ภาคผนวก ข.2

“คู่มือการใช้โปรแกรม R การใช้คำสั่งหรือฟังก์ชันพื้นฐาน”

เป็นส่วนหนึ่งของโครงการ

การพัฒนาการใช้โปรแกรม R ในการวิเคราะห์ข้อมูล

R Program Development for Research Utilization

โดย

หัชชา ศรีปลั่ง และคณะ

เดือน ปี ที่เสร็จโครงการตามสัญญา

ตุลาคม 2549

ต้นฉบับคู่มือ

“การใช้โปรแกรม R การใช้คำสั่งหรือฟังก์ชันพื้นฐาน”

ผู้แต่ง

นราทิพย์ จั่นสกุล

กัลยา นิติเรืองจรัส

คำนำ

มหาวิทยาลัยเป็นสถาบันทางการศึกษาระดับสูง ควรเป็นแบบอย่างที่ดีแก่สังคมในการใช้โปรแกรมทางคอมพิวเตอร์ที่ถูกกฎหมาย แต่ด้วยปัญหาที่โปรแกรมสำเร็จรูปทางสถิติที่ถูกต้องตามกฎหมายมีราคาค่อนข้างแพงและผู้ใช้มีงบประมาณจำกัด ดังนั้นทางเลือกหนึ่งคือการหันไปใช้โปแกรมที่มีการแจกจ่ายให้ฟรีโดยไม่ถือว่าเป็นการละเมิดลิขสิทธิ์ ปัจจุบันได้มีองค์กรต่าง ๆ ทั่วโลกร่วมมือกันสร้างโปรแกรมสำหรับการวิเคราะห์ข้อมูลทางสถิติ ให้เป็นทางเลือกเพื่อแก้ปัญหาดังกล่าว โปรแกรม **R** เป็นโปรแกรมทางเลือกหนึ่งที่ใช้ในการวิเคราะห์ข้อมูลทางสถิติที่สามารถนำมาใช้ได้ฟรีโดยไม่ถือว่าเป็นการละเมิดลิขสิทธิ์ ผู้ใช้สามารถ download ได้จาก web site ของโปรแกรม **R** www.r-project.org ได้โดยตรง แต่โปรแกรม **R** มีวิธีใช้ค่อนข้างใช้ยาก จึงไม่ได้รับความนิยมเท่าที่ควร ทั้งๆที่ประสิทธิภาพในการทำงานไม่ได้ด้อยไปกว่าโปรแกรมทางการค้าทั่วไป อีกทั้งในบางกรณี **R** ยังเป็นโปรแกรมที่มีประสิทธิภาพดีกว่าโปรแกรมอื่นๆ ด้วยซ้ำ

หน่วยระบาดวิทยา คณะแพทยศาสตร์ และสาขาวิชาสถิติ ภาควิชาคณิตศาสตร์ คณะวิทยาศาสตร์ มหาวิทยาลัยสงขลานครินทร์ ได้เล็งเห็นถึงปัญหาเหล่านี้ จึงได้พัฒนาโปรแกรม **R** ให้ง่ายต่อการใช้งาน สร้างคู่มือการใช้โปรแกรม **R** เป็นภาษาไทย และส่งเสริมให้คนได้รู้จักและหันมาใช้โปรแกรมนี้ในการวิเคราะห์ข้อมูลเพิ่มขึ้น คู่มือการใช้โปรแกรม **R** สำหรับการวิเคราะห์ข้อมูลสถิติฉบับนี้ ได้รวบรวมคำสั่งหรือฟังก์ชันพื้นฐานสำหรับการวิเคราะห์ข้อมูลพื้นฐานที่มีอยู่ในโปรแกรม **R** และที่คณะผู้วิจัยสร้างขึ้นใหม่ รวมทั้งวิธีการใช้ โดยยกตัวอย่างที่เป็นข้อมูลจริงซึ่งเหมาะกับนักวิจัยที่ทำการวิเคราะห์ข้อมูลด้วยสถิติพื้นฐานและอาจารย์ผู้สอนวิชาสถิติพื้นฐานในระดับปริญญาตรี เพื่อความสะดวกของผู้ใช้ คณะผู้วิจัยได้เตรียมโปรแกรม **R รุ่น 2.3.1** ซึ่งใช้ในการเตรียมต้นฉบับ รวบรวมคำสั่งที่สร้างขึ้นใหม่ไว้เป็น text file ชื่อ StatCan.txt และไฟล์ข้อมูลที่ใช้สาธิตไว้ใน CD ที่แนบมาพร้อมกับคู่มือเล่มนี้

คณะผู้วิจัยขอขอบคุณ เครือข่ายวิจัยสุขภาพ สำนักงานกองทุนสนับสนุนงานวิจัย (สกว) และมูลนิธิสาธารณสุขแห่งชาติ (มสช) ที่สนับสนุนทุนวิจัย ภาควิชาคณิตศาสตร์ คณะวิทยาศาสตร์ และหน่วยระบาดวิทยา คณะแพทยศาสตร์ มหาวิทยาลัยสงขลานครินทร์ ที่ให้ความอนุเคราะห์ในด้านเครื่องมือ และสถานที่ในการทำวิจัย คณะผู้วิจัยหวังเป็นอย่างยิ่งว่าคู่มือการใช้โปรแกรม **R** เล่มนี้จะเป็นประโยชน์สำหรับนักวิจัย อาจารย์และผู้อ่านทุกท่าน

คณะผู้วิจัย

กันยายน 2549

สารบัญ

สารบัญ (ต่อ)

สารบัญ (ต่อ)

บทที่ 1

โปรแกรม R (R Program)

1.1 แนะนำโปรแกรม R

โปรแกรมสำเร็จรูป **R** เป็น Open Source Software ที่นำเสนอให้ฟรี สามารถดัดแปลงได้ตามต้องการอย่างอิสระ แจกจ่ายได้ แต่ห้ามจำหน่ายโดยหวังผลกำไร 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 จึงเป็นโอกาสอันดีสำหรับนักวิชาการในประเทศไทยที่จะนำไปใช้ในด้านต่าง ๆ เช่น **R** ช่วยในการเรียนการสอนด้านสถิติให้เข้าใจเนื้อหาทางทฤษฎีได้ลึกซึ้ง เปิดโอกาสในนักวิจัยสามารถเขียนฟังก์ชันและกราฟใหม่ ๆ สำหรับใช้งานเฉพาะด้านของตน และที่สำคัญคือสามารถเข้าร่วมเรียนรู้กับนานาชาติ โดยการร่วมเขียน module ตามโจทย์และศักยภาพที่เรามีอยู่ วิสัยทัศน์ของการพัฒนาและพึ่งตนเองพร้อม ๆ กับประสานงานกับกลุ่มนานาชาติในการทำงานทางวิชาการที่ไม่ผูกติดกับผลประโยชน์ทางการค้า เพื่อพัฒนาสปิริตการสร้างสมบัติวิชาการที่เป็นสาธารณะของโลก ซึ่งประเทศไทยยังขาดการเรียนรู้ด้านนี้มาก จึงเป็นสิ่งที่ควรพัฒนาขึ้น

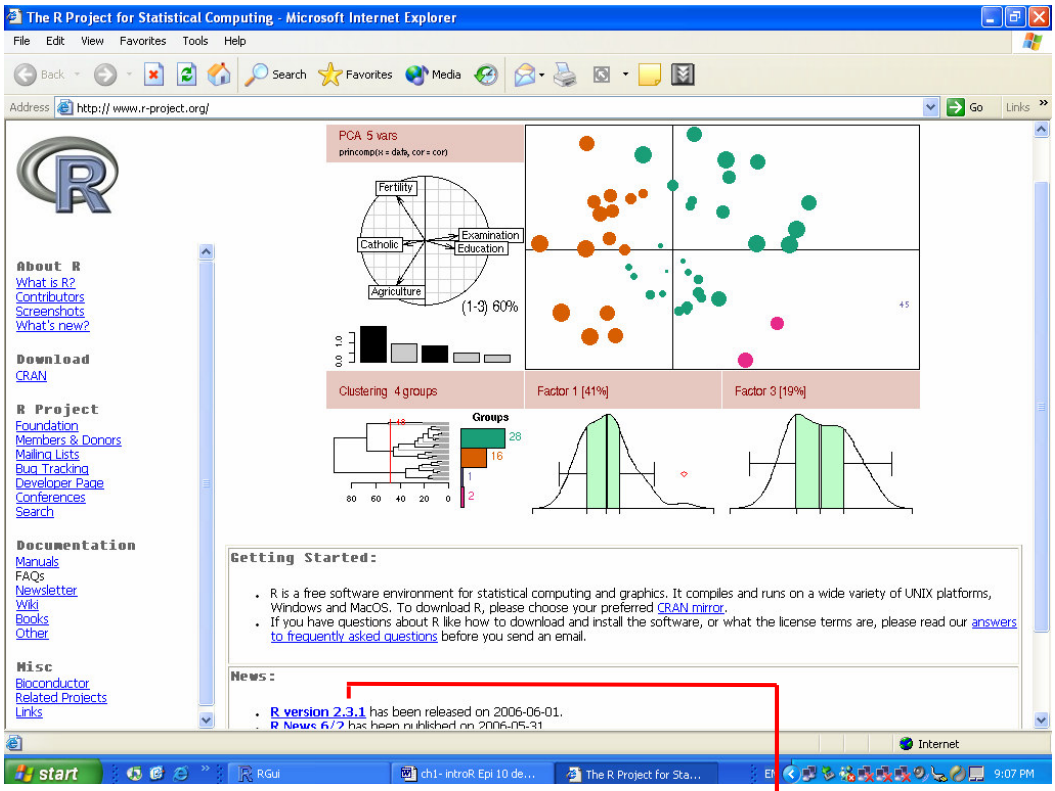
1.2 เหตุผลในการใช้โปรแกรม R

การประกาศนโยบายรับรองทรัพย์สินทางปัญญาเป็นนโยบายหนึ่งของรัฐบาล โดยเฉพาะอย่างยิ่งซอฟต์แวร์ทางคอมพิวเตอร์ โดยมีประกาศให้หน่วยราชการต่าง ๆ ใช้ซอฟต์แวร์ที่ถูกกฎหมายเท่านั้น อย่างไรก็ตาม ในทางปฏิบัติ หน่วยราชการต่าง ๆ ให้มีงบประมาณในการจัดซื้อซอฟต์แวร์จำกัด หากจัดซื้อจริงคาดว่าค่าใช้จ่ายต่างๆ จะสูงขึ้นอย่างมาก โปรแกรมทางคอมพิวเตอร์ที่ใช้ใน

