

CI and RMarkdown v0.73

teuton

6/24/2020

# Contents

<b>1 enviroment 設立</b>	<b>3</b>
1.1 安裝 git、R、RStudio 與相關套件 . . . . .	3
1.2 github 環境設立 . . . . .	4
1.2.1 註冊帳號與登入 . . . . .	4
1.2.2 在 RStudio 設定 git 帳戶 . . . . .	4
1.2.3 建立新的 repository . . . . .	4
1.2.4 將 Repository 複製到 RStudio . . . . .	5
1.2.5 新增與儲存 R Markdown 文件 . . . . .	5
1.2.6 將 RStudio 變更更新至 github . . . . .	5
1.3 自動編譯檔案，Travis ci 環境建立 . . . . .	5
1.3.1 上傳你的 R Markdown 文件 . . . . .	5
1.3.2 創立.travis.yml 檔案 . . . . .	6
1.3.3 創立 DESCRIPTION 檔案 . . . . .	6
1.3.4 創立 Makefile 檔案 . . . . .	6
1.3.5 建立 GITHUB_TOKEN . . . . .	7
1.3.6 登入 travis ci . . . . .	7
1.3.7 將 GITHUB_TOKEN 宣告成環境變數 . . . . .	7
1.3.8 查看自動編譯狀態 . . . . .	7
1.3.9 查看輸出檔案 . . . . .	7
1.3.10 CI 帶有中文字的 pdf 文件做法 . . . . .	7
<b>2 R Markdown 文件編輯</b>	<b>9</b>
2.1 R Markdown 優點 . . . . .	9
2.2 R Markdown 格式轉換原理 . . . . .	9
2.3 R markdown 常用文法解說 . . . . .	9
2.3.1 基本文法說明 . . . . .	9
2.3.2 內嵌程式碼相關說明 . . . . .	17
2.4 pdf 文件加入中文注意事項 . . . . .	21
2.5 R Markdown Workflow . . . . .	21
2.6 線性模型解說以 Boston 資料集為例 . . . . .	21
2.6.1 讀取資料 . . . . .	21
2.6.2 空值處理 . . . . .	22
2.6.3 確認資料集概況 . . . . .	22
2.6.4 建立模型 . . . . .	27
<b>3 R code style 簡介</b>	<b>45</b>
3.1 檔名注意事項 . . . . .	45
3.2 變數宣告注意事項 . . . . .	45
3.3 插入空白注意事項 . . . . .	46
3.4 其他注意事項 . . . . .	47
3.5 其他：Google 的 Code Style . . . . .	47

<b>4 使用 R Shiny 建立互動式文件</b>	<b>49</b>
4.1 建立 R Shiny 檔案 . . . . .	49
4.2 文法大綱 . . . . .	49
4.3 UI 撰寫 . . . . .	49
4.3.1 版面配置選項 . . . . .	49
4.3.2 定義使用者輸入物件 . . . . .	50
4.3.3 宣告輸出版面格式 . . . . .	50
4.4 server 撰寫 . . . . .	50
4.4.1 前置作業處理 . . . . .	50
4.4.2 輸出處理 . . . . .	50
4.5 以 Boston 資料集為例 . . . . .	50

# Chapter 1

## enviroment 設立

本章節解釋如何設定需要環境

### 1.1 安裝 git、R、RStudio 與相關套件

安裝 git，[按這裡前往安裝連結](#)，按照流程安裝即可

到 CRAN 官網上下載 R 語言，並且依照指示安裝

下載 RStudio Desktop 免費版本，[按這裡前往安裝連結](#)，並且依照指示安裝

打開 RStudio，找到 Packages 標籤，點選它，然後點選下一行的 install，在彈出視窗的第二行輸入 rmarkdown，點選 install，如圖1.1所示，這樣就能安裝 rmarkdown 套件，用同樣的方法安裝 knitr 以及 tinytex 套件

