



Basic

目錄

Introduction	1.1
Chapter 1 - 環境建置	1.2
安裝與更新 R	1.2.1
安裝 RStudio	1.2.2
Rstudio 基本介紹	1.2.3
切換 R 的版本	1.2.4
Console 開啓 R console	1.2.5
安裝載入 package	1.2.6
Chapter 2 - 基本運算	1.3
資料屬性	1.3.1
常見運算	1.3.2
Chapter 3 - 變數與資料	1.4
變數	1.4.1
向量	1.4.2
陣列	1.4.3
矩陣	1.4.4
因子	1.4.5
列表	1.4.6
資料框架	1.4.7
Chapter 4 - 資料匯入與輸出	1.5
匯入資料	1.5.1
輸出資料	1.5.2
讀取資料庫的資料	1.5.3
Chapter 5 - 流程控制	1.6
邏輯判斷式	1.6.1
條件執行	1.6.2
迴圈結構	1.6.3

Chapter 6 - 資料整理	1.7
重新編碼	1.7.1
資料變形	1.7.2
資料合併與分割	1.7.3
Chapter 7 - 自訂函數	1.8
定義函數	1.8.1
建立 .First 與 .Last 函數	1.8.2

Introduction

本書是根據本人學習 R 的經驗，加上整理歸納後所產出的筆記，有任何錯問題歡迎大家糾正與指教。

本書涵蓋內容

1. 環境建置
2. 基本運算
3. 變數與資料
4. 資料匯入與輸出
5. 流程控制
6. 資料整理
7. 自訂函數
8. 繪圖功能
9. 文件撰寫

作者資訊

本書由 [Joe](#) 所撰寫。

原始碼

本書原始碼放置在 [Github](#)，歡迎大家直接 fork 修正內容有問題地方。

Chapter 1 - 環境建置

1. 安裝與更新 R
2. 安裝 RStudio
3. Rstudio 基本介紹
4. 切換 R 版本
5. Concole 開啓 R console

安裝與更新 R

Step1：開啓 R 官方網站



About R
What is R?
Contributors
Screenshots
What's new?

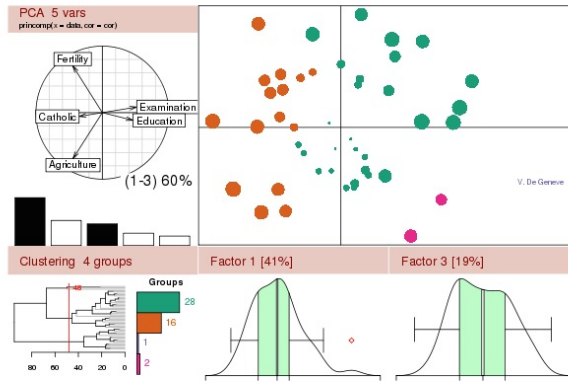
Download, Packages
CRAN

R Project
Foundation
Members & Donors
Mailing Lists
Bug Tracking
Developer Page
Conferences
Search

Documentation
Manuals
FAQs
The R Journal
Wiki
Books
Certification
Other

Misc
Bioconductor
Related Projects
User Groups
Links

The R Project for Statistical Computing



Getting Started:

- R is a free software environment for statistical computing and graphics. It compiles and runs on a wide variety of UNIX platforms, Windows and MacOS. To [download R](#), please choose your preferred [CRAN mirror](#).
- If you have questions about R like how to download and install the software, or what the license terms are, please read our [answers to frequently asked questions](#) before you send an email.

News:

- R version 3.1.1 (Suck it to Me) has been released on 2014-07-10.
- R version 3.0.3 (Warm Pudding) has been released on 2014-03-06.

Step2：選擇 Taiwan 任一個載點



About R
What is R?
Contributors
Screenshots
What's new?

Download, Packages
CRAN

R Project
Foundation
Members & Donors
Mailing Lists
Bug Tracking
Developer Page
Conferences
Search

Documentation
Manuals
FAQs
The R Journal
Wiki
Books
Certification
Other

Misc
Bioconductor
Related Projects
User Groups
Links

Spain	http://ftp.cixug.es/CRAN http://cran.es.r-project.org/	Oficina de software libre (CIXUG) Spanish National Research Network, Madrid
Sweden	http://ftp.sunet.se/pub/lang/CRAN/	Swedish University Computer Network, Uppsala
Switzerland	http://stat.ethz.ch/CRAN/	ETH Zuerich
Taiwan	http://cran.cs.pu.edu.tw/ http://cran.csie.ntu.edu.tw/	Providence University, Taichung National Taiwan University, Taipei
Thailand	http://mirrors.psu.ac.th/pub/cran/	Prince of Songkla University, Hatyai
Turkey	http://cran.pau.edu.tr	Pamukkale University, Denizli
UK	http://www.stats.bris.ac.uk/R/ http://mirrors.ebi.ac.uk/CRAN/ http://cran.ma.imperial.ac.uk/ http://mirror.mdx.ac.uk/R/ http://star-www.st-andrews.ac.uk/cran/	University of Bristol EMBL-EBI (European Bioinformatics Institute) Imperial College London Middlesex University London St Andrews University
USA	http://cran.cnr.Berkeley.edu http://cran.stat.ucla.edu/ http://streaming.stat.iastate.edu/CRAN/ http://ftp.usg.lsu.edu/CRAN/ http://rweb.quant.ku.edu/cran/ http://watson.nci.nih.gov/cran_mirror/ http://cran.mtu.edu/ http://cran.wustl.edu/ http://cran.case.edu/ http://ftp.osuosl.org/pub/cran/ http://lib.stat.cmu.edu/R/CRAN/ http://cran.mirrors.hoobly.com http://mirrors.nics.utk.edu/cran/	University of California, Berkeley, CA University of California, Los Angeles, CA Iowa State University, Ames, IA Indiana University University of Kansas, Lawrence, KS National Cancer Institute, Bethesda, MD Michigan Technological University, Houghton, MI Washington University, St. Louis, MO Case Western Reserve University, Cleveland, OH Oregon State University Statlib, Carnegie Mellon University, Pittsburgh, PA Hoobly Classifieds, Pittsburgh, PA National Institute for Computational Sciences, Oak Ridge, TN

Step3：選擇作業系統(作者本身是 Mac)



CRAN
[Mirrors](#)
[What's new?](#)
[Task Views](#)
[Search](#)

About R
[R Homepage](#)
[The R Journal](#)

Software
[R Sources](#)
[R Binaries](#)
[Packages](#)
[Other](#)

Documentation
[Manuals](#)
[FAQs](#)
[Contributed](#)

The Comprehensive R Archive Network

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:

- [Download R for Linux](#)
- [Download R for \(Mac\) OS X](#)
- [Download R for Windows](#)

R is part of many Linux distributions, you should check with your Linux package management system in addition to the link above.

Source Code for all Platforms

Windows and Mac users most likely want to download 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 (2014-07-10, Sock it to Me) [R-3.1.1.tar.gz](#), read [what's new](#) in the latest version.
- Sources of [R alpha and beta releases](#) (daily snapshots, created only in time periods before a planned release).
- 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](#)

Questions About R

- If you have questions about R like how to download and install the software, or what the license terms are, please read our [answers to frequently asked questions](#) before you send an email.

Step4：選擇下載的項目(有三種)

1. 下載 R 且包含 R GUI 界面的 software。
2. 下載 R。
3. 下載 R GUI 界面的原始程式碼。

=> 選擇 1 或 2 都可以，第一次安裝可以選擇 1，看看 GUI 畫面是什麼樣子，但之後會介紹 RStudio 比原始的 GUI 好用很多。



CRAN
[Mirrors](#)
[What's new?](#)
[Task Views](#)
[Search](#)

About R
[R Homepage](#)
[The R Journal](#)

Software
[R Sources](#)
[R Binaries](#)
[Packages](#)
[Other](#)

Documentation
[Manuals](#)
[FAQs](#)
[Contributed](#)

This binary distribution of R and the GUI supports 64-bit Intel based Macs on Mac OS X 10.6 (Snow Leopard) or higher.

Please check the MD5 checksum of the downloaded image to ensure that it has not been tampered with or corrupted during the mirroring process. For example type

```
md5 R-3.1.1-snowleopard.pkg
```

in the *Terminal* application to print the MD5 checksum for the R-3.1.1-snowleopard.pkg image. On Mac OS X 10.7 and later you can also validate the signature using `pkgutil --check-signature R-3.1.1-snowleopard.pkg`

Files:

R-3.1.1-snowleopard.pkg MD5-hash: 193314c5eeea63e8ceaaa700e3a00803 SHA1-hash: 28d551fe99e16706aa29115f16d1c5bfeb136da4 (ca. 68MB)	R 3.1.1 binary for Mac OS X 10.6 (Snow Leopard) and higher, signed package. Contains R 3.1.1 framework, R.app GUI 1.65 in 64-bit for Intel Macs. The above file is an installer package which can be installed by double-clicking. Depending on your browser, you may need to press the control key and click on this link to download the file. This package contains the R framework, 64-bit GUI (R.app) and Tcl/Tk 8.6.0 X11 libraries. The latter component is optional and can be omitted when choosing "custom install", it is only needed if you want to use the <code>cc1tk</code> R package. GNU Fortran is NOT included (needed if you want to compile packages from sources that contain FORTRAN code) please see the tools directory .
R-3.1.1-mavericks.pkg MD5-hash: 08b4954756672113b5034804fa6d4016 SHA1-hash: 7f5845ff7e5df481c1c5e725ea0521fa32e4b028 (ca. 55MB)	R 3.1.1 binary for Mac OS X 10.9 (Mavericks) and higher, signed package. It contains the same software versions as above, but this R build has been built with Xcode 5 to leverage new compilers and functionalities in Mavericks not available in earlier OS X versions.
Mac-GUI-1.65.tar.gz MD5-hash: cc8f463f41caf3a10e3fae859c8b68ef	Sources for the R.app GUI 1.65 for Mac OS X. This file is only needed if you want to join the development of the GUI, it is not intended for regular users. Read the <code>INSTALL</code> file for further instructions.

