```
7      David Beckham
8      Alan  Smith
9      Cesc  Fabregas
10     James McFadden
```

จะเห็นว่าแต่ละสดมภ์ถูกคั่นด้วย tab หรือ {ระยะกั้น} นั่นเอง แต่จะเห็นว่าสดมภ์เรียงไม่สม่ำเสมอก็เนื่องจากข้อความสั้นยาวไม่เท่ากัน ทำให้ระยะของ tab ไม่เท่ากันนั่นเอง

ด้วยวิธีเดียวกันนี้ ลองทำดูด้วยตัวเองเพื่อเรียนรู้ว่าการกั้นด้วยช่องว่าง และแบบความกว้างสดมภ์คงที่มีลักษณะแฟ้มข้อมูลเป็นเช่นไร

เมื่อทราบลักษณะโครงสร้างแฟ้มข้อมูลแล้ว ต่อไปนี้มาดูวิธีอ่านข้อมูลเหล่านั้นเข้ามาในสภาพแวดล้อม **R** กันต่อไป ในการอ่านแฟ้มข้อมูลที่เป็นข้อความ (text) เช่นนี้ จะใช้คำสั่ง `read.table()` ซึ่งมีรูปแบบการออกคำสั่งและ argument ที่สำคัญดังนี้ (ละ argument อื่นๆ ที่ไม่จำเป็นไว้ก่อน เนื่องจากแฟ้มข้อมูลโดยทั่วไปจะไม่ได้ใช้)

```
read.table(file, header = FALSE, sep = "")
```

ลำดับแรกภายในวงเล็บได้แก่ชื่อแฟ้มข้อมูล ซึ่งจะต้องใส่ไว้ในเครื่องหมาย `""` เสมอ ลำดับที่สองเป็นการบอกว่าภายในแฟ้มข้อมูลนั้นมีบรรทัด header หรือไม่ ในกรณีตัวอย่างข้างบนเราได้ใส่บรรทัด header คือชื่อสดมภ์ไว้ในบรรทัดที่ 1 ดังนั้นในกรณีนี้ `header` จึงมีค่าเป็น `TRUE` และ `sep` คือตัวคั่น ซึ่งจะกำหนดให้เป็น , สำหรับแฟ้มข้อมูลชนิด CSV และ `\t` (เครื่องหมายสำหรับ tab) สำหรับแฟ้มข้อมูลชนิด tab delimited ลองมาดูการอ่านแฟ้มข้อมูลชนิด CSV ดังนี้

```
> setwd("C:/Rworkplace")
> dat <- read.table("football.csv",header=TRUE,sep=",")
> dat
   Order     Name  Lastname
1      1  Freddie Ljungberg
2      2      Joe      Cole
3      3   Steven   Gerrard
4      4 Cristian   Ronaldo
5      5   Ashley      Cole
6      6    David   Bentley
7      7    David   Beckham
8      8     Alan     Smith
9      9     Cesc  Fabregas
10    10    James  McFadden
```

ในบรรทัดแรกนั้น `setwd("C:/Rworkplace")` เป็นการบอกว่าแฟ้มข้อมูลอยู่ในโฟลเดอร์ C:/Rworkplace ซึ่งผู้เขียนสร้างไว้เพื่อเก็บแฟ้มข้อมูล บรรทัดถัดมาเป็นการอ่านข้อมูลในแฟ้มที่มีรูปแบบเป็น CSV ซึ่งมี `header` ด้วย และในกรณีนี้ `sep` จะต้องระบุเป็น `","` (ต้องเขียนไว้ในเครื่องหมายคำพูดด้วย) อย่าลืมว่าเราต้องส่งข้อมูลมาเก็บไว้ในวัตถุซึ่งในที่นี้ตั้งชื่อให้ว่า `dat` เมื่อเรียกดู `dat` ก็จะเห็นข้อมูลที่อ่านเข้ามาได้

ถ้าเราจะอ่านข้อมูลจากแฟ้มชื่อ football.tab ก็จะต้องกำหนดตัวเลือก argument ให้สอดคล้องกับรูปแบบของแฟ้มดังนี้

```
> dat <- read.table("football.tab",header=TRUE,sep="\t")
```

หมายเหตุ ผู้เขียนนิยมเขียนคำสั่งระบุโฟลเดอร์ของข้อมูล `setwd()` เมื่อเริ่มทำงานกับแฟ้มข้อมูลภายนอกเสมอ เพื่อให้แน่ใจว่า **R** จะอ่านหรือบันทึกแฟ้มข้อมูลในโฟลเดอร์ที่ต้องการ จึงขอแนะนำให้ผู้ใช้ทุกคนทำตามนี้ด้วยเช่นกัน ในกรณีเช่นนี้ เราอาจใส่แฟ้มที่ต้องการไว้ในโฟลเดอร์ใดเพื่อจัดกลุ่มข้อมูลตามความถนัดของผู้ใช้ก็ได้

สำหรับระบบปฏิบัติการแมคอินทอช OSX และลินุกซ์ การเรียกโฟลเดอร์จะต่างออกไปจากระบบปฏิบัติการวินโดวส์ การใช้คำสั่ง `setwd()` จึงต้องเรียกโฟลเดอร์ตามระบบปฏิบัติการนั้นๆ เช่น `setwd("~/Rworkplace")`

แฟ้มข้อมูลของโปรแกรมสถิติอื่นๆ เช่น EpiData, Stata, SPSS

มีบ่อยครั้งที่ผู้ใช้งานโปรแกรม **R** ต้องอ่านข้อมูลที่สร้างจากโปรแกรมสถิติอื่น หรือแม้แต่โปรแกรม **EpiData** ที่กล่าวถึงในบทที่ 2 เรื่อง **สภาพแวดล้อมเสริมการทำงาน** คำสั่งสำหรับอ่านแฟ้มที่มีรูปแบบพิเศษเหล่านี้ อยู่ใน library ชื่อ foreign ซึ่งจะติดตั้งให้แล้วเมื่อติดตั้ง **R** ดังนั้นในการเรียกใช้งานก็เพียงแต่ออกคำสั่งเรียก library foreign มาใช้งาน แล้วออกคำสั่ง `read.epiinfo()`, `read.dta()`, หรือ `read.spss()` มาใช้งานได้เลย ดังตัวอย่าง

```
> library(foreign)
> setwd("C:/Rworkplace")
> dat1 <- read.epiinfo("oral.rec")
> str(dat1)
`data.frame':   160 obs. of  9 variables:
 $ STATUS    : num  1 1 1 1 1 1 1 1 1 1 ...
 $ ADDRESS   : num  1 1 2 2 3 2 1 4 1 1 ...
 $ DRINKINGST: num  1 1 1 1 1 1 2 2 3 1 ...
 $ AGE       : num  59 59 59 61 62 71 76 77 78 64 ...
 $ SEX       : num  1 1 1 1 1 2 2 2 2 1 ...
 $ BETEL     : num  2 2 2 2 2 1 1 1 2 2 ...
 $ SMOKINGSTA: num  1 1 1 1 1 1 2 1 1 1 ...
 $ VETGETABLE: num  3 2 1 5 3 5 4 5 3 2 ...
 $ FRUIT     : num  2 2 5 2 3 2 3 4 2 2 ...
 - attr(*, "prompts")= Named chr  "STATUS" "ADDRESS" "DrinkingStatu$
  ..- attr(*, "names")= chr  "STATUS" "ADDRESS" "DRINKINGST" "AGE" $
>
> dat2 <- read.dta("oral.dta")
> str(dat2)
`data.frame':   160 obs. of  9 variables:
```

```
 $ status  : num  1 1 1 1 1 1 1 1 1 1 ...
 $ address : num  1 1 2 2 3 2 1 4 1 1 ...
 $ drinking: num  1 1 1 1 1 1 2 2 3 1 ...
 $ age     : num  59 59 59 61 62 71 76 77 78 64 ...
 $ sex     : num  1 1 1 1 1 2 2 2 2 1 ...
 $ betel   : num  2 2 2 2 2 1 1 1 2 2 ...
 $ smokings: num  1 1 1 1 1 1 2 1 1 1 ...
 $ vetgetab: num  3 2 1 5 3 5 4 5 3 2 ...
 $ fruit   : num  2 2 5 2 3 2 3 4 2 2 ...
 - attr(*, "datalabel")= chr "Data file created by EpiData based on$
 - attr(*, "time.stamp")= chr "10 April 2006 16:"
 - attr(*, "formats")= chr  "%16.0f" "%16.0f" "%16.0f" "%16.0f" ...
 - attr(*, "types")= int  100 100 100 100 100 100 100 100 100
 - attr(*, "val.labels")= chr  "" "" "" "" ...
 - attr(*, "var.labels")= chr  "STATUS" "ADDRESS" "DrinkingStatus" $
 - attr(*, "version")= int 6
 - attr(*, "label.table")=List of 9
```

ตัวอย่างข้างต้นเป็นการอ่านแฟ้มข้อมูลของ **EpiData** ที่มีนามสกุล REC และแฟ้มข้อมูลของโปรแกรม **Stata** ที่มีนามสกุล DTA จากแฟ้มชื่อ ORAL.REC และ ORAL.DTA ซึ่งมีข้อมูลเหมือนกันเข้าสู่วัตถุชนิดกรอบข้อมูลชื่อ dat1 และ dat2 ตามลำดับ และเช่นเดียวกัน เราสามารถอ่านแฟ้มข้อมูลของโปรแกรม **SPSS** ที่มีนามสกุล SAV ได้ด้วยวิธีเดียวกัน สังเกตว่าในการอ่านข้อมูลเข้ามานั้น ต้องระบุว่าจะใส่ข้อมูลในกรอบข้อมูลชื่ออะไรด้วย มิฉะนั้นข้อมูลที่อ่านเข้ามาได้จะไม่สามารถเรียกใช้งานได้ และถูกลบไปจากหน่วยความจำ

แฟ้มข้อมูลชนิด DTA ของโปรแกรม **Stata** มีข้อดีคือสามารถเก็บข้อความอธิบายข้อมูล (label) ของข้อมูลต่างๆ และเรียกมาใช้งานใน **R** ได้ เช่น ตัวแปร sex เราอาจระบุให้ 1 = male และ 2 = female การระบุข้อความอธิบายเช่นนี้สามารถบันทึกไว้ได้ในแฟ้ม DTA และเรียกมาใช้ได้ในสภาพแวดล้อม **R**

นอกจากนี้ **R** ยังสามารถอ่านแฟ้มข้อมูลของโปรแกรม **Minitab** ข้อมูลในรูปแบบ DBF และข้อมูลของโปรแกรม **SAS** ได้อีกด้วย

แฟ้มข้อมูล ODBC

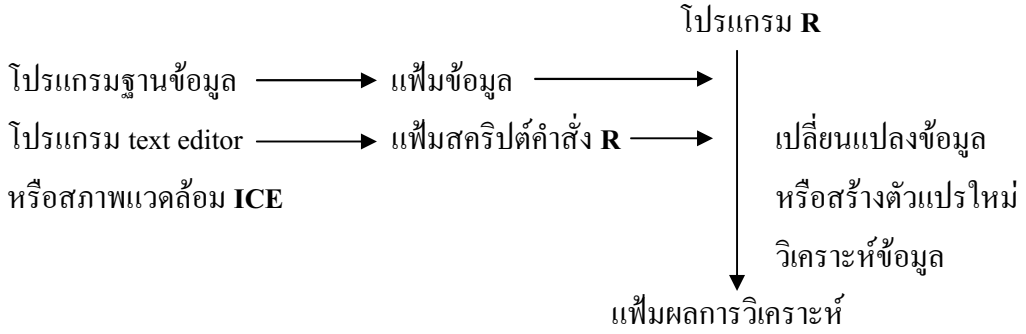
ไม่แนะนำให้เรียกใช้ข้อมูลที่บันทึกในรูปแบบของ ODBC โดยตรง โปรแกรมที่บันทึกข้อมูลแบบ ODBC มีหลายโปรแกรมเช่น **Excel**, **Access** และฐานข้อมูลแบบ SQL เช่นฐานข้อมูลของโปรแกรม **MySQL** การเรียกใช้ข้อมูลในรูปแบบนี้โดยตรงจะต้องใช้ library RODBC และมีขั้นตอนการทำงานที่ซับซ้อนพอสมควรซึ่งจะไม่กล่าวในที่นี้ ผู้สนใจอาจดาวน์โหลด library นี้ได้จาก **CRAN** และศึกษาการทำงานด้วยตนเอง

วิธีการที่แนะนำหากมีข้อมูลในลักษณะนี้อยู่ คือให้ถ่ายข้อมูลออกเป็นรูปแบบ text ก่อนแล้วจึงอ่านเข้าสภาพแวดล้อม **R** ด้วยคำสั่ง `read.table()` ดังที่กล่าวข้างต้น

หมายเหตุ ดังที่กล่าวแล้วว่าเราสามารถเรียกดูความช่วยเหลือของฟังก์ชันใน library ต่างๆ ได้ด้วยคำสั่ง `help.start()` ซึ่ง **R** จะเรียกโปรแกรม web browser ขึ้นมาแสดงความช่วยเหลือโดยเลือกตัวเลือก package แล้วจึงเลือก library ที่ต้องการ และเลือกฟังก์ชันที่ต้องการ ฟังก์ชันที่กล่าวมาแล้วข้างต้นนี้มี argument ให้เลือกตั้งค่าได้อีกมาก ผู้สนใจสามารถดูรายละเอียดในความช่วยเหลือนั้นๆ

การบันทึกแฟ้มข้อมูลแบบต่างๆ

โดยทั่วไปในการทำงานวิเคราะห์ข้อมูล ไม่แนะนำให้บันทึกแฟ้มข้อมูลจากหน่วยความจำลงสู่ดิสก์อีกครั้ง วิธีที่แนะนำให้ทำคือเตรียมข้อมูลให้ถูกต้องเสียก่อน จากนั้นเขียนแฟ้มสคริปต์เพื่อสร้างตัวแปรใหม่หรือปรับเปลี่ยนข้อมูลบางอย่าง วิเคราะห์ข้อมูล แล้วบันทึกผลการวิเคราะห์เป็นขั้นสุดท้าย ดังแผนผังนี้



รูปที่ 5.4 แผนผังการวิเคราะห์ข้อมูล

จากแผนผังนี้จะเห็นว่าเราจะสร้างแฟ้มข้อมูลด้วยโปรแกรมฐานข้อมูลต่างๆ ดังที่แนะนำมาแล้วข้างต้น จากนั้นใช้โปรแกรม text editor เช่น **Tinn-R** เขียนแฟ้มสคริปต์คำสั่งเพื่อเปลี่ยนแปลงข้อมูลหรือสร้างตัวแปรใหม่ และวิเคราะห์ข้อมูล และจบที่การบันทึกแฟ้มผลการวิเคราะห์ข้อมูลในที่สุด แต่ในสภาพแวดล้อม **ICE** เราสามารถออกคำสั่งเพื่อวิเคราะห์ข้อมูลแล้วจึงบันทึกสคริปต์คำสั่งเป็นแฟ้มได้

จะเห็นว่าโดยปกติแล้วเราไม่จำเป็นต้องบันทึกกรอบข้อมูลในโปรแกรม **R** กลับลงเป็นแฟ้มข้อมูลแต่อย่างใด แต่เราจะใช้สคริปต์คำสั่งของ **R** ในการเปลี่ยนแปลงข้อมูลหรือสร้างตัวแปรใหม่ ข้อดีของการทำงานในลักษณะนี้คือตัวแฟ้มข้อมูลจะตรงกับแบบฟอร์มเก็บข้อมูล ไม่มีรหัสใดเปลี่ยนไป หรือไม่มีตัวแปรใดขาดหรือเพิ่มขึ้นมา และขั้นตอนต่างๆ ในการเปลี่ยนแปลงข้อมูลหรือสร้างตัวแปรใหม่อยู่ในแฟ้มข้อมูลให้ตรวจสอบในภายหลังได้ และหากพบว่าผิดพลาดหรือต้องการเปลี่ยนแปลงการวิเคราะห์ใดๆ ก็เพียงแต่แก้ไขแฟ้มสคริปต์ แล้วทำงานกระบวนการทั้งหมดใหม่อีกครั้ง ก็จะได้ผลลัพธ์ใหม่ทันที

การทำงานในลักษณะนี้มีความสำคัญมากเพื่อให้ทุกขั้นสามารถตรวจสอบได้ ทั้งโดยตัวนักวิจัยเอง ผู้ควบคุมการวิจัย และแหล่งทุน และเมื่อมีการเปลี่ยนแปลงใดๆ ก็สามารถวิเคราะห์ซ้ำได้

แต่ในบางกรณีก็อาจมีความจำเป็นที่จะต้องบันทึกการเปลี่ยนในแฟ้มข้อมูลลงเป็นแฟ้ม ในกรณีที่ต้องการบันทึกแฟ้มข้อมูลเช่นนี้ ขอให้บันทึกด้วยชื่อที่ต่างจากแฟ้มข้อมูลเดิม เพื่อจะได้เก็บหลักฐานการกรอกข้อมูลดั้งเดิมไว้เป็นหลักฐาน คำสั่งในการบันทึกข้อมูลลงแฟ้มได้แก่คำสั่ง `write.table()`,`write.dta()` และอื่นๆ

แฟ้มข้อมูลแบบ text ในรูปแบบต่างๆ

อย่างที่กล่าวข้างต้นแล้วในเรื่องการอ่านข้อมูลว่าข้อมูลแบบ text ยังมีหลายรูปแบบ วิธีการออกคำสั่ง `write.table()` ก็มีตัวเลือกเพื่อตอบสนองการบันทึกในรูปแบบต่างๆ เหล่านั้น ลองบันทึกข้อมูลนักฟุตบอลข้างต้น ดังตัวอย่างต่อไปนี้

```
> setwd("C:/Rworkplace")
> dat <- read.table("football.csv",header=TRUE,sep=",")
> dat
...
> write.table(dat,"football2.csv",sep=",")
> dat2 <- read.table("football2.csv",header=TRUE,sep=",")
> dat2
...
> write.table(dat,"football3.txt",sep="\t")
> dat3 <- read.table("football3.txt",header=TRUE,sep="\t")
> dat3
...
```

แฟ้มข้อมูลของโปรแกรม Stata

คำสั่งในการบันทึกแฟ้มข้อมูลแบบ DTA ของโปรแกรม **Stata** ได้แก่คำสั่ง `write.dta()` ใน library foreign การออกคำสั่งก็คล้ายกับคำสั่ง `write.table()` ดังข้างต้น แต่จำได้ง่ายกว่า

มากเนื่องจากไม่จำเป็นต้องระบุตัวเลือกใดๆ ทั้งในการอ่านและเขียน เนื่องจากมีรูปแบบชัดเจนอยู่แล้ว ด้วยข้อดีหลายประการทั้งในความง่ายในการเรียกใช้และความสามารถในการเก็บข้อความอธิบายข้อมูล (label) จึงแนะนำให้ใช้รูปแบบนี้ในการบันทึกแฟ้มข้อมูล

```
> library(foreign)
> write.dta(dat,"football4.dta")
> dat4 <- read.dta("football4.dta")
> dat4
...
```

การบันทึกสภาพแวดล้อมของ R ลงแฟ้มและการเรียกกลับคืน

สภาพแวดล้อมของ **R** ในที่นี้หมายถึงวัตถุต่างๆ และบรรทัดคำสั่งที่ให้ **R** ทำงาน คำสั่งในการบันทึกวัตถุต่างๆ ลงแฟ้มข้อมูลคือคำสั่ง `save()` และ `save.image()` ทั้งสองคำสั่งนี้ต่างจากคำสั่ง `write.table()` และ `write.dta()` ตรงที่ `save()` และ `save.image()` สามารถบันทึกวัตถุชนิดต่างๆ ทุกชนิดได้ ไม่จำเพาะต้องเป็นกรอบข้อมูล และสามารถบันทึกวัตถุจำนวนมากลงในแฟ้มเดียวกันได้ โดยที่คำสั่ง `save.image()` เป็นกรณีพิเศษที่บันทึกวัตถุทั้งหมดที่มีอยู่ในหน่วยความจำของ **R** ในขณะนั้นลงแฟ้ม แม้ว่าไม่มีข้อจำกัดในการระบุนามสกุล แต่เพื่อความจำเพาะและง่ายต่อการจดจำ นิยมให้แฟ้มบันทึกวัตถุของ **R** มีนามสกุลเป็น Rdata เสมอ

แต่ในบางกรณีเราอาจต้องการบันทึกบรรทัดคำสั่งต่างๆ ที่ได้ทำไปแล้ว เพื่อไว้ใช้ทำงานวิเคราะห์ข้อมูลต่างๆ ซ้ำอีก โปรแกรม **R** ได้สร้างคำสั่งไว้ให้สำหรับการทำงานเช่นนี้ไว้ คือคำสั่ง `savehistory()` ซึ่งนิยมให้นามสกุลของแฟ้มเป็น Rhistory ลองมาดูวิธีใช้งานคำสั่งเหล่านี้ดังต่อไปนี้

```
> rm(list=ls()
> setwd("C:/Rworkplace")
> x <- letters[1:10]
> y <- 1:10
> z <- log(1:10)
> dat <- data.frame(x,y,z)
> save(x,y,z, file="test1.Rdata")
> save.image("test2.Rdata")
> savehistory("test.Rhistory")
```

ลองพิมพ์บรรทัดคำสั่งข้างต้น แล้วเปิดดูแฟ้ม test1.Rdata, test2.Rdata และ test.Rhistory เพื่อศึกษาการทำงานของคำสั่งเหล่านั้น มีข้อพึงระวังอยู่อย่างหนึ่งคือคำสั่ง `savehistory()` จะบันทึกคำสั่งทุกคำสั่งบน R console แม้ว่าคำสั่งนั้นจะทำงานผิดพลาดก็ตาม ดังนั้นเมื่อเรียกแฟ้มนั้นมาทำงานซ้ำอีกโดยไม่แก้ไขใดๆ ก็จะเกิดความผิดพลาดที่บรรทัดคำสั่งนั้นอีก ในที่นี้จึงแนะนำว่าเมื่อบันทึกแล้ว ให้เปิดแฟ้มขึ้นมาแล้วลบบรรทัดคำสั่งที่ทำงานไม่ถูกต้องหรือไม่ต้องการทิ้งเสีย เมื่อทำงานแฟ้มสคริปต์นั้นซ้ำอีกก็จะไม่เกิดข้อผิดพลาดในการทำงาน

แฟ้มสคริปต์คำสั่ง R

ในที่นี้ได้กล่าวถึงแฟ้มสคริปต์คำสั่ง **R** เป็นครั้งคราว แต่ยังไม่ได้กล่าวถึงประโยชน์และการใช้งานให้ชัดเจน จึงจะกล่าวถึงเสียในที่นี้ เนื่องจากต่อไปจะแนะนำให้ทำแฟ้มสคริปต์คำสั่ง **R** แทนการพิมพ์คำสั่งบน R console อย่างที่ได้กล่าวถึงในตอนต้นๆ

ดังตัวอย่างข้างต้น แทนที่จะต้องพิมพ์คำสั่งทีละบรรทัดบนหน้าจอ R console ก็สามารถพิมพ์และบันทึกไว้เป็นแฟ้มสคริปต์คำสั่ง **R** โดยใช้โปรแกรม text editor เช่น **Crimson Editor**, **jEdit** หรือโปรแกรมอื่นๆ ก็ได้ โดยไม่ต้องพิมพ์เครื่องหมาย prompt > นำหน้าบรรทัด ดังนี้

```
rm(list=ls()
setwd("C:/Rworkplace")
x <- letters[1:10]
y <- 1:10
z <- log(1:10)
dat <- data.frame(x,y,z)
```

จากนั้นบันทึกลงดิสก์ด้วยชื่ออะไรก็ได้ตามต้องการ แต่ให้ใช้นามสกุล R เพื่อจะได้ทราบว่าเป็นแฟ้มสคริปต์คำสั่ง **R** นั่นเอง เช่น ตั้งชื่อว่า test1.R และบันทึกในโฟลเดอร์ C:/Rworkplace (หรือใน ~/Rworkplace สำหรับระบบปฏิบัติการแมคอินทอช OSX และลินุกซ์) แล้วเรียกคำสั่งในแฟ้มนี้ทำงานใน **R** โดยใช้คำสั่ง `source()` โดยพิมพ์บนหน้าจอ R console ดังนี้

```
> setwd("C:/Rworkplace")
> source("test.R")
```

หมายเหตุ ความจริงแล้วในหน้าต่าง ice.commands ก็สามารถบันทึกบรรทัดคำสั่งเข้าสู่แฟ้มสคริปต์ได้เช่นกัน แต่จะต้องตั้งค่าในโมดุล ice.main ก่อน ซึ่งจะได้กล่าวถึงต่อไป

บทที่ 6 โมดุล **ice.main** ในการจัดการแฟ้มข้อมูลและวัตถุ

จัดการ working directory และ workspace

ทำงานแฟ้มสคริปต์ **R**

อ่านและเขียนแฟ้มข้อมูล

บันทึกคำสั่งลงแฟ้มสคริปต์

sink ผลลัพธ์ลงแฟ้ม

ดูวัตถุในหน่วยความจำและเส้นทางค้นหา

แสดงโครงสร้างของวัตถุ

แก้ไขวัตถุ

กำจัดวัตถุ

เปลี่ยนชื่อวัตถุ

ตัวอย่างชุดข้อมูล

ตั้งค่าตัวเลือก

ความช่วยเหลือ

ที่ผ่านมาข้างต้นเป็นการทำงานโดยการพิมพ์ข้อความลงบน R console หรือพิมพ์ลงใน text editor แล้วเรียกทำงาน ทั้งหมดนี้ยังไม่ได้ใช้ความสามารถของสภาพแวดล้อม **ICE** แต่อย่างใดเลย ในหัวข้อนี้จะได้เริ่มกล่าวถึงการใช้งานสภาพแวดล้อม **R-ICE** และต่อไปในหนังสือนี้จะทำงานกับสภาพแวดล้อม **R-ICE** เป็นหลัก

ลองมาดูกันที่โมดุล ice.main ซึ่งเป็นโมดุลหลักของ **R-ICE** เมนูของโมดุลนี้เป็นดังตารางที่ 6.1 ซึ่งเป็นตารางเดียวกับตารางที่ 3.2 นั่นเอง

