ภาคผนวก ข.1

“วิเคราะห์ข้อมูลทางสถิติด้วย R-ICE”

เป็นส่วนหนึ่งของโครงการ

การพัฒนาการใช้โปรแกรม R ในการวิเคราะห์ข้อมูล

R Program Development for Research Utilization

โดย

หัชชา ศรีปลั่ง และคณะ

เดือน ปี ที่เสร็จโครงการตามสัญญา

ตุลาคม 2549

ต้นฉบับคู่มือ

“วิเคราะห์ข้อมูลทางสถิติด้วย R-ICE”

ผู้แต่ง

รศ.นพ. หัชชา ศรีปลั่ง

การวิเคราะห์ทางสถิติพื้นฐานด้วย R-ICE

บทนำ

R เป็นซอฟต์แวร์ที่อนุญาตให้ใช้ได้โดยไม่ต้องเสียค่าใช้จ่ายใดๆ ภายใต้ลิขสิทธิ์แบบ GNU general public license (GPL) ของมูลนิธิ Free Software Foundation ในรูปรหัส source code ซึ่งสามารถคอมไพล์และทำงานได้บนระบบปฏิบัติการยูนิกซ์ตระกูลต่างๆ วินโดวส์และแมคอินทอช

สภาพแวดล้อม **R** เป็นซอฟต์แวร์ที่รวมเอาคุณสมบัติด้านการจัดการข้อมูล การคำนวณ และการแสดงทางกราฟิกไว้ด้วยกันอย่างดี โดยมีความสามารถในการจัดเก็บและจัดการข้อมูล สามารถคำนวณข้อมูลชนิด array และโดยเฉพาะ matrix ได้ มีเครื่องมือคือคำสั่งที่มีประสิทธิภาพสูงในการวิเคราะห์ข้อมูล มีความสามารถในการแสดงการวิเคราะห์ข้อมูลในทางกราฟิกทั้งบนหน้าจอและทางการพิมพ์ และยังเป็นภาษาคอมพิวเตอร์ที่ใช้งานง่ายและสามารถจัดการเงื่อนไข การทำงานวนซ้ำ และอื่นๆ อย่างครบถ้วน ดังนั้นคำว่า 'สภาพแวดล้อม' ในที่นี้จึงหมายถึงระบบที่มีการวางแผนและประสานสัมพันธ์กันอย่างดีตั้งแต่ขั้นการออกแบบไปจนถึงการทำงานและการแสดงผล ซึ่งต่างจากโปรแกรมวิเคราะห์ข้อมูลอื่นบางโปรแกรมที่นำชิ้นส่วนต่างๆ ที่มีที่มาหรือออกแบบมาต่างกันมารวมกันเพื่อให้ทำงานร่วมกัน การที่ **R** ถูกออกแบบให้เป็นระบบที่ทำงานได้ตั้งแต่การจัดการข้อมูล การวิเคราะห์ ไปจนถึงการแสดงผลกราฟิกโดยตัวเอง จึงทำให้การเขียนชุดคำสั่งเป็นไปอย่างต่อเนื่องตั้งแต่ต้นจนจบถึงการแสดงผล โดยไม่ต้องแบ่งการทำงานเป็นส่วนๆ ด้วยซอฟต์แวร์หลายตัว ซึ่งจะทำให้การทำงานยุ่งยากและซับซ้อนขึ้น

ในปี พศ. 2546 หน่วยระบาดวิทยา คณะแพทยศาสตร์ มหาวิทยาลัยสงขลานครินทร์ ได้รับทุนจาก**สำนักงานกองทุนสนับสนุนการวิจัย (สกว.)** ผ่านทาง**มูลนิธิสาธารณสุขแห่งชาติ (มสช.)** ให้พัฒนาโปรแกรม **R** เพื่อส่งเสริมให้มีการใช้งานโปรแกรมนี้เพื่อการวิเคราะห์ข้อมูลทางสถิติ ในหมู่นักวิชาการ นักวิจัย รวมถึงคนทั่วไปในประเทศไทย หนังสือเล่มนี้เป็นส่วนหนึ่งในโครงการดังกล่าว และนอกจากนี้ยังได้จัดทำเว็บไซต์อยู่ที่ http://medipe.psu.ac.th/R/ ซึ่งเป็นเว็บไซต์สำหรับการให้คำปรึกษาปัญหาในการใช้โปรแกรม **R** และสำหรับการเรียนการสอนโปรแกรม **R** ทางไกลให้แก่คนไทย ซึ่งเหมาะสำหรับผู้ใช้ขั้นเริ่มต้นและผู้ใช้ใหม่ทั่วไป และยังได้จัดทำเว็บไซต์ http://rforthai.blogspot.com เพื่อเผยแพร่ความรู้ในขั้นการเขียนฟังก์ชันและการทำแพคเกจในสภาพแวดล้อม **R** ซึ่งเหมาะสำหรับผู้ใช้ขั้นก้าวหน้าและขั้นสูง

บทที่ 1 สภาพแวดล้อม R

พัฒนาการของ **R** จากอดีตสู่ปัจจุบัน

การยอมรับ **R** ในวงการวิชาการ

R สู่อนาคต

จุดอ่อนของ **R**

ระบบคอมพิวเตอร์ที่ใช้ **R** ได้

การติดตั้ง **R**

เริ่มต้นใช้งาน **R**

ลองขอความช่วยเหลือจาก **R**

ระบบโฟลเดอร์และแฟ้มของโปรแกรม **R** และการตั้งค่าเริ่มต้น

การติดตั้งแพคเกจเพิ่มเติมเพื่อเพิ่มความสามารถให้ **R**

พัฒนาการของ R จากอดีตสู่ปัจจุบัน

R เป็นทั้งภาษาคอมพิวเตอร์และสภาพแวดล้อมในการคำนวณทางสถิติและกราฟิก ที่สร้างและพัฒนาภายใต้โครงการ GNU โดยมีลักษณะคล้ายภาษาและสภาพแวดล้อม **S** ซึ่งพัฒนาโดย John Chambers และคณะ แห่งห้องปฏิบัติการ Bell Laboratories ภาษาและสภาพแวดล้อม **S** นี้เองที่เป็นฐานของโปรแกรมสถิติที่มีชื่อการค้าว่า **S-PLUS** ซึ่งเป็นที่นิยมกันในต่างประเทศ ดังนั้นจึงอาจถือได้ว่า **R** เป็นส่วนขยายของ **S** และรหัสคำสั่งส่วนใหญ่ที่สร้างขึ้นสำหรับ **S** สามารถทำงานได้โดยไม่ต้องเปลี่ยนแปลงภายใต้ **R**

ในปัจจุบัน **R** ถูกพัฒนาอย่างต่อเนื่องจากกลุ่มบุคคลอิสระที่เรียกว่า **R** core team โดยทำงานพัฒนา **R** เพื่อเป็นซอฟต์แวร์สาธารณะโดยไม่มีผลตอบแทนทางการค้า บุคคลเหล่านั้นมีทั้งโปรแกรมเมอร์ และนักสถิติที่ทำงานประจำอยู่ในสถาบันการศึกษาหรือสถาบันวิชาการที่มีชื่อเสียงต่างๆ โดยมีเว็บไซต์อยู่ที่ http://www.r-project.org และได้รับการสนับสนุนทางการเงินจากมูลนิธิ R Foundation for Statistical Computing

R สามารถใช้ในการคำนวณทางสถิติได้อย่างกว้างขวาง เช่นเดียวกับภาษา **S** ที่ถูกนำไปใช้เป็นภาษาสำคัญในงานวิจัยในด้านวิธีการทางสถิติ ภาษา **R** จึงถูกนำไปใช้ในด้านนี้อย่างกว้างขวางด้วยเช่นกัน โดยมีข้อดีกว่าตรงที่ **R** เป็นซอฟต์แวร์ open source ซึ่งผู้ใช้สามารถใช้ได้โดยไม่ต้องเสียค่าใช้จ่ายและค่าลิขสิทธิ์ใดๆ

นอกจากความสามารถที่โดดเด่นในทางการคำนวณทางสถิติแล้ว จุดแข็งที่สำคัญอีกประการหนึ่งของ **R** คือความสามารถในด้านกราฟิกคุณภาพสูงในระดับตีพิมพ์ได้อย่างสวยงามด้วยคำสั่งที่ไม่ยุ่งยากนัก และสามารถใส่สัญลักษณ์ทางคณิตศาสตร์และสมการร่วมไปในกราฟได้ตามต้องการ แม้ว่าคำสั่งทางกราฟิกส่วนใหญ่ได้ตั้งค่าปริยายไว้อย่างเหมาะสมแล้ว แต่ผู้ใช้ก็ยังมีอิสระที่จะเปลี่ยนแปลงรูปแบบได้ตามต้องการ

และเนื่องจาก **R** เป็นภาษาคอมพิวเตอร์ภาษาหนึ่ง จึงเปิดโอกาสให้ผู้ใช้สามารถสร้างคำสั่งหรือฟังก์ชันใหม่ๆ เพื่อทำงานต่อยอดจากของเดิมที่มีอยู่แล้วได้ และเนื่องจาก **R** เป็นเสมือนสำเนียงหนึ่งของภาษา **S** ชุดคำสั่งที่สร้างสำหรับภาษา **S** จึงสามารถนำมาใช้กับ **R** ได้เกือบทั้งหมด (และในทางกลับกัน ชุดคำสั่งที่เขียนด้วย **R** ก็สามารถนำไปใช้กับโปรแกรมที่ทำงานกับภาษา **S** ได้เกือบทั้งหมดเช่นเดียวกัน) ผู้ใช้จึงสามารถแสวงหาคำสั่งที่เขียนขึ้นเผยแพร่ตามวารสารทางวิชาการสถิติหรือทางอินเตอร์เน็ตมาใช้ได้อย่างกว้างขวาง และนอกจากนี้ **R** ยังสามารถเชื่อมโยงกับภาษา **C**, **C++** และ **Fortran** ได้ด้วย ผู้ใช้ระดับเชี่ยวชาญหรือโปรแกรมเมอร์สามารถเขียนคำสั่งด้วยภาษาเหล่านั้นมาจัดการวัตถุในสภาพแวดล้อม **R** ได้ **R** จึงเป็นสภาพแวดล้อมเปิดที่เอื้ออำนวยการทำงานได้อย่างกว้างขวาง

ผู้ใช้จำนวนมากมักจะคิดว่า **R** เป็นโปรแกรมวิเคราะห์ข้อมูลทางสถิติ แต่กลุ่มผู้สร้าง **R** มักคิดว่า **R** เป็นสภาพแวดล้อมที่สามารถบรรจุความสามารถในทางสถิติไว้ภายใน โดยปกติแล้ว **R** จะมีความสามารถทำงานคำนวณพื้นฐานได้ในระดับหนึ่ง โดยการทำงานของแพคเกจพื้นฐาน 8 แพคเกจ แต่ข้อดีที่พิเศษอย่างยิ่งของ **R** คือสามารถเพิ่มความสามารถในการทำงานได้อย่างไม่จำกัดโดยการติดตั้งแพคเกจเพิ่มเข้าไปให้กับระบบ ซึ่งดาวน์โหลดได้จากเว็บไซต์ของ **CRAN** (Collaborative R Archive Network) ซึ่ง แพคเกจเหล่านั้นครอบคลุมการทำงานทางสถิติที่ทันสมัยในหลากหลายสาขา และสร้างขึ้นโดยนักวิชาการชั้นนำของโลกจำนวนมาก ที่ตั้งใจอุทิศผลงานให้กับวงวิชาการโดยไม่คิดค่าใช้จ่ายหรือลิขสิทธิ์ใดๆ

การยอมรับ R ในวงการวิชาการ

ดังได้กล่าวแล้วว่า **R** เป็นภาษาคอมพิวเตอร์ภาษาหนึ่งที่คล้ายคลึงกับภาษา **S** ซึ่งมีการนำไปใช้ในการวิจัยทางระเบียบวิธีทางสถิติอย่างกว้างขวางในวงวิชาการ **R** จึงได้รับการต้อนรับอย่างดีจากนักวิชาการทางสถิติ โดยมีผู้ร่วมในการผลิตแพคเกจและส่งให้กับเว็บไซต์ของ **CRAN** จำนวนมากอย่างสม่ำเสมอ

และนอกจากนี้ ยังมีการประชุมวิชาการผู้ใช้ **R** ในระดับโลกที่เรียกว่า **useR!** ซึ่งครั้งที่หนึ่งจัดขึ้นที่กรุงเวียนนา ประเทศออสเตรียเมื่อปี พศ. 2547 เรียกว่า **useR! 2004** และครั้งที่สองจัดในเดือนมิถุนายน พศ. 2549 เรียกว่าการประชุมวิชาการ **useR! 2006** ที่มหาวิทยาลัย Wirtschaftsuniversitat Wien ในกรุงเวียนนา ประเทศออสเตรีย ซึ่งเป็นเมืองที่เป็นที่ตั้งของมูลนิธิ **R** เพื่อการคำนวณทางสถิติ (**R** Foundation for Statistical Computing) นั่นเอง

ในการประชุมครั้งที่สองนั้น มุ่งเน้นในสามประเด็นหลักคือ 1. **R** ในฐานะที่เป็นภาษากลางของการวิเคราะห์ข้อมูลสถิติ 2. เป็นเวทีให้ผู้ใช้ **R** ได้ปรึกษาหารือ และแลกเปลี่ยนความคิดในการใช้ **R** ในด้านระเบียบวิธีทางสถิติ การวิเคราะห์ข้อมูล การแสดงผล และการนำไปใช้ในสาขาต่างๆ และ 3. นำเสนอภาพรวมของสิ่งใหม่ๆ ที่จะพัฒนาขึ้นในโครงการ **R** และนอกจากนี้ยังมีการอบรมหลักสูตรระยะสั้นในด้านต่างๆ

R สู่อนาคต

ชุมชนผู้ใช้ **R** ได้เติบโตขึ้นอย่างมากและรวดเร็ว และปัจจุบัน **R** เป็นที่ยอมรับและใช้กันในหมู่นักวิชาการและนักสถิติ จากสถาบันการศึกษาและองค์กรต่างๆ อย่างกว้างขวาง โดยเฉพาะกลุ่มประเทศในทวีปยุโรป ในการจัดหลักสูตรระยะสั้นขององค์กรนานาชาติเพื่อการวิจัยโรคมะเร็ง (International Agency for Research on Cancer) ซึ่งเป็นหน่วยงานสังกัดองค์การอนามัยโลกในปี

พศ. 2548 นั้น องค์กรดังกล่าวได้เลือกใช้ **R** ในการอบรม ทั้งนี้เพื่อจะได้แจกจ่ายแพคเกจและชุดคำสั่งต่างๆ ได้อย่างไม่ละเมิดลิขสิทธิ์ซอฟต์แวร์ทางการค้าใดๆ

ด้วยแนวคิดดังข้างต้น จึงเป็นไปได้ที่ **R** จะถูกนำไปใช้ในสถาบันการศึกษาและองค์กรต่างๆ ที่ต้องการเผยแพร่ผลงานให้แก่ชุมชนวิชาการอย่างกว้างขวาง เพราะจะไม่ติดขัดด้วยลิขสิทธิ์ซอฟต์แวร์ของโปรแกรมทางการค้า

และนอกจากนี้วารสารทางวิชาการต่างๆ ยอมรับผลการวิเคราะห์จาก **R** ทั้งสิ้น เนื่องจากนักวิชาการที่อ่านประเมินผลงานที่ส่งตีพิมพ์ในวารสารเหล่านั้นจำนวนไม่น้อยก็ใช้ **R** หรือ **S** ในการทำงานอยู่แล้วเช่นกัน

การพัฒนาโปรแกรม **R** นั้นเป็นไปค่อนข้างรวดเร็วมาก โดยปกติ **R** จะออกรุ่นใหม่ทุกๆ 3 เดือน และมีการแก้ไขข้อบกพร่องที่พบและออก patch ใหม่ทุกๆ สัปดาห์ แต่ส่วนใหญ่จะไม่เห็นความเปลี่ยนแปลงมากนัก เนื่องจากข้อบกพร่องที่แก้ไขมักเป็นเรื่องในระดับโปรแกรมที่ผู้ใช้งานทั่วไปมักไม่ได้ใช้ ขณะนี้ (ตุลาคม 2549) **R** เป็นรุ่นที่ 2.4.0 และกำลังทดสอบรุ่นที่ 2.5.0 อยู่ ซึ่งอาจจะออกใช้งานได้ภายในต้นหรือกลางปี 2550

ดังนั้นเป็นที่คาดว่า **R** จะมีพัฒนาการอย่างรวดเร็วและต่อเนื่องต่อไปอีกในระยะยาว เนื่องจากมีผู้ร่วมพัฒนาเป็นจำนวนมากจากทั่วทุกมุมโลก รวมทั้งในประเทศไทยด้วย โดยในประเทศไทยนั้น หน่วยงานแรกที่ใช้ **R** อย่างเป็นทางการในการเรียนการสอนนักศึกษา คือหน่วยระบาดวิทยา คณะแพทยศาสตร์ มหาวิทยาลัยสงขลานครินทร์ ในหลักสูตรระบาดวิทยานานาชาติ และหนังสือเล่มนี้ก็เป็นส่วนหนึ่งในความพยายามที่จะเผยแพร่การใช้งานโปรแกรม **R** ให้กว้างขวางในหมู่นักวิชาการและนักวิจัยไทย

จุดอ่อนของ R มีอะไรบ้าง

คงไม่มีซอฟต์แวร์ใดที่ไม่มีจุดอ่อน เพียงแต่ผู้ใช้จะสามารถยอมรับ และทำตัวให้เคยชินกับจุดอ่อนของซอฟต์แวร์นั้นได้หรือไม่เท่านั้น

จุดอ่อนที่สำคัญของ **R** ที่ผู้ใช้กล่าวถึงกันมาก คือการที่ **R** ทำงานแบบการพิมพ์บรรทัดคำสั่งทีละบรรทัด หรือเขียนเป็นแฟ้มสคริปต์เพื่อทำงานต่อเนื่องตั้งแต่ต้นจนจบ โดยทั่วไปนักสถิติขั้นสูงหรือนักวิจัยที่เชี่ยวชาญ จะชอบใช้โปรแกรมที่เขียนเป็นสคริปต์ได้เช่นนี้ เนื่องจากสามารถตรวจสอบคำสั่ง ลำดับการทำงาน แก้ไข และทำซ้ำได้ และโปรแกรมสถิติจำนวนไม่น้อยก็สามารถทำงานในแบบสคริปต์ได้ด้วย เช่นโปรแกรม **Stata** และ **SPSS** แต่ผู้ใช้ที่ต้องการเพียงการวิเคราะห์อย่างง่ายมักจะพบว่าการพิมพ์คำสั่งเช่นนี้เป็นการยุ่งยาก เนื่องจากจำคำสั่งไม่ได้

จุดอ่อนนี้จึงเป็นที่มาของหนังสือเล่มนี้นั่นเอง สภาพแวดล้อม **ICE** ที่จะกล่าวถึงในบทต่อไปจะเป็นคำตอบให้กับจุดอ่อนนี้ สภาพแวดล้อม **ICE** จะสามารถติดตั้งได้จากภายใน

สภาพแวดล้อม **R** และทำงานเหมือนหน้ากากที่ครอบสภาพแวดล้อมแบบการพิมพ์บรรทัดคำสั่ง เพื่อให้มีรายการเมนูและหน้าต่างให้ใส่ตัวเลือกต่างๆ สำหรับคำสั่งพื้นฐานที่ใช้บ่อยเป็นส่วนใหญ่ คำสั่งใดที่ไม่มีเมนูให้ ก็จะสามารถพิมพ์เข้าไปให้ทำงานได้ ซึ่งรายละเอียดต่างๆ จะได้กล่าวในบทต่อๆ ไปตลอดหนังสือทั้งเล่มนี้นั่นเอง

จุดอ่อนอื่นที่ผู้ใช้มักจะกล่าวถึงและก่อให้เกิดความสับสนแก่ผู้ใช้ใหม่คือการที่ **R** เป็นสภาพแวดล้อมแบบ object oriented คือสิ่งต่างๆ ในหน่วยความจำของ **R** จะเป็นวัตถุที่จะต้องมีชื่อกำกับเสมอ และอาจตั้งชื่อวัตถุเดียวกันเป็นหลายๆ ชื่อ ซึ่งก็จะทำให้มีข้อมูลซ้ำกันหลายชุดก็ได้ ซึ่งเป็นที่มาของความสับสน และ **R** ยังอนุญาตให้ attach กรอบข้อมูล (ชุดข้อมูล) เข้ากับเส้นทางค้นหาเพื่อจะได้ไม่ต้องเรียกชื่อข้อมูลภายในกรอบข้อมูลยาวเกินไป (จะได้กล่าวในบทที่เกี่ยวข้องต่อไป) แต่ก็มักจะทำให้ผู้ใช้เกิดความสับสนเมื่อทำการเปลี่ยนแปลงข้อมูลภายในกรอบข้อมูลนั้นได้ เพราะมักจะพบว่าข้อมูลไม่เปลี่ยนไปตามที่ต้องการเปลี่ยน (แต่ที่จริงเป็นความเข้าใจผิดของผู้ใช้เอง โดยเรียกดูข้อมูลผิดชุด) ซึ่งจุดอ่อนนี้เป็นสิ่งที่ต้องแลกกับความสามารถที่ **R** สามารถทำงานกับกรอบข้อมูลจำนวนมากได้พร้อมๆ กัน แต่เป็นสิ่งที่ผู้ใช้ระดับพื้นฐานมักไม่ได้ใช้ จึงรู้สึกว่าเป็นสิ่งที่ทำให้เกิดความสับสนมากกว่าจะได้ประโยชน์ แต่สำหรับผู้ใช้ระดับเชี่ยวชาญแล้ว จะพบว่าคุณสมบัตินี้คือจุดเด่นของ **R** ที่มีเหนือโปรแกรมอื่นนั่นเอง

จุดอ่อนอีกประการหนึ่งคือ **R** ไม่ได้รับประกันว่าจะไม่เกิดความผิดพลาดในการทำงาน และจะไม่ชดใช้ความเสียหายที่เกิดขึ้นหากเกิดความผิดพลาด เช่นทำให้โปรแกรมหยุดทำงานกลางคัน หรือทำให้เครื่องคอมพิวเตอร์หยุดทำงาน และทำให้ข้อมูลของผู้ใช้ต้องเสียหาย ที่กล่าวมานี้ไม่ใช่เรื่องที่เกิดขึ้นบ่อยหรือร้ายแรงแต่อย่างใด เพราะที่จริงโปรแกรมที่มีลิขสิทธิ์ทางการค้าอื่นๆ ก็มีโอกาสเกิดความผิดพลาดอย่างนี้ได้ทั้งสิ้น และในข้อความประกาศลิขสิทธิ์ก็มักจะกล่าวเช่นเดียวกันนี้ คือไม่รับประกันความผิดพลาดและไม่ชดใช้ความเสียหายที่เกิดขึ้น

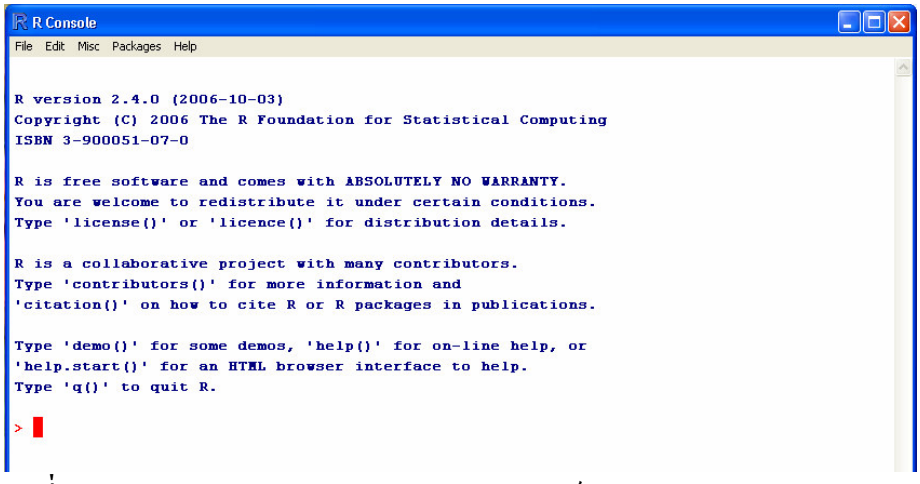
จึงมาถึงประเด็นที่ว่า **R** เกิดความผิดพลาดในการทำงานได้บ่อยเพียงใด เท่าที่ผู้เขียนได้ใช้งาน **R** มาตั้งแต่รุ่น 1.6 ก็พบว่าเกิดขึ้นได้น้อย และยิ่งน้อยมากในรุ่นปัจจุบัน ปัญหาส่วนใหญ่ที่เกิดขึ้นคือ **R** จะใช้หน่วยความจำที่มีอยู่ค่อนข้างมาก ซึ่งก็เป็นเรื่องธรรมดาที่โปรแกรมที่ทำงานด้านการคำนวณอย่างหนักจะต้องเป็นเช่นนี้ และหากผู้ใช้มีหน่วยความจำน้อย และพยายามเปิดโปรแกรมหลายโปรแกรมพร้อมกัน ก็จะทำให้โปรแกรมทำงานล่าช้าอย่างมากได้ และอาจถึงกับโปรแกรมใดโปรแกรมหนึ่งหยุดทำงานได้

จึงอาจสรุปได้ว่า **R** ใช้หน่วยความจำที่มีอยู่ค่อนข้างมาก ซึ่งก็เป็นเช่นเดียวกับโปรแกรมคำนวณทางสถิติอื่นๆ ผู้ใช้ที่มีหน่วยความจำคอมพิวเตอร์น้อย พึงต้องระมัดระวังในการเรียกใช้โปรแกรมที่ใช้หน่วยความจำมากพร้อมๆ กันหลายโปรแกรม

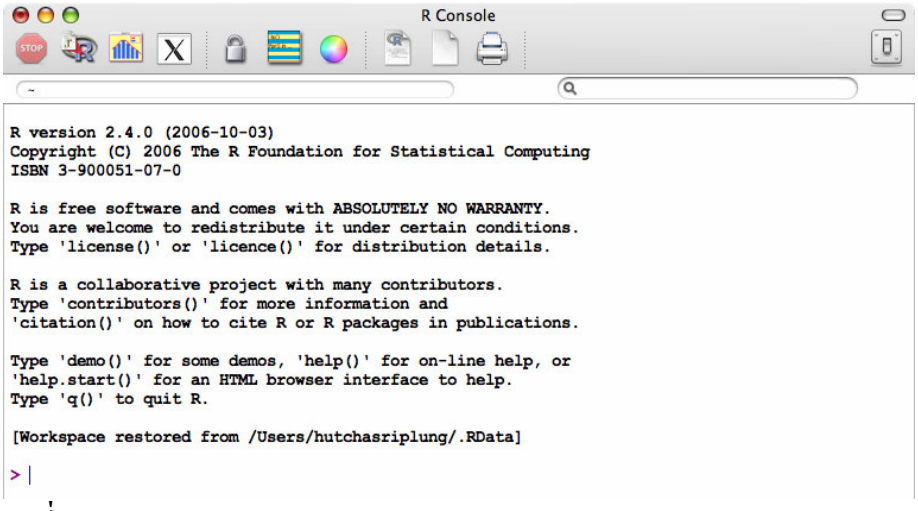
นอกจากนี้ยังมีจุดอ่อนอื่นอีกบางประการที่ผู้ใช้มือใหม่มักจะสับสน แต่อย่างที่กล่าวแล้วว่าไม่มีโปรแกรมใดที่ไม่มีจุดอ่อน ดังนั้นผู้ใช้ควรทำความเข้าใจสภาพแวดล้อม **R** ให้ดีโดยการอ่านคู่มือการใช้งานให้รอบคอบ และค่อยๆ เรียนรู้ไปตามลำดับ ก็จะอยู่กับ **R** ได้อย่างมีความสุขและใช้ประโยชน์ได้อย่างเต็มที่

ระบบคอมพิวเตอร์ที่ใช้ R ได้

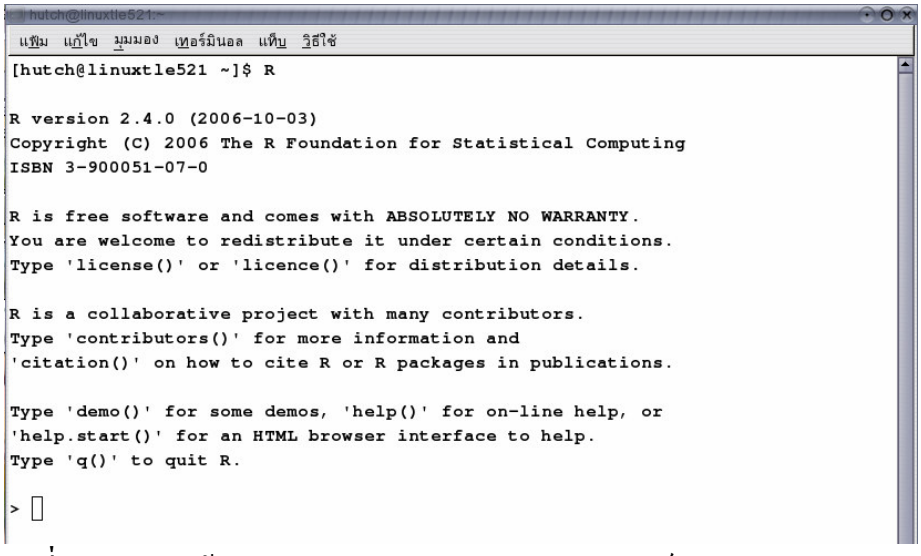
R ใช้ได้กับระบบปฏิบัติการหลักทุกระบบได้แก่ วินโดวส์ ลินุกซ์ และ แมคอินทอช OSX เมื่อเข้าไปที่เว็บไซต์ **CRAN** จะสามารถดาวน์โหลดโปรแกรมมาใช้งานได้กับทุกระบบปฏิบัติการหลักดังกล่าว และเนื่องจากระบบปฏิบัติการลินุกซ์มีหลายค่ายและหลายรุ่นมากจนไม่สามารถเตรียมตัวติดตั้งให้ได้ทั้งหมด ทาง **CRAN** จึงได้ให้ดาวน์โหลด source code เพื่อผู้ใช้จะได้ make ให้เข้ากับระบบปฏิบัติการในเครื่องของแต่ละคนได้เองด้วย สำหรับระบบปฏิบัติการวินโดวส์และแมคอินทอช OSX ซึ่งมีรุ่นที่แตกต่างกันไม่มากนักและมักใช้งานร่วมกันได้ จึงสามารถทำตัวติดตั้งให้กับระบบปฏิบัติการทั้งสองได้ การติดตั้ง **R** จึงทำได้อย่างง่ายดาย ลองดูตัวอย่างหน้าตาของ **R** บนระบบปฏิบัติการต่างๆ ในรูปต่อไปนี้



รูปที่ 1.1 **R** ใน RGUI บนระบบปฏิบัติการวินโดวส์



รูปที่ 1.2 **R** ใน RGUI บนระบบปฏิบัติการแมคอินทอช OSX

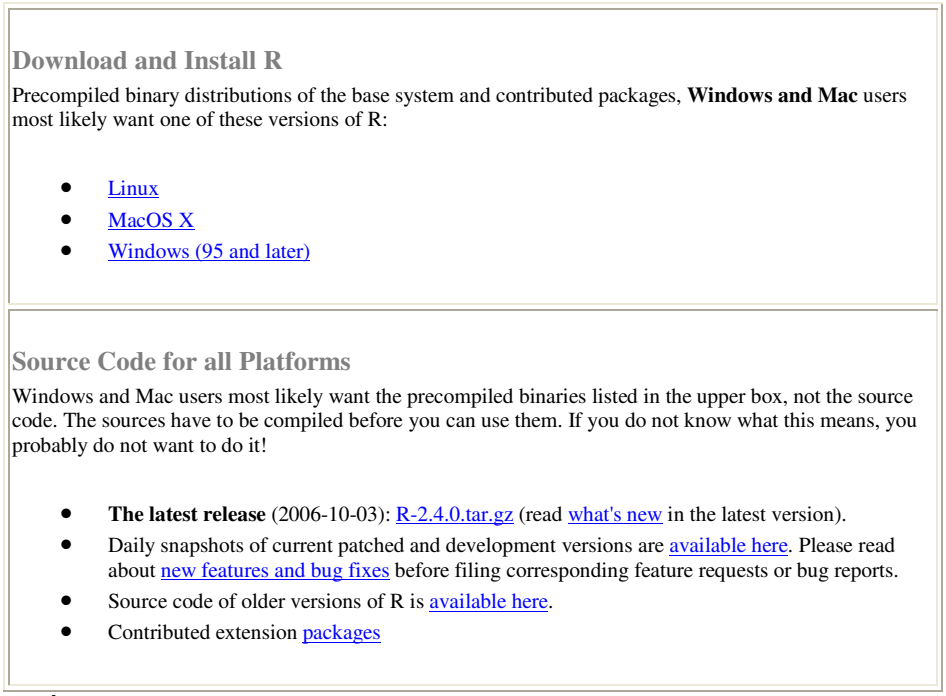


รูปที่ 1.3 **R** ในหน้าต่าง terminal บนระบบปฏิบัติการลินุกซ์

จะเห็นได้ว่าบนหน้าจอของ **R** บนทุกระบบปฏิบัติการมีข้อความเหมือนกันทั้งสิ้น และจะมีเครื่องหมาย > ที่ด้านล่างคือ prompt ของ **R** นั่นเอง นั่นคือ **R** จะทำงานเหมือนกันและได้ผลลัพธ์ในการคำนวณเหมือนกันบนทุกระบบปฏิบัติการ เพียงแต่รายการเมนูบนหน้าต่างจะแตกต่างกันไป ซึ่งนั่นเป็นเพียงส่วนเสริมให้การทำงานง่ายขึ้นเท่านั้น ไม่ใช่การทำงานหลักของ **R** แต่อย่างใด

การติดตั้ง R

สำหรับผู้ใช้ที่ต้องการใช้โปรแกรมรุ่นใหม่ล่าสุด สามารถดาวน์โหลดได้ที่เว็บไซต์ **CRAN** ที่ http://cran.r-project.org ซึ่งจะมีหน้าจอดังรูปที่ 1.4



Download and Install R

Precompiled binary distributions of the base system and contributed packages, **Windows and Mac** users most likely want one of these versions of R:

- Linux
- MacOS X
- Windows (95 and later)

Source Code for all Platforms

Windows and Mac users most likely want the precompiled binaries listed in the upper box, not the source code. The sources have to be compiled before you can use them. If you do not know what this means, you probably do not want to do it!

