第 5 章: 資料輸入與輸出

5: Data Import and Export

在 R 中, 資料以具有名稱的 "物件" 形式儲存, 它們可以是向量 (vector), 矩陣 (matrix), 陣列 (array), 列表 (Lists), 或 資料框架 (data frames) 等. 簡單的資料, 可以直接在 R 視窗中輸入, 大型資料, 通常先以資料庫軟體, 試算表軟體等輸入儲存成外部檔案, 由 R 從外部檔案中讀入, 而不是直接在 R 中使用鍵盤輸入.

R 可以讀入純文字 ASCII 格式的資料檔, R 也有許多套件可以讀入許多不同格式檔案與資料庫關聯檔案, SAS, SPSS, STATA, EXCEL, web (XML, HTML), image, texts, stock market, social media 等. 每一資料的儲存格式不盡相同, 通常由儲存資料的軟體所決定, 通常可由檔案的附檔名得知資料的儲存格式.

R 對輸入資料的原先設計, 假設使用者利用其他工具, 如文書檔編輯器, 資料庫軟體, 或試算表軟體等, 輸入原始資料並儲存成 R 的外部原始資料檔案, 任何外部原始資料檔案, 必須修改成特定的 R 輸入檔格式, 以使它們符合 R 輸入外部原始資料檔案的要求, 對統計分析來說, 這是比較容易地進行外部原始資料的管理. 如同多數專業統計軟體, 不是專業的資料庫軟體, 外部原始資料的管理, 通常都是藉由專業的資料庫軟體進行. 任何資料庫軟體都可以將資料輸出純文字 ASCII 格式的資料檔案, 且任何資料庫軟體或文書檔編輯器都可讀入純文字 ASCII 格式的資料檔案.

5.1 資料框架 Data Frame

統計計算可以進行分析的資料, 通常有一個的簡單的基本架構, 在 R 稱作 資料框架 (data frame). 資料框架是類似於在 SAS, STATA 等的 dataset 架構. 資料框架通常類似矩陣, 資料框架也類似矩形的 交叉列聯表 (cross table).

在 R 中, 資料框架是統計分析中最基本的資料結構, 許多統計模型分析都必須用到資料框架結構, 資料框架與矩陣類似, 不同的地方在資料框架變數不需要是相同的變數形式或種類, 實數變數, 整數變數, 文字類別變數, 邏輯變數等都可放在同一資料框架中, 如表 5.1. 但是矩陣物件只能有相同的變數形式. 資料框架的基本架構如表 5.1, 資料框架有一些特徵, 例如:

1. 每一欄 (column), 都是一個變數 (variable).
2. 第一列 (row), 可以是變數的 "變數名" (variable names) 或是 "欄位標籤" (column label).
3. 每一欄 (變數) 的變數值形式可以是實數, 文字, 邏輯變數.
4. 第一欄 (column) 有時候是 "列位標籤" (row label).
5. 若原始資料沒有 變數名 或 標籤, R 讀入資料時可以同時輸入變數名, 或讀入資料後, 再輸入變數名.

表 5.1: DMTKRcsv.txt: DM-TKR Study Data 以純文字空白分隔儲存原始資料檔案

No	age	sex	DM	DMyr	preAC	prePC	postAC	postPC	Med	SIDE	PREKS	POSKS	ABS	INFECT
1	67	0	0	10	120	160	140	180	0	0	56	92	1	0
2	67	0	0	11	100	150	150	220	0	1	62	62	0	1
3	72	1	0	4	150	200	120	150	2	0	60	94	1	0
4	82	1	0	8	150	200	160	250	0	1	47	90	1	0
5	73	1	0	3	85	110	140	200	0	0	44	88	0	0
...														

5.2 輸入外部 ASCII 資料檔案至 R 資料框架

任何資料庫軟體都可以將資料輸出純文字 ASCII 格式的資料檔案, 且任何資料庫軟體或文書檔編輯器都可讀入純文字 ASCII 格式的資料檔案. 統計分析通常需要讀取外部 原始資料 (raw data) 檔案, R 對輸入資料的原先設計時讀入純文字 ASCII 格式的資料檔案, 暫存到電腦的記憶體上, 形成 R 資料框架, 對初學者而言讀入或輸出外部純文字 ASCII 格式的資料檔案最爲容易.

