第 1 章: R 系統導論

1: Introduction to R System

R 系統是由 Ross Ihaka 與 Robert Gentleman 從 S 語言所發展出來, 主要是爲了統計分析與統計繪圖. R 除了資料處理與分析, R 擁有一完整陣列和矩陣的操作運算, 完整圖形工具, 也是一種相當完善的程式設計語言. S 語言在 1980 年代末期, 由 AT&T 實驗室, Rick Becker, John Chambers, 與 Allan Wilks 發展用來進行統計分析與統計繪圖, 1990 年代, Insightful 公司將 S 商品化, 並加入許多方便的操作介面, 稱爲 S-PLUS. R 與 S (或 S-PLUS) 語法大多相近, R 與 S 都是以物件導向爲主的程式語言, 透過交互作用方式很容易地進行統計分析與統計繪圖, 這與 SAS, SPSS 的方式有所不同. R 可視爲統計數學軟體, 也是一種程式語言. 但是 R 是一個免費的統計分析軟體 (open-source, GNU General Public License), R 目前由一群跨國際的志工人員組成的 R 核心發展組織 (R core-development team) 所維持, 運作與持續更新發展, 在今日, R 不僅是統計專業人員的研究工具, R 更已經是許多研究領域進行量化研究的重要工具, 例如, 財務與經濟計量, 化學計量, 大氣科學與氣候分析, 空間與地理資訊分析, 語言學, 生態學, 社會科學, 遺傳與生物資訊等研究領域 與 R 的相關書籍已有數百本以上, 且持續在增加中. R 計畫的網址在 http://www.r-project.org , 在這網址上可獲得許多研究領域相關的 R 資訊.

1.1 下載與安裝 R

R 有各種版本, 可以在 Microsoft Window XP, Unix, Linux, Apple Mac OS 等作業性系統運行, 以下則以 windows 作業系統爲主要說明. 若你的 windows 使用者名稱或帳號爲中文, 請先進入控制台, 新增或變更 “使用者名稱” (User), 更改使用者名稱爲英文, 然後才進行以下安裝. 任何與 R 的檔案名稱或路徑名稱, 請勿使用 “中文”, “空格” 或 “_” (underscore). R 與其他軟體成可以形成強大的 可重製性研究論文 或 動態文件 (Reproducible Research, Dynamic /Documentation), 可同時依序安裝 Rtools, Rstudio, Tex System, Pandoc, Git 等相關軟體 (software). 初學者請請讓軟體內設的路徑 (PATH) 自動安裝, 不要任意更動 PATH 與 安裝位置. 安裝任何與 R 相關軟體, 請按滑鼠右鍵以系統管理員身分執行.

(R 時常有更新版本), 下載與安裝 R 與 Rtools 簡述如下:

1. 上網至 http://www.r-project.org .
2. 按滑鼠點選網頁左邊連結 (Link) 下載區 **Download “CRAN”** .
3. 按滑鼠點選網頁 CRAN Mirrors 中的任一鏡像網址 (CRAN Mirrors), 如 https://cloud.r-project.org/. 按滑鼠點選上述鏡像網址內網頁中的 Download R for Windows.
4. 按滑鼠點選網頁 R for Windows 中的 **base**.
5. 按滑鼠右鍵, 點選檔案網址 Download R X.Y.Z for Windows, 儲存至個人檔案夾內.
6. 至下載的檔案夾內, 按滑鼠右鍵點擊 **R-X.Y.Z-win.exe**, 以系統管理員身分執行安裝.
7. 可選擇中文或英文進行安裝.

8. 在硬碟空間許可下, 請將 base 套件全部安裝.

9. 回到與 **base** 同一網頁視窗, 點選 Rtools. 點選 Rtoolsxx.exe. 下載檔案, 請按滑鼠右鍵以系統管理員身分執行安裝.

10. 詳細安裝 Windows, Mac 或 Linux 作業系統, 利用 google 或 Youtube 等, 搜尋相關訊息.

1.2 安裝 RStudio

直接在 R 視窗輸入 R 指令並不是很方便. 使用文字編輯軟體輸入, 修正, 儲存, 然後再送到 R 執行. 許多 R 文字/程式 編輯軟體可用 google keyword `R programming text editor`.