Step5：執行下載的檔案完成安裝。

更新 R 的版本

因為 R 版本會不定時更新，該如何更新 R，其實跟安裝 R 的步驟一致，但在 step4 請都選 (2) 選項，不然會多很多不同版本的 GUI software。R 可以同時安裝多的版本在同一台電腦，該如何查詢電腦所有 R 的版本，請看以下介紹。

Windows

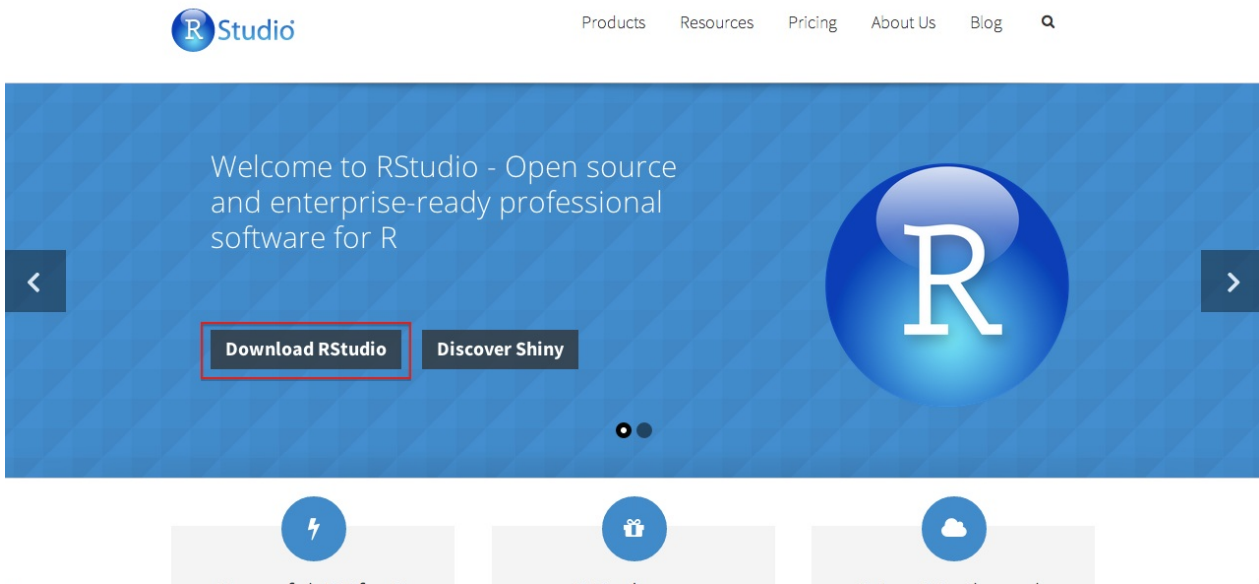
請查看 C:\Program Files\R

Mac OS X

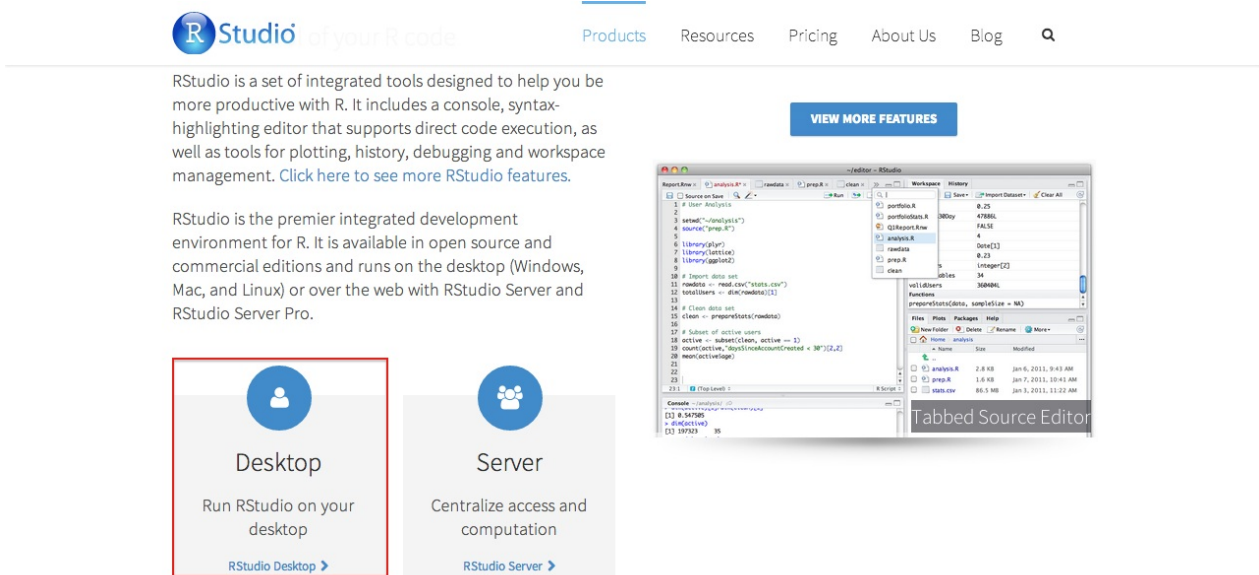
請查看 /Library/Frameworks/R.framework/Versions

安裝 RStudio


Step1：開啓 RStudio 官方網站



Step2：選擇 Desktop 版



Step3：選擇免費版



[Products](#)
[Resources](#)
[Pricing](#)
[About Us](#)
[Blog](#)

Overview

- Syntax highlighting, code completion, and smart indentation
- Execute R code directly from the source editor
- Quickly jump to function definitions
- Easily manage multiple working directories using projects
- Integrated R help and documentation
- Interactive debugger to diagnose and fix errors quickly
- Extensive package development tools

All of the features of open source; plus:

- A commercial license for organizations not able to use AGPL software
- Access to priority support

Support

Community forums only

- Priority Email Support
- 8 hour response during business hours (ET)

License

AGPL v3

Commercial Desktop License


Pricing

Free


\$995/year

[DOWNLOAD RSTUDIO DESKTOP](#)

[BUY NOW](#)



Step4：選擇作業系統(作者本身是 Mac)



[Products](#)
[Resources](#)
[Pricing](#)
[About Us](#)
[Blog](#)


RStudio requires R 2.11.1 (or higher). If you don't already have R, you can download it [here](#).

Installers for ALL Platforms

Installers	Size	Date	MD5
RStudio 0.98.994 - Windows XP/Vista/7/8	48 MB	2014-08-02	d10924e29736de2ce664c858d7c6d20e
RStudio 0.98.994 - Mac OS X 10.6+ (64-bit)	37.7 MB	2014-08-02	7f51b59ef178307f6816629eb6485166
RStudio 0.98.994 - Debian 6+/Ubuntu 10.04+ (32-bit)	56.2 MB	2014-08-02	6740bde2c8c4874059f7ef1a8b45e411
RStudio 0.98.994 - Debian 6+/Ubuntu 10.04+ (64-bit)	57.8 MB	2014-08-02	e2e4fcdffd203034e87a63f16fa11622c
RStudio 0.98.994 - Fedora 13+/openSUSE 11.4+ (32-bit)	56.4 MB	2014-08-02	03bf9e37b554b00092b6da7118688bed
RStudio 0.98.994 - Fedora 13+/openSUSE 11.4+ (64-bit)	57.8 MB	2014-08-02	bc22e2827de63e902977645a82288461

Zip/Tarballs

Zip/tar archives	Size	Date	MD5
RStudio 0.98.994 - Windows XP/Vista/7/8	66.7 MB	2014-08-02	62b41e6ad935859dc0951d8d2387dab9
Google Chrome RStudio 0.98.994 - Debian 6+/Ubuntu 10.04+ (32-bit)	56.1 MB	2014-08-02	189d1c17384c4df5a5d16b307916bc93
RStudio 0.98.994 - Debian 6+/Ubuntu 10.04+ (64-bit)	57.3 MB	2014-08-02	1c1f313f58c68266f5331a3361612