- **The latest release** (2006-10-03): R-2.4.0.tar.gz (read what's new in the latest version).
- Daily snapshots of current patched and development versions are available here. Please read about new features and bug fixes before filing corresponding feature requests or bug reports.
- Source code of older versions of R is available here.
- Contributed extension packages

รูปที่ 1.4 หน้าเว็บดาวน์โหลดโปรแกรม **R** จาก **CRAN** ที่ http://cran.r-project.org

ผู้ใช้ระบบปฏิบัติการวินโดวส์ให้เลือกตัวเลือก Windows (95 and later) ซึ่งเมื่อเข้าไปแล้วให้เลือก subdirectory ที่ชื่อ base จะได้แฟ้มติดตั้งที่มีนามสกุลเป็น EXE เช่น R-2.4.0-win32.exe เมื่อ คลิกตัวติดตั้งนี้ก็จะโปรแกรมไปตามค่าปริยายที่ตั้งไว้ โดยจะติดตั้งที่ ..\Program Files\R\R-2.4.0 หากไม่ทราบความหมายของตัวเลือกใดก็ไม่ควรเปลี่ยนแปลงไปจากค่าปริยายนั้น เมื่อติดตั้งเสร็จแล้ว จะปรากฏ **R** ขึ้นบนรายการเมนู Start และมี R shortcut ขึ้นบน desktop ด้วย

> **หมายเหตุ** ในหนังสือเล่มนี้ ผู้เขียนใช้สัญลักษณ์ .. หน้า \ (สำหรับวินโดวส์) หรือ / (สำหรับลินุกซ์และแมคอินทอช OSX) โดยจะหมายถึงโฟลเดอร์หรือฮาร์ดดิสก์หลักที่โฟลเดอร์ที่กล่าวถึงถูกติดตั้งอยู่ เช่น อาจเป็น C: หรือ D: หรือโฟลเดอร์อื่นก็ได้ เช่น ..\Program Files ก็หมายถึงฮาร์ดดิสก์ที่โฟลเดอร์ Program Files ถูกติดตั้งอยู่ และหากเป็น ..\R\R-2.4.0 ก็หมายถึงโฟลเดอร์ที่ R\R-2.4.0 ถูกติดตั้งอยู่

ในอนาคตหากมีรุ่นใหม่ออกมา เช่นรุ่น 2.5.0 **R** จะไม่ถูกติดตั้งทับของเก่า แต่จะติดตั้งในโฟลเดอร์ ..\Program Files\R\R-2.5.0 คือมีการสร้างโฟลเดอร์ย่อยสำหรับรุ่น 2.5.0 ให้ใหม่นั่นเอง ซึ่งก็หมายความว่าหากมีการติดตั้งแพคเกจใดๆ ไว้ใน R-2.4.0 ก็จะต้องติดตั้งใหม่เพื่อให้ใช้กับ R-2.5.0 ได้

สำหรับผู้ใช้ระบบปฏิบัติการแมคอินทอช OSX ให้เลือกตัวเลือก MacOSX ซึ่งเมื่อเข้าหน้าต่อไปจะเห็นรุ่นล่าสุดอยู่ด้านบนสุด เมื่อดาวน์โหลดจะได้แฟ้ม disk image เช่น R-2.4.0.dmg เมื่อ

ขยายแฟ้มนี้ออกจะได้ disk image ของ R-2.4.0 ซึ่งภายในจะมีแฟ้ม pkg อยู่ ให้ double click แฟ้มนี้ ระบบจะถามรหัสผ่านของเจ้าของเครื่อง เมื่อตอบถูกต้องจะติดตั้งตามขั้นตอนปกติของแมคอินทอชนั่นเอง แต่เนื่องจากระบบแมคอินทอช OSX รุ่น 10.3.x ต่างจากรุ่น 10.4.x ตรงที่รุ่นหลังนี้จะใช้กับเครื่องที่ใช้ชิพของ Intel ได้ แต่ตัวติดตั้งแพคเกจของสองรุ่นแตกต่างกัน จึงต้องดูให้ดีว่าตัวติดตั้งนั้นๆ ใช้กับ OS รุ่นที่เครื่องใช้อยู่หรือไม่ และเช่นเดียวกัน หากติดตั้งรุ่นใหม่ในอนาคต **R** รุ่นใหม่จะถูกติดตั้งในโฟลเดอร์ใหม่ ไม่ได้ติดตั้งทับของเก่าเช่นกัน

ส่วนผู้ใช้ระบบปฏิบัติการลินุกซ์จะยุ่งยากกว่าระบบอื่น เนื่องจากมีลินุกซ์หลายตระกูล การติดตั้งควร login เข้าเป็น root เมื่อเลือกตัวเลือกบนสุดของหน้าจอเว็บไซต์ของ **CRAN** จะมีทางเลือกให้เข้าดาวน์โหลดแฟ้มติดตั้งตามตระกูลของลินุกซ์ที่ใช้ ได้แก่ Debian, Mandrake, Red Hat, SUSE และ Vinelinux สำหรับตระกูล Red Hat (ซึ่งรวมถึงลินุกซ์ TLE ด้วย) ให้เลือก Red Hat ซึ่งจะมีตัวเลือกต่อไปว่าเป็น el3, el4, fc3 หรือ fc4 แล้วยังมีตัวเลือกย่อยอีกชั้นหนึ่ง เมื่อเลือกและดาวน์โหลดแล้วจะได้แฟ้มติดตั้งที่เป็น rpm หากหาตัวติดตั้งที่ตรงกับตระกูลและรุ่นที่ใช้งานไม่ได้ก็จะต้อง make application ขึ้นเอง

เมื่อดาวน์โหลดโปรแกรมติดตั้งมาแล้ว ก็ติดตั้งตามวิธีของลินุกซ์ตระกูลนั้นๆ เช่นลินุกซ์ TLE ก็ดาวน์โหลดโปรแกรมที่เป็น rpm มา เมื่อติดตั้งแล้วจะพบว่า **R** ถูกติดตั้งที่ /usr/bin หรือ /usr/local/bin หรือหากไม่พบในที่ดังกล่าว ให้ลองค้นหาด้วยคำหลัก survival ซึ่งเป็นแพคเกจย่อยภายใน **R** อย่าหาด้วยคำหลัก R เพราะเป็นคำที่สั้นเกินไป จะทำให้พบแฟ้มจำนวนมาก

ในกรณีที่ไม่มีโปรแกรมติดตั้งจำเพาะสำหรับลินุกซ์ที่ใช้อยู่ ผู้ใช้จำเป็นต้อง compile จาก source เอง ซึ่งในเครื่องจะต้องมี compiler อยู่ด้วย ก็ให้ดาวน์โหลดแฟ้มที่เป็น tar.gz ตรงช่องด้านล่าง เช่นในรูปที่ 1.4 ข้างต้นเป็น R-2.4.0.tar.gz เมื่อได้มาแล้วจึงขยายออกไว้ใน home แล้วจึง make ตามขั้นตอนในแฟ้มเอกสาร Readme ที่อยู่ในโฟลเดอร์ที่ขยายออกมาแล้วนั้น ซึ่งโดยทั่วไปจะให้ configure compiler ก่อนแล้วจึง make โดยพิมพ์คำสั่งในหน้าต่าง terminal ตามขั้นตอนดังนี้

```
./configure
make
```

ซึ่งแต่ละขั้นตอนจะใช้เวลานานสักเล็กน้อย เมื่อเสร็จกระบวนการข้างต้น จะมีการสร้างโฟลเดอร์ย่อยและแฟ้มทำงานต่างๆ ในโฟลเดอร์ R-2.4.0 ที่ถูกขยายออกมาจากแฟ้ม R-2.4.0.tar.gz นั่นเอง และในการติดตั้งครั้งแรก **R** จะถูกติดตั้งที่ home ให้ย้ายทั้งโฟลเดอร์นั้นไปติดตั้งโฟลเดอร์ R ใน /usr/bin หรือ /usr/local/lib จากนั้นจึงสร้างแฟ้ม R shell script เพื่อตั้งค่าต่างๆ และเรียกใช้งาน **R** จากหน้าต่าง terminal จากโฟลเดอร์ที่ **R** อยู่อย่างถูกต้อง โดยสร้างไว้ในโฟลเดอร์ /usr/bin หรือ /usr/local/bin ตามที่ได้ติดตั้งไว้ด้วย

ในการติดตั้งโปรแกรมรุ่นใหม่ต่อๆ ไป หากติดตั้งจากโปรแกรมติดตั้งอาจมีปัญหาว่าโปรแกรมจะเขียนทับแฟ้มเดิมทั้งหมด ขณะที่ฟังก์ชันบางอย่างอาจเปลี่ยนไป และทำให้การเรียก

แฟ้มสคริปต์ที่เขียนไว้สำหรับ **R** รุ่นก่อนหน้านี้ทำงานผิดไปจากที่เคยทำงานก็ได้ (ส่วนใหญ่ผู้ใช้ที่ต้องวิเคราะห์ข้อมูลซับซ้อนมีความจำเป็นที่จะต้องเขียนแฟ้มสคริปต์เพื่อทำงาน ผู้ใช้ที่ต้องส่งแฟ้มข้อมูลและแฟ้มสคริปต์ให้กับแหล่งทุน หรือนักศึกษาระดับหลังปริญญาที่จำเป็นต้องส่งแฟ้มสคริปต์วิเคราะห์ข้อมูลเพื่อการสอบวิทยานิพนธ์ ส่วนผู้ใช้ระดับพื้นฐานอาจไม่ต้องเขียนแฟ้มสคริปต์ในการทำงานก็ได้) หรือผู้ใช้ที่ต้องการเก็บสำเนา **R** รุ่นเก่าไว้ในเครื่อง สามารถทำได้โดยเปลี่ยนชื่อโฟลเดอร์ R เป็น R-x.x.x (x.x.x คือเลขรุ่นของ **R** เช่น อาจเป็น 2.4.0 หรือรุ่นอื่นๆ ที่ติดตั้งในเครื่องของเราอยู่) และเปลี่ยนชื่อ R shell script เป็น R-x.x.x เช่นกัน และต้องเข้าไปแก้ไขการตั้งค่าในแฟ้ม R shell script นั้นให้ชี้ไปยังชื่อโฟลเดอร์ที่เปลี่ยนใหม่ด้วย เช่น จาก

```
R_HOME_DIR=/usr/lib/R
```

ก็เปลี่ยนเป็น

```
R_HOME_DIR=/usr/lib/R.x.x.x
```

จากนั้นเมื่อติดตั้ง **R** รุ่นใหม่ ก็จะติดตั้งเสมือนว่าไม่เคยมี **R** อยู่ในเครื่องมาก่อน นั่นคือขั้นตอนนี้เป็นการเลียนแบบกระบวนการติดตั้งบนวินโดวส์ ที่แยกรุ่นของ **R** ไว้คนละโฟลเดอร์กันไม่ปะปนกันนั่นเอง จำไว้ว่ากระบวนการนี้ต้องทำก่อนติดตั้งโปรแกรมรุ่นใหม่ เพื่อไม่ให้ติดตั้งทับโปรแกรมรุ่นเก่าไปเสียก่อน

สำหรับผู้เขียนแล้ว ในการติดตั้งโปรแกรมรุ่นใหม่จะนิยมใช้วิธี compile ใหม่จาก source เลยดังที่ได้กล่าวแล้วข้างต้น แต่ถ้าในเครื่องที่ไม่มี compiler อยู่จะทำเช่นนั้นไม่ได้ ที่ทำเช่นนี้ก็เพื่อที่เราจะสามารถจัดการแฟ้มต่างๆ ได้เอง โดยการสำเนา โยกย้าย และเปลี่ยนชื่อ หรือเปลี่ยนข้อความภายในแฟ้มดังที่กล่าวแล้วข้างต้นเป็นขั้นตอนตามลำดับ

ผู้ใช้ที่ต้องการทำเช่นนี้ เช่นสมมุติว่าในอนาคตจะติดตั้ง R-2.5.0 แนะนำว่าหลังจาก make เสร็จ ให้เข้าไปเปลี่ยนชื่อโฟลเดอร์ R ใน /usr/lib เป็น R-2.4.0 แล้วย้ายโฟลเดอร์ R-2.5.0 ใน home ไปไว้ที่ /usr/lib แล้วเปลี่ยนชื่อโฟลเดอร์เป็น R (คือลบ -2.5.0 ออกจาก R-2.5.0 นั่นเอง) เมื่อเรียก R ในหน้าต่าง terminal หลังจากนี้ ระบบจะเข้าไปเรียก **R** ที่เป็นของรุ่น 2.5.0 แล้วสำเนาแฟ้ม R shell script ใน /usr/bin ให้เป็นชื่อ R-2.4.0 แล้วเข้าไปเปลี่ยน R_HOME_DIR ให้ชี้ไปที่ usr/lib/R-2.4.0 และตรวจสอบให้ดีว่าตัวแปรอื่นๆ ได้แก่ R_SHARE_DIR, R_INCLUDE_DIR, และ R_DOC_DIR ในบรรทัดต้นๆ ของแฟ้ม ได้ชี้ไปที่โฟลเดอร์ที่อยู่ใน R-2.4.0 เช่นกัน เมื่อพิมพ์ R-2.4.0 ที่หน้าต่าง terminal ก็จะเรียกใช้ **R** รุ่น 2.4.0 มาใช้งานได้ จะเห็นว่าวิธีการเช่นนี้จะทำให้การติดตั้งคล้ายคลึงกับการติดตั้งบนระบบปฏิบัติการวินโดวส์และแมคอินทอช OSX เช่นกันนั่นเอง

เริ่มต้นใช้งาน R

เนื่องจากผู้ใช้ส่วนใหญ่ใช้ระบบปฏิบัติการวินโดวส์ หนังสือเล่มนี้จึงใช้ตัวอย่างรูปภาพจากระบบปฏิบัติการวินโดวส์เป็นหลัก แต่หากมีส่วนใดที่มีความแตกต่างกันในระบบปฏิบัติการที่ต่างกันแล้ว ก็จะได้กล่าวถึงหรือแสดงรูปการทำงานหรือผลการทำงานบนระบบปฏิบัติการอื่นด้วย

บนระบบปฏิบัติการวินโดวส์นั้น สามารถเรียกใช้ **R** ได้ง่ายโดยเรียกจาก desktop shortcut นั่นเอง หรืออาจเรียกจาก Start >> All Programs >> R ก็ได้ บนระบบปฏิบัติการแมคอินทอช OSX นั้นเรียกได้จาก Application ถ้าหากใช้งานบ่อยก็อาจติดตั้ง **R** ไว้ที่ dock ด้วยก็ได้ ส่วนบนระบบปฏิบัติการลินุกซ์นั้น ให้เปิดหน้าต่าง terminal ขึ้นมาก่อน แล้วเรียก **R** โดยพิมพ์ R (อักษรตัวพิมพ์ใหญ่) ที่ prompt จากนั้นจะได้หน้าต่างตามรูปที่ 1.4 ข้างต้น

> **หมายเหตุ** ในที่นี้เมื่อเขียนถึงลำดับในการเรียกรายการเมนูย่อยจากเมนูที่ใหญ่กว่า จะใช้เครื่องหมาย >> โดยที่รายการเมนูที่ใหญ่กว่าจะเขียนไว้ทางซ้าย และรายการเมนูย่อยจะเขียนไว้ทางขวา เช่น Start >> All Programs หมายถึง รายการเมนูย่อย All Programs ในรายการเมนูใหญ่ Start

ข้อความนำภายในหน้าต่างบนระบบปฏิบัติการทุกระบบจะเหมือนกันหมดดังนี้

```
R version 2.4.0 (2006-10-03)
Copyright (C) 2006 The R Foundation for Statistical Computing
ISBN 3-900051-07-0

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

>
```

จะเห็นว่าในการออกคำสั่งในสภาพแวดล้อม **R** นั้น จะต้องเขียนคำสั่งแล้วตามด้วยวงเล็บเสมอ ภายในวงเล็บอาจมี argument หรือไม่ก็ได้ ซึ่งจะได้กล่าวถึงในภายหลังต่อไป เช่น หากต้องการความช่วยเหลือ ให้พิมพ์ `help()` และหากต้องการเลิกงานให้พิมพ์ `q()` และเครื่องหมาย > ที่ด้านล่างคือ prompt ของ **R** นั่นเอง

เนื่องจากโดยปกติแล้ว **R** จะทำงานด้วยการพิมพ์บรรทัดคำสั่งหลัง prompt สิ่งที่ต้องระวังคือ ตัวพิมพ์ใหญ่และตัวพิมพ์เล็ก มีความแตกต่างกันในการออกคำสั่งและพิมพ์ข้อความใน

สภาพแวดล้อม **R** เหมือนการทำงานในสภาพแวดล้อมยูนิกซ์ทั้งหลาย เช่นบนระบบปฏิบัติการลินุกซ์ หรือแมคอินทอช OSX

เมนูด้านบนของ RGUI ของระบบปฏิบัติการแต่ละระบบมีลักษณะที่แตกต่างกันไป ซึ่งจะยังไม่กล่าวถึงในที่นี้ จะกล่าวถึงแต่ละเรื่องที่จำเป็นต้องใช้เท่านั้นเมื่อกล่าวถึงเรื่องนั้นๆ โดยตรง

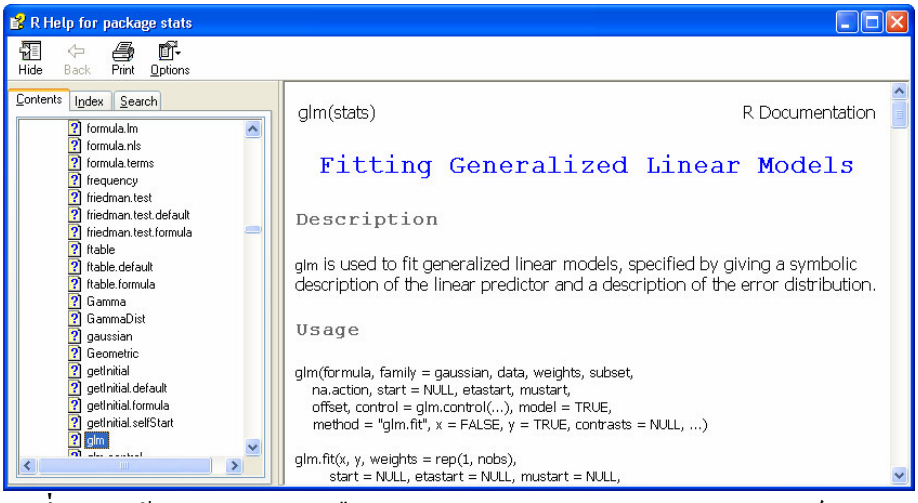
ลองขอความช่วยเหลือจาก R

การขอความช่วยเหลือโดยคำสั่ง `help()`

การขอความช่วยเหลือใน **R** สามารถทำได้หลายวิธี หากจำชื่อฟังก์ชันที่ต้องการดูความช่วยเหลือเกี่ยวกับการ argument ที่จะให้กับฟังก์ชัน ก็สามารถทำได้โดยการใช้คำสั่ง `help()` แล้วใส่ชื่อของฟังก์ชันนั้นไว้ภายในวงเล็บดังตัวอย่าง

```
help(glm)
```

บนระบบปฏิบัติการวินโดวส์นั้น ค่าปริยายของการให้ความช่วยเหลือถูกตั้งไว้ที่ chtml คือ chmhelp = TRUE และจะปรากฏหน้าจอความช่วยเหลือขึ้นเป็นหน้าต่างหนึ่งอธิบายวิธีการใช้งานฟังก์ชันนั้นในหน้าต่างดังรูปที่ 1.5



รูปที่ 1.5 หน้าจอความช่วยเหลือแบบ chtml บนระบบปฏิบัติการวินโดวส์

ความช่วยเหลือแบบ chtml มีข้อดีเหนือรูปแบบอื่นตรงที่มีการจัดเรียงหัวข้อความช่วยเหลือดัชนี และการค้นหาคำอยู่ทางด้านซ้ายของหน้าต่างด้วย ทำให้การค้นหาความช่วยเหลือมีประสิทธิภาพมากขึ้น

สำหรับระบบปฏิบัติการลินุกซ์นั้น ค่าปริยายของการให้ความช่วยเหลือถูกตั้งไว้เป็น text และข้อความช่วยเหลือนี้จะปรากฏบนหน้าต่าง terminal นั่นเอง ผู้ใช้จะออกจากโหมดความช่วยเหลือโดยการกดปุ่มอักษร q

ส่วนบนระบบปฏิบัติการแมคอินทอช OSX นั้น ค่าปริยายของการให้ความช่วยเหลือเป็น html ซึ่งก็คือรูปแบบเช่นเดียวกับเว็บนั่นเอง แต่การแสดงผลจะใช้ browser ของ RGUI ไม่ใช่ web browser ตามปกติ

ผู้ที่ใช้วินโดวส์หรือลินุกซ์อาจเปลี่ยนการแสดงความช่วยเหลือไปสู่ html ได้เช่นกัน ซึ่งจะทำให้ข้อความช่วยเหลือถูกแสดงโดยโปรแกรม web browser เช่น **Internet Explorer** หรือ **FireFox** โดยระบุเป็นตัวเลือก `html` ให้เป็น `TRUE` ดังตัวอย่าง

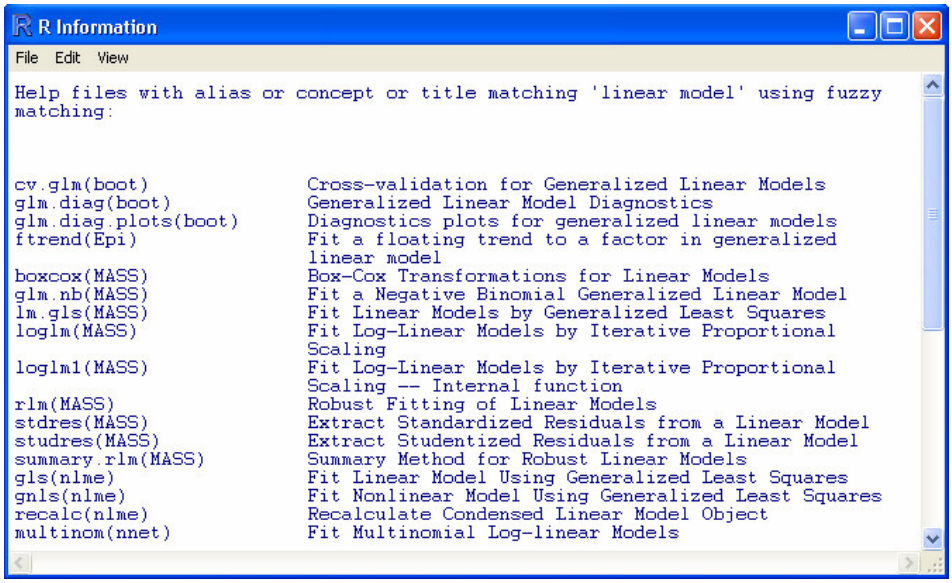
```
help(glm, html=TRUE)
```

การขอความช่วยเหลือโดยคำสั่ง `help.search()`

หากจำชื่อฟังก์ชันที่ต้องการขอดูความช่วยเหลือไม่ได้ **R** ยังเปิดโอกาสให้ค้นหาความช่วยเหลือโดยใช้คำหลักได้ด้วยการใช้คำสั่ง `help.search()` ซึ่งผู้ใช้สามารถใส่คำหลักที่ต้องการค้นหาไว้ภายในวงเล็บ และจะต้องเขียนเครื่องหมาย `""` ครอบคำหลักนั้น

> **หมายเหตุ** เมื่อจะอ้างถึงข้อความหรือชื่อแฟ้มใน **R** จะต้องเขียนไว้ภายในเครื่องหมาย `""` เสมอ มิฉะนั้น **R** จะตีความเป็นวัตถุที่อยู่ภายในหน่วยความจำ ซึ่งจะไม่ตรงตามความต้องการที่ผู้ใช้อยากให้ทำงาน

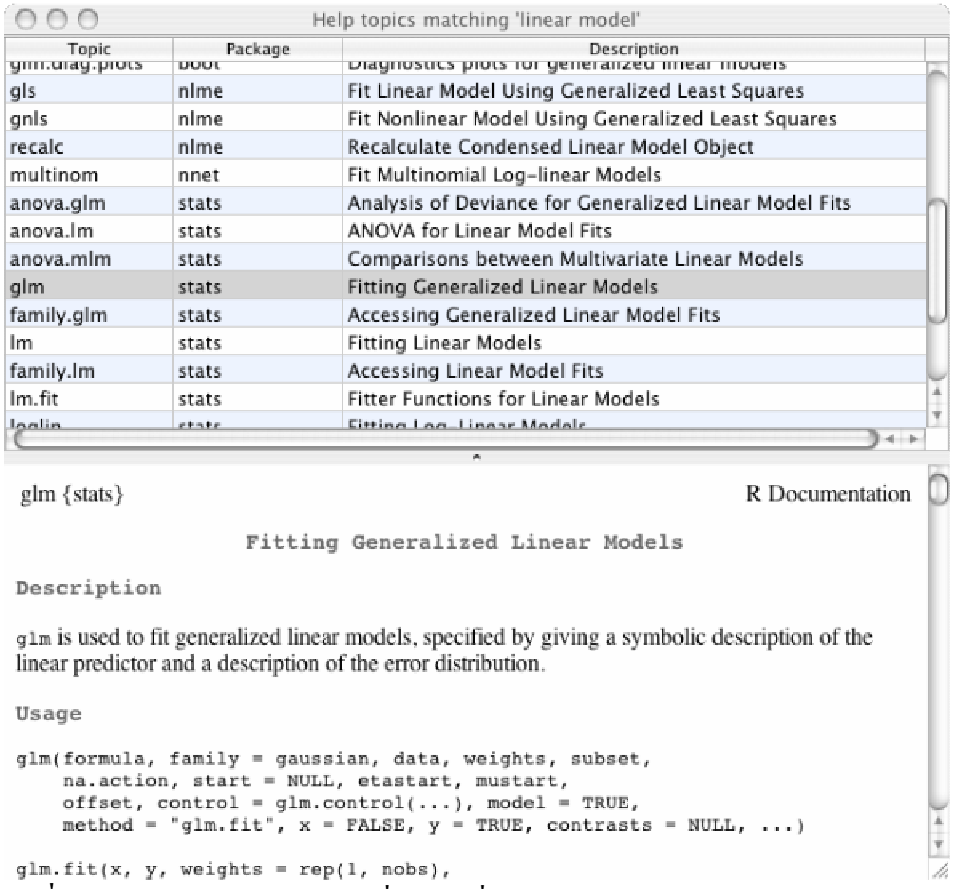
ลองค้นหาความช่วยเหลือโดยใช้คำหลักเป็น `"linear model"` ดังนี้



รูปที่ 1.6 หน้าจอความช่วยเหลือเมื่อใช้คำสั่ง `help.search("linear model")` บนระบบวินโดวส์

เลื่อนแถบเลือกด้านขวาของจอภาพเพื่อเลือกคำสั่งที่คิดว่าน่าจะตรงกับที่ต้องการ จากนั้นใช้คำสั่ง `help()` ดูความช่วยเหลือของคำสั่งนั้นต่อไป

บนระบบปฏิบัติการแมคอินทอช OSX จะได้หน้าจอที่แตกต่างไปเล็กน้อย คือนอกจากจะมีรายการคำสั่งที่มีคำดังกล่าวแล้ว ยังสามารถดูความช่วยเหลือของแต่ละคำสั่งได้เลยด้วย



รูปที่ 1.7 หน้าจอความช่วยเหลือเมื่อใช้คำสั่ง `help.search("linear model")` บนระบบแมคอินทอช OSX

ส่วนบนระบบลินุกซ์นั้น จะได้รายการคำสั่งบนหน้าต่าง terminal นั้นเอง ซึ่งเมื่อเลือกได้คำสั่งที่คิดว่าน่าจะตรงกับที่ต้องการแล้ว ก็ใช้คำสั่ง `help()` ดูความช่วยเหลือของคำสั่งนั้นต่อไป เช่นเดียวกับบนระบบวินโดวส์

การขอความช่วยเหลือโดยคำสั่ง `help.start()`

นอกจากนี้ **R** ยังเปิดโอกาสให้ค้นหาโดยดูจากรายการแพคเกจทั้งหมดที่ติดตั้งอยู่ในเครื่องด้วยคำสั่ง `help.start()` ด้วย ซึ่งเมื่อออกคำสั่งนี้โดยไม่ต้องใส่อะไรในวงเล็บ **R** จะเปิดเว็บบราวเซอร์ที่ตั้งไว้เป็นค่าปริยายของเครื่องนั้นขึ้นมาดังนี้

รูปที่ 1.8 หน้าจอความช่วยเหลือเมื่อใช้คำสั่ง `help.start()`

จะเห็นได้ว่าคำสั่ง `help.start()` นำไปสู่การเลือกอ่านเอกสารแนะนำ **R** และเอกสารอื่นๆ ที่ผลิตโดย **CRAN** และจากตัวเลือก Packages ผู้ใช้จะสามารถดูรายการฟังก์ชันในแพคเกจต่างๆ ที่ติดตั้งอยู่ในเครื่องนั้นได้ และยังจะค้นหาด้วยคำหลักเช่นเดียวกับคำสั่ง `help.search()` ได้ด้วยตัวเลือก Search Engine & Keywords อีกด้วย

ระบบโฟลเดอร์และแฟ้มของโปรแกรม R และการตั้งค่าเริ่มต้น

การจัดระบบโฟลเดอร์และแฟ้มของโปรแกรม **R** บนระบบปฏิบัติการที่แตกต่างกันก็มีความแตกต่างกันอยู่บ้าง แต่โดยทั่วไปจะมีการจัดโฟลเดอร์และแฟ้มคล้ายๆ กัน

บนระบบปฏิบัติการวินโดวส์นั้น ค่าปริยายของการติดตั้งโปรแกรม **R** จะติดตั้งที่โฟลเดอร์ Program Files โดยจะมีโฟลเดอร์หลักชื่อ R ภายในมีโฟลเดอร์ย่อยชื่อ R-2.4.0 ซึ่งภายในมีโฟลเดอร์ย่อยที่สำคัญดังนี้

..\R\R-2.4.0	- bin	เก็บตัวโปรแกรมหลักของ **R**
	- doc	เก็บแฟ้มความช่วยเหลือ
	- etc	เก็บแฟ้มการตั้งค่าของสภาพแวดล้อม **R**
	- library	เป็นที่อยู่ของแพคเกจต่างๆ
	- …	

หมายเหตุ เมื่อติดตั้งโปรแกรม **R** ในรุ่นถัดไป ตัวติดตั้งจะไม่ติดตั้งโปรแกรมทับในโฟลเดอร์เดิม แต่จะติดตั้งในโฟลเดอร์ใหม่ เช่น ..\R\R-2.5.0 โดยจะยังคงมีโฟลเดอร์ ..\R\R-2.4.0 อยู่ในที่เดิม หากผู้ใช้ได้ติดตั้งแพคเกจเพิ่มเติมอื่นใด แพคเกจเหล่านั้นจะไม่ถูกติดตั้งในโปรแกรม **R** รุ่นใหม่ที่ติดตั้งทีหลังนั้น ผู้ใช้จะต้องติดตั้งแพคเกจให้กับโปรแกรม **R** ที่ติดตั้งใหม่นั้นเอง

แฟ้มสำหรับการตั้งค่าเริ่มต้นเมื่อเริ่มเรียกใช้ **R** มีชื่อว่า Rprofile.site ในโฟลเดอร์ย่อย ..\R\R-2.4.0\etc\ ภายในประกอบด้วยข้อมูลดังนี้

```
# Things you might want to change

# options(papersize="a4")
# options(editor="notepad")
# options(pager="internal")

# to prefer Compiled HTML help
  options(chmhelp=TRUE)

# to prefer HTML help
# options(htmlhelp=TRUE)
```

จะเห็นว่าหน้าบรรทัดเกือบทุกบรรทัดมีเครื่องหมาย # อยู่ แสดงว่า **R** จะไม่ทำงานบรรทัดคำสั่งเริ่มต้นเหล่านั้น หากต้องการให้ **R** ทำงานบรรทัดคำสั่งใดเมื่อเริ่มต้น **R** ให้ลบ # ที่หน้าบรรทัดคำสั่งนั้นออก (เฉพาะบรรทัดคำสั่งเท่านั้น สังเกตจากเป็นบรรทัดที่มีฟังก์ชันและเครื่องหมายวงเล็บอยู่) เช่น หากต้องการตั้งค่าขนาดกระดาษให้เป็น A4 ทุกครั้งที่เรียกใช้ **R** ให้ลบเครื่องหมาย # หน้าบรรทัด `options(papersize="a4")` ออก (ค่าปริยายจะเป็น A4 อยู่แล้วโดยอัตโนมัติสำหรับเครื่องคอมพิวเตอร์ที่ตั้งค่าสถานที่และเวลาเป็นประเทศไทย) หรือหากต้องการให้ **R** เรียกใช้ความช่วยเหลือเป็นแบบ html ทุกครั้งที่เรียก (ขณะนี้ค่าปริยายเป็นแบบ chtml อยู่ เนื่องจากไม่มีเครื่องหมาย # หน้าบรรทัด) ก็ลบ # หน้าบรรทัด `options(htmlhelp=TRUE)` ออก แล้วใส่เครื่องหมาย # หน้าบรรทัด `options(chmhelp=TRUE)` แทน นอกจากนี้หากต้องการให้ **R** ทำงานใดทุกครั้งที่เริ่มเรียกใช้ **R** ก็สามารถพิมพ์บรรทัดคำสั่งเพิ่มเติมต่อท้ายได้ เช่นหากต้องการให้

เรียกใช้แพคเกจชื่อ Epi ทุกครั้ง ก็พิมพ์บรรทัดเพิ่มเติมต่อท้าย ก็จะได้ข้อความในแฟ้ม Rprofile.site ใหม่ดังนี้

```
# Things you might want to change

options(papersize="a4")
# options(editor="notepad")
# options(pager="internal")

# to prefer Compiled HTML help
# options(chmhelp=TRUE)

# to prefer HTML help
options(htmlhelp=TRUE)
library(Epi)
```

สำหรับผู้ที่ต้องการตั้งค่าเริ่มต้นไว้หลากหลายแบบให้เลือกใช้ได้เองตามต้องการ สามารถทำได้เช่นกัน แต่พึงสังเกตว่าค่าปริยายของโฟลเดอร์ที่ **R** เรียกใช้จะอยู่ที่ ..\R\R-2.4.0 ไม่ใช่ที่ ..\R\R-2.4.0\etc จึงต้องสร้างไว้ที่ ..\R\R-2.4.0 ซึ่งเรียกว่า home directory ของ **R** นั่นเอง ซึ่ง home directory นี้บนระบบปฏิบัติการลินุกซ์จะอยู่ที่ไดเรกตอรีที่ผู้ใช้เรียกใช้ **R** หรือโดยปกติก็คือ home ของผู้ใช้นั่นเอง เช่นเดียวกับบนระบบปฏิบัติการแมคอินทอช OSX ก็จะเป็นโฟลเดอร์ของผู้ใช้ผู้นั้น

ลองสร้างแฟ้มที่ชื่อ Rprofile1.txt ขึ้นใน home directory ที่กล่าวแล้ว อาจใช้โปรแกรม notepad หรือโปรแกรม text editor ง่ายๆ อื่นใดก็ได้ตามที่มีอยู่ โดยพิมพ์ข้อความต่อไปนี้

```
options(papersize="a4")
options(htmlhelp=TRUE)
cat("Welcome to Rprofile1\n")
```

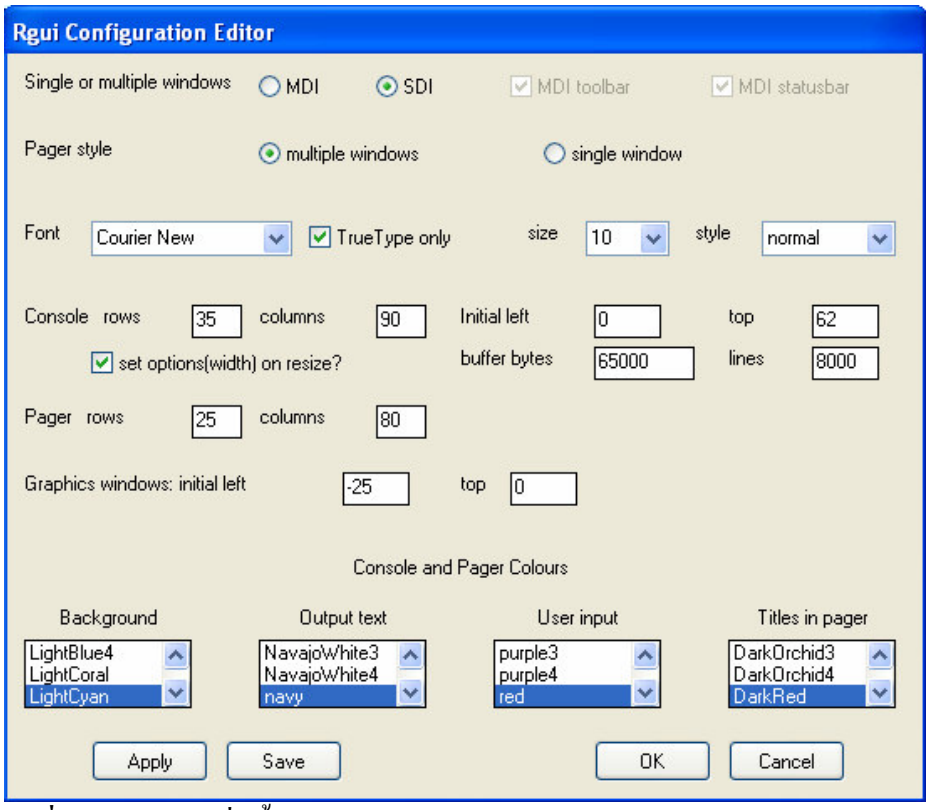
เมื่อเสร็จแล้วให้บันทึกลงแฟ้ม Rprofile1.txt ไว้ จากนั้นที่หน้าจอของ **R** พิมพ์บรรทัดคำสั่ง `source("Rprofile1.txt")` จะพบว่าบนหน้าจอ **R** ปรากฏข้อความดังต่อไปนี้

```
> options(papersize="a4")
> options(htmlhelp=TRUE)
> cat("Welcome to Rprofile1")
Welcome to Rprofile1
>
```