R 統計分析主要在資料框架 (data frame) 中的變數進行分析操作, R 使用資料框架函式指令 `read.table()` 或 `read.csv()` 讀入外部資料檔案, 對初學者而言最爲容易. 而其他函式, 例如, `scan()` 讀取外部資料, 對初學者而言較困難. 爲了可以直接讀取整個外部檔案進入 R 並形成 R 的資料框架物件, 純文字 ASCII 格式的外部檔案常常要求有特定的格式, 例如:

1. 多數檔編輯器, 資料庫, 試算表等軟體可以存取.
2. 第一的列 (the first row) 可以有該資料各個變數的 “變數名” (variable names) 或是 “行位名”, “欄位名” (column name) 或是 “欄位標籤” (column label).
3. 第一欄 (the first column) 有時候是 “列的標籤” (row label) 或是 “列的名字” (row name).
4. 其餘的列 (row), 是個別變數的觀測數值.
5. 變數值之間常以 空白鍵 (<space>) ” (blank space)” 分開, 也可以其他特定符號分隔.
6. 變數值之間以 空白鍵 或 特定符號 TAB 鍵分開意義不同, 不同軟體讀入這 2 種 ASCII 資料有所不同.
7. 若是變數資料是文字列型, 通常以雙引號包含, 但有時例外, 需視資料原始儲存軟體的設定.

8. 純文字 ASCII 格式, 變數值以空格分開的副檔名通常爲 .dat, .prn 或 .txt.
9. 若變數之間以 “,” (逗號) 分開變數值的 ASCII 形式檔案, 一般稱爲 comma-separated-variable format 或 CSV format, 檔案名通常以 .csv 作爲延伸檔名.
10. 初學者的原始資料之 變數名稱 (variable name) 不要使用 “中文”, “空格”, 盡量少用 “.” (點, dot), 或 “_” (underscore). 中文, 以免容易造成錯誤. 初學者盡量不要使用 “文字” 作爲變數值 (observed value).

初學者可練習建立純文字 ASCII 格式資料檔案. 首先建立檔案夾 “C:\RData”, 開啟 Microsoft Excel 程式, 在 Exccel 下, 輸入下列資料, 儲存檔名爲 Rlab00.xlsx.

group	base	final	gender
0	15	14	M
1	11	6	M
0	2	7	M
1	5	1	F
0	6	3	F
1	5	8	F

輸入完成後, 首先另存新檔爲 Rlab00.xlsx, 且工作表單 (sheet) 名稱改成 Rlab00. 然後再次另存新檔, 選擇 文字檔 Tab 字元分隔 (*.txt) 文字檔, 成爲 “C:\RData” 檔案夾內的 Rlab00.txt, (注意副檔名爲 .txt). 使用 記事本 (Notepad) 或 WordPad 打開 Rlab00.txt 檔案檢視內容. 請將下列指令寫入 Rlab00.r 視窗內, 執行下列指令, 可以讀入 Rlab00.txt 資料檔案.

```
> ## read.table()
> ## read data file: Rlab00.txt
> setwd("C:/RData")
> Rlab00.df = read.table("Rlab00.txt",
                         header = TRUE,
                         row.names = NULL, dec = ".")
> Rlab00.df
> head(Rlab00.df)
> str(Rlab00.df)
```

檢視 RStudio Console 視窗:

```
> setwd("C:/RData")
> Rlab00.df = read.table("Rlab00.txt",
                         header = TRUE,
                         row.names = NULL, dec = ".")
> Rlab00.df
  group base final gender
1     0   15    14      M
2     1   11     6      M
3     0    2     7      M
4     1    5     1      F
5     0    6     3      F
6     1    5     8      F
> head(Rlab00.df)
  group base final gender
1     0   15    14      M
2     1   11     6      M
3     0    2     7      M
4     1    5     1      F
5     0    6     3      F
6     1    5     8      F
> str(Rlab00.df)
'data.frame':   6 obs. of  4 variables:
 $ group : int  0 1 0 1 0 1
 $ base  : int  15 11 2 5 6 5
 $ final : int  14 6 7 1 3 8
 $ gender: Factor w/ 2 levels "F","M": 2 2 2 1 1 1
```

另一種原始資料以純文字 ASCII 格式檔儲存, 且變數值之間是以逗號 “,” (comma) 分隔的 ASCII 檔案, 其格式化稱爲 csv format (comma-separated-variable format), 副檔名通常爲 “.csv”. 初學者可練習建立純文字 ASCII .csv 格式資料檔案, 首先建立或確認檔案夾 “C:\RData”, 開啟 Microsift Excel 在 Exccel 下, 輸入下列資料, 儲存檔名爲 Rlab00TestScore.xlsx.