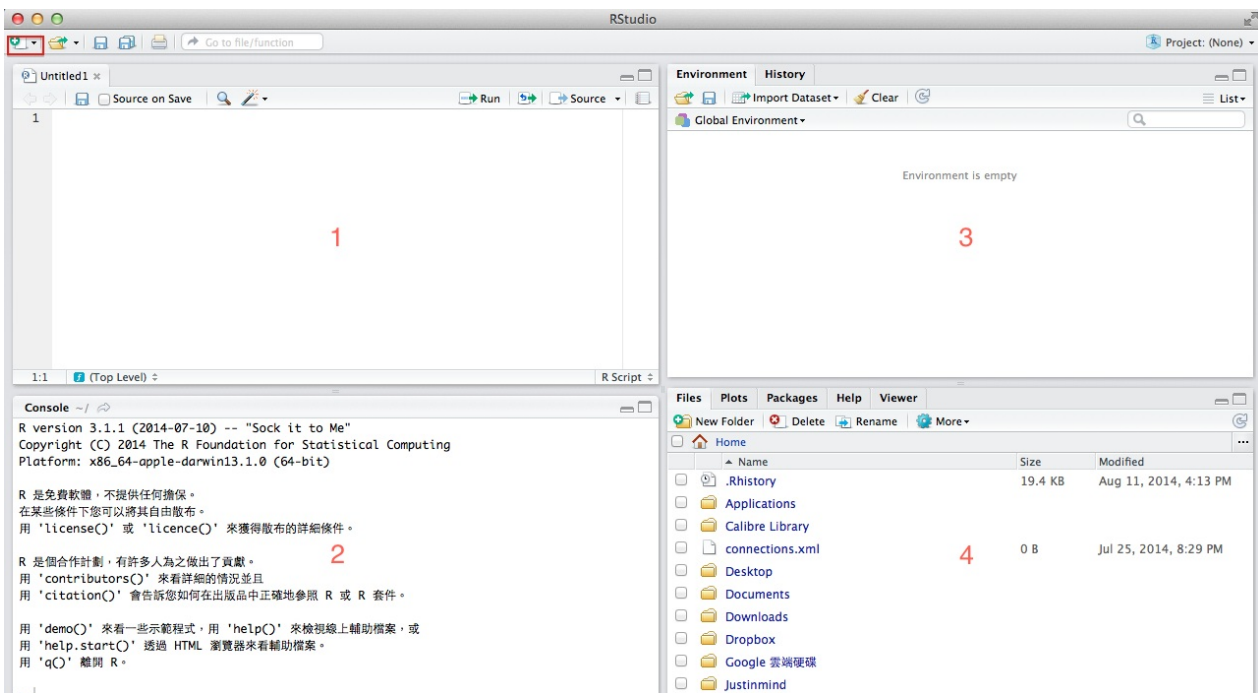
Step5：執行下載的檔案完成安裝。

Rstudio 基本介紹

簡介

RStudio 是一個 IDE，在作者剛開始學 R 的時候，其實並沒有 RStudio 只有 R 的 GUI 畫面，後來爲了提高開發效率，這套工具才就此誕生。Rstudio 特點很多就不再一一贅述，用過了以後就會覺得哪裡好用了，尤其是針對以前用過 GUI 的人更是一大福音。

介面概況



- Workspace (左上角)
- Console (左下角)
- Environment、History (右上角)
- Files、Plots、Packages、Help、Viewer (右下角)

Workspace

剛開啓時不會存在，可以在左上角有一個開啓新文件的 icon，選擇一個新的 R script 檔案，這邊主要是用來撰寫程式碼的部份。

Console

用來執行程式碼的地方，在 Workspace 選取欲執行的程式碼，按下以下快捷鍵就會發現程式碼自動在 Console 執行完成。

快捷鍵	Mac	Windows & Linux
Run current line/selection	Command + Enter	Ctrl + Enter

Environment、History

- Environment：是用來記載目前變數的數值，方便查看目前變數的狀況。
- History：是所有在 Console 執行過程式碼的歷史記錄。

Files、Plots、Packages、Help、Viewer

- Files：是讓使用者了解所在的工作環境是在哪個目錄，這個對讀取檔案非常重要。
- Plots：顯示使用者畫好的圖表。
- Package：記錄目前已安裝的 Package，打勾代表已經載入，安裝 Package 請點選「Install」。
- Help：查詢文件使用，在 Console 輸入 `help()`，`()` 輸入所要查詢方法的名稱，ex：`help(sum)`。
- Viewer：是用來顯示網頁或 html file。

快捷鍵

請參考 [RStudio 官方文件](#)

調整 RStudio 的 Appearance 與 Pane layout。

點選右上角的 RStudio > Preferences 裡面的 Appearance 與 Pane Layout。

- Appearance：可以調整字體、字體大小、Workspace 與 Console 的 theme。
- Pane Layout：調整開啟畫面的左上、左下、右上、右下的內容，可以自動調整。

實際操作

1. 開啓一個新的 R script 檔案
2. 輸入程式碼(程式碼如下)
3. 利用快捷鍵在 Console 執行
4. 在 Environment 會發現 x、y 與 z 的值。

```
x <- 3  
y <- 4  
z <- sum(x + y)  
help(sum)
```

註：<- 是 R 的 assignment operator 用來賦與變數值所使用，像 `x <- 3` 代表 x 是 3，另外提醒一下，變數的大小寫是有差別的，代表兩個不同變數，ex：x 與 X

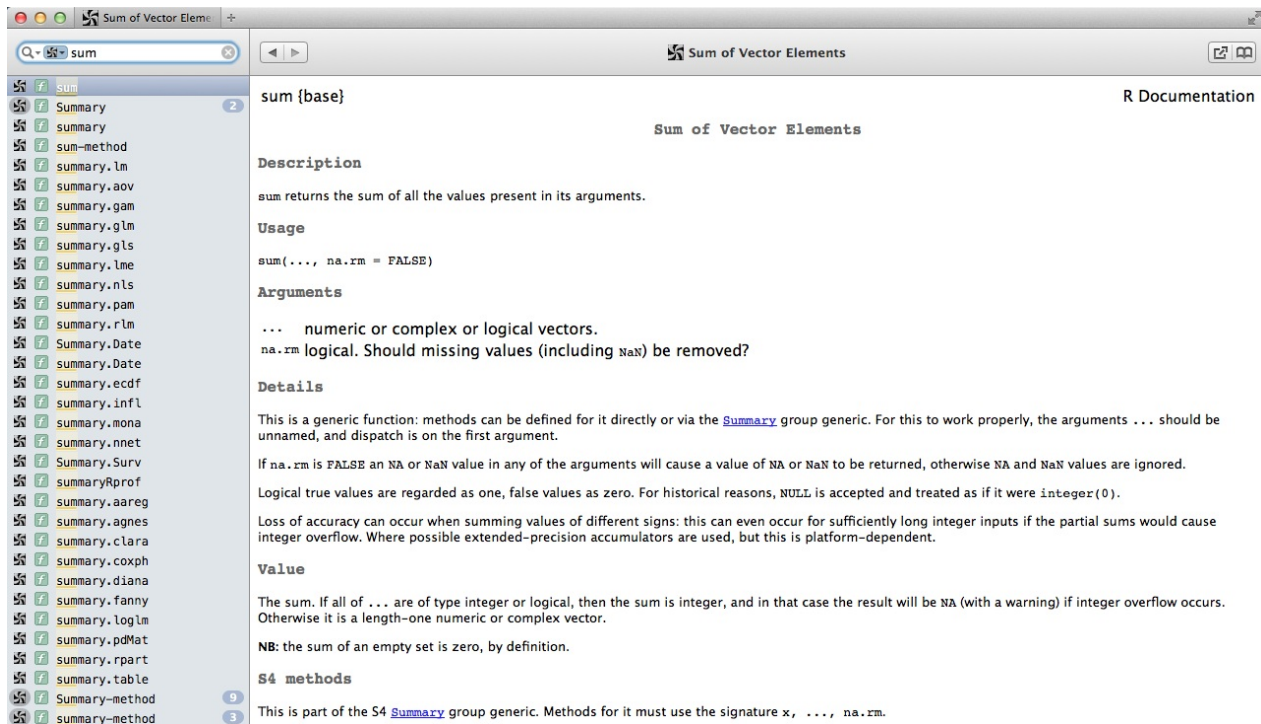
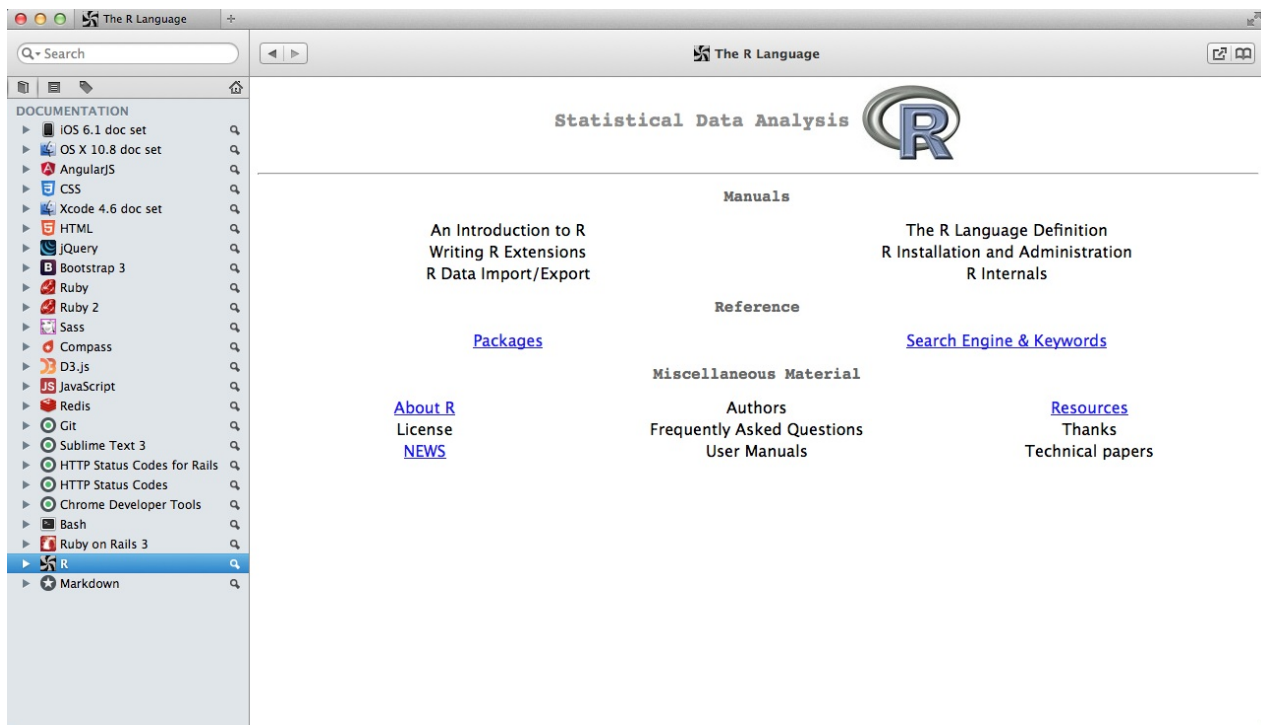
查詢函數的小技巧