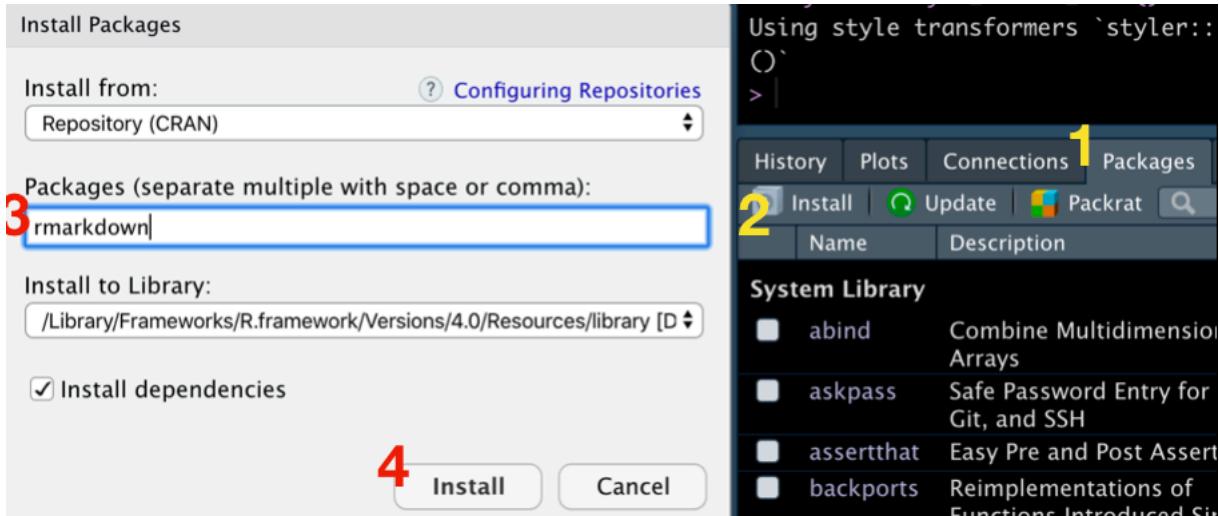


Figure 1.1: 安裝 R Markdown 以及相關套件

如果你沒裝 latex，就在 Rstudio 上安裝 tinytex：找到 console 標籤，點擊它，在裡面輸入 `tinytex::install_tinytex()`，如圖1.2所示，然後按下 Enter

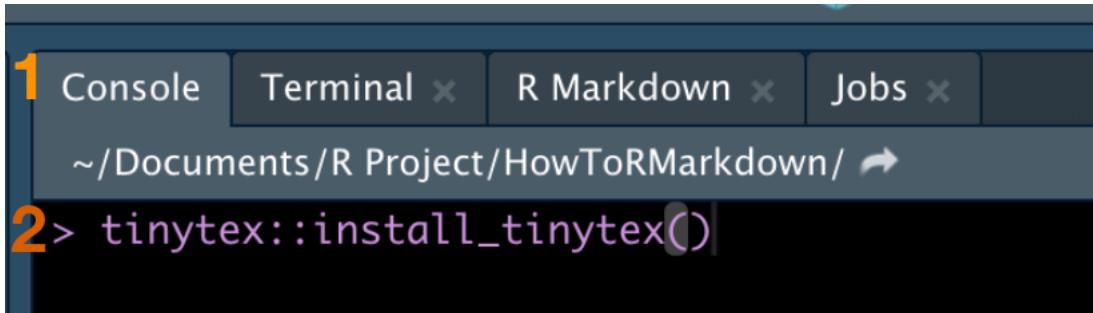


Figure 1.2: 於 RStudio 上安裝 tinytex

## 1.2 github 環境設立

### 1.2.1 註冊帳號與登入

[點此前往 github 網站](#)，並建立一個使用者帳號，需要名稱、信箱和密碼，使用信箱驗證後，帳號就可以用了

### 1.2.2 在 RStudio 設定 git 帳戶

在 RStudio 找到 Terminal 標籤，根據你註冊的帳號與郵箱輸入以下指令：

```
git config --global user.name '你的帳號名稱'
git config --global user.email '你的註冊信箱 @example.com'
git config --global --list
```