RStudio 是 R 的一種操作介面, 上網至 http://www.rstudio.com/, 點選 RStudio, 點選 Desktop, 點選 DOWNLOAD RSTUDIO DESKOP, 點選電腦的作業系統, 下載 Rstudio 檔案, 請按滑鼠右鍵以系統管理員身分執行安裝.

若想使用 Tex/LaTeX, 或 讓 Rstudio 產生 PDF 檔案, 需依序安裝 Tex system, Pandoc, Git, 安裝 TeX/LaTeX/XeLaTex 系統, 請至 https://www.latex-project.org/get/ 或直接安裝 MikTeX: http://miktex.org/. 安裝 Pandoc, 請至 http://pandoc.org/ 或 http://pandoc.org/installing.html. 安裝 Git, 請至 https://git-scm.com/. 任何軟體請按滑鼠右鍵以系統管理員身分執行安裝. 詳細的可重製性研究論文操作, 請參看 Xie (2015), Gandrug (2015).

書本內容以 "C:\RData" 爲工作路徑, 請選定作業區硬碟 (或 隨身碟) "C:", 在 C 碟內新增資料夾, "C:\RData". Mac/Unix/Linux: 設定工作路徑, 請 google. 請勿使用中文路徑. 任何步驟錯誤, 請移除 R 與 Rstudio, 重新安裝.

1.3 簡單實例

學習 R 最好的方法, 就是要開始使用 R, 初學者要了解 R, 可先進行一些簡單實例的演練, 首先須注意的是, 在 R 程式語言所使用的文字, 大小寫是有差別的. 直接在 R 視窗 或 RStudio console 輸入指令都會得到相同的結果.

- 請開啟 R 或 RStudio 程式.
- 請在 R 或 RStudio 的互動式窗 Console 輸入以下指令, 並按 <Enter> 鍵.

```
> 1+2         # calculator
> log(3.14)
> x = 1+2     # one plus two assign to x
> x           # print x
> x = c(1, 3, 5, 7) # get a vector
> mean(x)     # function
> log(x)      # function
```

- 若要自學基本 R 或 進階 R, 利用 google 搜尋 R Introduction, R Tutorial 等, 或 搜尋與觀看 YouTube 影片. 以下爲上述指令出現的結果.

```
> ## first R for fun
> 1+2         # calculator
[1] 3
> log(3.14)
[1] 1.144223
> x = 1+2     # one plus two assign to x
> x           # print x
[1] 3
> x = c(1, 3, 5, 7) # get a vector
> mean(x)     # function
[1] 4
> log(x)      # function
[1] 0.000000 1.098612 1.609438 1.945910
```

測驗 **1.** 請練習下列指令.

```
> factorial(4)
> sin(pi)
> x.vec <- c(2:5)
> exp(x.vec)
> matrix(c(1:6), nrows = 2, ncols = 3)
```

```
> weight = c(50, 45, 67, 53)
> mean(weight)
> sd(weight)
```

R 具有強大的繪圖功能, 可將程式語言與繪圖函數緊密結合, 控制標題, 顏色, 說明文字等, 繪製高品質的統計圖並可儲存於不同格式, 提供不同文書處裡軟體使用. 練習以下指令可以體會一些 R 的繪圖功能.

```
> demo(graphics)
> demo(image)
> example(contour)
> demo(persp)
> example(persp3d)
> demo(plotmath)
> demo(Hershey)
> install.packages(""lattice) # install package
> library("lattice")          # load package
> demo(lattice)
> example(wireframe)
> install.packages("rgl")
> library("rgl")
> demo(rgl)                   # Interact using your mouse.
```

1.4 物件與常用指令

R 是統計程式語言, 程式語言是語言的一種, 程式語言是人與電腦溝通的工具, 任命令電腦做事的語言. 程式語言是須嚴謹的邏輯語法, 電腦才會正確執行命令,

R 對於未有統計專業背景的初學者是困難學習的語言, 初學者學習 R 語言, 類似小孩學習語言, 通常須 0.5~1.0 年以上, 初學者請先直接學習現成實用的簡單指令, 初學者請不要先學太多抽像語法與閱讀抽像說明, 若已經學習其它一種程式語言相對較容學會易另一種程式語言. 學習程式語言最大挫折是城市經常出現無法理解的錯誤訊息, 可利用 google, 將錯誤訊息尋找容易理解的答案. R 最常犯的錯誤如下:

- 英文大小寫差異.
- 英文單字拚錯.