- 使用 `help(函數名稱)` 等同於 `?函數名稱`
- 使用 `args(函數名稱)` 得知函數參數有哪些
- 使用 `example(函數名稱)`，ex：`example(dim)` 取得函數的使用範例，但是切記不是所有方法都有
- 在 Console 輸入函數名稱後，按下 tab 鍵可以看到函數相關資訊

推薦查詢文件小工具

- [Dash](#)：針對 Mac 使用者。
- [Velocity](#)：針對 Windows 使用者，聽說背後的文件來源是來自 Dash。

以下是 Dash 的畫面。



切換 R 的版本

因為有時候會安裝多個 R 的版本，但是 RStudio 通常系統會抓最新的版本當作目前的版本，但是有時需要換成舊的 R 版本？請參考以下程式碼(針對 Mac)。

Step1 查看目前的版本

```
cd /Library/Frameworks/R.framework/Versions
ls -al

total 8
drwxrwxr-x  5 root  admin  170  8 10 17:02 .
drwxrwxr-x  8 root  admin  272  8 10 17:02 ..
drwxrwxr-x  6 root  admin  204  8 11 22:08 3.0
drwxrwxr-x  6 root  admin  204  8 11 22:04 3.1
lrwxr-xr-x  1 root  admin    3  8 10 17:02 Current -> 3.1
```

Step 2 刪除 Current

```
rm -rf Current
ls -al

total 0
drwxrwxr-x  4 root  admin  136  8 11 22:26 .
drwxrwxr-x  8 root  admin  272  8 10 17:02 ..
drwxrwxr-x  6 root  admin  204  8 11 22:08 3.0
drwxrwxr-x  6 root  admin  204  8 11 22:04 3.1
```

Step 3 建立新的 Current，並指向 3.0 版本

```
ln -s 3.0 Current
ls -al

total 8
drwxrwxr-x  5 root  admin  170  8 11 22:26 .
drwxrwxr-x  8 root  admin  272  8 10 17:02 ..
drwxrwxr-x  6 root  admin  204  8 11 22:08 3.0
drwxrwxr-x  6 root  admin  204  8 11 22:04 3.1
lrwxr-xr-x  1 joe   admin    3  8 11 22:26 Current -> 3.0
```

註：Windows & Linux 的部分可以参考 [RStudio](#) 官方文件

Concole 開啓 R console

此部份在本機時應該不會用到，因為大部份都是使用 RSudio，但是在 Server(尤其是 Ubuntu) 就有機會用到，可以在 console 直接輸入「R」，就可以進入 R console，如果想要離開可以輸入「q()」，或者也可以使用 RStudio Server 版本。

```
~/Projects
$ R

R version 3.0.2 (2013-09-25) -- "Frisbee Sailing"
Copyright (C) 2013 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin10.8.0 (64-bit)

R 是免費軟體，不提供任何擔保。
在某些條件下您可以將其自由散布。
用 'license()' 或 'licence()' 來獲得散布的詳細條件。

R 是個合作計劃，有許多人為之做出了貢獻。
用 'contributors()' 來看詳細的情況並且
用 'citation()' 會告訴您如何在出版品中正確地參照 R 或 R 套件。

用 'demo()' 來看一些示範程式，用 'help()' 來檢視線上輔助檔案，或
用 'help.start()' 透過 HTML 瀏覽器來看輔助檔案。
用 'q()' 離開 R。

> q()
Save workspace image? [y/n/c]: n
~/Projects
$
```

安裝載入 package

安裝 package

安裝 package 有以下兩種方式：

- 透過右下角 Packages -> Install Packages 安裝套件
- 在 console 輸入指令安裝

```
> install.packages("ggplot2") # 下載 ggplot2 套件
```

載入 package

載入 package 有以下料、兩種方式：

- 透過右下角 Package，將要載入的 package 打勾即可。
- 在 console 輸入指令載入

```
> library(ggplot2) # ggplot2 一個畫圖套件。
```

```
> require(ggplot2) # library 與 require 都是載入 package，但是最大的  
差別在於，library 如果是載入的 package 不存在，是會發生 error 程式停止，  
但是 require 卻不會。
```

不過以上下載的套件是指發佈在 CRAN 上的，但如果沒有發佈在 CRAN，而是發佈在 Github 上該怎麼辦，可以利用 devtools package 實作此部份。

```
> install.packages("devtools")  
> require(devtools)  
> install_github("rpensoft", "ropensci") # package 名稱爲 rpensoft，  
repository 名稱爲 ropensci，對應到的 github 網址爲 https://github.com/ropensci/rpensoft。
```

至於 Bioconductor package 下載的部份，可以參考[官網](#)的介紹。

Chapter 2 - 基本運算

1. 資料屬性
2. 常見運算

資料屬性

R 的基本資料屬性包含以下五種，可用 `class` 函數判斷資料屬性

- `character`：文字字串，用 `"` 包起來，ex：`"test"`
- `numeric`：實數
- `integer`：整數
- `complex`：複數
- `logical`：True 或 False

```
> class("test")
[1] "character"

> class(10.10)
[1] "numeric"

> class(10)
[1] "numeric"

> class(as.integer(3)) # 因為 R 計算上都是是以雙倍精確度來計算，所以必須指定為 integer，不然都會被當成 numeric 看待。
[1] "integer"

> class(as.integer(3.1)) # as.integer 可以將不是整數的數值變成整數
[1] "integer"

> class(as.integer(T)) # as.integer(T) = 1
[1] "integer"

> class(as.integer(F)) # as.integer(T) = 0
[1] "integer"

> class(2+2i)
[1] "complex"

> class(TRUE) # 注意都要大寫，不可寫 True，但可以簡化成 T
[1] "logical"

> class(T)
[1] "logical"
```

註：

- `as.integer` 切記不可以傳 `character` 進去，因為會產生 `NA`，如果傳 `complex` 進去，則會將虛數的部份則會自動捨棄。
- 可以用 `is.integer(x)` 判斷是否為整數。
- `complex` 也有跟 `integer` 類似的函數，`as.complex` 與 `is.complex`。
- `logical` 也有跟 `integer` 類似的函數，`as.logical` 與 `is.logical`

	character	numeric	integer	complex	logical
as.integer	X	O	O	O	O
is.integer	F	F	T	F	F

常見運算

加減乘除

```
> 1 + 2  
[1] 3  
  
> 1 - 2  
[1] -1  
  
> 1 * 2  
[1] 2  
  
> 1 / 2  
[1] 0.5
```

次方、平方根、商數與餘數

```
> 2 ^ 3 # 2 的 3 次方  
[1] 8  
  
> 2 ** 3 # 2 的 3 次方  
[1] 8  
  
> sqrt(4) # 4 的平方根  
[1] 2  
  
> 27 ^ (1/3) # 27 的立方根  
[1] 3  
  
> 11 %/% 5 # 11 除以 5 的商數  
[1] 2  
  
> 11 %% 5 # 11 除以 5 的餘數  
[1] 1
```

註：`#` 為 R 的註解符號，並不會執行。

sign：判斷是正、負數或 0

- 正數回傳 1
- 負數回傳 -1
- 0 回傳 0

```
> sign(10)
[1] 1

> sign(0)
[1] 0

> sign(-10)
[1] -1
```

abs：取絕對值

```
> abs(10)
[1] 10

> abs(0)
[1] 0

> abs(-10)
[1] 10
```

log

```
> log(10) # log 以 e 為底
[1] 2.302585

> log1p(9) # log(x) = log1p(x - 1)
[1] 2.302585

> log(10, 2) # 指定 log 以 2 為底
[1] 3.321928

> log2(10) # log2 代表以 2 為底
[1] 3.321928

> log10(10) # log10 代表以 10 為底
[1] 1
```

exp

```
> exp(10)
[1] 22026.47

> expm1(10) # expm1(x) = exp(x) - 1
[1] 22025.47
```

Chapter 3 - 變數與資料

1. 變數
2. 向量
3. 矩陣
4. 因子
5. 串列
6. 資料框架
7. 時間數列
8. 指標

變數

R 在給予變數值時是利用「<-」並不是程式語言中常見的「=」，在 [Google's R Style Guide](#) 與 [R 官方文件](#) 都強調不該使用「=」，因為在某些狀況是會失效的。另外在 R 變數命名上，大小寫是有區別的，所以 x 與 X 其實是不同的變數。

```
> x <- 1
> y <- 2
> x + y
[1] 3

> x1 <- x2 <- 1 # x1 與 x2 都是 1
> x1 + x2
[1] 2
```

R 的變數可以重複給予值，不會因為資料屬性的不同而發生錯誤，會因最後所給予的值為結果。所以程式碼複雜時，常常會因為一個變數重複給予不同的值而發生錯誤，這時可以用 `exists` 函數檢查。


```
> x = 1
> x
[1] 1
> x = 1.3
> x
[1] 1.3
> x = 1 + 2i
> x
[1] 1+2i
> x = "test"
> x
[1] "test"
> x = FALSE
> x
[1] FALSE

> x = 10
> exists("x")
[1] TRUE
```