一次一行，第三行可以確認你輸入的資訊，如圖1.3所示

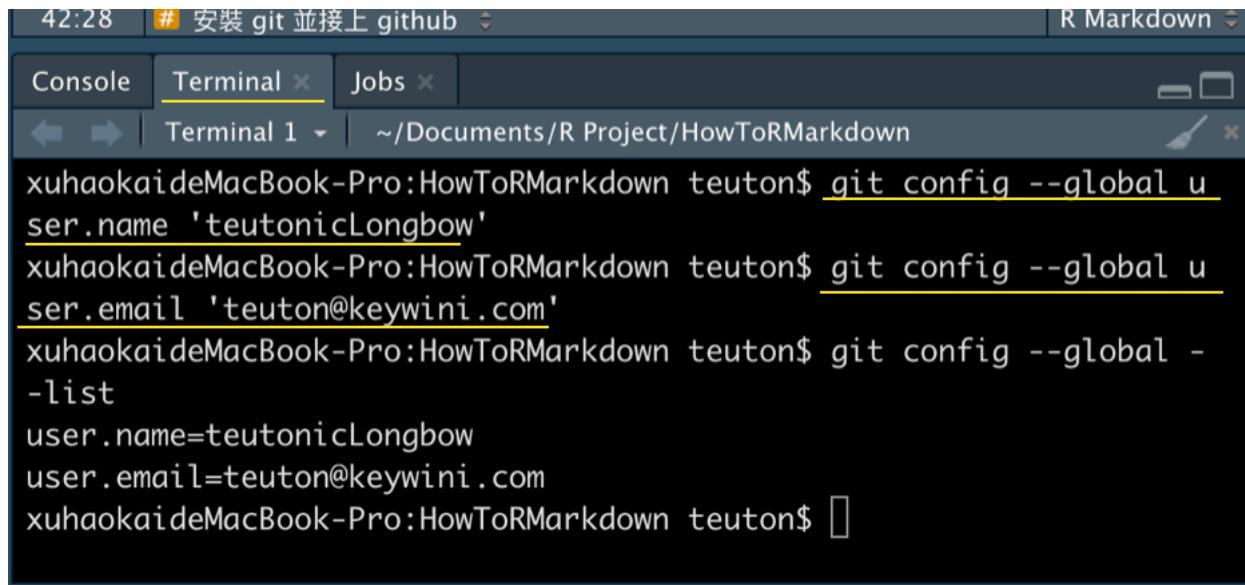


Figure 1.3: 設定 git 使用帳戶

### 1.2.3 建立新的 repository

在 github 網站登入後，點選 Repositories 附近的綠色 New，在 Repository Name 條目輸入計畫名稱，點選 Initialize this repository with a README，然後建立 repository

#### 1.2.4 將 Repository 複製到 RStudio

在 github 你的 Repository 頁面點擊綠色 Clone or download，複製其下面的網址（點一下網址右側的圖示即可）到 RStudio，點選最上列 File -> New Project... -> Version Control -> Git，在第一行貼上網址（第二行會自動填上），左下 Open in new session 打勾，然後 Create project，如果成功，在 Files 標籤內可以找到剛才新增的 README.md 檔

#### 1.2.5 新增與儲存 R Markdown 文件

在 RStudio 最上面一行點選 file -> New File -> R Markdown...，然後新增標題與作者，選擇想要輸出的文件格式後，點 OK，就會建立預設的文件，點擊上方 Knit 按鈕，就能產生編譯文件並且進行預覽

點選最上面 Files -> Save (或是使用 ctrl/cmd + S 快捷鍵) 儲存目前文件

#### 1.2.6 將 RStudio 變更更新至 github

在 RStudio 找到 Git 標籤，標籤內會顯示已修改的檔案，找到你剛才儲存的.Rmd 檔，將其左側的 Stage 欄位打勾，點選 Commit，在右上方寫一些有關這次更新檔案的註解，如圖1.4所示，然後點選 Commit，一段時間後重新整理 github，可以看到檔案已上傳