ตารางที่ 6.1 โครงสร้างของรายการเมนู ice.main

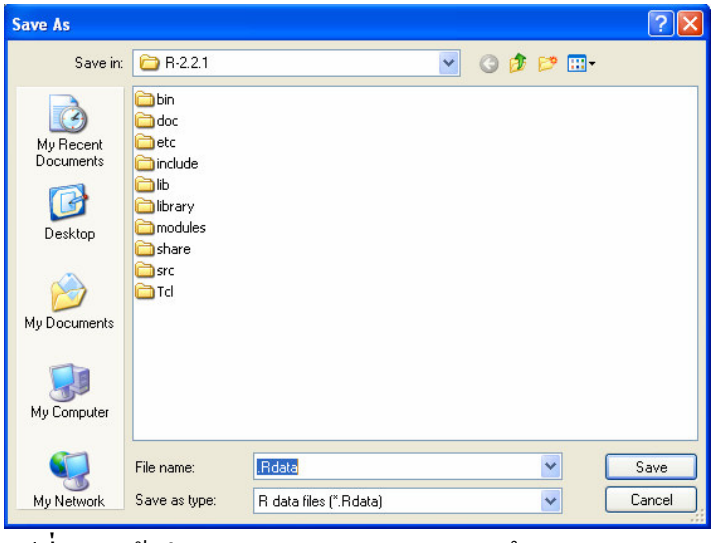
แฟ้ม	**วัตถุ**	**ตั้งค่า**	**ช่วยเหลือ**
ดู working directory	ลิสต์วัตถุในหน่วยความจำ	ตัวเลือกทั่วไป	ตำสั่ง
เปลี่ยน working directory	ค้นหาวัตถุใน search path		do
เรียกใช้ workspace	attach กรอบข้อมูล		fread
บันทึก workspace	detach กรอบข้อมูล		fwrite
เรียกใช้ history	แสดงโครงสร้างของวัตถุ		logg
บันทึก history	แก้ไขวัตถุ		output
แฟ้มสคริปต์	แก้ไขกรอบข้อมูล		เกี่ยวกับสภาพแวดล้อม ICE
สร้าง…	แก้ไขวัตถุชนิดอื่น		text
เปิด…	กำจัดวัตถุ		html
Sink ผลลัพธ์ลงแฟ้ม	กำจัดวัตถุทั้งหมด		chm
สร้างใหม่…	กำจัดวัตถุที่เลือก		**เกี่ยวกับ package ICE**
ต่อท้ายแฟ้ม…	เปลี่ยนชื่อวัตถุ		
หยุดการ sink	ตัวอย่างชุดข้อมูล		
ทำงานแฟ้มสคริปต์ R	ahcc		
สร้างแฟ้ม output จากแฟ้มสคริปต์ R	ancdata		
อ่านแฟ้มข้อมูลทุกชนิด	cca		
อ่านแฟ้มข้อมูลจำเพาะ	cca9		
แฟ้ม text	cca.match		
แฟ้ม Epi Info	ccan		
แฟ้ม dbf	evans		
แฟ้ม Stata	irdata		
แฟ้ม SPSS	mccdata		
เขียนแฟ้มข้อมูลทุกชนิด	vct		
เขียนแฟ้มข้อมูลจำเพาะ			
แฟ้ม text			
แฟ้ม dbf			
แฟ้ม Stata			
เลิกงาน			

จัดการ working directory และ workspace

รายการแรกคือ “ดู working directory” ภายใต้เมนูหลัก “แฟ้ม” ลองเลือกรายการนี้ดู จะเห็น prompt ที่ R console เปลี่ยนเป็น `ICE>>` และมีการเรียกฟังก์ชัน `dir()` และแสดงผลของฟังก์ชันดังนี้

```
ICE>> dir()
 [1] "AUTHORS"           "bin"               "CHANGES"
 [4] "COPYING"           "COPYING.LIB"       "COPYRIGHTS"
 [7] "doc"               " etc "             "FAQ"
...
```

รายการถัดไปคือ “เปลี่ยน working directory” ซึ่งจะเรียกใช้ฟังก์ชัน `setwd()` และรายการต่อมาเป็น “เรียกใช้ workspace” ซึ่งจะเรียกใช้ฟังก์ชัน `load()` รายการต่อไปเป็น “บันทึก workspace” ซึ่งจะเรียกฟังก์ชัน `save.image()` โดยมีหน้าต่างให้เลือกโฟลเดอร์และตั้งชื่อแฟ้มดังรูปที่ 6.1 และรายการอื่นๆ ก็จะทำงานในลักษณะเดียวกันนี้นั่นเอง



รูปที่ 6.1 หน้าต่าง Save As ของรายการ “บันทึก workspace”

ทำงานแฟ้มสคริปต์ R

รายการที่จะกล่าวต่อไปคือ “ทำงานแฟ้มสคริปต์ **R**” เป็นการสั่งทำงานแฟ้มสคริปต์ **R** ที่บันทึกไว้ ดังที่กล่าวก่อนหน้านี้แล้วว่าวิธีทำงานที่แนะนำคือให้สร้างแฟ้มสคริปต์คำสั่ง **R** เป็นแฟ้มแล้วเรียกใช้งานโดยคำสั่ง `source()` รายการนี้ก็ทำงานเช่นเดียวกัน แต่จะนำชื่อของแฟ้มมาเป็น prompt ด้วยเพื่อให้ทราบว่าคำสั่งเหล่านั้นมาจากแฟ้มอะไร ส่วนรายการ “สร้างแฟ้ม output จาก

แฟ้มสคริปต์ **R**" นั้นทำงานคล้ายกัน แต่จะส่งผลลัพธ์บันทึกลงแฟ้มและบอกวัน เวลา และรุ่นของ **R** ที่สร้างผลลัพธ์นั้นด้วย ดังตัวอย่าง

```
Output generated on: Wed Apr 12 14:28:09 2006
R version 2.2.1, 2006-04-03
...
```

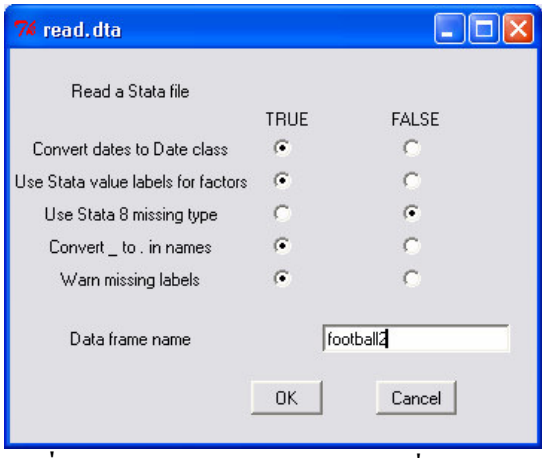
ในที่นี้แนะนำให้ตั้งชื่อเป็นชื่อเดิมของแฟ้มสคริปต์นั้น แต่เปลี่ยนไปใช้นามสกุล log หรือ out แล้วแต่ความชอบของผู้ใช้

อ่านและเขียนแฟ้มข้อมูล

กลุ่มรายการต่อไปเป็นกลุ่มรายการอ่านและเขียนแฟ้มข้อมูล ซึ่งก็คือกลุ่มคำสั่ง `read` และ `write` นั่นเอง แต่จะมีสิ่งที่น่าสนใจอยู่บางประการ รายการ "อ่านแฟ้มข้อมูลทุกชนิด" จะเลือกใช้คำสั่งที่จำเพาะต่อนามสกุลของแฟ้มข้อมูลนั้นๆ โดยใช้ค่าปริยายทั้งหมด และจะส่งผลลัพธ์เป็นกรอบข้อมูลที่มีชื่อเดียวกันกับชื่อของแฟ้มข้อมูลนั้นๆ ดังตัวอย่างการเรียกใช้แฟ้ม football.csv ดังนี้

```
ICE>> football <- read.csv("C:/Rworkplace/football.csv")
```

ice.main จะเลือกใช้คำสั่ง `read.csv()` ในการอ่านแฟ้มข้อมูล และตั้งชื่อกรอบข้อมูลผลลัพธ์เป็น `football` แต่หากการอ่านแฟ้มข้อมูลมีความผิดพลาดเนื่องจากจำเป็นต้องตั้งค่าตัวเลือก argument ใดเป็นพิเศษ ให้ใช้ตัวเลือก "อ่านแฟ้มข้อมูลจำเพาะ" ซึ่งจะมีรายการให้เลือกต่อไปว่าจะอ่านแฟ้มข้อมูลชนิดใด และจะมีหน้าต่างตัวเลือกที่จำเพาะสำหรับแฟ้มชนิดนั้นๆ ให้เลือกตั้งค่าได้ตามต้องการ



รูปที่ 6.2 หน้าต่างตัวเลือกสำหรับคำสั่ง `read.dta()`

ในการบันทึกแฟ้มข้อมูลก็เช่นกัน โดยทั่วไปให้ใช้รายการ “เขียนแฟ้มข้อมูลทุกชนิด” ไว้ก่อน แต่หากต้องการบันทึกโดยใช้ตัวเลือกจำเพาะก็จึงใช้รายการ “เขียนแฟ้มข้อมูลจำเพาะ”

ส่วนรายการ “แฟ้มสคริปต์” เป็นการสร้างใหม่หรือเปิดแฟ้มสคริปต์ที่มีอยู่แล้วเพื่อบันทึกบรรทัดคำสั่งที่สร้างขึ้นโดยรายการแต่ละรายงานในสภาพแวดล้อม **R-ICE** ทั้งหมดทุกโมดุลลงต่อในแฟ้มสคริปต์เดียวกัน คือที่กำหนดโดยรายการนี้ เมื่อเลือกรายการนี้แล้วจะมีหน้าต่างให้เลือกโฟลเดอร์และตั้งชื่อหรือเลือกแฟ้มสคริปต์ที่ต้องการ รายการนี้จะไม่มีอะไรแสดงบน R console เนื่องจากเป็นการสร้างหรือเปิดแฟ้มเท่านั้น หลังจากนี้เมื่อทำรายการใด เช่น อ่านแฟ้มข้อมูลโดยรายการ “อ่านแฟ้มข้อมูลทุกชนิด” ดังตัวอย่างนี้

```
ICE>> football <- read.csv("C:/Rworkplace/football.csv")
```

บันทึกคำสั่งลงแฟ้มสคริปต์

เมื่อบรรทัดคำสั่งปรากฏขึ้นบนจอ R console แล้ว หากต้องการบันทึกบรรทัดคำสั่งนี้ลงแฟ้มสคริปต์ ก็ทำได้โดยกดปุ่มรายการ “บันทึก” ด้านหลังสุดของรายการเมนู บนหน้าต่างโมดุล **R-ICE** ใดๆ ก็ตาม บรรทัดคำสั่งหลังสุดที่อยู่บนหน้าจอ R console จะถูกบันทึกลงแฟ้มนั้น เราอาจเปิดแฟ้มนั้นดูเมื่อใดก็ได้ โดยไม่จำเป็นต้องปิดแฟ้มก่อน และเมื่อบันทึกบรรทัดคำสั่งใหม่ต่อท้ายลงไปก็สามารถ update ข้อมูลในแฟ้มดูได้ด้วยวิธีการของโปรแกรม text editor ที่ใช้เปิดนั้นๆ เช่นในโปรแกรม **Tinn-R** ใช้รายการ File >> Reload หรือโปรแกรม **Crimson Editor** สามารถพบการเปลี่ยนแปลงในแฟ้มและแจ้งให้ผู้ใช้ทราบโดยอัตโนมัตินั่นเอง

ขั้นตอนนี้อาจทำให้ผู้ใช้งงอยู่บ้างในระยะแรก แต่หากใช้จนชินแล้วจะรู้สึกว่าสะดวกในการบันทึกบรรทัดคำสั่งมาก หากไม่ต้องการบันทึกบรรทัดคำสั่งใดที่สร้างจากรายการของโมดุลต่างๆ ก็ไม่ต้องกดปุ่ม “บันทึก” แต่หากต้องการบันทึกเก็บไว้ในแฟ้ม ก็กดปุ่ม “บันทึก” บนหน้าต่างใดก็ได้ บรรทัดคำสั่งสุดท้ายก็จะถูกบันทึกลงแฟ้ม

***sink* ผลลัพธ์ลงแฟ้ม**

รายการสุดท้ายที่จะกล่าวถึงในที่นี้คือรายการ “sink ผลลัพธ์ลงแฟ้ม” รายการนี้จะบันทึกผลลัพธ์ของการทำงานทุกคำสั่งที่พิมพ์ลงบนหน้าจอ R console ลงในแฟ้ม โดยจะเริ่มบันทึกเมื่อเริ่มออกคำสั่งรายการย่อย “สร้างใหม่” หรือ “ต่อท้ายแฟ้ม” และจะหยุดบันทึกเมื่อเลือกรายการย่อย “หยุดการ sink” รายการนี้มีข้อจำกัดคือจะบันทึกเฉพาะผลลัพธ์ของคำสั่งเท่านั้นโดยไม่บันทึกบรรทัดคำสั่งที่สั่งงาน จึงอาจทำให้ไม่ทราบว่าผลลัพธ์นั้นมาจากคำสั่งอะไร และหากลืมหยุดการ sink แฟ้มบันทึกนั้นก็จะยาวออกไปเรื่อยๆ จนกว่าจะปิดการทำงานของ **R** จึงไม่แนะนำให้ใช้

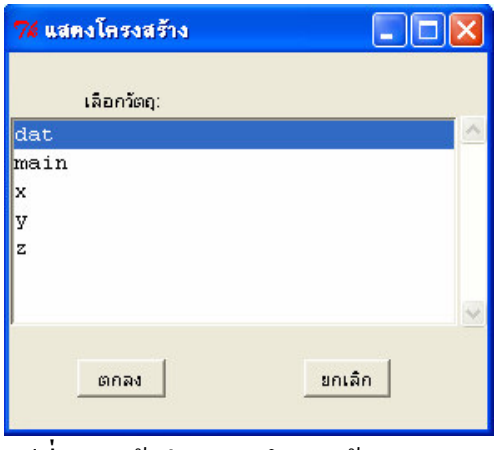
รายการนี้ แต่ควรสร้างแฟ้มสคริปต์และทำงานแฟ้มสคริปต์นั้นตามขั้นตอนที่กล่าวมาแล้วข้างต้นจะดีกว่า

ดูวัตถุในหน่วยความจำและเส้นทางค้นหา

รายการเมนูที่สองคือเมนู “วัตถุ” ซึ่งเป็นรายการเกี่ยวกับการจัดการวัตถุนั่นเอง รายการแรกคือ “ลิสต์วัตถุในหน่วยความจำ” เป็นการออกคำสั่ง `ls()` หรือ `objects()` และรายการถัดไปคือรายการ “ค้นหาวัตถุใน search path” เป็นการออกคำสั่ง `search()` ซึ่งทั้งสองคำสั่งนี้ก็เหมือนการทำงานของโมดุล ice.ojbects นั่นเอง แต่เป็นการทำงานผ่าน R console ซึ่งไม่ได้บอกว่าวัตถุนั้นๆ เป็นคลาสใด จึงมีข้อด้อยกว่าโมดุล ice.objects รายการถัดไปได้แก่ “attach กรอบข้อมูล” และ “detach กรอบข้อมูล” ทั้งสองรายการนี้เป็นการเรียกฟังก์ชัน `attach()` และ `detach()` ดังที่กล่าวในบทที่แล้วนั่นเอง

แสดงโครงสร้างของวัตถุ

รายการ “แสดงโครงสร้างของวัตถุ” เป็นการแสดงโครงสร้างของวัตถุนั้นๆ เมื่อเลือกรายการนี้จะมีหน้าต่างแสดงรายการวัตถุในหน่วยความจำมาให้เลือกดังรูปที่ 6.3 ฟังก์ชันที่เรียกใช้คือ `str()`



รูปที่ 6.3 หน้าต่างแสดงโครงสร้างของวัตถุ

เราสามารถเลือกวัตถุหลายอันเพื่อดูโครงสร้างพร้อมกันได้ ดังตัวอย่างข้างล่าง เมื่อเลือกทั้ง `dat, x, y` และ `z` บรรทัดคำสั่งที่ได้จะเป็น `str(); str(); ...` ซึ่งจะทำให้คำสั่ง `str()` ทำงานต่อเนื่องกันดังนี้

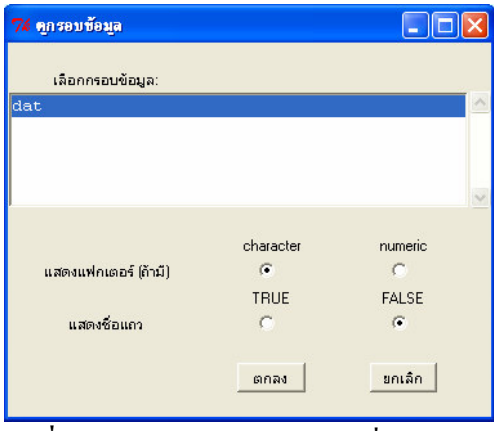
```
ICE>> str(dat); str(x); str(y); str(z))
`data.frame':   10 obs. of  3 variables:
```

```
$ x: Factor w/ 10 levels "a","b","c","d",..: 1 2 3 4 5 6 7 8 9
$ y: int  1 2 3 4 5 6 7 8 9 10
$ z: num  0.000 0.693 1.099 1.386 1.609 ...
chr [1:10] "a" "b" "c" "d" "e" "f" "g" "h" "i" "j"
int [1:10] 1 2 3 4 5 6 7 8 9 10
num [1:10] 0.000 0.693 1.099 1.386 1.609 ...
```

หมายเหตุ เครื่องหมาย ; เป็นคำสั่งที่ใช้สำหรับทำงานคำสั่งอื่นต่อเป็นทอดๆ จนหมดบรรทัด เสมือนว่าเป็นบรรทัดเดียว

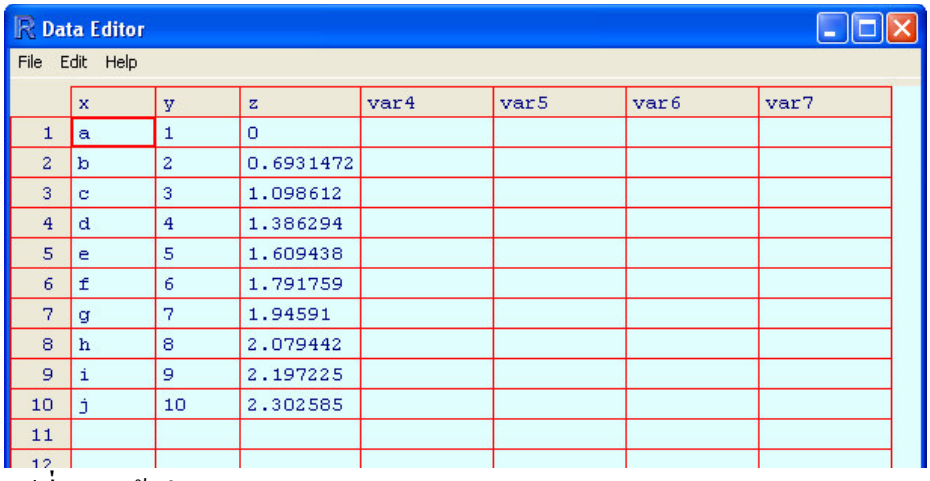
แก้ไขวัตถุ

มีตัวเลือกย่อยอีกสองรายการคือ “แก้ไขกรอบข้อมูล” และ “แก้ไขวัตถุชนิดอื่น” เมื่อเลือกรายการ “แก้ไขกรอบข้อมูล” จะมีหน้าต่างให้เลือกกรอบข้อมูลในหน่วยความจำดังรูปที่ 6.4 ซึ่งจะมีตัวเลือกแสดงแฟกเตอร์แบบใด เป็นแบบข้อความหรือแบบตัวเลข ซึ่งค่าปริยายตัวไว้ให้แสดงเป็นข้อความ และตัวเลือกแสดงชื่อแถวหรือไม่ ซึ่งค่าปริยายคือไม่แสดงชื่อแถว



รูปที่ 6.4 หน้าต่างดูกรอบข้อมูลเพื่อแก้ไข

เมื่อเลือกและกดปุ่ม “ตกลง” แล้ว จะมีหน้าต่าง Data Editor ให้แก้ไขข้อมูลในกรอบข้อมูลนั้น แต่ดังที่กล่าวมาก่อนหน้านี้ว่าไม่ควรทำการแก้ไขข้อมูลใดๆ ใน Data Editor โดยเฉพาะถ้าเป็นข้อมูลที่มีจำนวนมาก เนื่องจากจะไม่สามารถบันทึกกระบวนการแก้ไขเป็นสคริปต์คำสั่งได้ การแก้ไขข้อมูลควรเขียนเป็นสคริปต์คำสั่งดังจะกล่าวต่อไป เพื่อจะได้แสดงกฎเกณฑ์ของการแก้ไขเป็นบรรทัดคำสั่งอย่างชัดเจน ตรวจสอบได้



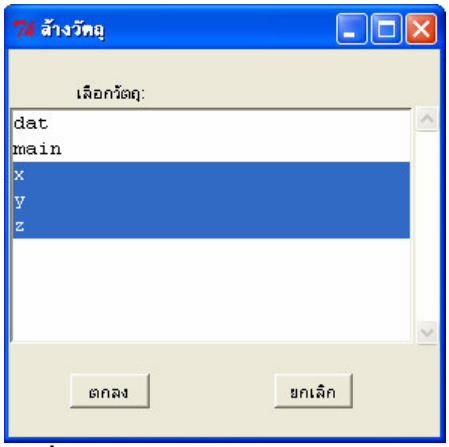
รูปที่ 6.5 หน้าต่าง Data Editor

สำหรับรายการ "แก้ไขวัตถุชนิดอื่น" เมื่อเลือกวัตถุที่จะแก้ไขแล้ว จะมีหน้าต่างแสดงวัตถุเพื่อให้แก้ไขได้เช่นเดียวกัน แต่ก็มีข้อจำกัดเช่นเดียวกับการแก้ไขกรอบข้อมูล

กำจัดวัตถุ

รายการ "กำจัดวัตถุ" นี้มีรายการย่อย "กำจัดวัตถุทั้งหมด" และ "กำจัดวัตถุที่เลือก" เมื่อเลือกรายการ "กำจัดวัตถุทั้งหมด" จะมีหน้าต่างให้ยืนยันว่าจะล้างหน่วยความจำทั้งหมดจริงหรือไม่ เพราะหากเลือกแล้วจะล้างวัตถุในหน่วยความจำออกทั้งหมด ที่จริงแล้วการล้างวัตถุออกจากหน่วยความจำทั้งหมดมักใช้บ่อยเมื่อเริ่มทำงานในแต่ละครั้ง เพื่อไม่ให้มีวัตถุอื่นที่ไม่ต้องการตกค้างในหน่วยความจำและทำให้เกิดความสับสนได้

เมื่อเลือกรายการ "กำจัดวัตถุที่เลือก" จะมีหน้าต่างให้เลือกวัตถุที่จะลบออกจากหน่วยความจำ รายการนี้สามารถเลือกวัตถุหลายๆ รายการที่จะกำจัดพร้อมกันได้ดังรูปที่ 6.6



รูปที่ 6.6 หน้าต่างล้างวัตถุในหน่วยความจำ

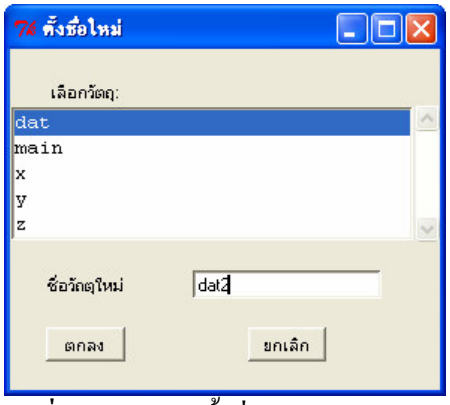
เมื่อเลือกวัตถุหลายรายการพร้อมกัน บรรทัดคำสั่งที่จะปรากฏขึ้นบนหน้าจอ R console จะมีลักษณะดังตัวอย่างข้างล่างนี้ คือลบข้อมูลใน list ของวัตถุนั่นเอง

```
ICE>> rm(list = c("x", "y", "z"))
```

เนื่องจากรายการนี้มีบรรทัดคำสั่งแสดงบนหน้าจอ จึงสามารถบันทึกลงแฟ้มสคริปต์คำสั่งได้

เปลี่ยนชื่อวัตถุ

รายการนี้เป็นการเปลี่ยนชื่อวัตถุที่มีอยู่ในหน่วยความจำของ **R** เมื่อเลือกแล้วจะมีหน้าจอให้เลือกวัตถุดังรูปที่ 6.7



รูปที่ 6.7 หน้าต่างตั้งชื่อใหม่

การตั้งชื่อใหม่นั้น แท้จริงแล้วเป็นการสร้างวัตถุขึ้นด้วยชื่อใหม่ แล้วลบวัตถุเดิมทิ้งไป ดังคำสั่งที่แสดงบน R console ดังนี้

```
ICE>> dat2 <- dat; rm(dat)
```

ตัวอย่างชุดข้อมูล

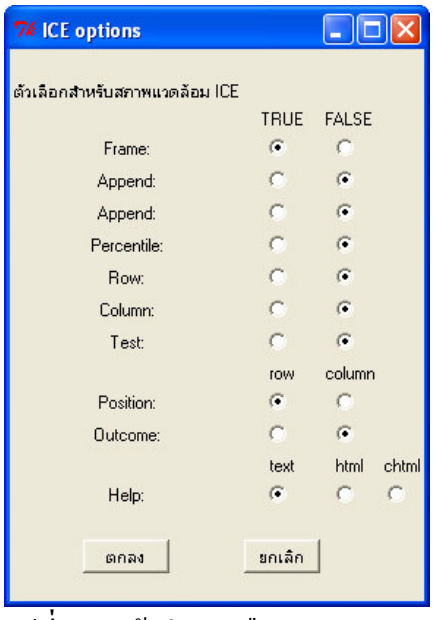
รายการนี้เป็นการเรียกชุดข้อมูลตัวอย่างมาใช้งาน ซึ่งต่อไปในบางหัวข้อเราจะใช้ชุดข้อมูลตัวอย่างใน library ice มาทำงาน ชุดข้อมูลตัวอย่างของ library ice มีดังนี้ `ahcc`, `ahcc2`, `ancdata`, `cca`, `cca9`, `cca.match`, `ccan`, `evans`, `irdata`, `mccdata` และ `vct` คำสั่งที่ปรากฏบน R console ได้แก่

```
ICE>> data(cca)
```

ซึ่งฟังก์ชัน `data()` นั้นใช้เรียกชุดข้อมูลใน library เท่านั้น ไม่ได้ใช้อ่านแฟ้มข้อมูลภายนอก

ตั้งค่าตัวเลือก

เราสามารถตั้งค่าตัวเลือกของสภาพแวดล้อม **ICE** ได้โดยใช้รายการนี้ ตัวเลือกเหล่านี้จะได้อธิบายในรายละเอียดต่อไปเมื่อได้มีโอกาสใช้ตัวเลือกเหล่านั้น ที่จะกล่าวในที่นี้คือตัวเลือกสุดท้าย คือตัวเลือก help ซึ่งจะบอกวิธีแสดงข้อความช่วยเหลือเป็นหน้าต่าง text หรือแสดงเป็น html ด้วย web browser หรือแสดงเป็น chm หรือ chtml นั่นเอง (เฉพาะบนระบบปฏิบัติการวินโดวส์) เราสามารถทดลองการทำงานของตัวเลือกนี้ได้ในหัวข้อถัดไป



รูปที่ 6.8 หน้าต่างตัวเลือก **ICE**

ความช่วยเหลือ

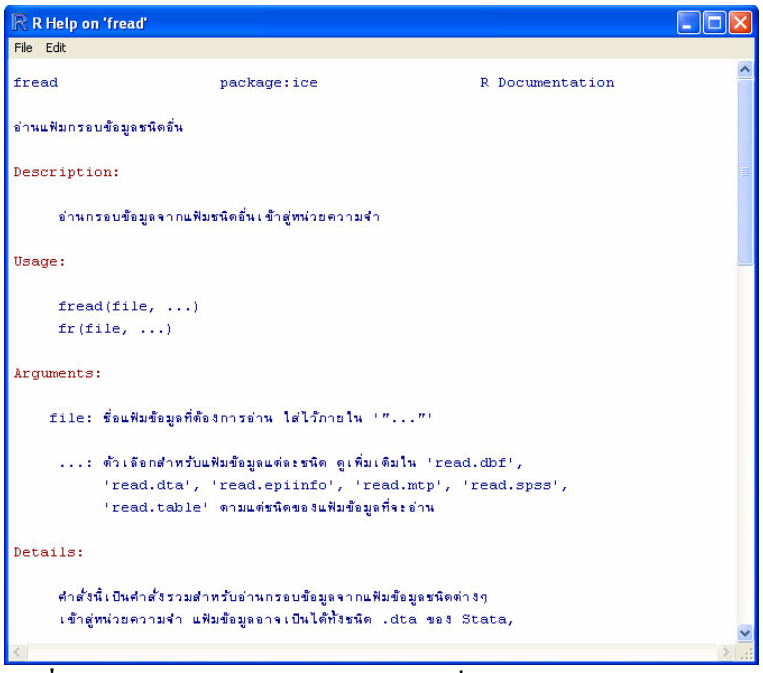
รายการความช่วยเหลือนี้มีรายการย่อยสามรายการได้แก่ “คำสั่ง” ซึ่งเป็นความช่วยเหลือสำหรับคำสั่งต่างๆ “เกี่ยวกับสภาพแวดล้อม **ICE**” และ “เกี่ยวกับ package **ICE**” ดังนี้ ที่สำคัญคือหากติดตั้ง **ICE** ภาษาไทย ก็จะต้องติดตั้งฟอนต์ที่สามารถแสดงตัวอักษรภาษาไทยได้ด้วย สำหรับระบบปฏิบัติการวินโดวส์ แนะนำฟอนต์ที่ดาวน์โหลดได้ฟรีสองฟอนต์คือ Courier MonoThai ซึ่งสร้างโดยคุณเศรษฐพล ลิ้มปราชญา โดยดาวน์โหลดได้ที่ http://software.thai.net/tis-620/courierthai.html และ Thaimono ซึ่งสร้างโดยคุณวรวิทย์ ฆังตระกูล โดยดาวน์โหลดได้ที่ http://software.thai.net/tis-620/modthai.html ติดตั้งฟอนต์ที่ดาวน์โหลดมานี้ในโฟลเดอร์ฟอนต์ แล้วเลือกรายการเมนู Edit >> GUI preferences... ในหน้าต่าง R Console แล้วพิมพ์ชื่อฟอนต์