- 缺少標點符號: 單/雙引號, 逗號, $, },],).
- 變數名/物件名, 指令, Raw Data: 多餘空白, 多餘空格, 多餘 Tab 等.
- 直接從 PDF 文件 或 Web 等 copy 程式造成亂碼.
- 未詳細閱讀錯誤訊息或警告訊息.

1.4.1 物件

R 與 S 是物件導向爲主的程式語言, (Object-Oriented Programming Language), R 中, 儲存的資料或可執行的函數, 都稱爲 “物件” (object). R 物件包含儲存資料的向量 (vector), 矩陣 (matrix), 陣列 (array), 列表 (Lists), 資料框架 (data frames) 或執行特定運算指令的函式 (function) 等. 初學者對於 R 程式中物件名詞較難理解, 可將 R 程式物件 與 賣場或電腦檔案類比如表 1.1.

表 1.1: R 程式物件類比

賣場	品項 (飲料, 家電, ...)	黑松汽水, Sony 電視
電腦	路徑, 檔案	路徑名稱, 檔案名稱
R	物件 (向量, 矩陣 „,)	物件名稱

1.4.2 物件命名

R 透過函式或指令, 很容易地對資料物件進行統計分析與統計繪圖. 須特別注意, 在 R 對物件或指令命名的英文大小寫是有差異, `s` 與 `S` 是不同的. 對物件命名時, 物件 名字 (name) 起始位置必須以 ”文字” 或 “.” (句點) 命名, 若物件 名字 以 “.” 爲起始, 則 名字 的第二個位置需爲文字, 物件名字其餘位置, 以文字 (A-Z 或 a-z), 數字 (0-9), “/”, “.”, 或 “-”, 等皆可. 中間不可有空格或 “_” (underscore).

R 也保留一些特定名字做爲特定的物件名字, 指令名字或常用函式名字使用, 例如, `c`, `s`, `C`, `T`, codeF 等, 這些名稱叫做 “保留 名字” (reserved names). 例如:

```
FALSE Inf NA NaN NULL TRUE
break else for function if in next repeat while
F T
c q s t C D I
diff mean pi range rank var
```

初學者對物件命名時, 應盡量避免定義一個物件名字, 與現有的物件同名, 所以命名時要避免重覆, 同時避免物件名字過短, 以免後來引起錯亂.

1.4.3 使用指令

R 基本介面是一個互動式指令視窗, 指令可分成 2 種, 一爲 運算式 (expression), 例如,

```
1+2
log(x)
mean(x)
```

另一個爲 指派運算 或 賦值運算 (assignment), 例如,

```
> x <- 1+2
> x = 4-5
```

當一個 R 程式需要使用者輸入指令時, 它會顯示 指令提示符號 (prompt symbol), 指令提示符號通常是一個 "`>`" (大於符號). 當使用者輸入完整的 運算式, 則運算式指令輸入後的結果, R 會馬上顯示在指令下方. 當使用者輸入完整的 指派運算, R 同樣會做運算式, 並且把結果 (值) 傳給變數, 但結果不會自動顯示在 R 視窗螢幕上. 在 R 中可利用 `options(prompt = "R>")` 將指令提示符號 `>` 改成 `R>`.

指派運算符號 (assignment symbol) 通常是 " `<-` ", 一個小於符號和一個短線符號組成, 例如, `x <- 1+2`, 讀成 `x` "得到" $(1+2)$. 在 R 中可以如傳統的程式語言或統計軟體, 使用 " `=` " (等號) 爲 指派運算符號, 例如, `x = 1+2`, 但在 R 中, " `=` " (等號) 還有其他用途, 使用者可依個人習慣使用 " `<-` " 或 " `=` ", 但是多數 R 專業人士建議使用 " `<-` ".