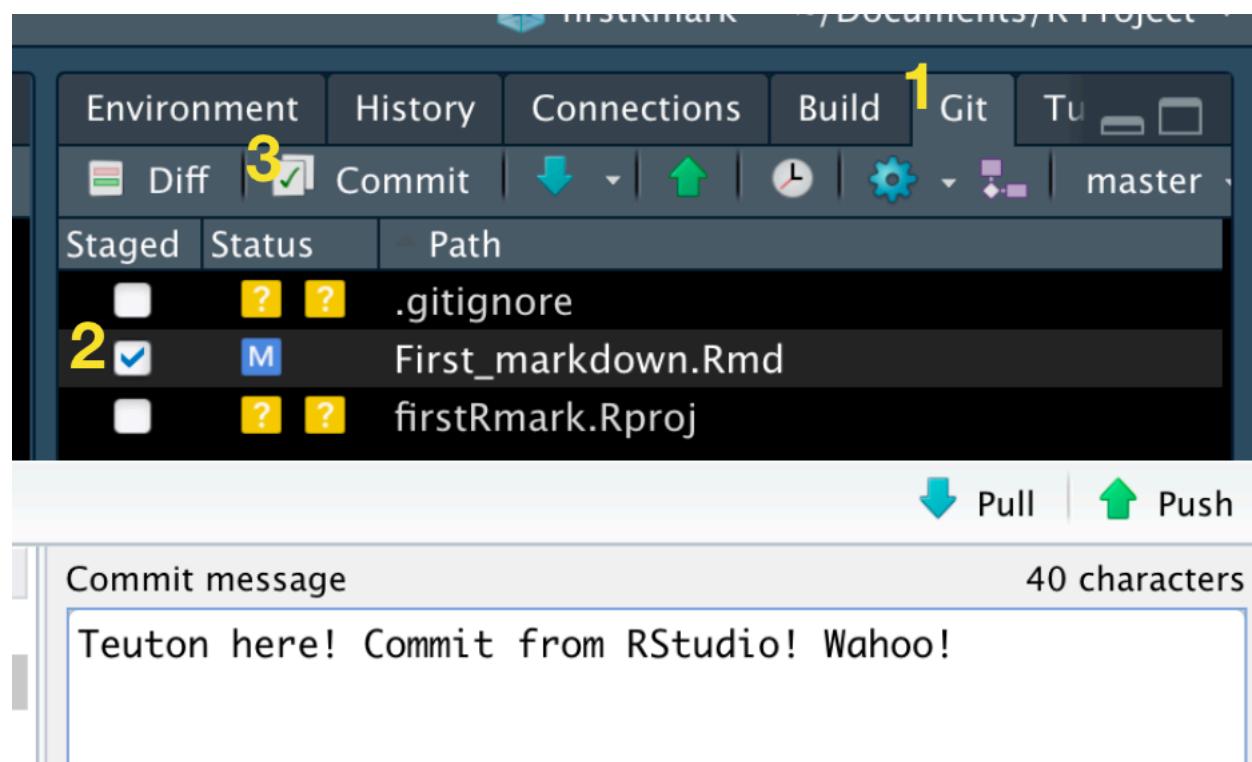


Figure 1.4: 將 RStudio 檔案更新至 github

### 1.3 自動編譯檔案，Travis ci 環境建立

#### 1.3.1 上傳你的 R Markdown 文件

點擊這裡查看從 RStudio 上傳的方法，你也可以在 github 點選你建立的 repository，點選 Upload files，選擇你要上傳的檔案，點選下方 Commit changes

### 1.3.2 創立.travis.yml 檔案

在 github 點選你建立的 repository，點選 Create new file，取名為.travis.yml，檔案內容輸入：

```
language: r
before_install:
  - sudo tlmgr install framed
script: make bacon
cache: packages
r_packages:
  - rmarkdown
  - knitr
deploy:
  provider: pages
  skip_cleanup: true
  github_token: $GITHUB_TOKEN # 在 github 設置，後面介紹
  keep_history: true
  on:
    branch: master
os:
  - osx
```

- language 宣告使用的程式語言為 r
- before\_install 告知虛擬機作業前需要額外安裝的程式，此處的安裝套件是為了處理 pdf 輸出而安裝
- script 告知虛擬機執行動作，此處呼叫 Makefile 執行其定義的 bacon 指令（編譯文件），後面會介紹
- cache 宣告虛擬機快取記憶體儲存內容，此處儲存 package，即自動編譯期間儲存套件資料，「似乎」可加快編譯時間
- r\_packages 宣告自動編譯需要安裝的 r 套件，如果未來的文件使用其他套件就要加入
- deploy 宣告將輸出送回 github 的方式，此處將編譯檔案自動更新至 github
- os 宣告虛擬機使用系統，此處為 macos，你可能會需要根據目前作業系統來修改此條目

以上輸入完成後，往下拉按 Commit new file

### 1.3.3 創立 DESCRIPTION 檔案

在 github 點選你建立的 repository，點選 Create new file，命名為 DESCRIPTION，內容是：

```
Imports:
  rmarkdown,
  knitr
```

完成後按下 Commit new file

### 1.3.4 創立 Makefile 檔案

在 github 點選你建立的 repository，點選 Create new file，命名為 Makefile，內容是：

```
RMDFILES = $(wildcard *.Rmd)

bacon: $(RMDFILES)
  $(foreach rmdfile,$(RMDFILES),Rscript -e 'rmarkdown::render("$(rmdfile)")';)

clean:
  rm -rf *.html *.md *.docx figure/ cache/
```

