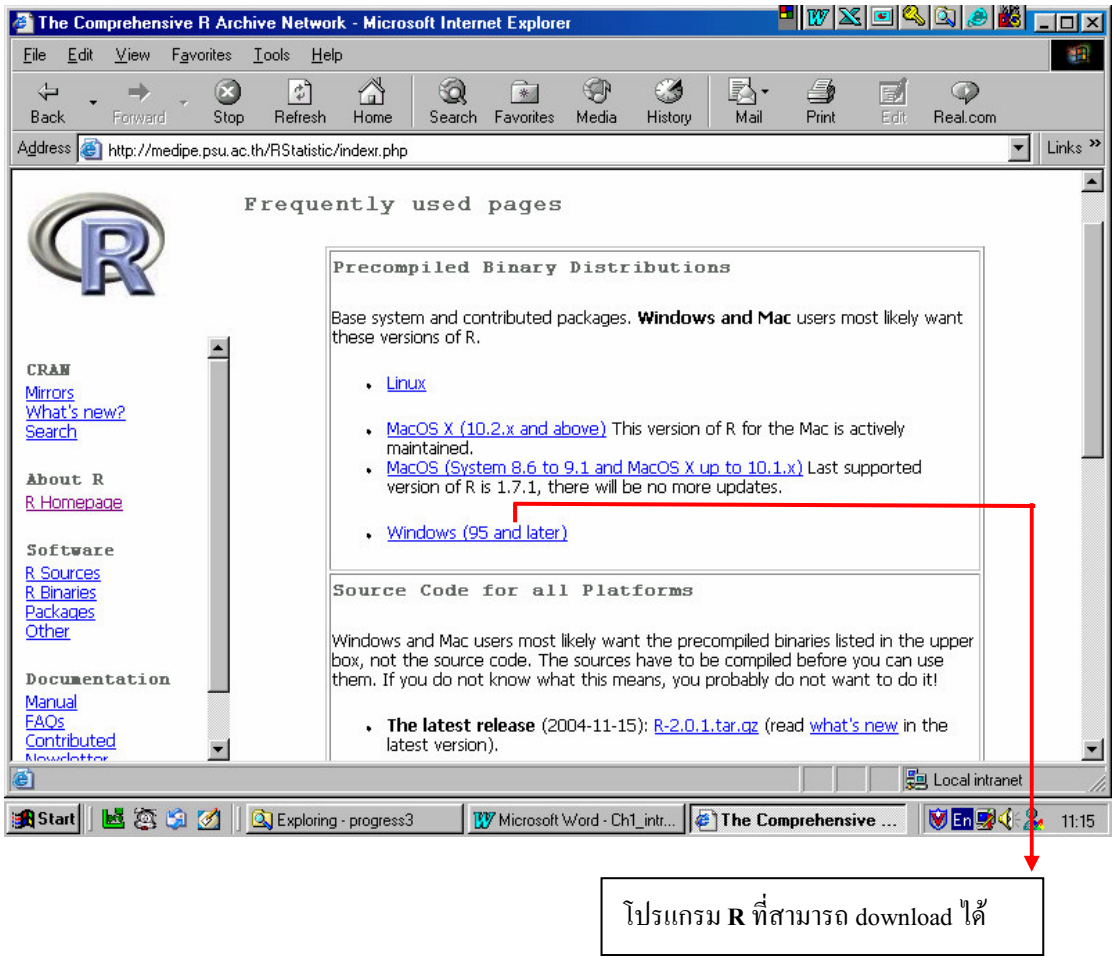
โปรแกรม **R** สามารถเข้าไป download ได้ที่ web site ของ **R** โดยตรงคือ http:// www.r-project.org ดังรูปที่ 1.1 ปัจจุบันโปรแกรม **R** ที่ download จะอยู่ใน version 2.4.0



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



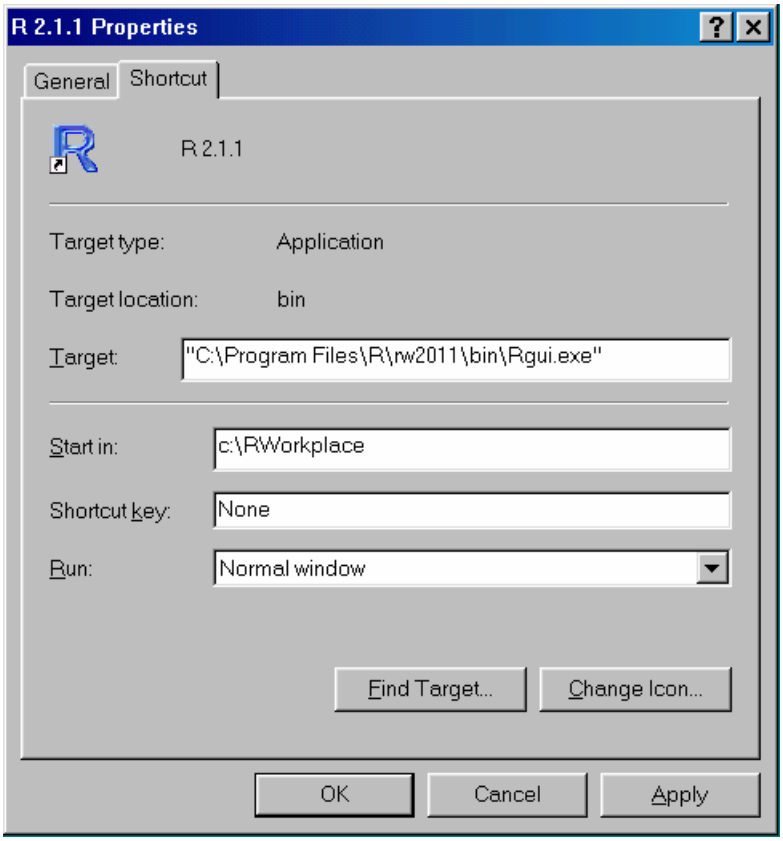
รูปที่ 1.2 Web page ของหน่วยระบาดวิทยาที่เก็บโปรแกรม R สำหรับให้ download

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"