ID	pretest	postest	Male
1	80	89	0
2	72	92	1
3	56	75	0
4	84	96	0

輸入完成後, 首先另存新檔爲 Rlab00TestScore.xlsx. 然後再次另存新檔, 選擇以 CSV 逗號分隔 (*.csv) 的格式, 成爲 “C:\RData” 檔案夾內的 Rlab00TestScore.csv,

(副檔名爲 .csv). 使用 記事本 (Notepad) 或 WordPad 打開 Rlab00TestScore.csv 檢視內容. 請將下列指令寫入 Rlab00.r 視窗內, 執行下列指令, 可以讀入 Rlab00TestScore.csv 資料檔案.

```
> ## read.csv()
> ## read data file: Rlab00TestScore.csv
> setwd("C:/RData")
> Rlab00Test.df = read.csv("Rlab00TestScore.csv",
                           header = TRUE,
                           row.names = NULL, dec = ".")
> Rlab00Test.df
  ID pretest postest male
1  1      80      89    1
2  2      72      92    1
3  3      56      75    0
4  4      84      96    0
> Rlab00Test.df$pretest
[1] 80 72 56 84
> mean(Rlab00Test.df$pretest)
[1] 73
```

資料框架函式 `read.table()`, `read.csv()` 或 `read.delim()` 讀入/輸入 純文字 ASCII 格式檔儲存的外部資料檔案 .

```
read.table(file, header = FALSE, sep = "", quote = "\"'",
           dec = ".", numerals = c("allow.loss", "warn.loss", "no.loss"),
           row.names, col.names, as.is = !stringsAsFactors,
           na.strings = "NA", colClasses = NA, nrows = -1,
           skip = 0, check.names = TRUE, fill = !blank.lines.skip,
           strip.white = FALSE, blank.lines.skip = TRUE,
           comment.char = "#",
           allowEscapes = FALSE, flush = FALSE,
           stringsAsFactors = default.stringsAsFactors(),
           fileEncoding = "", encoding = "unknown", text, skipNul = FALSE)
read.csv(file, header = TRUE, sep = ",", quote = "\"",
         dec = ".", fill = TRUE, comment.char = "", ...)
read.delim(file, header = TRUE, sep = "\t", quote = "\"",
           dec = ".", fill = TRUE, comment.char = "", ...)
```

函式中的常用引數:

1. `file = "filename"` 爲輸入外部檔案名, 若檔案不在工作路徑下, 需輸入完整路徑與檔案名. 例如, `file = "C:/RData/Rlab00.txt"`.
2. `header = T` 表示第一列 (raw), 是變數名稱.

3. `sep = ""` 表示以空白分開變數.
4. `sep = ","` 表示以 “,” 分開變數.
5. `dec = "."` 表示實數的小數點符號.
6. `row.names, col.names` 使用者可自行輸入列位或欄位名.
7. `as.is = !stringsAsFactors` 表示讀入變數值爲字串時, R 自動設成類別變數. 若設定 `as.is = TRUE` R 讀入成純文字變數值.
8. `na.strings = "NA"` 表示以 “NA” 表示缺失值, 若是使用其他符號或文字, 可更改, 例如, `na.strings = c("NA", "", ".")`.
9. `colClasses =` 手動設定每一個變數的屬性, 文字或數值, 可以加速讀入資料.
10. `skip = n` 表示跳過 n rows 才開始讀入資料.
11. `fill = TRUE` 與 `blank.lines.skip = FALSE` 可以處理資料檔案包含完全空白行或不完全空白行.
12. `read.csv()` R 自動讀以 csv 分隔的資料檔案.
13. `read.delim()` R 自動讀變數以 TAB 分隔的資料檔案.

假設在 R 的工作目錄中, 有一個 ASCII 形式的檔案, 如表 5.2.

表 5.2: DMTKRcsv.csv: DM-TKR Study Data 以 csv 格式純文字儲存原始資料檔案 (ASCII, CSV Format File)

No,age,sex,DM,DMyr,preAC,prePC,postAC,postPC,Med,SIDE,PREKS,POSKS,ABS,INFECT
1,67,0,0,10,120,160,140,180,0,0,56,92,1,0
2,67,0,0,11,100,150,150,220,0,1,62,62,0,1
3,72,1,0,4,150,200,120,150,2,0,60,94,1,0
4,82,1,0,8,150,200,160,250,0,1,47,90,1,0
5,73,1,0,3,85,110,140,200,0,0,44,88,0,0
6,76,0,0,1,120,150,120,200,0,1,52,94,1,0
7,76,0,0,1,120,150,120,200,0,0,48,96,0,0
8,77,0,1,35,200,250,230,300,1,1,42,90,1,0
9,64,0,0,5,130,180,100,150,0,0,40,94,1,0
10,64,0,0,5,130,180,100,150,0,1,45,96,0,0
... ...