NA 與 NULL

NA 代表是個空物件，已經有物件但是裡面沒東西，NULL 則是根本沒有任何東西，更詳細比較請參考 [R Bloggers R : NA vs. NULL](#)

向量

利用 **c(...)** 建立向量，但切記向量元素必須是同個資料屬性。

```
> c(1, 2, 3)
[1] 1 2 3

> c(1, TRUE, "test") # 全部都變成 character
[1] "1"      "TRUE"   "test"

> c(1.1, TRUE, "test") # 全部都變成 character
[1] "1.1"    "TRUE"   "test"

> c(1+2i, TRUE, "test") # 全部都變成 character
[1] "1+2i"   "TRUE"   "test"

> c(1, TRUE) # 全部自動轉成 integer
[1] 1 1

> c(1.1, TRUE) # 全部自動轉成 numeric
[1] 1.1 1.0

> c(1+2i, TRUE) # 全部自動轉成 complex
[1] 1+2i 1+0i

> c(1, 1.1) # 全部自動轉成 numeric
[1] 1.0 1.1

> c(1, 1.1, 1+2i) # 全部自動轉成 complex
[1] 1.0+0i 1.1+0i 1.0+2i
```

註：經由以上比較後，可以得到當放入的形態不同時，會被轉成同一形態，且可以每個形態的強弱不同，以下是強到弱排序。

character > complex > numeric > integer > logical

透過指標與名稱提取資料

我們可以透過以下二種方式取得向量元素。

- 指標
- 元素名稱

另外可以搭配 [] 或 [[]]，這樣分別會回傳向量元素的所有資訊或向量元素的數值，總共可以分成以下四種狀況。

- `x[i]`：回傳向量元素所有資訊
- `x[[i]]`：只回傳向量元素的值
- `x[元素名稱]`：回傳向量元素所有資訊
- `x[[元素名稱]]`：只回傳向量元素的值

```
> x <- c(joe=12, vicky=14, bob=17)

> x[1]
joe
  12

> x[[1]]
[1] 12

> x["joe"]
joe
  12

> x[["joe"]]
[1] 12

> x[1:2] # 一次取多個向量元素
joe vicky
  12   14
```

`c(...)` 類似的函數 `x:y`、`seq` 與 `rep`

- `x:y`：回傳 `x` 到 `y` 的整數向量，所以 `x` 與 `y` 都是整數。
- `seq(s, e, by)`：產生一個等差級數的向量。
 - `s` 是初始值

- `e` 是結束值
- `by` 是遞增值，預設是 1
- `rep(x, times, each)`：產生一個重覆循環的向量。
 - `x` 是需重覆循環的數值
 - `times` 是重覆循環次數
 - `each` 是 `x` 內元素重覆的次數

```
> 1:5
[1] 1 2 3 4 5

> seq(1, 5)
[1] 1 2 3 4 5

> seq(1, 5, 0.3) # 就算沒有剛好加到跟結束值一樣也沒關係
[1] 1.0 1.3 1.6 1.9 2.2 2.5 2.8 3.1 3.4 3.7 4.0 4.3 4.6 4.9

> rep(c(1, 2, 3), times = 3, each = 2)
[1] 1 1 2 2 3 3 1 1 2 2 3 3 1 1 2 2 3 3

> rep(1:4, times = 3, each = 2)
[1] 1 1 2 2 3 3 4 4 1 1 2 2 3 3 4 4 1 1 2 2 3 3 4 4
```

基本相關函數

- 向量加減乘除
- `length`：計算向量中的元素個數。
- `sum`：將向量所有元素加總。
- `prod`：將向量所有元素相乘。
- `cumsum`：回傳元素累加向量。
- `cumprod`：回傳元素累乘向量。
- `sort`：將向量元素排列，產生排序過的向量。
- `rank`：回傳各向量元素的排序值。

```
> c(3, 4, 2) + c(3, 4, 2)
[1] 6 8 4
> c(3, 4, 2) - c(3, 4, 2)
[1] 0 0 0
> c(3, 4, 2) * c(3, 4, 2)
[1] 9 16 4
> c(3, 4, 2) / c(3, 4, 2)
[1] 1 1 1

> length(c(3, 4, 2))
[1] 3

> sum(c(3, 4, 2))
[1] 9

> cumsum(c(3, 4, 2))
[1] 3 7 9

> cumprod(c(3, 4, 2))
[1] 3 12 24

> prod(c(3, 4, 2))
[1] 24

> sort(c(3, 4, 2))
[1] 2 3 4

> rank(c(3, 4, 2))
[1] 2 3 1
```

上述向量加減乘除時，向量個數都是一樣的狀況，但不同的狀況會發生什麼結果，請看以下測試。

加跟減

如果長度較長的向量長度是較短的倍數的話是可以相加或相減的。

```
> c(1, 2) + c(1, 2, 3)
[1] 2 4 4
警告訊息：
In c(1, 2) + c(1, 2, 3) : 較長的物件長度並非較短物件長度的倍數

> c(1, 2) + c(1, 2, 3, 4) # 1+1 2+2 1+3 2+4
[1] 2 4 4 6
```

乘跟除

乘跟除的情況與加跟減狀況一致，只是當長度不是倍數時會有結果但也會警告。

```
> c(1, 2) * c(1, 2, 3) # 1*1 2*2 1*3
[1] 1 4 3
警告訊息：
In c(1, 2) * c(1, 2, 3) : 較長的物件長度並非較短物件長度的倍數

> c(1, 2) * c(1, 2, 3, 4) # 1*1 2*2 1*3 2*4
[1] 1 4 3 8
```

陣列

利用 **rbind**、**cbind** 與 **array** 函數建立陣列

陣列可視為多維度的向量變數，跟向量一樣，所有陣列元素的資料屬性必須一致。

```
> x <- c(1, 2, 3)
> y <- c(4, 5, 6)

> rbind(x, y) # rbind 是利用 row(橫) 合併。
  [,1] [,2] [,3]
x    1    2    3
y    4    5    6

> cbind(x, y) # cbind 是利用 column(直) 合併。
      x y
[1,] 1 4
[2,] 2 5
[3,] 3 6

> array(x, c(1, 3)) # c(1, 3) 代表產生 1 x 3 陣列
  [,1] [,2] [,3]
[1,]   1   2   3

> array(x, c(2, 3)) # c(2, 3) 代表產生 2 x 3 陣列
  [,1] [,2] [,3]
[1,]   1   3   2
[2,]   2   1   3

> array(x, c(3, 3)) # c(3, 3) 代表產生 3 x 3 陣列
  [,1] [,2] [,3]
[1,]   1   1   1
[2,]   2   2   2
[3,]   3   3   3
```

透過指標提取資料

陣列與向量相同，可以透過指標或名稱選取陣列的元素。

```
> x <- c(1, 2, 3)
> y <- c(4, 5, 6)

> z = rbind(x, y)
> z
  [,1] [,2] [,3]
x    1    2    3
y    4    5    6

> z[,1] # 選取第一行(column、直)
x y
1 4

> z[1,] # 選取第一列(row、橫)
[1] 1 2 3

> z[1,1:2] # 選取第一列第一到二行
[1] 1 2
```

基本相關函數

- 陣列加減乘除
- `length`：計算陣列中的所有元素個數。
- `dim`：列出維度資訊
- `ncol`、`nrow`：計算(column、直) 或 (row、橫) 個數。
- `aperm`：將陣列轉置

```
> x <- c(1, 2, 3)
> y <- c(4, 5, 6)

> z = rbind(x, y)
> z
  [,1] [,2] [,3]
x    1    2    3
y    4    5    6
```



```
> z + z
  [,1] [,2] [,3]
x    2    4    6
y    8   10   12

> z - 2*z
  [,1] [,2] [,3]
x   -1   -2   -3
y   -4   -5   -6

> z * z # 相對應的陣列元素相乘
  [,1] [,2] [,3]
x    1    4    9
y   16   25   36

> z / z # 相對應的陣列元素相除
  [,1] [,2] [,3]
x    1    1    1
y    1    1    1

> length(z)
[1] 6

> dim(z) # 前者是 row，後者是 column。
[1] 2 3
> ncol(z)
[1] 3
> nrow(z)
[1] 2

> aperm(z) # 等同是從 rbind 轉成 cbind
      x y
[1,] 1 4
[2,] 2 5
[3,] 3 6
```

矩陣

利用 **matrix** 建立矩陣

當陣列是 2 維的狀況就是所謂的矩陣，可以利用 **matix** 產生矩陣，也可以用之前產生陣列的方法實作。

```
> matrix(c(1:4), nrow = 2, ncol = 2) # 預設是按照 column 填入資料
      [,1] [,2]
[1,]    1    3
[2,]    2    4

> matrix(c(1:4), nrow = 2, ncol = 2, byrow = TRUE) # 可以更改成按照 row 填入資料
      [,1] [,2]
[1,]    1    2
[2,]    3    4
```

透過指標提取資料

矩陣跟陣列一樣，還是可以透過指標選取矩陣的部份元素。

```
> x <- c(1, 2, 3)
> y <- c(4, 5, 6)

> z = rbind(x, y)
> z
  [,1] [,2] [,3]
x    1    2    3
y    4    5    6

> z[,1] # 選取第一行(column、直)
x y
1 4

> z[1,] # 選取第一列(row、橫)
[1] 1 2 3

> z[1,1:2] # 選取第一列第一到二行
[1] 1 2
```