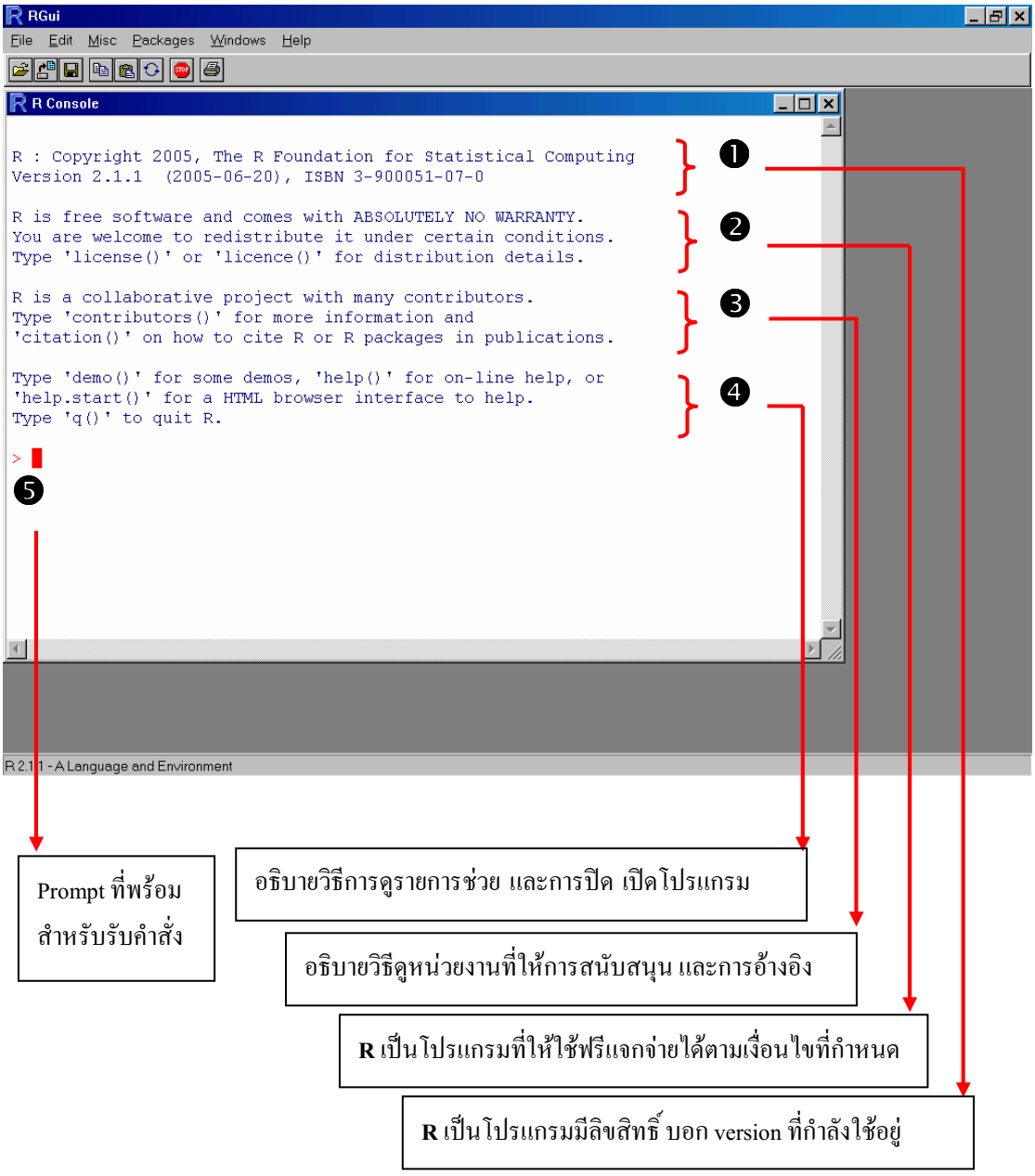


รูปที่ 1.3 การตั้งค่าใน Start in

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

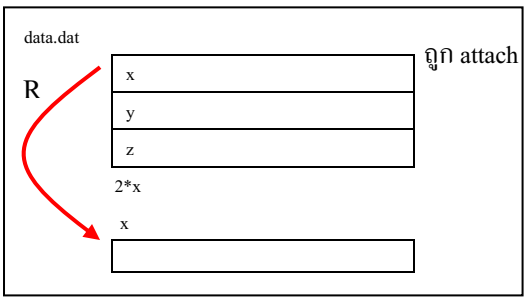
```
> data.dat$x
```

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

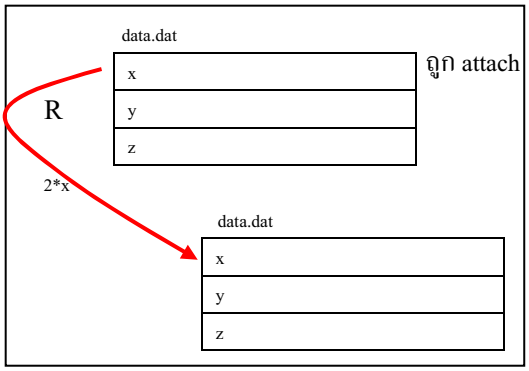
```
> x <- 2*x
```

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

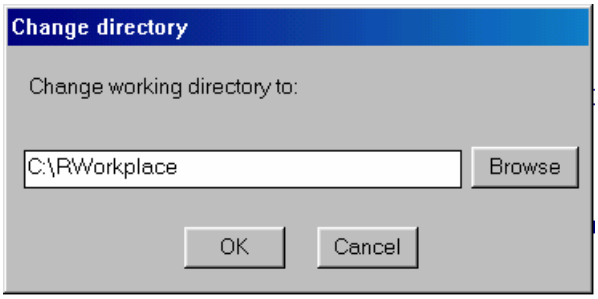
1.8.13 การกำหนดโฟลเดอร์สำหรับการทำงาน