使用函式 `read.table()` 自動內設以空格分開變數值, 必須更改分開變數值符號爲 ",". 使用函式 `read.csv()` 自動內設以 "," 分開變數值, 讀入資料檔案 DMTKRcsv.csv.

```
> ## Data Input
> ## read data file: DMTKRcsv.csv
> DMTKR.read.table.csv <- read.table("DMTKRcsv.csv",
                                     header = TRUE,
                                     row.names = NULL,
                                    sep = ",", dec = ".")
> head(DMTKR.read.table.csv, n = 3)
  No age sex DM DMyr preAC prePC postAC postPC Med SIDE PREKS POSKS ABS INFECT
1  1  67   0  0   10   120   160    140    180   0    0    56    92   1       0
2  2  67   0  0   11   100   150    150    220   0    1    62    62   0       1
3  3  72   1  0    4   150   200    120    150   2    0    60    94   1       0
> #
> DMTKR.read.csv <- read.csv("DMTKRcsv.csv",
                             header = TRUE,
                             row.names = NULL,
                             dec = ".")
> tail(DMTKR.read.csv, n = 3)
   No age sex DM DMyr preAC prePC postAC postPC Med SIDE PREKS POSKS ABS INFECT
76 76  65   1  1   18   110   140    130    200   0    0    54    86   0       0
77 77  71   1  0   13   180   260    200    300   0    0    50    94   1       0
78 78  59   1  0    4   120   170    130    170   2    1    49    94   0       0
```

測驗 **1.** 設定工作目錄爲 `setwd("C://RData")`, 在 R 工作目錄下的一個原始資料檔案爲 IQBirthOrderAge.dat, 其內容爲

```
iq  birth age class
110 1     25  "A"
115 1     24  "A"
120 1     22  "A"
118 1     24  "A"
110 2     20  "A"
108 2     20  "A"
105 2     20  "A"
104 3     24  "B"
98  3     25  "B"
99  4     30  "B"
98  4     24  "B"
100 5     29  "B"
90  5     30  "B"
93  5     30  "B"
90  6     28  "B"
```

使用函式 `read.table()` 進行 讀入/輸入 外部資料檔案 IQBirthOrderAge.dat, 讀入原始資料形成一個資料框架, 比較 `read.table()` 引數 `as.is = TRUE` 的差異.

```
> setwd("C:/RData") # download data into this folder
> iqagedata.df <- read.table("IQBirthOrderAge.dat",
                             header = TRUE,
                             row.names = NULL, dec = ".")
> str(iqagedata.df)
#
> iqagedata.df <- read.table("IQBirthOrderAge.dat",
                             header = TRUE, as.is = TRUE,
                             row.names = NULL, dec = ".")
str(iqagedata.df)
```

測驗 **2.** 設定工作目錄爲 `setwd("C:/RData")`, 在工作目錄下的一個原始資料檔案爲 AgeFat.csv, 其部分內容爲

```
age,fat
24,9.5
24,27.9
28,7.8
28,17.8
40,31.4
```

使用函式 `read.csv()` 進行 讀入或輸入 外部資料檔案 AgeFat.csv, 讀入原始資料形成一個資料框架.

```
> setwd("C:/RData") # set up a working directory
> agefat.df <- read.csv("AgeFat.csv", header = TRUE,
                        row.names = NULL, sep = ",", dec = ".")
> agefat.df
```

5.3 R 內建資料框架

R 有許多內建資料框架, 另外貢獻套件 (contributed packages) 也有許多內建的資料框架, 可以使用 `data()` 查看 R 內建資料框架名稱, 或使用 `library(help = "datasets")` 查看 R 內建資料框架名稱.

使用 `data(package = "package.name")` 查看名稱爲 `package.name` 套件中的資料框架名稱, 載入資料框架, 可用 `data(data.name)` 載入 R 內建名稱爲 `data.name` 資料框架使用, 或 `data(package.data.name, package = "package.name")` 載入名稱爲 `package.name` 套件中, 名稱爲 `pack.data.name` 資料框架使用.

```
> data()       # check the names of data frame
> data(Orange) # load R builded data frame called Orange
> help(Orange) # check the help file for Orange
> Orange
  Tree  age circumference
1    1  118            30
2    1  484            58
3    1  664            87
.........................
> #
> library(MASS)
> help(package = MASS)
> data(package = "MASS")     # check the names of data frames inside MASS package
> data(VA, package = "MASS") # load data frame called VA in MASS package
> help(VA)                   # check the help file for VA
> VA
   stime status treat age Karn diag.time cell prior
1     72      1     1  69   60         7    1     0
2    411      1     1  64   70         5    1    10
3    228      1     1  38   60         3    1     0
4    126      1     1  63   60         9    1    10
................................................
```