นั่นคือ เราสามารถตั้งค่าต่างๆ ได้โดยการสร้างแฟ้มสคริปต์ และเรียกใช้โดยคำสั่ง `source()` ซึ่งจะทำงานเช่นเดียวกับการตั้งค่าเริ่มต้นโดยใช้แฟ้ม Rprofle.site นั่นเอง และวิธีการนี้สามารถใช้ได้กับระบบปฏิบัติการทุกระบบ

แฟ้มสำหรับการตั้งค่าต่างๆ ของหน้าต่าง **R** มีชื่อ Rconsole โดยอยู่ในโฟลเดอร์ My Document ผู้ใช้แต่ละคนที่ใช้เครื่องเดียวกันจึงสามารถตั้งค่าปริยายของหน้าต่าง **R** ตามใจตัวเองได้ และไม่ไปยุ่งเกี่ยวกับการตั้งค่าของคนอื่น โดยปกติแล้วผู้ใช้ไม่ควรเปิดเข้าไปแก้ไขใดๆ กับข้อมูลในแฟ้มนี้ แต่อาจเปลี่ยนแปลงการตั้งค่าได้โดยการตั้งค่าผ่านรายการเมนู Edit >> GUI

preferences… บนหน้าต่าง R Console นั่นเอง ขอให้ลองตั้งค่าต่างๆ เช่น ขนาดของหน้าต่าง สีตัวอักษรและพื้นหลัง และอื่นๆ ตามชอบใจ อย่างไรก็ตามหน้าจอนี้ไม่มีปุ่ม default หรือ reset ให้ตั้งค่ากลับเป็นค่าปริยายดั้งเดิม หากต้องการทำเช่นนั้น ก็ให้ลบแฟ้ม Rconsole ทิ้ง แล้ว **R** จะสร้างแฟ้มขึ้นใหม่เอง



รูปที่ 1.9 หน้าต่างเพื่อตั้งค่า Rgui บนระบบปฏิบัติการวินโดวส์

ตัวเลือกตัวหนึ่งที่น่าสนใจคือตัวเลือกบรรทัดบนสุด Single or multiple windows สำหรับผู้ที่จะใช้สภาพแวดล้อม **ICE** ต่อไป แนะนำให้ตั้งค่าเป็น SDI นอกจากนี้หากจะใช้งานโปรแกรม **Tinn-R** ร่วมในการทำงานกับ **R** แล้ว โปรแกรม **Tinn-R** รุ่น 1.18.1.5 ขึ้นไป จะทำงานกับตัวเลือก SDI เท่านั้น ซึ่งก็คือการตั้งให้หน้าต่างของ **R** เป็นหน้าต่างเดี่ยว และหากมีการสร้างหน้าต่างกราฟหรือหน้าต่าง tcl/tk อื่นขึ้นมา ผู้ใช้สามารถเคลื่อนหน้าต่างเหล่านั้นไปวางที่ใดก็ได้บน desktop อีกจุดหนึ่งที่แนะนำให้เปลี่ยนค่าเริ่มต้นใหม่ก็คือบนแถวที่ 4 ในช่อง top ด้านขวาสุด เป็นการกำหนดตำแหน่งในแนวแกน Y ของมุมบนซ้ายของหน้าต่าง RGUI ให้กำหนดต่ำลงมาประมาณ 62 pixel ซึ่งจะทำให้มีที่ว่างสำหรับวางหน้าต่างเมนูหลักของ **ICE** ไว้เหนือหน้าต่าง Rgui ได้

บนระบบปฏิบัติการแมคอินทอช OSX นั้น โฟลเดอร์และแฟ้มต่างๆ ของ **R** อยู่ที่ /Library/Frameworks/R.framework/ ซึ่งจะเก็บข้อมูลแยกตามรุ่นของ **R** ที่ติดตั้ง ส่วนแฟ้ม Rprofile

อยู่ที่ {R_HOME}/library/base/R ผู้ใช้ระบบแมคอินทอช OSX อาจตั้งค่าปริยายของหน้าต่าง **R** ได้โดยการเลือก preference ตามวิธีการของ แมคอินทอช โดยไม่นิยมเปลี่ยนค่าต่างๆ ในแฟ้ม Rprofile

ส่วนระบบปฏิบัติการลินุกซ์นั้น แฟ้ม Rprofile อยู่ที่ {R_HOME}/library/base/R แต่การตั้งค่าในแฟ้มนี้ออกจะยุ่งยากพอสมควร ส่วนการตั้งค่าปริยายของหน้าต่าง terminal ก็สามารถทำได้แต่ก็มีความจำกัดอยู่ค่อนข้างมาก ส่วนตัวเลือกค่าเริ่มต้นของสภาพแวดล้อม **R** อาจตั้งและเรียกค่าตัวเลือกโดยคำสั่ง `options()` ด้วยตัวเองดังที่กล่าวไว้แต่ต้นแล้ว และเรียกใช้โดยใช้คำสั่ง `source()` ทุกครั้งที่เรียกใช้ **R** ได้เลย

ผู้ใช้มือใหม่อาจรู้สึกว่าวิธีที่ต้องทำเองเช่นนี้ออกจะยุ่งยาก แต่จริงๆ แล้วก็ไม่ได้ยุ่งยากอะไร เนื่องจากค่าปริยายถูกตั้งไว้อย่างเหมาะสมกับระบบปฏิบัติการนั้นๆ อยู่แล้ว เช่น ความช่วยเหลือบนระบบแมคอินทอช OSX จะอ่านได้สวยงามและอ่านภาษาไทยได้อยู่แล้ว และยังสามารถใช้ link ได้ด้วยหากใช้ค่าปริยายที่ตั้งไว้แต่แรก ส่วนระบบปฏิบัติการลินุกซ์นั้น การใช้ความช่วยเหลือเป็น text อาจไม่สะดวกสำหรับผู้ที่ต้องการใช้เชื่อมโยงต่อไปสู่แฟ้มความช่วยเหลืออื่นๆ ที่เกี่ยวข้องกัน ในกรณีเช่นนั้นผู้ใช้อาจสร้างแฟ้มสคริปต์เริ่มต้นเล็กๆ ขึ้นใช้เพื่อกำหนดให้เรียกใช้ความช่วยเหลือแบบ html ได้เช่นกัน

การสร้างแฟ้มสคริปต์อาจสร้างไว้หลายแฟ้มสำหรับหลายๆ งาน หรือกรณีให้เลือกใช้ได้ตามต้องการ และการสร้างแฟ้มสคริปต์เล็กๆ ไว้ทำงานตามความต้องการนี่เองคือวิธีการที่ **R** แนะนำให้ใช้อยู่แล้วเป็นปกติ การทำงานเช่นนี้จะได้กล่าวอีกครั้งในบทที่ 3 เรื่อง **สภาพแวดล้อม R-ICE** ซึ่งเราจะใช้วิธีนี้ในการกำหนดแพคเกจที่จะเรียกมาใช้งาน

การติดตั้งแพคเกจเพิ่มเติมเพื่อเพิ่มความสามารถให้ R

ดังได้กล่าวข้างต้นแล้วว่าเราสามารถเพิ่มความสามารถให้ **R** ทำงานได้ดีขึ้นอย่างไม่จำกัดโดยการติดตั้งแพคเกจที่ทำงานเฉพาะด้านเพิ่มเติม เช่นหากต้องการสร้างตารางและกราฟบางอย่างทางระบาดวิทยาเพิ่มเติม ก็อาจติดตั้งแพคเกจ Epi หรือ epitools เพิ่มได้ และหากต้องการให้มีรายการเมนูที่เป็นกราฟิกเพิ่มเติม ก็อาจติดตั้งแพคเกจ **ICE** เพิ่มได้

การติดตั้งแพคเกจอาจทำได้สองวิธี วิธีหนึ่งคือติดตั้งโดยตรงจากเว็บไซต์ของ **CRAN** กับการติดตั้งจากเครื่องของผู้ใช้โดยตรง วิธีแรกใช้ในกรณีที่แพคเกจนั้นได้รับการรับรองและบรรจุไว้ในเครือข่ายของ **CRAN** แต่พึงสังเกตว่าในการรับรองนั้น **CRAN** ไม่ได้รับรองว่าวิธีการคำนวณหรือทำงานของแพคเกจนั้นถูกต้อง แต่จะรับรองว่ารูปแบบการใช้งานและข้อความช่วยเหลือต่างๆ เป็นไปตามที่ **CRAN** กำหนด ส่วนการทำงานของแพคเกจนั้นจะถูกต้องตามที่ยอมรับในสาขาวิชานั้นๆ หรือไม่ ผู้ใช้ต้องอ่านและติดตามจากเอกสารอ้างอิงที่แพคเกจนั้นอ้างไว้เอง และจะเป็นการถูกต้องตามหลักวิชาการที่ผู้ใช้จะต้องเขียนอ้างอิงบทความที่มาของวิธีการนั้นๆ เช่นหากผู้ใช้

ต้องการใช้วิธีคำนวณ Firth's bias reduced logistic regression ในแพคเกจ logistf ก็ต้องอ้างอิง Firth D (1993). Bias reduction of maximum likelihood estimates. *Biometrika* 80, 27–38. ซึ่งเป็นผู้ตีพิมพ์วิธีการนี้ในวารสาร Biometrika ด้วย

การติดตั้งแพคเกจจาก **CRAN** นั้นก็ไม่ได้ยุ่งยากแต่อย่างใด แต่อาจใช้เวลาสักเล็กน้อย เนื่องจากเป็นการดาวน์โหลดทางอินเตอร์เน็ต ซึ่งหากทำผ่านโมเด็มก็จะใช้เวลาอยู่บ้าง ทั้งนี้ก็ขึ้นกับขนาดของแพคเกจนั้นเองว่าใหญ่เพียงใด

ในขั้นตอนแรกสุด ผู้ใช้จะต้องเข้าไปที่เว็บไซต์ของ **CRAN** และเลือกตัวเลือก Packages ด้านซ้ายมือ เพื่อหารายการแพคเกจที่ต้องการ ในขั้นแรกแนะนำให้ใช้การค้นหาของโปรแกรม web browser ที่ใช้อยู่นั้น เพราะจะมีรายการแพคเกจให้เลือกมากมายจนยากที่จะค้นหาด้วยตา เช่น หากต้องการหา Firth's bias reduced logistic regression ถ้าเลือกใช้คำหลักเป็น logistic จะพบว่ามีแพคเกจ brlr และ logistf ที่มีวิธีการนี้อยู่เช่นเดียวกัน ซึ่งแพคเกจ brlr นั้นสร้างขึ้นโดย David Firth ส่วนแพคเกจ logistf สร้างโดยคณะบุคคลอื่น ผู้ใช้สามารถเลือกได้เองว่าจะติดตั้งแพคเกจใด หรือจะลองติดตั้งทั้งสองแพคเกจเลยก็ได้ ลองดูตัวอย่างของแพคเกจ brlr ดังนี้

brlr: Bias-reduced logistic regression

Fits logistic regression models by maximum penalized likelihood

Version: 0.8-7
Date: 2006-01-10
Author: David Firth
Maintainer: David Firth
License: GPL Version 2 or later
URL: http://www.warwick.ac.uk/go/dfirth

Downloads:

Package source: brlr_0.8-7.tar.gz
Windows binary: brlr_0.8-7.zip
Index of contents: brlr.INDEX
Reference manual: brlr.pdf

เมื่อเลือกได้แล้ว ให้กลับไปที่ **R** หรือเปิด **R** ขึ้นมาใช้งาน สำหรับระบบวินโดวส์นั้นรายการเมนูด้านบนมีตัวเลือก Packages อยู่แล้ว เมื่อเลือกตัวเลือก Install package(s)… จากรายการนี้ จะมีรายการของ **CRAN** mirror มาให้เลือกว่าจะดาวน์โหลดจาก mirror ที่ตั้งอยู่ในประเทศใดรอบโลก อาจเลือกจากประเทศที่เวลาขณะนั้นเป็นเวลาดึกซึ่งมีคนเข้าใช้น้อย (แต่หากดาวน์โหลดไม่ได้ ก็อาจเกิดเนื่องจากในเวลานั้น mirror นั้นกำลังอัพเดทข้อมูลจากเซิร์ฟเวอร์หลักของ **CRAN**

อยู่ก็ได้ กรณีเช่นนี้ก็เปลี่ยน mirror ไปประเทศอื่น) เมื่อเลือกได้แล้ว กดปุ่ม “OK” จะมีรายการแพคเกจให้เลือกต่อมา สมมติว่าเลือกแพคเกจ logistf แล้วกดปุ่ม “OK” แพคเกจนั้นก็จะถูกดาวน์โหลดและติดตั้งเข้าสู่ **R** โดยใช้เวลาไม่กี่วินาทีเนื่องจากเป็นแพคเกจขนาดเล็ก

แม้ว่าแพคเกจจะถูกติดตั้งเข้าสู่โฟลเดอร์ library ของ **R** แล้ว แต่แพคเกจนั้นยังไม่ถูกเรียกใช้งาน การเรียกเข้ามาในหน่วยความจำเพื่อใช้งานจะต้องทำอีกหนึ่งขั้นตอน คือออกคำสั่ง `library()` แล้วใส่ชื่อของแพคเกจนั้นในวงเล็บ เช่น `library(logistf)` จากนั้นผู้ใช้จึงจะเรียกใช้ฟังก์ชันทุกอย่างที่มีในแพคเกจนั้นได้

การติดตั้งจากเครื่องคอมพิวเตอร์ของผู้ใช้โดยตรงก็ไม่ยุ่งยากแต่อย่างใด ผู้ใช้อาจดาวน์โหลดแพคเกจที่ต้องการจากที่ใดก็ได้ เช่น จาก **CRAN** นั่นเอง โดยต้องเลือกชนิดของแฟ้มให้เหมาะสมกับระบบปฏิบัติการที่ใช้ เช่นหากใช้ระบบวินโดวส์ ก็ต้องดาวน์โหลดแฟ้ม Windows binary ที่มีนามสกุลเป็น zip ส่วนผู้ใช้เครื่องแมคอินทอชและลินุกซ์ ให้เลือกแฟ้มที่มีนามสกุล tar.gz

สำหรับระบบวินโดวส์ ภายใต้รายการเมนู Packages ด้านบน เมื่อเลือกแล้วจะมีตัวเลือก Install package(s) from local zip files… ซึ่งเมื่อเลือกจะมีหน้าต่าง Select files ปรากฏขึ้นให้เลือกแฟ้มแพคเกจที่มีนามสกุล zip เพื่อติดตั้ง เมื่อติดตั้งแล้ว ทุกครั้งที่เปิดใช้งาน **R** และต้องการใช้แพคเกจนั้น ก็ให้ออกคำสั่ง `library()` และใส่ชื่อแพคเกจไว้ในวงเล็บ

สำหรับระบบแมคอินทอช OSX นั้น แม้ว่าจะมีรายการเมนู Packages & Data ที่ด้านบนของ **R** จะมีรายการ แต่พบว่าการติดตั้งผ่านทางตัวเลือกนี้ยังคงมีปัญหา ดังนั้นสำหรับผู้ใช้ระบบแมคอินทอช OSX และระบบลินุกซ์นั้น ให้ดาวน์โหลดแพคเกจที่ต้องการมาไว้ที่ home แล้วติดตั้งจากภายในเครื่องจะเป็นการสะดวกกว่า **การติดตั้งบนระบบลินุกซ์ ผู้ใช้ต้อง login เป็น root เสียก่อนจึงจะติดตั้งได้**

สมมติว่าต้องการจะติดตั้งแพคเกจ logistf จาก **CRAN** ผู้ใช้สามารถดาวน์โหลดจากเว็บไซต์ของ **CRAN** ดังกล่าวข้างต้น แต่จะต้องเลือกแฟ้มนามสกุล tar.gz ไม่ใช่ zip และดาวน์โหลดมาไว้ที่ home ของผู้ใช้ (สำหรับผู้ใช้แมคอินทอชอาจพบว่าแฟ้มที่ดาวน์โหลดมาได้ไปอยู่ที่ desktop ก็ให้ย้ายไปที่ home ก่อน) จากนั้นเปิดหน้าต่าง terminal แล้วออกคำสั่ง tar เพื่อขยายแฟ้มออกมา แล้วจึงติดตั้งด้วยคำสั่ง R CMD INSTALL (พิมพ์ตัวใหญ่ทั้งหมด พึงระวังว่าใน terminal ของระบบยูนิกซ์นั้น ตัวอักษรใหญ่และเล็กมีความแตกต่างกัน) ดังตัวอย่างการติดตั้งแพคเกจ logistf_1.05.tar.gz ข้างล่างนี้

```
$ tar xvfz logistf_1.05.tar.gz
$ R CMD INSTALL logistf
```

ตัวเลือก xvfz ของคำสั่ง tar นั้นเป็นการบอกให้ทำงานดังนี้ x คือการกู้หรือขยายข้อมูล v คือให้รายละเอียดของการ tar (ถ้าไม่ใส่ตัวเลือกนี้ก็ยังคงทำงานได้ แต่ tar จะไม่แสดงอะไรบนหน้าจอ

ทำให้บางครั้งไม่แน่ใจว่าเครื่องได้ทำงานแล้วหรือยัง) f เป็นการบอกให้แสดงรายการข้อมูล และ z เป็นการบอกว่าแฟ้มมีการบีบอัดข้อมูลแบบ gzip อยู่ ส่วนการออกคำสั่งเพื่อติดตั้งแพคเกจนั้น ให้พิมพ์ R CMD INSTALL ตัวใหญ่ทั้งหมดและพิมพ์แยกกันดังที่แสดง แล้วตามด้วยชื่อของแพคเกจนั้น โดยพิมพ์ตัวใหญ่เล็กตามชื่อของแพคเกจนั้น (บางแพคเกจมีอักษรตัวใหญ่ด้วย เช่น Epi) และไม่ต้องมีเลขรุ่นต่อท้าย

หมายเหตุ จำไว้ว่าการติดตั้งแพคเกจกับการเรียกใช้งานแพคเกจเป็นคนละเรื่องกัน การติดตั้งแพคเกจเป็นการติดตั้งเข้าสู่ระบบแฟ้มของ **R** แต่ยังไม่ได้เรียกใช้งานในหน่วยความจำ เราอาจติดตั้งแพคเกจต่างๆ หลายสิบหรือถึงร้อยแพคเกจในเครื่องก็ได้ แต่ยังไม่เรียกใช้งาน เมื่อใดที่ต้องการเรียกใช้งาน ก็เพียงออกคำสั่ง `library()` และใส่ชื่อแพคเกจไว้ในวงเล็บ

บทที่ 2 สภาพแวดล้อมเสริมการทำงาน

โปรแกรม text editor สำหรับเขียนสคริปต์ภาษา **R**

โปรแกรม **Crimson Editor** บนระบบปฏิบัติการวินโดวส์

โปรแกรม **jEdit** บนระบบปฏิบัติการลินุกซ์และแมคอินทอช OSX

โปรแกรมสำหรับสร้างแฟ้มข้อมูล

โปรแกรม **EpiData** บนระบบปฏิบัติการวินโดวส์

โปรแกรม **Wine** สำหรับเล่นโปรแกรม **EpiData** บนระบบปฏิบัติการลินุกซ์

ในบทนี้จะกล่าวถึงโปรแกรมที่แนะนำให้ใช้ร่วมกับสภาพแวดล้อม **R** บนระบบปฏิบัติการต่างๆ แต่พอสังเขป โดยจะกล่าวถึงการติดตั้งและเริ่มใช้งานเบื้องต้นเท่านั้น และจะไม่กล่าวถึงการใช้โปรแกรมโดยละเอียด ผู้ใช้ที่ต้องการใช้โปรแกรมเหล่านั้นให้เต็มความสามารถ ควรศึกษาจากคู่มือของโปรแกรมนั้นๆ เอง

เนื่องจากหนังสือเล่มนี้มีวัตถุประสงค์โดยอ้อมอย่างหนึ่ง คือสนับสนุนการใช้โปรแกรมที่ถูกกฎหมาย โดยไม่ละเมิดลิขสิทธิ์ทางการค้าของซอฟต์แวร์ใด ในที่นี้จึงแนะนำโปรแกรมในกลุ่ม open source ที่มีลิขสิทธิ์แบบ GNU general public license (GPL) หรือโปรแกรมที่แจกจ่ายได้ฟรี ถึงแม้จะไม่เป็น open source ก็ตามเป็นหลัก แต่หากผู้ใช้มีโปรแกรมอื่นที่ใช้งานได้ในลักษณะเดียวกันกับโปรแกรมที่จะกล่าวถึงต่อไปนี้ ที่ซื้ออย่างถูกต้องตามกฎหมายอยู่แล้ว ก็สามารถใช้โปรแกรมนั้นๆ ได้ตามชอบใจ

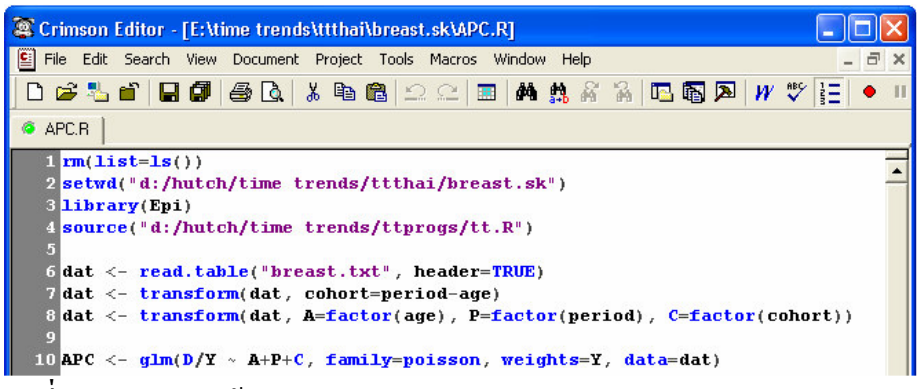
โปรแกรมที่อาจต้องใช้ร่วมกับสภาพแวดล้อม **R** มีสองกลุ่มคือ โปรแกรม text editor หรือโปรแกรมที่ใช้สำหรับเขียนสคริปต์ภาษา **R** นั่นเอง ในที่นี้จะแนะนำเฉพาะโปรแกรมที่รู้จักคำหลักของภาษา **R** ติดตั้งอยู่ในตัว ซึ่งในที่นี้โปรแกรมที่แนะนำคือ **Crimson Editor** สำหรับระบบปฏิบัติการวินโดวส์ และโปรแกรม **jEdit** สำหรับระบบปฏิบัติการลินุกซ์และแมคอินทอช OSX โปรแกรมอื่นๆ ที่มีสามารถทำให้รู้จักคำสงวนและชื่อฟังก์ชันของ **R** มีอยู่ในหน้าเว็บ http://www.sciviews.org/_rgui/projects/Editors.html โปรแกรมอีกกลุ่มหนึ่งคือโปรแกรมสำหรับป้อนข้อมูลหรือสร้างแฟ้มข้อมูลนั่นเอง ในที่นี้แนะนำให้ใช้โปรแกรมชื่อ **EpiData** ซึ่งขณะนี้ใช้ได้กับระบบปฏิบัติการวินโดวส์เท่านั้น แต่อาจใช้กับระบบปฏิบัติการลินุกซ์ได้โดยการเล่นผ่านโปรแกรม **Wine** และนอกจากนี้อาจใช้โปรแกรมฐานข้อมูลหรือโปรแกรมกระดานคำนวณอื่นๆ ทำงานแทนก็ได้ ซึ่งจะไม่กล่าวในบทนี้

โปรแกรม text editor สำหรับเขียนสคริปต์ภาษา R

โปรแกรม Crimson Editor บนระบบปฏิบัติการวินโดวส์

โปรแกรม **Crimson Editor** เป็นโปรแกรมที่มีลิขสิทธิ์แบบ freeware คือสามารถใช้ได้โดยไม่ต้องซื้อ โดยอาจดาวน์โหลดโปรแกรมจากเว็บไซต์ http://www.crimsoneditor.com หรือจากแผ่นซีดีที่แถมพร้อมหนังสือเล่มนี้ก็ได้ รุ่นปัจจุบันได้แก่รุ่น 3.7 ซึ่งไม่ได้แก้ไขเพิ่มเติมตั้งแต่ปี 2004 ซึ่งแสดงว่าไม่มี bug สำคัญที่เป็นปัญหาในการใช้งานแต่อย่างใด โปรแกรมนี้เป็นโปรแกรม text editor ทั่วไป ไม่จำเพาะสำหรับภาษา **R** แต่ได้มีแฟ้ม syntax ภาษา **R** ไว้ให้แล้ว ซึ่งจะต้องเรียกมาใช้งานต่อไปหลังจากติดตั้ง โปแกรม **Crimson Editor** สามารถทำสีสำหรับคำสั่งหรือฟังก์ชันในภาษา **R** ให้แตกต่างจากข้อความอื่นๆ และสามารถตรวจได้ว่าปิดวงเล็บต่างๆ ได้ครบถ้วน

การติดตั้งโปรแกรม **Crimson Editor** นั้นทำได้ง่าย โปรแกรมติดตั้งชื่อ cedt370r.exe สามารถคลิกเพื่อติดตั้งได้เลย เมื่อได้อ่านลิขสิทธิ์แล้ว กดปุ่ม “I Agree” แล้วกดปุ่ม “Next >” ในหน้าจอถัดไป เลือกโฟลเดอร์ที่จะติดตั้ง แล้วกดปุ่ม “Install” ก็จะติดตั้งโปรแกรมเสร็จเรียบร้อยในเวลาอันสั้นเนื่องจากโปรแกรมมีขนาดเล็ก ลองดูตัวอย่างหน้าต่างโปรแกรม **Crimson Editor** ดังรูปที่ 2.1



รูปที่ 2.1 ตัวอย่างหน้าต่างโปรแกรม **Crimson Editor**

จะเห็นว่าชื่อฟังก์ชันและคำสงวนของ **R** เช่น TRUE และ FALSE จะถูกทำเป็นสีน้ำเงิน ข้อความที่อยู่ภายในเครื่องหมายฟันหนู “ ” จะถูกทำเป็นสีม่วงแดง และคำอื่นๆ ที่โปรแกรมไม่รู้จัก เช่น ชื่อตัวแปร ชื่อกรอบข้อมูล รวมทั้งสัญลักษณ์ และเครื่องหมายต่างๆ จะแสดงเป็นสีดำ สีที่แตกต่างกันเหล่านี้จะปรากฏเมื่อบันทึกแฟ้มแล้วเท่านั้น และควรบันทึกเป็นนามสกุล R เพื่อแสดงให้รู้ว่าเป็นแฟ้มสคริปต์คำสั่งของ **R** และหากต้องการแสดงเลขบรรทัดก็สามารถทำได้โดยเลือกรายการเมนู View >> Line numbers หรือปุ่ม Ctrl+Shift+L

เมื่อติดตั้งโปรแกรมเรียบร้อยแล้ว เราจำเป็นต้องทำให้โปรแกรมรู้จักฟังก์ชันและคำสงวนของ **R** เสียก่อน โดยเลือกรายการเมนู Tools >> Preferences… แล้วเลือก Syntax type ในรายการด้านซ้ายมือ ส่วนในช่องด้านขวามือจะมีรายการภาษาต่างๆ ที่ติดตั้งไว้ก่อนแล้ว ซึ่งยังไม่มีภาษา **R** อยู่ในรายการ ให้เลื่อนรายการเหล่านี้ลงไปข้างล่างแล้วคลิกที่ –Empty– อันแรกสุดที่พบ เมื่อถึงขั้นนี้ 3 ช่องด้านล่างจะว่างอยู่ ให้พิมพ์ R (ตัวใหญ่) ในช่อง Description: ในช่อง Lang Spec: บรรทัดถัดไปให้กดปุ่ม “…” ด้านขวามือสุด จะมีรายการของแฟ้ม spec ให้เลือกใช้ ตรงนี้เลือก r.spc ช่องล่างสุดเป็น Keywords: ให้กดปุ่ม “…” เช่นกัน แล้วเลือก r.key จากนั้นกดปุ่ม “Apply” และ “OK” ด้านล่างสุดเป็นอันเสร็จ แต่อย่างไรก็ตาม เมื่อสร้างหรือเปิดแฟ้มขึ้นใหม่ **Crimson Editor** จะไม่รู้จักว่าเป็นภาษา **R** และทำสีไม่ถูกต้องจนกว่าเราจะได้เลือกรายการเมนู Document >> Syntax Type และเลือก R ในรายการที่ปรากฏเสียก่อน ซึ่งขณะนี้จะอยู่ล่างสุดนั่นเอง เราอาจลบภาษาอื่นๆ ที่ไม่

ต้องการออก หรือเลื่อน R ขึ้นไปอยู่บนสุดก็ได้ โดยเลือกรายการ customize... ที่ด้านล่างสุดของรายการภาษาคอมพิวเตอร์ต่างๆ ก็ได้

ข้อเด่นของโปรแกรม **Crimson Editor** คือเมื่อแฟ้มที่เปิดอยู่มีการเปลี่ยนแปลงไป เช่นในกรณีที่มีการแก้ไขแฟ้มสคริปต์หรือแฟ้มผลลัพธ์โดยสภาพแวดล้อม **R-ICE** โปรแกรมก็จะทราบและถามได้ทันทีและถามว่าจะดูและเรียกใช้สิ่งที่เปลี่ยนแปลงนั้นหรือไม่ ซึ่งมีประโยชน์มากในการทำงานในสภาพแวดล้อม **R-ICE** ซึ่งจะได้กล่าวต่อไป

หมายเหตุ ผู้เขียนได้ใช้ **Tinn-R** บนวินโดวส์ XP อีกโปรแกรมหนึ่งเช่นกัน แต่พบว่ามีปัญหาในการ Open และ Save จากรายการเมนู File โดยเมื่อหน้าต่าง Open หรือ Save เปิดขึ้นมาแล้ว ถ้ามีการเปลี่ยน directory เพื่อจะเข้าไปในโฟลเดอร์อื่น จะไม่สามารถทำงานต่อได้ โดยขึ้นข้อความว่า Not Responding ผู้เขียนยังไม่ได้พยายามหาสาเหตุของข้อผิดพลาดของโปรแกรมเช่นนี้

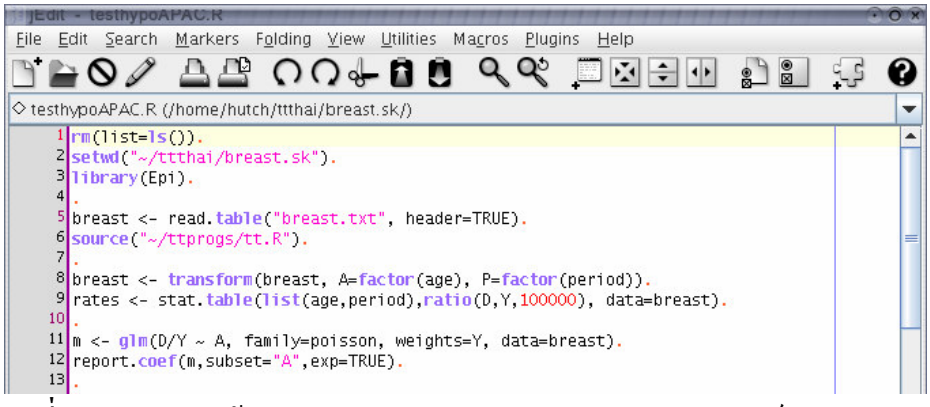
วิธีหลีกเลี่ยงปัญหานี้สามารถทำได้โดยหากจะเปิดแฟ้มใด ให้ใช้วิธีเข้าไปที่โฟลเดอร์นั้น แล้ว double click เพื่อเปิดแฟ้มนั้น หลีกเลี่ยงการใช้รายการเมนู File และหากต้องการบันทึกข้อมูล ก็ให้บันทึกลงโฟลเดอร์ที่ปรากฏในหน้าต่างโดยตรง แล้วจึงค่อยย้ายแฟ้มนั้นไปโฟลเดอร์อื่นโดยทำงานกับวินโดวส์โดยตรง เพื่อหลีกเลี่ยงการใช้รายการเมนู File เช่นกัน

โปรแกรม jEdit บนระบบปฏิบัติการลินุกซ์และแมคอินทอช OSX

โดยปกติแล้ว ผู้ใช้ระบบปฏิบัติการแมคอินทอช OSX ไม่จำเป็นต้องติดตั้งโปรแกรม text editor อื่นเพิ่มเติมแต่อย่างใด เนื่องจากโปรแกรม **R** บนแมค OSX มี editor เฉพาะสำหรับภาษา **R** อยู่ในตัวอยู่แล้ว แต่หากต้องการติดตั้งโปรแกรมเพิ่มเติมก็ย่อมทำได้ โปรแกรม **jEdit** เป็นโปรแกรม text editor ที่เขียนโดยภาษา Java ทำให้สามารถเล่นบนระบบปฏิบัติการใดๆ ก็ได้ที่ติดตั้ง Java engine ซึ่งหมายความว่าสามารถเล่นได้บนระบบปฏิบัติการวินโดวส์ด้วยเช่นกัน แต่เนื่องจากบนระบบปฏิบัติการวินโดวส์นั้น โปรแกรม **Crimson Editor** และ **Tinn-R** มีความสามารถที่จำเพาะกับโปรแกรม **R** ได้ดีกว่า จึงไม่จำเป็นต้องใช้โปรแกรม **jEdit** แต่อย่างใด และโดยปกติแล้วระบบปฏิบัติการวินโดวส์บางรุ่นไม่ได้ติดตั้ง Java engine ไว้ตั้งแต่แรก การติดตั้งจึงค่อนข้างจะยุ่งยากและเสียเวลา

การติดตั้งโปรแกรม **jEdit** ทำได้โดยดาวน์โหลดจากเว็บไซต์ http://www.jedit.org หรือจากแผ่นซีดีที่แถมมากับหนังสือนี้ เมื่อติดตั้งแล้ว ในการเรียกใช้นั้น สำหรับระบบปฏิบัติการลินุกซ์ใช้วิธีเปิดหน้าต่าง terminal แล้วพิมพ์ jedit โปรแกรม **jEdit** ก็จะถูกเรียกมาทำงาน เมื่อโปรแกรม

ทำงานแล้ว ผู้ใช้อาจย่อซ่อนหน้าต่าง terminal ไปได้ แต่อย่าปิดหน้าต่าง terminal มิฉะนั้นโปรแกรม **jEdit** จะถูกปิดไปด้วย ส่วนบนระบบปฏิบัติการแมคอินทอช OSX นั้น เมื่อติดตั้งแล้วจะมีไอคอน jEdit ก็สามารถเรียกโปรแกรมได้จากไอคอนนี้



รูปที่ 2.2 ตัวอย่างหน้าต่างโปรแกรม **jEdit** บนระบบปฏิบัติการลินุกซ์

เนื่องจากโปรแกรม **jEdit** ไม่ได้สร้างมาจำเพาะสำหรับภาษา **R** จึงจำเป็นต้องทำให้ **jEdit** รู้จักคำสงวนและสัญลักษณ์ต่างๆ ของ **R** เสียก่อน โดยดาวน์โหลดแฟ้ม r.xml ได้ที่ http://community.jedit.org โดยเลือกดาวน์โหลด R edit mode – extensive version ด้านล่างขวาของจอภาพ ในส่วนของ top 10 download