การวิเคราะห์ข้อมูลทางสถิติส่วนใหญ่เป็นโปรแกรมที่มีลิขสิทธิ์ ในปัจจุบันการวิเคราะห์ข้อมูลทางสถิติ ต้องอาศัยโปรแกรมสำเร็จรูป (software) ที่สามารถรองรับข้อมูลที่มีขนาดใหญ่ และระเบียบวิธีทางสถิติที่สลับซับซ้อน อย่างไรก็ตามซอฟต์แวร์ที่มีประสิทธิภาพดังกล่าว ถูกผลิตขึ้นเพื่อการค้า และมีราคาแพง หน่วยงานต่าง ๆ จึงใช้ซอฟต์แวร์อย่างผิดกฎหมาย มหาวิทยาลัยเป็นสถาบันการเรียนการสอน ควรเป็นแบบอย่างที่ดีแก่สังคมภายนอกในการใช้ซอฟต์แวร์อย่างถูกกฎหมาย ดังนั้นการพัฒนาการใช้ซอฟต์แวร์ที่ไม่ใช่เพื่อการค้าสำหรับวิเคราะห์ข้อมูล จึงเป็นสิ่งจำเป็นในการส่งเสริมให้มีการใช้อย่างแพร่หลายขึ้น เพื่อให้หลุดพ้นจากปัญหาเชิงการค้าให้ได้มากที่สุด

1.3 การ download โปรแกรม R

โปรแกรม **R** สามารถเข้าไป download ได้ที่ web site ของ **R** โดยตรงคือ http:// www.r-project.org ดังรูป 1.1 ปัจจุบันโปรแกรม **R** ที่ download จะอยู่ใน version 2.3.1

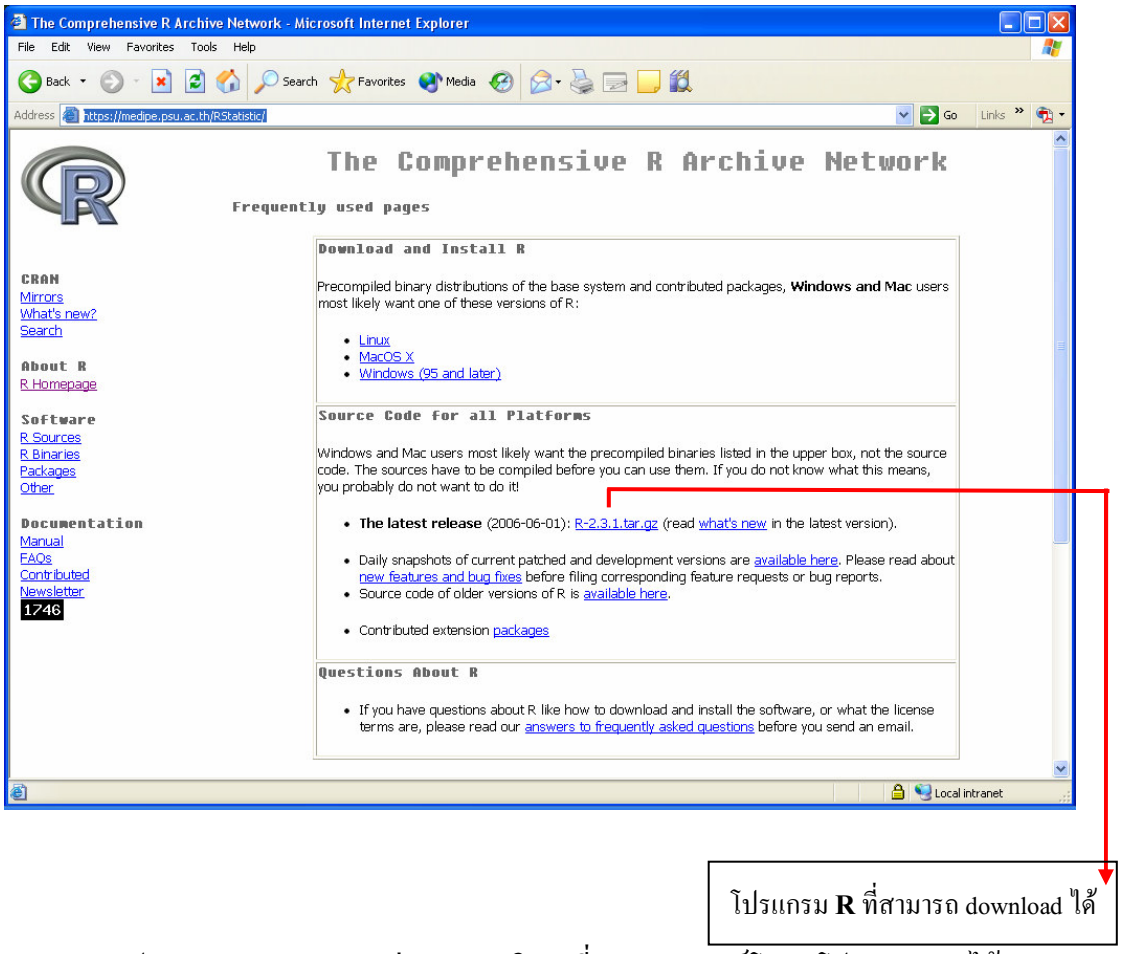


รูป 1.1 Web page **R**

โปรแกรม **R** ที่ให้ download ได้ฟรี

หมายเหตุ โปรแกรม **R** version 2.3.1 ออกเมื่อเดือนมิถุนายน 2549

นอกจากนี้ ผู้ใช้สามารถดาวน์โหลดโปรแกรม **R** จาก website ของหน่วยระบาดวิทยา คือ https://medipe.psu.ac.th/RStatistic/ ดังรูป 1.2



รูป 1.2 Web page ของหน่วยระบาดวิทยาที่สามารถดาวน์โหลดโปรแกรม **R** ได้

1.4 การติดตั้งโปรแกรม R

เมื่อ download โปรแกรมมาจาก web site ก็จะได้ไฟล์ชื่อ R-2.3.1-win32.exe ทำการติดตั้งโปรแกรม **R** ดังนี้

- Double click ที่ไฟล์ชื่อ R-2.3.1-win32.exe
- ทำตามขั้นตอน โดยการคลิก Next ต่อไปเรื่อย ๆ โปรแกรม ก็จะถูกติดตั้ง เมื่อติดตั้งเสร็จแล้ว ก็จะปรากฏ icon ของโปรแกรม ที่ desktop โดยเป็นสัญญักษณ์ตัวอักษร **R** ที่มีสีน้ำเงิน เมื่อดับเบิ้ลคลิกที่ไอคอน โปรแกรม **R** ก็จะถูกเปิดขึ้น