5.4 函式指令 attach() 與 detach(): 使用資料框架內的變數

當輸入外部資料檔案入資料框架, 或載入內建資料框架, 若要直接使用其中變數, 可以使用函式 attach(). 例如, 使用在 R 中資料框架 DMTKRtable 中的 age.

```
> DMTKRtable$age
 [1] 67 67 72 82 73 76 76 77 64 64 78 69 73 73 81 81 74 62 75 90 71 71 83
[24] 77 80 80 67 67 72 81 81 74 73 74 63 60 75 67 69 69 70 72 76 74 79 65
[47] 72 69 71 65 67 70 76 74 80 61 56 61 74 66 67 75 54 71 65 70 71 69 74
[70] 76 61 54 61 72 64 65 71 59
```

當使用若要固定使用同一資料框架持續進行統計分析時,

1. 可以使用 attach(data.frame.name) 載入名稱爲 data.frame.name 資料框架. 這樣就可以直接使用用變數 variable.name 來進行分析.
2. 不使用 attach(data.frame.name), 使用用變數 data.frame.name$variable.name 來進行分析.
3. 檢視變數名稱, 可以用 names(data.frame.name) 取得名爲 data.frame.name 資料框架內的所有變數名.

```
> attach(DMTKRtable)
> age
 [1] 67 67 72 82 73 76 76 77 64 64 78 69 73 73 81 81 74 62 75 90 71 71 83
[24] 77 80 80 67 67 72 81 81 74 73 74 63 60 75 67 69 69 70 72 76 74 79 65
[47] 72 69 71 65 67 70 76 74 80 61 56 61 74 66 67 75 54 71 65 70 71 69 74
[70] 76 61 54 61 72 64 65 71 59
> mean(age)  # the mean value of age
[1] 70.83333
> names(DMTKRtable) # get the variable names
 [1] "No"     "age"    "sex"    "DM"     "DMyr"   "preAC"  "prePC"
 [8] "postAC" "postPC" "Med"    "SIDE"   "PREKS"  "POSKS"  "ABS"
[15] "INFECT"
```

當不再使用某一特定資料框架時, 可以用 detach(data.frame.name), 移出 R 工作空間常駐位置.

```
> detach(DMTKRtable)
> age
Error: can not find the object "age"
> DMTKRtable$age
 [1] 67 67 72 82 73 76 76 77 64 64 78 69 73 73 81 81 74 62 75 90 71 71 83
[24] 77 80 80 67 67 72 81 81 74 73 74 63 60 75 67 69 69 70 72 76 74 79 65
[47] 72 69 71 65 67 70 76 74 80 61 56 61 74 66 67 75 54 71 65 70 71 69 74
[70] 76 61 54 61 72 64 65 71 59
> mean(DMTKRtable$age)
[1] 70.83333
```

對初學者而言使用 attach() 與 attach() 分析資料各有好有壞, 但個人並不建議使用. 因此建議初學者不要使用函式 attach(), 因爲許多資料都有 age, gender 等相同變數名稱, R 執行時容易錯誤使用或解讀相同變數名稱的物件.

測驗 **3.** 當輸入外部資料檔案入資料框架, 或載入內建資料框架, 若要直接使用其中變數, 可以使用函式 attach(). 例如, 在 R 中輸入外部資料檔案 Rlab00Test.csv, 成爲 Rlab00Test.df 資料框架物件, 要直接使用資料框架 Rlab00Test.df 的變數 postest, 計算平均值. 使用 函式 atach()可以直接使用變數 postest 若不再直接使用資料框架 Rlab00Test.df 內的變數時, 使用 函式 detach() 移出 R 工作環境, 然後接使 資料框架 Rlab00Test.df 的變數 postest, 計算平均值. 請執行下列指令, 比較 attach() 與 detach() 差異.

```
> ## attach() and detach()
> setwd("C:/RData")
> Rlab00Test.df = read.csv("Rlab00TestScore.csv",
                           header = TRUE, row.names = NULL, dec = ".")
> attach(Rlab00Test.df)
> postest
> mean(postest)
> detach(Rlab00Test.df)
> mean(Rlab00Test.df$postest)
> mean(postest)
> postest
```