要顯示賦值運算的結果, 可以輸入物件名, 使用函式 `print`, 或在賦值運算時, 前後加上小括號顯示賦值運算的結果.

```
## assign
> x <- 1   # assign object x
> x        # show x
[1] 1
> print(x) # print()
[1] 1
> msg <- "hello"
> msg     # show x
[1] "hello"
```

如果一條輸入的指令在第一行結束的時候, 在 R 語法上還不完整, 若用鍵盤上 <Enter> 按鍵時, 則 R 會給出另一個不同的提示符號, 通常是 “+” (加號), 且該提示符號 “+” 會出現在第二行, 和隨後的數行中, R 持續地等待使用者輸入指令. 當一指令在語法上是完整的時候, 使用鍵盤上 <Enter> 按鍵時, R 才會執行指令. 不同的完整指令要在同一行輸入時, 可用 ; (分號) 隔開, 或是另起一新輸入行輸入指令分別輸入不同的完整指令. 例如,

```
> ## input at the same line, use ;
> x <- 1+2; y <- 3+4
> ## input 2 lines separately
> x <- 1+2
> y <- 3+4
```

數個指令也可以放入一組大括弧內, { }, 數個指令放在一起, 構成一個 複合運算式 (compound expression), 這部份在函式的章節會再進一步說明.

在 R 中, 若要對任何指令, 物件, 程式語言加上 注釋 (commands), 則注釋從 # (井號) 開始, 到句子收尾之間的語句就是是注釋, 在 R 中, 注釋幾乎可以放在任何地方的任何一行之中. 習慣上, 整行的注釋使用雙井號作爲開始, ##, 運算式尾端注釋使用單井號開始 #.

```
> ## This is my R code
> log(pi)
> ## simple calculation
> 3+4 # calculator: two plus one
```

在 R 互動式窗 Console 中, 若要重複一個指令, 或是叫回之前輸入的指令, 可以用鍵盤上的 ↑ (“向上”) 箭頭按鍵, 調出之前已經輸入的指令, 視窗中便可顯示之前的輸入指令, 再利用鍵盤上 <DEL> 按鍵更改成所要輸入的指令. 可以再次練習以下

指令, 檢視 R 會傳結果.

```
> # This is my R code
> x = 1+2  # one plus two
> x
> x+4
> x-1
```

測驗 **2.** 請練習下列指令.

```
(y.vec <- c(1:5))  # ass ()
y.vec              # show y
print(y.vec)       # print (show) y
```

1.4.4 物件顯示

在 R 中產生和控制的實體稱爲 “物件” (“object”), 它們可以是向量, 陣列, 字串, 函式等不同型式. R 函式 `object()` 或 `ls()` 可以顯示當前保存在 R 環境中的物件名稱.

```
## show objects
> object() # show all objects used currently
> ls()     # show all objects used currently
> ls(x,y)  # show whether object x and y exist
```

透過函式 `rm()`, 可以刪除物件, 例如,

```
> ## delete objects
> rm(x.vec, y.vec) # delete x.vec and y.vec
```

可以刪除物件 `x.vec` 與 `y.vec`.

1.4.5 程式中止操作

當程式寫作不當, 造成 R 永無止境的執行運算. 若要中斷執行中的程式, 可按下 <Esc> 鍵中斷執行中的程式. 例如, 輸入

```
> for (i in 1:1000000) print (i) # press <Esc>
```

按下 <Esc> 鍵可中斷執行中的程式.

1.5 工作目錄

任何 R 工作中產生的物件或暫時檔案, 都會在電腦的 工作目錄 或 工作路徑 (working directory). R 指令內的路徑 (PATH) 分隔爲 // (例如, "C://RData//") 或 / ("C:/RData/"). 可用 Windowns 的 \\ ("C:\\RData"). 指令 `getwd()`, 可以顯示當前 工作目錄. 指令 `setwd()`, 可以改變當前 工作目錄. 請練習以下指令.

```
> getwd() # show your current working directory
> setwd("C:/RData/")
> getwd()
[1] "C:/RData"
```

測驗 **3.** 請再比較下列比較檔案路徑格式指令.

```
setwd("C:/RData")
getwd()
setwd("C:/RData/")
getwd()
```

使用 R 做統計資料分析, 不同的分析資料計畫, 最好使用不同的工作目錄, 以免相同名字的物件, 相互取代, 也便於管理. 在分析資料過程中, 將物件命名爲 `age`, `gender`, `m1.lm`, `m2.lm` 等, 是一件常有的情形, 在任一次的分析計畫中, 這樣的命名是有其特定含義, 但不同分析資料計畫, 在一個工作目錄下進行時, 區別資料內相同物件名字, 是一件非常困難的事情.