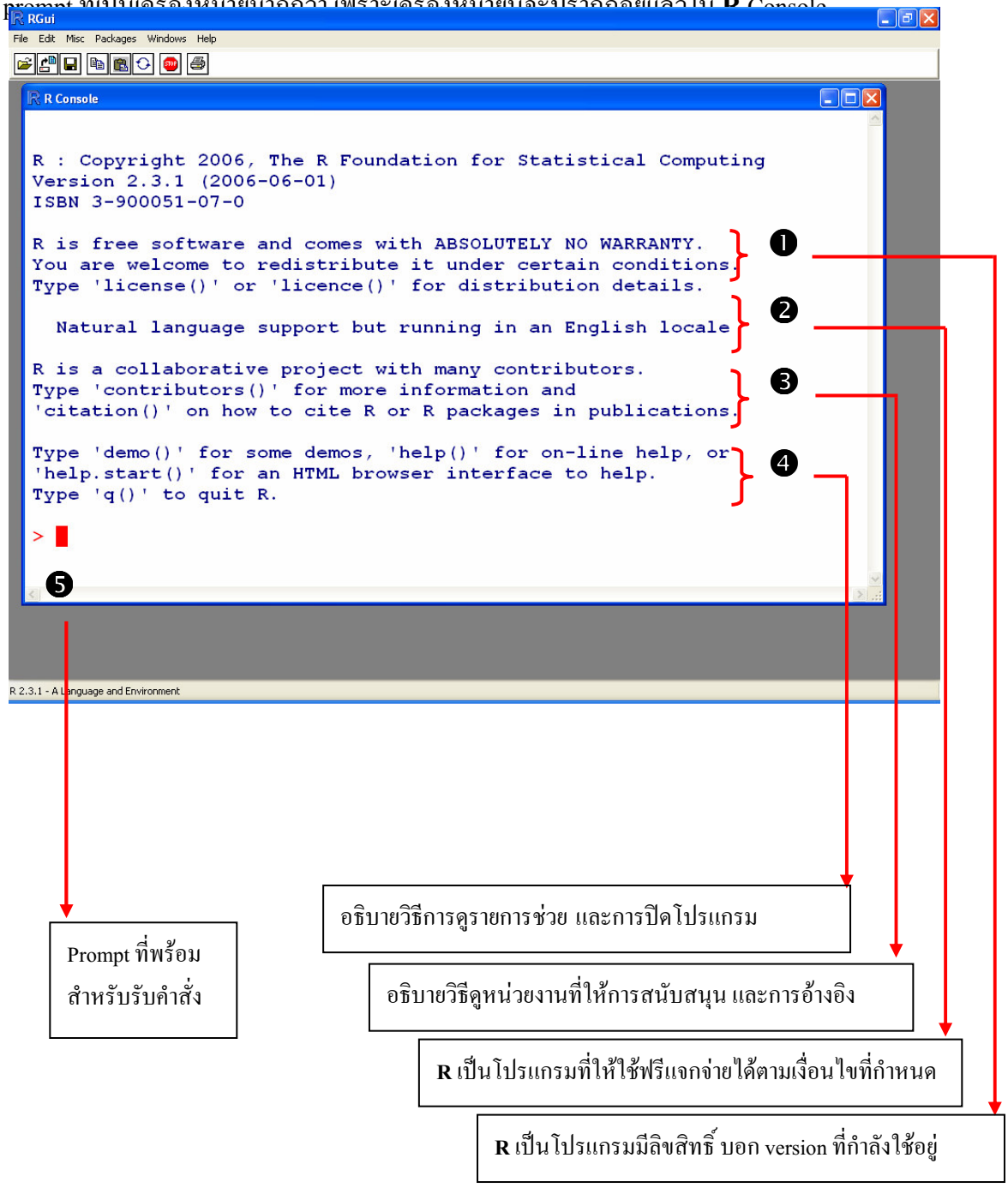
หลังจากการติดตั้ง โปรแกรม **R** จะถูกเก็บไว้ที่ C:\Program Files\R\R-2.3.1\ และ shortcut icon จะถูกสร้างไว้บน desktop โดยอัตโนมัติ แฟ้มข้อมูล ฟังก์ชัน ตัวแปร ผลลัพธ์ เป็นต้น ควรจะเก็บไว้ที่ working folder ที่ผู้ใช้ได้สร้างไว้ ซึ่งควรเก็บไว้ใน folder แยกต่างหากจะสะดวกกว่า เช่นเก็บไว้ที่ C:\RWorkplace โดยใช้ mouse วางที่ไอคอน **R** คลิ๊กที่ปุ่มขวาแล้วเลือก Properties จากนั้นพิมพ์คำว่า C:\RWorkplace ใน 'Start in' ของ Properties ดังรูป 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"

รูป 1.3 การตั้งค่าใน Start in

1.5 การเริ่มทำงานกับโปรแกรม R

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

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



รูป 1.4 หน้าจอ Rgui (**R** Graphic User Interface)

เนื่องจากโปรแกรม **R** เป็นโปรแกรมที่ต้องป้อนคำสั่ง หากไม่ทราบอะไรเลย สิ่งหนึ่งที่จะช่วยได้คือการเรียกดูรายการช่วย ดังอธิบายไว้แล้วในส่วนที่ ❹ คำสั่งที่จะขอดูรายการช่วย คือ

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

หากต้องการออกจากโปรแกรม ก็จะมีบอกไว้แล้วในส่วนที่ ❹ เช่นกัน นั่นคือ พิมพ์คำว่า

```
> q()
```

โปรแกรมจะถามว่า Save work space image? ถ้าเลือก Yes โปรแกรมก็จะ save work space ทุกอย่างที่เราได้สร้างไว้ โดยให้นามสกุลของไฟล์เป็น ".Rdata" ควรจะเลือก Yes ก็ต่อเมื่อต้องการบันทึกงานค้างที่ยังทำไม่เสร็จและอยากกลับมาทำต่อเนื่องในทันทีที่เปิด **R** มาใช้งาน

1.6 คำและความหมายในโปรแกรม R

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

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 เช่น ตัวแปร ตัวเลข เมื่อมีการตั้งชื่อของ เวกเตอร์, กรอบข้อมูล หรือฟังก์ชัน ซ้ำกัน วัตถุเดิมจะถูกทำลาย (ถูกปลดจากการอ้างอิง แล้วถูก garbage collector ทำลายในภายหลัง) ดังนั้น พึงระวังในการตั้งชื่อซ้ำกัน เพราะจะไม่มีการเตือน

ตัวอย่างคำสั่งในการสร้างวัตถุขึ้นมาใหม่ ดังต่อไปนี้

```
> x = 6 + 9        ❶

> x <- 6 + 9       ❷
```

คำอธิบาย

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

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

[1] 15 → 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 จะไม่สามารถอ้างอิงเรียกใช้ภายหลังได้ และจะถูกทำลายโดยอัตโนมัติ หากต้องการเก็บผลลัพธ์ไว้ใช้ในอนาคต ควรกำหนดให้เก็บไว้ในวัตถุตามชื่อที่กำหนด

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

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

คำอธิบาย

จากตัวอย่างคำสั่งที่ ❶ เป็นการกำหนดชื่อวัตถุที่เก็บผลลัพธ์เป็นชื่อ x เมื่อพิมพ์คำว่า x ผลลัพธ์ถูกแสดงในบรรทัดถัดมา คำสั่งที่ ❷ เป็นคำสั่งเช่นเดียวกับคำสั่งที่ ❶ แต่ใส่คำสั่งทั้งหมดไว้ในวงเล็บ โดยย่อมาจากคำสั่ง print(x <- c(10, 20, 30, 40, 50) เพื่อให้โปรแกรมแสดงผลลัพธ์ออกมาทันที โดยไม่ต้องพิมพ์ชื่อวัตถุ ดังคำสั่งที่ ❶

1.6.3 ลิสต์ (List)

ลิสต์เป็นวัตถุที่องค์ประกอบภายในเรียงกันเป็นลำดับ ซึ่งจะเรียกองค์ประกอบภายในนั้นว่า component ไม่มีความจำเป็นที่องค์ประกอบของลิสต์จะต้องเป็นชนิดเดียวกันทั้งหมด ลิสต์สามารถรวมเอาเวกเตอร์ตัวเลข, เมทริกซ์, เวกเตอร์ซ้อน, อาเรย์, ตัวอักษร, ฟังก์ชัน และอื่น ๆ เข้าไว้ด้วยกัน 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 รายละเอียด ต่าง ๆ ภายในลิสต์จะปรากฏขึ้น โดยมีเครื่องหมาย $ หน้าชื่อตัวแปรเป็นการบอกให้ทราบว่า ชื่อที่อยู่หลังเครื่องหมายเป็นองค์ประกอบ (component) ที่อยู่ในลิสต์ องค์ประกอบเหล่านี้จะยังไม่เรียกว่าตัวแปร คำว่าตัวแปรจะใช้เมื่อเป็นองค์ประกอบในกรอบข้อมูล เพราะองค์ประกอบในลิสต์ อาจเป็นฟังก์ชัน หรือกรอบข้อมูล ก็ได้

แบบฝึกหัด

ลองทำตามคำสั่งดังต่อไปนี้ และค่าที่ได้จากคำสั่งเหล่านี้คืออะไร

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

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

1.6.4 กรอบข้อมูล (Data frame)

คือลิสต์ที่อยู่ในคลาส “data.frame” แต่มีข้อจำกัดคือ ลิสต์ที่จะทำเป็นกรอบข้อมูลได้ ต้องประกอบด้วย เวกเตอร์, เมทริกซ์ตัวเลข หรือ กรอบข้อมูล

