

R_book_update

สวະโชติ ศรีสุทธียากร

2024-08-12

Table of contents

คำนำ

หนังสือเล่มนี้ปรับปรุงจากหนังสือสถิติและวิทยาการข้อมูลทางการศึกษา : R สำหรับการจัดระเบียบและจัดกระทำข้อมูล เนื้อหาหลักเป็นการปูพื้นฐานให้กับผู้ที่สนใจให้มีความรู้และทักษะที่จำเป็นสำหรับการทำงานด้านวิทยาการข้อมูล และการวิจัยทางการศึกษา โดยมีการปรับปรุงเนื้อหาและชุดคำสั่งในหนังสือเล่มเดิมให้มีความทันสมัย และเพิ่มเนื้อหาในส่วนของสถิติวิเคราะห์และการเรียนรู้ของเครื่องที่เกี่ยวข้องทำให้หนังสือมีความสมบูรณ์มากยิ่งขึ้น

เนื้อหาในหนังสือจำแนกออกเป็น 5 ส่วน ดังนี้

- **ส่วนแรก** แนะนำภาษา R โดยเริ่มตั้งแต่การติดตั้งโปรแกรม แนะนำ IDE ที่เหมาะสำหรับการใช้ภาษา R และความรู้พื้นฐานที่จำเป็นสำหรับการใช้ภาษา R สำหรับงานด้านสถิติและวิทยาการข้อมูลทางการศึกษา เนื้อหาส่วนนี้เหมาะสำหรับผู้ที่ไม่เคยใช้ภาษา R มาก่อน เนื้อหาส่วนนี้จะอยู่ในบทที่ 1 และ 2 ของหนังสือ ผู้ที่มีความรู้พื้นฐานหรือประสบการณ์กับภาษา R มาแล้วสามารถข้ามเนื้อหาในส่วนนี้ได้
- **ส่วนที่สอง** การเตรียมข้อมูล เกี่ยวข้องกับการแนะนำแหล่งข้อมูลที่สามารถเข้าไปศึกษาและดาวน์โหลดมาฝึกปฏิบัติประเภทของชุดข้อมูล การนำข้อมูลเข้าสู่ R การสำรวจข้อมูลเบื้องต้น การจัดระเบียบข้อมูลและจัดกระทำข้อมูล เพื่อให้ได้ตารางข้อมูลรวมทั้งข้อมูลที่พร้อมและสอดคล้องกับความต้องการของการวิเคราะห์ เนื้อหาในส่วนนี้อยู่ในบทที่ 3 และ 4 ของหนังสือ
- **ส่วนที่สาม** การสร้างทัศนภาพข้อมูล (data visualization) และการวิเคราะห์ข้อมูลเชิงสำรวจ (exploratory data analysis: EDA) เนื้อหาส่วนนี้จะกล่าวถึงหลักการเลือกใช้ ออกแบบและสร้างทัศนภาพข้อมูลด้วยภาษา R โดยใช้ `{ggplot2}` ที่เป็น library หลักตัวหนึ่งที่มีประสิทธิภาพสูงสำหรับสร้างทัศนภาพข้อมูล เนื้อหาเน้นการสร้างและการใช้ทัศนภาพข้อมูลที่เหมาะสมสำหรับการวิเคราะห์ข้อมูลเชิงสำรวจ ร่วมกับการใช้สถิติพื้นฐานเพื่อทำความเข้าใจสภาพของตัวแปร เปรียบเทียบความแตกต่างของข้อมูล การสำรวจความสัมพันธ์ระหว่างตัวแปร การวิเคราะห์เพื่อจัดกลุ่ม (clustering) การลดทอนมิติของข้อมูล (dimension reduction) นอกจากนี้จะกล่าวถึงวิธีการที่สามารถใช้เพื่อตรวจสอบความผิดปกติในข้อมูลที่จะเป็นปัญหาหรือเป็นปัจจัยที่ลดประสิทธิภาพหรือความถูกต้องในการวิเคราะห์ข้อมูล ได้แก่ ปัญหาค่าผิดปกติ และข้อมูลสูญหาย เนื้อหาในส่วนนี้จะอยู่ในบทที่ 5 - 7 และการวิเคราะห์ที่เกี่ยวข้องในส่วนนี้อาจเรียกว่าอยู่ในกลุ่ม การวิเคราะห์ข้อมูลเชิงบรรยาย (descriptive analytics)
- **ส่วนที่สี่** การวิเคราะห์เชิงวินิจฉัย (diagnostic analytics) การวิเคราะห์ส่วนนี้มีวัตถุประสงค์เพื่อหาคำอธิบายสภาพที่พบในข้อมูลที่เป็นประเด็นที่ผู้วิเคราะห์หรือผู้วิจัยให้ความสนใจ เป็นกลุ่มของเทคนิคทางสถิติและวิทยาการข้อมูลที่สามารถนำไปใช้เพื่อสร้างสารสนเทศเชิงลึกและนำไปใช้ประโยชน์ทั้งในเชิงวิชาการ และเชิงปฏิบัติ เช่น การวิเคราะห์เพื่อเปรียบเทียบค่าเฉลี่ย การวิเคราะห์สหสัมพันธ์ การวิเคราะห์การถดถอย และการวิเคราะห์ต้นไม้ตัดสินใจ
- **ส่วนที่ห้า** การวิเคราะห์เชิงทำนาย (predictive analytics) จะกล่าวถึงการสร้างโมเดลทำนายจากอัลกอริทึมการเรียนรู้ของเครื่องในกลุ่ม supervised learning ที่สามารถนำไปใช้เพื่อสร้างโมเดลทำนายที่สามารถสร้างสารสนเทศเชิงลึกที่เป็นประโยชน์โดยเฉพาะการวางแผน และการตัดสินใจในการดำเนินงานโดยเฉพาะด้านการศึกษา เนื้อหาในส่วนนี้จะกล่าวถึงโมเดลเบื้องต้นในการสร้างโมเดลทำนาย และการสร้างโมเดลทำนายด้วย `{tidymodels}` ที่ประกอบด้วย การเตรียมข้อมูลสำหรับการสร้าง การปรับแต่ง และการตรวจสอบประสิทธิภาพของโมเดลทำนาย

ความรู้เบื้องต้นที่จำเป็นสำหรับผู้อ่าน

ผู้อ่านไม่จำเป็นต้องมีความรู้พื้นฐานเกี่ยวกับโปรแกรม R มาก่อน แต่ควรมีพื้นฐานความรู้ เกี่ยวกับสถิติพื้นฐานหรือเคยเรียนรายวิชาสถิติพื้นฐานในระดับปริญญาบัณฑิตมาอย่างน้อย 1 รายวิชา นอกจากนี้การมีพื้นฐานทางคณิตศาสตร์ในระดับมัธยมศึกษาตอนปลายจะช่วยให้สามารถ ทำความเข้าใจเนื้อหาบางส่วนของหนังสือเล่มนี้ได้มากยิ่งขึ้น

ตัวอย่างคำสั่งและชุดข้อมูลที่ใช้เป็นตัวอย่างในหนังสือ

ภายในหนังสือมีการแสดงตัวอย่างคำสั่งที่ใช้สำหรับดำเนินการต่าง ๆ ในโปรแกรม R โดย ตัวอย่างคำสั่งที่ใช้ในหนังสือเล่มนี้อาจจำแนกเป็น 2 ประเภท ได้แก่ คำสั่งที่ไม่มีการแสดงผล และ คำสั่งที่มีการแสดงผล คำสั่งที่ไม่ได้มีการแสดงผลผลลัพธ์จะแสดงในลักษณะดังตัวอย่างต่อไปนี้

```
gender<-c("Male","Female","Male","Male","Female","Male","Male","Female")
age<-c(10,10,11,2,9,4,10,14)
weight<-c(59,35,75,20,63,23,47,59)
height<-c(142,135,150,95,141,108,142,155)
data<-data.frame(gender, age, weight, height)
```

ส่วนคำสั่งที่มีการนำเสนอผลลัพธ์ที่ได้จากการประมวลผล จะมีการแสดงผลลัพธ์ต่อจากการเรียก คำสั่งดังกล่าว โดยที่ส่วนที่เป็นผลลัพธ์จะขึ้นต่อท้ายจากคำสั่ง เช่น การเรียกดูชุดข้อมูล **data** จากคำสั่งด้านบน

```
data
```

```
  gender age weight height
1  Male  10    59    142
2 Female  10    35    135
3  Male  11    75    150
4  Male   2    20     95
5 Female   9    63    141
6  Male   4    23    108
7  Male  10    47    142
8 Female  14    59    155
```

หรือการหาผลลัพธ์จากการคำนวณทางคณิตศาสตร์ เช่น

```
log(x=10, base=exp(1))
```

```
[1] 2.302585
```

```
2^5
```

```
[1] 32
```

ไฟล์ข้อมูลทั้งหมดที่ใช้เป็นตัวอย่างในหนังสือสามารถดาวน์โหลดได้จาก ...