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

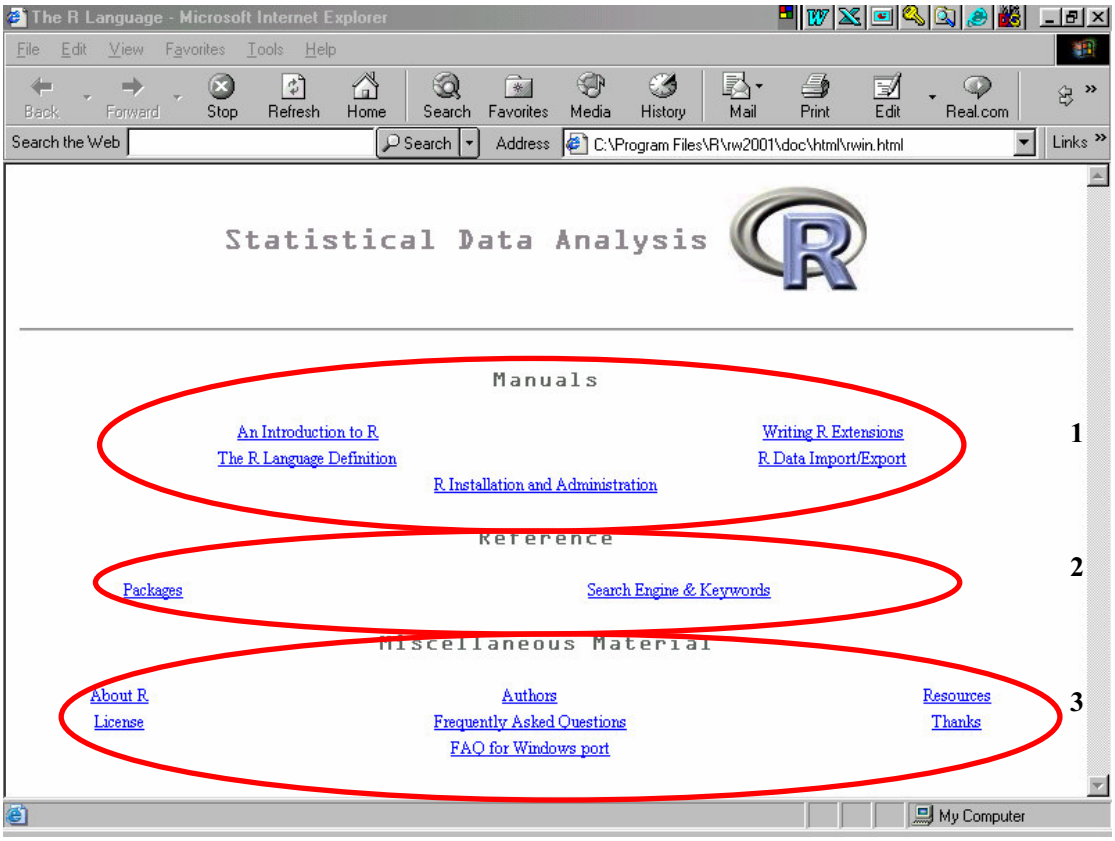
```
> help.start()          ❶
> help()                ❷ หรือ
> ?help                 ❸
```

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

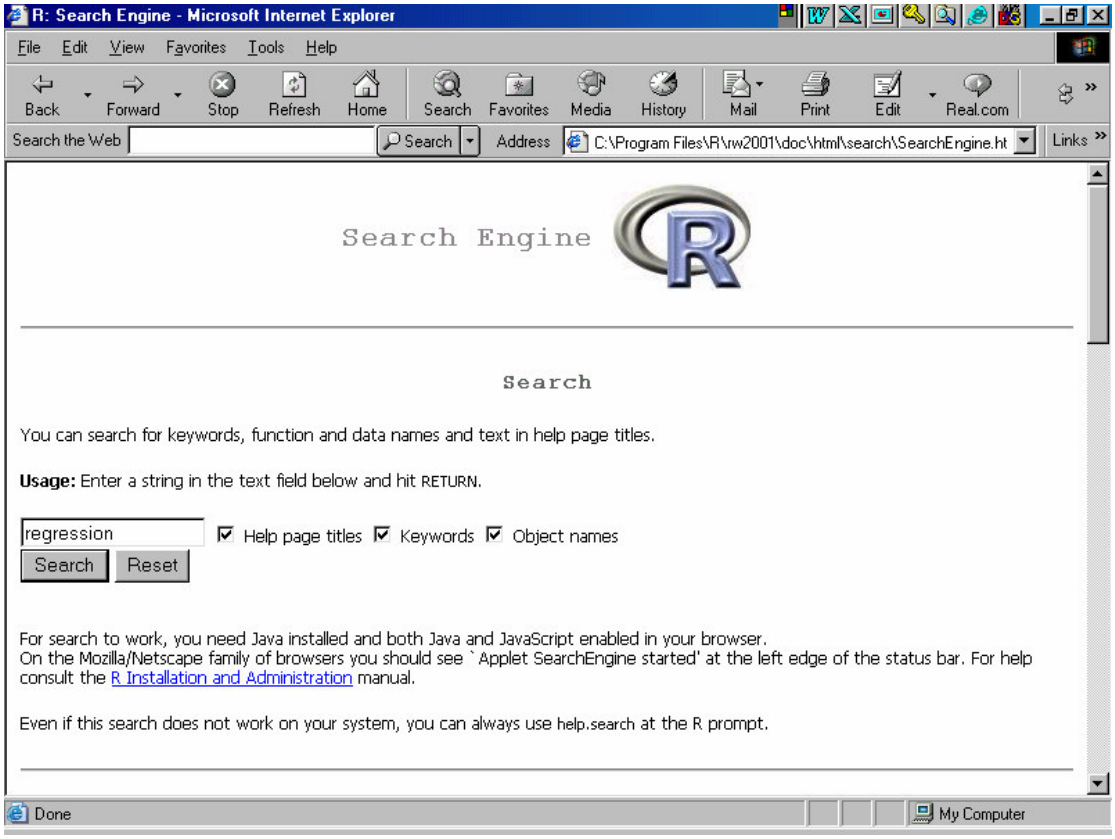
คำสั่งทั้งสองคำสั่งจะแสดงหน้าจอ online help ของการใช้คำสั่ง mean เหมือนกัน ฉะนั้นหากต้องการเรียกความช่วยเหลือที่สั้นกะทัดรัดก็ควรจะใช้คำสั่งที่ ❷