基本相關函數

- `t(x)`：將矩陣轉置。
- `%*%`：矩陣相乘。
- `diag`：產生一個對角矩陣，或回傳矩陣的對角線向量
- `det`：計算矩陣行列式值，一定是要對稱矩陣。
- `solve`：傳回矩陣的反矩陣，非常適合解線性方程式。
- `eigen`：計算矩陣的特徵向量與特徵值。
- `rownames`：修改或查詢 row 名稱。
- `colnames`：修改或查詢 column 名稱。

```
> x <- c(1, 2, 3)
> y <- c(4, 5, 6)

> z <- rbind(x, y)
  [,1] [,2] [,3]
x    1    2    3
y    4    5    6
```

```

> t(z)
      x y
[1,] 1 4
[2,] 2 5
[3,] 3 6

> z %*% z # 矩陣相乘要符合前者 column 維度 = 後者 row 維度，如果沒有會
發生錯誤！
錯誤在z %*% z : 非調和引數
> v <- z %*% t(z)
      x y
x 14 32
y 32 77

> w <- diag(c(1,2,3)) # 傳入向量回傳一個對角矩陣
      [,1] [,2] [,3]
[1,] 1 0 0
[2,] 0 2 0
[3,] 0 0 3

> diag(w) # 傳入矩陣回傳矩陣對角線向量
[1] 1 2 3

> det(v) #一定要對稱矩陣才可以計算。
[1] 54

> solve(v)
      x y
x 1.4259259 -0.5925926
y -0.5925926 0.2592593

> b = c(1,1) 解 Ax = b，求出 x 向量 A：變數 v，b：變數 b
> solve(v,b)
      x y
0.8333333 -0.3333333

> u = matrix(1:9, nrow = 3, ncol = 3)
> u

```

```

      [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
> eigen(u) # 特徵值
$values
[1] 1.611684e+01 -1.116844e+00 -5.700691e-16

$vectors # 特徵向量
      [,1]      [,2]      [,3]
[1,] -0.4645473 -0.8829060  0.4082483
[2,] -0.5707955 -0.2395204 -0.8164966
[3,] -0.6770438  0.4038651  0.4082483

> rownames(z) # 還沒修改的時候，是 x 與 y，因為當初是利用兩個向量，利用
rbind 組成，所以會利用向量的變數稱名當作 row 名稱。
[1] "x" "y"
> rownames(z) <- c("第一行", "第二行")
> rownames(z)
[1] "第一行" "第二行"

> colnames(z)
NULL
> colnames(z) <- c("第一列", "第二列", "第三列")
> colnames(z)
[1] "第一列" "第二列" "第三列"

> z
      第一列 第二列 第三列
第一行    1     2     3
第二行    4     5     6

```

因子

利用 **factor** 建立因子

因子有點像經過分級之後的向量，因子大多可以用在統計上的迴歸分析與實際設計等。

```
> x <- c(1, 2, 4, 3, 1, 2, 3, 4, 1)
> factor(x)
[1] 1 2 4 3 1 2 3 4 1
Levels: 1 2 3 4

> factor(x, labels = c("一", "二", "三", "四")) # 可自訂 Level 的名稱。
[1] 一 二 四 三 一 二 三 四 一
Levels: 一 二 三 四

> factor(x, ordered = TRUE) # ordered 代表可做排序
[1] 1 2 4 3 1 2 3 4 1
Levels: 1 < 2 < 3 < 4

> factor(c(1, 2, 1, NA, 2), exclude = NA) # 可利用 exclude 排除特定資料。
[1] 1    2    1    <NA> 2
Levels: 1 2

> factor(c(1, 2, 1, NA, 2), exclude = 2)
[1] 1    <NA> 1    <NA> <NA>
Levels: 1 <NA>

> factor(c(1, 2, 1, NA, 2), exclude = NULL) # 不排除任何資料。
[1] 1    2    1    <NA> 2
```

透過指標提取資料

```
> x[1] # [] 與 [[]] 結果一致，因為因子只有值沒有其他相關資料。  
[1] 1  
> x[[1]]  
[1] 1  
  
> x[1:2] # 指標可以使用向量。  
[1] 1 2  
  
> x[c(1, 3, 5)]  
[1] 1 4 1
```

基本相關函數

- `is.factor`：判斷是否為因子。
- `as.factor`：將變數轉為因子。
- `is.ordered`：判斷是否為排序過的因子。
- `as.ordered`：將因子排序。
- `which`：找出符合條件的指標。

```
> x <- c(1, 2, 4, 3, 1, 2, 3, 4, 1)

> as.factor(x)
[1] 1 2 4 3 1 2 3 4 1
Levels: 1 2 3 4

> is.factor(x)
[1] FALSE

> is.factor(as.factor(x))
[1] TRUE

> is.ordered(factor(x, ordered = TRUE))
[1] TRUE

> is.ordered(factor(x, ordered = FALSE))
[1] FALSE

> as.ordered(factor(x))
[1] 1 2 4 3 1 2 3 4 1
Levels: 1 < 2 < 3 < 4

> which(x == 1) # 找出 x 等於 1 的指標
[1] 1 5 9
```


列表

利用 **list** 建立列表

列表跟向量很相似，但最大的不同在於列表可以包含不同資料屬性的資料。

```
> x <- list(a = 1, b = TRUE, c = "test", d = c(1, 2, 3))
> x
$a
[1] 1

$b
[1] TRUE

$c
[1] "test"

$d
[1] 1 2 3
```

透過指標與名稱提取資料

```
> x <- list(a = 1, b = TRUE, c = "test", d = c(1, 2, 3))

> x[1]
$a
[1] 1

> x[[1]]
[1] 1

> x$b # 是利用 % 加上名稱提取資料
[1] TRUE

> x[[4]][1] # x[[4]] 取出第四個值，因為第四個值是向量，所以可以在取一次指標，取出向量的元素值。
[1] 1
```

基本相關函數

- `as.list`：建立列表
- `is.list`：判斷是否為列表
- `attributes`：查看所有元素的名稱，`names` 也有相同功能。

```
> x <- list(a = 1, b = TRUE, c = "test", d = c(1, 2, 3))

> as.list(c(1,2,3))
[[1]]
[1] 1

[[2]]
[1] 2

[[3]]
[1] 3

> is.list(x)
[1] TRUE

> attributes(x)
$names
[1] "a" "b" "c" "d"

> names(x)
[1] "a" "b" "c" "d"
```

資料框架

利用 **data.frame** 建立資料框架

資料框架類似資料表，常當作大量資料集，例如：匯入外部檔或讀取資料庫資料等。

```
> name <- c("Joe", "Bob", "Vicky")
> age <- c("28", "26", "34")
> gender <- c("Male", "Male", "Female")
> data <- data.frame(name, age, gender)
> View(data) # 自動點選 data 變數就會開啓資料的畫面。
```

透過指標與名稱提取資料

資料框架的提取資料方法跟矩陣或陣列的都很類似。

```
> data
      n  a    g
r1   Joe 28  Male
r2   Bob 26  Male
r3 Vicky 34 Female

> data[1,]
  name age gender
1  Joe  28   Male

> data[,1]
[1] Joe  Bob  Vicky
Levels: Bob Joe Vicky

> data[1, 1]
[1] Joe
Levels: Bob Joe Vicky

> data[, "name"]
[1] Joe  Bob  Vicky
Levels: Bob Joe Vicky

> data[1:2, "name"]
[1] Joe Bob
Levels: Bob Joe Vicky

> data$name[1:2]
[1] Joe Bob
Levels: Bob Joe Vicky
```

基本相關函數

- `head`：取得資料框架前六比資料(預設是 6)。
- `names`：修改或查詢 `column` 名稱。
- `colnames`：設定 `column` 名稱。
- `row.names`：修改或查詢 `row` 名稱。
- `rownames`：設定 `row` 的名稱
- `summary`：顯示資料基本資訊。

```
> data
      n  a    g
r1   Joe 28   Male
r2   Bob 26   Male
r3 Vicky 34 Female

> head(data) # 因為筆數不夠多，所以全部都顯示。
      n  a    g
r1   Joe 28   Male
r2   Bob 26   Male
r3 Vicky 34 Female

> head(data, 1L) # 只顯示第一筆資料。
      n  a    g
r1 Joe 28 Male

> names(data)
[1] "name" "age" "gender"
> names(data) <- c("n", "a", "g")
> names(data)
[1] "n" "a" "g"
> colnames(data)
[1] "n" "a" "g"

> row.names(data)
[1] "1" "2" "3"
> row.names(data) <- c("r1", "r2", "r3")
> row.names(data)
[1] "r1" "r2" "r3"
> rownames(data)
[1] "r1" "r2" "r3"

> summary(data)
      name    age    gender
Bob   :1   26:1  Female:1
Joe   :1   28:1   Male  :2
Vicky:1   34:1
```


Chapter 4 - 資料匯入與輸出

1. 匯入資料
2. 輸出資料
3. 讀取資料庫的資料

匯入資料

透過 **read.table** 匯入資料。

read.table 可以讀取大多數的 ASCII 資料，以下先以 CSV 檔為代表，因為是目前最普遍見到的匯入資料格式。

```
> data <- read.table("Desktop/data.csv", header = TRUE, sep = ",",  
  ) # 檔案路徑是相對於目前的工作目錄，header 是指資料是否有包含欄位名稱，sep  
  是指資料的分隔符號。  
  
> data <- read.table("Desktop/data.csv", header = TRUE, sep = ",",  
  , col.names = c("時間", "新聞標題")) # col.names 設定 column 欄位名  
  稱。  
  
> data <- read.table("Desktop/data.csv", header = FALSE, sep = "  
  ,", skip = 10) # skip 是指跳過前 X 筆資料，這個部份要注意，要跳過資料，c  
  olumn 欄位就不可以出現在資料裡，因為它也被算在要 skip 部份。  
  
> data <- read.table("Desktop/data.csv", header = TRUE, sep = ",",  
  , encoding = "UTF-8") # encoding 是指定檔案的文字編碼  
  
> data <- read.table("Desktop/data.csv", header = TRUE, sep = ",",  
  , na.strings = NA) # na.strings 指定發生 NA 要用什麼符號代替。
```