สำหรับระบบปฏิบัติการลินุกซ์ ให้นำแฟ้มนี้ไปใส่ในโฟลเดอร์ jedit/4.x/modes ที่ home ของผู้ใช้ สำหรับผู้ใช้ระบบปฏิบัติการแมคอินทอช OSX ให้เข้าไปในโฟลเดอร์ jEdit 4.x แล้วเปิดแฟ้มชื่อ catalog ในโฟลเดอร์นั้น และเปลี่ยนข้อความในบรรทัดต่อไปนี้

```
<MODE NAME="rebol" FILE="rebol.xml" FILE_NAME_GLOB=".r" />
```

เป็น

```
<MODE NAME="r" FILE="r.xml" FILE_NAME_GLOB=".r" />
```

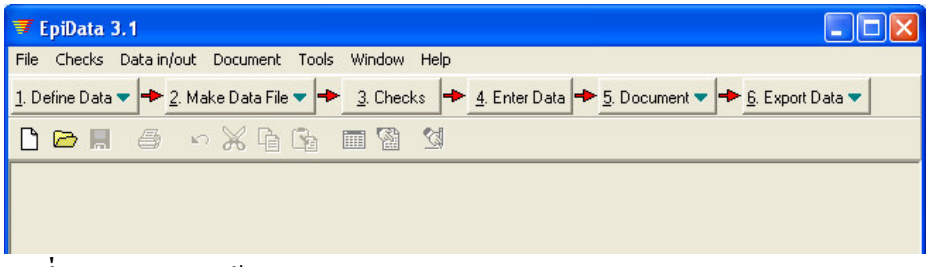
นอกจากนี้ยังอาจลบบรรทัดอื่นๆ ที่อ้างอิงถึงภาษาคอมพิวเตอร์ภาษาอื่นๆ ที่ไม่รู้จักหรือไม่ต้องการใช้ออกเสียก็ได้

โปรแกรมสำหรับสร้างแฟ้มข้อมูล

โปรแกรม EpiData บนระบบปฏิบัติการวินโดวส์

โปรแกรม **EpiData** เป็นโปรแกรมฟรีเช่นกัน แต่ไม่ได้มีลิขสิทธิ์แบบ GPL ใช้ทำงานในการสร้างแบบฟอร์มกรอกข้อมูลลงในคอมพิวเตอร์ ตรวจสอบความถูกต้องของข้อมูล กรอกข้อมูล

ซ้ำและตรวจเช็คความผิดพลาดในการกรอกข้อมูลที่ไม่ตรงกันของทั้งสองแฟ้มข้อมูลได้ด้วย โดยปกติแล้วโปรแกรม **EpiData** เก็บข้อมูลเป็นแฟ้มชนิด **EpiInfo** ซึ่งมีนามสกุล REC แต่ยังสามารถถ่ายข้อมูลออกเป็นอีกหลายรูปแบบ ซึ่งรวมทั้งแฟ้ม text แฟ้ม DBF และที่ดีที่สุดคือถ่ายข้อมูลออกเป็นแฟ้ม DTA ของโปรแกรม **Stata** ได้อีกด้วย ซึ่งแฟ้มข้อมูลชนิดนี้จะถ่ายข้อมูลชนิดแฟกเตอร์ได้ และยังถ่าย label ของข้อมูลไปใช้งานในโปรแกรม **R** ได้อีกด้วย สามารถดาวน์โหลดโปรแกรม **EpiData** ได้ฟรีที่ http://www.epidata.dk/ รุ่นล่าสุดคือรุ่น 3.1 โปรแกรมติดตั้งคือ setup_epidata.exe หน้าตาของโปรแกรม **EpiData** มีลักษณะดังรูปที่ 2.3 ส่วนวิธีการใช้งานจะไม่กล่าวในที่นี้ ในเว็บไซต์จะมีเอกสารคู่มือการใช้งานให้ดาวน์โหลดได้ด้วยเช่นกัน



รูปที่ 2.3 ตัวอย่างหน้าต่างโปรแกรม **EpiData**

โปรแกรม Wine สำหรับเล่นโปรแกรม EpiData บนระบบปฏิบัติการลินุกซ์

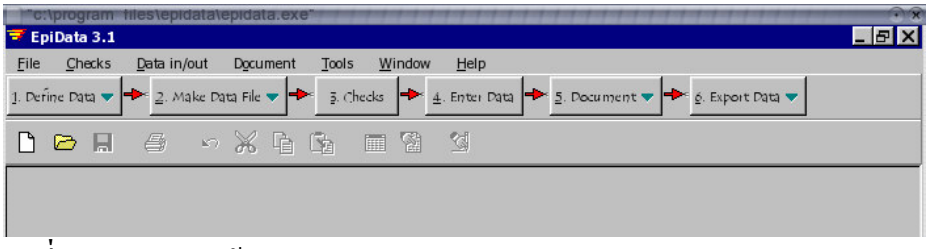
บนระบบปฏิบัติการลินุกซ์และแมคอินทอช OSX นั้น ไม่สามารถใช้โปรแกรม **EpiData** ได้ แต่บนระบบปฏิบัติการลินุกซ์มีโปรแกรมตัวหนึ่งชื่อ **Wine** ซึ่งมีลิขสิทธิ์แบบ GNU lesser general public license สามารถดาวน์โหลดฟรีได้ที่ http://www.winehq.com/

เมื่อติดตั้งแล้ว จำเป็นต้องจัดการติดตั้งฟอนต์ของระบบวินโดวส์ลงใน **Wine** ด้วย มิฉะนั้นอักษรต่างๆ ของโปรแกรมที่เล่นผ่าน **Wine** อาจมีฟอนต์ที่ผิดเพี้ยนได้ การติดตั้งฟอนต์ของวินโดวส์ทำได้โดยเปิดหน้าต่าง terminal ที่ home จะไม่เห็นโฟลเดอร์ชื่อ .wine ซึ่งเป็นโฟลเดอร์ที่เก็บแฟ้มต่างๆ ของระบบวินโดวส์ เพื่อที่จะเห็นโฟลเดอร์นี้ ให้เลือกรายการ View >> Show Hidden Files หรือหากเป็นภาษาไทยอยู่ จะเป็นรายการ มุมมอง >> แสดงแฟ้มที่ถูกซ่อน เข้าไปที่โฟลเดอร์ .wine/drive_c/windows/fonts แล้วคัดลอกฟอนต์ของวินโดวส์ใส่ในโฟลเดอร์นี้ นอกจากนี้อาจเป็นไปได้ที่ฟอนต์ต่างๆ อาจยังแสดงไม่ถูกต้องนัก อีกทางหนึ่งที่จะช่วยได้คือขณะใช้ **Wine** ให้เปลี่ยนภาษาของระบบลินุกซ์เป็นภาษาอังกฤษ แล้วล็อกอินเข้าใหม่ อาจช่วยให้การแสดงตัวอักษรดีขึ้นได้

การติดตั้งโปรแกรมลงใน **Wine** ค่อนข้างจะมีรายละเอียดมาก จึงจะไม่กล่าวในที่นี้ ขอให้ศึกษาในคู่มือของ **Wine** โดยละเอียด การติดตั้ง **EpiData** ใช้ตัวติดตั้ง setup_epidata.exe นั่นเอง และสามารถทำได้โดยผู้ใช้ ไม่จำเป็นต้องล็อกอินเป็น root แต่อย่างใด เนื่องจาก .wine จะต้องอยู่ใน

home ของผู้ใช้แต่ละคน เมื่อติดตั้งถูกต้องจะเสมือนการติดตั้งบนระบบวินโดวส์ทุกประการ เมื่อติดตั้งเสร็จ โปรแกรมจะติดตั้งลงที่ ./wine/drive_c/Program Files ผู้ใช้สามารถเรียกใช้งานโปรแกรม **Epidata** โดยออกคำสั่งที่หน้าต่าง terminal ดังต่อไปนี้ ซึ่งจะได้หน้าต่าง **EpiData** ดังในรูปที่ 2.4

```
$ wine "c:\program files\epidata\epidata.exe"
```



รูปที่ 2.4 ตัวอย่างหน้าต่างโปรแกรม **EpiData** เล่นผ่านโปรแกรม **Wine** บนระบบปฏิบัติการลินุกซ์

บริษัท CodeWeavers ได้สร้างโปรแกรม **CrossOver** ขึ้นมาโดยอาศัยพื้นฐานของโครงการ **Wine** แต่ทำให้ใช้งานได้ง่ายขึ้น ผู้สนใจอาจหาซื้อได้โดยการดาวน์โหลดทางอินเตอร์เน็ตที่ http://www.codeweavers.com ซึ่งโปรแกรมนี้มีทั้งบนระบบลินุกซ์และแมคอินทอช OSX

นอกจากนี้โปรแกรมฐานข้อมูลอื่นๆ ก็สามารถนำมาใช้งานในการกรอกข้อมูลได้ เช่น โปรแกรม **mySQL** โปรแกรม **Access** หรือแม้แต่โปรแกรมกระดานคำนวณเช่น **Excel** หรือ **OpenOffice** ก็ใช้ได้เช่นกัน เมื่อกรอกข้อมูลแล้ว ให้ถ่ายข้อมูลออกเป็นแบบ tab delimited หรือ comma separated values (CSV) ก็จะนำเข้าคำนวณในสภาพแวดล้อม **R** ได้

บทที่ 3 สภาพแวดล้อม R-ICE

สภาพแวดล้อม **R-ICE** มีประวัติความเป็นมาอย่างไร

สภาพแวดล้อม **R-ICE** คืออะไร

สภาพแวดล้อม **R-ICE** มีข้อจำกัดอะไรบ้าง

สภาพแวดล้อม **R-ICE** ประกอบด้วยอะไรบ้าง

จะติดตั้งสภาพแวดล้อม **R-ICE** ได้อย่างไร

*สภาพแวดล้อม **R-ICE** มีประวัติความเป็นมาอย่างไร*

เนื่องจากสภาพแวดล้อม **R** แท้ๆ นั้น การทำงานจัดการข้อมูลค่อนข้างยุ่งยากอยู่บ้าง เมื่อทางหน่วยระบาดวิทยา คณะแพทยศาสตร์ มหาวิทยาลัยสงขลานครินทร์ ได้นำ **R** มาใช้ในการสอนนักศึกษาหลังปริญญา และได้รับทุนจากสำนักงานกองทุนสนับสนุนการวิจัย (สกว.) ตั้งแต่ปี พศ. 2546 เพื่อเผยแพร่การใช้งานโปรแกรม **R** ในการวิเคราะห์ข้อมูลทางสถิติ จึงได้สร้างฟังก์ชันเพิ่มเติมขึ้นอีกหลายฟังก์ชันขึ้น โดยเฉพาะเกี่ยวกับการจัดการข้อมูลเพื่อให้สะดวกในการใช้งาน ซึ่งในระยะแรกได้สร้างเป็นฟังก์ชันเพื่อเขียนสคริปต์คำสั่งตามปกติ แต่ต่อมาเมื่อมีความรู้ความชำนาญในการเขียนฟังก์ชันมากขึ้น จึงได้เริ่มเรียนรู้แพคเกจ tcltk โดยเรียนรู้จากโครงการ R graphic user interface (RGUI) ต่างๆ นั่นเอง

เมื่อศึกษาแนวคิดในการสร้าง GUI ของโครงการ RGUI ต่างๆ พบว่าส่วนใหญ่สร้างเป็นโปรแกรมหรือสภาพแวดล้อมชิ้นเดียว ซึ่งหากจะปรับนำมาใช้ก็จะต้องเข้าไปเขียนแก้ไขรหัสคำสั่งหลายส่วน ซึ่งมักจะเกี่ยวเนื่องพัวพันกัน ทำให้แก้ไขได้ยาก จึงได้พัฒนาโครงการ RGUI ขึ้นมาใหม่ โดยเริ่มจากแนวคิดที่ให้แต่ละชิ้นงานทำงานแยกส่วนกัน แล้วนำมาประกอบเป็นระบบเดียวกัน โดยมีรูปแบบการทำงานเหมือนกัน ต้องใช้แพคเกจที่มีอยู่แล้วเป็นมาตรฐานใน **R** ซึ่งจะทำให้ทำงานได้บนระบบปฏิบัติการทุกระบบ และสามารถเปิดให้นักพัฒนาอื่นๆ เข้าร่วมได้

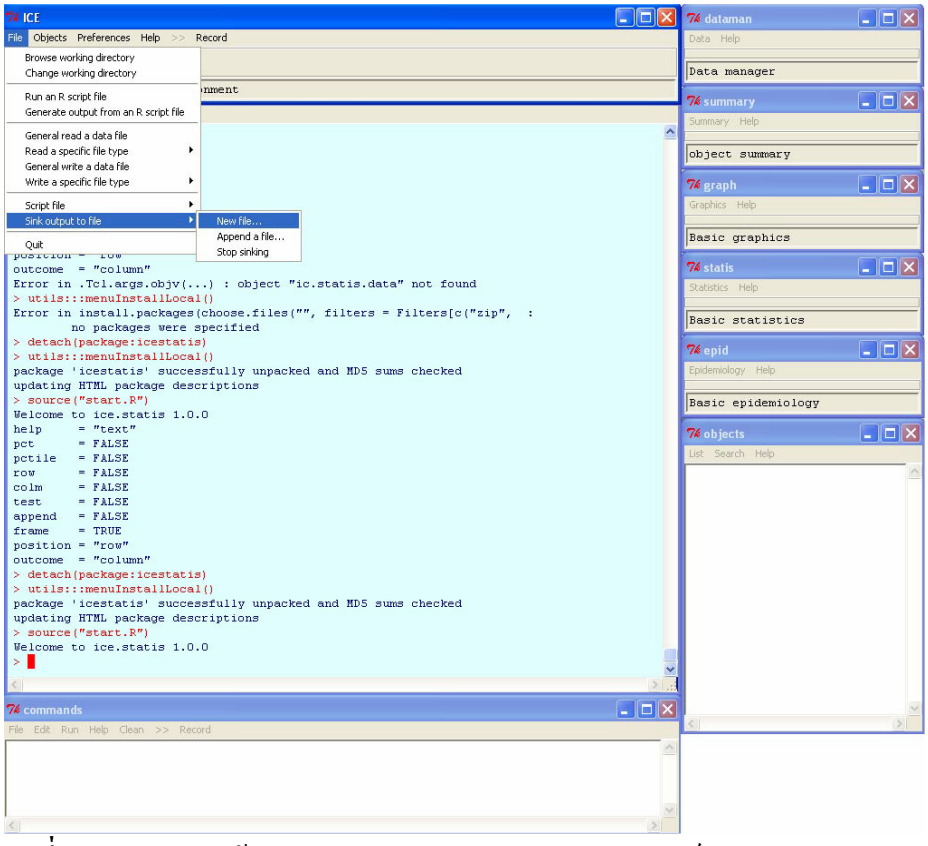
ด้วยแนวคิดดังกล่าว จึงไม่เลือกที่จะใช้ภาษา **C** หรือจาวาในการพัฒนา เนื่องจากหากใช้ภาษาทั้งสอง จะต้องสร้างส่วน GUI อยู่นอกสภาพแวดล้อม **R** แล้วมาเรียกใช้ **R** อีกทีหนึ่ง แพคเกจที่เป็นกราฟิกอยู่ภายใน **R** มีหลายแพคเกจ เช่น tcltk, RGtk และ R-wxPython ซึ่งผู้สนใจอาจหารายละเอียดของแพคเกจที่กล่าวถึงเหล่านั้นได้ที่ http://www.sciviews.org/_rgui/ แต่เนื่องจากมีแพคเกจ tcltk เพียงแพคเกจเดียวที่เป็นแพคเกจมาตรฐาน ซึ่งติดตั้งมากับโปรแกรม **R** และทำงานกับระบบปฏิบัติการทุกระบบได้อย่างไม่มีปัญหา แม้ว่าจะมีข้อจำกัดอยู่ไม่น้อย จึงได้เลือกที่จะพัฒนา RGUI โดยใช้แพคเกจ tcltk แต่ใช้หลักการที่แตกต่างจากโครงการอื่นๆ ที่มีมาก่อนหน้านี้ ดังจะได้กล่าวในรายละเอียดต่อไป

*สภาพแวดล้อม **R-ICE** คืออะไร*

แท้จริงแล้ว มีผู้พัฒนาตัวติดต่อกับผู้ใช้ทางกราฟิก หรือ graphic user interface อยู่อีกหลายโครงการ เช่น โครงการ JGR (อ่านว่า จากัวร์) ที่ใช้ภาษาจาวา โครงการ RCmdr (อ่านว่า อาร์ คอมมานเดอร์) ซึ่งใช้แพคเกจ tcltk เช่นกัน แต่เป็นสภาพแวดล้อมปิดและรวมศูนย์การทำงานไว้ที่โปรแกรมเพียงตัวเดียว โครงการ PMG (Poor Man's GUI) ซึ่งใช้แพคเกจ RGtk และ gtkDevice และโครงการ RKward (อ่านว่า อาร์ค-วาด) ซึ่งทั้งสองโครงการหลังเล่นกับระบบปฏิบัติการตระกูล

ยูนิกซ์เท่านั้น โครงการเหล่านี้ได้พัฒนามาก่อน ค่อนข้างเสถียร และมีการปรับปรุงแก้ไขจุดอ่อนไปพอสมควรแล้ว และน่าใช้ทั้งสิ้น ผู้สนใจอาจหารายละเอียดของแต่ละโครงการได้ที่เว็บไซต์ http://www.sciviews.org/_rgui/ แต่อย่างไรก็ตาม แนวคิดของ **R-ICE** ที่แยกส่วนประกอบต่างๆ เป็นโมดุลย่อยๆ ทำงานร่วมกันนั้น ยังไม่เคยมีการนำมาใช้กับโครงการ RGUI ใดมาก่อน

สภาพแวดล้อม **R-ICE** ที่พัฒนาขึ้นนี้ เป็นระบบหน้าต่างและรายการเมนูกราฟิกที่ทำงานจากภายในสภาพแวดล้อม **R** จึงอาจเรียกรวมกันเป็นสภาพแวดล้อม **R-ICE** ก็ได้ เมื่อเรียกใช้งานจะมีหน้าตาดังรูปที่ 3.1



รูปที่ 3.1 สภาพแวดล้อม **R-ICE** บนระบบปฏิบัติการวินโดวส์

สภาพแวดล้อม **R-ICE** ถูกสร้างมาจากแพคเกจ tcltk ซึ่งเป็นแพคเกจมาตรฐานของ **R** ที่ติดตั้งมากับ **R** บนทุกระบบปฏิบัติการ สภาพแวดล้อม **R-ICE** จึงสามารถทำงานได้บนทุกระบบปฏิบัติการโดยมีหน้าตาคล้ายกัน แต่อาจแตกต่างกันไปบ้างเล็กน้อยตามระบบปฏิบัติการที่ทำงานอยู่ ส่วนการทำงานนั้นยังคงเหมือนกันทุกประการ นอกจากว่าผู้ใช้ระบบแมค OSX จะต้องเปิดใช้ X11 server ด้วย โดยเรียกจากรายการเมนู Misc >> Run X11 Server ซึ่งอยู่ด้านล่างสุดของรายการ

สภาพแวดล้อม **R-ICE** ประกอบด้วยหน้าต่าง R console หรือหน้าต่าง terminal บนระบบปฏิบัติการลินุกซ์ ร่วมกับหน้าต่าง tk ที่สร้างขึ้นจากการทำงานของแพคเกจย่อยหลายหน้าต่างในสภาพแวดล้อม **R-ICE** ซึ่งผู้ใช้จะนำหน้าต่างเหล่านี้ ไปวางไว้ส่วนใดบนหน้าจอคอมพิวเตอร์ก็ได้ตามต้องการ

สภาพแวดล้อม **R-ICE** มีหลักการพื้นฐานดังนี้

1. ทำงานแบบแยกส่วน ที่เรียกว่าโมดุล คือรวมกลุ่มฟังก์ชันที่ทำงานในลักษณะคล้ายกันเป็นกลุ่มเดียวกัน และเป็นอิสระซึ่งกันและกัน เช่น กลุ่มจัดการแฟ้มข้อมูล กลุ่มจัดการข้อมูล กลุ่มคำนวณสถิติพื้นฐาน เป็นต้น

2. มีคำสั่งและการตั้งค่าปริยายที่ส่วนกลาง ทำหน้าที่ให้บริการแก่โมดุลอื่นๆ โดยการทำงานจะเริ่มต้นจากการเรียกใช้โมดุลส่วนกลางนี้ก่อน หลังจากนั้นแล้ว โมดุลต่างๆ จะทำงานโดยไม่ขึ้นกับโมดุลอื่นใดอีก ยกเว้นจะทำเป็นกลุ่มเกี่ยวเนื่องกลุ่มเดียวกัน ก็อาจเรียกใช้กันเองไปมาระหว่างกันได้

3. แต่ละโมดุลมีส่วนติดต่อกับผู้ใช้แบบกราฟิกเฉพาะของมันเอง และไม่ขึ้นกับส่วนติดต่อกับผู้ใช้ของโมดุลอื่น

4. โมดุลต่างๆ ทำงานเป็นระบบเดียวกันโดย

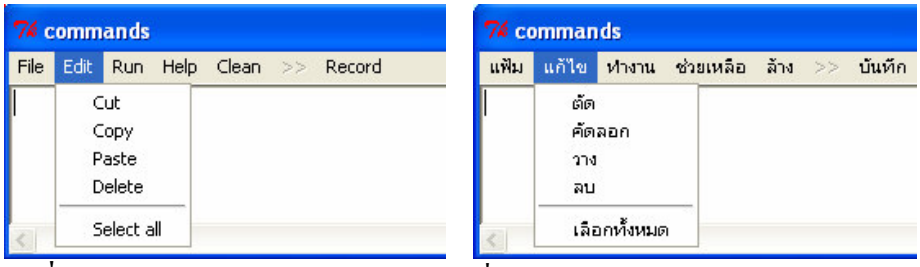
   ก. รูปแบบของรายการเมนูและหน้าต่างตัวเลือกต่างๆ มีลักษณะคล้ายกัน ทำให้ผู้ใช้สามารถเรียนรู้เริ่มต้นจากโมดุลหนึ่ง ก็สามารถใช้งานโมดุลอื่นๆ ได้ทันที

   ข. การทำงานของทุกรายการที่เลือกจะสร้างเป็นบรรทัดคำสั่งไปทำงานบน R console ได้ทั้งหมด และ

   ค. บรรทัดคำสั่งที่สร้างขึ้นนี้ สามารถบันทึกลงแฟ้มสคริปต์เดียวกันได้ ซึ่งในที่สุดเมื่อทำงานจากรายการเมนูเสร็จแล้ว จะสามารถทำซ้ำโดยการเรียกแฟ้มสคริปต์นั้นทำงานได้โดยไม่ต้องเลือกรายการเมนูต่างๆ ซ้ำอีก

แนวทางหลักๆ ดังกล่าวนี้ ทำให้เกิดการทำงานสองสายงานขนานกัน คือสายงานการพัฒนาคำสั่ง ซึ่งทำงานโดยนักสถิติ กับสายงานการพัฒนาตัวเชื่อมต่อผู้ใช้แบบกราฟิก ซึ่งทำงานโดยโปรแกรมเมอร์ ซึ่งสายงานการพัฒนาทั้งสองนี้จะทำงานเชื่อมโยงกันด้วยวิธีการออกคำสั่งของฟังก์ชันต่าง ข้อดีของการแบ่งงานลักษณะนี้คือนักสถิติและโปรแกรมเมอร์ไม่จำเป็นต้องเรียนรู้ข้ามสาขา แต่ละกลุ่มเพียงแต่สร้างแพคเกจของตนให้ทำงานประสานต่อเนื่องกันเท่านั้น

ข้อดีของระบบการทำงานในลักษณะดังกล่าวนี้อีกประการหนึ่งคือ โมดุลติดต่อกับผู้ใช้สามารถทำเป็นภาษาต่างๆ ได้ทุกภาษา เท่าที่หน้าต่าง tk จะแสดงภาษานั้นๆ ได้ ซึ่งในปัจจุบัน tcl/tk ในระบบวินโดวส์สามารถแสดงภาษาไทยได้ ดังรูปที่ 3.2 แต่บนระบบลินุกซ์และแมคอินทอช OSX ยังแสดงภาษาไทยไม่ถูกต้อง เชื่อว่าในอนาคตอันใกล้นี้จะแสดงภาษาไทยได้อย่างถูกต้องเช่นกัน



รูปที่ 3.2 ตัวอย่างหน้าต่าง ice.commands ที่แสดงรายการเมนูเป็นภาษาอังกฤษและภาษาไทย

สภาพแวดล้อม R-ICE มีข้อจำกัดอะไรบ้าง

เหตุผลที่ผู้เขียนเลือกใช้แพคเกจ tcltk ในการพัฒนาสภาพแวดล้อม **R-ICE** ขึ้นมานั้น ก็เนื่องจากเป็นแพคเกจที่ติดตั้งมาพร้อมกับ **R** ทำให้สามารถใช้ได้บนทุกระบบปฏิบัติการโดยไม่ต้องมีการดัดแปลงหรือเปลี่ยนแปลงใดๆ อีกทั้งเป็นแพคเกจที่อำนวยความสะดวกในการทำรายการเมนูและหน้าต่างตัวเลือกต่างๆ ค่อนข้างมาก

แม้ว่าสภาพแวดล้อม **R-ICE** จะช่วยให้ **R** สามารถทำงานในระบบกราฟิกและมีรายการเมนูต่างๆ คล้ายโปรแกรมที่มีลิขสิทธิ์ทางการค้า เช่น **SPSS** และ **Stata** เนื่องจากแพคเกจ tcltk ของ **R** ยังมีข้อจำกัดอยู่บ้าง เช่น ยังไม่สามารถแสดง shortcut key บนรายการเมนูได้ ดังเห็นได้ในรูปที่ 3.2

การกำหนดค่าปริยาย หรือจดจำตำแหน่งของหน้าต่างที่วางไว้เป็นครั้งสุดท้าย เพื่อที่เมื่อเปิดครั้งต่อไปหน้าต่างเหล่านั้นจะได้สร้างตรงตำแหน่งเดิมก็ทำไม่ได้ เมื่อเรียกหน้าต่างขึ้นมา หน้าต่างเหล่านั้นจะวางอยู่บนหน้าจอโดยไม่มีแบบแผน ทำให้ผู้ใช้ต้องเลื่อนไปวางในตำแหน่งที่ต้องการทุกครั้ง ทั้งนี้เพราะ tcltk ถูกพัฒนาขึ้นบนระบบยูนิกซ์ ซึ่งไม่สนใจตำแหน่งของหน้าต่างบนหน้าจอและปล่อยให้ผู้ใช้เลื่อนและวางหน้าต่างในที่ต่างๆ ได้ตามความต้องการ

และนอกจากนี้สภาพแวดล้อม **R-ICE** เพิ่งถูกพัฒนาขึ้นเป็นครั้งแรก ผู้พัฒนายังขาดประสบการณ์อยู่อีกมากในการทำงานกับแพคเกจ tcltk ทำให้ยังใช้ความสามารถของ tcltk ได้ไม่เต็มที่ เมื่อผู้ใช้ได้ใช้งานแล้วก็จะพบสิ่งที่ต้องแก้ไขต่อไปอีกเป็นจำนวนมาก แต่อย่างไรก็ตาม เมื่อได้เริ่มการพัฒนาในขั้นต้นแล้ว การปรับปรุงให้ดีขึ้นเป็นลำดับก็นับว่าไม่ยากและมีทางเป็นไปได้สูง

สภาพแวดล้อม R-ICE ประกอบด้วยอะไรบ้าง

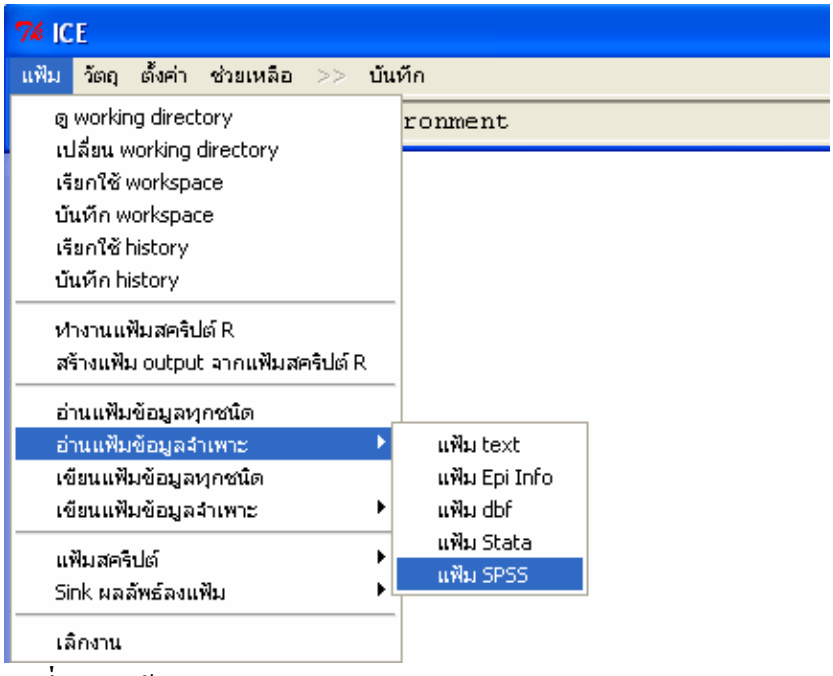
โมดุลภายใต้ระบบ **R-ICE** มี 8 โมดุลหลัก คือ 1. ice 2. ice.main 3. ice.dataman 4. ice.summary 5. ice.graph 6. ice.statis 7. ice.objects 8. ice.commands ซึ่งมีแต่โมดุล ice.main เท่านั้นที่มีตัวเลือกในการตั้งค่าปริยายของระบบ และการบันทึกคำสั่งลงแฟ้มสคริปต์ นอกจากนี้ ยังมีโมดุลที่รวมโมดุลกราฟิกหลายโมดุลเข้าด้วยกัน ได้แก่ ice.datamain เป็นการรวมโมดุล ice.main

กับ ice.dataman เข้าด้วยกัน ice.fullmain เป็นการรวมโมดุล ice.main, ice.dataman, ice.summary, ice.graph และ ice.statis เข้าด้วยกัน ทั้งนี้ เพื่อให้ผู้ใช้มีอิสระในการเลือกผสมการใช้งานโมดุลต่างๆ ได้ตามชอบใจ และนอกจากนี้ด้วยหลักการข้างต้น ยังสามารถสร้างโมดุลเพิ่มขึ้นได้อีก โดยสร้างโมดุลส่วนติดต่อกับผู้ใช้แบบกราฟิกครอบแพคเกจอื่นที่มีผู้สร้างไว้แล้วได้อย่างไม่จำกัด โมดุลต่างๆ เหล่านี้สามารถดาวน์โหลดมาใช้ได้จากเว็บไซต์ http://www.R-ICE.org และเนื่องจาก **R-ICE** เป็นระบบเปิดที่ยินดีต้อนรับนักพัฒนาทุกคน ทุกชาติ ทุกภาษา เว็บไซต์นี้จึงเปิดรับผู้สนใจที่จะสร้างส่วนติดต่อกับผู้ใช้เพิ่มเติมด้วย

ตารางที่ 3.1 แพคเกจหรือโมดุลต่างๆ ของ **R-ICE** และหน้าที่การทำงาน

แพคเกจ	**แพคเกจพึ่งพิง**	**หน้าที่การทำงาน**
แพคเกจส่วนกลาง		
ice	-	ฟังก์ชันส่วนกลางที่ถูกเรียกโดยแพคเกจอื่นๆ
แพคเกจหลัก		
ice.main	ice	รายการเมนูหลักของ ICE ประกอบด้วย File, Objects, Preference, Help และปุ่ม Record
แพคเกจสนับสนุน		
ice.dataman	ice	การจัดการข้อมูล
ice.summary	ice	การสรุปวัตถุและการทำตาราง
ice.graph	ice	การทำกราฟพื้นฐาน
ice.statis	ice	การทดสอบทางสถิติและโมเดลอย่างง่าย
ice.commands	-	หน้าต่างพิมพ์และแก้ไขบรรทัดคำสั่ง
ice.objects	-	หน้าต่างแสดงวัตถุในหน่วยความจำของ **R**

หน้าต่างหลักของสภาพแวดล้อม **R-ICE** คือหน้าต่างหลัก **ICE** ที่แนะนำให้วางไว้เหนือหน้าต่าง R console ดังแสดงในรูปที่ 3.1 ข้างต้น และแสดงภาพขยายให้ชัดเจนขึ้นดังรูปที่ 3.3 และมีรายละเอียดของรายการเมนูต่างๆ ดังตารางที่ 3.2



รูปที่ 3.3 หน้าต่างหลักของ **ICE**

ตารางที่ 3.2 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	

ในบทนี้จะยังไม่กล่าวถึงรายละเอียดการทำงานของรายการต่างๆ ในตารางที่ 3.2 แต่จะได้นำไปกล่าวในบทที่ 4 เรื่อง **การทำงานและวัตถุพื้นฐานในสภาพแวดล้อม R** และบทที่ 5 เรื่อง **การนำข้อมูลเข้าสู่สภาพแวดล้อม R** ต่อไป ส่วนโมดุลอื่นๆ ก็จะได้กล่าวในบทที่เกี่ยวข้องเช่นกัน

จะติดตั้งสภาพแวดล้อม ICE ได้อย่างไร

เนื่องจากสภาพแวดล้อม **R-ICE** เป็นชุดของแพคเกจที่ทำงานร่วมกันเป็นระบบเดียว การติดตั้งสภาพแวดล้อม **R-ICE** จึงทำเหมือนการติดตั้งแพคเกจในสภาพแวดล้อม **R** ทั่วไป ดังได้กล่าวแล้วในหัวข้อ ***การติดตั้งแพคเกจเพิ่มเติมเพื่อเพิ่มความสามารถให้ R*** ในบทเรื่อง **สภาพแวดล้อม R**

จากตารางที่ 3.1 ในหัวข้อ สภาพแวดล้อม **R** ประกอบด้วยอะไรบ้าง ข้างต้น จะเห็นได้ว่าแพคเกจที่เป็นแกนกลางของระบบคือแพคเกจ ice ดังนั้นจึงต้องติดตั้งแพคเกจนี้เป็นอันดับแรก ต่อมาจะติดตั้งแพคเกจ ice.main ส่วนแพคเกจอื่นๆ นอกจากนั้น จะติดตั้งทีหลังตามลำดับที่เรียงไว้ในตารางนั่นเอง ทบทวนอีกครั้งหนึ่ง แพคเกจจะถูกติดตั้งตามลำดับดังนี้ 1. ice 2. ice.main 3. ice.dataman 4. ice.summary 5. ice.graph 6. ice.statis 7. ice.commands และ 8. ice.objects หรือหากจะเลือกส่วนผสมแบบ fullmain ก็จะติดตั้ง 1. ice 2. ice.fullmain 3. ice.commands และ 4. ice.objects ก็ได้

เมื่อติดตั้งแพคเกจเสร็จแล้ว ขั้นตอนต่อไปคือการเรียกแพคเกจเหล่านั้นมาใช้งาน เนื่องจากมีแพคเกจหลายแพคเกจที่จะถูกเรียกใช้งานร่วมกัน หากจะต้องพิมพ์ `library(package.name)` เพื่อเรียกแพคเกจเหล่านั้นมาใช้งานก็จะเป็นเรื่องยุ่งยาก ผู้ใช้จึงควรสร้างแฟ้มสคริปต์ชื่อ start.ice ด้วยโปรแกรม text editor ตัวใดตัวหนึ่งที่ได้แนะนำไว้แล้วในบทที่ 2 เรื่อง **สภาพแวดล้อมเสริมการทำงาน** นั่นเอง โดยบรรจุบรรทัดคำสั่งต่อไปนี้

```
# 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()
```

ส่วนแรกเป็นการเรียกแพคเกจที่ถูกติดตั้งแล้วนั้นมาใช้งาน ส่วนล่างเป็นการเปิดหน้าต่างเพื่อสร้างสภาพแวดล้อม **R-ICE** ขึ้น หากไม่ต้องการเรียกใช้แพคเกจหรือฟังก์ชันใดมาใช้งาน ก็เพียงแต่ใส่เครื่องหมาย # หน้าบรรทัดนั้นๆ