1.9.2 กรณีไม่ทราบคำสั่งที่มีอยู่ใน R

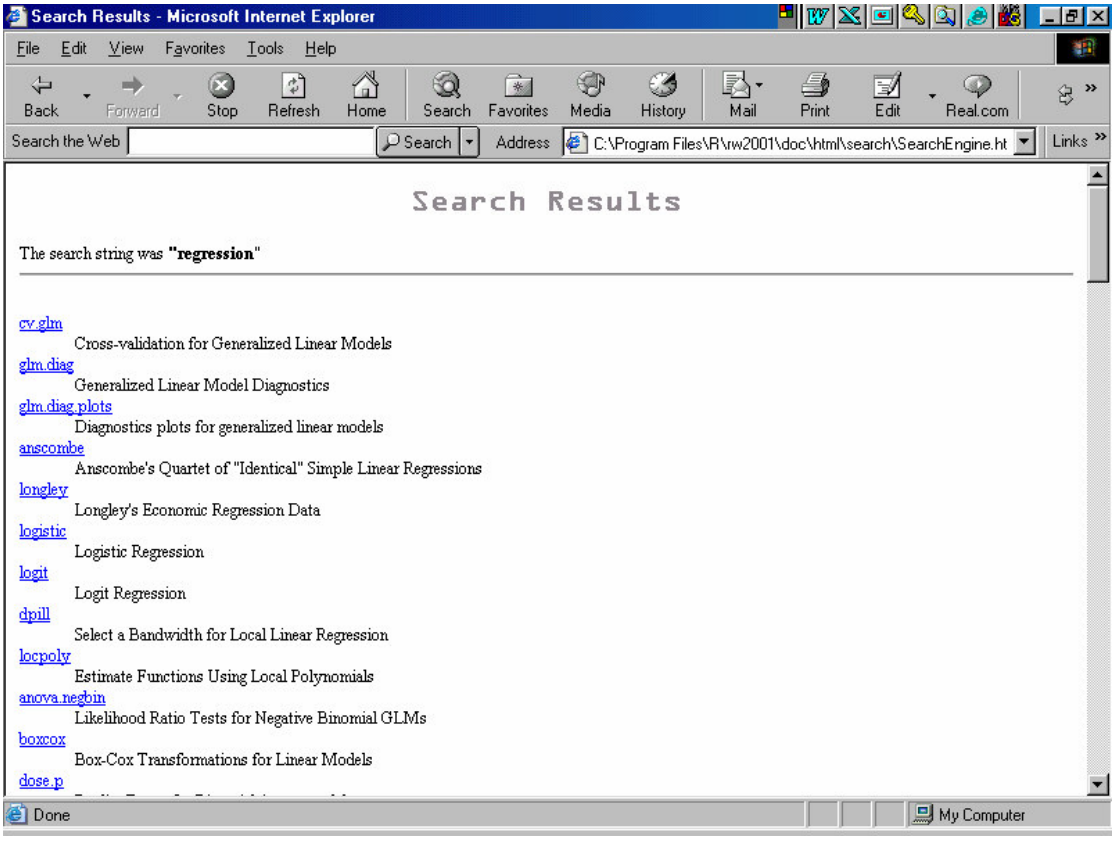
สำหรับกรณีนี้ หากทราบคำบางคำที่เกี่ยวข้องกับงานที่ต้องการใช้ ซึ่งอาจเป็นคำสำคัญ (keyword) เช่น คำว่า test การเรียกดูความช่วยเหลือจะต้องพิมพ์คำนั้นไว้ในเครื่องหมาย " " ดังเช่น

```
> 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

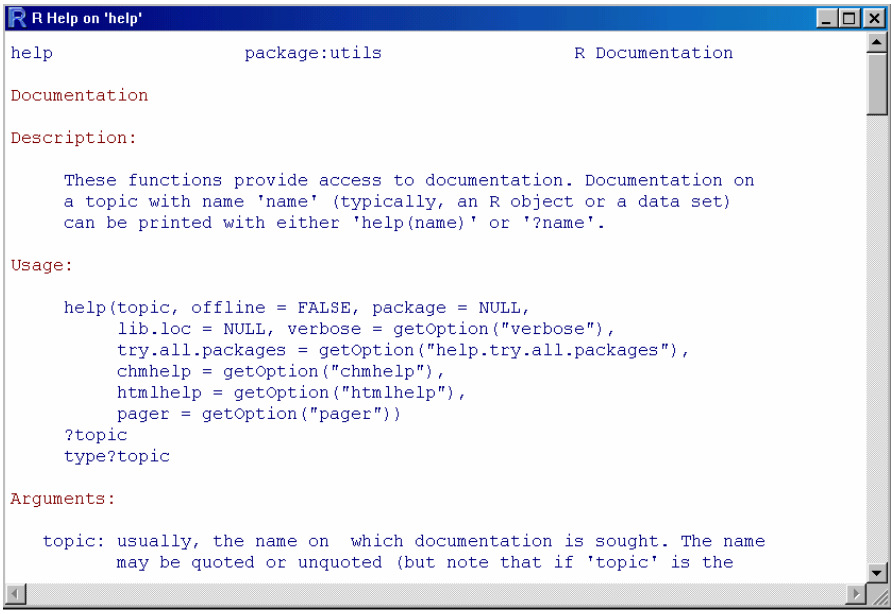
R แสดงหน้าจอ help เป็น 3 รูปแบบด้วยกัน คือ

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)
```