5.5 函式指令 scan() 輸入資料

函式 scan() 可以讀入純文字 ASCII 之外部檔案, 函式 scan() 回傳 list, 須再

轉換成 data frame. scan() 內設讀入變數值全是實數值, 若有文字變數值, 須使用引數 what. 使用函式指令 scan() 輸入資料, 須自行設定許多細項, 包含變數名, 列位或欄位數目等, 且常須使用引數 skip 與 nlines. 通常使用函式 scan() 輸入大型資料或是非常簡易資料.

```
scan(file = "", what = double(), nmax = -1, n = -1, sep = "",
     quote = if(identical(sep, "\n")) "" else "'\"", dec = ".",
     skip = 0, nlines = 0, na.strings = "NA",
     flush = FALSE, fill = FALSE, strip.white = FALSE,
     quiet = FALSE, blank.lines.skip = TRUE, multi.line = TRUE,
     comment.char = "", allowEscapes = FALSE,

     fileEncoding = "", encoding = "unknown", text, skipNul = FALSE)
```

函式 scan() 多數的引數與 read.table() 相同.

```
## {\McQ\cH21}\z{\MbQ\cH113}\z{\McQ\cH146}\z{\MbQ\cH101}
> x.vec = scan()
1: 1 3 5
4: 2 4 6
7: 11 13 15
10:
Read 9 items
> x.vec
[1]  1  3  5  2  4  6 11 13 15
```

```
scan(file = "C:/RData/Rlab00.txt",
     what = list(group = 1, base = 1, final = 1, gender = ""),
     skip = 1)
> x.list = scan(file = "C:/RData/Rlab00.txt",
           what = list(group = 1, base = 1, final = 1, gender = ""),
           skip = 1)
Read 6 records
> x.list
$group
[1] 0 1 0 1 0 1
$base
[1] 15 11  2  5  6  5
$final
[1] 14  6  7  1  3  8
$gender
[1] "M" "M" "M" "F" "F" "F"
> as.data.frame(x.list)
  group base final gender
1     0   15    14      M
2     1   11     6      M
3     0    2     7      M
4     1    5     1      F
```

```
23 5     0    6     3      F
24 6     1    5     8      F
```

5.6 輸入 EXCEL 資料檔案

R 有許多套件可以直接輸入 EXCEL 資料檔案. 但若 EXCEL 內包含 EXCEL 的函數公式計算, 圖表 等, 常會造成錯誤, 因此清理 EXCEL 資料是首要工作. 變數名最好不要有空格 (no blank) 或 特定符號, 工作表 (sheet) 命名最好不要使用中文, 在個別資料最後, 不要有多餘空白 row 與 column, 在 EXCEL 定義變數的屬性須小心, 例如, 負數, 百分位數, 日期, 文字或字串等等, 缺失值是否輸入特定符號或留空白, EXCEL 是否執行自動補 0 的行爲, 都會影響讀入資料的正確性. 清理資料完成後, 另存新檔, 然後再讀入, 但最最穩定的方法還是轉成 .txt 或 .csv.

使用 R 套件 “readxl" 中的 函式 read_excel() 輸入 EXCEL 資料檔案. 此套件 readxl 並不須依賴外部軟體如, perl, JAVA 等.

```
read_excel(path, sheet = 1,
           col_names = TRUE, col_types = NULL,
           na = "", skip = 0)
```

函式 read_excel() 的引數 與 函式 read.table() 類似.

```
> ## install.packages("readxl") # install package
> library(readxl)
> setwd("C:/RData")
> dd <- read_excel("Rlab00.xlsx", sheet = 1,
                   col_names = TRUE, col_types = NULL,
                   na = "", skip = 0)
> dim(dd)
[1] 6 4
> str(dd)
Classes  bl_df bl and 'data.frame': 6 obs. of  4 variables:
 $ group : num  0 1 0 1 0 1
 $ base  : num  15 11 2 5 6 5
 $ final : num  14 6 7 1 3 8

 $ gender: chr  "M" "M" "M" "F" ...
```

使用 R 套件 “XLConnect” 中的 函式 loadWorkbook() 與 函式 readWorksheet()

輸入 EXCEL 資料檔案. 使用 XLConnect 套件需安裝 Java, JAVA 與 R 需相同位元, 例如:

- R 23-bit 需使用 JAVA 32-bit.
- R 64-bit 需使用 JAVA 64-bit, JAVA 有時需手動下載安裝.

```
> ## install.packages("XLConnect") # {\MaQ\cH108}\z{\MaQ\cH233}\z{\McQ\cH95}\z{\MeQ\cH80}\z
    {\MaQ\cH75}
> library(XLConnect) # install package "XLConnect"
> setwd("C:/RData")
> wb = loadWorkbook("Rlab00.xlsx")
> dd = readWorksheet(wb, sheet = "Rlab00", header = TRUE)
> dim(dd)
[1] 6 4
> str(dd)
'data.frame':   6 obs. of  4 variables:
 $ group : num  0 1 0 1 0 1
 $ base  : num  15 11 2 5 6 5
 $ final : num  14 6 7 1 3 8
 $ gender: chr  "M" "M" "M" "F" ...
```

使用 R 套件 "xlxs" 內的函式 read.xlsx() 或 read.xlsx2() 輸入 EXCEL 資料檔案, 但函式內設將變數視爲類別變數, 因此若變數值爲實數, 則須另外指定變數屬性.

```
## install.packages("xlsx") # install package "xlsx"
> setwd("C:/RData")
> dd <- read.xlsx2("Rlab00.xlsx", sheetName = "Rlab00",
+                  header = TRUE)
> dim(dd)
[1] 6 4
> str(dd)
'data.frame':   6 obs. of  4 variables:
 $ group : Factor w/ 2 levels "0","1": 1 2 1 2 1 2
 $ base  : Factor w/ 5 levels "11","15","2",..: 2 1 3 4 5 4
 $ final : Factor w/ 6 levels "1","14","3","6",..: 2 4 5 1 3 6
 $ gender: Factor w/ 2 levels "F","M": 2 2 2 1 1 1
```

函式內設將變數視爲類別變數, 因此 group, base, final. 若變數值爲實數, 則須另外指定變數屬性.

```
library(xlsx) # install package "xlsx"
> dd <- read.xlsx2("Rlab00.xlsx", sheetName = "Rlab00",
                   header = TRUE,
```

```
                colClasses = c("numeric", "numeric",
                               "numeric", "character"))
> dim(dd)
[1] 6 4
> str(dd)
'data.frame':   6 obs. of  4 variables:
 $ group : num  0 1 0 1 0 1
 $ base  : num  15 11 2 5 6 5
 $ final : num  14 6 7 1 3 8
 $ gender: Factor w/ 2 levels "F","M": 2 2 2 1 1 1
```

使用 R 套件 "gdata" 內函式 `read.xlsx()` 輸入 EXCEL 資料檔案. 使用 R 套件 "gdata" 前, 必須先安裝 Perl 軟體, 安裝後同時找出 `perl.exe` 執行檔路徑位置, 例如, "C:/Perl/bin/perl.exe". `read.xlsx()` 類似 `read.table()`, `read.csv()`, 會先判斷變數屬性爲 類別 或 連續, 使用者也可自行指定變數屬性爲 類別 或 連續.

```
> library(gdata)
> setwd("C:/RData")
> dd <- read.xls("Rlab00.xlsx", sheet = 1,
                 perl = "C:/Per64/bin/perl.exe")
```

5.7 輸入 SAS, SPSS, STATA 資料檔案

R 幾乎可以輸入任何格式的檔案, 常見的統計軟體 SAS, SPSS, STATA 的資料都可輸入到 R 進行分析. 不同檔案的輸入常受到 R 套件的版本, 原生統計軟體的版本, 原生統計軟體的對變數屬定的設定, 缺失值, 日期格式與計算等因素, 常常會有錯誤無法讀入的情形. 每次最好使用 google 查詢最新的 R 套件, 若遇到錯誤訊息, 同樣利用 google 查詢錯誤訊息的解決方法.

5.8 輸出 R 資料至外部檔案

統計分析, 常常需儲存資料或分析結果, 輸出至外部檔案, 供其他軟體使用, 可以使用資料框架函式 `write.table()`, 與相對應函式 `read.table()`, 或 `write.csv()`,

與相對應函式 read.csv(). 輸出資料, 輸出資料框架或矩陣至外部資料檔案, 對初學者最爲容易. 對大型資料的輸出, 若變數值都是單一類型, 可以使用 MASS 套件中的 write.matrix(), 則會較有效率.

```
write.table(x, file = "", append = FALSE, quote = TRUE, sep = " ",
            eol = "\n", na = "NA", dec = ".", row.names = TRUE,
            col.names = TRUE, qmethod = c("escape", "double"),
            fileEncoding = "")

write.csv(...)
```

函式中的常用引數 與 read.table() 類似.