1.6 文字編輯軟體與整合系統 RStudio

R 有簡單的內建文字編輯器, 先從程式視窗上端, 點選表單 "編輯", 最下方 "GUI 偏好設定", 在 GUI 偏好設定, 可點選 "MDI mode" 之後, 則在 R 視窗上端表單 "檔案" 中, 可點選 "建立新的指令檔案" 或 "開啟指令檔案", 就可開啟一 (新) 文字檔案, 容許輸入指令, 當點選並反白所要執行的指令程式碼, 再點選視窗上端表單下, 小的執行圖示 "執行程式列" (圖示類似符號 `< = >`), 即可執行指令.

使用者另外可以用 記事本 (Notepad) 或 Wordpad 等文字編輯軟體, 可先寫入指令程式碼, 然後點選並反白所要執行的指令程式碼, 在文字編輯軟體點選複製, 然後在 R 點選小的執行圖示 “貼上”, 即可執行指令.

有許多文字編輯軟體用來支援 R 的使用, 目前多數使用 RStudio 介面最爲方便, 且目前使用 RStudio 與 GitHub, RPubs 整合最爲方便. RStudio 是 R 的一個程式編輯與執行 GUI 介面. 下載安裝 RStudio 後, 若要方便使用中文以及可重製性研究論文系統, 可更動下列選項. 開啟 RStudio:
使用滑數點選 `Tools`, 點選 `Global Options....`.

- 點選 `General`, 取消點選 `Restore .RData`, 在 `Save workspace to .RData on exit`: 點選 `Never`.
- 在 `Default text encoding`: 點選 `UTF-8`.
- 點選進入 `Appearance`, 點選你喜歡的式樣, 例如, 點選 `Zoom`: 140%, 點選 `Font size`: 14, 點選你喜歡的字型.
- 點選進入 `Sweave`, 在 `Waeve Rnw file using`: 點選 `knitr`, 在 `Typest LaTeX into PDF using`: 點選 `XeLaTeX`.
- 最後記得點選 `Apply` 或 `OK`.

接著開始使用 RStudio 撰寫第一個 R 程式檔案.

1. 開啟 RStudio, 檢視 RStudio 在左上角視窗最上端工具列.
2. 點選 `File` → `New File` → `R Script`, 開啟一新的 R 程式檔案.
3. 同樣在左上角視窗最上端工具列, 點選 `File`, 存檔案 (`Save as`), 在檔案夾 “C:\RData”, 儲存成爲 Rlab00.r 檔案.
4. 以 **.r** 或 **.R** 爲附檔名, 代表 R 程式檔案.
5. 在 `source` 視窗, 輸入以下程式.

6. 輸入完成後, 同樣在左上角視窗最上端工具列, 點選 File, 存檔案 Save.

7. 上述操作部分指令有捷徑可使用.

```
> ## Rlab00.r
> x <- 1
> print(x)
> x
> msg <- "hello"
> msg
> y <- 1:20
> y
> rm(x, msg, y)
```

將程式輸入在 R 程式檔案內, 將要執行的程式行列, 以滑鼠反白, 複製 (copy) 到 RSudio Console 視窗, 執行程式. 可以滑鼠反白, 同時按 <control>+<Enter> 鍵, 執行程式. 使用 R 程式檔案, 可保有 R 程式, 隨時修正或重複執行. 將下列指令輸入在 RStudio 新建立的 Rlab00.r 檔案內. 請再次練習下列指令, 並執行致令, 在 RSudio Consol 視窗檢視結果.

```
> ## Rlab00.r
> setwd("C:/RData")
> 2.4*3.8
> x.vec = rnorm(50)
> y.vec = rnorm(50)
> plot(x.vec, y.vec)
> ls()
> rm(x.vec, y.vec)
> ls()
```

學習 R 最好的方法, 就是開始使用 R. 初學者要了解 R, 可先進行一些簡單實例的演練. 在 R 程式語言所使用的文字大小寫是有差別的. 打開 Rstudio, 有 3-4 個小視窗. 在 RStudio 左上視窗輸入 R 程式. 每一行依次輸入依序執行, 檢視結果.