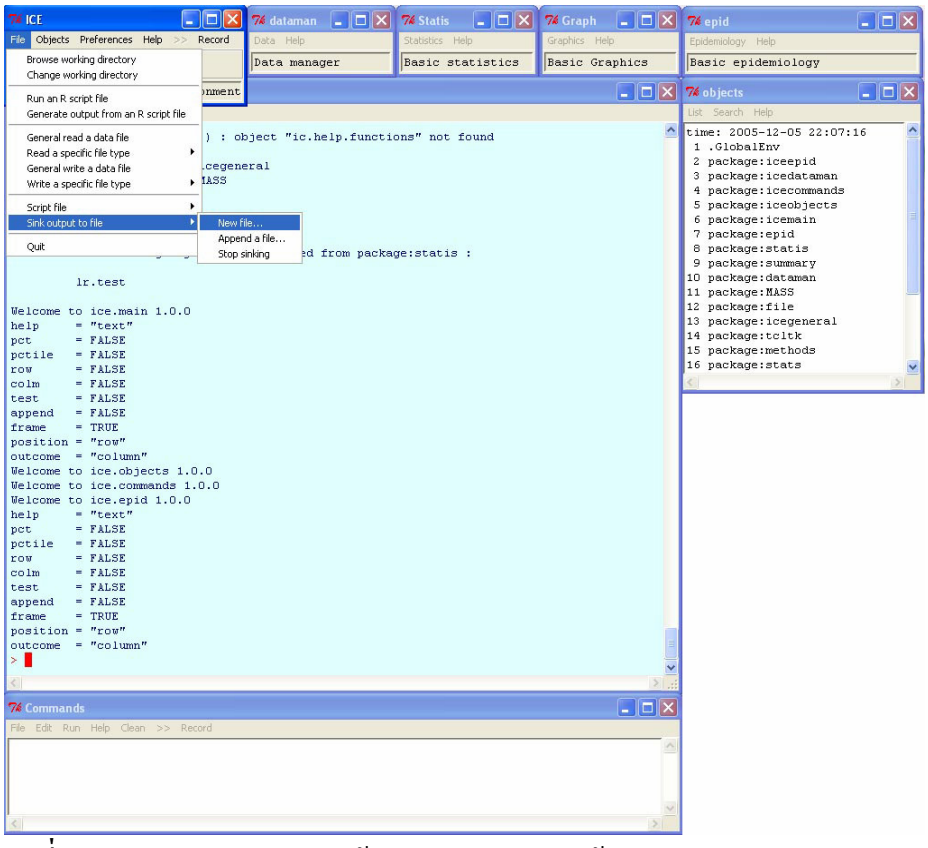
หมายเหตุ ต้องไม่ลืมว่าการติดตั้งแพคเกจเป็นการติดตั้งแพคเกจเข้าสู่เครื่องคอมพิวเตอร์ แต่ไม่ใช่การเรียกใช้งาน การเรียกใช้งานต้องทำในแต่ละครั้งเมื่อเปิดใช้สภาพแวดล้อม **R** ด้วยคำสั่ง `library()` และแพคเกจที่ถูกเรียกใช้งานในหน่วยความจำจะยังไม่ทำงานถ้าไม่เรียกใช้ฟังก์ชันที่มีอยู่ในแพคเกจนั้น ครึ่งล่างของสคริปต์นี้จึงเป็นการเรียกฟังก์ชันมาใช้งานนั่นเอง

บันทึกแฟ้ม ice.start ไว้ที่ home ของสภาพแวดล้อม **R** ซึ่งบนระบบปฏิบัติการวินโดวส์อยู่ที่ ..\R\R-2.4.x และบนระบบลินุกซ์และแมคอินทอช OSX จะอยู่ที่ home ของผู้ใช้แต่ละคนนั่นเอง ในการเรียกใช้งานแฟ้มสคริปต์นี้ ทำได้ง่ายๆ โดยการพิมพ์ `source("start.ice")` ที่ R console หรือบนระบบลินุกซ์ เมื่อเรียกใช้ และอยู่ในสภาพแวดล้อม **R** แล้ว (มี prompt เป็น >) ย้ำอีกครั้ง

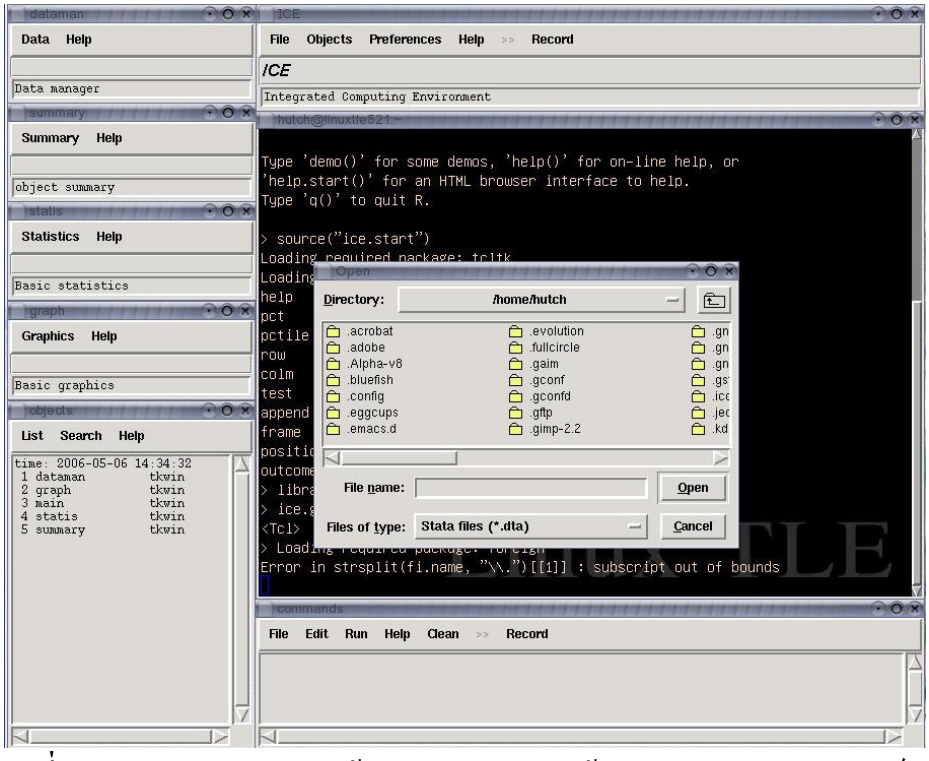
หนึ่งว่าการเรียกใช้สภาพแวดล้อม **ICE** ภายใต้สภาพแวดล้อม **R** ให้ทำโดยพิมพ์คำสั่งต่อไปนี้บน R console

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

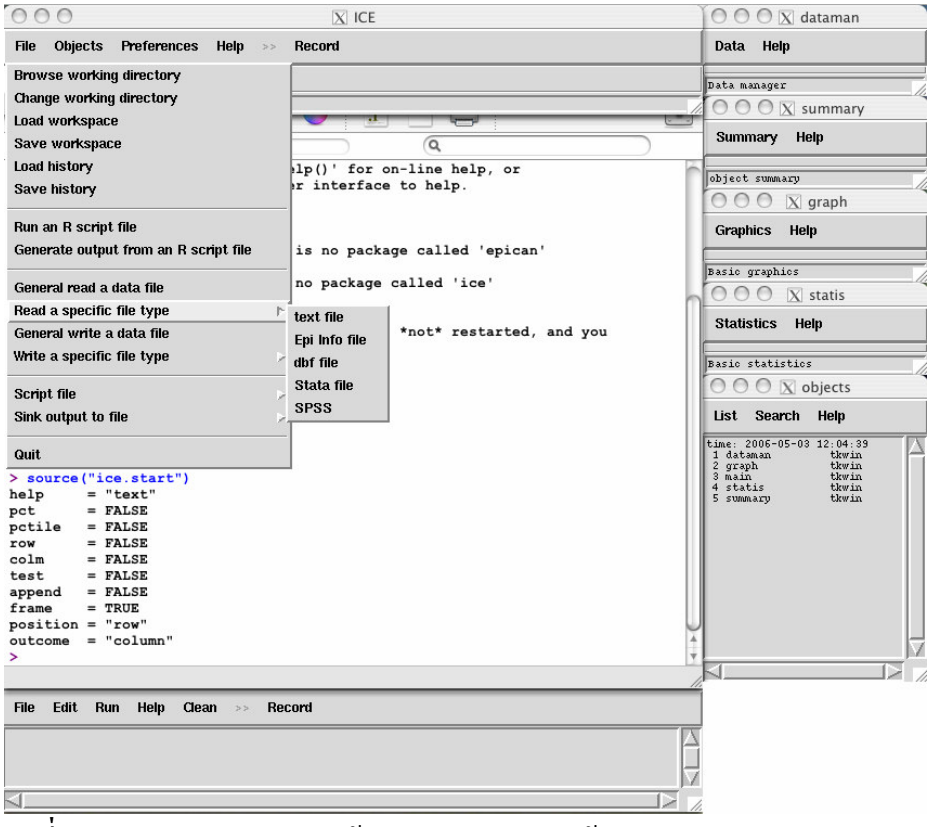
เมื่อพิมพ์และกดแป้น Enter แล้ว หน้าต่างของ **R-ICE** ต่างๆ จะถูกเปิดขึ้นมาบนหน้าจอ โดยจะยังไม่อยู่ในที่ที่ต้องการ ผู้ใช้มีอิสระที่จะจัดเรียงหน้าต่างเหล่านั้นในรูปแบบที่ถนัดกับการใช้งาน เช่น อาจจะจัดดังรูปที่ 3.1 หรือดังรูปที่ 3.4 หรือในรูปแบบอื่นๆ ตามใจชอบ



รูปที่ 3.4 ตัวอย่างการจัดเรียงหน้าต่างของสภาพแวดล้อม **R-ICE** บนระบบวินโดวส์



รูปที่ 3.5 ตัวอย่างการจัดเรียงหน้าต่างของสภาพแวดล้อม **R-ICE** บนระบบลินุกซ์



รูปที่ 3.6 ตัวอย่างการจัดเรียงหน้าต่างของสภาพแวดล้อม **R-ICE** บนระบบแมคอินทอช OSX

ผู้ใช้อาจรู้สึกยุ่งยากที่จะต้องมาจัดเรียงหน้าต่างทุกครั้งที่เรียกใช้งาน แต่นั่นก็ทำให้มีอิสระที่จะเลื่อนไปวางไว้ที่ใดก็ได้ เช่นหากเปิดโปรแกรมอื่นๆ ด้วย ก็จะวางหน้าต่างเพื่อหลบไม่ให้บังซ้อนทับกันได้

บทที่ 4 การทำงานและวัตถุพื้นฐานในสภาพแวดล้อม R

สภาพแวดล้อม **R**

การคำนวณพื้นฐาน

เวกเตอร์

เวกเตอร์ตัวเลข

เวกเตอร์ตรรกะ

เวกเตอร์ตัวอักษร

แฟกเตอร์

Missing value

เวกเตอร์ดัชนี

อาเรย์และเมทริกซ์

ลิสต์และกรอบข้อมูล

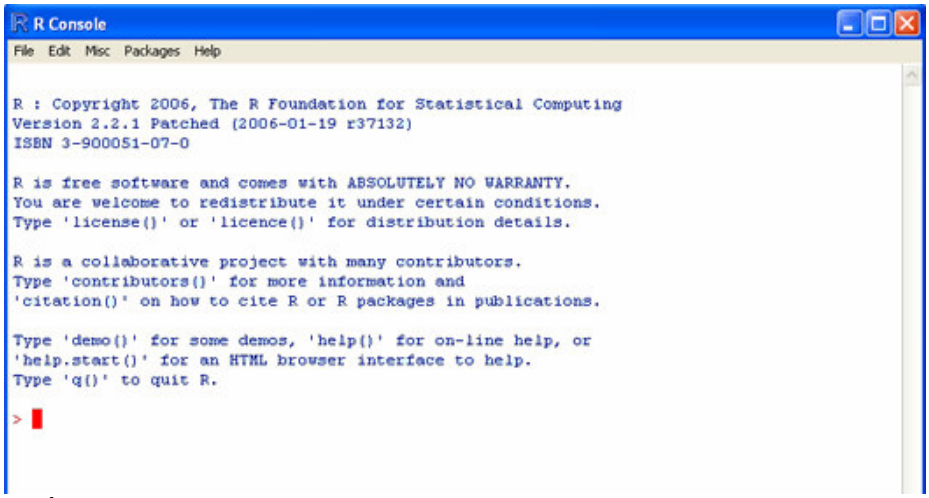
attach และ detach

การเรียกดูวัตถุในหน่วยความจำด้วยหน้าต่าง ice.objects

การพิมพ์คำสั่งด้วยหน้าต่าง ice.commands

สภาพแวดล้อม R

ก่อนที่จะกล่าวถึงสภาพแวดล้อม **R-ICE** จำเป็นต้องแนะนำการทำงานในสภาพแวดล้อม **R** เพื่อเป็นพื้นฐานเสียก่อน เมื่อเรียกใช้งาน **R** ขึ้นมา จะได้หน้าจอดังรูปที่ 4.1



รูปที่ 4.1 หน้าจอ R Console เมื่อเรียกใช้งาน **R**

ข้อความต้อนรับของ **R** จะกล่าวถึงขอดูลิขสิทธิ์ ด้วยฟังก์ชัน `license()` การดูรายชื่อผู้มีส่วนร่วมในการพัฒนาด้วยฟังก์ชัน `contributors()` และการเขียนอ้างอิงโปรแกรม **R** ด้วยฟังก์ชัน `citation()` และการเรียกดูการสาธิตด้วยฟังก์ชัน `demo()` และการเรียกความช่วยเหลือแบบต่างๆ ซึ่งได้กล่าวมาแล้วในบทที่ 1 เรื่อง **สภาพแวดล้อม R** หน้าจอในส่วนนี้เรียกว่า **R** console ซึ่งเราสามารถพิมพ์บรรทัดคำสั่งต่างๆ ลงไปให้ **R** ทำงาน และ **R** จะแสดงผลลัพธ์ของการทำงานกลับมาบนหน้าจอนี้เช่นกัน

การคำนวณพื้นฐาน

ที่หน้าจอนี้ ผู้ใช้สามารถพิมพ์คำสั่งต่างๆ ลงหลัง prompt ที่เป็นเครื่องหมาย `>` ได้ การคำนวณทางคณิตศาสตร์ต่างๆ สามารถทำได้ด้วย operator พื้นฐานทางคณิตศาสตร์ต่างๆ ได้แก่ `+`, `-`, `*`, `/`, `^` (บวก ลบ คูณ หาร และยกกำลัง) และ operator ทางตรรกศาสตร์ ได้แก่ `&`, `|`, `!` (and or และ not) รวมไปถึงการเปรียบเทียบต่างๆ เช่น <, >, <=, >=, ==, != (น้อยกว่า มากกว่า น้อยกว่าหรือเท่ากับ มากกว่าหรือเท่ากับ เท่ากับ และไม่เท่ากับ) และเครื่องหมายอื่นๆ

ในขั้นแรกนี้ ลองพิมพ์ 2+3 แล้วกดแป้น Enter จะได้ผลบวกของจำนวน 2 และ 3 เป็น 5

```
> 2+3
[1] 5
```

นั่นคือเราสามารถทำงานกับตัวเลขได้เหมือนเครื่องคิดเลข และยังสามารถคำนวณฟังก์ชันต่างๆ เช่น รากที่สอง (`sqrt()`), exponential (`exp()`) และ logarithm (`log()`) ได้ด้วย เช่น

```
> sqrt(2)+sqrt(3)
[1] 3.146264
```

บรรทัดถัดลงมาเป็นผลลัพธ์ เช่นในที่นี้ได้ผลของการบวกรากที่สองของ 2 กับรากที่สองของ 3 เป็น 3.146264 เครื่องหมาย [1] ด้านหน้าหมายถึงผลลัพธ์ตัวที่ 1 ซึ่งในที่นี้ผลของการคำนวณมีค่าเดียว จึงแสดงเพียงค่าเดียว นอกจากนี้ยังมี operator และคำสั่งให้ใช้อีกมากมาย ซึ่งอาจหารายละเอียดได้โดยดูใน library base และ stats โดยใช้คำสั่ง `help.start()` แล้วเลือกตัวเลือก Packages แล้วเลือกชื่อ library ที่ต้องการความช่วยเหลือของคำสั่งต่างๆ ใน library นั้นๆ

ต่อไปจะได้เห็นกรณีที่การคำนวณได้ผลลัพธ์มากกว่าหนึ่งค่า ก็จะแสดงค่าไปตามลำดับ ในแนวแถว และหากขึ้นบรรทัดใหม่ ก็จะบอกลำดับของตัวแรกในแถวถัดไปไว้หน้าบรรทัด

เวกเตอร์

เวกเตอร์ตัวเลข

โดยปกติในการคำนวณทางสถิติ เราจะมีข้อมูลเป็นชุดจากคนหลายคนหรือจากการวัดหลายครั้ง เราจะสร้างข้อมูลในลักษณะเวกเตอร์ คือเป็นชุดของข้อมูลหลายๆ ค่าเรียงกันดังนี้

```
> x <- c(10.4, 5.6, 3.1, 6.4, 21.7)
```

ตัวอย่างข้างบนเป็นการสร้างเวกเตอร์ตัวเลข (numeric vector) หนึ่งชุด ด้วยคำสั่ง `c()` แล้วกำหนดชื่อให้เป็น `x` โดยใช้เครื่องหมาย `<-` (ต้องพิมพ์เครื่องหมาย `<` และ `-` ติดกัน หากพิมพ์แยกกัน **R** จะอ่านเป็น น้อยกว่า ลบ) แม้ว่าเราสามารถใช้เครื่องหมาย = แทนได้ แต่ไม่แนะนำให้ใช้ และเรายังอาจใช้เครื่องหมาย `->` ได้เช่นกัน แต่ในกรณีนี้จะต้องกำหนดชื่อวัตถุไว้ด้านขวา ดังนี้

```
> c(10.4, 5.6, 3.1, 6.4, 21.7) -> x
```

เมื่อกดแป้น Enter หน้าจอจะไม่แสดงอะไรขึ้นมา เนื่องจากเวกเตอร์ที่สร้างขึ้นได้ถูกเก็บไว้ในวัตถุชื่อ `x` แล้ว ในกรณีนี้เราเรียกว่าเวกเตอร์ `x` มีขนาดหรือความยาว 5 หน่วย ถ้าเราออกคำสั่งบรรทัดถัดไปเพื่อหาค่าของ `1/x` ก็จะได้ผลลัพธ์ออกมา ดังนี้

```
> 1/x
[1] 0.09615385 0.17857143 0.32258065 0.15625000 0.04608295
```

จะเห็นว่าเราได้ผลลัพธ์ที่มีตัวเลข 5 จำนวน เป็นผลของ 1 หารด้วยค่าของ `x` ไปตามลำดับ และถ้าเราพิมพ์บรรทัดต่อไปเป็น

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

จะได้เวกเตอร์ y ที่มีจำนวนหน่วยย่อย 11 หน่วย ประกอบด้วย x สองชุดแยกกันด้วยเลข 0

เราอาจนำเวกเตอร์สองจำนวนขึ้นไปที่มีความยาวเท่ากัน มาคำนวณทางคณิตศาสตร์ได้เหมือนที่ทำกับเลขตัวเดียว โดยที่ **R** จะทำงานกับหน่วยย่อยที่ตรงกันไปตามลำดับ ดังนี้

```
> x <- c(1,2,3,4,5,6,7,8,9)
> y <- c(9,8,7,6,5,4,3,2,1)
> v <- 2*x + y + 1
> v
[1] 12 13 14 15 16 17 18 19 20
```

เราอาจสร้างเวกเตอร์ที่มีค่าเพิ่มขึ้นหรือลดลงทีละ 1 เรียงกันเป็นลำดับได้โดยเพียงเขียนคำสั่งสั้นๆ เช่น `1:30` จะได้เวกเตอร์ที่มีค่าเป็น `c(1, 2, ..., 29, 30)` และหากต้องการสร้างเวกเตอร์ที่มีค่าภายในเพิ่มขึ้นหรือลดลงทีละหน่วยที่ไม่ใช่ 1 ก็ใช้ฟังก์ชัน `seq()` ดังนี้

```
> seq(-5, 5, by=.2) -> a
```

ก็จะได้เวกเตอร์ a ที่มีค่าเป็น `c(-5.0, -4.8, -4.6, ..., 4.6, 4.8, 5.0)`

ถ้าต้องการสร้างเวกเตอร์ตัวเลขที่เรียงลำดับจากค่าหนึ่งถึงอีกค่าหนึ่ง ให้ใช้เครื่องหมาย : (colon) หากต้องการสร้างเวกเตอร์จากการทำซ้ำค่าเดียว หรือทำซ้ำค่าในเวกเตอร์ เราสามารถใช้คำสั่ง `rep()` ได้ ดังตัวอย่าง

```
> x <- 1:9
> b <- rep(x, times=5)
[1]  1 2 3 4 5 6 7 8 9 1 2 3 4 5 6 7 8 9 1 2 3 4 5 6 7 8 9 1 2
[30] 3 4 5 6 7 8 9 1 2 3 4 5 6 7 8 9
> s <- rep(x, each=5)
> s
[1]  1 1 1 1 1 2 2 2 2 2 3 3 3 3 3 4 4 4 4 4 5 5 5 5 5 6 6 6 6
[30] 6 7 7 7 7 7 8 8 8 8 8 9 9 9 9 9
```

ข้อควรระวัง มักเกิดความผิดพลาดขึ้นบ่อยจากการปิดวงเล็บต่างๆ ไม่หมดในการพิมพ์บรรทัดคำสั่งให้ **R** ทำงาน นอกจากเครื่องหมายวงเล็บกลม () แล้ว **R** ยังมีวงเล็บเหลี่ยม [] เพื่อใช้บอกตำแหน่งของหน่วยย่อยในเวกเตอร์หรือวัตถุใดๆ และเครื่องหมายวงเล็บปีกกา {} เพื่อใช้ในการเขียนฟังก์ชัน และนอกจากนี้ยังมีเครื่องหมายคำพูดให้ใช้อีกด้วย เครื่องหมายเหล่านี้จำเป็นต้องเขียนเป็นคู่ การเขียนเครื่องหมายนั้นๆ ไม่ครบคู่มักเป็นสาเหตุหลักที่ทำให้ **R** ทำงานผิดพลาดได้

ในการระบุตำแหน่งในเวกเตอร์หรือวัตถุที่มีหน่วยย่อยภายในนั้น จะใช้เครื่องหมายวงเล็บเหลี่ยม [] ดังเช่นการใช้งานในกรณีต่อไปนี้

```
> x <- (1:9)^2
> x
[1]  1  4  9 16 25 36 49 64 81
> x[6]
[1] 36
> x[3:7]
[1]  9 16 25 36 49
> x[6] <- 63
> x
[1]  1  4  9 16 25 63 49 64 81
```

บรรทัดแรกเป็นการสร้างเวกเตอร์ตัวเลขที่มีค่าระหว่างค่ากำลังสองของ 1 ถึง 9 (จึงต้องพิมพ์ 1:9 ไว้ในวงเล็บแล้วยกกำลังสองของทุกค่าในเวกเตอร์พร้อมกันทีเดียว) ซึ่งเราสามารถเรียกดูค่าของ x ทั้งหมดได้ด้วยการพิมพ์ x บนหน้าจอ แต่มีบ่อยครั้งที่เวกเตอร์มีความยาวมากๆ เราอาจต้องการเลือกดูค่าเพียงค่าเดียวหรือเพียงช่วงสั้นๆ ก็สามารถทำได้ด้วยการใช้เครื่องหมายวงเล็บเหลี่ยม [] เพื่อระบุตำแหน่งในเวกเตอร์ (หรือในวัตถุชนิดอื่นใดที่มีหน่วยย่อยภายใน) ที่ต้องการดูได้ดังตัวอย่างข้างต้น และนอกจากนี้ถ้าต้องการเปลี่ยนค่าใดในวัตถุนั้น ก็สามารถให้ค่าใหม่ไว้ด้านขวาของเครื่องหมาย <- และระบุตำแหน่งในวัตถุนั้นไว้ด้านซ้ายของเครื่องหมาย <- ดังตัวอย่างข้างต้น x[6] <- 63 เป็นการเปลี่ยนค่าในตำแหน่งที่ 6 ของ x ให้เป็น 63

เวกเตอร์ตรรกะ

นอกจากเวกเตอร์ตัวเลขแล้ว ยังมีเวกเตอร์ตรรกะ (logical vector) ที่มีค่าที่เป็นไปได้เป็น TRUE FALSE หรือ NA ค่า TRUE คือค่าที่เป็นจริง ค่า FALSE คือค่าที่เป็นเท็จ ส่วนค่า NA หมายถึงไม่มีค่า ซึ่งจะใช้ได้ทั้งในเวกเตอร์ตรรกะและเวกเตอร์ตัวเลขที่กล่าวมาแล้ว และจะได้กล่าวในรายละเอียดต่อไปในหัวข้อ **Missing Value**

สัญลักษณ์ที่ใช้เปรียบเทียบค่าตรรกะมีดังนี้ < (น้อยกว่า) <= (น้อยกว่าหรือเท่ากับ) > (มากกว่า) >= (มากกว่าหรือเท่ากับ) == (เท่ากับ) และ != (ไม่เท่ากับ) และ operator ที่ทำงานกับค่าตรรกะมี & (and), | (or), ! (not) และอื่นๆ ซึ่งดูได้ใน library base ด้วยวิธีที่กล่าวข้างต้น

เวกเตอร์ตัวอักษร

ค่าที่เป็นข้อความและเวกเตอร์ตัวอักษร (character vector) ถูกนำมาใช้บ่อยๆ เช่น ในการระบุชื่อนามสกุลของบุคคล ชื่อของสิ่งต่างๆ และการเขียนกราฟ การสร้างข้อความใน **R** ทำได้โดยใส่ข้อความไว้ภายในเครื่องหมายคำพูด " หรือ ' หากไม่ใส่เครื่องหมายคำพูดล้อมรอบข้อความแล้ว **R** จะตีความเป็นชื่อของวัตถุในหน่วยความจำของ **R** นั่นเอง ตัวอย่างเช่น

```
> name1 <- c("Donald","Edward","Marry")
> name2 <- c(name1, "Robert")
> name1
```

```
[1] "Donald" "Edward" "Marry"
> name2
[1] "Donald" "Edward" "Marry"  "Robert"
```

`name1` และ `name2` เป็นชื่อเวกเตอร์ตัวอักษร และสามารถนำมาใช้สร้างเวกเตอร์ตัวอักษรอื่นได้อีก เช่นตัวอย่างข้างต้น `name2` สร้างมาจาก `name1` โดยเพิ่มชื่อ `"Robert"` เข้าไปอีกหนึ่งชื่อ ดังนั้นหากต้องการให้ `name2` ประกอบด้วยชื่อ `name1` และ `Robert` ก็ต้องใส่เครื่องหมาย " ล้อมรอบคำ `name1` ด้วย เพื่อไม่ให้ **R** ตีความคำว่า `name1` เป็นชื่อวัตถุ

นอกจากนี้ยังสามารถใส่สัญลักษณ์พิเศษในลักษณะเดียวกับภาษา **C** หรือ **C++** ซึ่งจะต้องใส่เครื่องหมาย \ นำหน้าตัวอักษรหรือสัญลักษณ์ โดยทั่วไปมีที่ใช้อยู่ไม่มากนัก เช่น `\n` หมายถึงขึ้นบรรทัดใหม่ `\t` หมายถึงแท็บ เป็นต้น และหากต้องการพิมพ์เครื่องหมายใดๆ ที่เป็นสัญลักษณ์สงวนของภาษา **R** เช่นเครื่องหมาย \ ให้ใช้สัญลักษณ์ \ ซ้อนกันสองตัวเป็น `\\` และหากต้องการแสดงเครื่องหมายคำพูด (ฟันหนู) ให้ใช้ `\"` เป็นต้น ด้วยเหตุนี้เอง หากต้องการพิมพ์ชื่อโฟลเดอร์เต็มๆ บนระบบวินโดวส์ซึ่งปกติใช้อักษร \ เพื่อบอกโฟลเดอร์ย่อยจึงต้องใช้เครื่องหมาย `\\` หรือเลี่ยงไปใช้เครื่องหมาย / เหมือนบนระบบลินุกซ์และแมคอินทอชแทน ดังนี้ `"C:\\Program Files\\R\\R-2.2.1"` หรือ `"C:/Program Files/R/R-2.2.1"`

> **หมายเหตุ** หากต้องการใส่เครื่องหมาย " หรือ ' ไว้ในข้อความ เช่นต้องการคำว่า Robert's ทำได้โดยใส่ไว้ในเครื่องหมายคำพูดที่แตกต่างกัน ดังนี้ `"Robert's"` และหากต้องการสร้างข้อความ `Robert says "Hi".` ก็สามารถทำได้สองวิธี วิธีหนึ่งคือใช้เครื่องหมาย ' ล้อมรอบ `"Hi"` หรืออีกวิธีหนึ่งคือใช้ `\"` เพื่อพิมพ์อักษร " นั่นเอง ดังนี้ `"Robert says \"Hi\"."`

นอกจากคำสั่ง `c()` ที่ใช้ในการสร้างเวกเตอร์ตัวอักษรเหมือนเวกเตอร์อื่นๆ แล้ว ยังมีคำสั่งอื่นที่ใช้ในการทำงานกับข้อความและเวกเตอร์ตัวอักษรอีก ได้แก่ `nchar()`, `paste()`, `strsplit()`, `substr()`, และ `substring()` แต่ผู้ใช้ระดับเริ่มต้นมักไม่มีความจำเป็นต้องใช้แต่อย่างใด จึงจะไม่กล่าวในที่นี้

แฟกเตอร์

แฟกเตอร์ (factor) เป็นวัตถุชนิดเวกเตอร์แบบพิเศษที่ระบุการแบ่งกลุ่มของข้อมูลเป็นพวกๆ เช่น เพศ เชื้อชาติ ศาสนา อาชีพ ซึ่งไม่มีลำดับ และยังใช้กับการแบ่งกลุ่มที่มีลำดับเช่น การแบ่งกลุ่มการศึกษาเป็น ประถม มัธยม อุดมศึกษา เป็นต้น

เราอาจสร้างแฟกเตอร์ได้สองวิธี โดยสร้างจากเวกเตอร์ตัวอักษร หรือสร้างจากเวกเตอร์ตัวเลข ดังตัวอย่างต่อไปนี้

```
> sex <- c("male","male","female","male","female","female")
> typeof(sex)
[1] "character"
> f.sex <- factor(sex)
> typeof(f.sex)
[1] "integer"
> is.factor(f.sex)
[1] TRUE
> f.sex
[1] male   male   female male   female female
Levels: female male
```

ในขั้นแรกเราสร้างเวกเตอร์ตัวอักษรระบุเพศของตัวอย่างที่ศึกษา ซึ่งเมื่อตรวจสอบจะเห็นได้ว่า **R** เก็บเวกเตอร์นี้เป็นแบบ character จากนั้นเราเปลี่ยนเวกเตอร์ `sex` ให้เป็นแฟกเตอร์โดยใช้คำสั่ง `factor()` และนำผลไปเก็บในวัตถุชื่อ `f.sex` จากนั้นตรวจสอบวิธีที่ **R** เก็บข้อมูลจะเห็นได้ว่า **R** เปลี่ยนไปเก็บข้อมูลเป็นแบบตัวเลข แต่เมื่อเรียกดู `f.sex` กลับพบว่า **R** แสดงผลเป็นตัวอักษรไม่ใช่ตัวเลข แต่ได้บอกเพิ่มเติมว่า Levels: female male ซึ่งแสดงว่า **R** เก็บค่า female เป็น 1 และ male เป็น 2 ตามลำดับตัวอักษร (ในกรณีที่ไม่ได้ระบุค่ารหัส **R** จะเก็บข้อมูลแฟกเตอร์โดยเรียงลำดับจากเลข 1, 2, … ไปเรื่อยๆ จนหมดค่าที่เป็นไปได้ในเวกเตอร์นั้น โดยเรียงตามตัวอักษร)

ในกรณีเช่นเพศนี้ ในทางปฏิบัติในการเก็บข้อมูลเรามักจะระบุให้เพศชายเป็น 1 และเพศหญิงเป็น 2 เป็นรหัสสากลที่นิยมใช้กันทั่วโลก ดังนั้นที่ **R** ทำดังกล่าวข้างต้นจึงขัดกับที่ผู้ใช้ต้องการหรือคาดหวัง ดังนั้นพึงระวังในการกรอกข้อมูลเป็นตัวอักษร เพราะ **R** อาจทำงานผิดไปจากที่คาดได้ ในที่นี้จึงแนะนำให้สร้างวัตถุเป็นเวกเตอร์ตัวเลขก่อน โดย 1 มีความหมายเป็นชาย และ 2 มีความหมายเป็นหญิง แล้วจึงแปลงเป็นแฟกเตอร์ และให้คำอธิบายค่า ดังนี้

```
> sex <- c(1,1,2,1,2,2)
> typeof(sex)
[1] "double"
> f.sex <- factor(sex,label=c("male","female"))
> typeof(f.sex)
[1] "integer"
> f.sex
[1] male   male   female male   female female
Levels: male female
```

จะเห็นว่าคราวนี้ค่า 1 หมายถึงเพศชาย และค่า 2 หมายถึงเพศหญิงตามที่ต้องการแล้ว เนื่องจากในส่วนของ levels คำว่า male มาก่อน female

และหากต้องการให้แฟกเตอร์เก็บค่าที่มีการเรียงลำดับ เช่น ระดับการศึกษา ระยะของโรค ความพึงพอใจ ก็สามารถทำได้โดยใช้ตัวเลือก `ordered=TRUE` กับคำสั่ง `factor()` ที่กล่าวมาแล้ว หรือจะใช้คำสั่ง `ordered()` โดยตรงก็ได้ ดังนี้

```
> educ <- ordered(c(1,3,2,1,1,2,3,2,2,3,1,1,1,2),
+ label=c("primary","secondary","tertiary"))
> educ
 [1]  primary   tertiary  secondary primary   primary
 [6]  secondary tertiary  secondary secondary tertiary
 [11] primary   primary   primary   secondary
Levels: primary < secondary < tertiary
```

Missing value

ในเวกเตอร์หนึ่ง อาจมีบางครั้งที่ไม่ทราบค่าของบางตำแหน่งก็ได้ เช่น ในการตอบแบบสอบถาม ผู้ตอบอาจไม่ระบุอายุ ระดับการศึกษา หรือตัวบางตัวมาก็ได้ ในกรณีที่เป็น ***ค่าว่าง*** (missing value หรือ not available) เช่นนี้ โปรแกรมทางสถิติต่างๆ มีวิธีการใส่ค่าที่แตกต่างกัน แต่ห้ามใส่ค่า 0 เป็นอันขาด เพราะค่า 0 นั้นแสดงว่ามีค่า ในกรณีของ **R** จะใช้คำสงวน `NA` และสามารถใช้คำสั่ง `is.na()` เพื่อสร้างเวกเตอร์ตรรกะที่บอกตำแหน่ง ที่มีค่าเป็น `NA` ในเวกเตอร์ใดๆ ได้ด้วย ดังตัวอย่าง

```
> z <- c(1:3,NA);  ind <- is.na(z)
> ind
[1] FALSE FALSE FALSE  TRUE
```

ในบางครั้ง เราอาจหารค่าใดๆ ด้วยค่า 0 หรือผลการคำนวณค่าใดๆ กับ `Inf` (ค่าอนันต์) ซึ่งจะได้ผลลัพธ์ที่หาค่าไม่ได้ ในกรณีนี้ **R** จะให้ค่า `NaN` ซึ่งหมายความว่า ***ค่าที่ไม่ใช่ตัวเลข*** (not a number)

```
> x <- 0/0
> x
[1] NaN
> is.na(x)
[1] TRUE
> is.nan(x)
[1] TRUE
```