กรอบข้อมูลใหม่จะได้ตัวแปรมาจากสดมภ์ของเมทริกซ์ องค์ประกอบของลิสต์หรือตัวแปรในกรอบข้อมูลที่นำมารวมกันนั้น จะต้องมีขนาดความยาวเดียวกัน และโครงสร้างของเมทริกซ์ จะต้องมีความยาวของแถวเท่ากัน วัตถุประสงค์ของข้อจำกัดข้างต้น ก็คือ ต้องการทำให้ข้อมูลทั้งหมดบรรจุลงในสดมภ์ (ตัวแปร) ของกรอบข้อมูลได้พอดีด้วยฟังก์ชัน data.frame() ดังนี้

```
> x <- c(10, 5.6, -3, 6.4, 21.7)                    ❶
> y <- c(5.1, 3, 0.5, 11, 2.5)                      ❷
> z <- c("a1", "a2", "a3", "a4", "a5")              ❸

> data.dat <- data.frame(x,y,z)                     ❹
> rm(x, y, z)                                       ❺

> data.dat                                          ❻
     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)                              ❼
$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

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 และ ตัวอักษรต่าง 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** จึงมีการพัฒนาและเปลี่ยนแปลงเวอร์ชันบ่อยๆ โดยมีการเปลี่ยนแปลงเวอร์ชันทุก ๆ 1 – 2 เดือน ซึ่งแสดงให้เห็นว่ามีผู้พัฒนา **R** อยู่ทั่วทุกมุมโลก **R** 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)

ฟังก์ชัน	ความหมาย	ตัวอย่าง
abs()	ค่าสัมบูรณ์	abs(-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

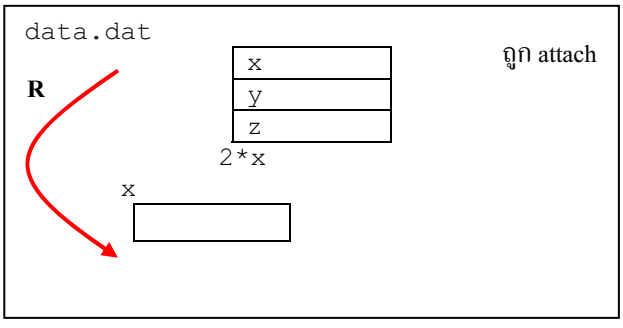
```
> data.dat$x
```

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** มองเห็นชื่อตัวแปรภายในกรอบข้อมูลที่ได้ attach แล้วนั้นโดยไม่ต้องเรียกชื่อกรอบข้อมูลนำหน้า ดังนั้น การบวก x เข้ากับ y เข้าด้วยกัน จะทำได้ง่ายขึ้น ดังนี้

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



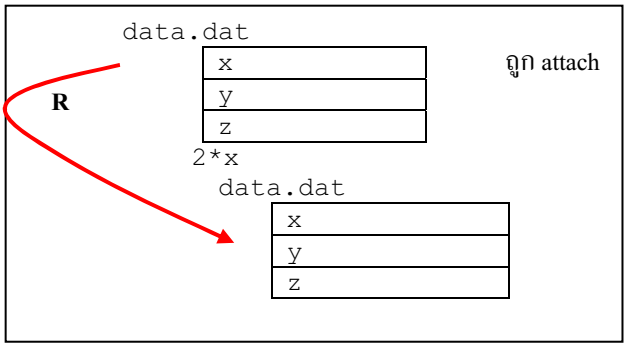
ถึงจุดนี้พึงสังเกตว่าคำสั่งกำหนดค่าต่อไปนี้

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

จะไม่เป็นการเปลี่ยนค่า x ใน data.dat ให้เป็น 2*x แต่จะเป็นการสร้างตัวแปร x ขึ้นมาใหม่ นอกกรอบข้อมูล 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) จะต้องใช้คำสั่ง detach() และเมื่อ attach กรอบข้อมูล data.dat ใหม่ ค่า x ที่เรียกได้จึงจะแสดงค่าใหม่ให้เห็น ดังนี้

```
> x
```
ให้ค่า x เดิม

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

หมายเหตุ	detach กรอบข้อมูลที่ได้ attach ไว้ทุกครั้งที่ทำงานกับกรอบข้อมูลเสร็จแล้ว มิฉะนั้นจะทำงานกับข้อมูลผิดชุดได้ หากต้องการทำงานกับกรอบข้อมูลนั้นอีก ก็ให้ attach ใหม่อีกครั้ง

ฟังก์ชัน search() และ searchpaths()

ฟังก์ชัน search เป็นการค้นหาวัตถุที่มีอยู่ใน **R** console ในขณะที่กำลังเปิดใช้งาน คำสั่งนี้จะแสดง package 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()  ❷
[1] ".GlobalEnv"
[2] "C:/PROGRA~1/R/RW2001/library/methods"
```

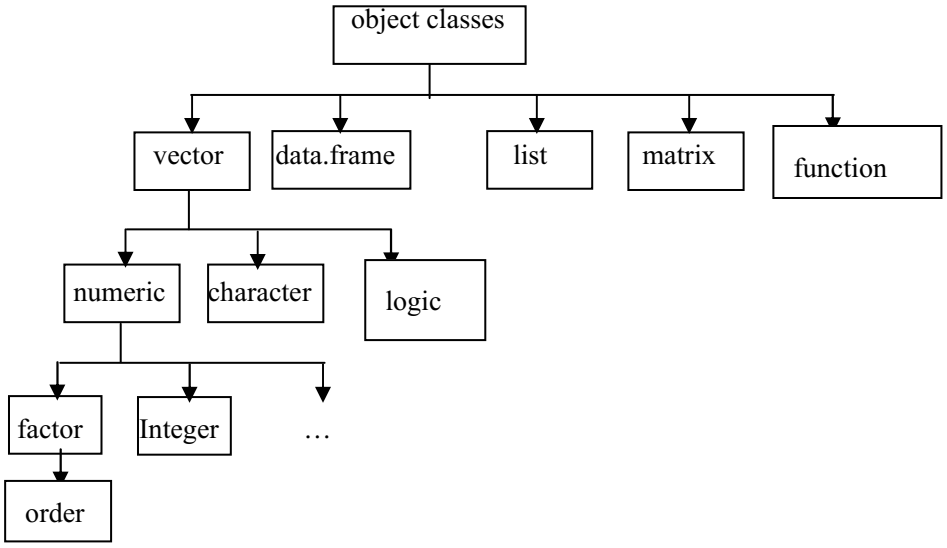
```
[3] "C:/PROGRA~1/R/RW2001/library/stats"
[4] "C:/PROGRA~1/R/RW2001/library/graphics"
[5] "C:/PROGRA~1/R/RW2001/library/grDevices"
[6] "C:/PROGRA~1/R/RW2001/library/utils"
[7] "C:/PROGRA~1/R/RW2001/library/datasets"
[8] "Autoloads"
[9] "C:/PROGRA~1/R/RW2001/library/base"
```

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, เมทริกซ์ เป็นต้น และในคลาสใหญ่คลาสหนึ่งอาจมีคลาสย่อยต่าง ๆ ได้อีก เช่น แฟกเตอร์ เป็นคลาสย่อยของคลาสเวกเตอร์ตัวเลข ยกตัวอย่างดังแผนภาพต่อไปนี้



1.6.7 แฟกเตอร์ (Factor)

แฟกเตอร์ เป็นคลาสย่อยของเวกเตอร์ตัวเลขที่มีคุณสมบัติพิเศษ คือ แต่ละค่าจะเป็นรหัสแทนความหมายใด ๆ เช่น เพศ ให้ตัวเลข 1 และ 2 โดยที่ เลขรหัส 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 มิติ ก็ได้ แต่ในทางสถิติมักจำกัด อาเรย์ไว้เพียง 3 มิติเท่านั้น

1.6.9 เมทริกซ์ (Matrix)

เมทริกซ์ เป็น เวกเตอร์หลาย ๆ เวกเตอร์มาประกอบกัน ซึ่งเวกเตอร์จะมีเพียงแถวเพียงหนึ่งแถว ส่วนเมทริกซ์ เป็นการรวมเวกเตอร์ที่เป็นตัวเลขเข้าด้วยกัน จึงมีแถวหลายแถว ดังนั้นเมทริกซ์จึงเป็นอาเรย์ที่มี 2 มิติ คือ แถวกับสดมภ์ เมทริกซ์ทำให้เป็นกรอบข้อมูลได้ แต่กรอบข้อมูลที่สามารถนำมาเปลี่ยนให้เป็นเมทริกซ์ได้นั้น ตัวแปรทุกตัวต้องเป็นชนิดเดียวกัน

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

1.7 สัญลักษณ์ที่ใช้ในโปรแกรม R

ในการเขียนคำสั่ง ในโปรแกรม **R** จะมีสัญลักษณ์พิเศษที่มักจะพบเจอในโปรแกรมนี้ ได้แก่

1.7.1 สัญลักษณ์ทั่ว ๆ ไป

สัญลักษณ์	ความหมาย
<−	Left assignment หมายถึง ให้นำค่าหรือผลลัพธ์จากการคำนวณหรือการทำงานของฟังก์ชันที่ปลายลูกศร ไปใส่ในวัตถุที่เขียนไว้ด้านหน้าหัวลูกศร (ด้านซ้าย)
−>	Right assignment หมายถึง ให้นำค่าหรือผลจากการคำนวณหรือการทำงานของฟังก์ชันที่ปลายลูกศร ไปใส่ในวัตถุที่เขียนไว้ด้านหน้าหัวลูกศร (ด้านขวา)
=	เป็นเครื่องหมาย assignment เช่นเดียวกับ <- แต่ไม่ควรใช้ เพราะจะสับสน

	กับเครื่องหมาย = = 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 การใช้ตัวอักษรใน R