Chapter 1

แนะนำโปรแกรม R

เนื้อหาภายในบทเรียนนี้เหมาะสำหรับผู้ที่ยังไม่เคยมีพื้นฐานเกี่ยวกับโปรแกรม R มาก่อน โดยจะแนะนำภาพรวมของภาษา R เริ่มตั้งแต่การดาวน์โหลดและติดตั้งโปรแกรม สภาพแวดล้อมของโปรแกรม และแนะนำ IDE ที่เหมาะสำหรับการใช้ร่วมกับโปรแกรม R รายละเอียดมีดังนี้

1.1 R คืออะไร?

R เป็นภาษาคอมพิวเตอร์ยุคใหม่ที่ถูกพัฒนาขึ้นให้มีความสามารถอย่างหลากหลาย มีประสิทธิภาพสูง และดีมากสำหรับใช้ในการทำงานทางด้านสถิติและวิทยาการข้อมูล โปรแกรม R ได้รับการริเริ่มพัฒนาขึ้นโดยผู้พัฒนาที่เป็นนักสถิติสองท่าน ได้แก่ Ross Ihaka และ Robert Gentleman จาก University of Auckland ประเทศนิวซีแลนด์ โดยพัฒนาต่อยอดมาจากภาษา S และ S+ และได้ทำการเผยแพร่ให้บุคคลทั่วไปได้ใช้งานตั้งแต่ปี ค.ศ. 1993 ภายใต้สัญญาอนุญาตสาธารณะทั่วไปของกนู (GNU General Public License) โปรแกรม R จัดเป็นโปรแกรมประเภท open source ซึ่งมีลักษณะเป็นโปรแกรมที่เผยแพร่ให้บุคคลทั่วไปมีสิทธิในการเข้าใช้งาน และพัฒนาโปรแกรมอย่างอิสระตามความต้องการโดยไม่เสียค่าใช้จ่ายไม่ว่าจะ เป็นการใช้งานทั่วไป การแก้ไขปรับปรุง หรือพัฒนาต่อยอดโปรแกรม (Maindonald and Ruan, 2010; Field, Mild and Field, 2012; Schumacker, 2012; Brundon and Comber, 2013)

โปรแกรม R สามารถทำงานได้บนแพลตฟอร์มที่หลากหลาย (multiple platform) โดยสามารถติดตั้งและทำงานบนระบบปฏิบัติการที่สำคัญได้ทุกระบบ ได้แก่ Windows, Mac OS, Linux, Unix รวมทั้ง Chrome OS จุดเด่นนี้ทำให้โปรแกรม R สามารถเข้าถึงผู้ใช้งานได้อย่างทั่วถึง

การที่ R เป็นโปรแกรม open source ยังเป็นปัจจัยสนับสนุนที่ทำให้เกิดชุมชนนักพัฒนาที่กว้างขวางและมี library จำนวนมากที่ถูกพัฒนาขึ้นอย่างรวดเร็ว ต่อเนื่อง ทำให้ R มีส่วนต่อขยายที่ครอบคลุมการดำเนินการทางด้านสถิติและวิทยาการข้อมูล รวมทั้งการวิจัยแทบทุกด้าน สามารถตอบสนองต่อความต้องการและมีการพัฒนาที่ทันต่อการเปลี่ยนแปลงอย่างรวดเร็วในวงการวิทยาการข้อมูล ทำให้ปัจจุบัน R เป็นเครื่องมือหลักตัวหนึ่งสำหรับนักสถิติ นักวิทยาการข้อมูล และนักวิจัยจากทั่วโลก สำหรับการทำงานวิจัยและการวิเคราะห์ข้อมูลสมัยใหม่

ผู้เขียนได้ลองวิเคราะห์และจัดกลุ่มสามารถหลักของโปรแกรม R พบว่า อาจจำแนกได้เป็น 5 ด้าน คือ การนำเข้าข้อมูล (importing data) การจัดระเบียบและจัดการข้อมูล (tidying and manipulating data) การคำนวณทางคณิตศาสตร์ (mathematical computations) การวิเคราะห์ข้อมูลและพัฒนาโมเดลทางสถิติ (data analysis and statistical modelling) ซึ่งครอบคลุมทั้งโมเดลสำหรับการวิเคราะห์เชิงวินิจฉัย การวิเคราะห์เชิงทำนาย และการนำเสนอ ข้อมูลและการสร้างทัศนภาพข้อมูล (data presentation and data visualization) โดยเมื่อพิจารณาความสามารถในแต่ละด้านข้างต้น พบว่ามีจุดเด่นหลายประการ ดังนี้

1. สามารถนำเข้าข้อมูลได้หลายประเภท ด้วยวิธีการที่หลากหลาย โดยสามารถดำเนินการได้ ทั้งการป้อนข้อมูลโดยตรง การนำเข้าจากไฟล์ข้อมูลประเภทต่าง ๆ ซึ่งครอบคลุมทุกประเภท ของไฟล์ข้อมูล การนำเข้าข้อมูลจากฐานข้อมูล ไปจนถึงการดาวน์โหลดและเก็บเกี่ยวข้อมูล จากเว็บไซต์ (web scrapping)

2. เป็นโปรแกรมที่มีความสามารถสูงในการจัดระเบียบและจัดกระทำข้อมูล โดยมีเครื่องมือที่มีประสิทธิภาพสูงมากมายตัวที่ช่วยจัดระเบียบตารางข้อมูลอยู่ในรูปแบบที่เหมาะสมสำหรับการวิเคราะห์ข้อมูลในงานต่าง ๆ และช่วยจัดกระทำข้อมูล เช่น การแบ่งส่วนย่อยของชุดข้อมูล การแปลงรหัสหรือแปลงค่าของตัวแปร หรือการจัดลำดับของข้อมูลตามค่าของตัวแปรที่กำหนด เป็นต้น
3. โปรแกรม R มีฟังก์ชันสำเร็จรูปทั้งทางคณิตศาสตร์และสถิติจำนวนมากที่ถูกติดตั้งมาพร้อมกับการติดตั้งโปรแกรมในครั้งแรก ซึ่งสามารถเรียกใช้เพื่อช่วยให้การดำเนินงานต่าง ๆ สามารถทำได้โดยง่ายและมีประสิทธิภาพ และนอกจากนี้ยังมีฟังก์ชันสำเร็จรูปอีกจำนวนมากจาก package เสริมต่าง ๆ บน CRAN server ที่ถูกพัฒนาขึ้นโดยนักวิชาการ หรือนักพัฒนาจากทั่วโลก โดยปัจจุบันมี package จำนวนมากกว่า 10,000 ตัว บน server ดังกล่าวที่ผู้ใช้สามารถดาวน์โหลดและติดตั้งมาใช้เพื่อเสริมความสามารถในการดำเนินงานของ โปรแกรม คุณสมบัตินี้ทำให้การดำเนินงานทางด้านสถิติและวิทยาการข้อมูลด้วยโปรแกรม R สามารถดำเนินงานได้ยืดหยุ่นมาก สามารถปรับแต่งและเลือกวิธีการดำเนินงานให้มีความเหมาะสม ทันสมัย และมีประสิทธิภาพสูงสุด โดยมีข้อจำกัดในการดำเนินงานที่น้อย
4. สามารถเชื่อมต่อและทำงานร่วมกับโปรแกรมวิเคราะห์ข้อมูลเฉพาะทางอื่น ๆ ได้หลาย โปรแกรม เช่น Mplus, MLWins, OpenBUGS, JAGS หรือ Stan ซึ่งช่วยให้สามารถใช้ ความสามารถของโปรแกรมห้กล่าวได้ และยกระดับความสามารถของโปรแกรม R ให้เทียบ เท่าและในบางกรณีอาจเหนือกว่าการใช้โปรแกรมวิเคราะห์ข้อมูลดังกล่าวโดยตรง
5. มีความประสิทธิภาพสูงมากสำหรับการทำงานด้านการฝึกหรือการสร้างทัศนภาพข้อมูล (data visualization) โดยเป็นโปรแกรมที่มี package หลายตัวที่ถูกพัฒนาขึ้นสำหรับสร้าง ทัศนภาพข้อมูลโดยเฉพาะ เช่น graphics, ggplot2, lattice, boken, rmarkdown, flexdashboard และ shiny ซึ่งช่วยให้ผู้ใช้สามารถสร้างทัศนภาพข้อมูลได้อย่างหลากหลาย ทั้งในรูปแบบทัศนภาพข้อมูลเชิงสถิติ (static data visualization) ทัศนภาพข้อมูลเชิงพลวัต (dynamic data visualization) และทัศนภาพข้อมูลเชิงปฏิสัมพันธ์ (interactive data visualization)
6. เป็นโปรแกรมภาษาที่ง่ายต่อการเรียนรู้และใช้งาน นอกจากนี้ยังมีชุมชนผู้ใช้โปรแกรม R และ แหล่งการเรียนรู้ออนไลน์ที่สามารถให้คำตอบแก่ผู้ใช้ได้อย่างกว้างขวางและตรงประเด็น ดัง นั้นผู้ใช้โปรแกรม R หรือผู้ที่ต้องการศึกษา R ที่ไม่ได้มีพื้นฐานการเขียนโปรแกรมมาก่อนจึง สามารถเรียนรู้ภาษา R เป็นภาษาแรกได้โดยง่าย