ซึ่งในกรณีนี้เมื่อตรวจสอบด้วยฟังก์ชัน `is.na()` ก็จะพบว่า `NaN` ให้ผลลัพธ์เป็นจริงเช่นกัน นั่นคือ `NaN` เป็นค่าที่หาค่าไม่ได้เช่นกัน แต่ไม่ได้เกิดจากการไม่ทราบค่า แต่เกิดจากการคำนวณที่ให้ผลลัพธ์หาค่าไม่ได้

สำหรับเวกเตอร์ตัวอักษรที่มีค่า `NA` อยู่ ในกรณีที่พิมพ์ผลลัพธ์ของเวกเตอร์บนหน้าจอโดยไม่มีเครื่องหมายคำพูดแสดงข้อความ **R** จะแสดงผลลัพธ์เป็น `<NA>` ทั้งนี้เพื่อให้ต่างจากข้อความว่า `"NA"` นั่นเอง (แต่ปกติไม่ควรใช้ข้อความนี้ เพราะจะเข้าใจผิดว่าเป็นค่าว่างได้)

```
> y <- c("Marry","Susan",NA,"NA","Christina")
> y
[1] "Marry"     "Susan"     NA          "NA"        "Christina"
> print.noquote(y)
```

```
[1] Marry     Susan     <NA>      NA        Christina
```

ยังมีค่าว่างอีกลักษณะหนึ่งที่ปกติผู้ใช้ทั่วไปไม่ได้ใช้ แต่อาจเกิดขึ้นจากการทำงานของ **R** ดังเช่นกรณีของ NaN ก็คือค่า NULL ซึ่งมีความหมายว่า ***วัตถุว่าง*** นั่นคือวัตถุนั้นทั้งหมดไม่มีค่า (แต่ไม่ได้หมายความว่าวัตถุนั้นประกอบด้วยค่า NA ทั้งหมด) มักจะเกิดขึ้นจะการส่งค่ากลับของบางฟังก์ชัน ที่ผลลัพธ์ของการทำงานอาจเกิดเป็นวัตถุว่างขึ้น ซึ่งมักแสดงว่ามีการออกคำสั่งบางอย่างผิดพลาด เช่น เลือกโมเดลในการคำนวณไม่ถูกต้อง ทำให้ **R** หาผลลัพธ์ของโมเดลนั้นไม่ได้

เวกเตอร์ดัชนี

เราอาจเลือกเซ็ตย่อยของหน่วยต่างๆ ของเวกเตอร์ใดๆ โดยการระบุเวกเตอร์ดัชนี (index vector) ไว้ภายในวงเล็บเหลี่ยม [ ] ต่อท้ายชื่อของเวกเตอร์นั้นๆ เวกเตอร์ดัชนีอาจเป็นเวกเตอร์แบบใดแบบหนึ่งใน 4 แบบต่อไปนี้

1. **เวกเตอร์ตรรกะ** ซึ่งจะต้องมีความยาวเท่ากับความยาวของเวกเตอร์ที่จะเลือก เฉพาะค่าที่เป็นจริงเท่านั้นที่จะถูกเลือก

```
> y <- c("Marry","Susan",NA,"Christina")
> z <- y[!is.na(y)]
> z
[1] "Marry"     "Susan"     "Christina"
> !is.na(y)
[1]  TRUE  TRUE FALSE  TRUE
```

2. **เวกเตอร์ตัวเลขบอกลำดับ**ของหน่วยย่อยของเวกเตอร์นั้น และหากต้องการเลือกค่าเดิมซ้ำหลายครั้งก็ระบุตำแหน่งของค่านั้นซ้ำได้ตามต้องการ หากระบุตัวเลขที่มีค่าเกินความยาวของเวกเตอร์นั้น จะให้ผลเป็น NA หรือหากเป็นจำนวนลบจะเกิดความผิดพลาดในการทำงาน ดูตัวอย่างดังนี้

```
> y <- c("Marry","Susan",NA,"Christina")
> z <- y[c(4,2,1)]
> z
[1] "Christina" "Susan"     "Marry"
> z <- y[c(7,4,2,1)]
> z
[1] NA          "Christina" "Susan"     "Marry"
> z <- y[c(1,1,2,4)]
> z
[1] "Marry"     "Marry"     "Susan"     "Christina"
```

3. **เวกเตอร์ตัวเลขที่นำหน้าทั้งเวกเตอร์ด้วยเครื่องหมายลบ** เป็นการบอกว่าไม่เลือกค่าที่ตำแหน่งนั้น เช่น

```
> y <- c("Marry","Susan",NA,"Christina")
> z <- y[-c(3)]
```

```
> z
[1] "Marry"     "Susan"     "Christina"
```

4. **เวกเตอร์ข้อความ** แบบนี้ใช้ได้กับเวกเตอร์ที่ระบุชื่อกำกับค่าด้วย attribute names (จะไม่กล่าวในที่นี้ ผู้สนใจอาจอ่านเพิ่มเติมได้ในคู่มือ An Introduction to R ด้วยคำสั่ง `help.start()` ของ **R** นั่นเอง) ดังตัวอย่างต่อไปนี้

```
> fruit <- c(5, 10, 1, 20)
> names(fruit) <- c("orange", "banana", "apple", "peach")
> lunch <- fruit[c("apple","orange")]
> lunch
 apple orange
     1      5
```

ข้อดีของการระบุชื่อให้กับตัวเลขก็คือทำให้จำค่าได้ง่าย แต่จะมีที่ใช้ในบางกรณีเท่านั้น คือค่าตัวเลขนั้นจะต้องเป็นรหัสของสิ่งของ ไม่ใช่ตัวเลขจริงที่นำมาคำนวณทางคณิตศาสตร์ นั่นคือเป็นแฟกเตอร์ แต่แทนที่จะระบุคำอธิบายหรือ `label` ให้กับค่าเหล่านั้น ก็ระบุเป็นชื่อ `names` แทน

หากระบุเวกเตอร์ดัชนีไว้ข้างที่เป็นด้านรับของการกำหนดค่าด้วย `<-` จะเป็นการกำหนดค่าใหม่ให้กับค่าในลำดับนั้นๆ ของเวกเตอร์ ดังตัวอย่างการเปลี่ยนค่า `NA` เป็นเลข 9 ต่อไปนี้

```
> x <- c(1,1,2,1,2,2,2,NA,2,1,1,1,NA)
> x[is.na(x)] <- 9
> x
 [1] 1 1 2 1 2 2 2 9 2 1 1 1 9
```

อาเรย์และเมทริกซ์

อาเรย์ (array) หรือแถวลำดับ และเมทริกซ์ (matrix) เป็นวัตถุที่ผู้ใช้ทั่วไปไม่ค่อยได้ใช้นัก นอกจากจะเป็นโปรแกรมเมอร์หรือนักวิชาการทางสถิติ ดังนั้นจึงจะกล่าวเพียงสั้นๆ ให้รู้จักไว้บ้างในที่นี้เท่านั้น ผู้สนใจรายละเอียดอาจหาอ่านได้ในคู่มือของ **R** โดยการใช้ฟังก์ชัน `help.start()`

การสร้างเวกเตอร์ที่กล่าวมาแล้วข้างต้นทั้งหมด เป็นการสร้างเวกเตอร์ในมิติ (dimension) เดียวเท่านั้น คือมีค่าเรียงกันไปตามลำดับ จะเห็นได้จากการที่ **R** แสดงค่าในแนวบรรทัดเรียงต่อกันไป และสามารถระบุตำแหน่งในเวกเตอร์ได้โดยการระบุตัวเลขในมิติเดียว

อาเรย์เป็นเวกเตอร์ที่มีหลายมิติ โดยทั่วไปเรามักคุ้นเคยกับอาเรย์ที่มีสองมิติ ซึ่งในกรณีนี้จะเรียกด้วยชื่อพิเศษว่า เมทริกซ์ และบางครั้งเราอาจพอเข้าใจกับอาเรย์สามมิติ แต่ถ้าเป็นอาเรย์ที่มีมิติมากกว่านั้นมักจะทำให้เกิดความสับสนมากกว่าความเข้าใจ ลองดูตัวอย่างต่อไปนี้น่าจะทำให้เข้าใจอาเรย์ได้มากขึ้น

```
> x <- 1:20
> x
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18
 [19] 19 20
> class(x)
```

```
[1] "integer"
> dim(x) <- c(5,4)
> x
     [,1] [,2] [,3] [,4]
[1,]    1    6   11   16
[2,]    2    7   12   17
[3,]    3    8   13   18
[4,]    4    9   14   19
[5,]    5   10   15   20
> class(x)
[1] "matrix"
```

ในบรรทัดแรกเป็นการสร้างเวกเตอร์ตัวเลขที่มีค่าตั้งแต่ 1 ถึง 20 ขึ้น ซึ่งตอนนี้ `x` จะมีคลาส (class หรือวิธีเก็บข้อมูล) เป็น integer หรือเวกเตอร์ของเลขจำนวนเต็ม จากนั้นเราทำให้เวกเตอร์ `x` มีมิติเป็น 5x4 หรือมี 5 แถว และ 4 สดมภ์หรือคอลัมน์ด้วยคำสั่ง `dim(x)` และใส่เวกเตอร์ระบุมิติไว้ด้านท้ายของเครื่องหมาย `<-` โดยระบุจำนวนแถวไว้หน้าและจำนวนสดมภ์ไว้เป็นลำดับที่สอง เมื่อเราเรียกดู `x` อีกครั้งจะเห็นว่า **R** แสดง `x` ในลักษณะที่เป็นสองมิติ คือมี 5 แถวและ 4 สดมภ์ และตอนนี้ถ้าเราเรียกดูคลาสอีกครั้ง **R** บอกว่าขณะนี้ `x` ถูกเก็บข้อมูลแบบ matrix

เนื่องจากการจัดเรียงข้อมูลจะจัดในแนวแถวก่อน ข้อมูลจึงเรียงจาก 1 ถึง 5 ในสดมภ์แรก และ 6 ถึง 10 ในสดมภ์ที่สอง และต่อไปตามลำดับ แต่ถ้าเราต้องการให้มีการเรียงในแนวสดมภ์ก่อน คือเรียง 1 ถึง 4 ในแถวบนสุด และต่อๆ ไป จะต้องใช้ฟังก์ชัน `t()` หรือ transpose ช่วย โดยทำดังนี้

```
> x <- 1:20
> dim(x) <- c(4,5)
> x
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    5    9   13   17
[2,]    2    6   10   14   18
[3,]    3    7   11   15   19
[4,]    4    8   12   16   20
> x <- t(x)
> x
     [,1] [,2] [,3] [,4]
[1,]    1    2    3    4
[2,]    5    6    7    8
[3,]    9   10   11   12
[4,]   13   14   15   16
[5,]   17   18   19   20
```

จะเห็นว่าในขั้นแรกเราสร้างอาเรย์ที่มี 4 แถว 5 สดมภ์ก่อน แล้วจึงพลิกแถวเป็นสดมภ์และสดมภ์เป็นแถวด้วยฟังก์ชัน `t()` นั่นเอง

ดังที่กล่าวข้างต้นในเรื่องเวกเตอร์แล้วว่าเครื่องหมาย [] ใช้สำหรับระบุตำแหน่งในวัตถุใดๆ ที่มีหน่วยย่อย เราสามารถเรียกดูค่าที่อยู่ในตำแหน่งแถวที่ 4 สดมภ์ที่ 3 ได้โดยการระบุตำแหน่งเป็น `x[4,3]` และหากต้องการดูค่าทุกค่าในแถวที่ 4 ก็ระบุเป็น `x[4,]` และหากต้องการดูทุกค่าในสดมภ์ที่ 3 ก็เขียนเป็น `x[,3]` ซึ่งจะได้ผลดังนี้

```
> x[4,3]
```

```
[1] 15
> x[4,]
[1] 13 14 15 16
> x[,3]
[1]  3  7 11 15 19
```

เราอาจจัดให้เวกเตอร์เรียงเป็นอาเรย์สามมิติหรือมากกว่านั้นก็ได้ แต่การแสดงผลจะซับซ้อนและเข้าใจยาก ดังตัวอย่างต่อไปนี้จะจัดให้ x เรียงเป็น 5 แถว 3 สดมภ์ และมี 2 ชั้น (stratum)

```
> x <- 1:30
> dim(x) <- c(5,3,2)
> x
, , 1

     [,1] [,2] [,3]
[1,]    1    6   11
[2,]    2    7   12
[3,]    3    8   13
[4,]    4    9   14
[5,]    5   10   15

, , 2

     [,1] [,2] [,3]
[1,]   16   21   26
[2,]   17   22   27
[3,]   18   23   28
[4,]   19   24   29
[5,]   20   25   30
```

ขอให้สังเกตว่าผลคูณของเลขจำนวนมิติที่จะจัดได้ จะต้องเท่ากับจำนวนหน่วยย่อย (หรือความยาว) ของเวกเตอร์ตั้งต้น ในที่นี้ 5x3x2 ได้เท่ากับ 30

อาเรย์เป็นพื้นฐานของวัตถุชนิด table หรือตาราง นั่นเอง แต่จะเห็นว่าผู้ใช้ทั่วไปคงไม่ชอบใช้อาเรย์ที่มีมากกว่าสองมิติเพราะดู และเข้าใจยาก แต่จะคุ้นเคยเป็นอย่างมากกับอาเรย์สองมิติหรือเมทริกซ์มากกว่า

> **หมายเหตุ** เราสามารถเปลี่ยนอาเรย์บให้กลับเป็นเวกเตอร์มิติเดียวได้โดยการใช้ฟังก์ชัน `c()` ดังนี้ `x <- c(x)` ขอให้ทดลองดูด้วยตัวเองกับ `x` ในตัวอย่างข้างต้น

เรามาดูอีกวิธีหนึ่งในการสร้างเมทริกซ์โดยการเชื่อมเวกเตอร์ที่มีความยาวเท่ากันเข้าด้วยกัน ดังนี้

```
> y <- 1:10
> z <- log(1:10)
> m <- cbind(y,z)
```

```
> m
       y         z
 [1,]  1 0.0000000
 [2,]  2 0.6931472
 [3,]  3 1.0986123
 [4,]  4 1.3862944
 [5,]  5 1.6094379
 [6,]  6 1.7917595
 [7,]  7 1.9459101
 [8,]  8 2.0794415
 [9,]  9 2.1972246
[10,] 10 2.3025851
> class(m)
[1] "matrix"
```

จะเห็นว่าเราผนวกเวกเตอร์สองเวกเตอร์ y และ z เข้าด้วยกันเป็นวัตถุ m ด้วยฟังก์ชัน cbind() ซึ่งเป็นคำสั่งให้ผนวกเวกเตอร์เข้าด้วยกันโดยเชื่อมกันหรือเรียงลงมาตามแนวสดมภ์ เมื่อตรวจดูคลาสพบว่า m เป็นเมทริกซ์นั่นเอง

ผู้ใช้ทั่วไปคงรู้สึกว่าการสร้างเมทริกซ์โดยวิธีนี้เข้าใจง่ายกว่า และมีประโยชน์ในการใช้มากกว่า เพราะมีความคล้ายคลึงกับชุดข้อมูลที่เรามีมากกว่านั่นเอง และนั่นจะนำไปสู่วัตถุอีกชนิดหนึ่งที่สำคัญคือ กรอบข้อมูล หรือ data frame นั่นเอง

ข้อจำกัดของอาเรย์และเมทริกซ์ก็เช่นเดียวกับเวกเตอร์คือ ข้อมูลภายในต้องเป็นชนิดเดียวกันทั้งหมด ดังเช่นเราไม่สามารถให้สดมภ์ที่ 1 และ 2 เป็นชื่อกับนามสกุล ขณะที่สดมภ์อื่นๆ เป็นตัวเลขข้อมูลของคนเหล่านั้นได้ เมื่อมีหน่วยใดในแถวลำดับเป็นข้อความหรือตัวอักษร จะทำให้แถวลำดับหรือเมทริกซ์นั้นทั้งหมดเป็นชนิดตัวอักษรด้วย จึงมีวัตถุอีกกลุ่มหนึ่งที่ใช้ทำงานในลักษณะที่มีบางสดมภ์เป็นชนิดตัวอักษร และบางสดมภ์เป็นตัวเลขเช่นนี้ ซึ่งต่อไปเราจะใช้งานวัตถุชนิดนี้เป็นพื้นฐานที่สำคัญในการทำงานทางสถิติต่อไปนั่นเอง

ลิสต์และกรอบข้อมูล

ลิสต์ (list) เป็นวัตถุที่ประกอบด้วยวัตถุอื่นๆ อยู่ภายใน ซึ่งอาจเป็นลิสต์เองก็ได้ โดยพื้นฐานแล้วลิสต์ถูกสร้างจากวัตถุพื้นฐานเช่นเวกเตอร์นั่นเอง ดังตัวอย่าง

```
> names <- c("Mary","Susan","Christina")
> y <- 1:10
> z <- log(5:1)
> lst <- list(names,y,z)
> lst
[[1]]
[1] "Mary"      "Susan"     "Christina"

[[2]]
 [1]  1  2  3  4  5  6  7  8  9 10

[[3]]
[1] 1.6094379 1.3862944 1.0986123 0.6931472 0.0000000
```

จะเห็นว่าเราสามารถผสมเวกเตอร์ที่มีความยาวไม่เท่ากันเข้าเป็นลิสต์ได้ โดยใช้ฟังก์ชัน `list()` หรือ `as.list()` โดยทั่วไปแล้วผู้ใช้ระดับพื้นฐานมักไม่มีโอกาสใช้ลิสต์ในการทำงานทางสถิตินัก แต่จะมีประโยชน์มากกับผู้ใช้ที่ต้องการเขียนฟังก์ชันใช้เอง เนื่องจากเป็นวัตถุที่สำคัญในการส่งข้อมูลออกจากฟังก์ชัน

ย้อนกลับไปดูการสร้างเมทริกซ์แบบที่สอง ที่สร้างมาจากการผนวกเวกเตอร์เข้าด้วยกันทางด้านสดมภ์ เราสามารถเปลี่ยนให้เมทริกซ์ที่สร้างโดยวิธีนี้เป็นวัตถุชนิดกรอบข้อมูลได้โดยใช้ฟังก์ชัน `data.frame()` แต่ลองดูตัวอย่างต่อไปนี้ จะเห็นว่าเราสามารถใช้คำสั่ง `data.frame()` เพื่อเชื่อมเวกเตอร์เข้าด้วยกันได้โดยตรง

```
> x <- letters[1:10]
> y <- 1:10
> z <- log(1:10)
> dat <- data.frame(x,y,z)
> dat
   x  y         z
1  a  1 0.0000000
2  b  2 0.6931472
3  c  3 1.0986123
4  d  4 1.3862944
5  e  5 1.6094379
6  f  6 1.7917595
7  g  7 1.9459101
8  h  8 2.0794415
9  i  9 2.1972246
10 j 10 2.3025851
```

วัตถุชนิดกรอบข้อมูลจะเป็นพื้นฐานในการทำงานคำนวณทางสถิติต่อไปในหนังสือนี้ จึงเป็นวัตถุที่เราจะให้ความสนใจเป็นพิเศษ

ในการอ้างอิงตำแหน่งในเมทริกซ์ เราใช้เครื่องหมาย [] ดังกล่าวข้างต้นแล้ว ในกรอบข้อมูลเราก็สามารถใช้เครื่องหมาย [] ได้ แต่มักไม่นิยมกัน แต่จะใช้เครื่องหมาย $ เพื่อระบุสดมภ์ และใช้ [] เพื่อระบุแถวในสดมภ์นั้นๆ แทน ดังนี้

```
> dat$x
 [1] a b c d e f g h i j
Levels: a b c d e f g h i j
> class(dat$x)
[1] "factor"
> dat$y
 [1]  1  2  3  4  5  6  7  8  9 10
> dat$z[10]
[1] 2.302585
```

นั่นคือเรานิยมเรียกชื่อของสดมภ์เสมือนหนึ่งว่าเป็นเวกเตอร์หนึ่ง แล้วจึงระบุตำแหน่งของหน่วยย่อยในสดมภ์ด้วยเครื่องหมาย [] นั่นเอง

ดังนั้นอาจสรุปได้ว่ากรอบข้อมูลก็คล้ายกับเมทริกซ์นั่นเอง แต่กรอบข้อมูลจะมีความคล่องตัวในการคำนวณมากกว่า ตรงที่ **R** จะพยายามเก็บข้อมูลในรูปแบบตัวเลข ซึ่งสามารถนำมา

คำนวณได้เลย และหากมีข้อมูลในสดมภ์ใดเป็นข้อความหรือตัวอักษร **R** จะเก็บเป็นแฟกเตอร์โดยปริยาย ซึ่งทำให้สามารถนำมาคำนวณทางสถิติได้ ยกเว้นว่าเราจะระบุให้เก็บเป็นเวกเตอร์ตัวอักษร หรือในการนำข้อมูลเข้าโดยฟังก์ชันอ่านแฟ้มข้อมูลบางชนิด ค่าปริยายอาจเป็นเวกเตอร์ตัวอักษรแทนที่จะเป็นแฟกเตอร์ก็ได้

attach และ detach

ในบางครั้งเมื่อทำงานกับกรอบข้อมูลเพียงกรอบข้อมูลเดียว การที่ต้องพิมพ์ชื่อกรอบข้อมูลทุกครั้งเมื่อเรียกชื่อเวกเตอร์หรือตัวแปรภายในก็เป็นเรื่องยุ่งยาก จึงมีวิธีหนึ่งที่จะเลี่ยงการเรียกชื่อกรอบข้อมูลซ้ำๆ เช่นนี้คือการใช้คำสั่ง `attach()` และ `detach()`

```
> rm(x,y,z)
> attach(dat)
> x
 [1] a b c d e f g h i j
Levels: a b c d e f g h i j
> y
 [1]  1  2  3  4  5  6  7  8  9 10
> z
 [1]  0.0000000  0.6931472  1.0986123  1.3862944  1.6094379  1.7917595
1.9459101 2.0794415
 [9] 2.1972246 2.3025851
> table(x,y)
   y
x   1 2 3 4 5 6 7 8 9 10
  a 1 0 0 0 0 0 0 0 0  0
  b 0 1 0 0 0 0 0 0 0  0
  c 0 0 1 0 0 0 0 0 0  0
  d 0 0 0 1 0 0 0 0 0  0
  e 0 0 0 0 1 0 0 0 0  0
  f 0 0 0 0 0 1 0 0 0  0
  g 0 0 0 0 0 0 1 0 0  0
  h 0 0 0 0 0 0 0 1 0  0
  i 0 0 0 0 0 0 0 0 1  0
  j 0 0 0 0 0 0 0 0 0  1
> detach(dat)
```

บรรทัดแรกสุดเป็นการใช้ฟังก์ชัน `rm()` เพื่อลบวัตถุชื่อ `x`, `y` และ `z` ที่อยู่ในหน่วยความจำ ซึ่งได้สร้างไว้ก่อนหน้านี้แล้วข้างต้น บรรทัดต่อมา `attach(dat)` เป็นการตรึงกรอบข้อมูล `dat` เข้ากับเส้นทางค้นหา เมื่อกรอบข้อมูล `dat` ได้ถูก `attach` แล้ว ต่อไปเราจะสามารถเรียกชื่อเวกเตอร์หรือตัวแปรภายใน `dat` ได้โดยไม่ต้องเขียน `dat$` นำหน้าอีกแต่อย่างใด เมื่อทำงานกับกรอบข้อมูลนั้นเสร็จแล้ว จึงเลิกการตรึงกรอบข้อมูลด้วยคำสั่ง `detach(dat)`

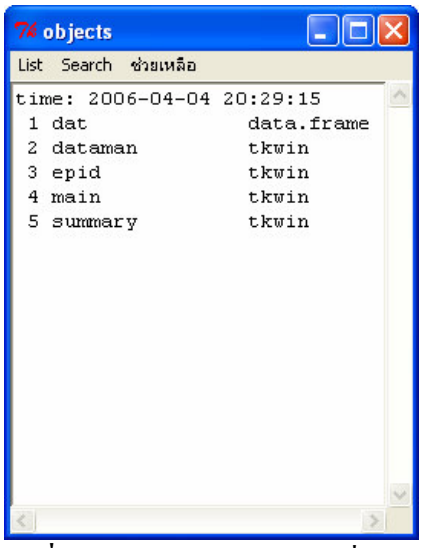
หมายเหตุ เมื่อตรึงข้อมูลเข้ากับเส้นทางค้นหาด้วยคำสั่ง `attach()` แล้ว อย่าเปลี่ยนแปลงข้อมูลในกรอบข้อมูลนั้นอีก หากจำเป็นต้องเปลี่ยนแปลงข้อมูลภายใน จะต้อง `detach()`

กรอบข้อมูลนั้นก่อน ทั้งนี้เพราะเมื่อ `attach()` กรอบข้อมูลใดแล้ว จะมีข้อมูลซ้ำกันอยู่สองชุด ชุดหนึ่งอยู่ในหน่วยความจำปกติ และอีกชุดหนึ่งอยู่ในหน่วยความจำส่วนพิเศษที่อยู่ในเส้นทางค้นหาของ **R** การเปลี่ยนแปลงข้อมูลใดๆ จะเกิดขึ้นเฉพาะกับชุดที่อยู่ในหน่วยความจำปกติเท่านั้น แต่เมื่อเรียกใช้งาน **R** จะไปเรียกใช้ข้อมูลในเส้นทางค้นหา ทำให้เสมือนหนึ่งว่าข้อมูลไม่ได้เปลี่ยนไปแต่อย่างใด ขณะที่ผู้ใช้คิดว่าข้อมูลนั้นได้เปลี่ยนไปแล้วและมักไม่ได้ตรวจสอบว่าข้อมูลสองชุดนั้นไม่ตรงกัน แต่หากได้ทำตามขั้นตอนต่อไปนี้คือ 1. `detach()` กรอบข้อมูล 2. เปลี่ยนแปลงข้อมูล และ 3. `attach()` กรอบข้อมูลนั้นซ้ำอีกครั้ง ข้อมูลทั้งสองส่วนก็จะตรงกัน

การเรียกดูวัตถุในหน่วยความจำด้วยหน้าต่าง ice.objects

ที่ผ่านมาทั้งหมดนั้น ยังไม่ได้กล่าวถึงวิธีการใช้งาน ice แต่อย่างใดเลย ในหัวข้อนี้จะเป็นครั้งแรกที่เราจะได้ใช้ ice กันเสียที โมดุลหรือชิ้นส่วนแรกที่จะแนะนำในที่นี้คือ ice.objects ซึ่งเป็นชิ้นส่วนที่ใช้สำหรับเรียกดูวัตถุในหน่วยความจำและเส้นทางค้นหานั่นเอง

หากยังไม่ได้ติดตั้งสภาพแวดล้อม **R-ICE** ขอให้กลับไปดูในบทที่ 3 เรื่อง **สภาพแวดล้อม R-ICE** ซึ่งเมื่อติดตั้งถูกต้อง เมื่อเรียกสภาพแวดล้อม **ICE** จะมีหน้าต่าง ice.objects ปรากฏขึ้น ดังรูปที่ 4.2 ซึ่งหากเมื่อเลือกตัวเลือก “List” ด้านบนซ้ายสุด ก็จะมีข้อความดังรูป



รูปที่ 4.2 หน้าต่าง ice.objects เมื่อเลือกตัวเลือก “List”

ซึ่งมีผลเท่ากับพิมพ์คำสั่ง `ls()` บนหน้าจอ R console นั่นเอง ดังข้างล่าง

```
> ls()
[1] "dat"      "dataman" "epid"    "main"    "summary"
```

นั่นคือดูว่ามีวัตถุอะไรบ้างอยู่ในหน่วยความจำของ **R** แต่มีข้อดีกว่าอย่างชัดเจนตรงที่ ice.objects ได้บอกชนิดของวัตถุนั้นให้ด้วยเลย ซึ่งคำสั่ง `ls()` ไม่ได้บอก ดังนี้

และหากเลือกตัวเลือก “Search” จะได้ผลดังรูปที่ 4.3

รูปที่ 4.3 หน้าต่าง ice.objects เมื่อเลือกตัวเลือก “Search”

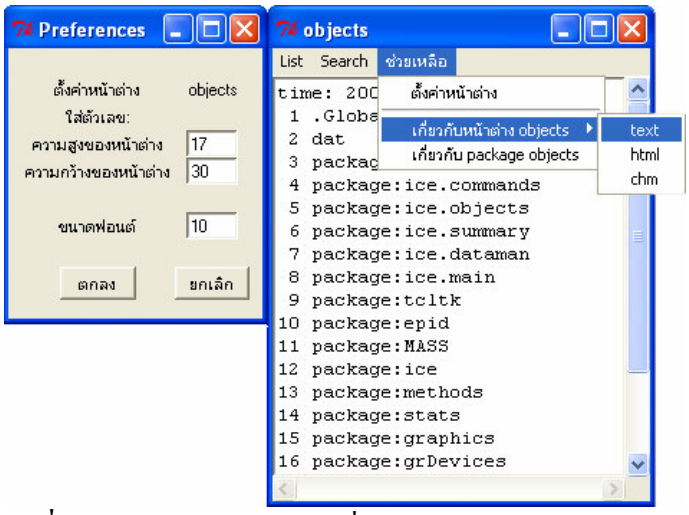
มีผลเท่ากับพิมพ์คำสั่ง `search()` บนหน้าจอ R console นั่นเอง ซึ่งจะได้ผลดังข้างล่าง

```
> search()
 [1] ".GlobalEnv"            "dat"
 [3] "package:ice.epid"      "package:ice.commands"
 [5] "package:ice.objects"   "package:ice.summary"
 [7] "package:ice.dataman"   "package:ice.main"
 [9] "package:tcltk"         "package:epid"
[11] "package:MASS"          "package:ice"
[13] "package:methods"       "package:stats"
[15] "package:graphics"      "package:grDevices"
[17] "package:utils"         "package:datasets"
[19] "Autoloads"             "package:base"
```

นั่นคือดูว่ามีวัตถุอะไรบ้างอยู่ในเส้นทางค้นหาของ **R** ซึ่งในกรณีนี้จะพบว่ามีกรอบข้อมูล `dat` อยู่ด้วย แต่ไม่ใช่กรอบข้อมูลเดียวกับ `dat` ที่เห็นเมื่อเลือกตัวเลือก “List” เพียงแต่ขณะนี้ทั้งสองมีข้อมูลเหมือนกันทุกประการ แต่หากเราเปลี่ยนแปลงข้อมูลใน `dat` ตัวที่ถูกเปลี่ยนคือ `dat` ที่ `ls()` ได้ ไม่ใช่ `dat` ที่ `search()` ได้ แต่หากเรียกดูข้อมูลภายใน **R** จะเรียกดู `dat` ในเส้นทางค้นหา ซึ่งข้อมูลยังคงเดิม ดังที่กล่าวแล้วในหัวข้อข้างต้น

เมนู “ช่วยเหลือ” มีรายการตัวเลือกย่อยอีกสามตัวเลือกดังรูปที่ 4.4 เมื่อเลือกตัวเลือก “ตั้งค่าหน้าต่าง” จะปรากฏหน้าต่างย่อย Preferences ดังหน้าต่างย่อยในรูปที่ 4.4 ให้เลือกปรับขนาด

ความสูง ความกว้าง และขนาดฟอนต์ของหน้าต่าง objects ได้ อย่างไรก็ตามยังมีข้อจำกัดที่เมื่อเปลี่ยนค่านี้แล้วหน้าต่างเก่าจะถูกลบไปและสร้างหน้าต่างขึ้นมาใหม่ ซึ่งผู้ใช้จะต้องเลื่อนไปวางในที่ต้องการอีกครั้ง

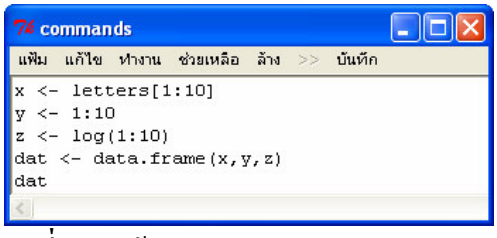


รูปที่ 4.4 หน้าต่าง ice.objects เมื่อเลือกตัวเลือก “ช่วยเหลือ”

ส่วนตัวเลือก “เกี่ยวกับหน้าต่าง objects” เป็นข้อความอธิบายเกี่ยวกับการใช้งาน package ice.objects ส่วนตัวเลือก “เกี่ยวกับ package objects” เป็นข้อความอธิบายเกี่ยวกับ package ice.objects อย่างสั้น เช่น รุ่นของแพคเกจ วันที่สร้าง และอื่นๆ คือเป็น about ice.objects นั่นเอง

การพิมพ์คำสั่งด้วยหน้าต่าง ice.commands

ในการพิมพ์คำสั่งเพื่อทำงานในสภาพแวดล้อม **R** นั้น บนระบบปฏิบัติการวินโดวส์และแมคอินทอช OSX จะมีหน้าต่าง RGUI มาให้ ส่วนบนระบบปฏิบัติการลินุกซ์จะต้องพิมพ์ในหน้าต่าง terminal การพิมพ์บรรทัดคำสั่งในลักษณะนี้จะพิมพ์ทีละบรรทัด เมื่อกดแป้น Enter แล้ว **R** ก็จะทำงานบรรทัดคำสั่งนั้นทันที มีบางครั้งที่ผู้ใช้อาจต้องการให้ทำงานคำสั่งทีละหลายบรรทัดพร้อมกัน ในกรณีนี้เราสามารถทำได้โดยพิมพ์บรรทัดคำสั่งในหน้าต่าง ice.commands ซึ่งมีหน้าตาดังรูปที่ 4.4



รูปที่ 4.5 หน้าต่าง ice.commands

เราสามารถพิมพ์บรรทัดคำสั่งดังในรูปทีละบรรทัด เมื่อกดปุ่ม Enter ก็จะขึ้นบรรทัดใหม่ให้พิมพ์คำสั่งบรรทัดถัดไปโดยไม่ส่งไปทำงานใน **R** เมื่อพิมพ์จนจบชุดคำสั่งที่ต้องการแล้ว เราจะส่งคำสั่งทั้งหมดไปทำงานใน **R** พร้อมกันทีเดียวโดยเลือกเมนู "ทำงาน" และเลือกตัวเลือก "ทำงานทั้งหมด" บรรทัดคำสั่งทั้งหมดก็จะถูกส่งไปทำงานบน R console พร้อมกันตามต้องการ

เมนูต่างๆ ของ ice.commands มีดังตารางที่ 4.1 ในส่วนเมนู "แฟ้ม" มีรายการดังตารางที่ 4.1 และการทำงานของรายการในหน้าต่าง ice.commands เป็นดังตารางที่ 4.2

ตารางที่ 4.1 โครงสร้างของรายการเมนู ice.commands

แฟ้ม	**แก้ไข**	**ทำงาน**	**ช่วยเหลือ**	**ล้าง**	**บันทึก**
เปิดแฟ้มสคริปต์	ตัด	ทำงานทั้งหมด	ตั้งค่าหน้าต่าง		
บันทึกทั้งหมดลงแฟ้ม	คัดลอก	ทำงานส่วนที่เลือก	เกี่ยวกับหน้าต่าง commands		
บันทึกส่วนที่เลือกลงแฟ้ม	วาง	ทำงานแฟ้มภายนอก…	text		
	ลบ		html		
	เลือกทั้งหมด		chm		
			เกี่ยวกับ package commands		