โปรแกรม **R** จะใช้อักษรภาษาอังกฤษเป็นภาษาหลักและถือว่าภาษาอังกฤษตัวพิมพ์เล็กและตัวพิมพ์ใหญ่ 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(name, age, wt, ht)
# สร้างกรอบข้อมูล person.dat มีตัวแปรชื่อ name, age, wt และ ht
```

1.8.6 การเรียกใช้คำสั่งเดิม

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

1.8.7 การเรียกดูวัตถุที่ถูกสร้างไว้แล้วใน R console

การทำงานใน **R** ผู้ใช้มักจะสร้างวัตถุต่าง ๆ ขึ้นมา บางครั้งไม่สามารถจำชื่อได้หมด หรือ ชื่อวัตถุที่ถูกสร้างใหม่มีชื่อเหมือนกับชื่อตัวแปรในกรอบข้อมูล ในกรณีที่ชื่อวัตถุและชื่อตัวแปรในกรอบข้อมูลมีชื่อเหมือนกัน **R** จะเรียกใช้วัตถุที่อยู่นอกกรอบข้อมูลมาทำงานก่อนเสมอ ดังนั้นจึงต้องมีการตรวจสอบเสมอว่า ได้สร้างวัตถุอะไรไปบ้าง ตรวจสอบว่ามีวัตถุอะไรที่อยู่นอกกรอบข้อมูล 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 name wt และ ht ถูกลบไปแล้ว โดยในกรอบข้อมูล person.dat มีตัวแปรชื่อ age, name, wt และ ht เช่นกัน ดังนั้นการลบวัตถุ age, name, wt และ ht ออก ก็จะช่วยลดการสับสนของการเรียกใช้ตัวแปรในภายหลัง หากต้องการลบวัตถุทุกชนิดออกจาก **R** console ก็สามารถทำได้ด้วยการใช้คำสั่งดังนี้

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

1.8.10 การลบผลลัพธ์ออกจากหน้าจอ R console

ในบางครั้งเมื่อผู้ใช้ทำงานไปนาน ๆ ก็จะมีผลลัพธ์ปรากฏขึ้นหน้าจอเป็นจำนวนมาก หากไม่ต้องการผลลัพธ์ดังกล่าว สามารถลบออกจากหน้าจอ **R** console ได้ โดยการไปเลือกที่เมนู Edit และ เลือก 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

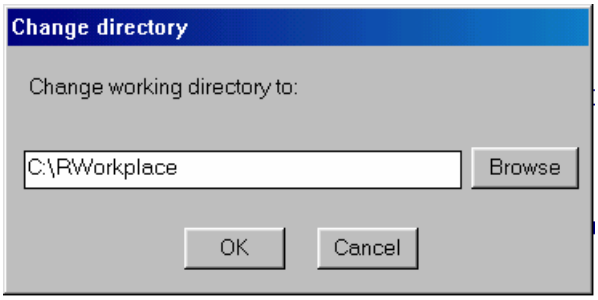
หากคำสั่งที่พิมพ์ไม่ครบถ้วน เมื่อกด Enter **R** จะไม่กระทำตามคำสั่ง แต่จะขึ้นบรรทัดใหม่ พร้อมด้วยเครื่องหมาย + นั่นหมายถึงคำสั่งยังไม่ครบถ้วน เช่น ขาดวงเล็บปิด ก็จะต้องพิมพ์วงเล็บปิดแล้วจึงกด Enter **R** จึงจะกระทำตามคำสั่งนั้น แต่ถ้าหากคำสั่งนั้น ๆ ตกวงเล็บหลายวงเล็บ **R** จะไม่ให้ผู้ใช้แก้ไขคำสั่งนั้น 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… ก็จะปรากฏดังรูป 1.5 ให้พิมพ์ C:\RWorkplace (ถ้าเป็นการเปลี่ยน Folder หรือ Directory ที่เลือกจากเมนู จะใช้เครื่องหมาย \ เหมือนกับโปรแกรมอื่น ๆ)



รูป 1.5 การเปลี่ยน working directory

1.8.14 การเรียกดูโฟลเดอร์ที่ทำงานอยู่ในปัจจุบัน

ในกรณีที่ต้องการทราบโฟลเดอร์ที่กำลังทำงานอยู่ในปัจจุบันสามารถเรียกดูได้จากคำสั่ง **`getwd()`** ดังนี้