ดังกล่าวลงในช่องชื่อฟอนต์ เนื่องจากชื่อฟอนต์ดังกล่าวไม่มีอยู่ในรายการฟอนต์ของ **R** และคลิกเลือกถูกที่ TrueType only หลังชื่อฟอนต์ แล้วกดปุ่ม “Save” และ “OK”

เมื่อบันทึกแฟ้ม Rconsole นี้แล้ว ให้ปิดโปรแกรม **R** แล้วเปิดใหม่อีกครั้ง คราวนี้ **R** จะสามารถแสดงภาษาไทยได้อย่างถูกต้อง

คำสั่ง

คำสั่งที่มีอยู่ในรายการช่วยเหลือนี้มีเฉพาะคำสั่งใน library ice ที่เกี่ยวข้องกับ ice.main เท่านั้น หากติดตั้ง **ICE** ภาษาไทย ข้อความช่วยเหลือก็จะแสดงเป็นภาษาไทย ดังตัวอย่างในรูปที่ 6.9

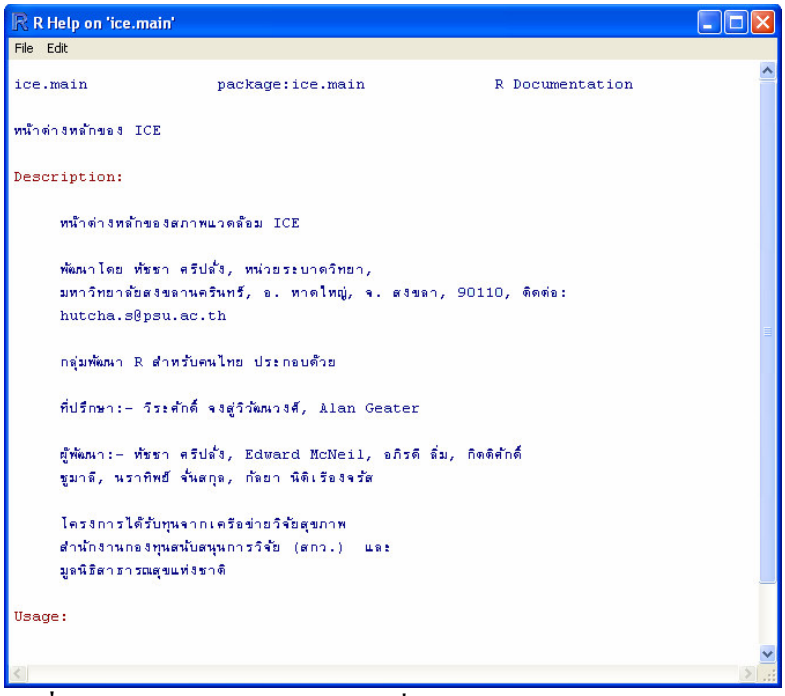


รูปที่ 6.9 หน้าต่างความช่วยเหลือของคำสั่ง `fread()`

เกี่ยวกับสภาพแวดล้อม ICE

รายการนี้แสดงข้อความช่วยเหลือเกี่ยวกับสภาพแวดล้อม **ICE** โดยบอกถึงกลุ่มผู้พัฒนาและวิธีใช้งาน โดยเป็นการเรียกบรรทัดคำสั่งต่อไปนี้

```
ICE>> help(ice.main, htmlhelp = FALSE, chmhelp = FALSE)
```



รูปที่ 6.10 หน้าต่างความช่วยเหลือเกี่ยวกับสภาพแวดล้อม **ICE**

เกี่ยวกับ **package ICE**

รายการนี้แสดงข้อความช่วยเหลือเกี่ยวกับ package **ICE** โดยบอกถึงกลุ่มผู้พัฒนา โดยเป็นการเรียกบรรทัดคำสั่งต่อไปนี้

```
ICE>> library(help = "ice.main")
```

ซึ่งมีข้อความในลักษณะที่เป็นคำอธิบายทางเทคนิคของการเขียนโปรแกรมมากกว่า จึงไม่ค่อยจำเป็นนักสำหรับผู้ใช้ทั่วไป

บทที่ 7 โมดุล ice.dataman ในการจัดการกรอบข้อมูล

สร้างกรอบข้อมูลเปล่า
ทำซ้ำกรอบข้อมูล
เปลี่ยนชื่อกรอบข้อมูล
ทำกรอบข้อมูลย่อย
สร้างตัวแปรใหม่จากตัวแปรที่มีอยู่แล้ว
เปลี่ยนชื่อตัวแปร
เปลี่ยนค่าในตัวแปร
เปลี่ยนรหัสค่าในตัวแปร
เปลี่ยนรหัส NA
ทำแฟกเตอร์เป็นเวกเตอร์ข้อความ
ทำแฟกเตอร์เป็นเวกเตอร์ตัวเลข
ทำเวกเตอร์ตัวเลขเป็นแฟกเตอร์
Label แฟกเตอร์
Relabel แฟกเตอร์
ความช่วยเหลือเกี่ยวกับฟังก์ชัน
ตัวอย่างของฟังก์ชัน

ในสภาพแวดล้อม **R-ICE** นั้น กระบวนการทำงานวิเคราะห์ข้อมูลแนะนำให้กรอกข้อมูลเป็นแฟ้มข้อมูลโดยเฉพาะ จากนั้นจึงนำข้อมูลเข้ามาสู่สภาพแวดล้อม **R** เป็นกรอบข้อมูล แล้วจึงจัดการข้อมูลภายในกรอบข้อมูลนั้นใหม่ถ้าจำเป็น เช่นการเปลี่ยนค่า จัดกลุ่มข้อมูล เปลี่ยนชนิดของข้อมูล หรือสร้างตัวแปรใหม่จากตัวแปรที่มีอยู่แล้ว โมดุล ice.dataman เป็นโมดุลสำคัญอีกโมดุลหนึ่งของสภาพแวดล้อม **ICE** ซึ่งใช้ในการจัดการกับวัตถุพื้นฐานโดยเฉพาะกรอบข้อมูลในลักษณะดังกล่าวนั่นเอง

เมนูต่างๆ ของ ice.dataman ทำงานโดยการเรียกคำสั่งจากแพคเกจ ice เป็นส่วนใหญ่ แต่บางคำสั่งก็เรียกจากแพคเกจ base โครงสร้างของรายการเมนู ice.dataman เป็นดังตารางที่ 7.1 รายการเมนูหลักมีสองรายการคือ “ข้อมูล” หรือ “Data” และ “ช่วยเหลือ” หรือ “Help” ในรายการ “ข้อมูล” เป็นรายการทำงานต่างๆ เพื่อจัดการกับข้อมูล ส่วนรายการ “ช่วยเหลือ” ประกอบด้วยรายการ “ฟังก์ชัน” ซึ่งแสดงหน้าต่างข้อความช่วยเหลือของคำสั่งต่างๆ ที่เกี่ยวข้องหรือถูกเรียกใช้จากรายการ “ข้อมูล” คือเป็นการเรียกคำสั่ง `help(`*`function_name`*`)` นั่นเอง ส่วนรายการ “ตัวอย่าง” เป็นการแสดงตัวอย่างการทำงานของฟังก์ชันต่างๆ ซึ่งก็ตรงกับการเรียกคำสั่ง `example(`*`function_name`*`)` นั่นเอง และรายการ “เกี่ยวกับหน้าต่าง dataman” และรายการ “เกี่ยวกับแพคเกจ dataman” เป็นคำอธิบายหน้าต่างและข้อมูลทางเทคนิกของ dataman อย่างสั้นๆ

ในบทนี้และบางบทต่อจากนี้ จะอธิบายการใช้งานตัวเลือกต่างๆ แต่เพียงบางตัวเลือกเท่านั้นพอให้เข้าใจแนวทางการทำงานของตัวเลือกต่างๆ ในสภาพแวดล้อม **R-ICE** ซึ่งจะทำงานเช่นเดียวกันทั้งหมด จึงไม่จำเป็นต้องอธิบายรายละเอียดซ้ำๆ ทุกคำสั่ง โดยสรุปคือเมื่อเลือกรายการตัวเลือกใดๆ แล้ว มักจะมีหน้าต่างหนึ่งหน้าต่าง หรือชุดของหน้าต่างเพื่อให้ผู้ใช้เลือกวัตถุที่จะมาทำงานด้วย และตัวเลือกของการทำงานนั้นๆ เมื่อเลือกเสร็จแล้ว ผู้ใช้กดปุ่ม “ตกลง” โมดุลนั้นๆ ก็จะส่งบรรทัดคำสั่งให้ **R** ทำงาน โดยบรรทัดคำสั่งเหล่านั้นจะแสดงให้เห็นบนหน้าจอ R console นั่นเอง ซึ่งผู้ใช้สามารถบันทึกบรรทัดที่ต้องการเก็บไว้ลงแฟ้มสคริปต์ได้ กระบวนการทำงานจะวนซ้ำเช่นนี้ทุกครั้งที่เลือกตัวเลือกใดๆ ก็ตามในระบบ **R-ICE** ดังนั้นจึงไม่มีความจำเป็นต้องอธิบายซ้ำไปทุกคำสั่ง

โดยส่วนใหญ่แล้ว การอธิบายการทำงานของตัวเลือกต่างๆ ส่วนใหญ่ต่อไปนี้จึงเป็นการอธิบายว่า เมื่อเลือกตัวเลือกนั้นแล้ว เท่ากับเป็นการเรียกใช้คำสั่งใดของโมดุล ice หรือคำสั่งพื้นฐานของ **R** ในแพคเกจใด จากนั้นผู้ใช้สามารถดูคำอธิบายการทำงานและตัวเลือกหรือเงื่อนไขของคำสั่งนั้นๆ ได้โดยดูจากตัวเลือกชื่อของคำสั่งนั้นภายใต้หัวข้อ “ช่วยเหลือ” หรือ “Help” ต่อไปได้ด้วยตัวเอง ทั้งนี้เพื่อไม่ให้หนังสือเล่มนี้หนามากเกินกว่าจะพกพาได้สะดวก

ตารางที่ 7.1 โครงสร้างของรายการเมนู ice.dataman

ข้อมูล	ช่วยเหลือ
สร้างกรอบข้อมูลเปล่า	ฟังก์ชัน
ทำซ้ำกรอบข้อมูล	create
เปลี่ยนชื่อกรอบข้อมูล	subset
ทำกรอบข้อมูลย่อย	rename
สร้างตัวแปรใหม่จากตัวแปรที่มีอยู่แล้ว	change
เปลี่ยนชื่อตัวแปร	recode
เปลี่ยนค่า	as.na
ในเวกเตอร์	recode.na
ในกรอบข้อมูล	to.character
เปลี่ยนรหัสค่า	to.numeric
ในเวกเตอร์	to.factor
ในกรอบข้อมูล	label
เปลี่ยนรหัส NA	relabel
ในเวกเตอร์	sort.data
ในกรอบข้อมูล	ตัวอย่าง
แฟกเตอร์เป็นข้อความ	create
แฟกเตอร์	subset
ในกรอบข้อมูล	rename
แฟกเตอร์เป็นตัวเลข	change
แฟกเตอร์	recode
ในกรอบข้อมูล	as.na
ตัวเลขเป็นแฟกเตอร์	recode.na
เวกเตอร์ตัวเลข	to.character
ในกรอบข้อมูล	to.numeric
Label แฟกเตอร์	to.factor
แฟกเตอร์	label
ในกรอบข้อมูล	relabel
Relabel แฟกเตอร์	sort.data
แฟกเตอร์	เกี่ยวกับหน้าต่างdataman
ในกรอบข้อมูล	text
Sort กรอบข้อมูล	html
	chm
	เกี่ยวกับแพคเกจ dataman

สร้างกรอบข้อมูลเปล่า

มีบางครั้งที่ผู้ใช้จะต้องการสร้างกรอบข้อมูลเปล่าขึ้นใช้งาน เพื่อทำงานกับกรอบข้อมูลเล็กๆ โดยการกรอกข้อมูลเข้าไปโดยตรงจากตารางข้อมูลของ **R** ที่มีไว้ให้ ก็สามารถใช้ตัวเลือกนี้จัดการได้ เมื่อเลือกตัวเลือกนี้แล้วจะปรากฏตารางให้กรอกข้อมูลเช่นเดียวกับการเรียกคำสั่ง

`create()` นั่นเอง หลังจากปิดตารางแล้วจะปรากฏบรรทัดคำสั่งดังข้างล่างบนหน้าต่าง R console

```
dataman>> new.dat <- create()
```

จะเห็นได้ว่าคำสั่งนี้จะสร้างกรอบข้อมูลใหม่ชื่อ `new.dat` ขึ้นมา ซึ่งจะบรรจุข้อมูลที่กรอกในตารางข้อมูลนั่นเอง หากผู้ใช้ต้องการเปลี่ยนชื่อกรอบข้อมูลก็สามารถทำได้โดยใช้ตัวเลือก “เปลี่ยนชื่อกรอบข้อมูล” ข้างล่างต่อไป

หมายเหตุ โปรดสังเกตว่าเมื่อมีการเรียกคำสั่งจากโมดุล dataman มาทำงาน สัญลักษณ์ prompt จะเปลี่ยนเป็น `dataman>>` และเช่นเดียวกัน ถ้าเลือกใช้งานโมดุลใด prompt ก็จะแสดงชื่อของโมดุลนั้นให้ทราบเช่นกัน

ทำซ้ำกรอบข้อมูล

ถ้าต้องการทำซ้ำกรอบข้อมูลที่มีอยู่เดิม เช่นเพื่อสร้างกรอบข้อมูลใหม่ที่เหมือนเดิมทุกประการ (ในชื่อใหม่) เมื่อเรียกตัวเลือกนี้แล้วจะปรากฏหน้าต่างให้เลือกกรอบข้อมูลที่มีอยู่แล้วและช่องให้กรอกชื่อกรอบข้อมูลใหม่ ซึ่งเมื่อทำเสร็จแล้ว **R** จะแสดงบรรทัดคำสั่งต่อไปนี้

```
dataman>> new.dat2 <- new.dat
```

โดยที่ `new.dat2` เป็นชื่อกรอบข้อมูลใหม่ที่ผู้ใช้กรอกเข้าในช่องกรอกชื่อที่กล่าวข้างต้น และ `new.dat` คือชื่อกรอบข้อมูลที่มีอยู่แล้วในหน่วยความจำนั่นเอง

เปลี่ยนชื่อกรอบข้อมูล

เมื่อต้องการเปลี่ยนชื่อกรอบข้อมูลที่มีอยู่แล้วเป็นชื่ออื่น ให้เลือกตัวเลือกนี้ ซึ่งจะมีหน้าต่างคล้ายตัวเลือก “ทำซ้ำกรอบข้อมูล” แต่บรรทัดคำสั่งที่ใช้ทำงานที่แสดงบน R console จะต่างไป ดังนี้

```
dataman>> new.dat2 <- new.dat; rm(new.dat)
```

นั่นคือเมื่อสร้างกรอบข้อมูลใหม่ขึ้นแล้ว จะลบกรอบข้อมูลเก่าออกไปด้วย ผลลัพธ์จึงเป็นการเปลี่ยนชื่อกรอบข้อมูลนั่นเอง

ทำกรอบข้อมูลย่อย

เป็นรายการสำหรับการทำกรอบข้อมูลใหม่ที่มีขนาดเล็กกว่าเดิม จากกรอบข้อมูลเดิม คือใช้คำสั่ง `subset()` ของแพคเกจ `base` นั่นเอง เมื่อเลือกรายการนี้จะมีหน้าต่างให้เลือกกรอบข้อมูลที่จะนำมาทำ subset และช่องตั้งชื่อกรอบข้อมูลใหม่ คล้ายการทำงานในสองรายการข้างต้น แต่เมื่อกดปุ่ม “ตกลง” หรือ “OK” แล้ว จะมีหน้าต่างอีกหนึ่งหน้าต่างให้เลือกตัวแปรที่อยู่ภายใน และเงื่อนไขของการเลือกรายการ ซึ่งการเลือกตัวแปรอาจเลือกจากรายการตัวแปรด้านบน หรือเลือกโดยพิมพ์เงื่อนไขในช่องล่างสุดก็ได้ ซึ่งจะได้กล่าวต่อไป การเลือกตัวแปรสามารถคลิกเพื่อเลือกเพิ่มทีละตัวแปรก็ได้ หรือจะใช้ปุ่ม Crtl ช่วยเพื่อเลือกทีละหลายตัวแปรพร้อมกันก็ได้ หากไม่ต้องการตัวแปรที่เลือกไปแล้วก็กดคลิกซ้ำที่ตัวแปรนั้น ตัวแปรนั้นก็จะไม่ถูกเลือก

ช่องถัดลงมาเป็นช่อง “เงื่อนไขเซ็ตย่อย” หรือ “Subset condition” ใช้เพื่อตั้งเงื่อนไขในแนวแถว คือเป็นเงื่อนไขที่จะใช้กับตัวแปรทุกตัวที่เลือกแล้ว เพื่อคัดเฉพาะรายการที่มีคุณสมบัติเข้ากับเงื่อนไขที่ตั้ง เช่น อาจตั้งว่า

```
sex==1 & age>25
```

ดังนี้ ก็เป็นการเลือกเฉพาะรายการที่ตัวแปร `sex` มีค่าเท่ากับ 1 และตัวแปร `age` มีค่ามากกว่า 25 เงื่อนไขนี้จะใช้กับทุกตัวแปรที่เลือกไว้แล้ว

ส่วนช่องล่างสุด “เลือกตัวแปร” หรือ “Select variables” เป็นการเลือกตัวแปรเช่นกัน แต่ใช้ในกรณีที่ตัวแปรมีจำนวนมาก การคลิกเลือกทีละตัวแปรก็จะเสียเวลามาก เราสามารถพิมพ์เงื่อนไขตัวแปรในรูปแบบต่อไปนี้ได้ เช่นกรอบข้อมูลเดิมมีข้อมูลดังนี้

```
1      id
2      name
3      sex
4      age
…
7      smoke
…
11     Status
```

หากต้องการเลือกแต่เฉพาะตัวแปรลำดับที่ 1 แล้วข้ามไป 3 ถึง 7 แล้วข้ามไป 11 ก็สามารถพิมพ์เลขลำดับเข้าไปได้โดยตรงดังนี้เลย

```
1, 3:7, 11
```

หรือจะพิมพ์ชื่อตัวแปรก็ได้ ดังนี้

```
id, sex:smoke, status
```

ซึ่งถ้าหากมีตัวแปรจำนวนมากในกรอบข้อมูล การเรียกด้วยเลขลำดับจะทำให้พิมพ์ได้สั้นลง และทำงานได้รวดเร็วขึ้นมาก ขอให้สังเกตว่าเครื่องหมายจุลภาค “,” เป็นการคั่นตัวแปรที่ไม่มีลำดับ

ติดกัน หากต้องการเลือกตัวแปรที่เรียงติดกันยาวจากลำดับที่ 3 ถึง 7 หรือจาก `sex` ถึง `smoke` ก็สามารถใช้เครื่องหมาย ":" ได้ดังตัวอย่างข้างต้น

ในช่องทั้งสองนี้ถ้าหากเว้นว่างไว้ จะหมายถึงไม่ต้องการเลือกเงื่อนไขด้านแถว หรือเงื่อนไขด้านสดมภ์ นั่นคือด้านแถวก็เลือกหมดทุกรายการ ส่วนด้านสดมภ์ก็เลือกตามที่เลือกจากรายการตัวแปรด้านบนนั่นเอง

ถึงจุดนี้เมื่อกดปุ่ม "ตกลง" หรือ "OK" แล้ว dataman จึงจะส่งบรรทัดคำสั่งให้ **R** ทำงาน ดังเช่น

```
dat3 <- subset(dat2, sex==1 & age>25, select = c(1,3:7,11))
```

นั่นคือเป็นการบอกให้ **R** ทำกรอบข้อมูลย่อยชื่อ `dat3` จากกรอบข้อมูล `dat2` โดยเลือกเฉพาะรายการที่ `sex==1 & age>25` และตัวแปรลำดับที่ `1,3:7,11`

*ถึงตรงนี้คงจะเห็นถึงวิธีการทำงานของสภาพแวดล้อม **R-ICE** ได้ชัดเจนแล้วว่าเป็นการนำการทำงานแบบกราฟิก คือหน้าต่าง รายการตัวเลือก ช่องกรอกข้อความ ปุ่มกด และอื่นๆ มาใช้เพื่อให้ผู้ใช้ได้เลือกหรือกรอกข้อมูลตามลำดับ และมีคำอธิบายพอให้เข้าใจว่าสิ่งที่เลือกหรือกรอกนั้นคืออะไร รายการและช่องกรอกข้อมูลเหล่านั้นก็คือ argument ต่างๆ ที่จะนำไปใส่ให้กับฟังก์ชันนั่นเอง ทำให้ผู้ใช้ใส่รายการ argument ได้ครบถ้วน ปิดวงเล็บได้ทั้งหมด และไม่จำเป็นต้องจดจำลำดับและวิธีเขียนซึ่งมักจะจำได้ยาก เพราะจากประสบการณ์ในการใช้ **R** มานาน ยังพบว่าแม้จะเป็นฟังก์ชันที่ใช้อยู่ประจำ ก็ยังอาจพิมพ์ลำดับหรือชื่อ argument ผิดได้ง่ายๆ และอาจทำให้เกิดข้อผิดพลาดในการทำงาน ซึ่งหาก **R** ฟ้องว่าพิมพ์คำสั่งผิดพลาดและไม่ทำงานก็ยังพอจะแก้ไขได้ แต่หากผิดพลาดแล้วยังทำงานได้ จะได้ผลลัพธ์ผิดไปจากที่ตั้งใจ และผู้ใช้อาจไม่ทราบเลยว่ามีข้อผิดพลาดเกิดขึ้น*

ตารางที่ 7.2 สรุปฟังก์ชันและรูปแบบการใช้งานของรายการต่างๆ ในโมดุล ice.dataman

รายการ	ฟังก์ชัน	รูปแบบการใช้งานฟังก์ชัน
สร้างกรอบข้อมูลเปล่า	`create()`	`dat <- create()`
ทำซ้ำกรอบข้อมูล	`<-`	`new.dat <- old.dat`
เปลี่ยนชื่อกรอบข้อมูล	`<- ; rm()`	`new.dat <- old.dat; rm(old.dat)`
ทำกรอบข้อมูลย่อย	`subset()`	`subset(x, subset, select)`
สร้างตัวแปรใหม่จากตัวแปรที่มีอยู่แล้ว	`transform()`	`transform(_data, ...)`
เปลี่ยนชื่อตัวแปร	`rename()`	`remane(data, x, var.names)`
เปลี่ยนค่าในตัวแปร	`change()`	`change(data, x, condition, value, append = get.ec.option("append"), var.names)`
เปลี่ยนรหัสค่าในตัวแปร	`recode()`	`recode(x, old, new)` `recode(data, x, old, new, append =`

		`get.ec.option("append"), var.names)`
เปลี่ยนตัวเลขเป็น NA	`as.na()`	`as.na(x, code)`
		`as.na(data, x, code, append = get.ec.option("append"), var.names)`
เปลี่ยนรหัส NA	`recode.na()`	`recode.na(x, code)`
		`recode.na(data, x, code, append = get.ec.option("append"), var.names)`
ทำแฟกเตอร์เป็นเวกเตอร์ข้อความ	`to.character()`	`to.character(data, x, append = get.ec.option("append"), var.names)`
ทำแฟกเตอร์เป็นเวกเตอร์ตัวเลข	`to.numeric()`	`to.numeric(data, x, append = get.ec.option("append"), var.names)`
ทำเวกเตอร์ตัวเลขเป็นแฟกเตอร์	`to.factor()`	`to.factor(data, x, labels, ordered = FALSE, append = get.ec.option("append"), var.names)`
Label แฟกเตอร์	`label()`	`label(data, x, labels, sep="", ordered = FALSE, append = get.ec.option("append"), var.names)`
Relabel แฟกเตอร์	`relabel()`	`relabel(data, x, ref, append = get.ec.option("append"), var.names)`
เรียงลำดับข้อมูล	`sort.data()`	`sort.data(data, x, decreasing = FALSE)`

จากตารางข้างต้น มีบางฟังก์ชันที่เกี่ยวข้องกับการจัดการกรอบข้อมูลเป็นฟังก์ชันพื้นฐานของ **R** ที่มีอยู่แล้วในแพคเกจ `base` ได้แก่ฟังก์ชัน `<-, subset(), transform()` แต่ส่วนใหญ่จะอยู่ในแพคเกจ `ice` อย่างไรก็ตามยังมีอีกบางฟังก์ชันที่มีอยู่ในแพคเกจ `ice` แต่ไม่ได้อยู่ในรายการนี้เนื่องจากมีที่ใช้น้อย ได้แก่ฟังก์ชัน `join(), shift(), pack()` และ `expand()`

ในตารางที่ 7.2 นี้จะเห็นว่าบางฟังก์ชันเรียกใช้ฟังก์ชัน `get.ec.option()` เป็น argument ด้วย ซึ่งฟังก์ชันนี้มีหน้าที่ค้นหาค่าปกติของสภาพแวดล้อม **ICE** ที่ถูกตั้งค่าไว้ในรายการ “ตัวเลือกทั่วไป” ของโมดุล ice.main เช่นในกรณี `get.ec.option("append")` ก็เป็นการเรียกใช้ค่าปกติของ append นั่นเอง

บทที่ 8 โมดุล ice.summary ในการสรุปข้อมูล

บรรยายกรอบข้อมูล

แสดงกรอบข้อมูล

ทำรายการระเบียนซ้ำ

ดูโครงสร้างวัตถุ

summary และ summarize วัตถุ

ทำตารางแจงความถี่และตารางสัมพันธ์ไขว้

ในการสรุปวัตถุต่างๆ ในสภาพแวดล้อม **R** นั้น มีคำสั่งที่เกี่ยวข้องจำนวนหนึ่ง ได้แก่ ฟังก์ชันที่ปรากฏในตารางที่ 8.1 อย่างไรก็ตามโมดุลนี้ได้รวบรวมเฉพาะฟังก์ชันง่ายๆ ที่ใช้บ่อยในการสรุปวัตถุเท่านั้น ยังมีฟังก์ชันที่เกี่ยวข้องกับการสรุปวัตถุอีกจำนวนมากซึ่งส่วนใหญ่จะอยู่ในแพคเกจพื้นฐานของ **R** ที่ไม่ได้นำมาบรรจุในโมดุล เช่น ฟังก์ชัน `ftable()`, `apply()`, `tapply()` เป็นต้น นอกจากนี้ยังมีแพคเกจในทางระบาดวิทยาโดยเฉพาะ เช่น Epi ซึ่งมีฟังก์ชันในการสรุปวัตถุและข้อมูลในรูปแบบพิเศษอีกด้วย ผู้ใช้ที่ทำงานเฉพาะเรื่องสามารถดาวน์โหลดแพคเกจเหล่านั้นมาใช้งานได้ แต่ในปัจจุบันยังไม่มีการทำเมนูกราฟิกให้กับแพคเกจเหล่านั้น