- 輸入一行後, 按 <Enter> 鍵執行程式.
- 或 輸入一行後, 同時按 <Ctrl>+<Enter> 鍵執行程式.

另外可以得等整批指令輸入後, 將所有程式反白, 使用上述方式同時執行.

```
> ## 1st section: for fun
> ## Suppose you want to put R data in hard disc C:
```

```
> ## Creat a folder in C:/Rdata
> setwd("C:/RData")
> help.start()
> 4.5*7.25 + 2.1/4.1
> x.vec = rnorm(50)
> y.vec = 0.5*x.vec + rnorm(50)
> par(mar = c(4, 4, 4, 1))
> plot(x = x.vec, y = y.vec)
> ls()
> rm(x.vec, y.vec)
> ls()
>
> ## 2nd section: modeling
> x.vec = c(1:25)
> z.vec = x.vec^2
> y.vec = (2 + rnorm(25)) + (4+rnorm(25))*z.vec +
   (rnorm(25)*sd(z.vec))
> plot(x = x.vec, y = y.vec)
>
> ## 3rd section: graphics
> dd = data.frame(x = x.vec, y = y.vec, z = z.vec)
> fit.lm = lm(y ~ x, data = dd)
> summary(fit.lm)
> fit.lowess = lowess(x = dd$x.vec, y = dd$y)
> plot(x = dd$x, y = dd$y, pch = 16)
> lines(x = dd$x, y = fit.lowess$y, col = "blue")
> abline(fit.lm, col = "red")
```

1.7 函式 Function

R 有許多 函式 (function), 函式是一種物件, 是指令的集合, 執行特定功能或運算工作的指令, 資料整理, 資料分析等, 透過函式, 擴展了 R 在程式語言的功能性與便利性. 函式內通常需輸入 引數 (argument).

R 基本系統 (base) 提供了一部分常用函式, 而更多不同類型的函式, 則由許多不同的學者貢獻到 R 系統 (contribution) 中, 這些函數都是用 R 程式語言寫成的. 例如, 統計常用函式 `mean()`, `var()`, `sd()`, `log()` 等.

```
> ## function
> ## function c() = concatenate elements, return a vector x.vec
> x.vec = c(1:5) > x.vec # show x.vec
[1] 1 2 3 4 5
```

```
> mean(x = x.vec) # mean() calculate mean, return a scalar
[1] 3
> var(x = x.vec)  # mean() calculate variance
[1] 2.5
> log(x = x.vec)  # take log for all elements in vector x.vec
[1] 0.0000000 0.6931472 1.0986123 1.3862944 1.6094379
```

一個函式內通常需輸入 引數 (argument) 或是 統計公式, 統計模型, (formals). 引數可以是一個以上, 有些引數一定要輸入, 稱爲 必要引數 (required argument), 有些引數可以不用輸入, 稱爲 自選引數 (optional argument), 另外一種引數則爲 省略引數 (ellipsis argument) 這 3 種引可以同時存在一個函式內, 引數可以是數值, 文字, 資料框架 或 R 的任何物件.

R 函式以下面的函式語句形式使用必備引數:

```
> function.name(arg.1, arg.2, ...)
```

必備引數是使用者一定必須輸入引數值, 該函式運算的最終結果 (值), 就是函式回傳給的物件.

另一種 R 函式以下面的函式語句形式使用選擇引數:

```
> functionname(arg.1 = value.1, arg.2 = value.2, arg.3 = value.3, ...)
```

其中, `arg.1`, `arg.2`, ... 爲函式的引數或參數, 而 `value.1`, `value.2`, ... 爲引數的內設值或使用者自行輸入值. 例如, 函式 `log()` 指令:

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

`log` 函式在 R 內設以自然數爲底計算, 其中 `x` 爲必備引數, 使用者必須自行輸入所要計算的數值作爲引數值. 而 `base = exp(1)` 爲選擇引數, 若使用者沒有輸入引數值, 函式 `log()` 內設以自然數 e 爲底, 當然, 使用者可以更動底數的設定值, 例如更動爲以 2 爲底的獨度對數運算, `log(x, base = 2)`.

```
> x.vec <- c(1, 2, 3, 4, 5)
> log(x = x.vec)
[1] 0.0000000 0.6931472 1.0986123 1.3862944 1.6094379
> log(x = x.vec, base = 2)
[1] 0.000000 1.000000 1.584963 2.000000 2.321928
```

1.8 套件 Packages

特定的統計分析方法許多專用的函式集合成一組. “套件” (package). 許多學者針對特定分析, 寫成專用的 R 函式, 學者常將這些特定的統計分析方法專用的函式集合成一組 “套件” (package), 例如, survival 套件, 專用來進行存活分析, R 目前有 > 1000+ 組 packages, 且套件數目一直在快速增加, 個別套件內則有不同的函式. 在 R 中, 由一些標準 (基本) 套件構成 base R, 包含 R 可以進行一些標準統計和繪圖所需的的基本函數, 在任何 R 的安裝版本中, 都會被自動安裝與載入. 另外, 許多學者爲 R 提供了基本套件以外的套件, 稱爲 “貢獻套件” (contributed package).

若在 R 第一次使用某一特定功能的套件, 則需事先安裝此特定套件. 安裝套件有不同的方法, 若已經先連接網際網路, 常用的方法爲 (1) 使用 RStudio 內設套件所在的位置. 或 (2) 自行設定使用套件所在的位置 (Repository Site), 使用以下指令.

```
> setRepositories(graphics = getOption("menu.graphics"),
      ind = NULL, addURLs = character())
```

初學者通常可選擇第 (1) 項. 由 RStudio 右下套件視窗. 使用滑鼠選擇 Packages → Install. 輸入所要安裝的套件名稱, 例如,ggplot2, MASS, survival, 等.

第 (2) 種方式, 使用函式 install.packages 安裝所需的套件,

```
> install.packages("PackageName", dependencies = TRUE)
```

可以安裝 PackageName 套件. 例如, 可將下列指令寫入 Console 視窗內.

```
> install.packages("knitr")
> install.packages("ggplot2")
> install.packages("MASS")
> install.packages("survival")
> library(knitr)    # or library("knitr")
> library(ggplot2)
> library(MASS)
> library(survival)
```

若在 R 中, 要使用某一特定的套件, 須先載入此特定的套件,

- 請先確認已經安裝套件或更新套件.

- 使用 library() 或 require() 函式.

例如使用函式 library(package.name) 或 library("package.name"), 載入 package.name 套件. 例如, 載入 survival 套件, 可以使用以下的指令,

```
> library(survival)   # use survival packages
> library("survival")
> library(datasets)   # use datasets packages
> require(ggplot2)
```

通常建議使用 library(), 因爲 library() = loads a package, 會同時檢視套件與關聯的函式是否安裝妥當, 而 require() = tries to load a package, 且當套件與關聯的函式未安裝妥當時, 並不會提出警告或出現 error 訊息, 例如, 函式 foo() 未安裝妥當, 使用 require() 不會提出警告, 但當使用者在後續使用到 函式 foo() 時, 才會出現 error 訊息, 或是不出現 error 訊息而出現產生的物件, my.obj, 不存在, 因此很難追尋到錯誤來源.

通常在使用 library(package.name) 之後, 就可直接使用 package.name 套件內的函式 function.name(). 個別套件內有許多函式, 不同函式可能有相同名稱的函式, 若要避免誤用相同名稱的函式, 可以使用 "::" 串聯 package.name 與 function.name() 如下:

```
> package.name::function.name()
```

就可正確使用特定套件 package.name 內的特定函式 function.name(). 例如使用指令

```
> ggplot2::ggplot()
```

明確指出使用套件 ggplot2 內的函式 ggplot().

1.9 解說與輔助文件

R 有良好的解說文件, 可利用 Google 搜尋任何 R 相關疑惑. R 內部最常使用的線上協助爲啟動網頁流覽器 help.start(),

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

尋找特定函式 funName 解說, 在 R 中可直接輸入下列任一種函式, help(funName), ?funName, help.search("funName"), apropos("funName") 等等. 例如, 尋找函式mean() 函式解說. 在 R 中可直接輸入下列任一種指令.

```
> help(mean)
> ?mean
> help.search("mean")
> apropos("mean")
```

若要詢問函式內的引數, 可用函式 args("funName").