ตารางที่ 4.2 การทำงานของรายการในหน้าต่าง ice.commands

รายการ	**การทำงาน**
เปิดแฟ้มสคริปต์	เป็นการเปิดแฟ้มสคริปต์คำสั่งของ **R** ที่บันทึกไว้
บันทึกทั้งหมดลงแฟ้ม	บันทึกบรรทัดคำสั่งทั้งหมดในหน้าต่าง commands
บันทึกส่วนที่เลือกลงแฟ้ม	บันทึกบรรทัดคำสั่งเฉพาะส่วนที่เลือกในหน้าต่าง commands
ตัด	ตัดส่วนข้อความในหน้าต่าง commands และบันทึกใน clipboard
คัดลอก	คัดลอกข้อความในหน้าต่าง commands บันทึกใน clipboard
วาง	วางข้อความใน clipboard ลงในหน้าต่าง commands
ลบ	ลบส่วนของข้อความในหน้าต่าง commands
เลือกทั้งหมด	เลือกข้อความในหน้าต่าง commands ทั้งหมด
ทำงานทั้งหมด	ทำงานบรรทัดคำสั่งในหน้าต่าง commands ทั้งหมด
ทำงานส่วนที่เลือก	ทำงานบรรทัดคำสั่งในหน้าต่าง commands ส่วนที่เลือกไว้
ทำงานแฟ้มภายนอก...	ทำงานแฟ้มสคริปต์คำสั่งของ **R** ที่บันทึกไว้
ตั้งค่าหน้าต่าง	ปรับขนาดความสูง ความกว้าง และขนาดฟอนต์ของหน้าต่าง commands
เกี่ยวกับหน้าต่าง commands	ข้อความอธิบายเกี่ยวกับการใช้งาน package ice.commands
เกี่ยวกับ package commands	ข้อความอธิบายเกี่ยวกับ package ice.commands อย่างสั้น

หมายเหตุ ขณะนี้ตัวเลือก “ตัด”’ “คัดลอก ” “วาง” “ลบ” และ “เลือกทั้งหมด” สามารถใช้ short cut คือ ctrl-x, ctrl-c, ctrl-v, delete และ ctrl-aได้ แต่เนื่องจากข้อจำกัดของแพคเกจ tcltk จึงไม่สามารถแสดง short cut ไว้บนเมนูได้

บทที่ 5 การนำข้อมูลเข้าสู่สภาพแวดล้อม R และการบันทึกแฟ้มข้อมูล

การกรอกข้อมูลโดยใช้ฟังก์ชันภายใน **R**

การนำเข้าข้อมูลจากแฟ้มข้อมูลชนิดต่างๆ

- แฟ้มข้อมูลแบบ text ในรูปแบบต่างๆ
- แฟ้มข้อมูลของโปรแกรมสถิติอื่นๆ เช่น **EpiData**, **Stata**, **SPSS**
- แฟ้มข้อมูล ODBC

การบันทึกแฟ้มข้อมูลแบบต่างๆ

- แฟ้มข้อมูลแบบ text ในรูปแบบต่างๆ
- แฟ้มข้อมูลของโปรแกรม **Stata**

การบันทึกสภาพแวดล้อมของ **R** ลงแฟ้มและการเรียกกลับคืน

แฟ้มสคริปต์คำสั่ง **R**

การใช้โมดุล ice.main ในการทำงานกับแฟ้มข้อมูลและสคริปต์

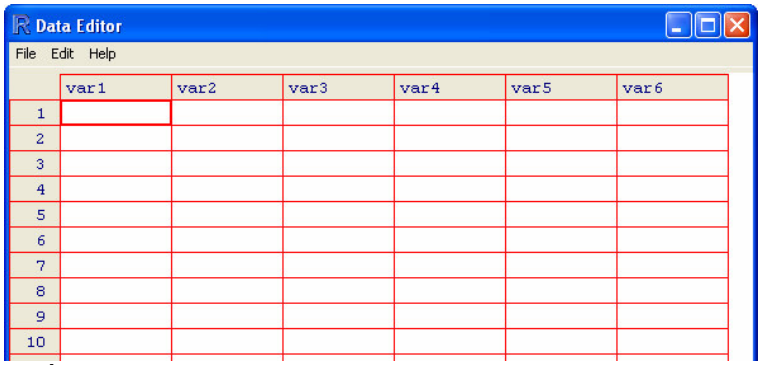
หลังจากได้ทราบหลักการทำงานเบื้องต้นของ **R** และการออกคำสั่งเพื่อทำงานแล้ว ก่อนที่จะทำงานวิเคราะห์ข้อมูลต่อไป ยังมีอีกส่วนหนึ่งที่สำคัญคือตัวข้อมูลที่จะวิเคราะห์นั่นเอง ดังนั้นในบทนี้จะได้กล่าวถึงการนำข้อมูลเข้าสู่สภาพแวดล้อม **R** เพื่อการวิเคราะห์ทางสถิติต่อไป

การกรอกข้อมูลโดยใช้ฟังก์ชันภายใน R

ในการใช้ **R** เพื่อการคำนวณทางสถิติที่มีข้อมูลจำนวนมาก เรามักไม่กรอกข้อมูลลงในโปรแกรม **R** โดยตรง แต่มักจะกรอกข้อมูลโดยใช้โปรแกรมสำหรับกรอกข้อมูล เช่น **EpiData** หรือโปรแกรมฐานข้อมูลอื่นๆ ซึ่งมีความสามารถในการตรวจสอบความถูกต้องของข้อมูลได้ดี นอกจากนี้ยังอาจใช้โปรแกรมตารางคำนวณ เช่น **Excel** หรือโปรแกรมของ **OpenOffice** ในการกรอกข้อมูลก็ได้ อย่างไรก็ตามสำหรับข้อมูลขนาดเล็กแล้ว **R** มีฟังก์ชันในการสร้างกรอบข้อมูลในรูปแบบคล้ายตารางคำนวณให้ด้วย นั่นก็คือฟังก์ชัน `edit()` ซึ่งในการออกคำสั่งนี้ จะต้องเขียนในรูปแบบ `edit(data.frame())` เพื่อบอก **R** ว่าเรากำลังจะสร้างกรอบข้อมูล และจะต้องส่งผลลัพธ์ให้กับวัตถุ ดังตัวอย่าง

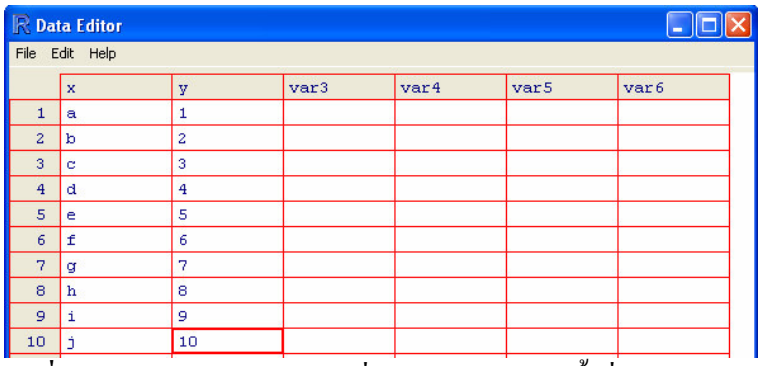
```
> dat <- edit(data.frame())
```

เมื่อออกคำสั่งนี้แล้ว จะได้หน้าต่าง Data Editor ในลักษณะคล้ายตารางคำนวณดังรูปที่ 5.1



รูปที่ 5.1 หน้าต่าง Data Editor ของคำสั่ง `edit(data.frame())`

เราสามารถกรอกข้อมูลลงในตารางนี้ได้ โดยที่แต่ละแถวในแนวนอนจะเป็นรายการของตัวอย่างหรือผู้ป่วยแต่ละราย และแต่ละสดมภ์ในแนวตั้งจะเป็นตัวแปร หรือก็คือเวกเตอร์ในภาษา **R** นั่นเอง นอกจากนี้ยังสามารถตั้งชื่อตัวแปรหรือเวกเตอร์ได้โดยการเลือกที่หัวข้อ `var1` ซึ่งจะปรากฏหน้าต่างเล็กๆ ให้ใส่ชื่ออื่นแทนชื่อ `var1` ที่โปรแกรมได้ตั้งไว้ให้แต่แรกได้ ดังตัวอย่างในรูปที่ 5.2 ได้กรอกข้อมูล `a, b, …, j` ในสดมภ์แรก และ `1, 2, …, 10` ในสดมภ์ที่สอง และตั้งชื่อตัวแปรตัวแรกว่า `x` และตัวแปรตัวที่สองว่า `y`



	x	y	var3	var4	var5	var6
1	a	1				
2	b	2				
3	c	3				
4	d	4				
5	e	5				
6	f	6				
7	g	7				
8	h	8				
9	i	9				
10	j	10				

รูปที่ 5.2 หน้าต่าง Data Editor เมื่อกรอกข้อมูลและตั้งชื่อตัวแปรแล้ว

การบันทึกข้อมูลทำได้โดยกดที่เครื่องหมายกากบาทด้านบนขวาของหน้าต่างหรือเลือกเมนู File >> Close ซึ่งเมื่อปิดหน้าต่างแล้ว ข้อมูลที่กรอกเข้าไปในตารางจะถูกบันทึกไว้ในวัตถุชนิดกรอบข้อมูลตามชื่อที่ตั้งไว้นั่นเอง ซึ่งเราอาจตรวจสอบดูได้ว่าข้อมูลที่กรอกเข้าไปนั้นมีอยู่จริงโดยการพิมพ์ชื่อกรอบข้อมูลเพื่อเรียกดูข้อมูลภายในดังนี้

```
> dat <- edit(data.frame(dat))
> dat
   x  y
1  a  1
2  b  2
3  c  3
4  d  4
5  e  5
6  f  6
7  g  7
8  h  8
9  i  9
10 j 10
```

หมายเหตุ หากออกคำสั่ง `edit(data.frame())` โดยไม่กำหนดชื่อให้ ข้อมูลที่กรอกจะสูญหายไปทั้งหมด จึงเป็นเรื่องสำคัญที่ต้องไม่ลืมตั้งชื่อให้กับกรอบข้อมูลที่สร้างขึ้นนี้

การนำเข้าข้อมูลจากแฟ้มข้อมูลชนิดต่างๆ

ข้อมูลที่สามารถนำมาใช้คำนวณทางสถิติในสภาพแวดล้อม **R** อาจสร้างขึ้นจากโปรแกรมต่างๆ ได้หลายโปรแกรม เช่น text editor ดังเช่นที่แนะนำไปแล้วในบทที่ 2 เรื่อง **สภาพแวดล้อมเสริมการทำงาน** หรืออาจใช้โปรแกรมตารางคำนวณ เช่น **Excel** หรือโปรแกรมตารางคำนวณของ **OpenOffice** ก็ได้ หรืออาจสร้างจากโปรแกรมที่สร้างขึ้นเพื่อใช้กรอกข้อมูลโดยตรง เช่น **EpiData**

หรือ **EpiInfo** หรือโปรแกรมฐานข้อมูลก็ได้ หรือแม้แต่แฟ้มข้อมูลของโปรแกรมคำนวณทางสถิติบางโปรแกรมที่นิยมใช้กันเช่น **SPSS** หรือ **Stata** ก็สามารถนำเข้าไปคำนวณในสภาพแวดล้อม **R** ได้ทั้งสิ้น

แฟ้มข้อมูลแบบ text ในรูปแบบต่างๆ

รูปแบบของแฟ้มข้อมูลข้อความอย่างธรรมดาที่สุดคือรูปแบบของ text ซึ่งก็ยังมีรูปแบบที่แตกต่างกันไป ที่สำคัญมี 4 รูปแบบ คือ แบบ comma separated values (CSV) ซึ่งจะใช้เครื่องหมายจุลภาค (,) คั่นแยกระหว่างตัวแปรแต่ละสดมภ์ออกจากกัน สำหรับบางโปรแกรม ก็จะใช้เครื่องหมายคำพูด "_" ล้อมรอบด้วยเพื่อบอกขอบเขตของข้อมูลให้ชัดเจนยิ่งขึ้น แต่บางโปรแกรมจะใส่เครื่องหมายคำพูดเฉพาะเมื่อในข้อความนั้นมีเครื่องหมายจุลภาคอยู่ด้วย ซึ่งอาจทำให้สับสนว่าข้อความมีสองสดมภ์ได้ อีกแบบหนึ่งคือ tab delimited ซึ่งมักจะใช้นามสกุล TXT หรือ TAB ซึ่งแฟ้มเช่นนี้จะใช้เครื่องหมาย tab เพื่อแยกสดมภ์ออกจากกัน ดังนั้นข้อพึงระวังของรูปแบบนี้คือต้องไม่มีเครื่องหมาย tab อยู่ในข้อความที่เป็นเนื้อหานั้นเอง แบบที่สามที่นิยมกันคือคั่นด้วยช่องว่าง ซึ่งอาจจะเป็นช่องว่างหนึ่งช่องหรือมากกว่าก็ได้ และสุดท้ายคือรูปแบบที่มีความกว้างของข้อความคงที่ เพื่อจะให้เข้าใจรูปแบบต่างๆ เหล่านี้ ลองสร้างแฟ้มข้อมูลขนาดเล็กขึ้นมาสักหนึ่งแฟ้มด้วยโปรแกรมตารางคำนวณของ **Excel** หรือ **OpenOffice** ดังนี้

	A	B	C
1	Order	Name	Lastname
2	1	Freddie	Ljungberg
3	2	Joe	Cole
4	3	Steven	Gerrard
5	4	Cristian	Ronaldo
6	5	Ashley	Cole
7	6	David	Bentley
8	7	David	Beckham
9	8	Alan	Smith
10	9	Cesc	Fabregas
11	10	James	McFadden

รูปที่ 5.3 สร้างข้อมูลด้วยโปรแกรมตารางคำนวณ

โดยทั่วไปเรามักจะจัดเรียงข้อมูลให้แต่ละสดมภ์ในแนวตั้งเป็นตัวแปรแต่ละตัว และให้แต่ละแถวในแนวนอนเป็นรายการแต่ละรายการ โดยนิยมให้แถวที่ 1 เป็นชื่อของตัวแปร ในที่นี้ให้สดมภ์ A เป็นเลขลำดับ B เป็นชื่อ และ C เป็นนามสกุลของนักฟุตบอล เมื่อกรอกข้อมูลเสร็จตามรูปแล้ว ให้เลือกรายการ File >> Save As… แล้วลองเลือกรูปแบบเป็น CSV โดยตั้งชื่อว่า football.csv จะได้ข้อความดังนี้ (เปิดด้วย text editor เช่น **NotePad** หรือโปรแกรมที่แนะนำในบทที่ 2 เรื่อง **สภาพแวดล้อมเสริมการทำงาน**)

หากเป็นโปรแกรมตารางคำนวณของ **OpenOffice** ค่าปริยายที่ตั้งไว้จะทำให้ได้ผลดังนี้

```
"Order","Name","Lastname"
1,"Freddie","Ljungberg"
2,"Joe","Cole"
3,"Steven","Gerrard"
4,"Cristian","Ronaldo"
5,"Ashley","Cole"
6,"David","Bentley"
7,"David","Beckham"
8,"Alan","Smith"
9,"Cesc","Fabregas"
10,"James","McFadden"
```

หากเป็น **Excel** จะได้เป็น

```
Order,Name,Lastname
1,Freddie,Ljungberg
2,Joe,Cole
3,Steven,Gerrard
4,Cristian,Ronaldo
5,Ashley,Cole
6,David,Bentley
7,David,Beckham
8,Alan,Smith
9,Cesc,Fabregas
10,James,McFadden
```

นั่นคือแต่ละสดมภ์ถูกคั่นด้วยเครื่องหมายจุลภาค (,) โดยอาจมีเครื่องหมายคำพูด “” ล้อมรอบสดมภ์ที่เป็นข้อความหรือไม่ก็ได้

ลองบันทึกแฟ้มใหม่โดยเลือกตัวเลือกเป็น tab delimited (โปรแกรมตารางคำนวณของ **OpenOffice** ใช้คำว่า {ระยะกั้น}) โดยตั้งชื่อว่า football.tab จะได้ผลดังนี้

```
"Order"     "Name"      "Lastname"
1     "Freddie"   "Ljungberg"
2     "Joe" "Cole"
3     "Steven"    "Gerrard"
4     "Cristian"  "Ronaldo"
5     "Ashley"    "Cole"
6     "David"     "Bentley"
7     "David"     "Beckham"
8     "Alan"      "Smith"
9     "Cesc"      "Fabregas"
10    "James"     "McFadden"
```

และหากเป็น **Excel** จะได้แฟ้มดังนี้

```
Order Name  Lastname
1     Freddie     Ljungberg
2     Joe   Cole
3     Steven      Gerrard
4     Cristian    Ronaldo
5     Ashley      Cole
6     David Bentley
```

```
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

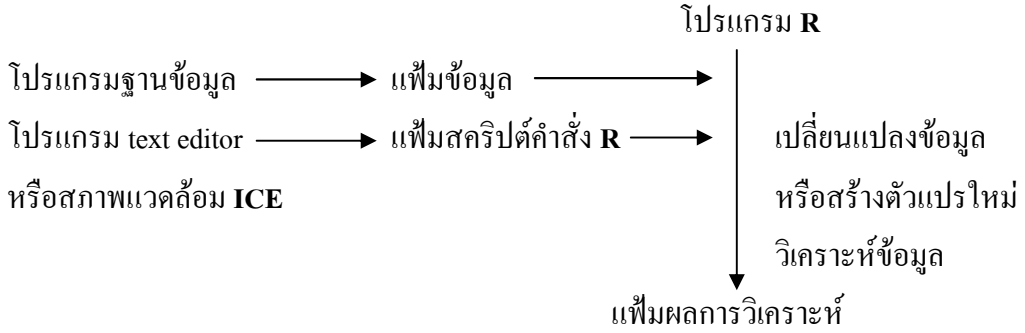
ไม่แนะนำให้เรียกใช้ข้อมูลที่บันทึกในรูปแบบของ ODBC โดยตรง โปรแกรมที่บันทึกข้อมูลแบบ ODBC มีหลายโปรแกรมเช่น **Excel**, **Access** และฐานข้อมูลแบบ SQL เช่นฐานข้อมูลของโปรแกรม **MySQL** การเรียกใช้ข้อมูลในรูปแบบนี้โดยตรงจะต้องใช้ library RODBC และมีขั้นตอนการทำงานที่ซับซ้อนพอสมควรซึ่งจะไม่กล่าวในที่นี้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

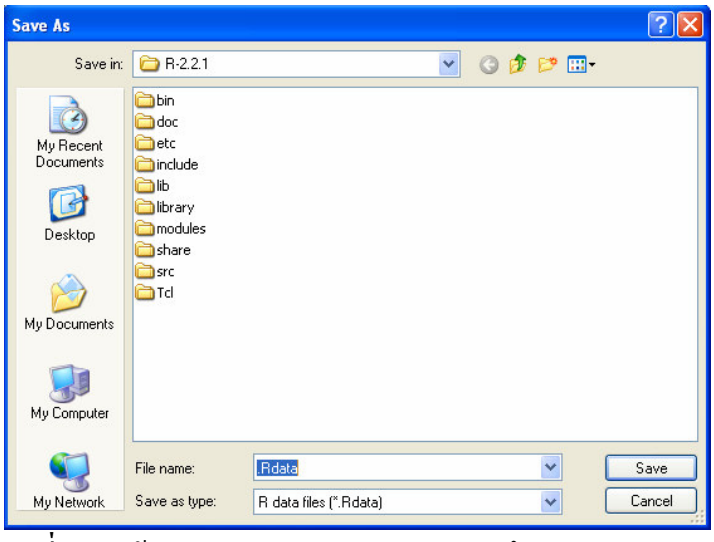
แฟ้ม	**วัตถุ**	**ตั้งค่า**	**ช่วยเหลือ**
ดู working directory	ลิสต์วัตถุในหน่วยความจำ	ตัวเลือกทั่วไป	ตำสั่ง
เปลี่ยน working directory	ค้นหาวัตถุใน search path		do
เรียกใช้ workspace	attach กรอบข้อมูล		fread
บันทึก workspace	detach กรอบข้อมูล		fwrite
เรียกใช้ history	แสดงโครงสร้างของวัตถุ		logg
บันทึก 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

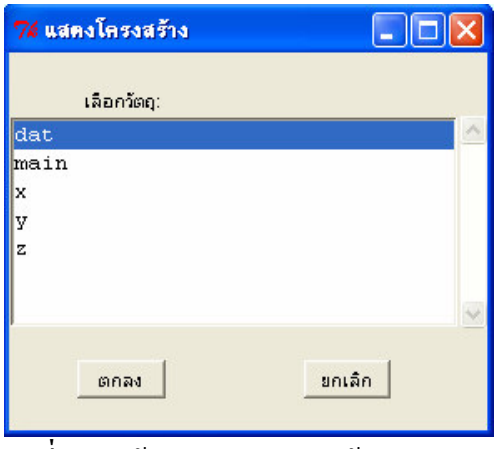
รายการนี้ แต่ควรสร้างแฟ้มสคริปต์และทำงานแฟ้มสคริปต์นั้นตามขั้นตอนที่กล่าวมาแล้วข้างต้นจะดีกว่า

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

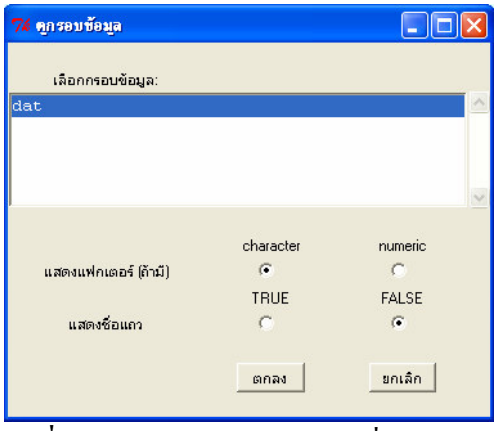
```
ICE>> str(dat); str(x); str(y); str(z))
`data.frame':   10 obs. of  3 variables:
```

```
$ 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

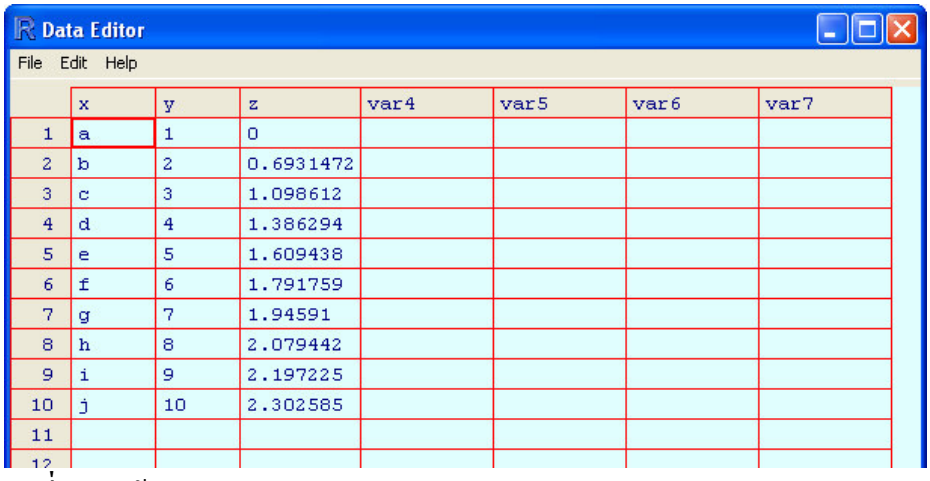
แก้ไขวัตถุ

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



	x	y	z	var4	var5	var6	var7
1	a	1	0				
2	b	2	0.6931472				
3	c	3	1.098612				
4	d	4	1.386294				
5	e	5	1.609438				
6	f	6	1.791759				
7	g	7	1.94591				
8	h	8	2.079442				
9	i	9	2.197225				
10	j	10	2.302585				
11							

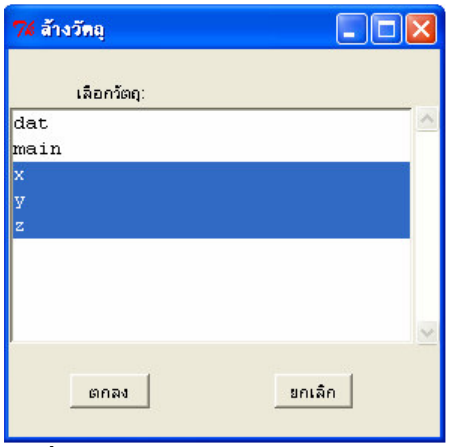
รูปที่ 6.5 หน้าต่าง Data Editor

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

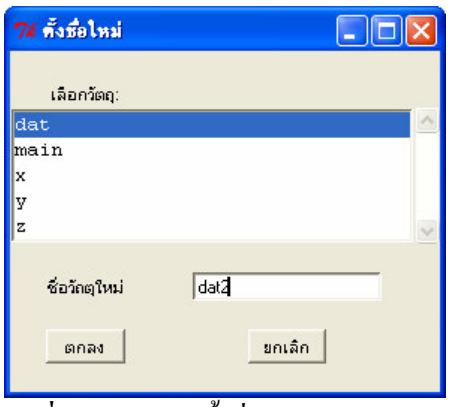
เมื่อเลือกวัตถุหลายรายการพร้อมกัน บรรทัดคำสั่งที่จะปรากฏขึ้นบนหน้าจอ R console จะมีลักษณะดังตัวอย่างข้างล่างนี้ คือลบข้อมูลใน list ของวัตถุนั่นเอง

```
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

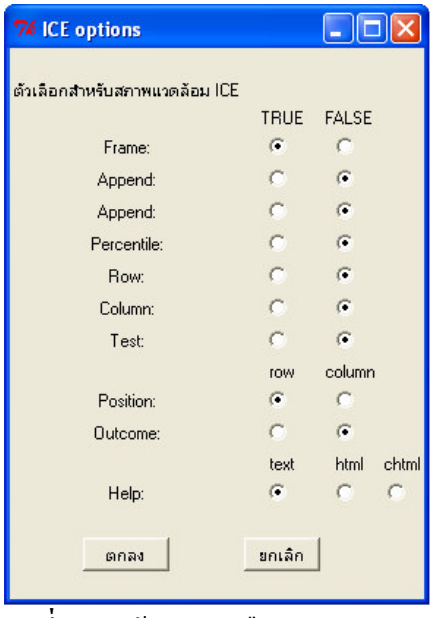
รายการนี้เป็นการเรียกชุดข้อมูลตัวอย่างมาใช้งาน ซึ่งต่อไปในบางหัวข้อเราจะใช้ชุดข้อมูลตัวอย่างใน library ice มาทำงาน ชุดข้อมูลตัวอย่างของ library ice มีดังนี้ `ahcc`, `ahcc2`, `ancdata`, `cca`, `cca9`, `cca.match`, `ccan`, `evans`, `irdata`, `mccdata` และ `vct` คำสั่งที่ปรากฏบน R console ได้แก่

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

ซึ่งฟังก์ชัน `data()` นั้นใช้เรียกชุดข้อมูลใน library เท่านั้น ไม่ได้ใช้อ่านแฟ้มข้อมูลภายนอก

ตั้งค่าตัวเลือก

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



รูปที่ 6.8 หน้าต่างตัวเลือก **ICE**

ความช่วยเหลือ

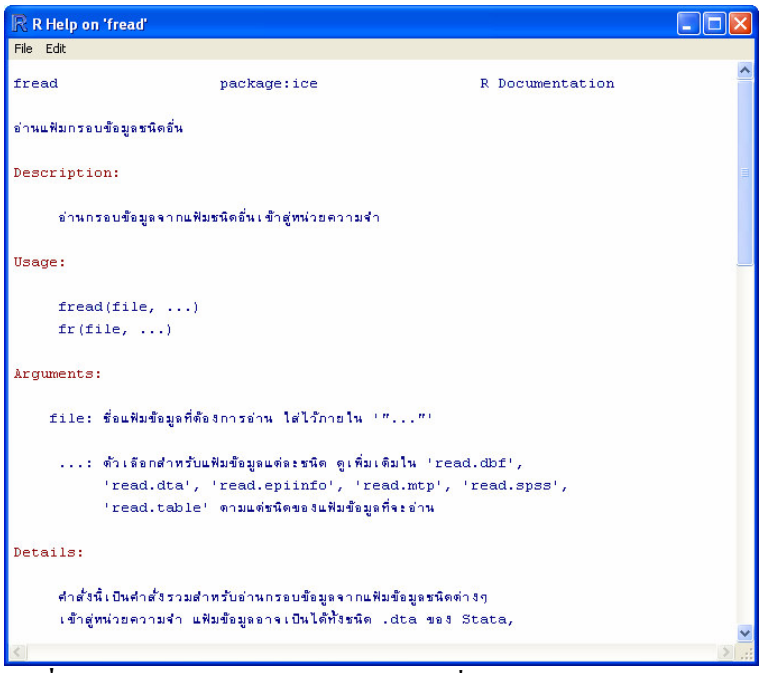
รายการความช่วยเหลือนี้มีรายการย่อยสามรายการได้แก่ “คำสั่ง” ซึ่งเป็นความช่วยเหลือสำหรับคำสั่งต่างๆ “เกี่ยวกับสภาพแวดล้อม **ICE**” และ “เกี่ยวกับ package **ICE**” ดังนี้ ที่สำคัญคือหากติดตั้ง **ICE** ภาษาไทย ก็จะต้องติดตั้งฟอนต์ที่สามารถแสดงตัวอักษรภาษาไทยได้ด้วย สำหรับระบบปฏิบัติการวินโดวส์ แนะนำฟอนต์ที่ดาวน์โหลดได้ฟรีสองฟอนต์คือ 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

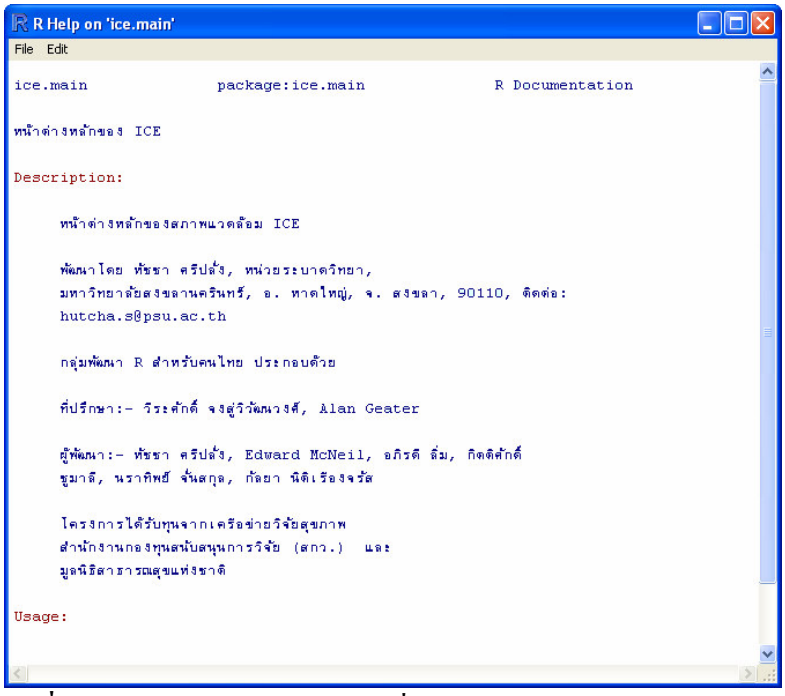


รูปที่ 6.9 หน้าต่างความช่วยเหลือของคำสั่ง `fread()`

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

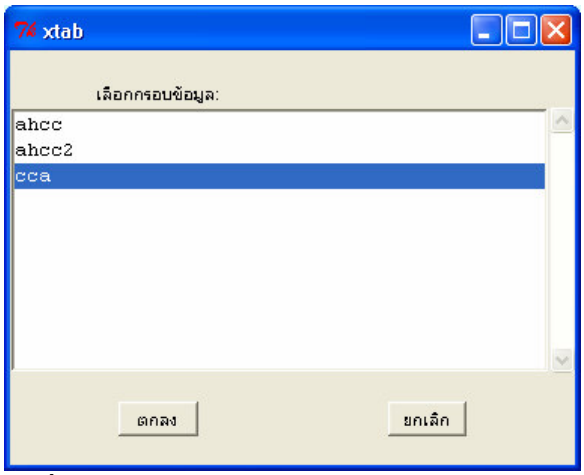
สิ่งที่ฟังก์ชัน `tab()` ต่างไปจากฟังก์ชัน `table()` ในแพคเกจ `base` คือ ฟังก์ชัน `tab()` สามารถเลือกที่จะแสดงตารางร้อยละได้ด้วย และนอกจากนี้ยังเลือกที่จะแสดงจำนวนข้อมูลที่เป็น NA เสมือนหนึ่งว่าเป็นค่าหนึ่งได้ด้วย และยังเลือกที่จะแสดง label ของแฟกเตอร์ หรือจะแสดงเป็นตัวเลขระดับก็ได้ และในกรณีที่เป็นตารางความสัมพันธ์ระหว่างตัวแปรสองตัว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

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

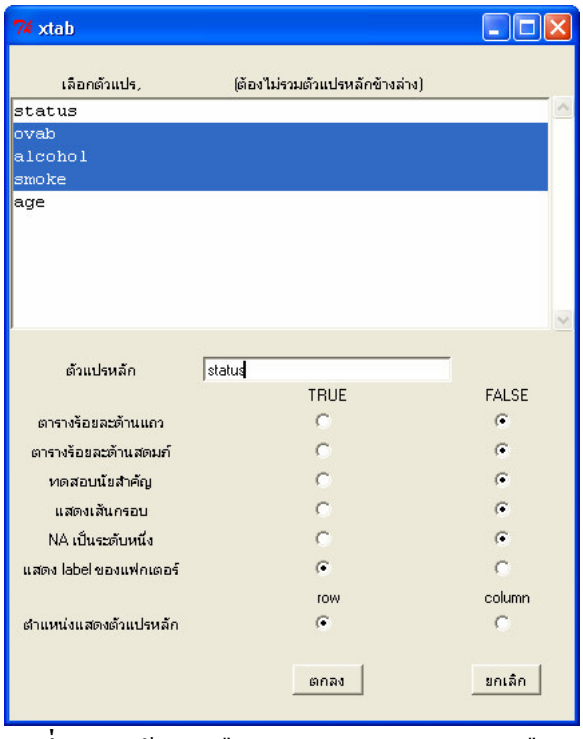
ส่วนการทำตารางสัมพันธ์ไขว้ก็เช่นเดียวกัน ในการวิจัยเพื่อหาความสัมพันธ์ระหว่างสิ่งที่คิดว่าเป็นเหตุกับสิ่งที่คิดว่าเป็นผล การทำตารางความสัมพันธ์ก็จะทำให้สามารถดูในเบื้องต้นได้ว่าคู่ของตัวแปรใดสัมพันธ์กันหรือไม่ และมากน้อยเพียงใด คำสั่งที่เรียกใช้คือคำสั่ง `xtab()`

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 ในโมดุล 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

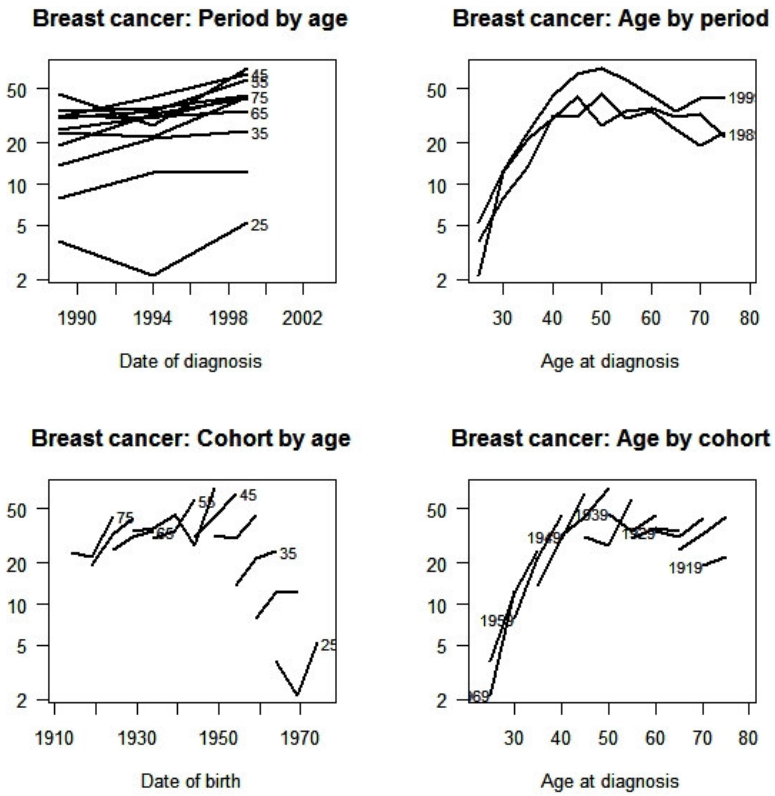
กราฟิก	**ช่วยเหลือ**
หน้าต่าง graphic ใหม่	เกี่ยวกับหน้าต่าง graph
หลาย graph ในหนึ่งหน้าต่าง	text
Histogram	html
เวกเตอร์	chm
กรอบข้อมูล	เกี่ยวกับแพคเกจ graph
Pie	
เวกเตอร์	
กรอบข้อมูล	
Boxplot	
formula	
เวกเตอร์	
กรอบข้อมูล	

*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))`

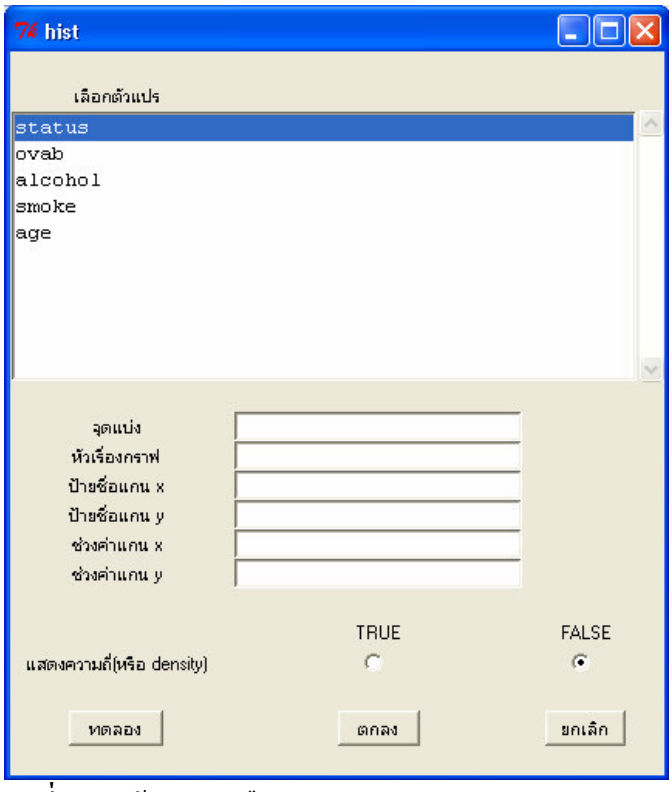
สำหรับตัวเลือกนี้เมื่อเลือกแล้ว ก็จะมีหน้าต่างให้กรอกว่าต้องการวาดกราฟิกกี่แถวและกี่สดมภ์ในหนึ่งหน้าต่าง เนื่องจากไม่ได้มีอะไรที่ต้องอธิบายมาก จึงจะไม่แสดงรูปหน้าต่างในที่นี้

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

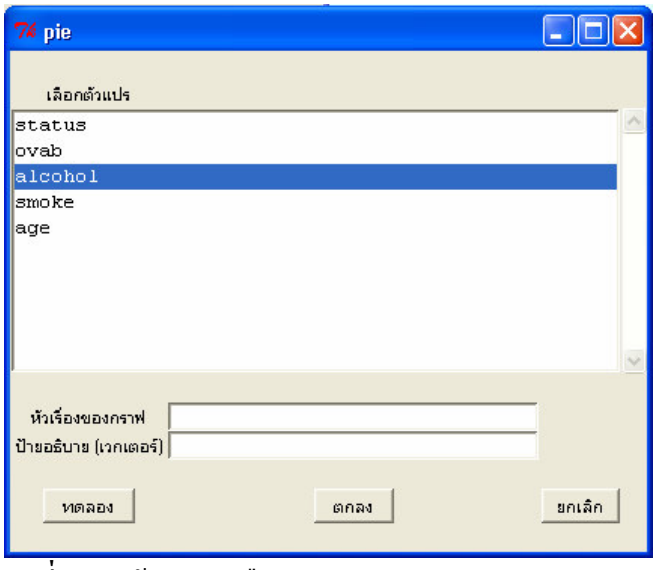
```
graph>> hist(x, breaks=, freq=FALSE, main=, xlim=, ylim=, xlab=,
ylab=)
```

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

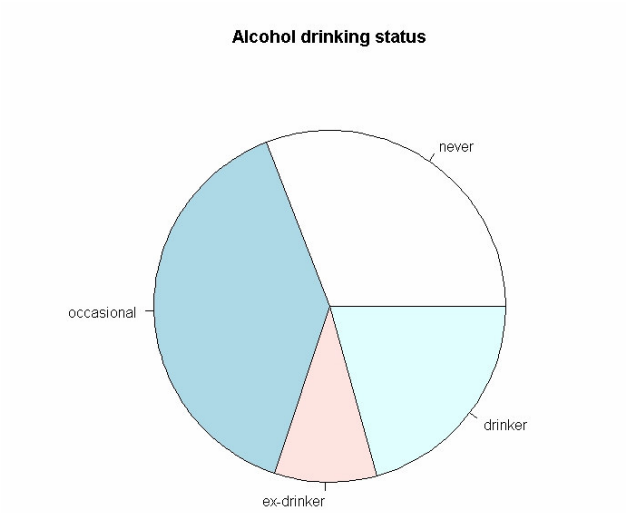
ตัวเลือกที่ปรากฏบนหน้าต่างนี้มีดังต่อไปนี้ “หัวเรื่องของกราฟ” และ “ป้ายอธิบาย (เวกเตอร์)” ซึ่งจะมีช่องให้กรอกข้อความเข้าไป ซึ่งก็คล้ายกับในหัวข้อการทำกราฟ histogram ที่กล่าวข้างต้นดังรูปที่ 9.3



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")
```