ตารางที่ 8.1 โครงสร้างของรายการเมนู ice.summary

สรุป	**ช่วยเหลือ**
บรรยายกรอบข้อมูล	ฟังก์ชัน
แสดงกรอบข้อมูล	describe
ทำรายการระเบียนซ้ำ	display
ในเวกเตอร์	list.rep
ในกรอบข้อมูล	str
ดูโครงสร้างวัตถุ	summary
summary วัตถุ	summarize
summarize วัตถุ	tab
ในเวกเตอร์	ftab
ในกรอบข้อมูล	xtab
ทำตารางอย่างง่าย	ตัวอย่าง
ในเวกเตอร์	describe
ในกรอบข้อมูล	display
ทำตารางแจงความถี่	list.rep
ทำตารางสัมพันธ์ไขว้	str
	summary
	summarize
	tab
	ftab
	xtab
	เกี่ยวกับหน้าต่าง summary
	text
	html
	chm
	เกี่ยวกับแพคเกจ summary

บรรยายกรอบข้อมูล

กรอบข้อมูลที่ถูกบันทึกและอ่านเข้ามาในรูปแบบ DTA ของโปรแกรม **Stata** จะมีส่วนที่เป็นคำบรรยายกรอบข้อมูลและคำบรรยายตัวแปรแต่ละตัวติดมาด้วย ซึ่งจะดูได้โดยใช้ตัวเลือกนี้ ซึ่งจะเท่ากับการออกคำสั่ง `describe()` เมื่อเลือกตัวเลือกนี้จะมีหน้าต่างให้เลือกกรอบข้อมูลที่ต้องการบรรยาย เมื่อเลือกแล้วจะปรากฏข้อความใน R console ดังตัวอย่างข้างล่าง

```
summary>> describe(cca)
No. of observation = 262
Population-based case-control study of cholangiocarcinoma
  Variable   Description
1 status     case or control
2 ovab       antibody to Opisthorchis viverrini
3 alcohol    alcohol drinking status
4 smoke      smoking status
5 age        age in year
```

แสดงกรอบข้อมูล

ตัวเลือกนี้ใช้สำหรับแสดงบางส่วนของกรอบข้อมูลโดยระบุเงื่อนไขด้านแถวและสดมภ์ได้เหมือนตัวเลือก “ทำกรอบข้อมูลย่อย” ในบทที่ 7 เรื่อง **โมดุล ice.dataman ในการจัดการกรอบข้อมูล** ข้างต้นนั่นเอง รายการและช่องกรอกข้อความก็มีลักษณะเช่นเดียวกัน ยกเว้นว่าด้านล่างจะมีให้เลือกว่าจะแสดงบนหน้าจอ R console หรือจะแสดงโดย editor

รายการนี้เป็นการเรียกใช้คำสั่ง `display()` ซึ่งอยู่ในแพคเกจ `ice` นั่นเอง

ทำรายการระเบียนซ้ำ

มีบางครั้งที่ในการกรอกข้อมูลอาจทำโดยผู้กรอกข้อมูลหลายคน หรือทำหลายครั้ง ทำให้มีโอกาสที่จะกรอกข้อมูลบางรายการซ้ำได้ หรือพิมพ์หมายเลขรายการผิดพลาด เราอาจใช้คำสั่ง `list.rep()` เพื่อดูว่ามีเลขระเบียนซ้ำมากกว่า 1 หรือ 2 รายการตามแต่เราจะกำหนดได้ ซึ่งรายการนี้จะเรียกใช้ฟังก์ชันนี้เพื่อทำงานในลักษณะนี้นั่นเอง

ดูโครงสร้างวัตถุ

รายการนี้เป็นการดูโครงสร้างของวัตถุเช่นเดียวกับที่ได้กล่าวแล้วในบทที่ 6 เรื่อง **โมดุล ice.main ในการจัดการแฟ้มข้อมูลและวัตถุ** ภายใต้รายการเมนู “วัตถุ” นั่นเอง แต่ในที่นี้ได้จัดรายการนี้ไว้ในที่นี้ด้วยเพื่อให้ผู้ใช้ที่ต้องการดูโครงสร้างของกรอบข้อมูลจะได้หารายการนี้ได้ง่าย ภายใต้รายการสรุปข้อมูลของ ice.summary ด้วย

summary และ summarize วัตถุ

หัวข้อนี้ที่จริงแล้วประกอบด้วยรายการสองรายการ คือ summary และ summarize ซึ่งทำงานคล้ายกันในบางส่วน คือสรุปวัตถุ แต่สรุปในรูปแบบที่แตกต่างกัน โดยที่ summary วัตถุเป็นการเรียกใช้ฟังก์ชัน `summary()` ซึ่งจะให้รายละเอียดของวัตถุที่อยู่ภายในเป็นลักษณะของวัตถุเช่นกัน ส่วนรายการ summarize วัตถุ เป็นการเรียกใช้ฟังก์ชัน `summarize()` ซึ่งจะเน้นการแสดงผลในรูปแบบตาราง ผู้ใช้สามารถเลือกใช้รายการทั้งสองนี้ได้ตามใจชอบ เมื่อเลือกรายการดังกล่าวแล้ว ผลลัพธ์จะแสดงบนหน้าจอ R console

ทำตารางแจงความถี่และตารางสัมพันธ์ไขว้

หัวข้อนี้ประกอบด้วยรายการสามรายการ ได้แก่ “ทำตารางอย่างง่าย” “ทำตารางแจงความถี่” และ “ทำตารางสัมพันธ์ไขว้” ซึ่งเรียกใช้ฟังก์ชัน `tab()`, `ftab()` และ `xtab()` ตามลำดับ

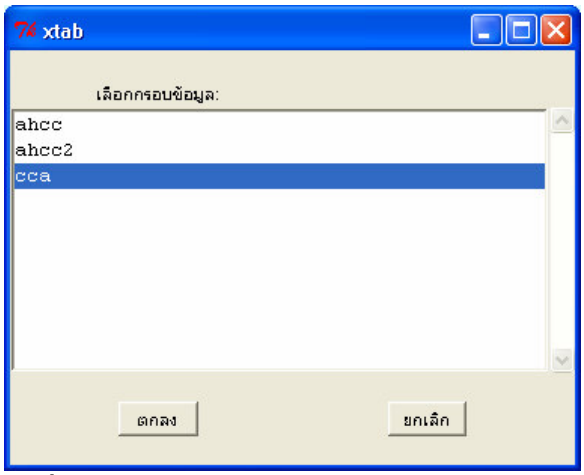
สิ่งที่ฟังก์ชัน `tab()` ต่างไปจากฟังก์ชัน `table()` ในแพคเกจ `base` คือ ฟังก์ชัน `tab()` สามารถเลือกที่จะแสดงตารางร้อยละได้ด้วย และนอกจากนี้ยังเลือกที่จะแสดงจำนวนข้อมูลที่เป็น NA เสมือนหนึ่งว่าเป็นค่าหนึ่งได้ด้วย และยังเลือกที่จะแสดง label ของแฟกเตอร์ หรือจะแสดงเป็นตัวเลขระดับก็ได้ และในกรณีที่เป็นตารางความสัมพันธ์ระหว่างตัวแปรสองตัว ก็สามารถที่จะเลือกแสดงการทดสอบ Chi-squared และหากสามารถคำนวณการทดสอบ Fisher’s exact probability ได้ก็จะแสดงให้ด้วยเช่นกัน จึงเห็นได้ว่าฟังก์ชัน `tab()` ค่อนข้างสะดวกในการนำไปใช้งานเพื่อแสดงสรุปวัตถุในเชิงการวิเคราะห์ทางสถิติเบื้องต้นมากกว่า เนื่องจากไม่ต้องแยกคำนวณหรือเขียนบรรทัดคำสั่งหลายบรรทัด เพื่อทำงานทั้งหมดที่กล่าวมา

การทำตารางแจงความถี่นับว่ามีความสำคัญในการสรุปข้อมูลในเบื้องต้น เพื่อให้เห็นภาพทั่วไปของข้อมูล เช่น ดูได้ว่ามีค่าที่เกิดขอบเขตที่เป็นไปได้หรือไม่ หรือดูอย่างหยาบๆ ว่าค่าแต่ละค่าของตัวแปรนั้นมีการกระจายอย่างไร คำสั่งที่เรียกใช้คือคำสั่ง `ftab()`

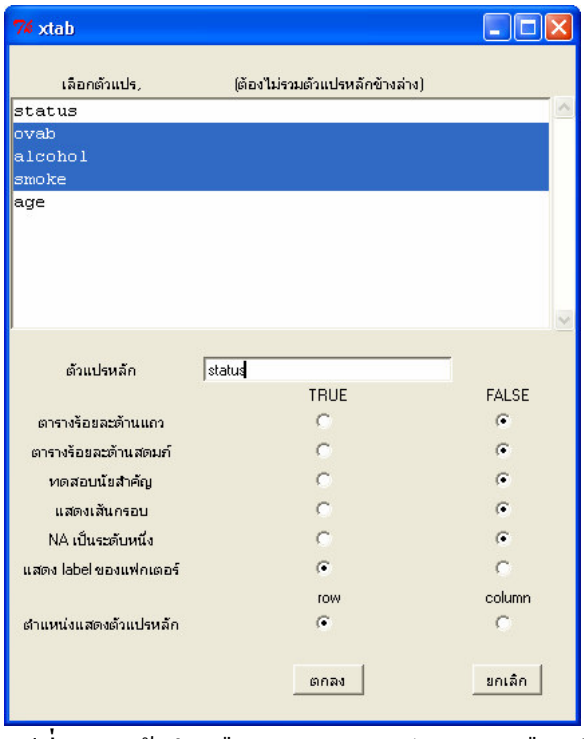
ส่วนการทำตารางสัมพันธ์ไขว้ก็เช่นเดียวกัน ในการวิจัยเพื่อหาความสัมพันธ์ระหว่างสิ่งที่คิดว่าเป็นเหตุกับสิ่งที่คิดว่าเป็นผล การทำตารางความสัมพันธ์ก็จะทำให้สามารถดูในเบื้องต้นได้ว่าคู่ของตัวแปรใดสัมพันธ์กันหรือไม่ และมากน้อยเพียงใด คำสั่งที่เรียกใช้คือคำสั่ง `xtab()`

ข้อเด่นของคำสั่ง `ftab()` และ `xtab()` คือผู้ใช้สามารถออกคำสั่งเพียงครั้งเดียวเพื่อทำตารางจำนวนมากได้ ซึ่งถ้าหากใช้คำสั่งพื้นฐาน `table()` หรือ `tab()` ผู้ใช้จะต้องออกคำสั่งหลายครั้งตามจำนวนตารางที่ต้องการทำ ผลลัพธ์ของการเลือกตัวเลือกทั้งสามนี้จะแสดงออกบน R

console ลองดูตัวอย่างการทำงานของรายการ “ทำตารางสัมพันธ์ไขว้” ซึ่งจะมีหน้าต่างปรากฏดังรูปที่ 8.1 และ 8.2



รูปที่ 8.1 หน้าต่างเลือกกรอบข้อมูลที่จะนำมาทำตารางสัมพันธ์ไขว้



รูปที่ 8.2 หน้าต่างเลือกรายการตัวแปรและตัวเลือกต่างๆ ของการทำตารางสัมพันธ์ไขว้

และได้ผลของการเลือกรายการ “ทำตารางสัมพันธ์ไขว้” บนหน้าต่าง R console ดังต่อไปนี้

```
summary>> xtab(cca, status, c(ovab,alcohol,smoke), row=TRUE,
colm=FALSE, test=TRUE, position="row", frame=FALSE,
na.included=FALSE, factor.labels=TRUE)
row: status, column: ovab
        negative positive Total
control      118       11   129
```

```
case          62       65   127
Total        180       76   256
<row percent>
        negative positive   Total
control    91.47     8.53     100
case       48.82    51.18     100

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

row: status, column: alcohol
        never occasional ex-drinker drinker Total
control    48         60          7      13   128
case       31         40         17      40   128
Total      79        100         24      53   256
<row percent>
        never occasional ex-drinker drinker   Total
control 37.50      46.88       5.47   10.16     100
case    24.22      31.25      13.28   31.25     100

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

row: status, column: smoke
        never occasional ex-smoker smoker Total
control    64          6        14     44   128
case       47          8        21     52   128
Total     111         14        35     96   256
<row percent>
        never occasional ex-smoker smoker   Total
control 50.00       4.69     10.94  34.38     100
case    36.72       6.25     16.41  40.62     100

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

ตารางที่ 8.2 สรุปฟังก์ชันและรูปแบบการใช้งานของรายการต่างๆ ในโมดูล ice.summary

รายการ	ฟังก์ชัน	รูปแบบการใช้งานฟังก์ชัน
บรรยายกรอบข้อมูล	`describe()`	`describe()`
ทำรายการระเบียนซ้ำ	`list.rep()`	`list.rep(data, x, n = 2)`
ดูโครงสร้างวัตถุ	`str()`	`str(object, ...)`
summary วัตถุ	`summary()`	`summary(object, ...)`
summarize วัตถุ	`summarize()`	`summarize(object, subset, select, group, pctile = get.ec.option("pctile"), frame = get.ec.option("frame"))`
ทำตารางอย่างง่าย	`tab()`	`tab(data, x, pct = get.ec.option("pct"), na.included = FALSE, factor.labels = TRUE)`
		`tab(data, x, y, group, row = get.ec.option("row"), colm = get.ec.option("colm"), test = get.ec.option("test"), frame = get.ec.option("frame"), na.included = FALSE, factor.labels = TRUE)`
ทำตารางแจงความถี่	`ftab()`	`ftab(data, columns, pct=get.ec.option("pct"), frame = get.ec.option("frame"), na.included = FALSE, factor.labels = TRUE)`
ทำตารางสัมพันธ์ไขว้	`xtab()`	`xtab(data, key, columns, row = get(".ec.row",`

```
.GlobalEnv), colm = get(".ec.colm", .GlobalEnv),
test = get(".ec.test", .GlobalEnv), position =
get.ec.option("position"), frame =
get.ec.option("frame"), na.included = FALSE,
factor.labels = TRUE)
```

ในตารางที่ 8.2 นี้จะเห็นว่าบางฟังก์ชันเรียกใช้ฟังก์ชัน `get.ec.option()` เป็น argument ด้วย ซึ่งฟังก์ชันนี้มีหน้าที่ค้นหาค่าปกติของสภาพแวดล้อม **ICE** ที่ถูกตั้งค่าไว้ในรายการ “ตัวเลือกทั่วไป” ของโมดุล ice.main เช่นในกรณี `get.ec.option("frame")` ก็เป็นการเรียกใช้ค่าปกติของ frame นั่นเอง

บทที่ 9 โมดุล ice.graph ในการทำกราฟ

สร้างหน้าต่าง graphic ใหม่

ทำหลายกราฟในหนึ่งหน้าต่าง

ทำกราฟ histogram

ทำกราฟ pie

ทำกราฟ boxplot

ในการทำงานทางสถิตินั้น บ่อยครั้งที่เราต้องการดูการกระจายของข้อมูล หรือภาพสรุปของข้อมูล รายการเมนูนี้เป็นการทำกราฟิกพื้นๆ สำหรับงานเช่นนี้ เช่น การทำ histogram หรือการทำ pie chart เป็นต้น ส่วนการทำกราฟิกที่ซับซ้อน อาจหาฟังก์ชันที่เขียนขึ้นเป็นการเฉพาะได้จากแหล่งต่างๆ หรือไม่เช่นนั้นก็ต้องศึกษาวิธีการเขียนกราฟิกใน **R**

ตารางที่ 9.1 โครงสร้างของรายการเมนู ice.graph

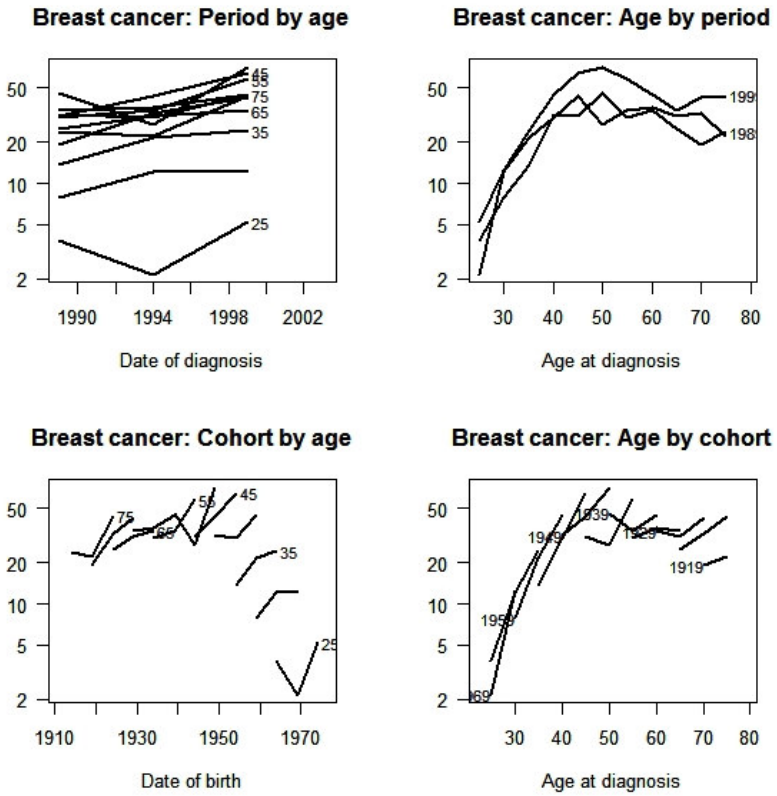
กราฟิก	**ช่วยเหลือ**
หน้าต่าง graphic ใหม่	เกี่ยวกับหน้าต่าง graph
หลาย graph ในหนึ่งหน้าต่าง	text
Histogram	html
เวกเตอร์	chm
กรอบข้อมูล	เกี่ยวกับแพคเกจ graph
Pie	
เวกเตอร์	
กรอบข้อมูล	
Boxplot	
formula	
เวกเตอร์	
กรอบข้อมูล	

*สร้างหน้าต่าง **graphic** ใหม่*

โดยปกติแล้วเมื่อออกคำสั่งสร้างกราฟ **R** จะสร้างกราฟิกขึ้นในหน้าต่าง x window ที่เปิดอยู่เดิม แต่ถ้าไม่มีหน้าต่างกราฟิกเปิดอยู่ก่อนก็จะสร้างขึ้นใหม่ นั่นหมายความว่าหากออกคำสั่งสร้างกราฟิกใหม่ กราฟิกใหม่นั้นก็จะวาดทับกราฟิกเดิม ถ้าไม่ต้องการให้วาดทับ แต่ต้องการให้วาดบนหน้าต่างใหม่ ก็สามารถออกคำสั่ง `x11()`, `X11()` หรือ `windows()` เพื่อสร้างหน้าต่างว่างใหม่ได้ ตัวเลือกนี้คือการทำงานเช่นเดียวกันนี้นั่นเอง โดยเมื่อเลือกตัวเลือกนี้ จะปรากฏหน้าต่างให้กำหนดขนาดของหน้าต่าง x window ค่าปกติของความกว้างและความสูงคือ 7 ซึ่งนับว่ามีขนาดค่อนข้างใหญ่ ผู้ใช้อาจลดขนาดลงเป็น 5 ทั้งคู่ก็ได้ อย่างไรก็ตาม ผู้ใช้สามารถใช้เมาส์เปลี่ยนขยายหรือย่อขนาดของหน้าต่างภายหลังได้ตามใจชอบ

ทำหลายกราฟในหนึ่งหน้าต่าง

โดยปกติแล้วการวาดภาพในหน้าต่างกราฟิกของ **R** จะวาดแต่ละรูปเต็มพื้นที่หน้าต่าง แต่มีบางครั้งที่เราอาจต้องการวาดรูปหลายรูปลงในหน้าต่างเดียวกัน ก็สามารถใช้ตัวเลือกนี้ได้ ดูตัวอย่างการวาดกราฟิกหลายกราฟในหน้าต่างเดียวกันดังในรูปที่ 9.1



รูปที่ 9.1 ตัวอย่างการวาดกราฟิกสี่รูปในหน้าต่าง x window เดียวกัน

ตัวอย่างข้างต้น เป็นการวาดภาพกราฟิกสองแถวและสองสดมภ์ในหน้าต่างเดียวกัน การที่จะทำเช่นนี้ได้ จะใช้คำสั่ง `par(mfrow)` ซึ่งในที่นี้การสั่งให้แสดงกราฟิกสองแถวสองสดมภ์ จะออกคำสั่งเป็น `par(mfrow=c(2,2))`

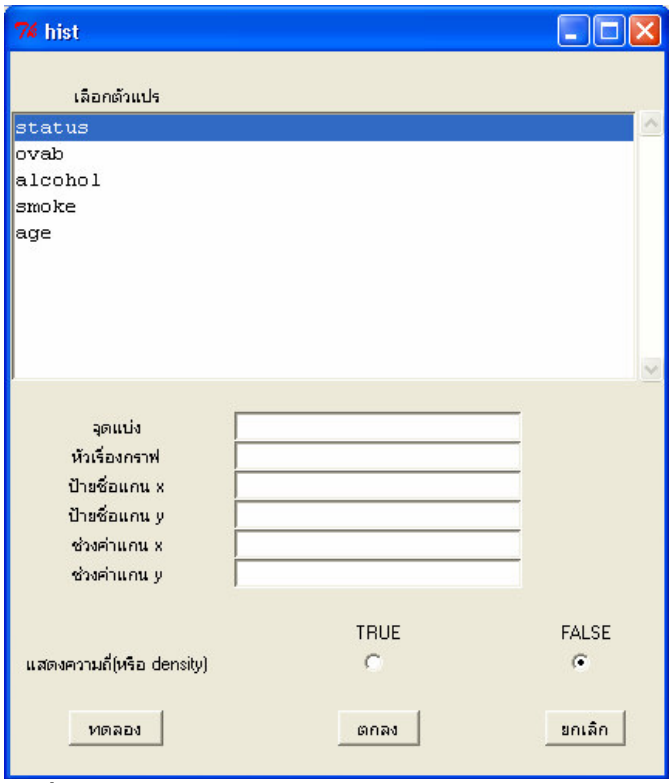
สำหรับตัวเลือกนี้เมื่อเลือกแล้ว ก็จะมีหน้าต่างให้กรอกว่าต้องการวาดกราฟิกกี่แถวและกี่สดมภ์ในหนึ่งหน้าต่าง เนื่องจากไม่ได้มีอะไรที่ต้องอธิบายมาก จึงจะไม่แสดงรูปหน้าต่างในที่นี้

ทำกราฟ ***histogram***

กราฟ histogram เป็นกราฟิกพื้นฐานที่มักจะต้องทำกันบ่อยครั้ง เพื่อบรรยายการกระจายของข้อมูลที่มีลักษณะเป็นข้อมูลต่อเนื่อง เช่น อายุ น้ำหนัก ส่วนสูง รายได้ เป็นต้น การทำกราฟ histogram จะทำให้เห็นการกระจายด้วยตา อย่างไรก็ตามการจะให้แน่ใจว่าการกระจายเป็นแบบปกติหรือไม่ ก็จำเป็นต้องมีวิธีการทดสอบทางสถิติ ซึ่งจะได้กล่าวต่อไป

เมื่อเลือกตัวเลือกนี้ จะมีทางให้เลือกสองทาง ทางหนึ่งคือเลือกทำ histogram ของเวกเตอร์ กับการเลือกทำ histogram ของตัวแปรในกรอบข้อมูล หน้าต่างตัวเลือกของทั้งสองทางเลือกจะคล้ายกัน จึงจะกล่าวแต่เฉพาะการเลือกตัวแปรในกรอบข้อมูลเพียงอย่างเดียวเท่านั้น หากเลือกกรอบข้อมูล ก็จะมีตัวเลือกให้เลือกตัวแปรภายในอีกครั้งหนึ่ง ถึงตรงนี้ให้เลือกตัวแปรที่เป็นค่าต่อเนื่อง อย่าเลือกตัวแปรที่เป็นรหัส เพราะไม่สมควรที่จะนำตัวแปรที่เป็นค่ารหัสมาทำกราฟ histogram

ตัวเลือกที่ปรากฏบนหน้าต่างนี้มีดังต่อไปนี้ “จุดแบ่ง” “หัวเรื่องกราฟ” “ป้ายชื่อแกน x” “ป้ายชื่อแกน y” “ช่วงค่าแกน x” และ “ช่วงค่าแกน y” ซึ่งจะมีช่องให้กรอกข้อความเข้าไป นอกจากนี้ด้านล่างสุดเป็นตัวเลือกให้ แสดงความถี่ (หรือ density) ซึ่งจะต้องเลือก TRUE หรือ FALSE ดังรูปที่ 9.2



รูปที่ 9.2 หน้าต่างตัวเลือกสำหรับรายการ “ทำกราฟ histogram”

“จุดแบ่ง” เป็นตัวเลือกที่ค่อนข้างทำความเข้าใจยาก ในที่นี้จะกล่าวเฉพาะวิธีที่ง่ายที่สุดคือการบอกเป็นเวกเตอร์ของจุดแบ่งของอัตรภาคชั้นนั่นเอง โดยปกติ **R** จะหาจุดแบ่งที่เหมาะสมให้โดยอัตโนมัติ แต่หากไม่พอใจก็อาจระบุเองได้ โดยเขียนในรูปเวกเตอร์ด้วยฟังก์ชัน `c()` เช่น `c(0,10,20,30,40,50)` เป็นต้น

สำหรับตัวเลือกป้ายชื่อแกน x และ y นั้นคือป้ายชื่อหรือ label ที่จะให้กับแกน x และ y นั่นเอง หรือคือค่าที่ให้กับตัวเลือก `xlab` และ `ylab` สามารถพิมพ์ข้อความที่ต้องการเข้าไปได้

โดยตรง ส่วนตัวเลือกช่วงค่าแกน x และ y นั้น คือค่าที่ให้กับตัวเลือก `xlim` และ `ylim` ให้พิมพ์เป็นเวกเตอร์ตัวเลขสองตัว ตัวหน้าคือค่าต่ำสุด และตัวหลังคือค่าสูงสุดของแกน ดังเช่น `c(0,100)` ก็จะเป็นการกำหนดให้ช่วงค่าต่ำสุดของแกนเป็น 0 และค่าสูงสุดของแกนเป็น 100

ด้านล่างสุดมีปุ่ม 3 ปุ่ม คือ “ทดลอง”, “ตกลง” และ “ยกเลิก” ปุ่ม “ทดลอง” เป็นปุ่มที่ใช้ทดสอบการเปลี่ยนแปลงค่าตัวเลือกต่างๆ ทั้งนี้กราฟจะเปลี่ยนไปตามตัวเลือกที่เลือกไว้ แต่ยังไม่ส่งคำสั่งไปที่จอ R console แต่อย่างใด เราสามารถเปลี่ยนค่าตัวเลือกเป็นอย่างอื่นได้ถ้าไม่ชอบใจ เมื่อทุกอย่างเป็นดังที่ต้องการแล้ว จึงกดปุ่ม “ตกลง” ภาพกราฟิกจะถูกสร้างในหน้าต่างอีกครั้ง แล้วประโยคคำสั่งสุดท้ายนั้นจะถูกส่งไปยัง R console พร้อมให้บันทึกลงแฟ้มคริปต์ได้

หากต้องการเปลี่ยนแปลงหรือเพิ่มเติมตัวเลือกมากกว่าที่มีให้ในหน้าต่าง ก็สามารถทำได้โดยแก้ไขประโยคคำสั่งที่ได้ โดยอ่านวิธีใช้ค่า `par()` อื่นๆ เพิ่มเติมได้จาก `help()` คำสั่ง `histo()` และ `par()` บรรทัดคำสั่งที่เป็นโครงร่างของตัวเลือกนี้คือ

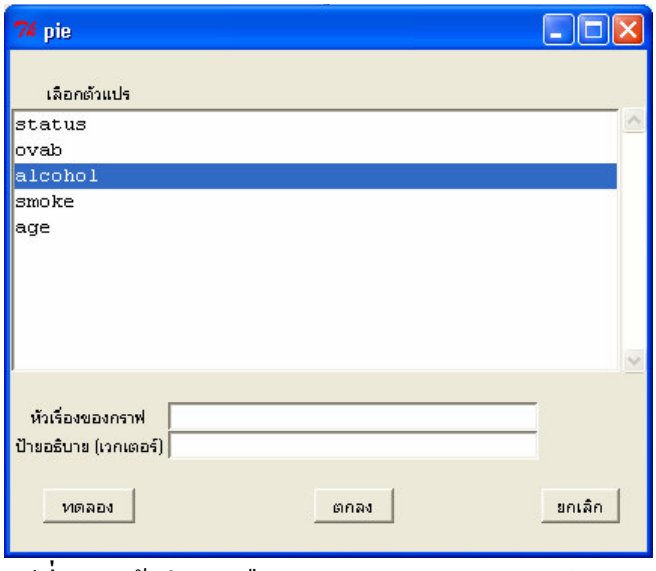
```
graph>> hist(x, breaks=, freq=FALSE, main=, xlim=, ylim=, xlab=,
ylab=)
```

ทำกราฟ ***pie***

กราฟ pie ก็เป็นกราฟที่มักทำบ่อยเช่นกัน เมื่อมีข้อมูลเป็นแบบกลุ่ม เช่น เพศ เชื้อชาติ ศาสนา การทำกราฟ pie จะทำให้เห็นภาพการกระจายของกลุ่มต่างๆ ในตัวแปรนั้นได้ชัดเจนขึ้น

เมื่อเลือกตัวเลือกนี้ จะมีทางให้เลือกสองทาง ทางหนึ่งคือเลือกทำกราฟ pie จากเวกเตอร์โดยตรง กับการเลือกทำกราฟ pie จากตัวแปรในกรอบข้อมูล หากเลือกกรอบข้อมูล ก็จะมีตัวเลือกให้เลือกตัวแปรภายในอีกครั้งหนึ่ง ถึงตรงนี้ให้เลือกตัวแปรที่เป็นรหัสหรือกลุ่ม อย่าเลือกตัวแปรที่เป็นค่าต่อเนื่อง เพราะไม่สมควรที่จะนำตัวแปรที่เป็นค่าต่อเนื่องมาทำกราฟ pie เพราะจะได้ชิ้นของ pie จำนวนมากจนดูไม่ออก

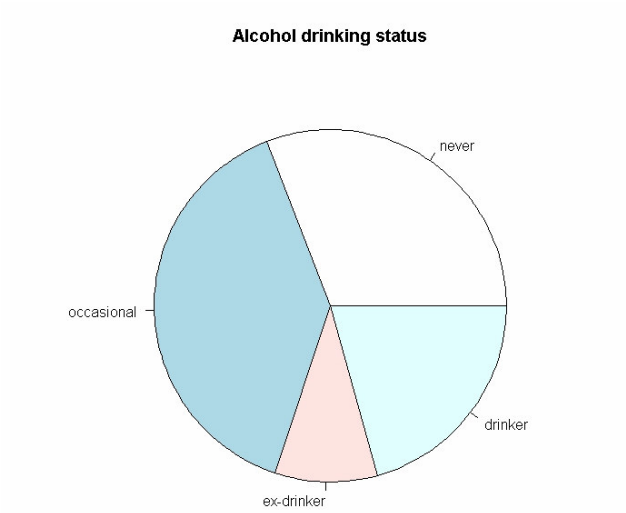
ตัวเลือกที่ปรากฏบนหน้าต่างนี้มีดังต่อไปนี้ “หัวเรื่องของกราฟ” และ “ป้ายอธิบาย (เวกเตอร์)” ซึ่งจะมีช่องให้กรอกข้อความเข้าไป ซึ่งก็คล้ายกับในหัวข้อการทำกราฟ histogram ที่กล่าวข้างต้นดังรูปที่ 9.3



รูปที่ 9.3 หน้าต่างตัวเลือกสำหรับรายการ “ทำกราฟ pie”

ด้านล่างสุดมีปุ่ม 3 ปุ่มเช่นเดียวกับหน้าต่างของรายการทำกราฟ histogram คือ “ทดลอง” “ตกลง” และ “ยกเลิก” ปุ่ม “ทดลอง” เป็นปุ่มที่ใช้ทดสอบการเปลี่ยนแปลงค่าตัวเลือกต่างๆ ทั้งนี้ กราฟจะเปลี่ยนไปตามตัวเลือกที่เลือกไว้ แต่ยังไม่ส่งคำสั่งไปที่จอ R console แต่อย่างใด เราสามารถเปลี่ยนค่าตัวเลือกเป็นอย่างอื่นได้ถ้าไม่ชอบใจ เมื่อทุกอย่างเป็นดังที่ต้องการแล้ว จึงกดปุ่ม “ตกลง” ภาพกราฟิกจะถูกสร้างในหน้าต่างอีกครั้ง แล้วประโยคคำสั่งสุดท้ายนั้นจะถูกส่งไปยัง R console พร้อมให้บันทึกลงแฟ้มคริปต์ได้

หากต้องการเปลี่ยนแปลงหรือเพิ่มเติมตัวเลือกมากกว่าที่มีให้ในหน้าต่าง ก็สามารถทำได้โดยแก้ไขประโยคคำสั่งที่ได้ โดยอ่านวิธีใช้ค่า `par()` อื่นๆ เพิ่มเติมได้จาก `help()` คำสั่ง `pie()` และ `par()`



รูปที่ 9.4 ผลของการเลือกรายการ “ทำกราฟ pie” จากข้อมูลตัวแปร alcohol ในกรอบข้อมูล cca

บรรทัดคำสั่งของตัวเลือกนี้ประกอบด้วยสามประโยคคำสั่งย่อย ประโยคแรกเป็นการสร้างเวกเตอร์ `.xVal` ขึ้นโดยการทำตารางจากเวกเตอร์หรือตัวแปรในกรอบข้อมูลที่เลือก เช่นในตัวอย่างเลือกตัวแปร alcohol ที่อยู่ในกรอบข้อมูล cca เมื่อทำเช่นนี้ **R** จะรู้ขนาดหรือสัดส่วนของแต่ละกลุ่มในตัวแปรเพื่อนำมาสร้างเป็นกราฟ pie จากนั้นค้นหาชื่อของกลุ่มต่างๆ ในเวกเตอร์นั้น โดยใช้ฟังก์ชัน `names()` แล้วนำไปใส่ให้กับเวกเตอร์ `.xVal` จากนั้นจึงนำเวกเตอร์ `.xVal` นั้นไปสร้างกราฟ pie โดยใช้ฟังก์ชัน `pie()` ขอให้สังเกตว่าเราไม่สามารถสร้างกราฟ pie ได้จากเวกเตอร์ตัวเลขหรือแฟกเตอร์ได้โดยตรง ต้องสร้างมาจากวัตถุชนิด table อีกทอดหนึ่ง จึงต้องผ่านกระบวนการหลายขั้นตอนดังข้างต้น