若對於有特殊含義的字元, 可以加上雙引號或者單引號, 即 “字串”, 查詢特殊符號也要用雙引號 (") 括起來. 例如,

```
> help("if")
> ?" =  = "
```

協助函式 help.search() 可以讓使用者尋找某一特定主題, 允許使用者使用任何方式搜尋輔助文件, 例如,

```
> help.search("linear models")
```

範例函式 example(): 可以執行某一特定函式輔助文件中的例子. 例如, example() 可以執行某一特定函式輔助文件中的例子, 例如, 輸入 example(plot) 則出現在繪圖函式 plot() 之輔助文件中內建的例子與結果.

```
> example(plot)
```

函式 data() 可以顯示 R 目前所有的資料組的物件名稱,

```
> data()
```

函式 data(data.name) 則載入某特定資料 data.name, 例如,

```
> data(Titanic) # load Titanic data frame
Titanic # show Titanic data frame
```

若要取得相關套件的輔助文件, 可以使用 library(help = package.name). 例如, 查看 MASS 套件的相關輔助文件, 可用下的指令

```
> library(MASS)              # load package MASS
> library(help = MASS)         # describe MASS
> help(lda, package = "MASS") # lda is a function in MASS
> help(lda)                  # help document for lda
```

函式 `data(package = .packages(all.available = TRUE))` 可以查看現存所有套件中之資料框架 (data frame), 若要查看或使用套件中的內建某一資料框架, 可用以下的指令

```
> library(stats)    # load package stats
> library(datasets) # load package datasets
> data()            # check all abailable data sets
> data(Puromycin)   # load Puromycin data frame
> help(Puromycin)   # help document for Puromycin data frame
> ## alternative way
> data(package = "datasets") # load package datasets
> data(Puromycin, package = "datasets") # load Puromycin data frame
> Puromycin # show Puromycin data frame
   conc rate     state
1  0.02   76   treated
2  0.02   47   treated
3  0.06   97   treated
4  0.06  107   treated
5  0.11  123   treated
.....................
```

測驗 **4.** 請依序輸入下列指令, 依照 Console 視窗指示, 在 Console 視窗內按下 **RETURN** 按鍵.

```
> demo(image)
> example(contour)
> demo(graphics)
> demo(persp)
> demo(plotmath)
> demo(Hershey)
```

1.10 R 與 GUI 介面

使用 RStudio 介面, 是一種 R 的整合系統, 除此之外, 另有 Microsoft R Open (前身爲 Revolution Analytics), Microsoft R Server 的整合系統, 以及 Oracle R

Enterprise 的整合系統, 這些整合系統對可重製性文件, 版本控制, 大數據分析的專案分析非常有助益, 對進階使用 R 的資料分析人員, 可以任選一種整合系統作爲執行專案分析的工具. 若不是使用 RStudio 介面, 由 John Fox, McMaster university (英國) 所寫的 Rcmdr 套件, 提供初學者使用類似 SPSS 圖形界面 (GUI, Graphics User Interface), 方便使用者可以點選方式, 進行資料處裡與常用的統計分析, 參見 https://www.rcommander.com/. 首先須直接使用 R, 將 R 打開, 點選 Console 視窗上方的編輯, 點選 編輯 其中的 GUI 偏好使用, 點選最上方 Single or multiple "**SDI**", 然後儲存. 第一次使用 Rcmdr 套件, 須先連接網際網路, 安裝與更新套件.

```
1 > install.packages("Rcmdr", dependencies = TRUE)
```

隨後, Rcmdr 套件可用以下的指令載入, R 會自動安裝與載入其他必要套件.

```
1 > library("Rcmdr")
```

使用 Rcmdr GUI 表單點選, R 會自動產生相關指令, 使用者可以在指令視窗中做修改. 對於 R 的初學者, 或是基礎統計學上課, Rcmdr 套件是可以考慮的選用套件. 關於使用 Rcmdr 套件的詳細資訊, 可以進入網頁 http://socserv.mcmaster.ca/jfox/Misc/Rcmdr/ 查看.