1.2 การดาวน์โหลดและการติดตั้ง R

R เป็นโปรแกรมที่ได้รับการพัฒนาและได้รับการปรับปรุงอย่างต่อเนื่อง ในหนังสือเล่มนี้ใช้โปรแกรม R version 4.4.1 (Race of Your Life) สำหรับผู้อ่านที่ยังไม่มี โปรแกรมสามารถดาวน์โหลดโปรแกรมได้จาก <http://www/r-project.org/> โดยเมื่อเข้าสู่ website ให้คลิกที่คำว่า “download R” เพื่อดาวน์โหลดซอฟต์แวร์จาก CRAN (Comprehensive R Archive Network) โดยให้ดาวน์โหลดตัว base distribution ที่เหมาะสมกับระบบปฏิบัติการของ ตนเองแล้วดำเนินการติดตั้งโปรแกรมตามขั้นตอนที่ตัวช่วยการติดตั้งแนะนำ

เมื่อติดตั้งโปรแกรมเสร็จสมบูรณ์ และดำเนินการเปิดโปรแกรม ผู้อ่านจะพบกับหน้าต่างที่เรียกว่า R Console ในรูป 1.2 หน้าต่างดังกล่าวมีหน้าที่ รับคำสั่ง/ข้อมูลเข้าสู่โปรแกรม ส่งผ่านคำสั่งดังกล่าวไปยังหน่วยประมวลผลของเครื่อง รายงานผลลัพธ์/สถานะการทำงานต่าง ๆ ให้กับผู้ใช้ การป้อนคำสั่งในหน้าต่าง Console สามารถทำได้โดยพิมพ์คำสั่งไว้ ที่บริเวณด้านหลังเครื่องหมาย > (เรียกว่า เครื่องหมาย prompt) โดยเมื่อพิมพ์คำสั่งเสร็จแล้วให้ผู้ใช้ กดปุ่ม Enter โปรแกรมจะทำการประมวลผล และแสดงผลลัพธ์ในบรรทัดถัดไป อย่างไรก็ตามการ เขียนคำสั่งใน R console มีข้อจำกัดประการหนึ่งคือผู้ใช้สามารถเขียนคำสั่งและประมวลได้ทีละ บรรทัด ผู้ใช้สามารถประมวลผลหลายคำสั่งภายในบรรทัดเดียวกันได้ โดยการใช้เครื่องหมาย semicolon (;) คั่นระหว่างคำสั่ง เช่น 1+1; 2^2+3; 2*3+4 และเมื่อกด Enter จะได้ผลลัพธ์ของ คำสั่งทั้งหมดในคราวเดียว

จากข้อจำกัดในการทำงานบนหน้าต่าง Console ข้างต้น จึงมีการพัฒนาหน้าต่าง Editor ขึ้นเพื่อใช้สำหรับใช้เขียนชุดคำสั่งที่มีความซับซ้อนเข้าสู่โปรแกรม หน้าต่าง Editor ยอมให้ผู้ใช้ สามารถป้อนคำสั่งหรือข้อมูลได้หลายบรรทัด โดยยังไม่จำเป็นต้องส่งประมวลผลในทันที สามารถ เลือกประมวลผลคำสั่งทีละบรรทัด บางบรรทัด หรือทุกบรรทัดได้อย่างอิสระตามความต้องการ นอกจากนี้คำสั่งที่เขียนในหน้าต่าง Editor ยังสามารถเก็บบันทึกไว้ในไฟล์นามสกุล .R (โดยทั่วไป เรียกว่า script file) ซึ่งช่วยให้ผู้ใช้สามารถจัดระเบียบในการทำงานได้ สามารถสืบค้นย้อน ประวัติการทำงานจากคำสั่งที่เขียนไว้ก่อนหน้าได้ นอกจากนี้ยังสามารถนำกลับมาใช้ซ้ำ แก้ไข หรือดัดแปลงให้เหมาะสำหรับการทำงานอื่น ๆ ต่อไปได้อีกด้วย

หน้าต่าง Editor นี้ไม่ได้ปรากฏให้ผู้ใช้ใช้งานได้ทันทีเมื่อเปิดโปรแกรม ผู้ใช้จำเป็นต้อง เรียกเปิดหน้าต่างดังกล่าวขึ้นมาใช้งานโดยคลิกเลือกที่เมนู File บนแถบเมนูด้านบน จากนั้นเลือก New Script (สำหรับระบบปฏิบัติการ Windows) หรือเลือก New Document (สำหรับระบบ ปฏิบัติการ Mac OS) โดยหากต้องการให้ R ประมวลผลคำสั่งในบรรทัดใด ให้ผู้ใช้คลิกเลือกคำสั่งหรือ ทำ highlight กลุ่ม

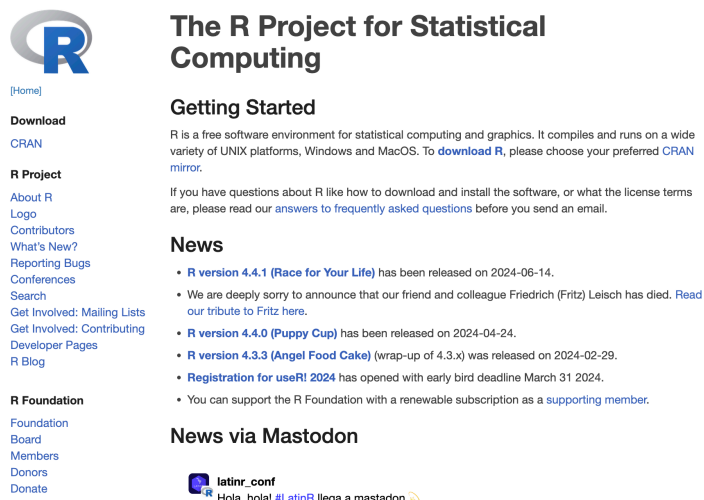


Figure 1.1: Website ของ R project

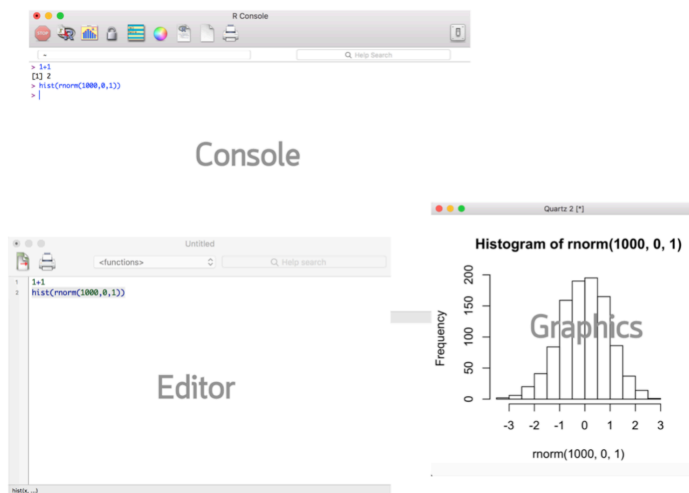


Figure 1.2: ภาพหน้าจอของ R

บรรทัดของคำสั่งที่ต้องการประมวลผล จากนั้นกดปุ่ม `Ctrl+R` (สำหรับระบบปฏิบัติการ Windows) หรือกดปุ่ม `⌘+Enter` (สำหรับระบบปฏิบัติการ Mac OS)

หน้าต่างสุดท้ายเรียกว่าหน้าต่าง Graphics (สำหรับระบบปฏิบัติการ Mac OS เรียก หน้าต่างนี้ว่า Quartz) เป็นหน้าต่างสำหรับแสดงผลพล็อตเชิงกราฟิกที่ประมวลผลได้จากโปรแกรม ผู้ใช้จะพบกับหน้าต่างนี้เมื่อมีการเรียกใช้คำสั่งที่ให้ผลลัพธ์เชิงกราฟิก ดังตัวอย่างในรูป 1.2 ที่แสดง การสร้าง histogram เพื่อสำรวจการแจกแจงของข้อมูลจำลองจากเลขสุ่มที่มีการแจกแจงแบบปกติ มาตรฐานจำนวน 1000 ค่าโดยใช้ฟังก์ชัน `hist()`

1.3 RStudio IDE

ปัจจุบันการทำงานในโครงการหรืองานวิจัยที่มีการใช้สถิติและวิทยาการข้อมูล ผู้วิเคราะห์จะต้องเกี่ยวข้องกับการใช้เครื่องมือและเทคนิคจำนวนมากและหลากหลาย ทำให้การทำงานภายใต้สภาพแวดล้อมของ R โดยตรงอาจทำได้ยากโดยเฉพาะในกลุ่มผู้ใช้ที่ไม่ได้เชี่ยวชาญการเขียนโปรแกรม นอกจากนี้ยังอาจมีความยากในการบริหารจัดการไฟล์ข้อมูลและไฟล์ที่เกี่ยวข้องกับการดำเนินงาน

จากความจำเป็นดังกล่าวจึงมีการพัฒนาโปรแกรมประเภท IDE (Integrated Development Environment) ขึ้นเพื่อให้ผู้ใช้ R มีเครื่องมือหรือตัวช่วยในการเขียนคำสั่งหรือพัฒนาโปรแกรมการวิเคราะห์ข้อมูลได้ง่ายและมีประสิทธิภาพมากขึ้น โดย Rstudio มีคุณสมบัติที่เป็นจุดเด่นดังนี้