```
graph>> .xVal <- as.vector(table(cca$alcohol));
names(.xVal) <- names(table(cca$alcohol));
pie(.xVal, labels=, main="Alcohol drinking status")
```

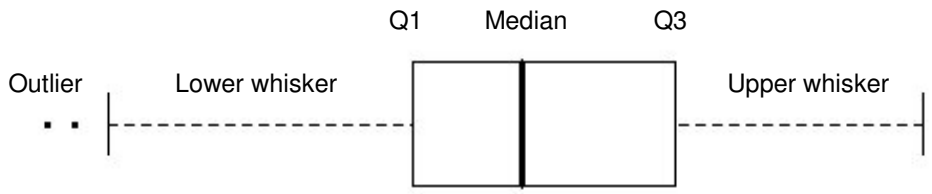
ทำกราฟ ***boxplot***

กราฟ boxplot หรือ box-and-whisker ก็เป็นกราฟที่มักใช้บ่อยเช่นกัน เมื่อมีข้อมูลเป็นค่าต่อเนื่อง และต้องการดูการกระจายของตัวแปรนั้น ซึ่งอาจจะแบ่งตามกลุ่มต่างๆ หรือไม่ก็ได้

เมื่อเลือกตัวเลือกนี้ จะมีทางให้เลือกสามทาง ทางหนึ่งคือเลือกทำกราฟ boxplot จาก formula ทางสถิติทางหนึ่ง อีกทางหนึ่งคือทำกราฟ boxplot จากเวกเตอร์โดยตรง หรือจะเลือกทำกราฟ boxplot จากตัวแปรในกรอบข้อมูล หากเลือกกรอบข้อมูล ก็จะมีตัวเลือกให้เลือกตัวแปร

ภายในอีกครั้งหนึ่ง ตัวเลือกสองตัวหลังเป็นการทำกราฟ boxplot สำหรับตัวแปรเดียว โดยไม่มีการแบ่งกลุ่ม แต่หากต้องการดูการกระจายโดยแบ่งกลุ่มแล้ว จะต้องเลือกตัวเลือกแรกเท่านั้น

ถึงตรงนี้จะได้อธิบายลักษณะของกราฟ boxplot โดยสังเขปก่อน กราฟ boxplot เป็นกราฟที่แสดงค่า 5 ค่า คือ ค่าน้อยที่สุด, quatile ล่าง, มัธยฐาน, quatile บน และค่ามากสุด และนอกจากนี้ยังแสดงค่าที่ตกนอกช่วงการกระจายด้วย (หากมี) โดยแสดงเป็นรูปกล่อง ซึ่งแสดง quatile บนและล่างเป็นรูปกล่อง หรือ box ซึ่งภายในจะมีเส้นที่แสดงค่ามัธยฐาน (median) อีกเส้นหนึ่ง ซึ่งอาจไม่เห็นก็ได้ถ้าการกระจายมีการเบ้มากๆ และจำนวนตัวอย่างมีน้อย และเส้นหนวด (whisker) ออกไปสองข้าง เพื่อแสดงช่วงไปถึงค่าน้อยสุดและมากสุดของการกระจาย อย่างไรก็ตามบางครั้งอาจมีบางค่าตกนอกช่วงการกระจาย ในกรณีนี้จะแสดงเป็นจุดที่เลยไปจากปลายของเส้นหนวด ซึ่งเรียกว่า outlier



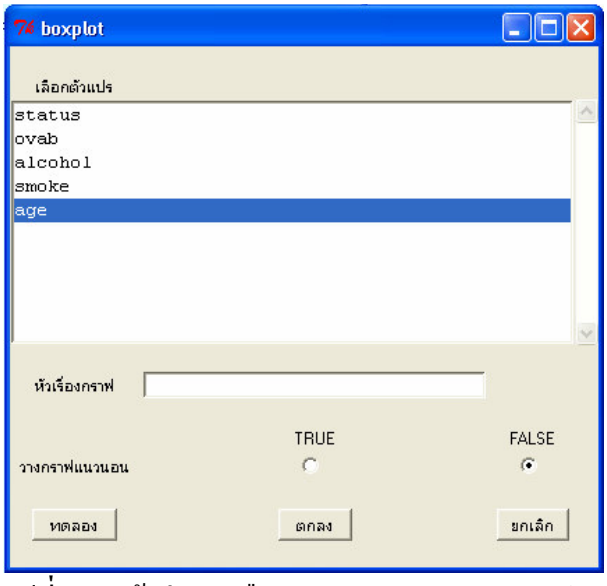
รูปที่ 9.5 ลักษณะของกราฟ boxplot

สำหรับ formula ทางสถิตินั้นมีที่ใช้มากในการทดสอบทางสถิติซึ่งจะกล่าวในบทต่อไป และการทำ regression ใน **R** จะมีการเขียนในลักษณะดังนี้

```
y ~ a + b + …
```

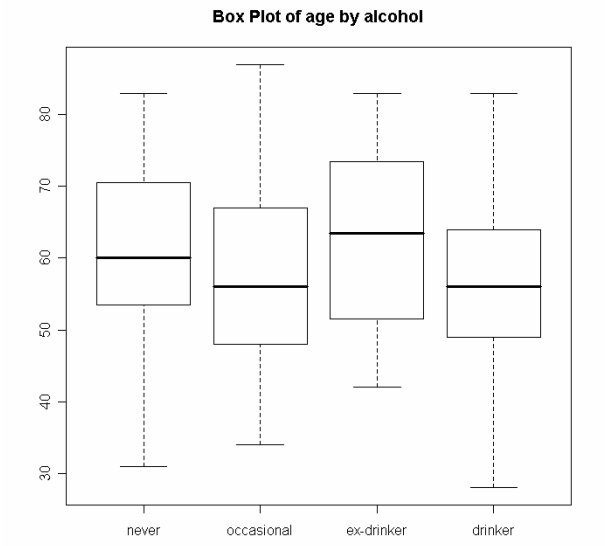
โดยที่ y เป็นตัวแปรตาม หรือในกรณีของการวาดกราฟก็จะเป็นตัวแปรที่จะพล็อต เครื่องหมาย ~ เป็นการบอกการเป็น formula ทางสถิติ ส่วนตัวแปรทางด้านขวามือเป็นตัวแปรต้น หรือในกรณีการวาดกราฟหรือในการทดสอบทางสถิติบางอย่างเช่น ANOVA ก็จะเป็นตัวแปรแบ่งกลุ่ม

ตัวเลือกที่ปรากฏบนหน้าต่างนี้มีดังต่อไปนี้ “หัวเรื่องกราฟ” ซึ่งจะมีช่องให้กรอกข้อความเข้าไป และตัวเลือก “วางกราฟแนวนอน” ซึ่งให้เลือกว่า TRUE หรือ FALSE นั่นคือหากเลือก TRUE กราฟจะวางในแนวนอน ค่าปกติเป็น FALSE คือกราฟวางในแนวตั้ง ดังรูปที่ 9.6



รูปที่ 9.6 หน้าต่างตัวเลือกสำหรับรายการ “ทำกราฟ boxplot” จากตัวแปรในกรอบข้อมูล

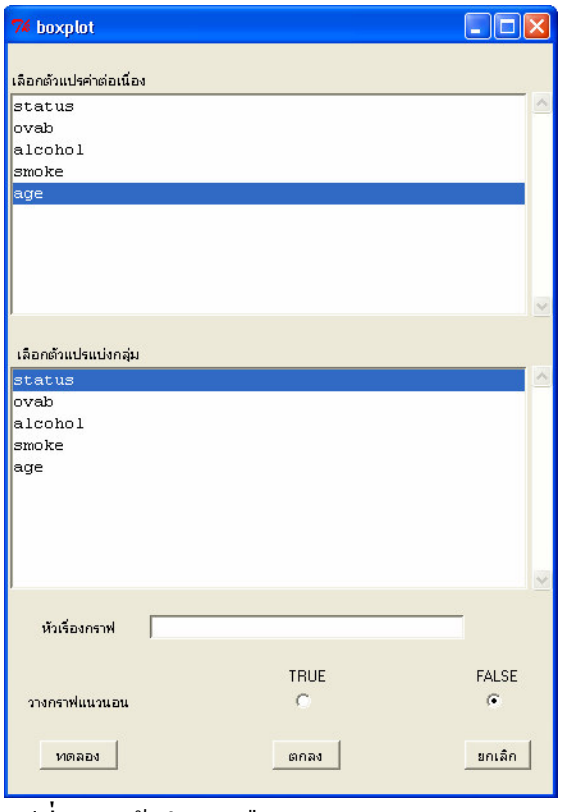
แต่หากเลือกตัวเลือกทำกราฟ boxplot จาก formula ทางสถิติแล้ว ด้านล่างของรายการเลือกตัวแปรที่จะมาทำ boxplot จะมีรายการให้เลือก “ตัวแปรแบ่งกลุ่ม” อีกด้วย ซึ่งเมื่อเลือกการแบ่งกลุ่มแล้ว ตัวแปรที่เลือกข้างต้นจะถูกทำเป็น boxplot หลายอัน เท่ากับจำนวนกลุ่มของตัวแปรนี้ ทำให้เราสามารถเปรียบเทียบการกระจายของตัวแปรตามกลุ่มต่างๆ ได้ ดังรูปที่ 9.7



รูปที่ 9.7 ตัวอย่างกราฟ boxplot ที่มีการแบ่งกลุ่ม

ลองดูประโยคคำสั่งที่ได้จากการเลือกตัวเลือกทำกราฟ boxplot จาก formula ข้างล่าง ซึ่งอาจทำตามได้โดยเลือกรายการ วัตถุ >> ตัวอย่างชุดข้อมูล >> cca จากโมดุล ice.main เพื่อเรียกใช้

ชุดข้อมูลตัวอย่างที่มีมาให้แล้ว (ดูความหมายของตัวแปรต่างๆ ในภาคผนวก) ลองเลือกทำ boxplot ระหว่าง age กับตัวแปร alcohol โดยใช้ตัวเลือก Boxplot >> formula ดังรูปที่ 9.8



รูปที่ 9.8 หน้าต่างตัวเลือกสำหรับรายการ “ทำกราฟ boxplot” จากตัวเลือก formula

เมื่อกดปุ่ม “ตกลง” จะได้ประโยคคำสั่งต่อไปนี้ และได้ผลการทำกราฟในหน้าต่างกราฟิกด้วย

```
graph>> boxplot(age ~ alcohol, data=cca, main="Box Plot of age by
alcohol", horizontal=FALSE)
```

ในตารางที่ 9.2 เป็นการสรุปฟังก์ชันและรูปแบบการใช้งานของรายการต่างๆ ของโมดุล ice.graph ซึ่งยังมีไม่มากนัก ค่าปกติที่มีให้เลือกของแต่ละรายการก็ยังคงมีไม่มากนัก แต่จะได้ปรับปรุงให้มากขึ้นและใช้งานได้สะดวกขึ้นในรุ่นถัดๆ ไป

ตารางที่ 9.2 สรุปฟังก์ชันและรูปแบบการใช้งานของรายการต่างๆ ในโมดุล ice.graph

รายการ	ฟังก์ชัน	รูปแบบการใช้งานฟังก์ชัน
หน้าต่าง graphic ใหม่	`X11()`	`X11(width = 7, height = 7)`
หลายกราฟในหนึ่งหน้าต่าง	`par(mfrow)`	`par(mfrow=c(1,1))`

Histogram	`hist()`	`hist(x, breaks = "Sturges", freq = FALSE, main = paste("Histogram of" , xname), xlim = range(breaks), ylim = NULL, xlab = xname, ylab)`
Pie	`pie()`	`.xVal <- as.vector(table(x)); names(.xVal) <- names(table(x)); pie(.xVal, labels = names(.xVal), main = NULL)`
Boxplot	`boxplot()`	`boxplot(x, main = paste("Box Plot of", xname), horizontal = FALSE)`

บทที่ 10 โมดุล **ice.statis** ในการทดสอบทางสถิติพื้นฐาน

ช่วงความเชื่อมั่น

ทดสอบ Fisher’s exact และทดสอบ chi-squared

ทดสอบ anova

ทดสอบ Student t และทดสอบ Wilcoxon/Mann-Whitney

ทดสอบ Shapiro Wilk

ทดสอบ likelihood ratio

ในการทำงานทางสถิตินั้น สิ่งที่สำคัญที่สุดคงไม่พ้นการทดสอบทางสถิติ ไม่ว่าจะเป็นการทดสอบนัยสำคัญ การหาช่วงความเชื่อมั่น การทดสอบโมเดลทางสถิติ และอื่นๆ **R** ได้ถูกสร้างมาเพื่อประโยชน์ของงานนี้เป็นสำคัญ จึงมีการทดสอบและโมเดลทางสถิติให้ใช้มากมายจนไม่สามารถบรรจุในรายการเมนูได้ทั้งหมด รายการในเมนูของ **R-ICE** นี้จึงมีแต่การทดสอบทางสถิติอย่างพื้นๆ เท่านั้น ส่วนการทดสอบที่ซับซ้อนมากขึ้นและการใช้โมเดลทางสถิติจะได้ทำเป็นโมดุลต่างหากในอนาคต

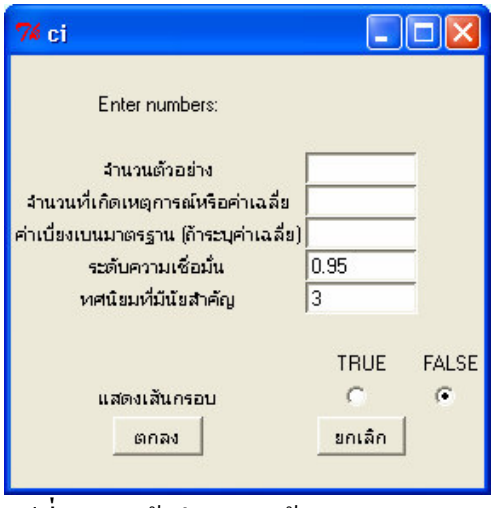
ตารางที่ 10.1 โครงสร้างของรายการเมนู ice.statis

ข้อมูล	**ช่วยเหลือ**
ช่วงความเชื่อมั่น	ฟังก์ชัน
คำนวณจากตัวเลข	ci
เวกเตอร์	fisher.test
กรอบข้อมูล	chisq.test
ทดสอบ Fisher's exact	anova.test
แฟกเตอร์	st.test
เมทริกซ์	wilcoxon.test
ในกรอบข้อมูล	shapiro.wilk.test
ทดสอบ chi-squared	lr.test
แฟกเตอร์	ตัวอย่าง
เมทริกซ์	ci
ในกรอบข้อมูล	fisher.test
ทดสอบ anova	chisq.test
ทดสอบ Student t	anova.test
formula	st.test
สองเวกเตอร์	wilcoxon.test
ในกรอบข้อมูล	shapiro.wilk.test
ทดสอบ Wilcoxon/Mann-Whitney	lr.test
formula	เกี่ยวกับหน้าต่าง statis
สองเวกเตอร์	text
ในกรอบข้อมูล	html
ทดสอบ Shapiro Wilk	chm
เวกเตอร์	เกี่ยวกับแพคเกจ statis
ในกรอบข้อมูล	
ทดสอบ likelihood ratio	

ช่วงความเชื่อมั่น

ปัจจุบันนิยมบอกค่าประมาณของสิ่งใดสิ่งหนึ่งในรูปของช่วงความเชื่อมั่น มักไม่นิยมบอกเป็นค่าที่แน่นอน กับค่า p-value ที่หาจากการทดสอบ Fisher's exact หรือ chi-squared กันแล้ว ช่วงความเชื่อมั่นของค่าใดค่าหนึ่งเท่ากับ $A \pm Z_{\alpha/2}$ SE โดยที่ A คือค่าประมาณของค่านั้น เช่นค่าสัดส่วน ค่าเฉลี่ย หรืออื่นๆ $Z_{\alpha/2}$ คือค่าการกระจายปกติที่ให้ค่า probability น้อยกว่าหรือเท่ากับ $\alpha/2$ และ SE คือค่า standard error ในที่นี้จะไม่กล่าวรายละเอียดในการคำนวณ ผู้สนใจอาจหาได้จากตำราทางสถิติทั่วไปได้

เมื่อเลือกรายการนี้ จะมีทางให้เลือกสามทางเลือกคือ "คำนวณจากตัวเลข" "เวกเตอร์" และ "กรอบข้อมูล" ในกรณีที่เรามีค่าที่รู้อยู่แล้ว ตัวอย่างเช่น ต้องการทราบว่าช่วงความเชื่อมั่นของสัดส่วนของคนที่รู้สึกพอใจกับการบริการจากการสอบถามผู้ใช้บริการ 125 คน มีคนตอบว่าพอใจอยู่ 89 คน เมื่อเลือกรายการนี้จะปรากฏหน้าต่างดังรูปที่ 10.1 ในช่อง "จำนวนตัวอย่าง" ก็ใส่ค่า 125 ลงไป และในช่อง "จำนวนที่เกิดเหตุการณ์หรือค่าเฉลี่ย" ในที่นี้ให้ใส่จำนวนที่เกิดเหตุการณ์ลงไป ซึ่งก็คือค่า 89 ส่วนในช่อง "ค่าเบี่ยงเบนมาตรฐาน (ถ้าระบุค่าเฉลี่ย)" ให้เว้นไว้ เนื่องจากในกรณีนี้เราไม่มีค่าเฉลี่ยของเหตุการณ์ แต่เป็นจำนวนเหตุการณ์ที่เกิดขึ้น ค่าปกติของช่วงความเชื่อมั่นเป็นร้อยละ 95 ปกติก็ไม่ต้องเปลี่ยนค่า ส่วนจำนวนทศนิยมนั้น ค่าปกติที่ให้ไว้คือ 3 ตำแหน่ง



รูปที่ 10.1 หน้าต่างกรอกข้อมูลตัวเลขสำหรับการคำนวณช่วงความเชื่อมั่น

จากตัวอย่างข้างต้น เมื่อกดปุ่ม "ตกลง" ที่หน้าต่าง R console จะมีข้อความดังนี้

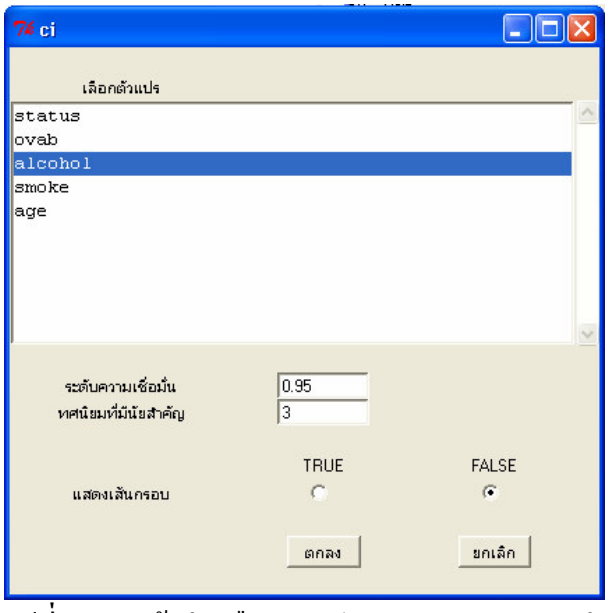
```
statis>> ci(125, 89, , level=0.95, digits=3, frame=FALSE)
 Obs. Prob.  S.E. Lower Upper
  125 0.712 0.041 0.624 0.789
```

นั่นคือจะมีการแสดงประโยคคำสั่ง แล้วตามด้วยผลลัพธ์ของการคำนวณค่าสัดส่วน และช่วงความเชื่อมั่นร้อยละ 95 นั่นคือสัดส่วนของผู้ที่พอใจเป็น 0.712 หรือร้อยละ 71.2 และช่วงความเชื่อมั่นร้อยละ 95 อยู่ในช่วงร้อยละ 62.4 ถึง 78.9 นั่นเอง ซึ่งในทางสากลนิยมเขียนในรูปวงเล็บดังนี้ 0.712 (95%CI: 0.624 – 0.789) หรือ 71.2% (95%CI: 62.4% - 78.9%)

หากรู้ค่าเฉลี่ยของค่าใดค่าหนึ่ง เช่น อายุเฉลี่ย ก็จะต้องบอกค่าเบี่ยงเบนมาตรฐานด้วย เช่น ในกรณีเดิม ถ้าอายุของผู้ตอบแบบสอบถามเป็น 37 ปี ค่าเบี่ยงเบนมาตรฐานเป็น 10.5 ปี ผลลัพธ์จะเป็นดังนี้