```
> getwd()
```

1.9 การเรียกใช้การช่วยเหลือ

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

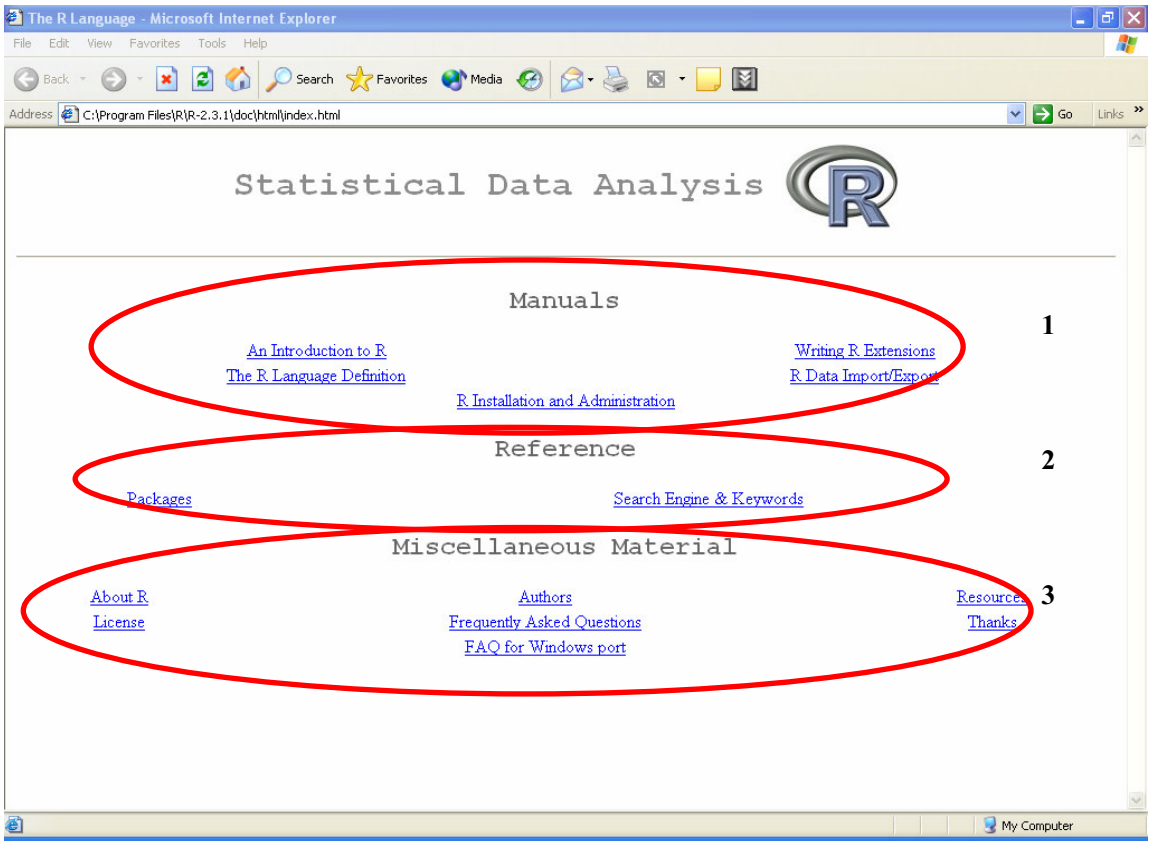
```
> help.start()        ❶ หรือ
> help()              ❷ หรือ
> ?help               ❸
```

คำอธิบาย

คำสั่งที่ ❶ ใช้ในกรณีที่ต้องการทราบรายละเอียดต่าง ๆ ใน **R** เมื่อพิมพ์คำสั่งนี้ หน้าจอ online help ดังรูป 1.6 ก็จะปรากฏขึ้น ซึ่งจะมีให้เลือก 3 ส่วนด้วยกัน 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



รูป 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

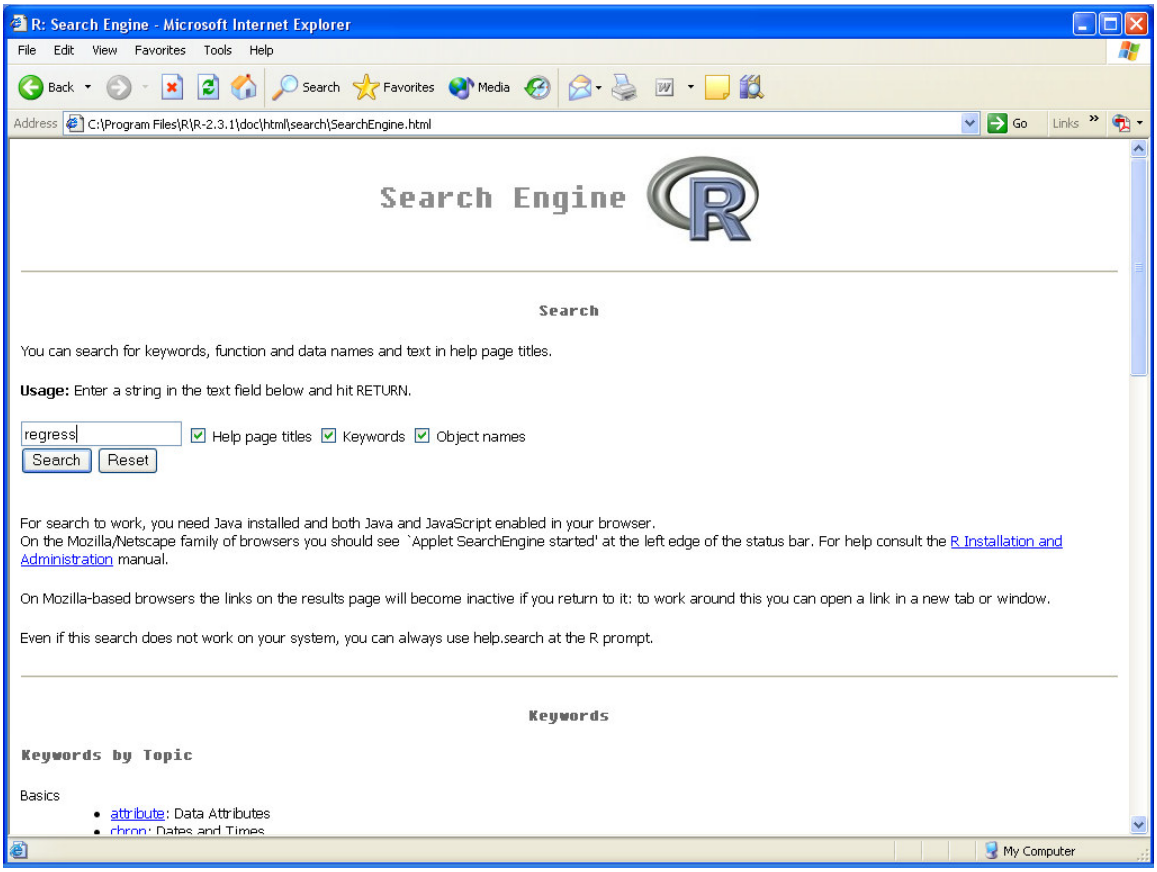
คำสั่งทั้งสองคำสั่งจะแสดงหน้าจอ online help ของการใช้คำสั่ง mean เหมือนกัน ฉะนั้นหากต้องการเรียกความช่วยเหลือที่สั้นกะทัดรัดก็สามารถใช้คำสั่งที่ ❷

1.9.2 กรณีไม่ทราบคำสั่งที่มีอยู่ใน R

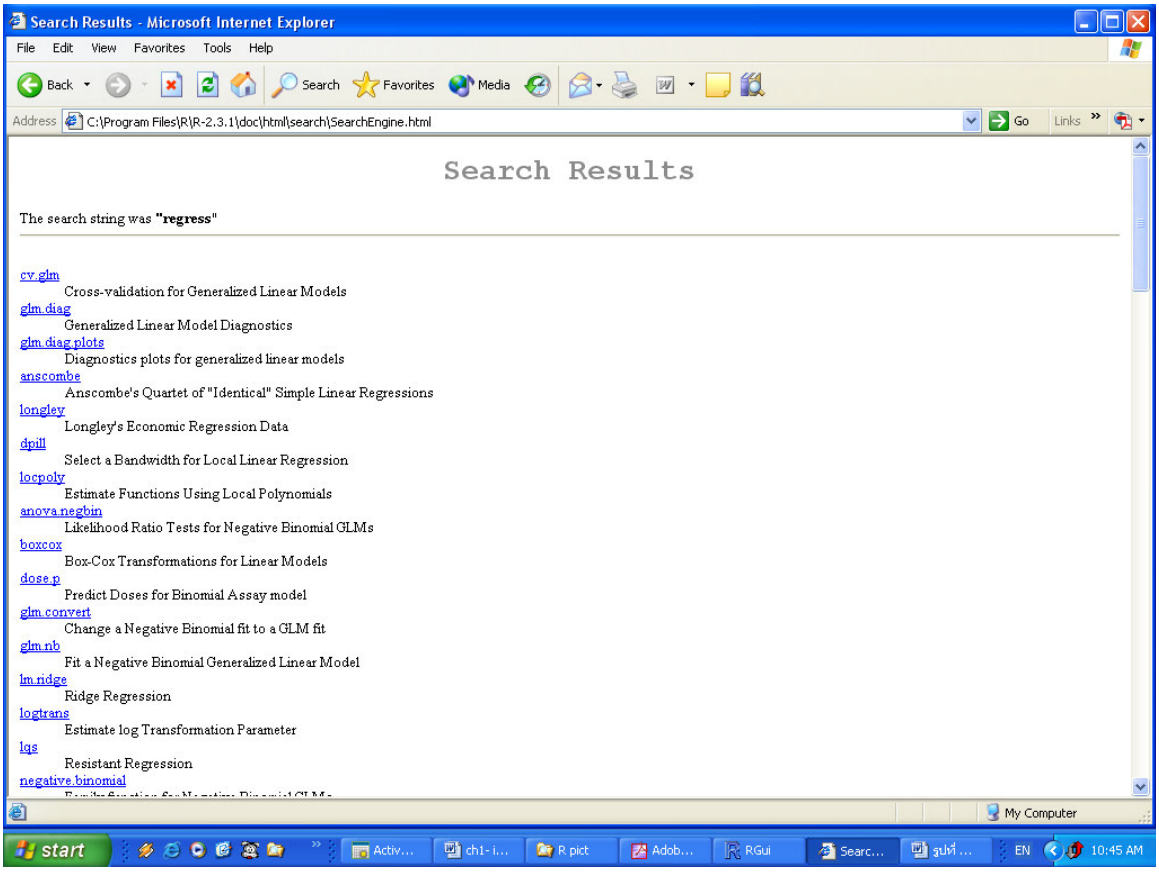
สำหรับกรณีนี้ หากทราบคำบางคำที่เกี่ยวข้องกับงานที่ต้องการใช้ ซึ่งอาจเป็นคำสำคัญ (keyword) เช่น คำว่า test การเรียกดูความช่วยเหลือจะต้องพิมพ์คำนั้นไว้ในเครื่องหมาย " " ดังเช่น

- `help.search("test")`

อย่างไรก็ตาม การพิมพ์ **help.start()** เพื่อเปิด online help ดังรูป 1.6 แล้วเลือก Search Engine & Keywords จะเป็นวิธีที่ง่ายและมีตัวเลือกให้มากกว่า เช่นต้องการค้นคำว่า regression ก็ให้คลิกที่เมนู ดังกล่าวก็จะปรากฏหน้าจอดังรูป 1.7 ขึ้น และให้พิมพ์คำว่า regression ในช่องว่าง จากนั้นจึงกด ปุ่ม Search ผลการค้นหาดังในรูป 1.8



รูป 1.7 หน้าจอ Search Engine ใน **R**



รูป 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

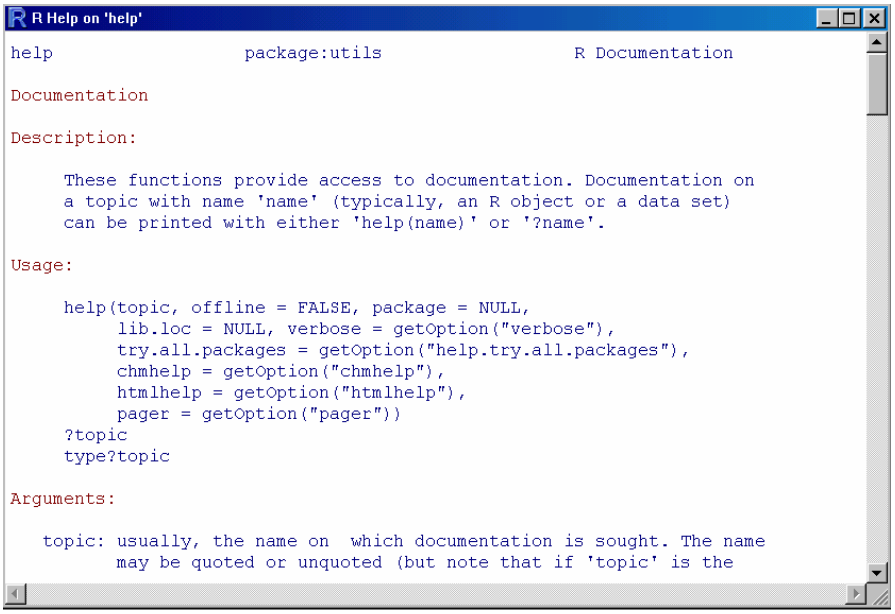
R แสดงหน้าจอ help เป็น 3 รูปแบบด้วยกัน คือ

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

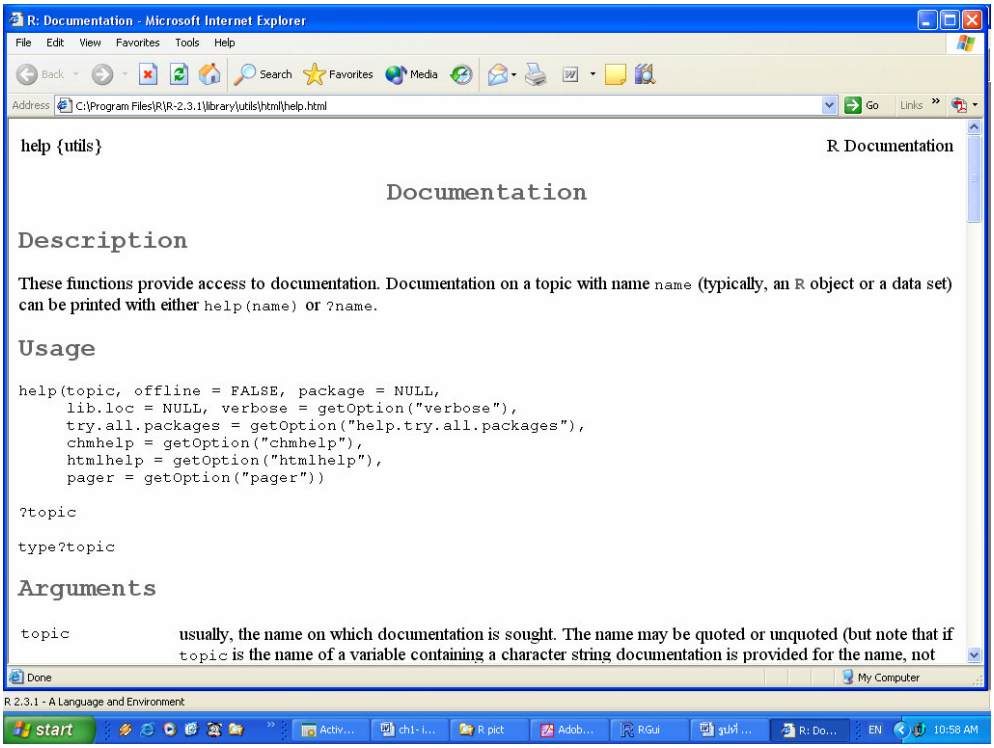
```
> help(help)
```

จากคำสั่งนี้ **R** จะแสดงหน้าจอ help ดังรูป 1.9



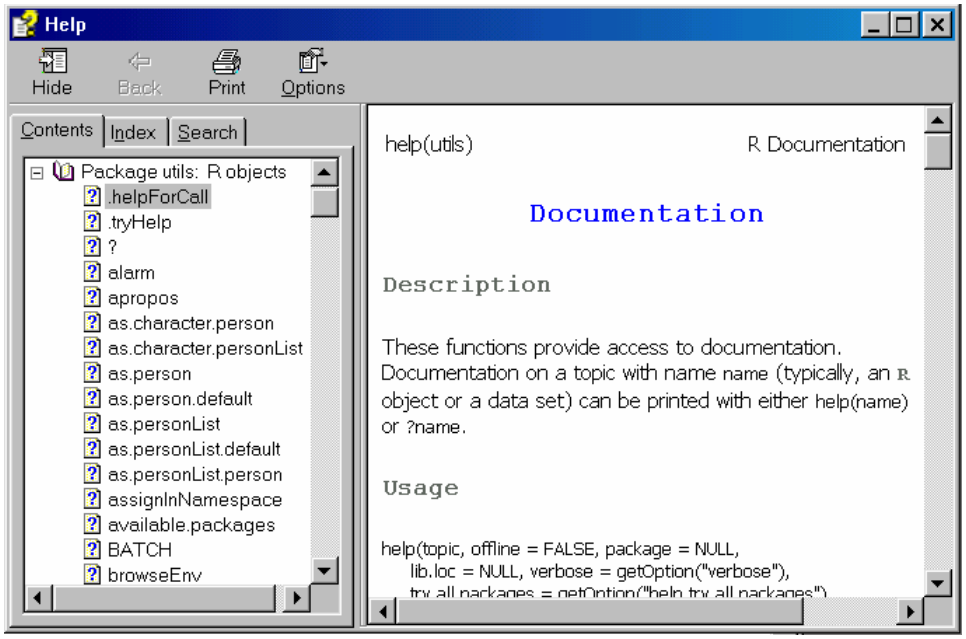
รูป 1.9 หน้าจอ help ภายใต้ RGUI

`> help(help, html=T)` จะแสดงหน้าจอ help ด้วย web browser ดังรูป 1.10



รูป 1.10 หน้าจอ help จาก web browser

`> help(help, chm=T)` จะแสดงหน้าจอ help ดังรูป 1.11



รูป 1.11 หน้าจอ help จาก Complied html

แบบฝึกหัด

จงใช้โปรแกรม **R** คำนวณค่าต่าง ๆ ต่อไปนี้

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

	เป็นโรค	ไม่เป็นโรค	รวม
ชาย	50 (a)	120 (b)	170
หญิง	75 (c)	205 (d)	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 $OR = ad/bc$

4. จงสร้าง data frame ของข้อมูลต่อไปนี้

x	y	z	x	y	z
35	65	6	7	9	4
7	9	7	20	35	5
23	41	4	3	1	4
23	41	7	19	33	5
25	45	5	0	0	6
29	53	6	28	51	1
25	45	6	30	55	5

บทที่ 2
การจัดการกับข้อมูล (Data Management)

การจัดการกับข้อมูลเป็นการเตรียมข้อมูลให้พร้อมก่อนการนำไปวิเคราะห์ข้อมูลเพื่ออะไร ซึ่งในบทนี้จะกล่าวถึง การสร้างตัวแปร การสร้างแฟ้มข้อมูลด้วยโปรแกรม **R** การอ่านข้อมูลที่สร้างจากโปรแกรมต่างๆ เข้ามาในโปรแกรม **R** การเก็บบันทึกผลการประมวลผล การปรับเปลี่ยนข้อมูล การเลือกข้อมูล และการสุ่มตัวอย่าง

2.1 คำสั่งใน R ที่ใช้ในการจัดการกับข้อมูล

คำสั่ง	คำอธิบาย
การสร้างวัตถุใน R	
c()	นำข้อมูลหรือวัตถุที่ต้องการมาเรียงต่อกัน วัตถุนั้นจะเป็นค่าคงที่ เวกเตอร์ หรือเป็นวัตถุชนิดอื่นๆก็ได้
seq()	สร้างเวกเตอร์ที่เป็น sequence
rep()	สร้างเวกเตอร์ที่เป็นตัวเลขหรืออักษรซ้ำ ๆ
คำสั่ง	**คำอธิบาย**
factor()	สร้างเวกเตอร์ที่ค่าเป็นตัวแปรเชิงกลุ่มหรือ factor
as.integer()	เปลี่ยนคุณสมบัติของวัตถุที่อยู่ใน () ให้เป็น integer
as.factor()	เปลี่ยนคุณสมบัติของวัตถุที่อยู่ใน ()ให้เป็น factor
as.numeric()	เปลี่ยนคุณสมบัติของวัตถุที่อยู่ใน ()ให้เป็น numeric
data.frame()	สร้างกรอบข้อมูล (แฟ้มข้อมูล)
cbind()	สร้างเมทริกซ์ จากเวกเตอร์หรือกรอบข้อมูลโดยนำมาเรียงต่อกันในแนวคอลัมน์
rbind()	สร้างเมทริกซ์ จากเวกเตอร์หรือกรอบข้อมูลโดยนำมาเรียงต่อกันในแนวแถว
attributes(x)	ตรวจสอบว่าวัตถุ x มีลักษณะใดและมีองค์ประกอบอะไรบ้าง
การอ่านแฟ้มข้อมูล	
library (foreign)	เรียก `library` ชื่อ `foreign` เพื่อเตรียมอ่านแฟ้มข้อมูลที่

	สร้างขึ้นด้วยโปรแกรมอื่น เช่น **SPSS STATA**
read.spss()	อ่านแฟ้มข้อมูล **spss** เข้าใน **R**
read.table()	อ่านแฟ้มข้อมูล ที่มีนามสกุล .txt เข้าใน **R**
read.csv()	อ่านแฟ้มข้อมูล ที่มีนามสกุล .csv เข้าใน **R**
การบันทึกผลการวิเคราะห์	
write.table()	บันทึกผลการวิเคราะห์ไว้ในแฟ้มที่มีนามสกุล .txt
การปรับเปลี่ยนตัวแปร	
round()	ปัดทศนิยม
replace()	แทนที่ค่าในตัวแปรใน () ตามเงื่อนไขที่กำหนด
ifelse()	ปรับเปลี่ยนค่าตัวแปรใน () ตามเงื่อนไขที่กำหนด
cut()	กำหนดจุดตัดของตัวแปรใน () ที่มีค่าต่อเนื่อง ณ จุดที่ระบุ
levels()	อธิบายรหัสตามที่กำหนด
การเลือกข้อมูล/สุ่มตัวอย่าง	
subset()	เลือกข้อมูลบางส่วนจากกรอบข้อมูล ตามเงื่อนไขที่ระบุ
sample()	เลือกตัวอย่างสุ่ม

2.2 การสร้างเวกเตอร์ (Vector) หรือ ตัวแปร (Variable)

เวกเตอร์ (หรือตัวแปร) มีหลายลักษณะ เช่น เป็นตัวเลข (numerical vector) ตัวอักษร (character vector) ตรรก (logical vector) การนำข้อมูลหลายๆค่ามาสร้างเวกเตอร์ต้องอาศัยฟังก์ชัน **c( )**

วิธีใช้

2.2.1 การสร้างเวกเตอร์ตัวเลข (numerical vector)

ตัวอย่าง 2.1 ถ้ามีตัวแปร 2 ตัวที่มีค่าดังต่อไปนี้ x = 5, 10, 15, 20, 25 และ y = 2,4,6,8

สร้าง vector (ตัวแปร) x และ y ได้ดังนี้