1. RStudio มีสภาพแวดล้อมที่เป็นมิตรกับผู้ใช้มากขึ้น โดยมีการออกแบบส่วนตอบสนองกับผู้ใช้ที่มีการวางเครื่องมืออำนวยความสะดวกในการสั่งการทำงานของโปรแกรม R ทำให้การดำเนินการหลายส่วนสามารถทำได้โดยไม่ต้องเขียนชุดคำสั่ง ซึ่งช่วยลดระยะเวลาการทำงานและทำให้ใช้งานภาษา R ได้ง่ายมากขึ้น เช่น การติดตั้งและเรียกใช้ library ต่าง ๆ การค้นหาไฟล์ข้อมูล การนำเข้าไฟล์ข้อมูล และ เรียกดูข้อมูลของตัวแปร
2. มีตัวแก้ไขคำสั่ง (Script Editor) ที่มีประสิทธิภาพสูงขึ้น โดยมีจุดเด่นที่เหนือกว่าการใช้ R Editor หลายประการ เช่น การเน้นสีของไวยากรณ์ภาษาทำให้การตรวจสอบแก้ไขคำสั่งสามารถทำได้ง่ายมากขึ้น การแนะนำคำสั่งที่เป็นไปได้ การแจ้งเตือนและตรวจสอบข้อผิดพลาด นอกจากนี้ Editor ของ Rstudio ยังสามารถทำงานร่วมกับ Github Copilot ที่เป็น AI ที่ถูกฝึกสอนมาให้มีความสามารถในการช่วยสร้างคำสั่งซึ่งทำให้การทำงานด้านวิทยาการข้อมูลมีความสะดวกสบาย และมีประสิทธิภาพขึ้นอย่างมาก
3. การจัดการ workspace และไฟล์ที่เกี่ยวข้อง RStudio จะจัดการ workspace และไฟล์ทั้งหมดที่เกี่ยวข้องกับโปรเจกต์ในโฟลเดอร์เดียว ทำให้การจัดการไฟล์มีความเป็นระบบ และลดความสับสนเมื่อทำงานกับหลายโปรเจกต์
4. การจัดการเวอร์ชัน (version control) RStudio รองรับการใช้งานร่วมกับ Git ภายในโปรเจกต์ ทำให้สามารถควบคุมเวอร์ชันของโค้ดได้อย่างมีประสิทธิภาพ เช่น การ commit, push, pull, และดูประวัติการเปลี่ยนแปลงของโค้ดได้จากภายใน RStudio โดยตรง
5. RStudio รองรับการใช้งาน R Markdown และ Quarto ซึ่งเป็นเครื่องมือที่ช่วยให้ผู้ใช้สามารถสร้างเอกสารที่รวมโค้ด, ข้อความ, และผลลัพธ์ไว้ในไฟล์เดียวกัน สามารถสร้างรายงานแบบทำซ้ำได้ (reproducible report) ในรูปแบบต่าง ๆ เช่น เอกสารรายงาน/บทความ หนังสือ สไลด์นำเสนอ และ เว็บไซต์

เมื่อเปิด RStudio ขึ้นมาผู้ใช้จะพบกับหน้าต่างหรือสภาพแวดล้อมดังรูป 1.3 ที่แบ่งออกเป็น 4 ส่วน ได้แก่ (1) **Source** เป็นส่วนสำหรับใช้เขียนคำสั่ง (scripts) เขียน markdown หรือเอกสาร quarto ซึ่งทำให้ผู้ใช้สามารถจัดการกับโค้ดได้อย่างเป็นระบบสามารถส่งรันโค้ดบางส่วนหรือทั้งไฟล์ได้อย่างสะดวก ในทำนองเดียวกับ R Editor หากต้องการให้ R ประมวลผลคำสั่งในบรรทัดใด ให้ผู้ใช้คลิกเลือกคำสั่งหรือทำ highlight กลุ่มบรรทัดของคำสั่งที่ต้องการประมวลผล จากนั้นกดปุ่ม `Ctrl+R` (สำหรับระบบปฏิบัติการ Windows) หรือกดปุ่ม `⌘+Enter` (สำหรับระบบปฏิบัติการ Mac OS) (2) **Console** เป็นส่วนที่แสดงผลจากการรันโค้ดหรือคำสั่งต่างๆ โดยผู้ใช้สามารถพิมพ์คำสั่งโดยตรงเพื่อให้รันทันทีได้ทันที (3) **Environment/History** ในส่วนนี้จะแสดงข้อมูลของตัวแปรต่างๆ ที่ถูกสร้างขึ้นใน session ของการทำงาน รวมถึงประวัติของคำสั่งที่ถูกใช้ไปก่อนหน้านี้ ช่วยให้ผู้ใช้สามารถย้อนดูหรือเรียกใช้คำสั่งเดิมได้ง่าย และ (4) **Files/Plots/Packages/Help/Viewer** เป็นส่วนสำหรับการจัดการไฟล์ การดูกราฟหรือแผนภาพที่สร้างขึ้น การจัดการ package/library ที่ติดตั้ง การค้นหาความช่วยเหลือใน R และการแสดงผลข้อมูลที่เกี่ยวข้อง เช่น ไฟล์ HTML หรือไฟล์ภาพที่ถูกสร้างขึ้น

ผู้ใช้สามารถเข้าไประบบ RStudio ได้ที่ <https://posit.co/download/rstudio-desktop/> ทั้งนี้ก่อนการติดตั้ง RStudio ผู้ใช้จำเป็นต้องมีโปรแกรม R ตัวพื้นฐานติดตั้งอยู่ในเครื่องก่อนแล้ว

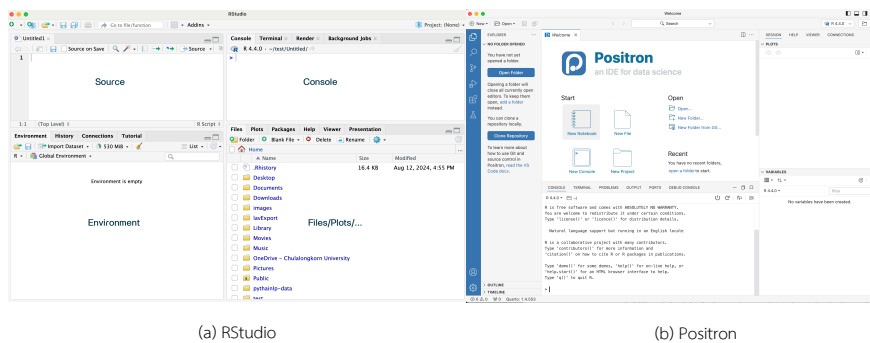


Figure 1.3: ภาพแวดล้อมของ IDE

1.4 Positron IDE

Positron ได้การนิยามตัวเองว่าเป็น IDE สำหรับวงการวิทยาการข้อมูลที่ต้องการเขียนโปรแกรมหลากหลายภาษา ออกแบบมาเพื่อผู้ใช้ VS Code หรือ JupyterLab ที่ทำงานด้านวิทยาการข้อมูลด้วยภาษา Python หรือ R แต่ยังรู้สึกไม่พอใจกับฟีเจอร์หรือความสามารถในการปรับแต่งที่มีอยู่เดิม Positron เน้นการใช้งานหลักในภาษา Python และ R พร้อมความสามารถในการเพิ่มภาษาอื่น ๆ ผ่านส่วนขยาย (รองรับส่วนขยายของ VS Code) โครงการ Positron นี้พัฒนาในรูปแบบโอเพนซอร์สบน GitHub และยังคงพัฒนาโดยบริษัท Posit ควบคู่ไปกับ RStudio โดยไม่ได้ทิ้งซอฟต์แวร์เดิมเพื่อพัฒนาใหม่ทั้งหมด ขณะนี้ Positron อยู่ในช่วงเริ่มต้นของการพัฒนา โดยมีลักษณะการใช้งานคล้ายกับ VS Code ที่รองรับ Python และ R โดยไม่ต้องติดตั้งส่วนขยายเพิ่มเติม Posit มีแผนที่จะเพิ่มฟีเจอร์สำหรับงานวิทยาการข้อมูลเพิ่มเติมในอนาคต

สภาพแวดล้อมใน Positron มีโครงสร้างคล้ายกับ Rstudio และมีหน้าต่างย่อยสำหรับทำงานที่คล้ายคลึงกับรูป 1.4 แสดงสภาพแวดล้อมของ Positron

ผู้ใช้งานสามารถดาวน์โหลด Positron ได้ที่ <https://github.com/posit-dev/positron/releases> และสามารถติดต่อข่าวสารและรายละเอียดเกี่ยวกับ Positron ได้ที่ <https://github.com/posit-dev/positron?tab=readme-ov-file>

1.5 ควรใช้ RStudio หรือ Positron

...