```
statis>> ci(125, 37, 10.5, level=0.95, digits=3, frame=FALSE)
  Obs.    Mean    S.E.  Lower  Upper
   125      37  0.939 35.141 38.859
```

โดยปกติแล้วเรามักกรอกข้อมูลลงในกรอบข้อมูล โดยไม่นับหรือคำนวณค่ามาก่อน แต่ต้องการให้โปรแกรมคำนวณให้จากตัวแปรในกรอบข้อมูลเลย ซึ่งทำได้โดยเลือกตัวเลือก “กรอบข้อมูล” ซึ่งก็จะมีหน้าต่างให้เลือกกรอบข้อมูลที่ต้องการทำงานด้วย แล้วมีหน้าต่างให้เลือกตัวแปรที่ต้องการคำนวณค่าดังรูปที่ 10.2



รูปที่ 10.2 หน้าต่างเลือกตัวแปรสำหรับการคำนวณช่วงความเชื่อมั่น

ตัวเลือกนี้จะพิจารณาว่าตัวแปรนั้นเป็นแฟกเตอร์หรือเวกเตอร์ตัวเลข และคำนวณค่าที่เหมาะสมให้ ลองใช้ชุดข้อมูลตัวอย่าง cca โดยการเลือกจากรายการเมนู ice.main โดยเลือกรายการวัตถุ >> ตัวอย่างชุดข้อมูล >> cca แล้วเลือกคำนวณช่วงความเชื่อมั่นของตัวแปร age และ alcohol จะได้ผลดังนี้

```
statis>> ci(cca, age, level=0.95, digits=3, frame=FALSE)
```

```
age
  Obs.   Mean S.E.  Lower  Upper
   262 58.989 0.75 57.519 60.459
statis>> ci(cca, alcohol, level=0.95, digits=3, frame=FALSE)
alcohol
           Obs. Prop.  S.E. Lower Upper
never        79 0.309 0.001 0.254 0.367
occasional  100 0.391 0.001 0.332 0.449
ex-drinker   24 0.094 0.000 0.059 0.133
drinker      53 0.207 0.001 0.160 0.258
```

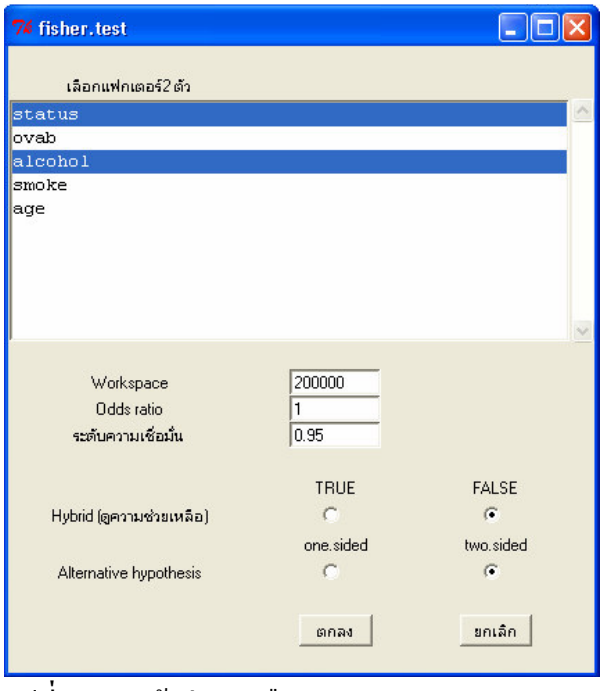
เมื่อพบว่าในกรอบข้อมูล cca นั้น ตัวแปร age เป็นเวกเตอร์ตัวเลข ก็จะคำนวณค่าเฉลี่ยให้ และเมื่อพบว่าตัวแปร alcohol เป็นแฟกเตอร์ ก็จะคำนวณค่าสัดส่วนของแต่ละระดับให้

ทดสอบ ***Fisher's exact*** *และทดสอบ* ***chi-squared***

รายการทดสอบ Fisher's exact และทดสอบ chi-squared มีความคล้ายคลึงกันมาก ใช้ในกรณีที่ตัวแปรเป็นแบบกลุ่มหรือแฟกเตอร์เท่านั้น ไม่ใช้กับกรณีที่ตัวแปรเป็นค่าต่อเนื่อง โดยที่การทดสอบ Fisher's exact มีข้อกำหนดว่าตัวแปรใดตัวหนึ่งต้องมีกลุ่มเพียงสองกลุ่ม (dichotomous หรือ binary) เท่านั้น เช่นในกรณีการทดสอบความสัมพันธ์ระหว่างการดื่มเหล้ากับการเป็นมะเร็งตับในกรอบข้อมูลตัวอย่าง cca ตัวแปร alcohol แบ่งเป็น 4 กลุ่มหรือระดับ ส่วนตัวแปร status นั้นมี 2 ระดับ ในกรณีนี้เราสามารถทดสอบ Fisher's exact ได้ ส่วนการทดสอบ chi-squared นั้นไม่มีข้อจำกัดดังกล่าว เช่น เราสามารถทดสอบความสัมพันธ์ระหว่างตัวแปร alchol และ smoke ซึ่งมี 4 ระดับทั้งคู่

โดยทั่วไปแล้ว ค่า p-value ที่ได้จากการทดสอบ Fisher's exact มีความถูกต้องมากกว่าการทดสอบ chi-squared เพราะการทดสอบ chi-squared เป็นการประมาณค่า p-value ที่จะได้จากการทดสอบ Fisher's exact นั่นเอง ทั้งนี้เพราะในสมัยที่ยังไม่มีเครื่องคอมพิวเตอร์ใช้ การคำนวณ Fisher's exact probability นั้นยุ่งยาก ใช้เวลานานอย่างยิ่ง แต่ในปัจจุบันเราสามารถใช้คอมพิวเตอร์คำนวณได้ในเวลารวดเร็วพอๆ กัน ดังนั้นในกรณีที่สามารถทดสอบได้ทั้งสองอย่าง ให้เลือกทดสอบ Fisher's exact

เมื่อเลือกรายการหนึ่งรายการใดในทั้งสองนี้ จะมีทางให้เลือกสามทางเลือกคือ "แฟกเตอร์" "เมทริกซ์" และ "ในกรอบข้อมูล" การทดสอบกับแฟกเตอร์หรือตัวแปรในกรอบข้อมูลนั้น จะต้องเลือกแฟกเตอร์หรือตัวแปรสองตัว โดยเมื่อเรียกใช้ตัวอย่างชุดข้อมูล cca แล้วเลือกตัวเลือกนี้ จะได้หน้าต่างให้เลือกตัวแปรดังรูปที่ 10.3



รูปที่ 10.3 หน้าต่างตัวเลือกของรายการ ทดสอบ Fisher's exact >> ในกรอบข้อมูล

ผลของการเลือกแสดงในหน้าจอ R console ดังนี้

```
statis>> fisher.test(cca$status, cca$alcohol, workspace=200000, hybr$

        Fisher's Exact Test for Count Data

data:  cca$status and cca$alcohol
p-value = 9.134e-06
alternative hypothesis: two.sided
```

และหากเลือกรายการทดสอบ chi-squared จะได้ผลดังนี้

```
statis>> chisq.test(cca$status, cca$alcohol)

        Pearson's Chi-squared test

data:  cca$status and cca$alcohol
X-squared = 25.5796, df = 3, p-value = 1.168e-05
```

ซึ่งทั้งสองกรณีให้ค่า p-value ต่ำมากๆ เช่นกัน แต่ค่าที่คำนวณได้ไม่เท่ากัน ซึ่งในกรณีนี้ค่า p-value จากการทดสอบ Fisher's exact มีความถูกต้องมากกว่า

อย่างไรก็ตามการเลือกรายการทดสอบนี้จะไม่แสดงตารางสรุปข้อมูลให้เห็น ทำให้ผู้ใช้ไม่เห็นภาพการกระจายของข้อมูลตามกลุ่มต่างๆ ลองดูหัวข้อการทำตารางในบทที่ 8 จะเห็นว่าเราสามารถเลือกทดสอบ Fisher's exact และ chi-squared ได้ด้วย ทำให้สามารถเห็นทั้งการกระจายของข้อมูลและทำการทดสอบได้ในคราวเดียวกัน

แต่ถ้าเรามีข้อมูลที่ได้มาในรูปของตารางสรุปดังข้างล่าง ไม่ได้เป็นตัวแปรในกรอบข้อมูล เราจะทดสอบ Fisher's exact หรือ chi-squared อย่างไร

```
         never occasional ex-drinker drinker
control     48         60          7      13
case        31         40         17      40
```

ในกรณีเช่นนี้ เราต้องกรอกข้อมูลเหล่านี้เข้าไปในกรอบข้อมูลเปล่า ดังที่กล่าวในบทที่ 7 เรื่องโมดุล ice.dataman โดยกรอกข้อมูลไปตามแนวนอนเช่นตารางที่เห็น หรืออาจกรอกในแนวตั้งตามทิศทางของ case และ control ก็ได้ ไม่ว่าจะจัดเรียงข้อมูลแบบใด ผลการทดสอบ Fisher's exact และ chi-squared จะให้ค่า p-value เท่ากัน เมื่อได้กรอบข้อมูล new.dat แล้ว จะต้องพิมพ์คำสั่งบน R console เพื่อเปลี่ยนให้ new.dat ซึ่งขณะนี้เป็นข้อมูลชนิด data.frame ให้เป็นข้อมูลชนิด matrix โดยพิมพ์คำสั่งต่อไปนี้

```
new.dat <- as.matrix(new.dat)
```

จากนั้นเราสามารถทดสอบ Fisher's exact หรือ chi-squared ได้ โดยการเลือกตัวเลือก "เมทริกซ์" นั่นเอง ซึ่งจะให้ผลเหมือนกับที่เราทดสอบกับแฟกเตอร์หรือตัวแปรในกรอบข้อมูล ดังนี้

```
dataman>> new.dat <- create()
> new.dat
  var1 var2 var3 var4
1   48   60    7   13
2   31   40   17   40
> dat2 <- as.matrix(new.dat)
statis>> chisq.test(dat2)

        Pearson's Chi-squared test

data:  dat2
X-squared = 25.5796, df = 3, p-value = 1.168e-05
```

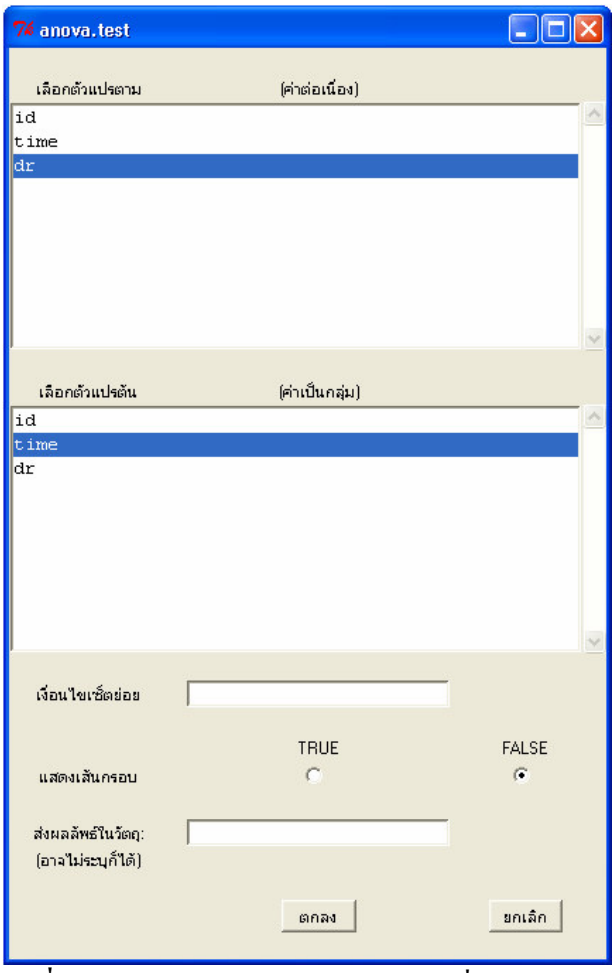
หมายเหตุ ขอให้สังเกตลักษณะของ prompt ให้ดี หากเป็น `dataman>>` นั่นหมายความว่าเป็นผลจากการใช้รายการในโมดุล ice.dataman หากเป็น `>` หมายถึงผู้ใช้พิมพ์คำสั่งบน R console โดยตรง และหากเป็น `statis>>` ก็หมายถึงผู้ใช้เลือกรายการทำงานจากโมดุล ice.statis นั่นเอง

ทดสอบ anova

การทดสอบ anova ใช้ทดสอบความต่างของข้อมูลที่เป็นค่าต่อเนื่อง และจัดเป็นกลุ่มๆ ตามตัวแปรบางอย่าง ว่าแต่ละกลุ่มมีความต่างกันหรือไม่ เช่น ความสูงของชายกับหญิงต่างกันหรือไม่ ความสูงของคนแต่ละประเทศต่างกันหรือไม่โดยปกติถ้ามีเพียงสองกลุ่มจะใช้การทดสอบ Student t

แต่เมื่อมีกลุ่มมากกว่าสองกลุ่มจะใช้การทดสอบ anova ในที่นี้จะไม่กล่าวรายละเอียดของหลักการในการทดสอบต่างๆ เนื่องจากวัตถุประสงค์ของหนังสือนี้คือการแนะนำการใช้งาน **R-ICE** ไม่ใช่เป็นหนังสือเรียนทางสถิติ ผู้สนใจด้านความรู้พื้นฐานทางสถิติของการทดสอบแบบต่างๆ หาความรู้เพิ่มเติมได้จากตำราทางสถิติทั่วไป

ก่อนอื่นให้เรียกตัวอย่างชุดข้อมูล ahcc2 มาใช้งานก่อน โดยเรียกจากรายการเมนู วัตถุ >> ตัวอย่างชุดข้อมูล เมื่อเลือกชุดข้อมูลแล้ว จึงเลือกรายการ “ทดสอบ anova” ของโมดุล ice.statis จะมีหน้าต่างให้เลือกกรอบข้อมูล จะเห็นชุดข้อมูล ahcc2 อยู่ในรายการ เมื่อเลือกชุดข้อมูล ahcc2 แล้วจะเห็นตัวแปรภายในซึ่งมีสามตัวแปร คือ id, time และ dr ตัวแปร dr เป็นค่าของการตรวจชนิดหนึ่งซึ่งทำสองครั้ง ตามเวลา time ส่วน id เป็นเลขลำดับตัวอย่าง ที่จริงชุดข้อมูลนี้ไม่ควรนำมาทำการทดสอบ anova เนื่องจากเป็นชุดข้อมูลที่ทำซ้ำในคนเดียวกันสองครั้ง ซึ่งควรใช้การทดสอบ paired Student t มากกว่า แต่ในที่นี้จะเป็นการทดลองให้เห็นขั้นตอนการทำงานและผลลัพธ์ของรายการนี้ รูปที่ 10.4 แสดงให้เห็นหน้าต่างการเลือกดังกล่าวข้างต้น



รูปที่ 10.4 หน้าต่างการเลือกตัวแปรและเงื่อนไขของการทดสอบ anova

ช่องตัวเลือกด้านบน ให้เลือกตัวแปรตาม ซึ่งเป็นค่าต่อเนื่อง ในที่นี้เลือก dr ดังที่กล่าวข้างต้น ช่องตัวเลือกด้านล่าง ให้เลือกตัวแปรต้น ที่ระบุค่าของกลุ่ม ในที่นี้เลือก time เราอาจระบุเงื่อนไขเซ็ตย่อยและอื่นๆ หรือไม่ก็ได้ เมื่อกดปุ่ม “ตกลง” จะได้ผลลัพธ์ดังนี้

```
statis>> anova.test(dr ~ time, data=ahcc2, , frame=FALSE)

Analysis of Variance Table

Response: dr
          Df  Sum Sq Mean Sq F value Pr(>F)
time       1   197.2   197.2  1.0543 0.3082
Residuals 68 12718.6   187.0
```

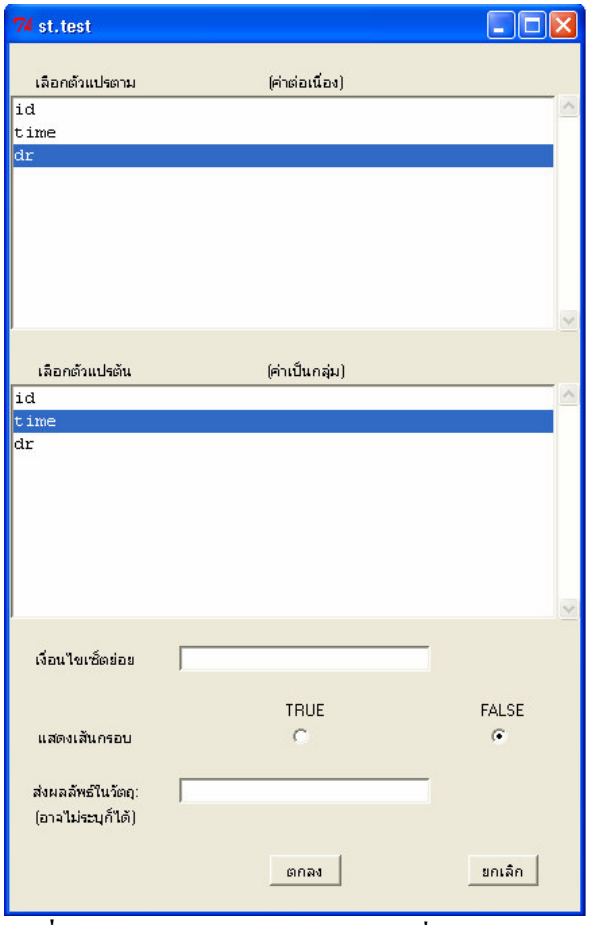
นั่นคือเราจะได้บรรทัดคำสั่งของ `anova.test()` ที่ระบุ formula และกรอบข้อมูลที่ใช้ จากนั้นแสดงผลลัพธ์ของการทำงานบรรทัดคำสั่งนั้น ซึ่งจะให้ตาราง anova นั่นเอง ในที่นี้เราพบว่าค่าการตรวจสองครั้งไม่มีความแตกต่างกันอย่างมีนัยสำคัญทางสถิติ

หมายเหตุ โปรดสังเกตว่าการทดสอบ anova มีรูปแบบการทำงานในลักษณะ formula ดังที่กล่าวในเรื่อง ทำกราฟ boxplot ในบทที่ 9 นั่นเอง โดยมี dr เป็นตัวแปรตาม และ time เป็นตัวแปรต้น เราสามารถเลือกตัวแปรต้นได้หลายตัวพร้อมกัน แต่ในที่นี้ได้เลือกเพียง time เท่านั้น เพราะไม่มีตัวแปรต้นตัวอื่น การเลือกหรือยกเลิกการเลือก ใช้วิธีคลิกบนตัวแปรนั้นเช่นกัน หากตัวแปรนั้นยังไม่ได้ถูกเลือก การคลิกก็จะเป็นการเลือก หากตัวแปรนั้นถูกเลือกอยู่แล้ว หากคลิกซ้ำก็จะเป็นการยกเลิกการเลือก

*ทดสอบ **Student t** และ **Wilcoxon/Mann-Whitney***

การทดสอบ Student t และ Wilcoxon/Mann-Whitney เป็นการทดสอบความต่างของข้อมูลที่เป็นค่าต่อเนื่อง และจัดเป็นสองกลุ่ม เพื่อหาว่าแต่ละกลุ่มมีความต่างกันหรือไม่ การทดสอบ Student t มีเงื่อนไขบางประการคือ การแจกแจงของข้อมูลแต่ละกลุ่มต้องเป็นแบบปกติ และเงื่อนไขที่ไม่จำเป็นนักคือความแปรปรวน (variance) ของแต่ละกลุ่มต้องมีค่าใกล้เคียงกัน ขณะที่การทดสอบ Wilcoxon/Mann-Whitney เป็นการทดสอบแบบ non-parametric จึงไม่มีข้อจำกัดดังกล่าว แต่การทดสอบ Student t จะให้ผลที่ถูกต้องมากกว่า ถ้าเงื่อนไขทุกอย่างเป็นไปได้แล้ว ควรเลือกใช้การทดสอบ Student t จะดีกว่า ในที่นี้จะไม่กล่าวในทางทฤษฎีของการทดสอบ แต่จะมาทดลองทำการทดสอบโดยใช้โมดุล ice.statis กัน

ลองใช้ชุดข้อมูลตัวอย่าง ahcc2 ในการทดสอบ Student t โดยทำตามขั้นตอนที่อธิบายในเรื่องการทดสอบ anova ข้างต้น เนื่องจากข้อมูลชุดนี้มีการจัดข้อมูลเพื่อการทดสอบในรูปแบบ formula ดังนั้นให้เลือกเงื่อนไขการทดสอบแบบ formula ซึ่งจะมีหน้าต่างสำหรับการเลือกดังรูปที่ 10.5



รูปที่ 10.5 หน้าต่างเลือกตัวแปรและเงื่อนไขของการทดสอบ Student t ในรูปแบบ formula

ผลของการทดสอบเป็นดังนี้

```
statis>> st.test(dr ~ time, data=ahcc2, , frame=FALSE)
dr stratified by time
  Obs. NA Mean  S.D.  Min. Median Max.
1 40   0  60.73 15.53 12   57.5   103
2 30   10 57.33 10.69 36   57.5   95

Welch Two Sample t-test: ahcc2$ dr by time
Diff. of Mean     Lower Lim.     Upper Lim.
       3.3917        -2.8680         9.6513

t = 1.0813, df = 68, Ho: diff == 0
       Ha:   diff < 0  diff != 0   diff > 0
  p-value:     0.8583     0.2834     0.1417
```

และผลของการทดสอบ Wilcoxon/Mann-Whitney เป็นดังนี้

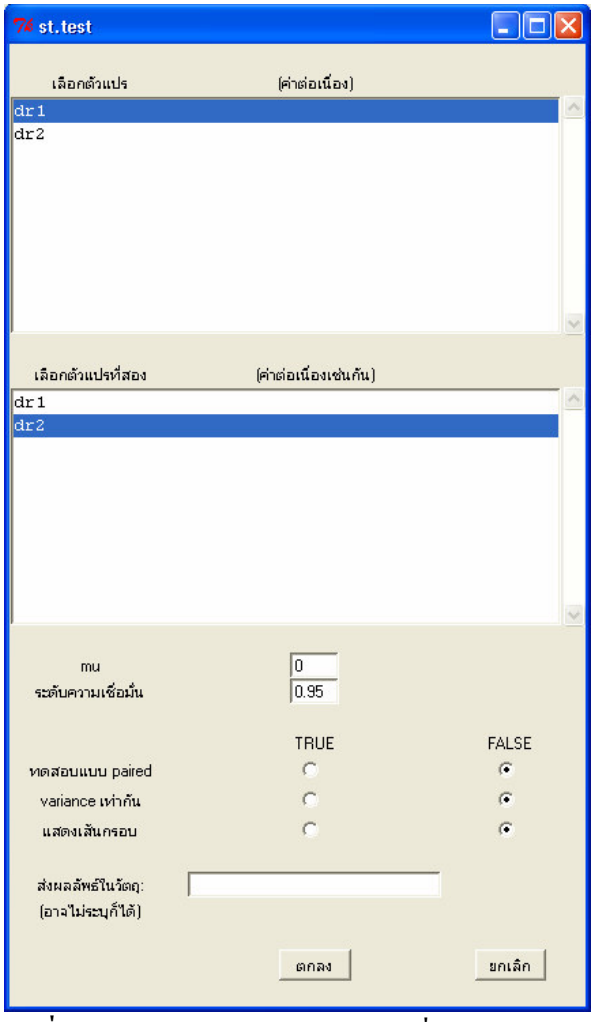
```
statis>> wilcoxon.test(dr ~ time, data=ahcc2, , frame=FALSE)
dr stratified by time
  Obs. Min. Median Max.
1 40   12   57.5   103
2 30   36   57.5   95

Wilcoxon rank sum test with continuity correction
data frame: ahcc2, variable: dr by time
V = 672.5, Ho: true mu == 0
          Ha:   true mu < 0  true mu != 0   true mu > 0
     p-value:         0.807        0.3925        0.1963
```

จะเห็นว่าทั้งบรรทัดคำสั่งและผลลัพธ์ที่แสดงของการทดสอบทั้งสอง มีหน้าตาคล้ายกันมาก และผลการทดสอบก็ให้ค่า p-value ใกล้เคียงกันด้วย แต่ในกรณีนี้เป็นการทดสอบ unpaired เท่านั้น

ส่วนบนของผลจะเป็นการสรุปข้อมูลให้เห็นภาพว่าแต่ละกลุ่มเป็นอย่างไร สำหรับการทดสอบ Student t จะสรุปให้เห็นค่าเฉลี่ยและค่าเบี่ยงเบนมาตรฐาน ส่วนการทดสอบ Wilcoxon/Mann-Whitney จะแสดงเป็นค่ามัธยฐาน ค่าน้อยสุด และค่ามากสุด เนื่องจากเป็นการทดสอบแบบ non-parametric นั่นเอง

ในส่วนล่างจะแสดงค่า p-value ของการทดสอบสมมุติฐานว่า ความแตกต่างน้อยกว่า 0, ไม่เท่ากับ 0 หรือมากกว่า 0 ตามลำดับ โดยปกติในการทดสอบสองหาง เราจะใช้ค่าที่ทดสอบว่าความแตกต่างไม่เท่ากับ 0 ส่วนการทดสอบหางเดียวก็เลือกว่าจะใช้ค่าใด ขึ้นกับว่าค่าเฉลี่ยหรือค่ามัธยฐานของกลุ่มใดมากหรือน้อยกว่ากัน

ในการทดสอบที่มีตัวแปรสองกลุ่มเช่นนี้ เราอาจจัดวางตัวแปรเป็นสองตัวแปรแยกกันก็ได้ ในกรณีเช่นนี้ ตัวเลือกของการทดสอบจะไม่ใช้ตัวเลือก “formula” หากเป็นสองตัวแปรในกรอบข้อมูลเดียวกัน ให้เลือกตัวเลือก “ในกรอบข้อมูล” แต่หากสร้างเป็นเวกเตอร์สองเวกเตอร์ ก็ให้เลือกตัวเลือก “สองเวกเตอร์” ในที่นี้จะแสดงหน้าต่างที่ได้จากการเลือกตัวเลือก “ในกรอบข้อมูล” ดังรูปที่ 10.6



รูปที่ 10.6 หน้าต่างเลือกตัวแปรและเงื่อนไขของการทดสอบ Student t ในรูปแบบ ในกรอบข้อมูล

ลองทดลองกับชุดข้อมูลตัวอย่าง ahcc ซึ่งมีการจัดเรียงข้อมูลแบบนี้ จะได้ผลเช่นเดียวกันดังนี้

```
statis>> st.test(ahcc, dr1, dr2, mu=0, paired=FALSE, var.equal=FALSE,
conf.level=0.95, frame=FALSE)
    Obs. NA Mean  S.D.  Min. Median Max.
dr1 40   0  60.73 15.53 12   57.5   103
dr2 30   10 57.33 10.69 36   57.5   95

Welch Two Sample t-test: ahcc$dr1 and ahcc$dr2
Diff. of Mean    Lower Lim.    Upper Lim.
       3.3917       -2.8680        9.6513

t = 1.0813, df = 68, Ho: diff == 0
       Ha:   diff < 0  diff != 0   diff > 0
  p-value:     0.8583     0.2834     0.1417

statis>> wilcoxon.test(ahcc, dr1, dr2, mu=0, paired=FALSE,
correct=TRUE, conf.level=0.95, frame=FALSE)
    Obs. Min. Median Max.
dr1 40   12   57.5   103
dr2 30   36   57.5   95
```