```
> x <- c(5,10,15,20,25)
> y<-c(2,4,6,8)
```

สร้าง vector (ตัวแปร) x y ที่มีค่า `0, 5, 10, 15, 20, 25,2,4,6,8`

x y

```
> xy<-c(0,x,y)
```

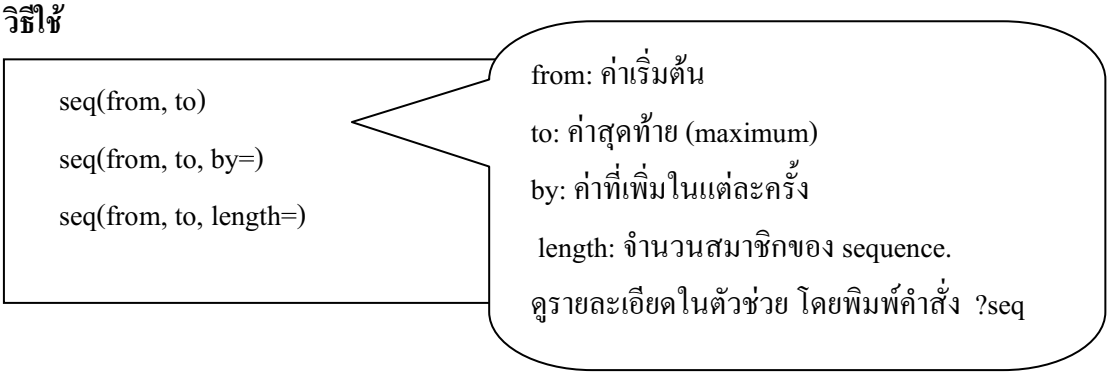
เรียกดูค่าของตัวแปร x และ xy