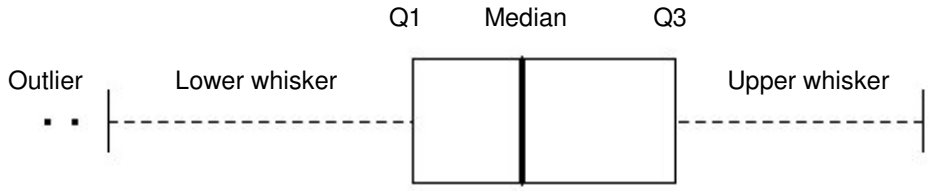
ทำกราฟ ***boxplot***

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



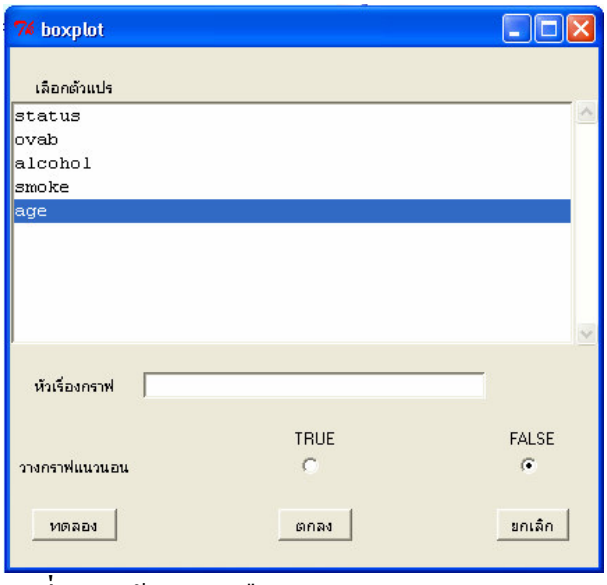
รูปที่ 9.5 ลักษณะของกราฟ boxplot

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

```
y ~ a + b + …
```

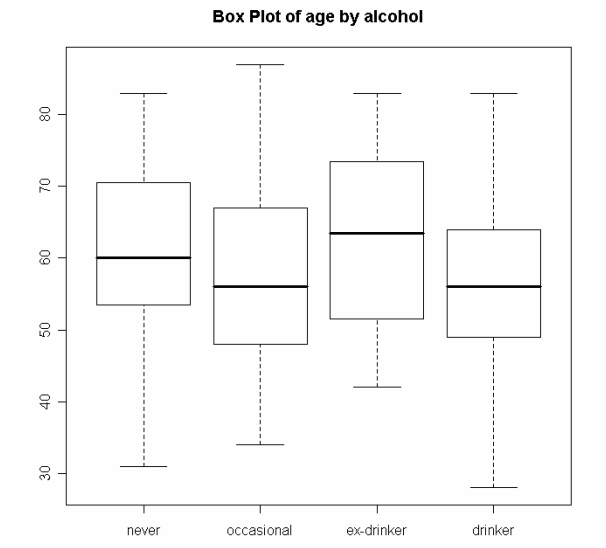
โดยที่ y เป็นตัวแปรตาม หรือในกรณีของการวาดกราฟก็จะเป็นตัวแปรที่จะพล็อต เครื่องหมาย ~ เป็นการบอกการเป็น formula ทางสถิติ ส่วนตัวแปรทางด้านขวามือเป็นตัวแปรต้น หรือในกรณีการวาดกราฟหรือในการทดสอบทางสถิติบางอย่างเช่น ANOVA ก็จะเป็นตัวแปรแบ่งกลุ่ม

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

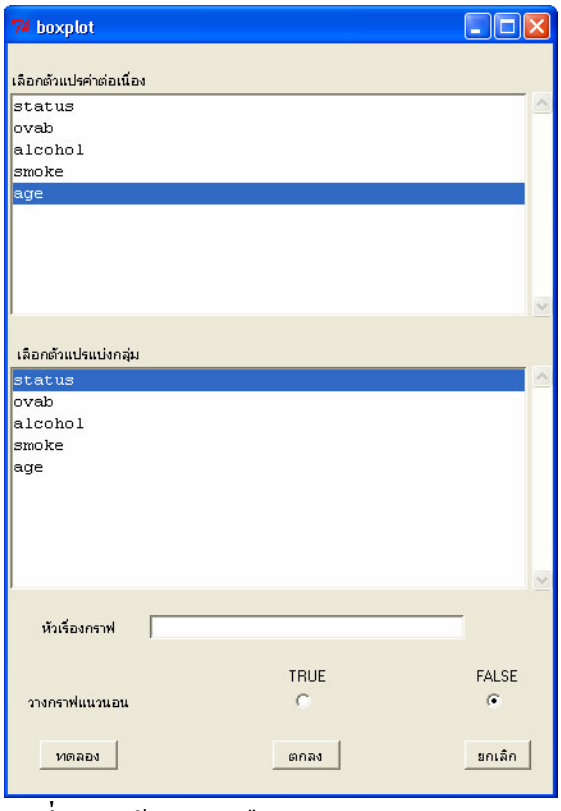
แต่หากเลือกตัวเลือกทำกราฟ boxplot จาก formula ทางสถิติแล้ว ด้านล่างของรายการเลือกตัวแปรที่จะมาทำ boxplot จะมีรายการให้เลือก “ตัวแปรแบ่งกลุ่ม” อีกด้วย ซึ่งเมื่อเลือกการแบ่งกลุ่มแล้ว ตัวแปรที่เลือกข้างต้นจะถูกทำเป็น boxplot หลายอัน เท่ากับจำนวนกลุ่มของตัวแปรนี้ ทำให้เราสามารถเปรียบเทียบการกระจายของตัวแปรตามกลุ่มต่างๆ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

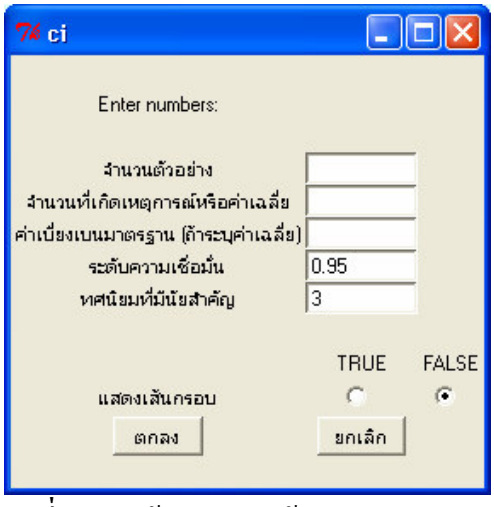
ตารางที่ 10.1 โครงสร้างของรายการเมนู ice.statis

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

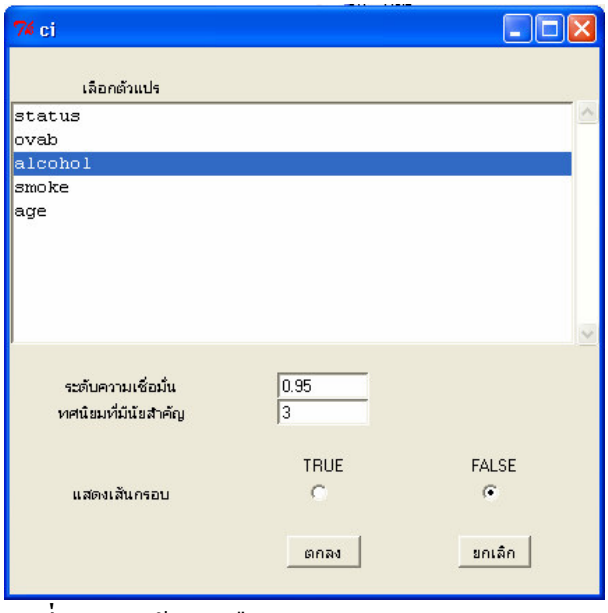
```
statis>> ci(125, 89, , level=0.95, digits=3, frame=FALSE)
 Obs. Prob.  S.E. Lower Upper
  125 0.712 0.041 0.624 0.789
```

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
```

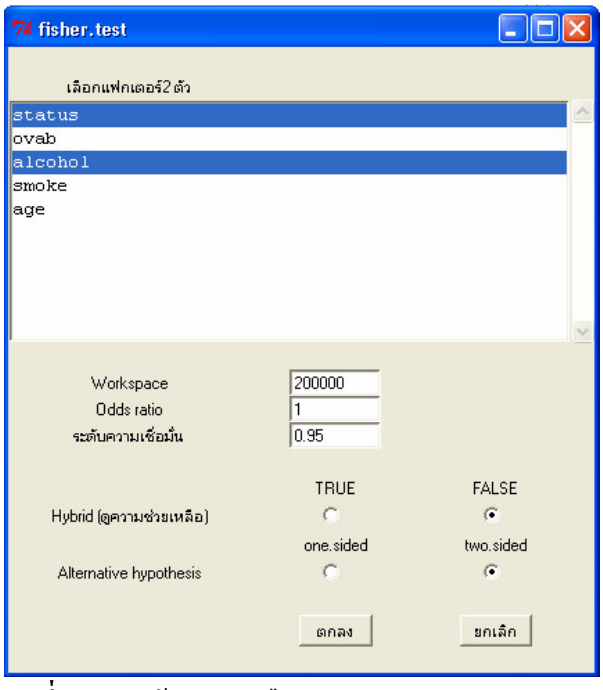
เมื่อพบว่าในกรอบข้อมูล cca นั้น ตัวแปร age เป็นเวกเตอร์ตัวเลข ก็จะคำนวณค่าเฉลี่ยให้ และเมื่อพบว่าตัวแปร alcohol เป็นแฟกเตอร์ ก็จะคำนวณค่าสัดส่วนของแต่ละระดับให้

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
```

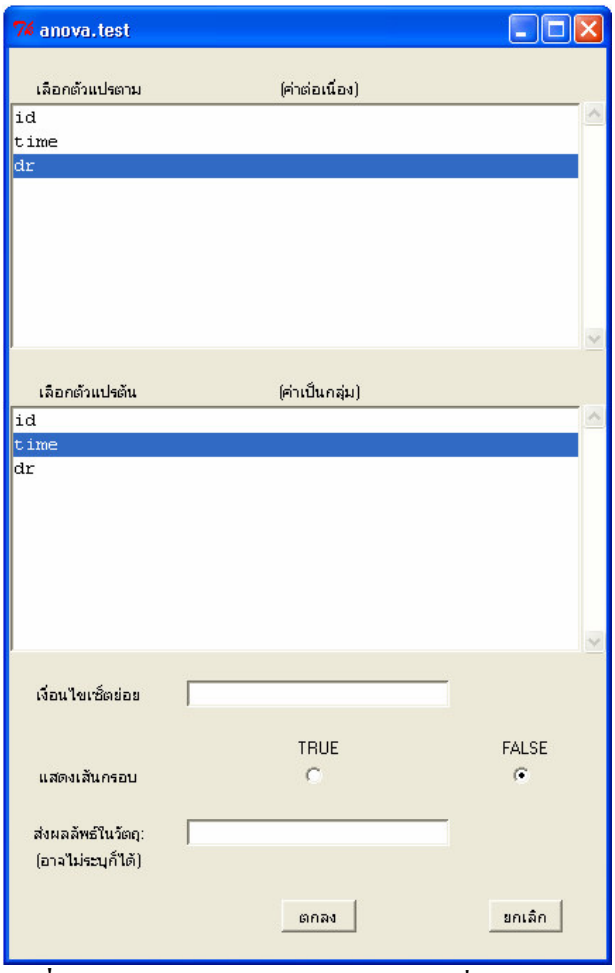
> **หมายเหตุ** ขอให้สังเกตลักษณะของ prompt ให้ดี หากเป็น `dataman>>` นั่นหมายความว่าเป็นผลจากการใช้รายการในโมดุล ice.dataman หากเป็น `>` หมายถึงผู้ใช้พิมพ์คำสั่งบน R console โดยตรง และหากเป็น `statis>>` ก็หมายถึงผู้ใช้เลือกรายการทำงานจากโมดุล ice.statis นั่นเอง

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
```

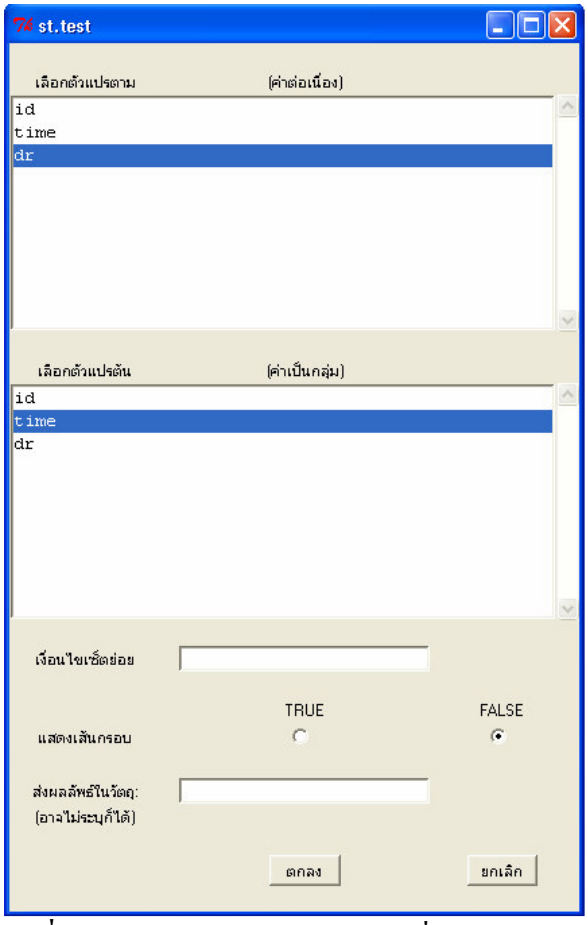
นั่นคือเราจะได้บรรทัดคำสั่งของ `anova.test()` ที่ระบุ formula และกรอบข้อมูลที่ใช้ จากนั้นแสดงผลลัพธ์ของการทำงานบรรทัดคำสั่งนั้น ซึ่งจะให้ตาราง anova นั่นเอง ในที่นี้เราพบว่าค่าการตรวจสองครั้งไม่มีความแตกต่างกันอย่างมีนัยสำคัญทางสถิติ

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

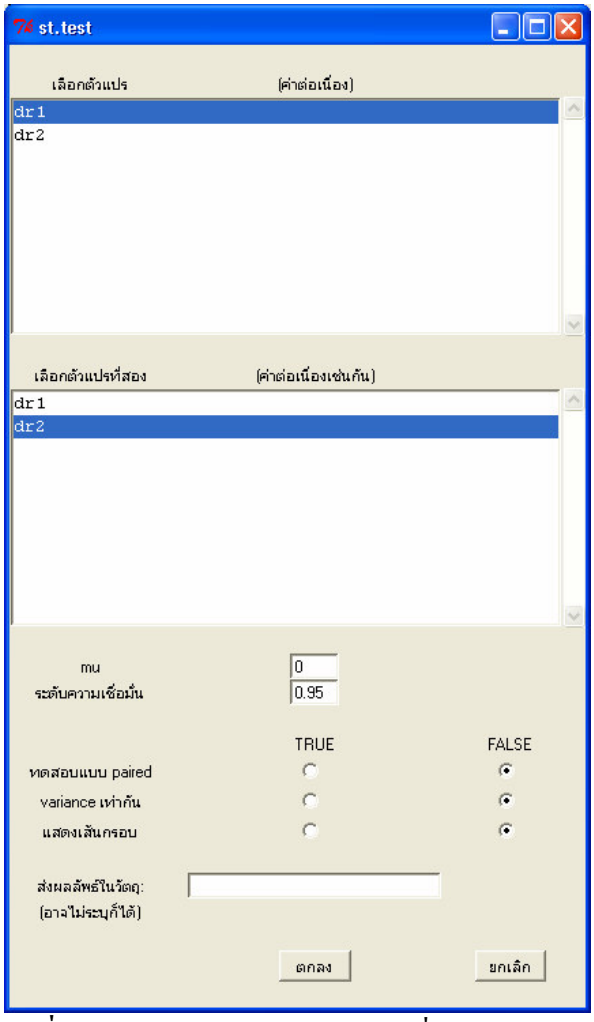
```
statis>> wilcoxon.test(dr ~ time, data=ahcc2, , frame=FALSE)
dr stratified by time
  Obs. Min. Median Max.
1 40   12   57.5   103
2 30   36   57.5   95

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

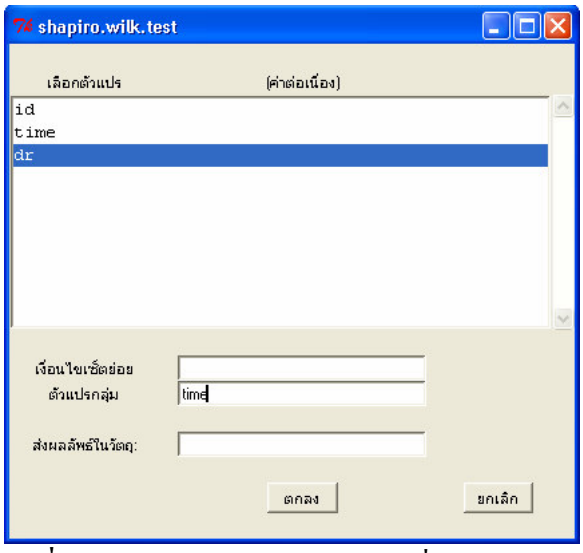
```
statis>> st.test(ahcc, dr1, dr2, mu=0, paired=TRUE, var.equal=FALSE,
conf.level=0.95, frame=FALSE)
    Obs. NA Mean  S.D.  Min. Median Max.
dr1 40   0  60.73 15.53 12   57.5   103
dr2 30   10 57.33 10.69 36   57.5   95

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**

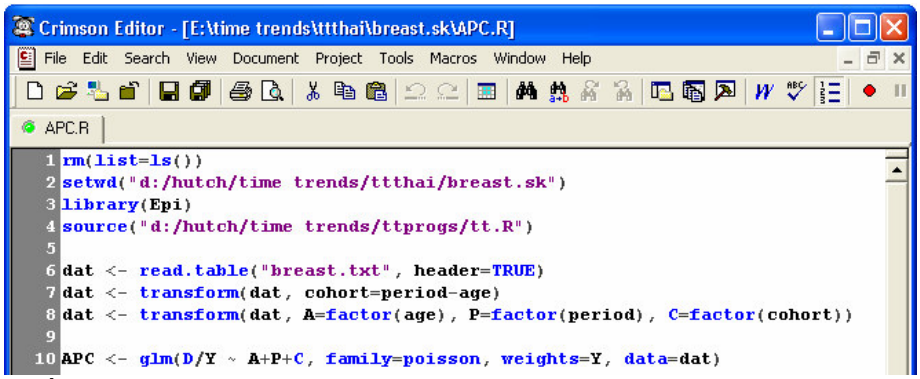
จากที่กล่าวมาในบทต่างๆ ข้างต้นนั้น เนื่องจากมีรายละเอียดมากมาย จึงอาจทำให้จับขั้นตอนการทำงานในสภาพแวดล้อม **R-ICE** ไม่ได้ชัดเจน ในบทนี้จึงจะสรุปขั้นตอนต่างๆ อีกครั้ง

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

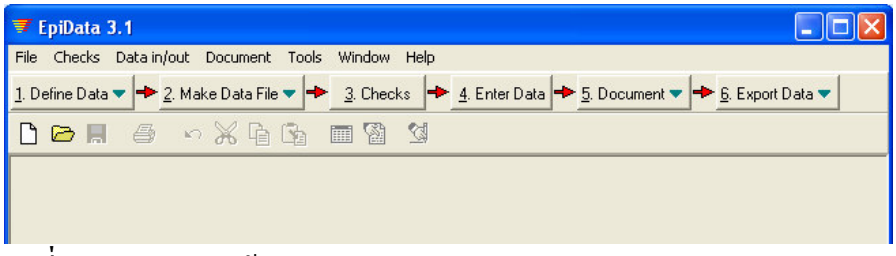
โปรแกรมช่วยเหลือ หรือสภาพแวดล้อมเสริมในการทำงานที่ควรติดตั้งร่วม ในกลุ่มแรกคือ text editor สำหรับผู้ใช้วินโดวส์แล้ว แนะนำให้ใช้โปรแกรม **Crimson Editor** ซึ่งสามารถดาวน์โหลดโปรแกรม setup ได้ฟรีที่ http://www.crimsoneditor.com ซึ่งการติดตั้งก็ไม่ยุ่งยากแต่อย่างใด ข้อดีของโปรแกรมนี้คือสามารถส่งบรรทัดคำสั่งให้ไปทำงานใน **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



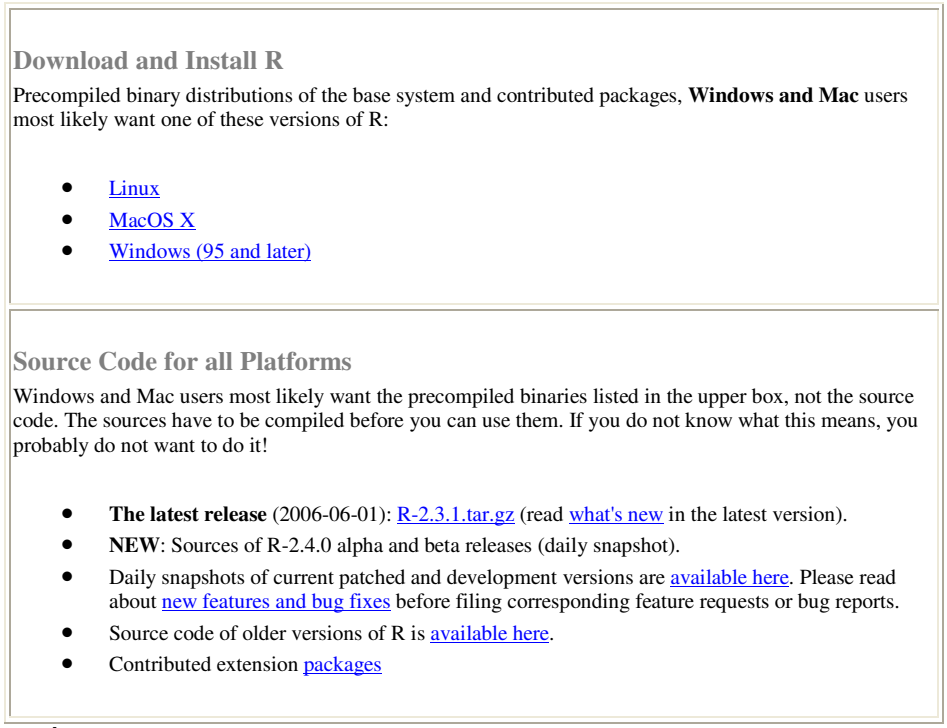
รูปที่ 11.2 ตัวอย่างหน้าต่างโปรแกรม **EpiData**

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

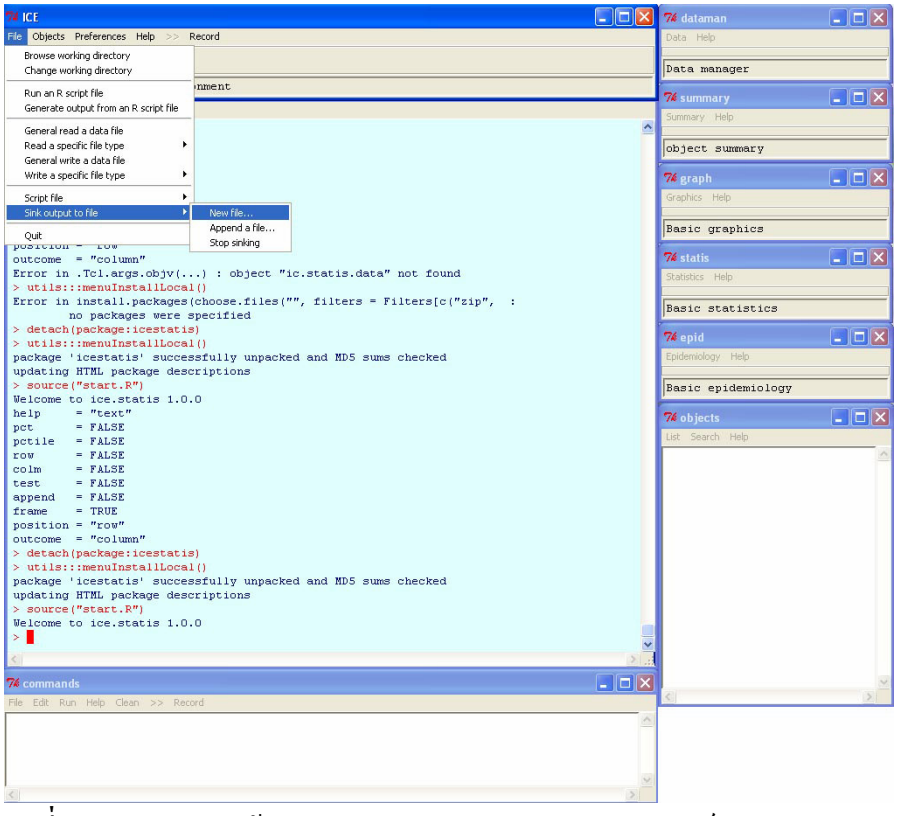
การที่ **R** เป็นภาษาโปรแกรม ทำให้การออกคำสั่งจำเป็นต้องเขียนในรูปแบบคล้ายบรรทัดคำสั่งโปรแกรมคอมพิวเตอร์ คือมีการใช้วงเล็บ และเครื่องหมายต่างๆ มากมาย ทำให้ผู้ใช้มือใหม่รู้สึกไม่สะดวกในการใช้งาน เพราะมักจะสับสนกับเครื่องหมายต่างๆ เหล่านั้น หรือสับสนกับการซ้อนคำสั่งเข้าด้วยกัน ทำให้มีวงเล็บจำนวนมาก และมักจะปิดไม่หมด 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

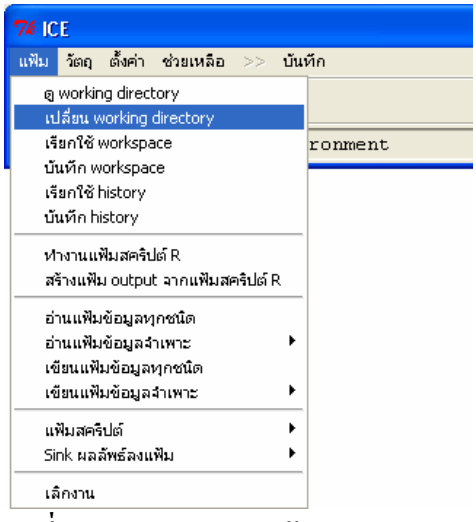
```
## A case-control study of cholangiocarcinoma in Nakhon Phanom.
## Original data file: cca.dta.
## Script file created on 2006-09-19, last modified 2006-09-28.
```

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**

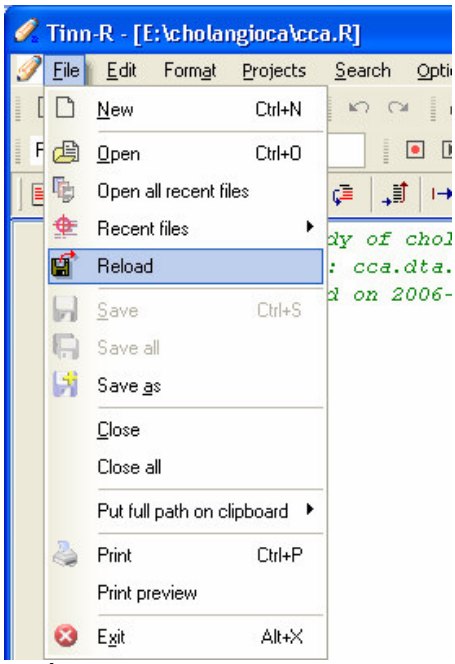
จากนั้นใช้รายการเมนู แฟ้ม >> เปลี่ยน working directory ของโมดุล ice.main (หรือ ice.fullmain) เพื่อเปลี่ยน working directory ของ **R** ไปที่โฟลเดอร์ที่สร้างขึ้นนั้น ดังรูปที่ 11.5



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

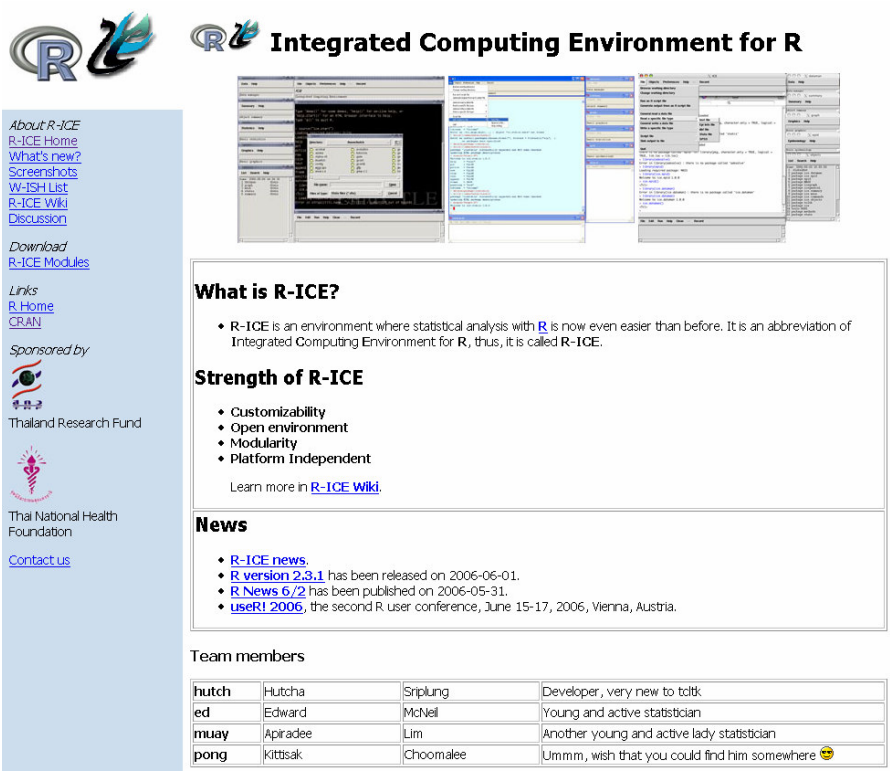
```
Output generated on: Thu Sep 28 10:18:21 2006
Version 2.3.1 Patched (2006-06-27 r38428)
cca>  setwd("E:/cholangioca")
NULL

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***

โมดูลต่างๆ ของ **R-ICE** ยังจะพัฒนาต่อไปเรื่อยๆ ตราบใดที่ยังมีผู้ต้องการใช้งาน โดยสามารถติดตามดาวน์โหลดโมดูลรุ่นใหม่ๆ 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
```