文字編碼問題

匯入 CSV 檔的時候會碰到一種比較特別的問題，就是作業系統編碼不同的問題，Windows 的中文編碼是 big5，而 Linux / Mac 都是 UTF-8，所以在 Linux / Mac 匯入來自於 Windows CSV 檔常常會發生亂碼，那該如何解決此問題，本人的做法是將資料讀進來轉成 UTF-8，在輸出一份 CSV 檔，以下先以一個 CSV 檔為主，加以調整修改就可以改成一次跑一個資料夾下的所有 CSV 檔。

```

> data <- readLines("Desktop/A_lvr_land_A.CSV", encoding="big5")
# 讀取實價登入資料，是一行一行讀取進來。
> data <- iconv(data, "big5", "utf8") 將資料轉成 UTF-8。

> column_count <- length(strsplit(data[1], ",")[[1]])
row_count <- length(data) # 計算 column 與 count 個數。

> fix_data <- matrix(NA, nrow = row_count, ncol = column_count)
# 建立一個空的 NA 矩陣，維度來自於 row_count 與 column_count。

> for(row in 1:row_count) {
+   for(col in 1:column_count) {
+     fix_data[row,col] <- strsplit(data[row], ",")[[1]][col]
+   } # 執行 for loop 將資料塞入 fix_data。
+ }

> write.table(fix_data[2:row_count,], file = "fix_A_lvr_land_A.CSV", sep = ",", col.names = fix_data[1,]) # 將資料輸出，輸出注意一點，因為第一 row 是欄位名稱，所以記得指標要從 2 開始，指標 1 的部分要放到 col.names。

```

讀取 XML 檔案。

這邊順便也介紹讀取 XML 方法，是因為讀取 XML 檔案比較不會發生上述的編碼問題，所以如果兩者都會的話，以後有提供 CSV 與 XML 檔案時，就可以選 XML 檔案以免製造自己的麻煩。

```

> library(XML) # 要先安裝 XML package 再載入。
> data <- xmlToDataFrame("Desktop/A_lvr_land_A.XML")

```

匯入 RDA 檔案

匯入 RDA 檔案有以下兩種方式。

- 在工作目錄直接點選 RDA 檔案，選擇匯入即可。

- 在 console 下 load(檔案名稱)。

輸出資料

透過 **write.table** 輸出資料

以下利用 `write.table` 輸出 CSV 檔案。

```
data <- iris # iris 是 R 內建的資料。  
write.table(data, file = "test.CSV", sep = ",")
```

輸出 **XML** 檔案

跟輸入 XML 檔案一樣，是使用 XML package 來實做，但會需要利用到建立 tag 的函數。

```
> data <- iris  
> xml <- xmlTree()  
> xml$addTag("document", close = FALSE) # 建立一個名為 document 的  
tag。  
> for (i in 1:nrow(data)) {  
+   xml$addTag("row", close = FALSE) # 建立一個名為 row 的 tag。  
+   for (j in names(data)) {  
+     xml$addTag(j, data[i, j]) # j 為欄位名稱，所以是依序建立不同欄為  
名稱的 tag，並賦予值與結束此 tag。  
+   }  
+   xml$closeTag() # 建立 tag 時，如果有下參數 close = FALSE 的話，記  
得要在結束 tag 地方下 closeTag()。  
+ }  
> xml$closeTag()  
> saveXML(xml, "test.xml")  
[1] "test.xml"
```

XML 檔案輸出結果。

```
<?xml version="1.0" encoding="UTF-8"?>
<document>
  <row>
    <Sepal.Length>5.1</Sepal.Length>
    <Sepal.Width>3.5</Sepal.Width>
    <Petal.Length>1.4</Petal.Length>
    <Petal.Width>0.2</Petal.Width>
    <Species>setosa</Species>
  </row>
  .
  .
  .
  .
</document>
```

輸出 **RDA** 檔案

利用 `save` 函數輸出 RDA 檔案。

```
> data <- iris
> save(data, file="test.rda")
```

讀取資料庫的資料

讀取資料庫的資料

利用 DBI、gWidgets、RMySQL 與 dbConnect 等 package 讀取資料庫資料，以下是讀取 MySQL 資料庫當作範例。

```
> library(DBI) # DBI 是 R Database Interface
> library(gWidgets)
> library(RMySQL) # MySQL 在 R 的 Interface
> library(dbConnect)
> con <- dbConnect(MySQL(), dbname = "house_development", username="root", password="123456") # 建立資料庫連線
>
> dbSendQuery(con, "SET NAMES utf8") # 需設定 UTF-8，不然中文會亂碼。
<MySQLResult:(94576,0,0)>
> dbClearResult(dbListResults(con)[[1]])
[1] TRUE
>
> dbGetQuery(con, "show variables like 'character_set_%'") # 查詢
資料庫基本設定
Variable_name Value
character_set_client utf8
character_set_connection utf8
character_set_database utf8
character_set_filesystem binary
character_set_results utf8
character_set_server utf8
character_set_system utf8
character_sets_dir /usr/local/Cellar/mysql/5.6.12/share/mysql/charsets/

data = dbGetQuery(con, "select * from raws") # 使用 SQL query 讀取資料。
```

註：如果資料庫不太熟，卻又想學習基本的語法的話，推薦 [新SQL 基礎講座 <增訂第二版>](#)，此本的好處是記載不同資料庫的語法且淺顯易懂。

Chapter 5 - 流程控制

1. 邏輯判斷式
2. 條件執行
3. 迴圈結構

邏輯判斷式

常見的邏輯判斷符號

以下介紹常見的邏輯判斷的符號。

- `<`、`>`：小於、大於。
- `<=`、`>=`：小於等於、大於等於。
- `==`、`!=`：等於、不等於。
- `A %in% B`：A 是否在 B 中。
- `&&`、`&`：交集，`&` 適用於向量式的邏輯判斷，`&&` 適用於單一值的邏輯判斷。
- `||`、`|`：聯集，`|` 適用狀況與 `&` 相同，`||` 適用狀況與 `&&` 相同。

```
> x <- 5
> y <- 10

> x > 3
[1] TRUE
> x >= 6
[1] FALSE

> x <= 6
[1] TRUE
> x < 3
[1] FALSE

> !(x > 3) # 前面多加一個有否定功能，所以 True 變成 False，如果是 False
           則變成 True。
[1] FALSE

> x %in% c(1:6)
[1] TRUE

> x > 4 || y > 10
[1] TRUE
> x > 4 && y > 10
[1] FALSE

> z = c(1,2,3)
> z > 0 & z > -1
[1] TRUE TRUE TRUE
> z > 0 && z > -1 # && 只可以比較單一值，所以只有抓 z 的第一元素跟 0 與
                  -1 比較。
[1] TRUE
```

條件執行

常見的條件執行

以下介紹三種常見的條件執行。

- if else
- if else if else
- switch

```
# if A 判斷式
# A 判斷式為 True，會執行此區段程式碼。
# else
# A 判斷式為 False，會執行此區段程式碼。

> x <- 1

> if (x > 0) {
+   y <- 5
+ } else {
+   y <- 10
+ }

> if (x > 0) y <- 5 else y <- 10 # 單行的寫法。
> y
[1] 5

> y <- ifelse(x > 0, 5, 10) # 利用 ifelse(判斷式, True 給 5, False
  給 10)。
> y
[1] 5

# if A 判斷式
# A 判斷式為 True，會執行此區段程式碼。
# else if B 判斷式
# B 判斷式為 True，會執行此區段程式碼。
# else
```

A 與 B 判斷式都是 False，會執行此區段程式碼。

```
+ y <- 5
+ } else if (x > 2) {
+ y <- 10
+ } else {
+ y <- 3
+ }
> y
[1] 3
```

switch(回傳數值代表執行第幾個程式片段， 程式片段 1, ..., 程式片段 N)

switch(回傳名稱代表執行哪個名稱的程式片段， 程式名稱 A 片段, ..., 程式名稱 N 片段)

```
> switch(3, 10, 3 + 5, 3 / 3)
```

```
[1] 1
```

```
> switch(1, 10, 3 + 5, 3 / 3)
```

```
[1] 10
```

```
> switch(2, 10, 3 + 5, 3 / 3)
```

```
[1] 8
```

```
> switch("first", first = 1 + 1, second = 1 + 2, third = 1 + 3)
```

```
[1] 2
```

```
> switch("second", first = 1 + 1, second = 1 + 2, third = 1 + 3)
```

```
[1] 3
```

```
> switch("third", first = 1 + 1, second = 1 + 2, third = 1 + 3)
```

```
[1] 4
```