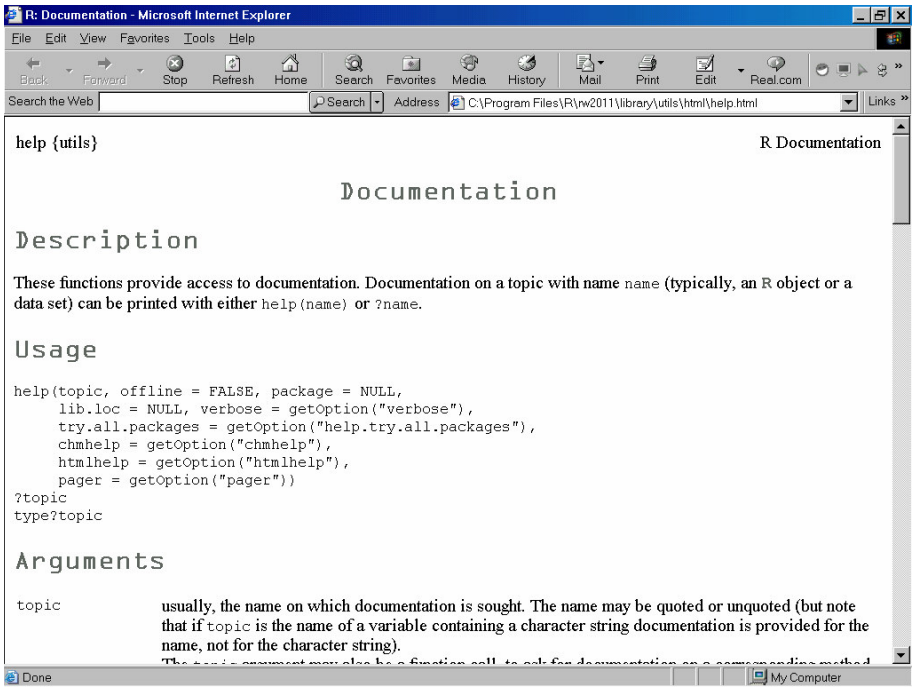
จากคำสั่งนี้ **R** จะแสดงหน้าจอ help ดังรูปที่ **1.9**



รูปที่ 1.9 หน้าจอ help ภายใต้ RGUI

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

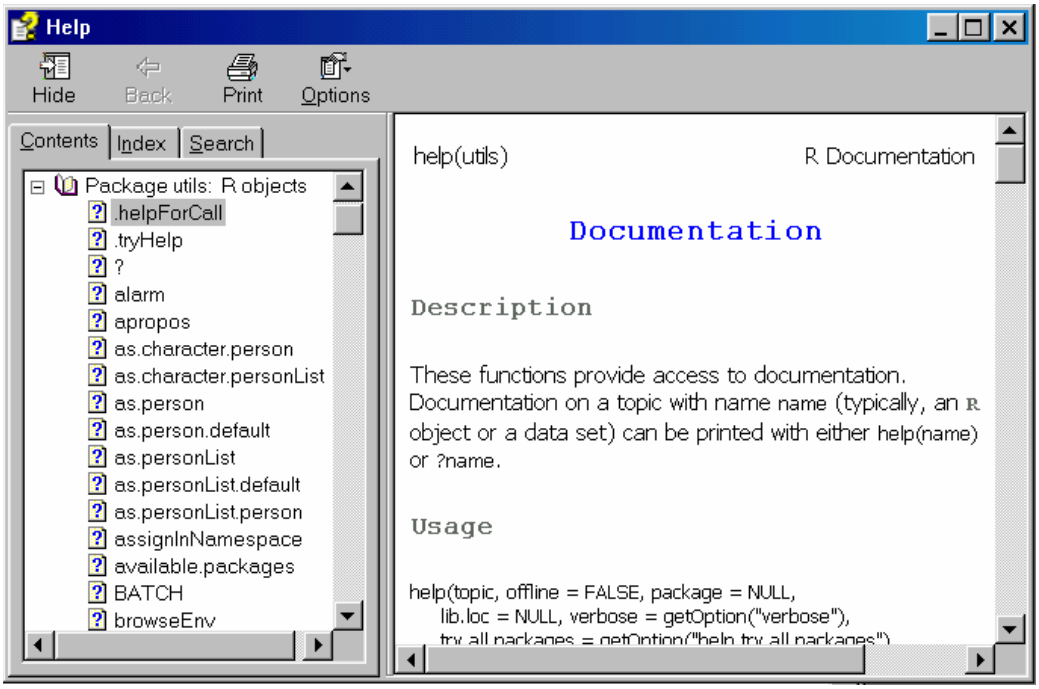
คำสั่งข้างต้นจะแสดงหน้าจอ help ด้วย web browser ดังรูปที่ 1.10



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 <-`

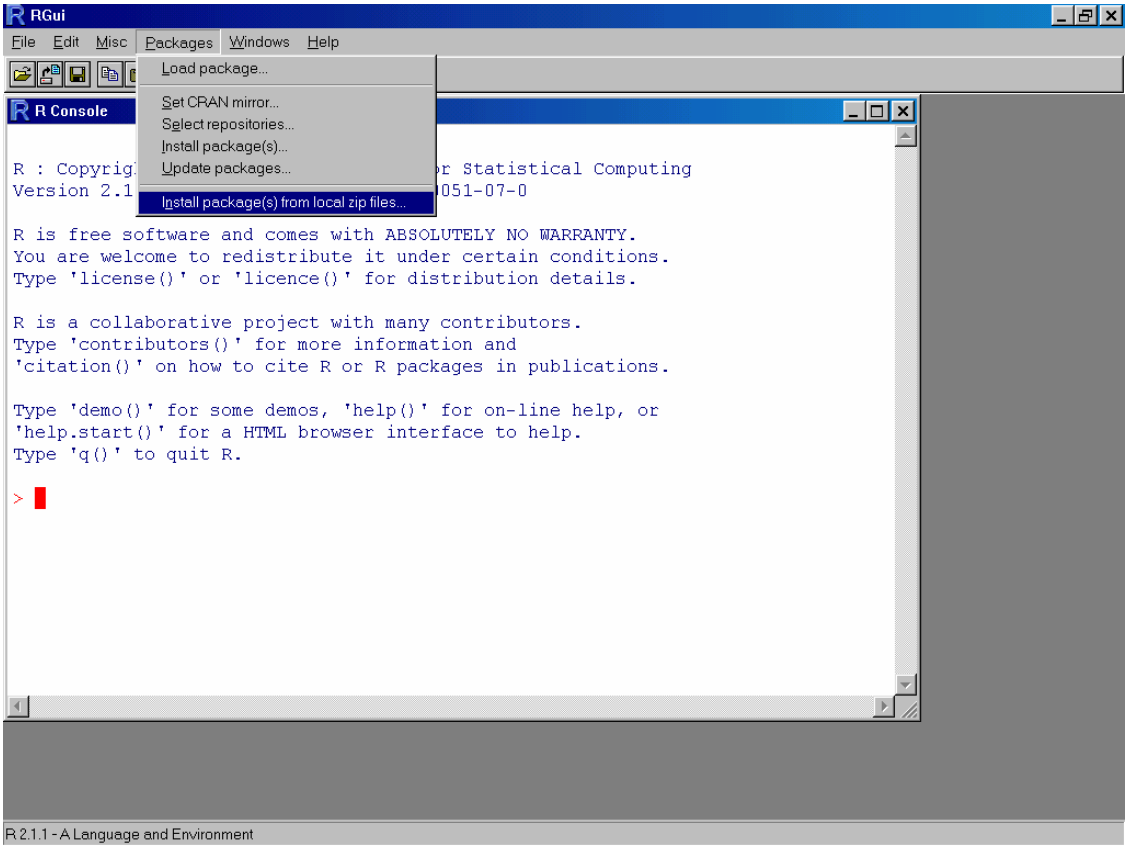
หากต้องการเปลี่ยนแปลงใดๆ กับข้อมูลเดิม ให้สร้างเป็นตัวแปรใหม่ อย่าตั้งชื่อซ้ำกับตัวแปรเดิม ทั้งนี้จะได้ประโยชน์คือ จะรักษาข้อมูลเดิมไว้ได้ ซึ่งหากจะยกเลิกการเปลี่ยนแปลงนั้นๆ ก็เพียงลบตัวแปรที่สร้างขึ้นจากตัวแปรเดิมนั้น แล้วสร้างใหม่ด้วยเงื่อนไขใหม่ เช่นหากต้องการแบ่งช่วงค่าตัวแปร ก็สร้างตัวแปรใหม่ขึ้นมาเพื่อกำหนดการแบ่งช่วง 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()`