```
Wilcoxon rank sum test with continuity correction
data: dr1 and dr2
V = 672.5, Ho: true mu == 0
          Ha:   true mu < 0  true mu != 0   true mu > 0
     p-value:         0.807        0.3925        0.1963
```

แต่หากชุดข้อมูลมาจากการตรวจซ้ำสองครั้งในตัวอย่างชุดเดียวกัน แต่คนละเวลา คนละเงื่อนไข เพื่อจะหาว่าเมื่อเงื่อนไขเปลี่ยนไป จะทำให้มีการเปลี่ยนแปลงของตัวแปรนั้นหรือไม่ ในกรณีนี้เราต้องทดสอบแบบ paired ลองดูในรูปที่ 10.6 อีกครั้ง จะเห็นว่ามีตัวเลือกให้เลือกว่าจะทดสอบแบบ paired หรือไม่ ค่าปกติคือไม่ใช่ (FALSE) แต่ในกรณีนี้ให้เลือก TRUE ซึ่งชุดข้อมูลนี้เป็นการตรวจซ้ำในคนเดียวกันสองครั้งจริง จึงควรเลือกเงื่อนไขนี้ ผลที่ได้จะเป็นดังนี้

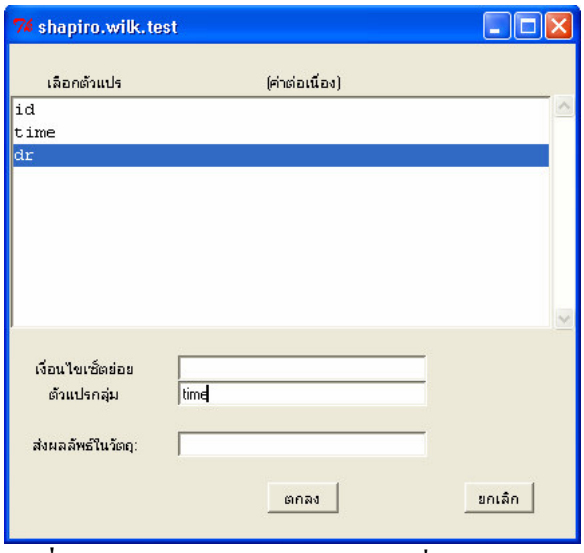
```
statis>> st.test(ahcc, dr1, dr2, mu=0, paired=TRUE, var.equal=FALSE,
conf.level=0.95, frame=FALSE)
    Obs. NA Mean  S.D.  Min. Median Max.
dr1 40   0  60.73 15.53 12   57.5   103
dr2 30   10 57.33 10.69 36   57.5   95

Paired t-test: ahcc$dr1 and ahcc$dr2
Diff. of Mean    Lower Lim.    Upper Lim.
       0.7667       -4.3510        5.8844

t = 0.3064, df = 29, Ho: diff == 0
       Ha:   diff < 0  diff != 0   diff > 0
  p-value:     0.6193     0.7615     0.3807

statis>> wilcoxon.test(ahcc, dr1, dr2, mu=0, paired=TRUE,
correct=TRUE, conf.level=0.95, frame=FALSE)
    Obs. Min. Median Max.
dr1 40   12   57.5   103
dr2 30   36   57.5   95

Wilcoxon signed rank test with continuity correction
data: dr1 and dr2
V = 252, Ho: true mu == 0
          Ha:   true mu < 0  true mu != 0   true mu > 0
     p-value:        0.6597        0.6957        0.3478
```

จะเห็นว่าค่า p-value ของการทดสอบแบบ paired และ unpaired มีความแตกต่างกันอยู่บ้าง แต่ในกรณีนี้พอสรุปได้ว่าค่า dr จากการตรวจทั้งสองครั้งไม่แตกต่างกัน

ทดสอบ ***Shapiro Wilk***

ดังได้กล่าวมาแล้วข้างต้นว่าการทดสอบ Student t นั้น มีข้อกำหนดว่าการแจกแจงของข้อมูลในแต่ละกลุ่มจะต้องเป็นแบบปกติ ซึ่งจะตรวจสอบได้ด้วยการทดสอบ Shapiro Wilk normality ตัวการทดสอบเองนั้นไม่ยุ่งยาก แต่ต้องระวังว่าหากว่าการจัดเรียงข้อมูลเป็นแบบ long คือข้อมูลทั้งหมดทุกกลุ่มต่อกันภายในตัวแปรเดียว ซึ่งเป็นรูปแบบที่จะใช้ตัวเลือก formula แล้ว

จำเป็นต้องแบ่งการทดสอบออกเป็นแต่ละกลุ่ม อย่าทดสอบทีเดียวทั้งตัวแปร เพราะจะมีข้อมูลจากหลายกลุ่มปนกันอยู่ เมื่อเลือกรายการนี้ เราสามารถเลือกได้ว่าจะทดสอบกับเวกเตอร์หรือกับตัวแปรในกรอบข้อมูล ดังรูปที่ 10.7



รูปที่ 10.7 หน้าต่างเลือกตัวแปรและเงื่อนไขของการทดสอบ Shapiro Wilk ในกรอบข้อมูล

นอกจากเลือกตัวแปรได้แล้ว ยังสามารถเลือกเงื่อนไขเซ็ตย่อย และที่สำคัญคือตัวแปรกลุ่มได้ ดังตัวอย่างชุดข้อมูล ahcc2 เราเลือกตัวแปร dr เพื่อทดสอบ แล้วเลือกตัวแปรกลุ่มเป็น time โดยพิมพ์ time เข้าไปในช่องว่าง เมื่อกดปุ่ม “ตกลง” จะได้ผลการแสดงบนหน้าจอ R console ดังข้างล่าง

```
statis>> shapiro.wilk.test(ahcc2, dr, , group=time)
group: ahcc2$time == 1
Shapiro-Wilk normality test
  data: ahcc2$dr
  W = 0.9417, p-value = 0.0394

group: ahcc2$time == 2
Shapiro-Wilk normality test
  data: ahcc2$dr
  W = 0.9013, p-value = 0.009
```

จะเห็นว่าในบรรทัดคำสั่งมีการระบุตัวเลือก group ด้วย ซึ่งจะทำให้มีการทดสอบแยกตามกลุ่มพร้อมกันไป ผลจากการทำงานแสดงแยกกันตามกลุ่ม ซึ่งในที่นี้จะเห็นว่าค่า p-value มีค่าน้อยกว่า 0.05 ทั้งสองกลุ่ม แสดงว่าการแจกแจงของ dr ในทั้งสองกลุ่มแตกต่างจากการแจกแจงปกติ ซึ่งไม่เป็นไปตามข้อตกลงของการทดสอบ Student t จึงไม่สามารถใช้การทดสอบนี้ได้

ทดสอบ *likelihood ratio*

การทดสอบ likelihood ratio เป็นการทดสอบที่ใช้เปรียบเทียบว่าโมเดลทางสถิติสองโมเดลมีความต่างกันหรือไม่ เนื่องจากเป็นการทดสอบที่ต้องทำหลังจากการทำโมเดลทางสถิติซึ่งเป็นการคำนวณทางสถิติขั้นสูง ซึ่งจะต้องใช้วัตถุชนิด glm หรืออนุพันธ์ของ glm ในการทำงาน จึงจะไม่อธิบายรายละเอียดในที่นี้ เพราะจะเป็นเรื่องยืดยาวมาก แต่จะแสดงการทำงานของตัวเลือกนี้พอให้เห็นเป็นตัวอย่างเท่านั้น ดังนี้

```
> mod1 <- glm(status ~ ovab + alcohol + smoke, data=cca,
family=binomial)
> mod2 <- glm(status ~ ovab + alcohol, data=cca, family=binomial)
summary>> summary(mod1)

Call:
glm(formula = status ~ ovab + alcohol + smoke, family = binomial,
    data = cca)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.4432  -0.7645  -0.7261   0.7436   1.7094

Coefficients:
                  Estimate Std. Error z value Pr(>|z|)
(Intercept)        -1.0805     0.2832  -3.815 0.000136 ***
ovabpositive        2.3427     0.3782   6.194 5.87e-10 ***
alcoholoccasional   0.0015     0.4154   0.004 0.997118
alcoholex-drinker   1.2307     0.6461   1.905 0.056807 .
alcoholdrinker      1.5907     0.5288   3.008 0.002630 **
smokeoccasional     0.1979     0.7182   0.276 0.782922
smokeex-smoker      0.4397     0.5242   0.839 0.401536
smokesmoker        -0.1180     0.4207  -0.281 0.779070
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 346.56  on 249  degrees of freedom
Residual deviance: 268.11  on 242  degrees of freedom
  (12 observations deleted due to missingness)
AIC: 284.11

Number of Fisher Scoring iterations: 4

summary>> summary(mod2)

Call:
glm(formula = status ~ ovab + alcohol, family = binomial, data = cca)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-2.3902  -0.7842  -0.7678   0.6991   1.6525

Coefficients:
                  Estimate Std. Error z value Pr(>|z|)
(Intercept)       -1.07052    0.27943  -3.831 0.000128 ***
ovabpositive       2.35506    0.37564   6.269 3.62e-10 ***
alcoholoccasional  0.04892    0.35381   0.138 0.890036
```

```
alcoholex-drinker  1.36447    0.55498   2.459 0.013948 *
alcoholdrinker     1.51283    0.43696   3.462 0.000536 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 346.56  on 249  degrees of freedom
Residual deviance: 269.47  on 245  degrees of freedom
  (12 observations deleted due to missingness)
AIC: 279.47

Number of Fisher Scoring iterations: 4

statis>> lr.test(mod1,mod2)
Likelihood ratio test: Chi-squared df(3) = 1.3597, p-value = 0.715
```

จากตัวอย่างจะเห็นว่าเราสร้างโมเดลขึ้นสองโมเดล ชื่อ `mod1` และ `mod2` โดยใช้ฟังก์ชัน `glm()` ซึ่งทั้งสองใช้ family เป็น binomial นั่นคือเป็นการสร้าง logistic model สองโมเดลนั่นเอง โดยที่ `mod1` มีตัวแปรต้นคือ `ovab+alcohol+smoke` และ `mod2` มีตัวแปรต้นคือ `ovab+alcohol` เนื่องจากผลจากการ `summary()` แสดงให้เห็นแล้วว่า `smoke` ไม่น่าจะมีผลต่อการทำนายการเกิดมะเร็ง คำถามคือเราจะตัดตัวแปร `smoke` ออกจากโมเดลได้หรือไม่ นั่นคือใช้ `mod2` แทนที่จะใช้ `mod1` บรรทัดสุดท้ายจึงใช้ฟังก์ชัน `lr.test()` ซึ่งเรียกจากตัวเลือกนี้นั่นเอง เพื่อตอบคำถามว่าโมเดลทั้งสองต่างกันหรือไม่ คำตอบที่ได้คือไม่ต่างกัน โดยมี p-value = 0.715 คือมีค่ามากกว่า 0.05

โดยสรุปแล้ว รายการและฟังก์ชันต่างๆ ที่เกี่ยวข้องกับ ice.statis สรุปได้ดังตารางที่ 10.2

ตารางที่ 10.2 สรุปฟังก์ชันและรูปแบบการใช้งานของรายการต่างๆ ในโมดุล ice.statis

รายการ	ฟังก์ชัน	รูปแบบการใช้งานฟังก์ชัน
ช่วงความเชื่อมั่น	`ci()`	`ci(data, x, subset, group, level=0.95, digits = 3, frame = get.ec.option("frame"))` `ci(n, m, sd, level = 0.95, digits = 3, frame = get.ec.option("frame"))`
ทดสอบ Fisher's exact	`fisher.test()`	`fisher.test(x, y = NULL, workspace = 200000, hybrid = FALSE, control = list(), or = 1, alternative = "two.sided", conf.int = TRUE, conf.level = 0.95, simulate.p.value = FALSE, B = 2000)`
ทดสอบ chi-squared	`chisq.test()`	`chisq.test(x, y = NULL, correct = TRUE, p = rep(1/length(x), length(x)), rescale.p = FALSE, simulate.p.value = FALSE, B = 2000)`
ทดสอบ anova	`anova.test()`	`anova.test(formula, data, subset, frame = get.ec.option("frame"))`
ทดสอบ Student t	`st.test()`	`st.test(data, x, y = NULL, mu = 0, paired = FALSE, var.equal = FALSE, conf.level=0.95, frame = get.ec.option("frame"))` `st.test(formula, data, subset, frame = get.ec.option("frame"))`

ทดสอบ Wilcoxon/Mann-Whitney	`wilcoxon.test()`	`wilcoxon.test(data, x, y = NULL, mu = 0, paired = FALSE, exact = NULL, correct = TRUE, conf.level = 0.95, frame = get.ec.option("frame"))` `wilcoxon.test(formula, data, subset, frame = get.ec.option("frame"))`
ทดสอบ Shapiro-Wilk	`shapiro.wilk.test()`	`shapiro.wilk.test(x)` `shapiro.wilk.test(data, select, subset, group)`
ทดสอบ likelihood ratio	`lr.test()`	`lr.test(model1, model2)`

บทที่ 11 สรุปขั้นตอนการทำงานในสภาพแวดล้อม R-ICE

ติดตั้งโปรแกรมช่วยเหลือ

ติดตั้ง **R** และ **R-ICE**

ใช้งาน **R**

เรียกใช้ **R-ICE** จากภายใน **R**

ทำงานกับแฟ้มสคริปต์

ติดตามความก้าวหน้าของ **R-ICE**

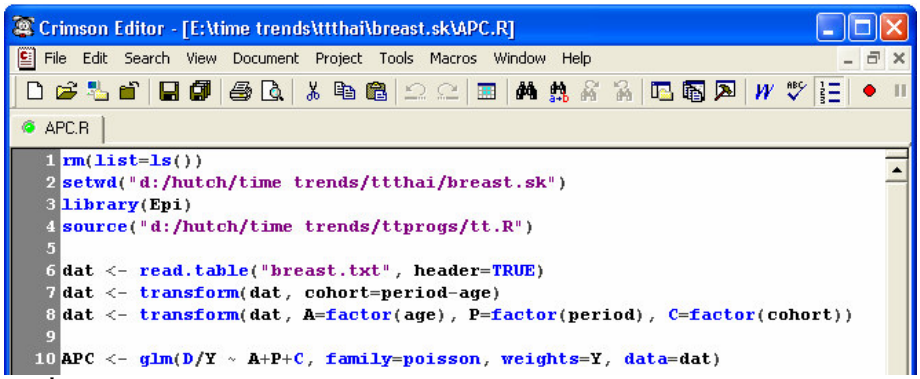
จากที่กล่าวมาในบทต่างๆ ข้างต้นนั้น เนื่องจากมีรายละเอียดมากมาย จึงอาจทำให้จับขั้นตอนการทำงานในสภาพแวดล้อม **R-ICE** ไม่ได้ชัดเจน ในบทนี้จึงจะสรุปขั้นตอนต่างๆ อีกครั้ง

โปรแกรม **R** มีใช้งานทั้งบนวินโดวส์ ลินุกซ์ และแมคอินทอช วิธีการทำงานก็แตกต่างกันไปตามแต่ระบบปฏิบัติการนั้นๆ หนังสือเล่มนี้จึงได้เน้นที่ระบบปฏิบัติการวินโดวส์เป็นหลัก เนื่องจากคนไทยส่วนใหญ่ใช้วินโดวส์อยู่

ติดตั้งโปรแกรมช่วยเหลือ

เนื่องจากโปรแกรม **R** ถูกพัฒนาขึ้นครั้งแรกบนระบบปฏิบัติการตระกูลยูนิกซ์ โปรแกรม **R** ที่พัฒนาบนระบบปฏิบัติการอื่น จึงยังคงรักษารูปแบบการทำงานแบบยูนิกซ์อยู่ ประการแรกคือ จำเป็นต้องใช้โปรแกรมหลายๆ โปรแกรมทำงานร่วมกัน ไม่ได้เป็นระบบการทำงานที่รวมความสามารถทุกอย่างทั้ง text editor และ data editor เข้าด้วยกันเช่นโปรแกรมสำเร็จรูปทางสถิติที่นิยมใช้บนวินโดวส์ เช่น **SPSS** หรือ **Stata** และนอกจากนี้ **R** ยังทำงานด้วยการพิมพ์บรรทัดคำสั่ง ซึ่งอาจทำให้ผู้ใช้งานในระดับง่ายๆ รู้สึกไม่สะดวกในการทำงาน อย่างไรก็ตาม หากได้ใช้จนคุ้นชินแล้วก็จะทำได้โดยไม่ติดขัดแต่อย่างใด

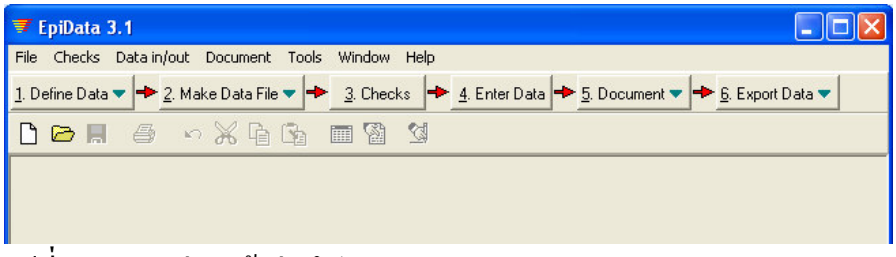
โปรแกรมช่วยเหลือ หรือสภาพแวดล้อมเสริมในการทำงานที่ควรติดตั้งร่วม ในกลุ่มแรกคือ text editor สำหรับผู้ใช้วินโดวส์แล้ว แนะนำให้ใช้โปรแกรม **Crimson Editor** ซึ่งสามารถดาวน์โหลดโปรแกรม setup ได้ฟรีที่ http://www.crimsoneditor.com ซึ่งการติดตั้งก็ไม่ยุ่งยากแต่อย่างใด ข้อดีของโปรแกรมนี้คือสามารถส่งบรรทัดคำสั่งให้ไปทำงานใน **R** ได้โดยตรง เหมือนกับว่าเป็น editor ของ **R** ไปในตัวนั่นเอง และยังรู้จักชื่อฟังก์ชันและคำสงวนต่างๆ ของ **R** ทำให้ทำสีที่ต่างออกไปจากคำอื่นๆ ได้ ตรวจสอบการเปิดปิดวงเล็บต่างๆ ได้



รูปที่ 11.1 ตัวอย่างหน้าต่างโปรแกรม **Crimson Editor**

สำหรับผู้ใช้ระบบลินุกซ์ โปรแกรมที่สามารถตรวจสอบชื่อฟังก์ชันและคำสงวนของ **R** ได้ที่แนะนำให้ใช้คือโปรแกรม **jEdit** ส่วนโปรแกรม **R** บนระบบแมคอินทอช OSX นั้นมี text editor มาให้พร้อมแล้ว ไม่ต้องติดตั้งโปรแกรมเพิ่มแต่อย่างใด

โปรแกรมอีกกลุ่มหนึ่งที่น่าจะมีประโยชน์คือโปรแกรม data editor ที่แนะนำให้ใช้คือโปรแกรม **EpiData** ซึ่งมีกระบวนการขั้นตอนป้องกันความผิดพลาดในการกรอกข้อมูลจำนวนมากๆ ได้ ซึ่งระบบป้องกันความผิดพลาดในการกรอกข้อมูลเป็นสิ่งสำคัญมากในการทำวิจัยที่มีข้อมูลจำนวนมาก



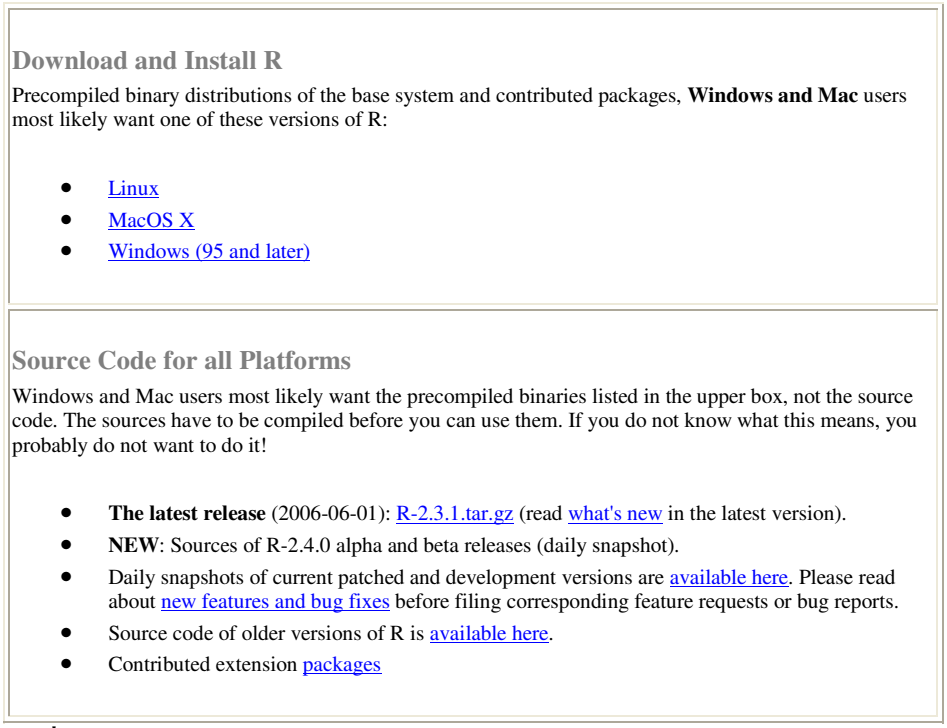
รูปที่ 11.2 ตัวอย่างหน้าต่างโปรแกรม **EpiData**

โปรแกรมนี้สามารถดาวน์โหลดได้ฟรีที่ http://www.epidata.dk โปรแกรมนี้ใช้ได้เฉพาะบนระบบวินโดวส์เท่านั้น แต่ผู้ใช้ลินุกซ์สามารถใช้งานได้ผ่านโปรแกรม **Wine** ซึ่งสามารถดาวน์โหลดได้ฟรีที่ http://www.winehq.com ส่วนผู้ใช้แมคอินทอช OSX อาจลองใช้โปรแกรม **CrossOver** ได้ที่เว็บไซต์ http://www.codeweavers.com

นอกจากโปรแกรม **EpiData** แล้ว ผู้ที่มีโปรแกรมระบบฐานข้อมูลอื่นๆ ก็สามารถนำมาใช้ในการกรอกข้อมูลได้ และถ้ามีข้อมูลจำนวนน้อย ก็อาจใช้ตารางคำนวณของโปรแกรม **Excel** หรือของ **OpenOffice** ได้เช่นกัน เพียงแต่โปรแกรมเหล่านี้ไม่มีระบบต้องตรวจสอบความถูกต้องในการกรอกข้อมูล ผู้ใช้จำเป็นต้องระมัดระวังในการกรอกข้อมูลด้วยตนเอง

ติดตั้ง R และ R-ICE

การติดตั้ง **R** นั้นไม่ยุ่งยากในทุกระบบปฏิบัติการ เนื่องจากมีโปรแกรม setup หรือ installer หรือ disk image ให้ดาวน์โหลดได้ฟรีที่เว็บไซต์ของ **CRAN** ที่ http://cran.r-project.org ดังรูปที่ 11.3



รูปที่ 11.3 หน้าเว็บดาวน์โหลดโปรแกรม **R** จาก **CRAN**

รายละเอียดในการติดตั้งโปรแกรม **R** ในแต่ละระบบปฏิบัติการและการตั้งค่าเริ่มต้นต่างๆ สามารถอ่านได้จากบทที่ 1 เรื่อง **สภาพแวดล้อม R** ส่วนการดาวน์โหลดและติดตั้งโมดุลของ **R-ICE** นั้น สามารถทำได้จากเว็บไซต์ http://www.R-ICE.org การติดตั้งก็สามารถทำได้เช่นเดียวกับการติดตั้งแพคเกจของ **R** นั่นเอง

ใช้งาน R

ก่อนที่จะใช้งาน **R-ICE** จะต้องเรียกใช้โปรแกรม **R** ก่อน ซึ่งเราสามารถพิมพ์บรรทัดคำสั่งต่างๆ บนหน้าต่าง R console ได้โดยตรง แม้ว่า **R-ICE** จะกำลังทำงานอยู่ด้วยก็ตาม การเรียกใช้ **R** บนระบบปฏิบัติการวินโดวส์จะเรียกผ่านโปรแกรม Rgui.exe ซึ่งจะถูกเรียกโดยตรงเมื่อเรียกผ่าน desktop shortcut และเมื่อเรียกผ่านเมนู Start ของวินโดวส์ บนระบบปฏิบัติการลินุกซ์จะเรียกผ่านหน้าต่าง terminal โดยพิมพ์ R (ตัวใหญ่) ที่ prompt ส่วนบนระบบแมคอินทอช OSX นั้นเรียกผ่านแอพลิเคชัน **R**

R เป็นทั้งภาษาโปรแกรม และเป็นสภาพแวดล้อมในการคำนวณทางสถิติและกราฟิก ในหนังสือเล่มนี้จะใช้งานทางด้านสถิติเป็นหลัก แต่ที่จริงแล้ว **R** ยังมีความสามารถสูงทางด้านกราฟิกเช่นกัน **R** เป็นสภาพแวดล้อมแบบ object oriented ซึ่งทุกสิ่งทุกอย่างที่ผู้ใช้ทำงานอยู่ใน

สภาพแวดล้อม **R** จะเป็นวัตถุทั้งหมด ที่ผู้ใช้รู้สึกได้คือต้องกำหนดชื่อให้ มิฉะนั้นจะมีเพียงการแสดงออกทางหน้าจอ แล้วจะถูกลบกำจัดออกไป และเรียกใช้ใหม่อีกไม่ได้

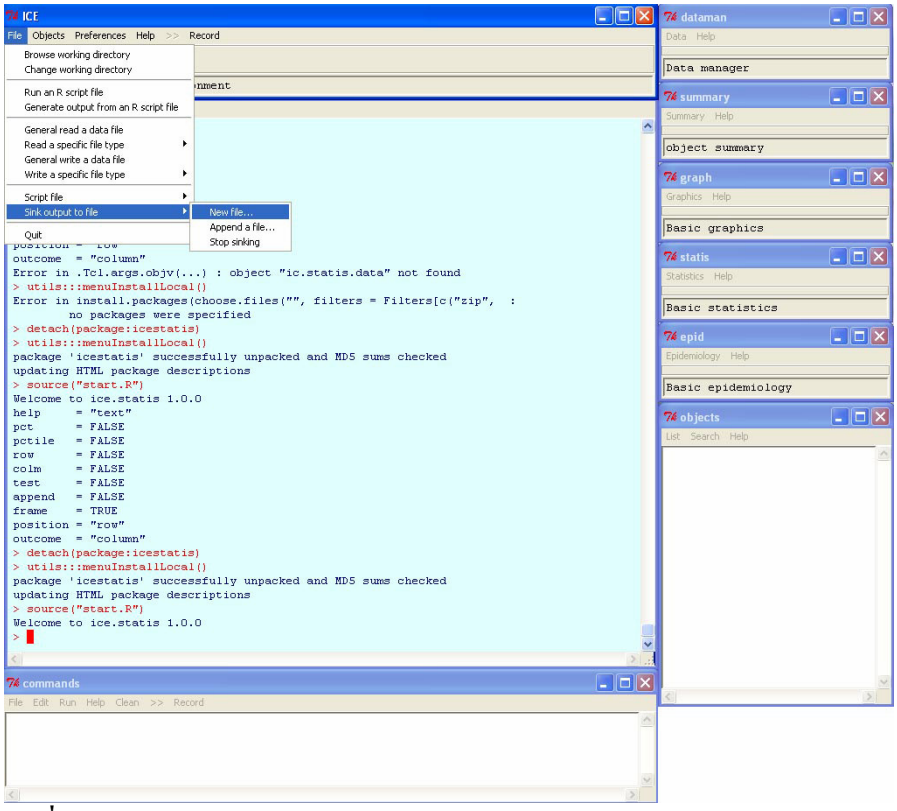
การที่ **R** เป็นภาษาโปรแกรม ทำให้การออกคำสั่งจำเป็นต้องเขียนในรูปแบบคล้ายบรรทัดคำสั่งโปรแกรมคอมพิวเตอร์ คือมีการใช้วงเล็บ และเครื่องหมายต่างๆ มากมาย ทำให้ผู้ใช้มือใหม่รู้สึกไม่สะดวกในการใช้งาน เพราะมักจะสับสนกับเครื่องหมายต่างๆ เหล่านั้น หรือสับสนกับการซ้อนคำสั่งเข้าด้วยกัน ทำให้มีวงเล็บจำนวนมาก และมักจะปิดไม่หมด ทำให้ทำงานไม่ถูกต้องหรือไม่ทำงาน นั่นเป็นเหตุผลหลักที่สภาพแวดล้อม **R-ICE** ได้ถือกำเนิดขึ้นมา อย่างไรก็ตาม การทำงานในระดับสูงก็หลีกเลี่ยงไม่ได้ที่จะต้องเขียนบรรทัดคำสั่งอยู่นั่นเอง เพราะเป็นการยากที่จะสร้างรายการเมนูให้กับทุกฟังก์ชันใน **R** ผู้ที่ยังไม่คุ้นเคยกับการทำงานในสภาพแวดล้อม **R** ควรศึกษาบทที่ 4 เรื่อง **การทำงานและวัตถุพื้นฐานในสภาพแวดล้อม R** และในการนำแฟ้มข้อมูลเข้าและบันทึกแฟ้มข้อมูล ก็ขอให้ศึกษาในบทที่ 5 เรื่อง **การนำข้อมูลเข้าสู่สภาพแวดล้อม R และการบันทึกแฟ้มข้อมูล**

ในการติดตั้งสภาพแวดล้อม **R-ICE** นั้น ได้กล่าวรายละเอียดไว้แล้วในบทที่ 3 เรื่อง **สภาพแวดล้อม R-ICE** โดยสรุปในการติดตั้งสภาพแวดล้อม **R-ICE** ก็ทำเหมือนการติดตั้งแพคเกจของ **R** อื่นๆ นั่นเอง สภาพแวดล้อม **R-ICE** ประกอบด้วยหลายโมดุล และเรายังอาจเลือกส่วนผสมต่างๆ ตามความต้องการได้อีกด้วย

บนระบบปฏิบัติการวินโดวส์ ให้ดาวน์โหลดโมดุล **R-ICE** ที่ต้องการติดตั้งมาไว้ที่เครื่อง ให้เลือกแฟ้มที่เป็น zip และอย่าแตกแฟ้ม zip นั้นออกเป็นอันขาด จากนั้นเรียกใช้โปรแกรม **R** แล้วติดตั้งโมดุลต่างๆ ของ **R-ICE** โดยเลือกรายการเมนู Packages >> Install package(s) from local zip files… ด้านล่างสุด โมดุลต่างๆ เหล่านั้นก็จะถูกติดตั้งเข้าสู่โปรแกรม **R** ได้อย่างง่ายดาย ส่วนผู้ใช้ระบบลินุกซ์และแมคอินทอช OSX ให้ดาวน์โหลดแฟ้ม tar.gz (อย่าดาวน์โหลดแฟ้ม zip) มาที่ home แล้วติดตั้งตามที่อธิบายในหัวข้อ **การติดตั้งแพคเกจเพิ่มเติมเพื่อเพิ่มความสามารถให้ R** ในบทที่ 1 เรื่อง **สภาพแวดล้อม R**

เนื่องจากโมดุล **R-ICE** ถูกสร้างโดยใช้แพคเกจ tcltk ทำให้อาจมีปัญหาในการแสดงอักษรภาษาไทยบนระบบลินุกซ์และแมคอินทอช OSX ดังนั้นผู้ใช้ระบบปฏิบัติการทั้งสองระบบนี้ขอให้ดาวน์โหลดโมดุลภาษาอังกฤษมาใช้ แต่คาดว่าในอนาคต สภาพแวดล้อม tcltk บนระบบปฏิบัติการทั้งสองจะสามารถแสดงภาษาไทยได้อย่างถูกต้อง

และนอกจากนี้ผู้ใช้ระบบแมคอินทอช OSX จะต้องไม่ลืมเปิดใช้ X11 server โดยเรียกจากรายการเมนู Misc >> Run X11 Server ซึ่งอยู่ด้านล่างสุดของรายการ



รูปที่ 11.4 สภาพแวดล้อม **R-ICE** บนระบบปฏิบัติการวินโดวส์

เรียกใช้ R-ICE จากภายใน R

เมื่อติดตั้งโมดุลต่างๆ ของ **R-ICE** แล้ว ก็ไม่จำเป็นต้องติดตั้งซ้ำอีก ยกเว้นเมื่อได้ติดตั้งโปรแกรม **R** รุ่นใหม่เท่านั้น ในการเรียกใช้สภาพแวดล้อม **R-ICE** ควรสร้างแฟ้มข้อความธรรมดาขึ้นหนึ่งแฟ้ม อาจตั้งชื่อว่า `start.ice` ก็ได้ เพื่อทำหน้าที่จัดการกระบวนการ start up โดยใส่แฟ้มไว้ที่โฟลเดอร์ {R_HOME} ของ **R** ซึ่งอยู่ที่ `..\R\R-x.x.x\` บนระบบวินโดวส์ หรือที่ `/usr/lib/R/` บนลินุกซ์ หรือที่ `/Library/Frameworks/R.framework/` บนแมคอินทอช OSX โดยมีบรรทัดข้อความดังนี้