Chapter 2

พื้นฐาน R

เมื่อผู้อ่านดาวน์โหลดและติดตั้งโปรแกรม R รวมทั้ง RStudio หรือ Positron แล้ว บทเรียนนี้จะกล่าวถึงการใช้ภาษาเบื้องต้น ได้แก่ การคำนวณทางคณิตศาสตร์พื้นฐาน ฟังก์ชัน มโนทัศน์เกี่ยวกับตัวแปรในภาษา R ประเภทของตัวแปรและการสร้างตัวแปร และการอ้างอิงค่าหรือสมาชิกภายในตัวแปรที่สร้างขึ้น ในการเรียนรู้และประมวลผลค่าส่งตามเนื้อหาในบทเรียนนี้ ผู้อ่านสามารถพิมพ์คำสั่งบนเอกสาร Script โดยหากใช้ RStudio ผู้อ่านสามารถเปิดเอกสาร Script ได้โดยคลิกที่แถบเมนูด้านบน File -> New File -> R Script สำหรับใน Positron ให้สร้างไฟล์เอกสาร .R โดยบนแถบเมนูให้คลิกที่ File -> New File -> R File R ดังรูป 2.1

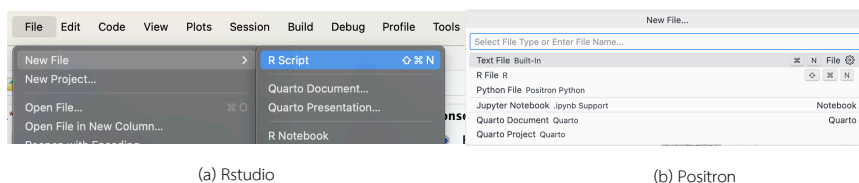


Figure 2.1: การเปิดหน้าต่าง Editor หรือ Script ใน IDE

รายละเอียดมีดังนี้

2.1 การคำนวณทางคณิตศาสตร์พื้นฐาน

ภาษา R มีฟังก์ชันพื้นฐานสำหรับการคำนวณทางคณิตศาสตร์จำนวนมาก เช่น การดำเนินการพีชคณิตพื้นฐานได้แก่ การบวก (+) ลบ (-) คูณ (*) หาร (/) ยกกำลัง (^) และ รากที่สอง (`sqrt()`) รวมทั้งการคำนวณผลลัพธ์จากฟังก์ชันอัติโนมัติ (implicit function) ต่าง ๆ เช่น ฟังก์ชันตรีโกณมิติ `sin()`, `cos()`, `tan()` ฟังก์ชันลอการิธึมธรรมชาติ `log()` ฟังก์ชันเอกซ์โพเนนเชียล `exp()` และ วงเล็บ () เป็นต้น ชุดคำสั่งด้านล่างแสดงตัวอย่างการดำเนินการทาง คณิตศาสตร์ใน R ผู้อ่านลองพิมพ์คำสั่งดังกล่าวลงในเครื่องคอมพิวเตอร์ของตนเอง จากนั้นสังเกต ผลลัพธ์ที่ได้

```
1+1; 3-2; 4*5; 10/2
3^3; sqrt(625); 81^(1/3)
5%%3; (3^3+5-1)
log(10); exp(5)
```

2.2 ฟังก์ชัน (functions)

ในหัวข้อที่ผ่านมาจะเห็นว่าการใช้งานฟังก์ชันในโปรแกรม R ไปบางตัวทั้งฟังก์ชันทาง คณิตศาสตร์ เช่น `sqrt()`, `exp()` และ `log()` และฟังก์ชันกราฟิกคือ `hist()` เป็นต้น ผู้ อ่านจะสังเกตว่าการใช้ฟังก์ชันดังกล่าวในการทำงานช่วยให้ผู้ใช้ลดขั้นตอนในการทำงานที่ไม่ จำเป็นไปได้ นอกจากนี้ยังช่วยให้ syntax ของผู้เขียนโปรแกรมสั้นลง ทำงานได้ไวขึ้นและมี ประสิทธิภาพสูงขึ้น เช่น หากต้องการหาค่าสัมบูรณ์ของ -10 ในกรณีที่ไม่ได้ฟังก์ชันเข้ามาช่วยใน การประมวลผล ผู้วิเคราะห์จำเป็นต้องเขียนอัลกอริทึมเพื่อหาค่าสัมบูรณ์เองโดยอาจใช้คำสั่ง IF, ELSE เพื่อควบคุมเงื่อนไขการทำงานของ R ดังตัวอย่างคำสั่งด้านล่าง ซึ่งจะได้ผลลัพธ์เท่ากับ 10

```
x<-(-10)
## เขียนกระบวนการเพื่อหา absolute ของ x
if(x<0){-(x)} else {x}
```

```
[1] 10
```

อย่างไรก็ตามเมื่อเปรียบเทียบกับการใช้ฟังก์ชันเข้ามาช่วยในการทำงาน โดยในกรณีนี้ นำเอาฟังก์ชัน `abs()` เข้ามาช่วยคำนวณค่าสัมบูรณ์ ผู้อ่านจะเห็นว่าการเขียนคำสั่งลดลงเหลือเพียง บรรทัดเดียวเท่านั้น ดังนี้

```
abs(-10)
```

```
[1] 10
```

จากตัวอย่างในข้างต้นผู้อ่านจะสังเกตเห็นว่าการใช้ฟังก์ชันในการดำเนินงานช่วยลดขั้นตอนและประหยัดเวลาในการทำงานได้อย่างมาก ในสภาพแวดล้อมการทำงานบนโปรแกรม R **ฟังก์ชัน (function)** คือชุดคำสั่งสำเร็จรูปที่ถูกพัฒนาขึ้นสำหรับการทำงานเฉพาะด้าน การใช้ ฟังก์ชันในการดำเนินงานจะช่วยให้ผู้ใช้ประหยัดเวลา ลดความผิดพลาดในการทำงาน และทำให้ กระบวนการทำงานมีประสิทธิภาพมากขึ้น ฟังก์ชันในโปรแกรม R ไม่ได้จำกัดการใช้งานแต่ด้าน การคำนวณทางคณิตศาสตร์เท่านั้น แต่ยังมีฟังก์ชันที่สามารถใช้ดำเนินงานลักษณะอื่นได้อีกหลาย ประเภท เช่น การคัดเลือกตัวแปร การคัดกรองข้อมูล การสร้างแผนภาพหรือกราฟทางสถิติ และการประมวลผลเพื่อหาคำตอบในทางสถิติ เป็นต้น

ฟังก์ชันแต่ละตัวมีส่วนประกอบจำนวน 3 ส่วนหลัก ได้แก่ (1) ส่วนข้อมูลนำเข้า (input) ส่วนนี้เป็นส่วนที่ผู้ใช้โปรแกรมต้องกำหนดหรือกรอกเข้าไปในฟังก์ชันเพื่อควบคุมการทำงานให้เป็นไปตามที่ต้องการ (2) ส่วนประมวลผล (process) ส่วนนี้เป็นส่วนการทำงานเบื้องหลัง ปกติแล้วผู้ ใช้มักจะไม่เห็นการทำงานในส่วนนี้ของฟังก์ชัน การประมวลผลนี้จะดำเนินการโดยขึ้นกับชุดคำสั่งที่ ผู้พัฒนาได้กำหนดไว้ และข้อมูลนำเข้าที่ผู้ใช้ระบุ และ (3) ส่วนผลลัพธ์ (output) เป็นผลลัพธ์หรือ คำตอบที่ได้จากฟังก์ชัน ซึ่งอาจรายงานให้ผู้ใช้ทราบในหน้าต่าง Console ในทันทีที่ประมวลผล เสร็จสิ้น หรืออาจเก็บผลลัพธ์ดังกล่าวเอาไว้ในตัวแปร ซึ่งผู้ใช้จะต้องเรียกดูด้วยตนเองอีกครั้งหนึ่ง โดยปกติการเรียกใช้ฟังก์ชันใน R มีรูปแบบคำสั่งดังนี้

```
function_name(arg1, arg2, ...)
```

โดยที่ **function_name** คือชื่อของฟังก์ชัน และ **arg1** กับ **arg2, ...** เป็นส่วนข้อมูลนำเข้าของ ฟังก์ชันเรียกว่า อาร์กิวเมนต์ (argument) ใช้สำหรับป้อนข้อมูลที่จำเป็นและควบคุมการทำงานของ ฟังก์ชันเพื่อให้ผลลัพธ์เป็นไปตามที่ผู้ใช้งานต้องการ ทั้งนี้ฟังก์ชันสามารถมีอาร์กิวเมนต์ได้มากกว่าหนึ่ง ตัวขึ้นอยู่กับลักษณะงานของแต่ละฟังก์ชัน ยกตัวอย่างเช่น ฟังก์ชัน `log(x, base=exp(1))` ที่ มีอาร์กิวเมนต์ 2 ตัวได้แก่ **x** และ **base** เมื่อกำหนดค่าทั้งสองฟังก์ชันจะหาค่า logarithm ของ ค่า **x** เมื่อกำหนดฐานของ logarithm ให้มีค่าเท่ากับ **base** โดยในคำสั่งข้างต้นกำหนดให้ **base = exp(1)** ซึ่งมีค่าเท่ากับ $e \approx 2.71828...$ เรียกว่า natural logarithm ตัวอย่างด้านล่าง แสดงการหา ค่า natural logarithm ของ 10 ด้วยการใช้ฟังก์ชัน `log()` ข้างต้น

```
log(x = 10, base = exp(1))
```

```
[1] 2.302585
```

```
log(10)
```

```
[1] 2.302585
```

จากตัวอย่างข้างต้นผู้อ่านจะสังเกตเห็นว่าการเรียกใช้ฟังก์ชันใน R สามารถลดทอนการ เขียนอาร์กิวเมนต์บางตัวได้ ในกรณีที่อาร์กิวเมนต์นั้นถูกกำหนดค่าเริ่มต้น (default value) เอาไว้ จากตัวอย่างที่ผ่านมาจะเห็นว่า อาร์กิวเมนต์ base ถูกกำหนดค่าเริ่มต้น (default) ให้มีค่าเท่ากับ `exp(1)` ระหว่างการเขียนคำสั่ง `log(x, base=exp(1))` กับ `log(x)` จึงได้คำตอบเดียวกัน ดังนั้น อาร์กิวเมนต์ `base` จึงเป็นอาร์กิวเมนต์ที่สามารถละการเขียนได้