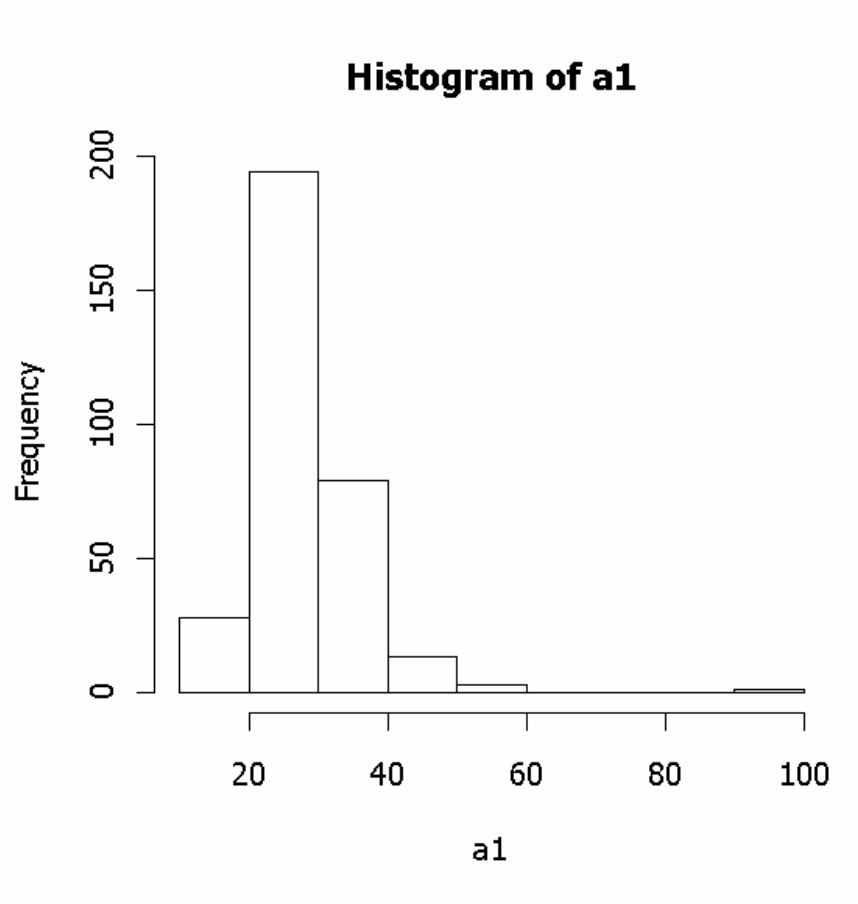
เป็นคำสั่งสำหรับการสร้างฮีสโตแกรมด้วยการ plot ความถี่ของข้อมูลภายในแต่ละอัตรภาคชั้นที่ถูกแบ่งด้วยค่า breaks ความสูงของแท่งจะเป็นสัดส่วนกับจำนวนข้อมูลที่ตกอยู่ในอันตรภาคชั้นนั้น ๆ เป็นคำสั่งที่อยู่ใน library Base ซึ่งมีรูปแบบการออกคำสั่งดังนี้

```
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

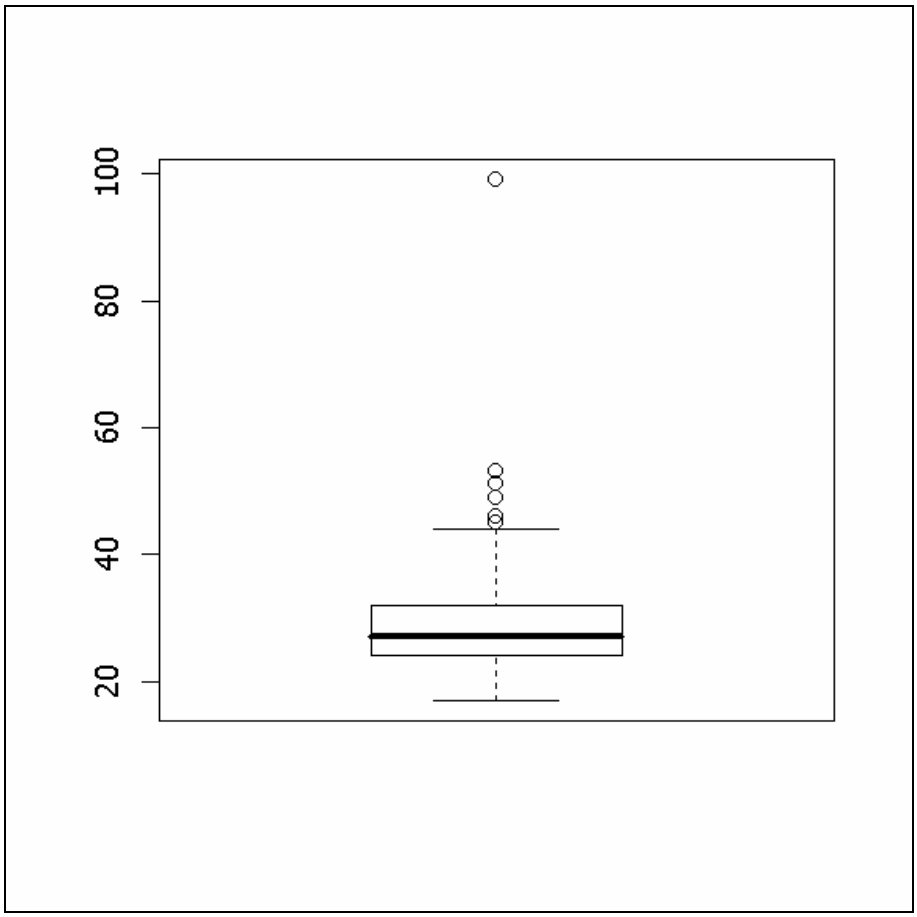
```
boxplot(x, ..., range = 1.5, width = NULL, varwidth = FALSE,
      notch = FALSE, outline = TRUE, names, plot = TRUE,
      border = par("fg"), col = NULL, log = "",
      pars = list(boxwex = 0.8, staplewex = 0.5, outwex = 0.5),
```