```
> x
[1] 5 10 15 20 25

> xy
 [1] 0  5 10 15 20 25  2  4  6  8
```

ถ้าพิมพ์เฉพาะคำสั่ง `c(5,10,15,20,25)` โดยไม่กำหนดชื่อ vector ที่จะเก็บค่าเหล่านี้ไว้ เราจะไม่สามารถนำผลลัพธ์จากคำสั่ง `c(5,10,15,20,25)` มาใช้งานในครั้งต่อไปได้

2.2.2 การสร้างเวกเตอร์เป็นตัวเลขที่เป็นระบบ โดยใช้ฟังก์ชัน seq()

วิธีใช้



ตัวอย่าง 2.2 สร้างตัวแปร มีค่าตั้งแต่ 1 ถึง 5 โดยเพิ่มทีละ 1
สามารถใช้คำสั่ง ดังต่อไปนี้ ซึ่งจะให้ผลลัพธ์เหมือนกัน

1) `x<-seq(from=1, to=5)` หรือ 2) `x<-seq(1,5)`
หรือ 3) `x<-seq(1:5)` หรือ 4) `x<- 1:5`

```
> x
[1] 1 2 3 4 5
```

ตัวอย่าง 2.3 สร้างตัวแปร มีค่าตั้งแต่ 1 ถึง 10 โดยเพิ่มทีละ2 หน่วย

```
> x<-seq(from=1, to=10,by=2)
```
หรือเขียนสั้นๆ `x<-seq(1,10,2)`

```
> x
[1] 1 3 5 7 9
```

ตัวอย่าง 2.4 สร้างตัวแปรมีค่าในช่วง a ถึง b โดยมีจำนวนสมาชิกเท่ากับค่าที่ระบุ (length)

```
> x<- seq(from=0, to=20, length=5)
[1]  0  5 10 15 20

> y<- seq(from=0, to=18, length=5)
[1]  0.0  4.5  9.0 13.5 18.0
```

2.2.3 การสร้างเวกเตอร์ที่มีค่าซ้ำ โดยใช้ฟังก์ชัน `rep()`

วิธีใช้

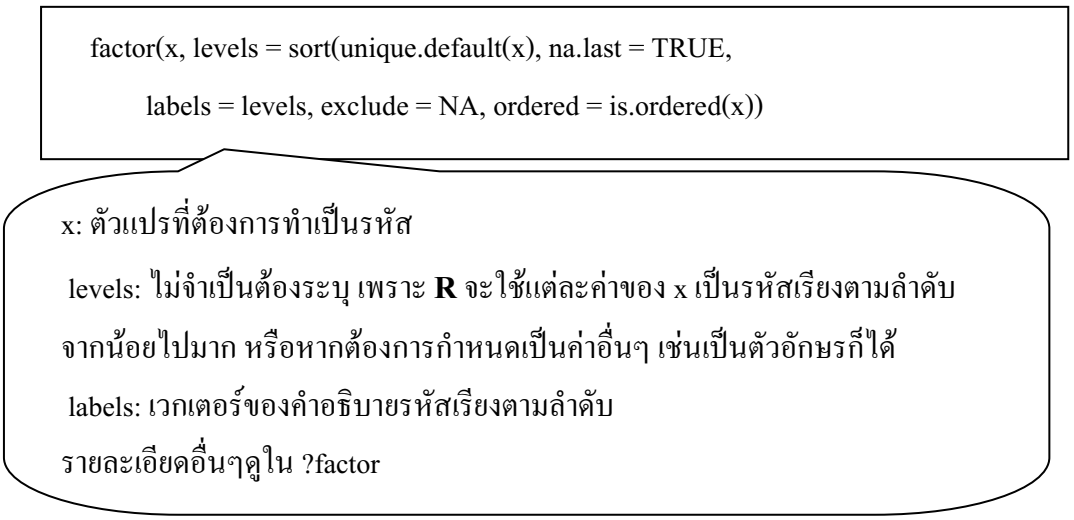
ตัวอย่าง 2.5 สร้างเวกเตอร์ขนาด 10 (มีสมาชิก 10 ตัว) ประกอบด้วยค่า 2 และ4 อย่างละ 5 ตัว

```
> x<-c(2,4)

> y<- rep(x,times=5)          # สร้างเป็นตัวแปร y จากการนำค่าของ x มาต่อกัน 5 ซ้ำ
 [1] 2 4 2 4 2 4 2 4 2 4

> rep(c(3,2),times=5)
 [1] 3 2 3 2 3 2 3 2 3 2

> rep(c(2,4),times=c(5,5))
 [1] 2 2 2 2 2 4 4 4 4 4

> rep(c(2,4),each = 5)
[1] 2 2 2 2 2 4 4 4 4 4
```

2.2.4 การสร้างเวกเตอร์เป็นรหัส (factor)

Factor เป็นคลาสย่อยของเวกเตอร์ตัวเลขที่มีคุณสมบัติพิเศษ คือตัวเลข แต่ละตัวจะเป็นรหัสแทนแต่ละกลุ่มของตัวแปร เช่น ตัวแปรเพศ ให้รหัส 1 แทนชาย และ 2 แทน หญิง สร้าง factor โดยใช้คำสั่ง factor

วิธีใช้



ตัวอย่าง2.6 เปลี่ยนตัวแปรเพศซึ่งเดิมกำหนดค่าเป็นตัวเลข (1,2) ให้เป็นรหัส พร้อมทั้งให้มีคำอธิบายรหัส

```
> sex<-rep(c(1,2),times=c(2,3))
> sex
[1] 1 1 2 2 2

> sex.f <-factor(sex)
> sex.f
[1] 1 1 2 2 2
Levels: 1 2

> levels(sex.f)<-c("male","female")

> sex.f
[1] male male female female female
Levels: male female

> attributes(sex.f)
```

TIP: รวบหลายขั้นตอนภายใต้คำสั่งเดียว

```
>  rm(sex.f)
sex.f<- factor(rep(c(1,2),
     times=c(2,3)),
     labels=c("male","female"))
```

ตรวจสอบว่า sex.f เป็น factor หรือไม่

```
> is.factor(sex.f)
[1] TRUE
```

2.2.5 การสร้างเวกเตอร์ตัวอักษร (character vector)

R จะรับรู้ตัวอักษรภายใต้เครื่องหมาย "" เช่น "1" เป็นตัวอักษรไม่ใช่ตัวเลข

```
> y <- c("m","m","f","f","f")
> y
[1] "m" "m" "f" "f" "f"
```

2.2.6 การเปลี่ยนและการตรวจสอบชนิดของเวกเตอร์

วิธีใช้

as.integer(x, ...)
as.factor(x,…)
as.numeric(x,…)

x เวกเตอร์ที่ต้องการเปลี่ยนให้เป็น integer หรือ factor หรือ numeric

is.integer(x, ...)
is.factor(x,…)
is.numeric(x,…)

x เวกเตอร์ที่ต้องการตรวจสอบว่าเป็น integer หรือ factor หรือ numeric หรือไม่

ตัวอย่าง 2.7

```
x<-c(1.2,2.8,3.0,4.1)
> xi<-as.integer(x)
> xi
[1] 1 2 3 4
```

เปลี่ยนตัวแปร x ให้มีค่าเป็นจำนวนเต็มเก็บไว้ในชื่อ xi

```
> x.f<-as.factor(xi)
> x.f
[1] 1 2 3 4
Levels: 1 2 3 4
> x.fac<-factor(xi)
> x.fac
[1] 1 2 3 4
Levels: 1 2 3 4
```

เปลี่ยนตัวแปร xi ให้เป็นรหัส โดยใช้คำสั่ง as.factor

หรือ factor ซึ่งให้ผลเหมือนกัน

```
> is.integer(x)
[1] FALSE
> is.integer(xi)
[1] TRUE
```

ตรวจสอบว่า ตัวแปรx หรือ xi มีค่าเป็นจำนวนเต็มหรือไม่

```
> is.factor(x.f)
[1] TRUE
> is.factor(x.fac)
[1] TRUE
```

ตรวจสอบว่า ตัวแปร `x.f` หรือ `x.fac` มีค่าเป็นรหัสหรือไม่

2.3 การสร้างแฟ้มข้อมูล (Data File) หรือ กรอบข้อมูล (Data frame)

แฟ้มข้อมูลใน **R** จะมีลักษณะเป็น กรอบข้อมูลซึ่งประกอบด้วยตัวแปร (เวกเตอร์) หลายๆตัว ที่ทุกตัวมีความยาวเท่ากัน หรือ ประกอบด้วยเมทริกซ์ก็ได้ โดยแต่ละสดมภ์ (column) คือ หนึ่งตัวแปร และแต่ละแถว (row) คือลักษณะต่างๆของหนึ่ง case

2.3.1 สร้างแฟ้มข้อมูล (กรอบข้อมูล) ด้วยฟังก์ชัน data.frame ()

วิธีใช้

```
data.frame(..., row.names = NULL, check.rows = FALSE,check.names = TRUE)
```

... วัตถุหรือเวกเตอร์ที่ต้องการนำมารวมเป็นกรอบข้อมูล(แฟ้มข้อมูล)
row.names:เวกเตอร์ที่บอกชื่อแถวของกรอบข้อมูล แต่ไม่ต้องระบุก็ได้
ดูรายละเอียดจาก ?data.frame

ตัวอย่าง2. 8 สร้างกรอบข้อมูล

```
> x<-c(1,2,3,4)
> s<-c("m","m","f","f")
> xs.dat<-data.frame(x,s)
> xs.dat
1 1 m
2 2 m
3 3 f
4 4 f

> attributes(xs.dat)
$names
[1] "x" "s"

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

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

ตัวอย่าง2. 9 สร้างกรอบข้อมูลและกำหนดชื่อ row

```
> rname<-c("one","two","three","four")
> xs.dat<-data.frame(x,s,
  row.names = rname)
> xs.dat
       x s
  one  1 m
  two  2 m
  three 3 f
  four  4 f
```