1. x 爲 R 資料物件名.
2. file = "filename" 爲輸出儲存的外部檔案名.
3. append = FALSE 表示複蓋檔案內原有資料, 設定 append = TRUE 表示附加在檔案內原有資料之後.
4. dec = "." 表示實數的小數點符號.
5. quote = TRUE 表示輸出時, 文字變數值附加上雙引號.
6. sep = "" 表示輸出以空白分開變數.
7. sep = "," 表示輸出以 “,” 分開變數.
8. na = "NA" 表示以 “NA” 表示缺失值, 若是使用其他符號或文字, 可更改, 例如, na = c(".").
9. col.names = TRUE 表示輸出第一列 (raw), 變數名稱.
10. row.names = TRUE 輸出列位位名.
11. write.csv() R 自動輸出以 csv 分隔的資料檔案.

例如, 輸出資料框架 Orange, 儲存成爲 ASCII 形式的 CSV 格式之外部檔案 Orange.csv 與 OrangeMASS.csv.

```
> write.table(Orange, file = "Orange.csv",
    sep = ",", col.vars = NA)
> ?write.table
> #
> library(MASS)
> ?write.matrix
> write.matrix(Orange, file = "OrangeMASS.csv", sep = ",")
```

5.9 物件輸入與儲存

R 的指令可以用文件編輯軟體事先輸入儲存成 ASCII 純文字檔, 然後在 R 中呼叫程式檔案進入 R 執行. 使用函式 `source()`, 可將程式檔案呼叫進入 R 執行. 例如, 下列指令將名字爲 "commands.R" 的 ASCII 純文字檔案, 呼叫入 R 中執行.

```
> source("commands.R")
> source("Rlab00.r")
```

函式 `sink()` 可以將以執行的指令, 程式碼與結果輸出至 ASCII 純文字檔的檔案儲存, 例如, 下列指令將程式碼與結果輸出至名字爲 "record.out" 的檔案儲存.

```
> sink("record.out")
```

在 R 工作中產生的所有物件都可以儲存, 在以後的 R 工作繼續使用. 在每一次 R 工作結束的時候, 使用者可以儲存所有當前所可用的物件, 這些物件會寫入當前工作目錄下, 一個名字叫 .RData 的文字檔案中, 並且所有在這次 R 工作中用過的指令或運算程式, 都也會儲存在目前的工作目錄之下, 一個名字叫 .Rhistory 的文字檔案中.

日後當 R 再次在同一工作目錄下啟動時, 在 .RData 檔案儲存物件會重新載入 R 中使用, 同時, 儲存相關的歷史指令檔案 .Rhistory 也會被載入使用. 因此結束 R 工作後, 當再次啟動 R 時, 會載入 .RData 與 .Rhistory 儲存的物件, 佔據電腦內的動態記憶體, 若是長期且重覆在相同的工作目錄使用 R 進行分析, 則儲存的 .RData 與 .Rhistory 內的物件會佔據電腦內大量的動態記憶體, 使得 R 執行運算變得非常慢, 除非執行產生的資料物件非常費時, 例如, 讀入與產生 R 生物資訊的資料物件非常費時, 否則不須儲存資料物件.

若要儲存重要的指令或函式, 可在 R 中使用的指令, 則另外以文字編輯軟體以程式檔案儲存. 另外可以使用函式 save() 儲存資料物件 mydata.df 如下.

```
> save(mydata.df, file = "FileName.RData")
```

若要載入存放在其它檔案夾內且以 FileName.RData 儲存的資料, 則輸入 load(), 例如,

```
> load("FileName.RData")
```

就可使用 FileName.RData 內的儲存資料物件 mydata.df. 其他類似函式 saveRDS() 儲存物件 objName, 使用 readRDS() 取出儲存的物件.

```
> save(mydata.df, file = "C:/RData/FileName.RData")
> load(file = "C:/RData/FileName.RData")
> save(mydata.df, file = "C:/RData/FileName.Rda")
> load(file = "C:/RData/FileName.Rda")
#
> saveRDS(objName, file = "objName.Rds")
> readRDS(file = "objName.Rds")
```

save() 與 saveRDS(), load() 與 readRDS() 本質上有些許差異.

```
> ## save and load
> x <- c(1:5)
> save(x, file = "x.Rda")   # where is your working directory?
> saveRDS(x, file = "x.Rds")
>
> ## assign using readRDS
> new_x1 <- readRDS("x.Rds")
> new_x1
[1] 1 2 3 4 5
> x
[1] 1 2 3 4 5
>
> ## assign using load -- note the result
> new_x2 <- load("x.Rda")
> new_x2
[1] "x"
> x
[1] 1 2 3 4 5
```