```
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()`

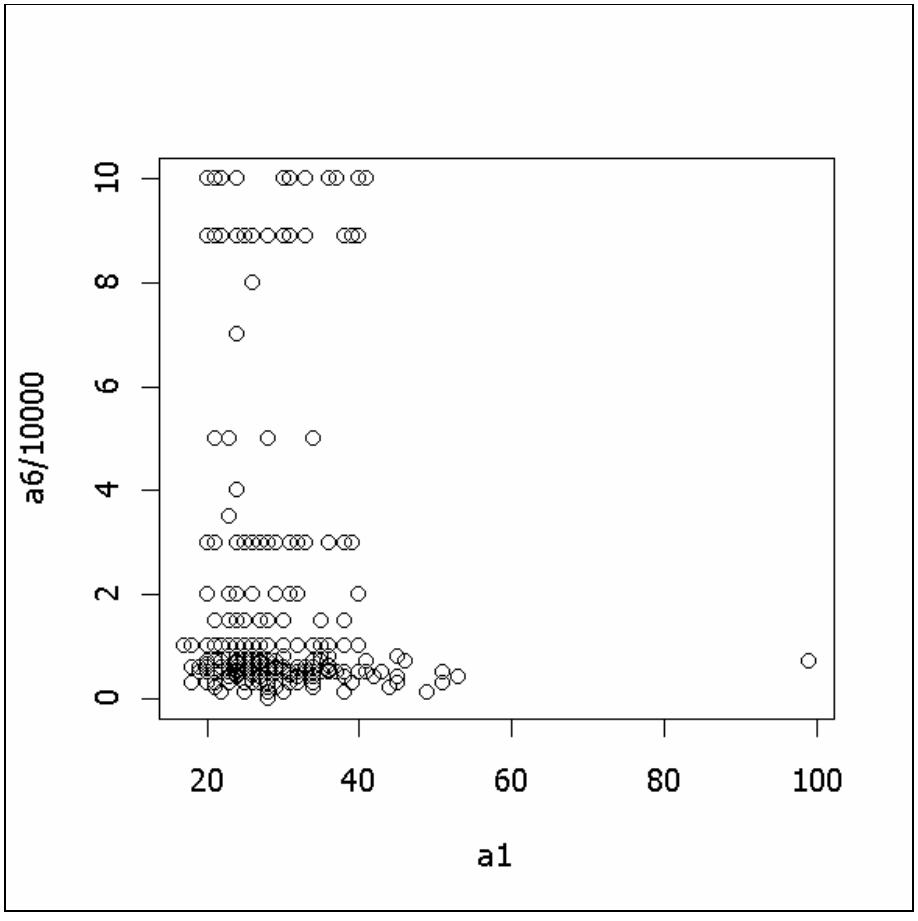
เป็นคำสั่งสำหรับการสร้าง scatter plot เป็นคำสั่งที่อยู่ใน library base ซึ่งมีรูปแบบการออกคำสั่งดังนี้