2.3 การเรียกคู่มือของฟังก์ชัน

R เป็นโปรแกรมที่มีฟังก์ชันให้เลือกใช้งานจำนวนมากในทางปฏิบัติจึงยากที่จะจำวิธีการใช้ ฟังก์ชันทั้งหมด การทำงานบนโปรแกรม R โดยปกติจึงมีการเรียกคู่มือการใช้ฟังก์ชันที่ใช้เป็นประจำ โดยผู้ใช้ R สามารถเรียกคู่มือของฟังก์ชันที่ต้องการได้โดยพิมพ์คำสั่ง ? ตามด้วยชื่อฟังก์ชัน หรือใช้ฟังก์ชัน `help()` เพื่อเรียกคู่มือดังกล่าว เช่น หากต้องการเรียกคู่มือการใช้ ฟังก์ชัน `log()` ข้างต้น สามารถพิมพ์คำสั่งได้ดังนี้

```
?log()
help(log)
```

ตัวอย่างด้านล่างแสดงคู่มือการใช้งานฟังก์ชัน `log()` ข้างต้น

R มีฟังก์ชันจำนวนมากจากหลาย library สำหรับการทำงานในด้านวิทยาการข้อมูล ในหนังสือเล่มนี้ผู้อ่านจะได้รู้จักและเรียนรู้การประยุกต์ใช้ฟังก์ชันต่าง ๆ ที่จำเป็นในกระบวนการวิเคราะห์ข้อมูล ตั้งแต่การทำความสะอาดข้อมูล การตรวจสอบข้อมูลที่ขาดหายไป การรวมและแยกข้อมูล การสร้างทัศนภาพข้อมูล ไปจนถึงการคำนวณทางสถิติต่าง ๆ นอกจากการใช้ฟังก์ชันที่ผู้อื่นได้สร้างเอาไว้ ในบางกรณีผู้วิเคราะห์อาจจำเป็นต้องสร้างฟังก์ชันของตนเองเพื่อใช้ในการทำงานหรือเพื่อแก้ปัญหาต่าง ๆ ซึ่งจะกล่าวถึงในประเด็นนี้อีกครั้งในส่วนท้ายของบทเรียนนี้ เนื้อหาส่วนถัดไปจะกล่าวถึงมีโนทัศน์ของตัวแปรในภาษา R ซึ่งพื้นฐานที่มีความสำคัญมากในการทำงานด้านสถิติและวิทยาการข้อมูล

2.4 ตัวแปร (variables)

คำว่า “ตัวแปร” ภายใต้สภาพแวดล้อมของภาษา R มีความหมายที่แตกต่างไปจากตัวแปรใน เิงการวิจัยหรือการวิเคราะห์ข้อมูลทางสถิติ กล่าวคือ ตัวแปรเป็นวัตถุ (object) ประเภทหนึ่งที่อยู่ภายใต้สภาพแวดล้อมของภาษา R มีหน้าที่บันทึก/เก็บข้อมูลหรือผลลัพธ์ที่ได้จากการประมวลผลเอาไว้ในหน่วยความจำ ของคอมพิวเตอร์ ซึ่งทำให้ผู้ใช้สามารถเรียกดูค่าที่เก็บไว้ดังกล่าวในภายหลังหรือนำไปใช้ต่อใน การดำเนินการขั้นตอนอื่น ๆ โดยไม่ต้องป้อนข้อมูลหรือประมวลผลใหม่ซ้ำ ๆ

ตัวแปรในภาษา R สามารถจำแนกได้หลากหลายประเภท ซึ่งทำให้การสร้างตัวแปร และการดำเนินการสำหรับตัวแปรแต่ละประเภทมีรายละเอียดที่แตกต่างกันในบางส่วน หัวข้อนี้จะ กล่าวถึงการสร้างตัวแปรพื้นฐานที่เรียกว่าตัวแปรสเกลาร์ (scalar) ที่ใช้เก็บข้อมูลได้หนึ่งค่าต่อ ตัวแปร การสร้างตัวแปรแบบสเกลาร์ใน R สามารถทำได้โดยใช้คำสั่ง `<-` (อ่านว่า assign) เช่น `x<-10` หมายถึงกำหนดให้ R เก็บค่าคือ 10 ที่อยู่ทางส่วนปลายของลูกศรไว้ในตัวแปรชื่อ `x` ที่อยู่ ทางส่วนหัวของลูกศร และเมื่อสร้างตัวแปร `x` ในข้างต้นแล้ว ผู้ใช้สามารถเรียกดูหรือใช้ค่าที่เก็บไว้ในตัวแปรได้โดยการเรียกชื่อของตัวแปรดังกล่าว ดังตัวอย่างต่อไปนี้

```
## assign 10 into variable 'x'
x <- 10
## print x
x
```

```
[1] 10
```

log (base) R Documentation

Logarithms and Exponentials

Description

`log` computes logarithms, by default natural logarithms, `log10` computes common (i.e., base 10) logarithms, and `log2` computes binary (i.e., base 2) logarithms. The general form `log(x, base)` computes logarithms with base `base`.

`log1p(x)` computes $\log(1 + x)$ accurately also for $|x| \ll 1$.

`exp` computes the exponential function.

`expm1(x)` computes $\exp(x) - 1$ accurately also for $|x| \ll 1$.

Usage

```
log(x, base = exp(1))
logb(x, base = exp(1))
log10(x)
log2(x)

log1p(x)

exp(x)
expm1(x)
```

Arguments

`x` a numeric or complex vector.

`base` a positive or complex number: the base with respect to which logarithms are computed. Defaults to $e = \exp(1)$.

Details

All except `logb` are generic functions: methods can be defined for them individually or via the [Math](#) group generic.

Figure 2.2: คู่มือการใช้งานฟังก์ชัน `log()` ที่ได้จากการพิมพ์คำสั่ง `?log()`

เมื่อทำการนิยามและกำหนดค่าให้กับตัวแปรแล้ว ผู้วิเคราะห์สามารถนำตัวแปรที่สร้างขึ้น ไปใช้ในการดำเนินการ หรือประมวลผลร่วมกับตัวแปรอื่น ๆ ต่อไปได้อย่างง่าย ดังตัวอย่างต่อไปนี้

```
## assign 10 into x
x<-10
## assign 5 into y
y<-5
## create z from x+y
z<-x+y
## print z
z
```

```
[1] 15
```

```
## create t from x,y,z
t<-exp(x+y)/z
## print t
t
```

```
[1] 217934.5
```

```
## find squart root of t
sqrt(t)
```

```
[1] 466.8345
```

หมายเหตุ

1. การกำหนดค่าให้กับตัวแปรนอกจากจะใช้ฟังก์ชัน assign (<-) ดังในตัวอย่างข้างต้นแล้ว ยัง สามารถใช้ฟังก์ชัน = ซึ่งให้ผลลัพธ์เหมือนกัน ข้อสังเกตที่น่าสนใจคือการกำหนดค่าให้กับ ตัวแปรด้วยฟังก์ชัน = ตัวแปรจะต้องอยู่ด้านซ้ายของฟังก์ชัน และค่าหรือข้อมูลที่ต้องการ กำหนดให้กับตัวแปรจะต้องอยู่ทางด้านขวา ในขณะที่การกำหนดค่าให้กับตัวแปรด้วย ฟังก์ชัน <- สามารถทำในลักษณะใดก็ได้เพียงแต่กลับหัวลูกศรตามตำแหน่งของตัวแปร เช่น `x<-4` หรือ `4->x` ซึ่งจะให้ผลลัพธ์ที่เหมือนกัน
2. การเขียนคำสั่งต่าง ๆ ใน R ผู้วิเคราะห์สามารถใช้สัญลักษณ์ # เพื่อช่วยจดบันทึกเตือน ความจำเกี่ยวกับคำสั่งที่ใช้ในการทำงาน โดยข้อความทั้งหมดที่อยู่ภายหลัง # จะไม่ถูกนำไปประมวลผล
3. การตั้งชื่อตัวแปรสามารถตั้งชื่อได้อย่างอิสระตามความต้องการของผู้ใช้ โดยสามารถ ประกอบได้ทั้งตัวอักษรและตัวเลข แต่มีข้อจำกัดในการตั้งชื่อคือห้ามขึ้นต้นชื่อตัวแปรด้วย ตัวเลขและอักขระพิเศษ เช่น !, @, #, \$, %, ^, &, * เป็นต้น นอกจากนี้อักขระ ตัวเล็กและตัวใหญ่ ภาษา R จะถือว่ามีความแตกต่างกัน (case-sensitive) เช่น

```
## assign 5 to x
y<-5
## assign 100 to x
Y<-100
## print y
y
```

```
[1] 5
```

```
## print Y
Y
```

```
[1] 100
```

2.4.1 ตัวแปรจำแนกตามลักษณะข้อมูล

ตัวแปรแบบสเกลาร์ยังสามารถจำแนกได้อีก 3 ประเภท ตามลักษณะของข้อมูลที่จัดเก็บไว้ในตัวแปร ได้แก่ ตัวแปรตัวเลข (numeric variables) ตัวแปรตัวอักษร (character variable) และ ตัวแปรตรรกะ (logical variables) รายละเอียดดังนี้