```
# Calling libraries
library(ice)
library(ice.main)
library(ice.dataman)
library(ice.summary)
library(ice.graph)
library(ice.statis)
library(ice.commands)
library(ice.objects)

# Opening tk windows for ICE environment
ice.main()
ice.dataman()
ice.summary()
ice.graph()
```

```
ice.statis()
ice.commands()
ice.objects()
```

หรือหากจะเลือกส่วนผสมแบบ fullmain ก็จะเป็น

```
# Calling libraries
library(ice)
library(ice.fullmain)
library(ice.commands)
library(ice.objects)

# Opening tk windows for ICE environment
ice.fullmain()
ice.commands()
ice.objects()
```

เมื่อเปิดใช้งาน **R** แล้ว บนหน้าจอ R console เราสามารถพิมพ์บรรทัดคำสั่งต่อไปนี้เพื่อเรียกใช้งานสภาพแวดล้อม **R-ICE** ได้

```
> source("start.ice")
```

จากนั้นหน้าต่างต่างๆ ของ **R-ICE** ควรจะปรากฏขึ้นอย่างไม่เป็นระเบียบ เราสามารถนำหน้าต่างเหล่านั้นไปวางในที่ต่างๆ บนจอได้ตามต้องการ

ทำงานกับแฟ้มสคริปต์

แม้ว่าโดยปกติแล้ว เราสามารถใช้งาน **R** ได้โดยการพิมพ์บรรทัดคำสั่ง หรือ **R-ICE** โดยการเลือกรายการเมนูและดูผลการทำงานทางหน้าจอ R console แต่เพื่อให้การทำงานวิเคราะห์ข้อมูลสามารถทำซ้ำและตรวจสอบได้ในภายหลัง แนะนำให้ทำงานกับแฟ้มสคริปต์เสมอ

ในการใช้สภาพแวดล้อม **R-ICE** แนะนำทำเป็นระบบและขั้นตอนดังนี้ ขั้นแรกสร้างโฟลเดอร์สำหรับการทำงานวิเคราะห์ข้อมูลชุดนั้นขึ้นมาหนึ่งโฟลเดอร์ สมมติในที่นี้ใช้ชื่อว่า `cholangioca` ย้ายแฟ้มข้อมูลที่สร้างไว้แล้ว สมมติว่าชื่อ `cca.dta` ซึ่งเป็นแฟ้มข้อมูลที่สร้างจากโปรแกรม **EpiData** แล้วบันทึกเป็นแฟ้มข้อมูลของ **Stata** ที่มีนามสกุล DTA จากนั้นใช้โปรแกรม **Tinn-R** สร้างแฟ้มสคริปต์เปล่าขึ้นมาหนึ่งแฟ้ม สมมติว่าตั้งชื่อให้เป็น `cca.R` แล้วพิมพ์บรรทัดบนสุดเขียนประโยคอธิบายการทำงานของสคริปต์นั้น โดยใส่ # (จะใส่กี่ตัวก็ได้ไม่มีข้อจำกัด) ที่หน้าบรรทัด เช่น

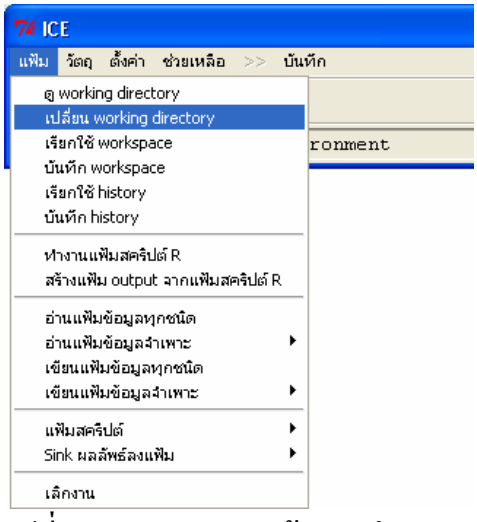
```
## A case-control study of cholangiocarcinoma in Nakhon Phanom.
## Original data file: cca.dta.
## Script file created on 2006-09-19, last modified 2006-09-28.
```

บันทึกแฟ้มนั้นลงในโฟลเดอร์ที่สร้างขึ้น ขณะนี้ในโฟลเดอร์นั้นก็จะมีแฟ้มอยู่สองแฟ้ม คือ `cca.dta` ซึ่งเป็นแฟ้มข้อมูล กับ `cca.R` ซึ่งเป็นแฟ้มสคริปต์

แนะนำ เนื่องจากผู้เขียนใช้จอขนาดค่อนข้างกว้าง จึงมีเนื้อที่ในการวางหน้าต่างต่างๆ ได้มาก จึงชอบวาง R console และหน้าต่าง ice.main ไว้ด้านบนของ R console อีกทีหนึ่ง ส่วนหน้าต่างอื่นๆ ของ **R-ICE** วางไว้ด้านขวาของ R console อีกที วางหน้าต่างแฟ้มสคริปต์ของ **Crimson Editor** ไว้ด้านขวาสุด สำหรับผู้ที่มีหน้าจอเล็ก อาจปรับเปลี่ยนหน้าต่างไว้ในที่อื่นก็ได้ แต่ควรให้เหลื่อมซ้อนกัน จะได้คลิกเพื่อเปลี่ยนหน้าต่างได้ง่าย แต่ถ้าไม่มีพื้นที่จริง ก็ใช้วิธีเปลี่ยนหน้าต่างโดยใช้แถบ task bar หรือ Alt-Tab ก็ได้

ข้อพึงระวัง สำหรับผู้ใช้โปรแกรม **Tinn-R** อาจมีปัญหาในการบันทึกแฟ้มข้อมูลที่สร้างขึ้นใหม่จริงๆ จาก **Tinn-R** ขอให้ดูวิธีเลี่ยงปัญหานี้ในบทที่ 2 เรื่อง **สภาพแวดล้อมเสริมการทำงาน**

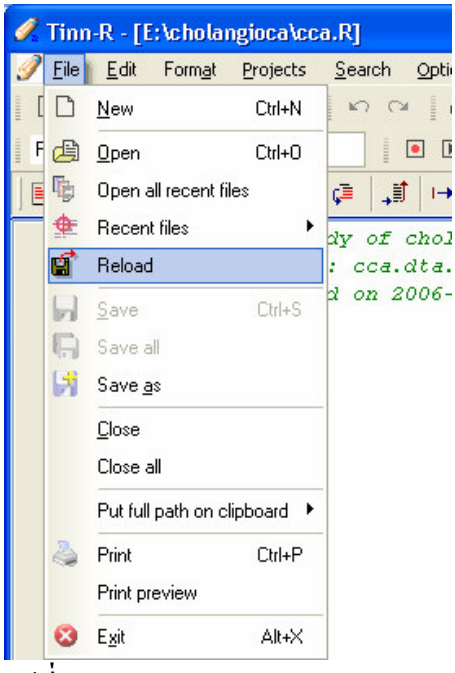
จากนั้นใช้รายการเมนู แฟ้ม >> เปลี่ยน working directory ของโมดุล ice.main (หรือ ice.fullmain) เพื่อเปลี่ยน working directory ของ **R** ไปที่โฟลเดอร์ที่สร้างขึ้นนั้น ดังรูปที่ 11.5



รูปที่ 11.5 รายการเมนู **แฟ้ม** ของโมดุล ice.main

แล้วเลือกรายการเมนู แฟ้ม >> แฟ้มสคริปต์ >> เปิด... ของโมดุล ice.main (หรือ ice.fullmain) ซึ่งอยู่ค่อนไปทางด้านล่างลงไปอีก เพื่อกำหนดการบันทึกบรรทัดคำสั่งจากรายการเมนู (หรือปุ่ม) “บันทึก” ของ ice.main ให้ทราบว่าจะต้องบันทึกลงแฟ้มที่สร้างไว้นั้น ทั้งนี้ไม่มี

ปัญหาการบันทึกซ้ำซ้อนกันระหว่าง **R-ICE** กับ **Crimson Editor** แต่อย่างใด แต่สำหรับผู้ใช้โปรแกรม **Tinn-R** เมื่อมีการบันทึกจาก **R-ICE** แล้วจะไม่เห็นการเปลี่ยนแปลงในแฟ้มสคริปต์ที่ **Tinn-R** กำลังเปิดอยู่ จนกว่าจะเรียกรายการเมนู File >> Reload ของ **Tinn-R** ดังรูปที่ 11.6



รูปที่ 11.6 รายการเมนู **File >> Reload** ของโปรแกรม **Tinn-R**

จากนั้นเรียกรายการ แฟ้ม >> เปลี่ยน working directory ของโมดุล ice.main ซ้ำอีกครั้งหนึ่ง แล้วเลือกรายการเมนู (หรือปุ่ม) “บันทึก” ของ ice.main จากนั้นย้ายกลับไปที่หน้าต่างของ **Crimson Editor** ถ้าหากเห็นบรรทัดคำสั่งว่า

```
setwd("E:/cholangioca")
```

แสดงว่าทุกอย่างทำงานได้อย่างถูกต้องเป็นระบบเดียวกันแล้ว เราก็จะเริ่มทำงานต่อไปได้

ขั้นแรกก็เป็นการเรียกชุดข้อมูลเข้ามาใช้งาน โดยรายการเมนู แฟ้ม >> อ่านแฟ้มข้อมูลทุกชนิด แล้วเลือกแฟ้ม `cca.dta` (หรือแฟ้มอื่นที่เราต้องการจะทำงานด้วย) เข้ามา จะเกิดบรรทัดคำสั่งขึ้นบน R console โดยไม่เห็นข้อมูลแต่อย่างใด ดังนี้

```
ICE>> cca <- read.dta("E:/cholangioca/cca.dta")
```

ซึ่งหมายความว่ามีการเรียกแฟ้มข้อมูล `cca.dta` เข้ามาในกรอบข้อมูลที่ตั้งชื่อว่า cca ตามชื่อของแฟ้มข้อมูลนั่นเอง จากนั้นเลือกรายการเมนู (หรือปุ่ม) “บันทึก” ของ ice.main แล้วลองย้ายกลับไปที่หน้าต่างของ **Crimson Editor** อีกครั้ง ควรจะเห็นบรรทัดคำสั่งว่า

```
cca <- read.dta("E:/cholangioca/cca.dta")
```

จากนั้นเราก็สามารถทำงานอื่นๆ ต่อไปตามลำดับได้แล้ว และเมื่อต้องการจะดูสิ่งที่บันทึก ก็อย่าลืมย้ายไปที่หน้าต่าง **Crimson Editor**

ลองทำงานอีกสักบรรทัดหนึ่งคือเลือกรายการ วัตถุ >> แสดงโครงสร้างของวัตถุ แล้วบันทึกลงแฟ้ม แล้วลองดูใน **Crimson Editor** อีกครั้ง ควรจะพบบรรทัดคำสั่ง

```
str(cca)
```

เพื่อทดสอบการทำงานของรายการ แฟ้ม >> ทำงานแฟ้มสคริปต์ **R** ของโมดุล ice.main ให้เลือกรายการดังกล่าว จะเห็นแฟ้ม `cca.R` ให้เลือกแฟ้มเดียว เมื่อเลือกแล้วจะพบว่าแฟ้มสคริปต์นั้นถูกทำงานด้วยการแสดงบรรทัดคำสั่งต่อไปนี้บน R console

```
ICE>> source("E:/cholangioca/cca.R", echo = TRUE,$
```

ซึ่งจะให้ผลการทำงานดังนี้

```
cca> setwd("E:/cholangioca")

cca> cca <- read.dta("E:/cholangioca/cca.dta")

cca> str(cca)
`data.frame':   262 obs. of  5 variables:
 $ status : Factor w/ 2 levels "control","case": $
```

การบันทึกผลการทำงานของแฟ้มสคริปต์ลงแฟ้ม output ทำได้โดยเลือกรายการ แฟ้ม >> สร้างแฟ้ม output จากแฟ้มสคริปต์ **R** จะปรากฏรายการให้เลือกแฟ้มสคริปต์ เมื่อเลือก `cca.R` (หรือแฟ้มที่ต้องการทำงาน) แล้ว จะถูกถามว่าบันทึกลงแฟ้มชื่ออะไรอีกครั้ง ให้ใส่ชื่อแฟ้ม แล้วขอให้ใช้นามสกุลว่า `OUT` เพื่อให้ทราบว่าเป็นแฟ้ม output เมื่อกดปุ่ม “Save” แล้ว เราจะได้แฟ้ม `cca.out` เพิ่มขึ้นอีกหนึ่งแฟ้มในโฟลเดอร์ `cholangioca` ที่เราสร้างขึ้นนั้น ลองเปิดแฟ้มดูด้วย **Crimson Editor** จะเห็นข้อความดังนี้

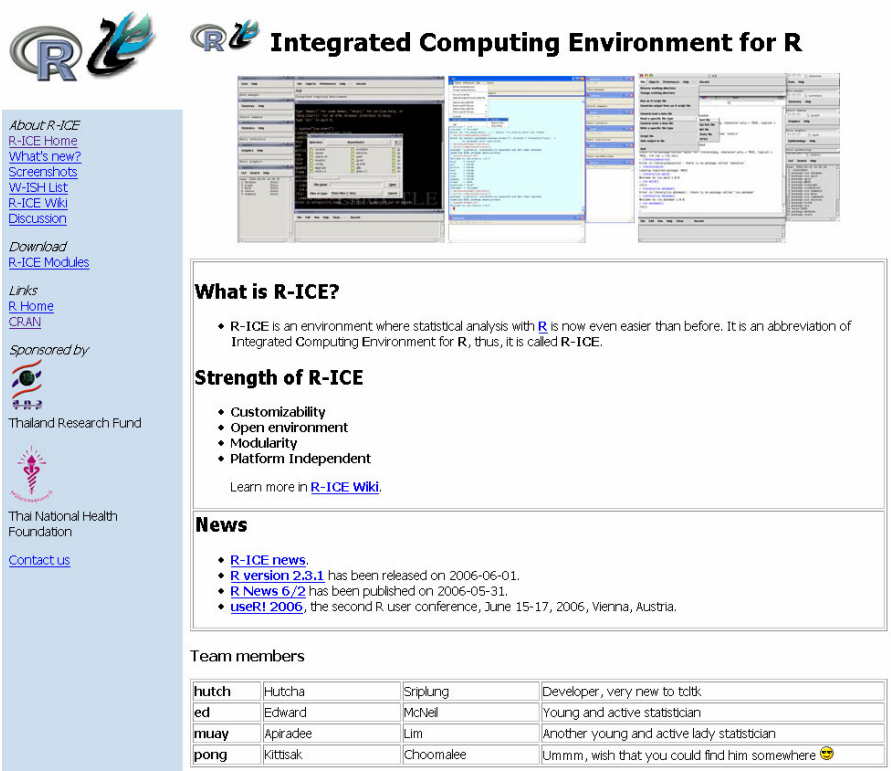
```
Output generated on: Thu Sep 28 10:18:21 2006
Version 2.3.1 Patched (2006-06-27 r38428)
cca>  setwd("E:/cholangioca")
NULL

cca>  cca <- read.dta("E:/cholangioca/cca.dta")
cca>  str(cca)
`data.frame':     262 obs. of  5 variables:
 $ status : Factor w/ 2 levels "control","case":$
```

นั่นแสดงว่าระบบการสร้างแฟ้ม บันทึก และทำงานแฟ้มสคริปต์ของ **R-ICE** ร่วมกับ **Crimson Editor** ทำงานได้อย่างประสานกันเรียบร้อยเป็นอย่างดี ทุกอย่างที่เราทำงานด้วยรายการเมนูสามารถถูกบันทึกลงแฟ้มสคริปต์ และแฟ้มสคริปต์นั้นสามารถเรียกกลับมาทำงานให้ผลลัพธ์เหมือนเดิมได้

คงเห็นแล้วว่าการทำงานอย่างเป็นระบบแบบนี้ หากทำจนคุ้นชินแล้วก็ไม่ยุ่งยากอะไร ทั้งยังเป็นการฝึกการทำงานอย่างเป็นระบบ ทำซ้ำได้ ตรวจสอบได้ อันเป็นคุณสมบัติที่พึงประสงค์ของการทำวิจัยที่ดีนั่นเอง ในการส่งงานให้กับอาจารย์ที่ปรึกษา หรือแหล่งทุน ก็จะไม่มีปัญหาในการตรวจสอบแต่อย่างใด

ติดตามความก้าวหน้าของ ***R-ICE***

โมดุลต่างๆ ของ **R-ICE** ยังจะพัฒนาต่อไปเรื่อยๆ ตราบใดที่ยังมีผู้ต้องการใช้งาน โดยสามารถติดตามดาวน์โหลดโมดุลรุ่นใหม่ๆ ได้ที่ http://www.R-ICE.org ซึ่งมีหน้าแรกดังรูปที่ 11.7



รูปที่ 11.7 เว็บไซต์ www.R-ICE.org

ภาคผนวก

ชุดข้อมูลตัวอย่าง

ชุดข้อมูลตัวอย่างในสภาพแวดล้อม **R-ICE** นั้นมีหลายชุด แต่ที่นำมาเป็นตัวอย่างในหนังสือเล่มนี้มีไม่มากนัก จึงจะกล่าวถึงแต่เพียงชุดข้อมูลที่ใช้เท่านั้น

ahcc

ข้อมูลชุดนี้เป็นค่าของการตรวจอย่างหนึ่งซึ่งทำในผู้ป่วยมะเร็งท่อน้ำดีตับ `dr1` เป็นค่าการตรวจก่อนให้อาหารเสริม AHCC และ `dr2` เป็นค่าที่ได้จากการตรวจหลังให้อาหารเสริม 3 เดือน ซึ่งผลการตรวจในลำดับแถวเดียวกันเป็นของผู้ป่วยคนเดียวกัน จะเห็นว่าในการตรวจครั้งที่สองมีบางคนไม่มีผลการตรวจเนื่องจากเสียชีวิตไปก่อนแล้ว เนื่องจากโรคนี้ทำให้ผู้ป่วยเสียชีวิตเร็วมากหลังพบว่าเป็นโรค โครงสร้างและการสรุปข้อมูลเป็นดังนี้

```
> str(ahcc)
`data.frame':   40 obs. of  2 variables:
 $ dr1: num  81 45 51 55 56 52 50 39 85 76 ...
 $ dr2: num  NA 60 59 57 55 64 NA 62 NA 57 ...
> summary(ahcc)
      dr1              dr2
 Min.   : 12.00   Min.   :36.00
 1st Qu.: 53.00   1st Qu.:50.75
 Median : 57.50   Median :57.50
 Mean   : 60.73   Mean   :57.33
 3rd Qu.: 71.25   3rd Qu.:62.00
 Max.   :103.00   Max.   :95.00
                  NA's   :10.00
```

ahcc2

ข้อมูลชุดนี้เป็นข้อมูลชุดเดียวกันกับ `ahcc` ข้างต้น แต่จัดเรียงข้อมูลในอีกรูปแบบหนึ่ง คือมีตัวแปร `id` เป็นหมายเลขผู้ป่วยซึ่งจะซ้ำกัน 40 คน โดยมีตัวแปร `time` บอกว่าเป็นการตรวจครั้งแรกหรือครั้งที่สอง และตัวแปร `dr` เป็นค่าการตรวจแต่ละครั้ง โครงสร้างและการสรุปข้อมูลเป็นดังนี้

```
> str(ahcc2)
`data.frame':   80 obs. of  3 variables:
 $ id  : int  1 2 3 4 5 6 7 8 9 10 ...
 $ time: int  1 1 1 1 1 1 1 1 1 1 ...
 $ dr  : int  81 45 51 55 56 52 50 39 85 76 ...
> summary(ahcc2)
       id             time           dr
 Min.   : 1.00   Min.   :1.0   Min.   : 12.00
 1st Qu.:10.75   1st Qu.:1.0   1st Qu.: 53.00
 Median :20.50   Median :1.5   Median : 57.50
 Mean   :20.50   Mean   :1.5   Mean   : 59.27
 3rd Qu.:30.25   3rd Qu.:2.0   3rd Qu.: 64.00
 Max.   :40.00   Max.   :2.0   Max.   :103.00
```

```
NA's   : 10.00
```

cca

ข้อมูลชุดนี้ได้จากการศึกษาแบบ case-control โดย case เป็นผู้ป่วยมะเร็งท่อน้ำดีตับในจังหวัดนครพนม และ control เป็นคนปกติเพศเดียวกัน และอายุต่างกันไม่เกิน 5 ปี ที่ไม่ได้เป็นมะเร็งตับที่อาศัยอยู่ในจังหวัดนครพนมเช่นกัน ในชุดข้อมูลนี้ สถานะการเป็นโรคหรือไม่เป็นโรคคือตัวแปร `status` เพื่อหาว่าอะไรบ้างที่เกี่ยวข้องกับการเกิดโรค ได้วัดภูมิคุ้มกันของร่างกายต่อพยาธิใบไม้ตับ (`ovab`) การดื่มเหล้า (`alcohol`) สูบบุหรี่ (`smoke`) และอายุ (`age`) และอื่นๆ อีกมากมาย แต่ในชุดข้อมูลนี้คัดตัวแปรมาแต่เฉพาะเท่าที่สรุปให้เห็นต่อไปนี้เท่านั้น

```
> str(cca)
`data.frame':   262 obs. of  5 variables:
 $ status : Factor w/ 2 levels "control","case":$
 $ ovab   : Factor w/ 2 levels "negative","posit$
 $ alcohol: Factor w/ 4 levels "never","occasion$
 $ smoke  : Factor w/ 4 levels "never","occasion$
 $ age    : num  60 69 71 64 43 50 77 55 59 45 ...
> summary(cca)
     status          ovab           alcohol
 control:131   negative:180   never     : 79
 case   :131   positive: 76   occasional:100
               NA's    :  6   ex-drinker: 24
                              drinker   : 53
                              NA's      :  6

        smoke          age
 never     :111   Min.   :28.00
 occasional: 14   1st Qu.:50.00
 ex-smoker : 35   Median :58.00
 smoker    : 96   Mean   :58.99
 NA's      :  6   3rd Qu.:69.00
                  Max.   :87.00
```