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

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

ตัวอย่างการใช้คำสั่งดังนี้

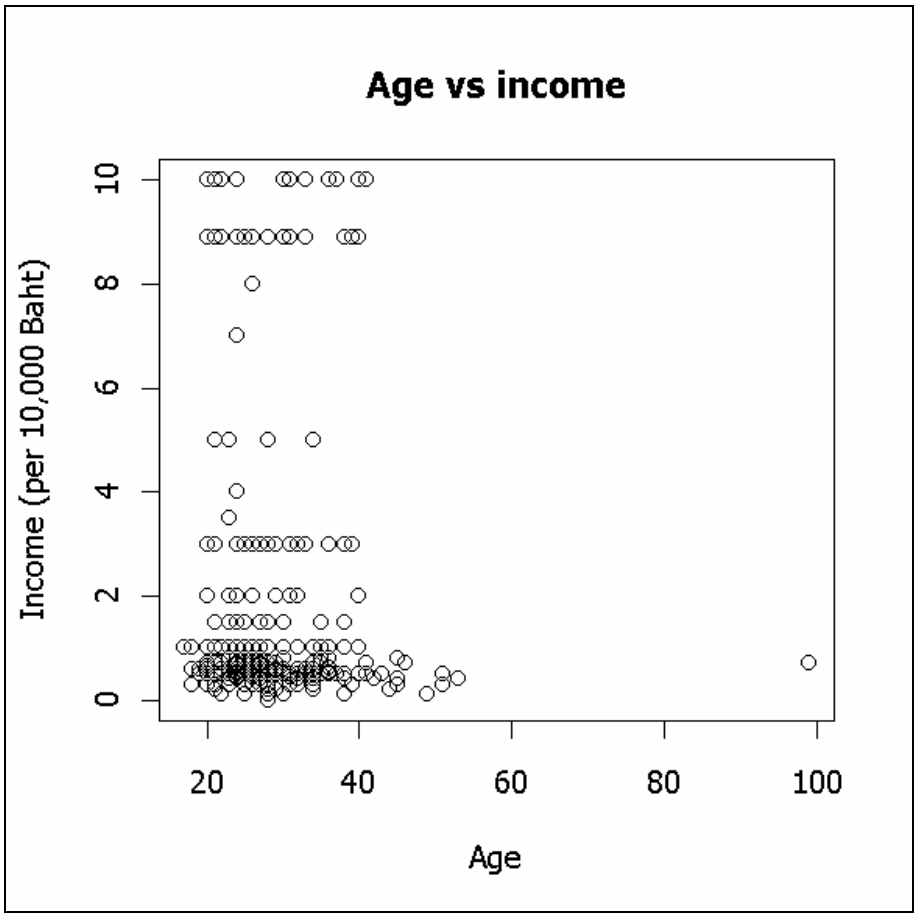
```
> plot(a1, a6)
```



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)
```

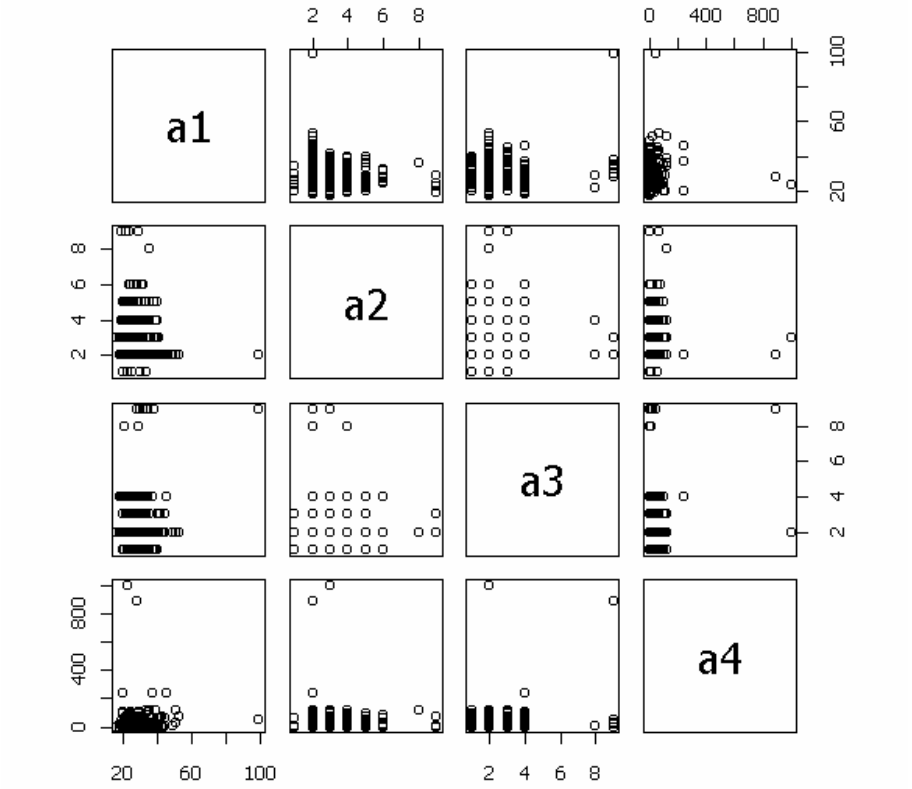
หรือ

```
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()*

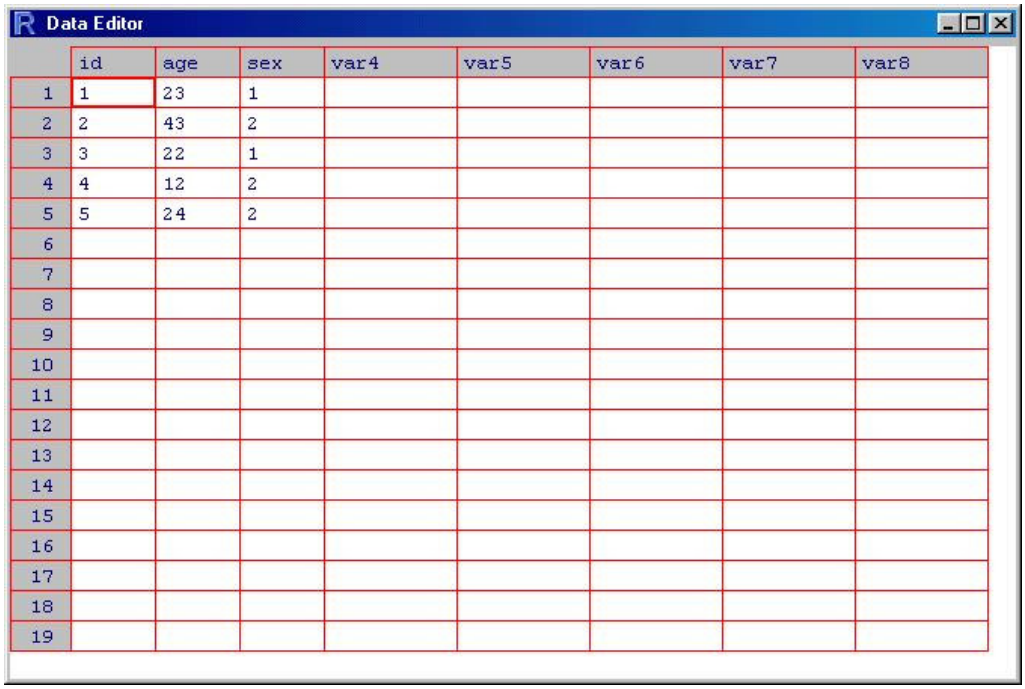
แถบบนสุดของ Data Editor จะถูกกำหนดให้เป็นขอบเขตสำหรับการกำหนดชื่อตัวแปร เมื่อใช้เม้าส์คลิกบนแถบ var1 ก็จะปรากฏ Variable editor ออกมาเพื่อที่จะให้พิมพ์ชื่อตัวแปร ดังรูปที่ 4.2

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