(1) **ตัวแปรตัวเลข (numeric variables)** ตัวแปรประเภทนี้ใช้จัดเก็บข้อมูลที่มีค่าเป็นจำนวนจริง (real number) และสามารถนำไปดำเนินการทางคณิตศาสตร์ได้ การสร้างตัวแปรที่เก็บข้อมูล ตัวเลขสามารถดำเนินการได้โดยใช้ฟังก์ชัน `<-` ผู้ใช้สามารถเรียกดูผลลัพธ์ที่เก็บไว้ในตัวแปรรวมทั้งค่าที่เก็บไว้ในตัวแปรไปดำเนินการในขั้นตอนอื่น ๆ ต่อไป ดังตัวอย่างในข้างต้น

(2) **ตัวแปรตัวอักษร (character variables)** ตัวแปรประเภทนี้ใช้จัดเก็บข้อมูลที่เป็นตัวอักษรหรือ ข้อความที่ไม่มีค่าในเชิงปริมาณ และไม่สามารถนำมาดำเนินการใด ๆ ทางคณิตศาสตร์ได้ การ สร้างตัวแปรประเภทนี้สามารถทำได้ในทำนองเดียวกับการสร้างตัวแปรตัวเลขโดยใช้ฟังก์ชัน `<-` เหมือนกัน แต่จำเป็นต้องเขียนเครื่องหมาย quotation ("`\"`") คร่อมตัวอักษรหรือข้อความที่ต้องการ จัดเก็บไว้ในตัวแปร ยกตัวอย่างเช่น หากต้องการสร้างตัวแปร `gender1` เพื่อเก็บข้อมูล `Male` และตัวแปร `gender2` เพื่อเก็บข้อมูล `Female` สามารถทำได้ดังนี้

```
## assign "Male" into gender1
## assign "Female" into gender2
gender1<-"Male"
gender2<-"Female"
```

```
## print gender1
gender1
```

```
[1] "Male"
```

```
## print gender2
gender2
```

```
[1] "Female"
```

ข้อสังเกตหนึ่งเกี่ยวกับตัวแปรตัวอักษรคือ ไม่สามารถนำตัวแปรตัวอักษรมาดำเนินการทางคณิตศาสตร์ได้ เนื่องจากตัวแปร ดังกล่าวไม่ได้มีความหมายในเชิงปริมาณ ถึงแม้ว่าข้อมูลที่เก็บอยู่ในตัวแปรตัวอักษรจะมีลักษณะที่ เหมือนกับตัวเลขก็ตาม ผู้อ่านลองหาผลบวกของตัวแปรต่อไปนี้

```
## find gender1 + gender2
gender1 + gender2
```

Error in `a + b` : non-numeric argument to binary operator

```
## assign character "1" into a
a<-"1"
## assign character "3" into a
b<-"3"
## find a+b
a+b
```

Error in `a + b`: ! non-numeric argument to binary operator

(3) **ตัวแปรตรรกะ (logical variables)** ตัวแปรประเภทนี้ใช้จัดเก็บข้อมูลที่เป็นค่าความจริงของ ประพจน์ (statement) โดยในทางคณิตศาสตร์ประพจน์คือข้อความที่สามารถระบุค่าความจริงของ ข้อความได้ว่าเป็นจริง (TRUE) หรือเป็นเท็จ (FALSE) การสร้างตัวแปรเพื่อเก็บข้อมูลตรรกะสามารถ อาจทำได้ 2 วิธีการ **วิธีการแรก** คือการสร้างตัวแปรตรรกะโดยตรงด้วยการป้อนข้อมูลค่าความจริงทีละค่าในทำนองเดียวกับข้อมูลตัวเลขและตัวอักษรโดยใช้คำสั่ง `<-` โดยข้อมูลค่าความจริงที่เป็นจริง กำหนดโดยค่า TRUE หรือ T ส่วนค่าความจริงที่เป็นเท็จกำหนดโดยค่า FALSE หรือ F ดังนี้

```
## assign TRUE into x
x<-TRUE
## print x
x
```

[1] TRUE

```
## assign F (FALSE) into y
y<-F
## print y
y
```

[1] FALSE

อย่างไรก็ตามในทางปฏิบัติมักไม่พบการสร้างตัวแปรตรรกะด้วยวิธีการข้างต้น ทั้งนี้เป็นเพราะในการทำงานจริงตัวแปรตรรกะมักใช้ประโยชน์ในการตรวจสอบเงื่อนไขเพื่อกำหนดทางเลือกในการประมวลผล ดังนั้นตัวแปรตรรกะส่วนใหญ่จึงมักถูกสร้างขึ้นจากกระบวนการตรวจสอบเงื่อนไขมากกว่า การสร้างตัวแปรตรรกะ**วิธีการที่สอง**จึงทำได้จากการสร้างผลลัพธ์ที่ได้จากการ ตรวจสอบเงื่อนไขด้วยตัวดำเนินการเชิงตรรกะ (logical operator) ได้แก่

- `<` (น้อยกว่า)
- `>` (มากกว่า)
- `<=` (น้อยกว่าหรือเท่ากับ)
- `>=` (มากกว่าหรือเท่ากับ)
- `==` (เท่ากับ)
- `!=` (ไม่เท่ากับ)

ดังตัวอย่างต่อไปนี้

```
# assign 65 to student
student1 <- 65
```

```
# Is student1 greater than 50?
student1 > 50
```

```
[1] TRUE
```

```
# Is student1 equal to 50?
student1 == 70
```

```
[1] FALSE
```

จากตัวอย่างข้างต้นจะเห็นว่าการสร้างตัวแปร student1 เพื่อเก็บคะแนนที่มีค่าเท่ากับ 65 จากนั้นมีการใช้ตัวดำเนินการตรรกะเพื่อตรวจสอบเงื่อนไขจำนวน 2 เงื่อนไข ดังนี้ (1) คะแนนที่เก็บไว้ในตัวแปร student1 มีค่ามากกว่า 50 คะแนนหรือไม่ และ (2) คะแนนใน student1 มีค่าเท่ากับ 70 คะแนนหรือไม่ จะเห็นว่าผลลัพธ์ที่ได้จากการตรวจสอบเงื่อนไขทั้งสองคือค่าความจริงที่มีค่าเป็นไปได้ 2 ค่าคือ **TRUE** หรือ **FALSE** เท่านั้น และจากการกำหนดเงื่อนไขข้างต้นจะได้ว่า เงื่อนไขแรกมีค่าความจริงเท่ากับ **TRUE** และเงื่อนไขที่สองมีค่าความจริงเท่ากับ **FALSE** ตามลำดับ

เนื่องจากค่าความจริงที่ประมวลผลได้นั้นเป็นข้อมูลตัวหนึ่งภายใต้สภาพแวดล้อมของ R ผู้ใช้จึงสามารถเก็บค่าของข้อมูลดังกล่าวไว้ในตัวแปรเช่นเดียวกับการสร้างตัวแปรตรรกะในวิธีการที่หนึ่ง ดังตัวอย่างต่อไปนี้

```
result1 <- student1 > 50
result2 <- student1 == 70
```

```
result1
```

```
[1] TRUE
```

```
result2
```

```
[1] FALSE
```

การตรวจสอบเงื่อนไขของตัวแปรดังกล่าวมีประโยชน์หลายประการในการทำงานด้านสถิติและวิทยาการข้อมูล เช่น การคัดกรองหรือสำรวจข้อมูลด้วยการกำหนดเงื่อนไข หรือการประมวลผลที่มีความซับซ้อนหรือมีหลากหลายกรณี ยกตัวอย่างเช่น ผู้วิเคราะห์มีข้อมูลคะแนนสอบของนักเรียนหลายคนเก็บบันทึกอยู่ในเวกเตอร์ **vector_data**(รายละเอียดเรื่องเวกเตอร์จะกล่าวในส่วนถัดไป) ดังนี้

```
vector_data <- c(10,30,50,30,20,60,70,80,10,20,60)
vector_data
```

```
[1] 10 30 50 30 20 60 70 80 10 20 60
```

หากผู้วิเคราะห์ต้องการทราบว่านักเรียนกี่คนที่มีคะแนนสอบตก (ต่ำกว่า 50 คะแนน) สามารถใช้ตัวแปรตรรกะเข้ามาช่วยสำรวจได้ดังนี้

```
## create logical vector to represent student exam results
fail_student <- vector_data < 50
fail_student
```

```
[1] TRUE TRUE FALSE TRUE TRUE FALSE FALSE FALSE TRUE TRUE FALSE
```

จะเห็นว่านักเรียนที่สอบตกคือนักเรียนที่มีผลลัพธ์จากการตรวจสอบเงื่อนไขเป็น TRUE เราอาจนับจำนวน TRUE ได้จากการแจกแจงความถี่ผลลัพธ์ใน `fail_student` ด้วยฟังก์ชัน `table()` ซึ่งผลการแจกแจงความถี่ด้านล่างจะเห็นว่ามึ้นักเรียนที่สอบตกจำนวน 6 คน จาก 11 คน

```
## tally student who fail and pass
table(fail_student)
```

```
fail_student
FALSE TRUE
      5    6
```