迴圈結構

常見的迴圈結構

以下介紹三種迴圈與兩種改變迴圈狀態的方法。

- for
- while
- repeat
- break
- next

```
> y <- 0
> for (x in c(1:10)) y <- x + y # 1 加到 10，迴圈就是重複執行相同動作
，x 依序帶入 1 到 10，第一次帶入 1 時，y = 0，所以是 1 + 0 = 1，第二次帶
入時，x = 1，y 已經變成 1，所以變成 1 + 1 = 2，後面一直延續到 x = 10 迴
圈就會停止。
```

```
> y <- 0
> for (x in c(1:10)) {
+   y <- x + y
+ }
```

```
> x <- 1
> y <- 0
> while (x <= 10) { # while 只要符合判斷式，就會一直重複執行括號內程式
碼，直到不符合為止。
+   y <- x + y
+   x <- x + 1 # 這行很重要，如果沒有這行，程式碼會一直執行不會停止，因為
判斷式是 x 小於等於 10，x 初始值是 0，如果不對 x 做些動作，x 會一直小於等
於 10，所以這邊加 1，是希望執行到 x = 11 時，迴圈就會停止。
+ }
```

```
> x <- 1
> y <- 0
repeat { # repeat 與 while 有點類似，只是判斷式的部份，可以比較自由寫在
括號內任一地方，且跳出迴圈是利用 break 方式。
```

```
> repeat {  
+   if (x > 10) break # break 是會執行跳出迴圈的動作，  
意味程式停止。  
+  
+   y <- x + y  
+   x <- x + 1  
+ }  
  
> x <- 1  
> y <- 0  
> repeat {  
+   if (x > 10) { # 此部份判斷式多了一個 x = 5 部份  
+       break  
+   } else if (x == 5) {  
+       x <- x + 1  
+       next # next 是指跳過此次的迴圈，執行下一次的迴圈，所以這邊會被跳過  
x = 5，代表 5 不會被加到，那有人有疑惑那為何上面要有一個 x <- x + 1，因為  
跳過 x 還是需要執行加 1 的動作，不然程式只會到 5 不會到 6，想要測試此狀況，  
就把 x <- x + 1 改成 print(x)，就會發現程式一直跑停不下來。  
+   }  
+  
+   y <- x + y  
+   x <- x + 1  
+ }
```

Chapter 6 - 資料整理

1. 重新編碼
2. 資料變形
3. 資料合併與分割

重新編碼

分析資料前常常需要再次整理資料，方便日後做分析，整理資料第一步往往是將資料的調整值經過一些調整，以下介紹幾種重新編碼的方法。

- 透過邏輯判斷式
- 利用 cut 函數

```
> data <- iris # 使用 R 內建的資料。

> data$Sepal.Length <- ifelse(data$Sepal.Length > 5, 1, 2) # Sepal.Length 如果大於 5 會變成 1，不會就會變成 2

> data$Species <- ifelse(data$Species %in% c(setosa), "IsSetosa", "Notsetosa") # %in% 代表有包含到的概念

> x <- c(1, 5, 12, 18, 19, 21, 25, 31)
> cut(x, c(0, 10, 20, 30, 40), c(5, 15, 25, 35)) # cut 函數是透過切割點，重新賦予資料新的數值，本範例的切割的範圍是 0 ~ 10、10 ~ 20、20 ~ 30、30 ~ 40，0 ~ 10 範圍的賦予新的值是 5。
[1] 5 5 15 15 15 25 25 35
Levels: 5 15 25 35
```


資料變形

在第三章常提到 `as` 開頭的函數，例如：`as.vector`、`as.array`、`as.matric`、`as.data.frame` 等，其實就是資料變形函數的一種，以下介紹其他常用到的資料變形函數。

- `stack` 與 `unstack` 函數
- `reshape` 函數

```
> data <- iris # 使用 R 內建的資料。

> stack_data <- stack(data) # stack 函數將各行資料排成一直行，unstack 則是還原成未 stack 之前形態。

> library("longitudinalData")
> data <- artificialJointLongData # 此種資料稱作為縱向資料(Longitudinal Data)，通常是單一物體重複測量值所產生的資料，記錄方式可以用長型資料(long format)或寬型資料(wide format)。

> data_long = reshape(data, direction="long", varying = list(c("v0", "v1", "v2", "v3", "v4", "v5", "v6", "v7", "v8", "v9", "v10"), c("w0", "w1", "w2", "w3", "w4", "w5", "w6", "w7", "w8", "w9", "w10"), c("x0", "x1", "x2", "x3", "x4", "x5", "x6", "x7", "x8", "x9", "x10")), v.names = c("v", "w", "x"), idvar = "id") # 利用 reshape 函數將資料從 wide 轉成 long

> data_wide = reshape(data_long, direction="wide", v.names = c("v", "w", "x"), idvar = "id") # 利用 reshape 函數將資料從 long 轉成 wide
```

解釋以上 `reshape` 函數一些參數的用意。

- `direction`：資料要轉成哪種類型，`long` 或 `wide`。
- `varying`：用於 `wide` 轉 `long` 時，哪些欄位要變成一個欄位的長期觀察資料，例如 `v0 ~ v1` 要變成 `v`，`w0 ~ w10` 變成 `w`，`x0 ~ x10` 變成 `x`。
- `v.names`：對於 `wide` 轉 `long` 時，當作是 `long` 資料的欄位，對於 `long` 轉

wide，哪些欄位要變成多個欄位觀察值。

- idvar：非重複性觀察資料，多半是可以分辨觀察物體的變數，，例如：id、名稱等。

資料合併與分割

資料整理最後來介紹如何合併與分割資料。

- union、cbind 與 rbind 函數
- merge 函數
- split 函數
- subset 函數

資料合併

```
> x <- c(1, 2, 3)
> y <- c(10, 20, 30)
> union(x, y) # union 如英文名稱就是取聯集。
[1] 1 2 3 10 20 30

> rbind(x, y) # 透過 row 合併。
  [,1] [,2] [,3]
x    1    2    3
y   10   20   30

> cbind(x, y) # 透過 column 合併。
  x y
[1,] 1 10
[2,] 2 20
[3,] 3 30

> x <- cbind(c("Tom", "Joe", "Vicky"), c(27, 29, 28))
> y <- cbind(c("Tom", "Joe", "Vicky"), c(178, 186, 168))
> colnames(x) <- c("name", "age")
> colnames(y) <- c("name", "tall")

> merge(x, y, by = "name") # 將 data.frame 透過一個欄位進行合併。
  name age tall
1  Joe  29  186
2  Tom  27  178
3 Vicky 28  168
```

```
> x <- cbind(c("Tom", "Joe", "Vicky", "Bob"), c(27, 29, 28, 25))
> y <- cbind(c("Tom", "Joe", "Vicky", "Bruce"), c(178, 186, 168,
170))
> colnames(x) <- c("name", "age")
> colnames(y) <- c("name", "tall")
```

> merge(x, y, by = "name", all = T) # all 是用來詢問是否顯示所有資料，像 Bob 與 Bruce 都有一欄資料沒有，所以沒下 all = T，應該不會出現 Bob 與 Bruce 資料。

	name	age	tall
1	Bob	25	<NA>
2	Joe	29	186
3	Tom	27	178
4	Vicky	28	168
5	Bruce	<NA>	170

> merge(x, y, by = "name", all.x = T) # 只顯示 x 有的資料，所以 Bruce 就不會出現。

	name	age	tall
1	Bob	25	<NA>
2	Joe	29	186
3	Tom	27	178
4	Vicky	28	168

> merge(x, y, by = "name", all.y = T) # 只顯示 y 有的資料，所以 Bob 就不會出現。

	name	age	tall
1	Joe	29	186
2	Tom	27	178
3	Vicky	28	168
4	Bruce	<NA>	170

資料分割

```
> data <- iris
> split(data, sample(rep(1:2, 75))) # rep(1:2, 75) 產生 1,2 交錯的
向量，但加了前面的 sample 則是隨機抽取，所以向量 1,2 會被打亂，split 會依
照 sample(rep(1:2, 75)) 分組，都是 1 的會在同一組，都是 2 的也會在同一
組。

> data <- iris

> subset(data, Sepal.Length > 5) # 只會出現 Sepal.Length > 5 的資料

> subset(data, Sepal.Length > 5, select = Sepal.Length) # 只會出現
Sepal.Length > 5 的資料且欄位只有 Sepal.Length，select 代表會出現的
欄位。

> subset(data, Sepal.Length > 5, select = -Sepal.Length) # select
= 負的代表不要出現的欄位。
```

Chapter 7 - 自訂函數

1. 定義函數
2. 建立 `.First` 與 `.Last` 函數

定義函數

R 可以將常重複執行的程式碼定義成函數，如以下定義

```
函數名稱 <- function(參數){  
  程式重複執行的部份  
}  
  
> add <- function(a, b){  
+   a + b  
+ }  
> add(1, 3)  
[1] 4 # R 預設最後一個運算式當作回傳的值  
  
> add_special <- function(a, b){  
+   c = a + b  
+   return(0) # 可以自訂要回傳的部份  
+ }  
> add_special(1, 3)  
[1] 0  
  
> add_default <- function(a = 1, b = 2){  
# 可以自訂函數的預設值  
+   a + b  
+ }  
> add_default() #沒給參數值  
[1] 3  
> add_default(2) # 給 a = 2，因為給定參數是依照參數順序，如果想給 b，但  
不想給定 a，可以利用以下方式  
[1] 4  
> add_default(b = 3)  
[1] 4
```

建立 .First 與 .Last 函數

R 可以利用 .First 與 .Last 函數，建立 R 環境 開始與結束時會自動執行的函數。

- 建立 .Rprofile 檔案
- 建立 .First 與 .Last 函數

```
touch ~/.Rprofile  
  
open -a Textedit ~/.Rprofile
```

.Rprofile 檔案內容

```
.First = function()  
{  
  library(ggplot2) # 開啓 RStudio 時會自動載入ggplot2package  
}  
  
.Last = function()  
{  
  
}
```

註：以上是在 Mac 環境下操作，但不同環境應該大同小異。