2.4.2 ตัวแปรจำแนกตามโครงสร้าง

นอกจากการจำแนกตัวแปรตามลักษณะของข้อมูลแล้ว อีกมุมมองหนึ่งคือการจำแนกตาม โครงสร้างการจัดเก็บข้อมูล ซึ่งมีหลายประเภท ตัวแปรสเกลาร์ (scalar) ที่ได้กล่าวในรายละเอียด ไปแล้วในหัวข้อก่อนหน้านี้ เป็นตัวแปรที่มีโครงสร้างซับซ้อนน้อยที่สุดเพราะสามารถเก็บข้อมูลได้ เพียงตัวแปรละ 1 ค่าเท่านั้น ในบทเรียนนี้จะกล่าวถึงตัวแปรที่มีโครงสร้างการเก็บข้อมูลที่ซับซ้อน มากขึ้นอีกหลายประเภท ได้แก่ เวกเตอร์ (vectors) เมทริกซ์ (matrices) ตัวแปรแบบลิสต์ (list) และ ชุดข้อมูล (dataframe) รายละเอียดมีดังนี้

(1) เวกเตอร์ (vectors)

เวกเตอร์ คือตัวแปรที่มีโครงสร้างสำหรับจัดเก็บข้อมูลคล้ายกับตารางที่มีจำนวนหนึ่ง คอลัมน์ กล่าวคือ เวกเตอร์เป็นตารางที่มีมิติเท่ากับ $n \times 1$ โดยที่ n คือจำนวนสมาชิกของเวกเตอร์ หากกำหนดให้ \vec{u} คือเวกเตอร์ที่มีขนาด 5×1 โดยที่ 1, 4, 6, 4 และ 8 คือสมาชิกภายในเวกเตอร์ ในทางคณิตศาสตร์จะสามารถเขียนสัญลักษณ์แทนเวกเตอร์ \vec{u} ได้ดังนี้

$$\vec{u} = \begin{pmatrix} 1 \\ 4 \\ 6 \\ 4 \\ 8 \end{pmatrix}_{5 \times 1}$$

จากลักษณะของเวกเตอร์ข้างต้นจะเห็นว่าโครงสร้างการจัดเก็บข้อมูลที่ยอมให้ผู้วิเคราะห์สามารถเก็บข้อมูลในตัวแปรเดียวกันได้มากกว่าหนึ่งค่า ซึ่งสามารถนำไปใช้เป็นโครงสร้างสำหรับการเก็บข้อมูลของตัวแปรที่สนใจได้หลายหน่วยข้อมูล

53 69 52 62 57 54 55 67 53 58




Figure 2.3: ตัวอย่างคะแนนสอบวิชาสถิติของนิสิตจำนวน 10 คน

การสร้างเวกเตอร์เพื่อจัดเก็บข้อมูลในโปรแกรม R สามารถทำได้หลายวิธี วิธีการพื้นฐาน คือการใช้ฟังก์ชัน `concatenate c()` เพื่อต่อเชื่อมข้อมูลหลายค่าเข้าด้วยกันให้อยู่ในรูปแบบของ เวกเตอร์ จากนั้นใช้ฟังก์ชัน `<-` เพื่อสร้างตัวแปรสำหรับจัดเก็บเวกเตอร์ที่สร้างขึ้นในข้างต้น ยกตัวอย่างเช่น ต้องการเก็บข้อมูลคะแนนสอบวิชาสถิติของนิสิตจำนวน 10 คน ดังรูป 2.3 ไว้ในตัวแปรแบบเวกเตอร์ของภาษา R และตั้งชื่อว่า `score` สามารถเขียนคำสั่งได้ดังนี้

```
## create score vector
score <- c(53,69,52,62,57,54,55,67,53,58)
## print score
score
```

```
[1] 53 69 52 62 57 54 55 67 53 58
```

ตัวอย่างข้างต้นจะเห็นว่าการพิมพ์ชื่อของเวกเตอร์เป็นการเรียกดูสมาชิกทั้งหมด ภายในเวกเตอร์นั้นเหมือนกับการเรียกดูตัวแปรสเกลาร์ นอกจากนี้เวกเตอร์มีข้อมูลที่จัดเก็บได้ หลายตัวจึงทำให้สามารถเรียกดูข้อมูลบางส่วนหรือทั้งหมดของเวกเตอร์ก็ได้ ในกรณีที่ต้องการเรียกดูสมาชิกเพียงบางส่วนของเวกเตอร์สามารถทำได้โดยใช้ลำดับของสมาชิกที่ต้องการภายในเวกเตอร์นั้นเป็นตัวอ้างอิงสมาชิกที่ต้องการ รูปแบบของคำสั่งประกอบด้วยชื่อของเวกเตอร์แล้วตามด้วยเครื่องหมาย `[i]` โดยที่ `i` คือลำดับของสมาชิกที่ต้องการ เช่น จากเวกเตอร์ `score` หากต้องการเรียกดูคะแนนสอบของนิสิตคนที่ 3 สามารถเขียนคำสั่งเป็น `score[3]` หรือหากต้องการ เรียกคะแนนสอบวิชาสถิติของนิสิตคนที่ 5, 6, ..., 9 สามารถเขียนคำสั่งเป็น `score[5:9]` ผลลัพธ์ที่ได้จะขึ้นอยู่กับจำนวนสมาชิกที่สอดคล้องกับเงื่อนไขที่กำหนด ดังนี้

```
## filter 3rd element from score
score[3]
```

```
[1] 52
```

```
## filter 5th-9th elements from score
score[5:9]
```

```
[1] 57 54 55 67 53
```

หมายเหตุ : เป็นตัวดำเนินการหนึ่งในภาษา R ที่ใช้สร้างลำดับเลขคณิตอย่างง่าย โดย `a:b` จะได้ผลลัพธ์เป็นลำดับเลขคณิตที่มีพจน์แรกและพจน์สุดท้ายเป็นเลข `a` และ `b` ตามลำดับ โดยที่สมาชิกที่อยู่ระหว่างตัวเลขทั้งสองมีระยะห่างหรือผลต่างร่วมที่เท่ากับ 1 ทั้งนี้ผลลัพธ์ที่ได้จะอยู่ในสถานะเวกเตอร์ ดังนั้น `5:9` ในตัวอย่างข้างต้นจึงหมายถึงการสร้างเวกเตอร์ที่มีสมาชิกเป็น 5, 6, 7, 8 และ 9 ตามลำดับ

```
5:9
```

```
[1] 5 6 7 8 9
```

ในกรณีต้องการคัดกรองสมาชิกที่ไม่ได้เรียงกันเป็นลำดับ เช่น ต้องการคัดกรองให้เหลือเฉพาะคะแนนสอบของนิสิตคนที่ 2, 5, 7 และ 10 จากเวกเตอร์ `score` ผู้วิเคราะห์สามารถสร้างเวกเตอร์ `element` เพื่อระบุตำแหน่งของสมาชิกในเวกเตอร์ `score` ที่ต้องการเลือก จากนั้นจึงใช้เวกเตอร์ `element` เป็นตัวคัดกรอง ดังตัวอย่าง

```
## create element vector
element <- c(2, 5, 7, 10)
## using element to filter score
score[element]
```

```
[1] 69 57 55 58
```

การอ้างอิง/คัดกรองสมาชิกภายในเวกเตอร์ข้างต้น สามารถนำมาใช้เพื่อแก้ไขหรือเปลี่ยนแปลงค่าของสมาชิกภายในเวกเตอร์ได้อีกด้วย โดยใช้จํานวนร่วมกับตัวดำเนินการ `<-` ยกตัวอย่างเช่น หากพบว่าในเวกเตอร์ `score` มีการบันทึกคะแนนสอบของนิสิตคนที่ 6 คลาดเคลื่อนไป โดยที่ถูกต้องจะต้องมีค่าเท่ากับ 60 คะแนน สามารถดำเนินการแก้ไขและบันทึกค่าใหม่ได้ดังนี้

```
## assign new value to 6th element of score
score[6] <- 60
## print score
score
```

```
[1] 53 69 52 62 57 60 55 67 53 58
```

ในการทำงานเกี่ยวกับตัวแปรสเกลาร์ เราอาจจำแนกเวกเตอร์ได้เป็น 3 ประเภท ได้แก่ เวกเตอร์ตัวเลข (numeric vectors) เวกเตอร์ตัวอักษร (character vectors) และเวกเตอร์ตรรกะ (logical vectors) ทั้งนี้การสร้างเวกเตอร์ทั้ง 2 ประเภทที่เหลือมีลักษณะที่เป็นไปในหลักเดียวกับการสร้างตัวแปรตัวอักษร และตรรกะ กล่าวคือการสร้างเวกเตอร์ตัวอักษรสามารถทำได้โดยใช้ตัวดำเนินการ `<-` ร่วมกับ `c()` เหมือนเดิม แต่สมาชิกแต่ละตัวที่เป็นข้อความหรือตัวอักษรจะต้องถูกระบุไว้ภายใต้เครื่องหมาย quotation (" ") ในทำงานองเดียวกับการสร้างตัวแปรตัวอักษร ดังตัวอย่าง

```
## create gender vector
gender <- c("M", "F", "M", "M", "M", "F", "M", "F", "F", "M")
## print gender
```


Chapter 3

Summary

In summary, this book has no content whatsoever.

1 + 1

[1] 2