注意：這裡的兩行縮排必須使用 Tab，不能使用空白鍵！編輯此檔案時，確認 github 右側的縮排設定是 Tab。完成後按下 Commit new file

### 1.3.5 建立 GITHUB\_TOKEN

點選 github 網站右上方圖片選單，下拉按下 settings，左側點選 Developer settings，左側點選 Personal access tokens，點選右側 Generate new token，在 Note 欄位寫點東西，將下方 repo 選項打勾，如圖1.5所示

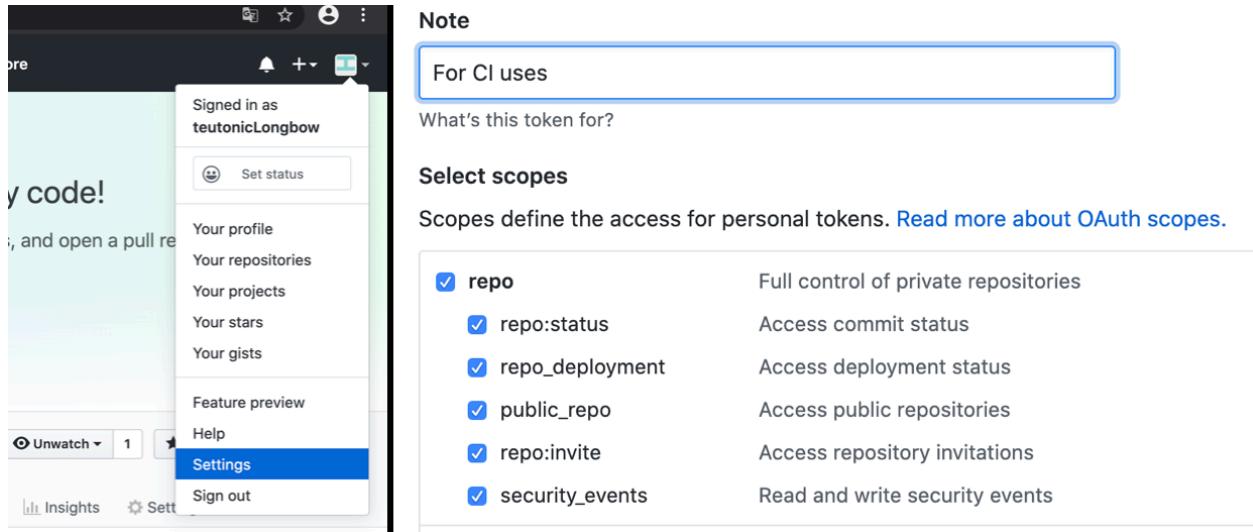


Figure 1.5: 建立 GITHUB\_TOKEN 第一部分

好，喘口氣，因為下一步搞砸了就要重來

點選下面的 Generate token，在新出現的頁面會出現一串很長的亂碼，這串亂碼好比密碼，點選其右側的小圖示來複製這串字，你可以暫時貼在剪貼簿上備用，網頁一旦重新載入亂碼就會消失！

### 1.3.6 登入 travis ci

[點擊這裡前往 Travis](#)，使用你的 github 帳號登入，並允許這個網站連接 github 的權利

### 1.3.7 將 GITHUB\_TOKEN 宣告成環境變數

到 travis 點選右上方的圖示，點選 Settings，點選綠色的 Activate，點選 Approve & Install，過一段時間之後重新整理頁面，找到你剛新增的 repository，點選右側的 settings，將頁面往下拉找到 Enviroment Variables 條目，在 VALUE 貼上你在 github 產生的亂碼，NAME 寫 GITHUB\_TOKEN，完成後按下右邊的 Add，成功了之後，如果你剛才有把亂碼貼在剪貼簿上，記得將其刪除

### 1.3.8 查看自動編譯狀態

回到 travis，點選最上方 Dashboard，點選中間 My builds 即可查看，以後你只要修改 repository 的任何一個檔案，這裡就會新增一個工作條目來自動編譯所有的.Rmd 文件，點擊 #n build 來確認編譯過程或是錯誤資訊

### 1.3.9 查看輸出檔案

編譯成功後，回到 github 網頁，點選你的 repository，點選左側的 Branch 標籤，下拉點選 gh-pages，在這裡就可以找到你上傳的檔案以及輸出檔，如圖1.6所示，點擊查看

### 1.3.10 CI 帶有中文字的 pdf 文件做法

主要問題是，虛擬機上的 latex 編譯器不知道如何處理中文字

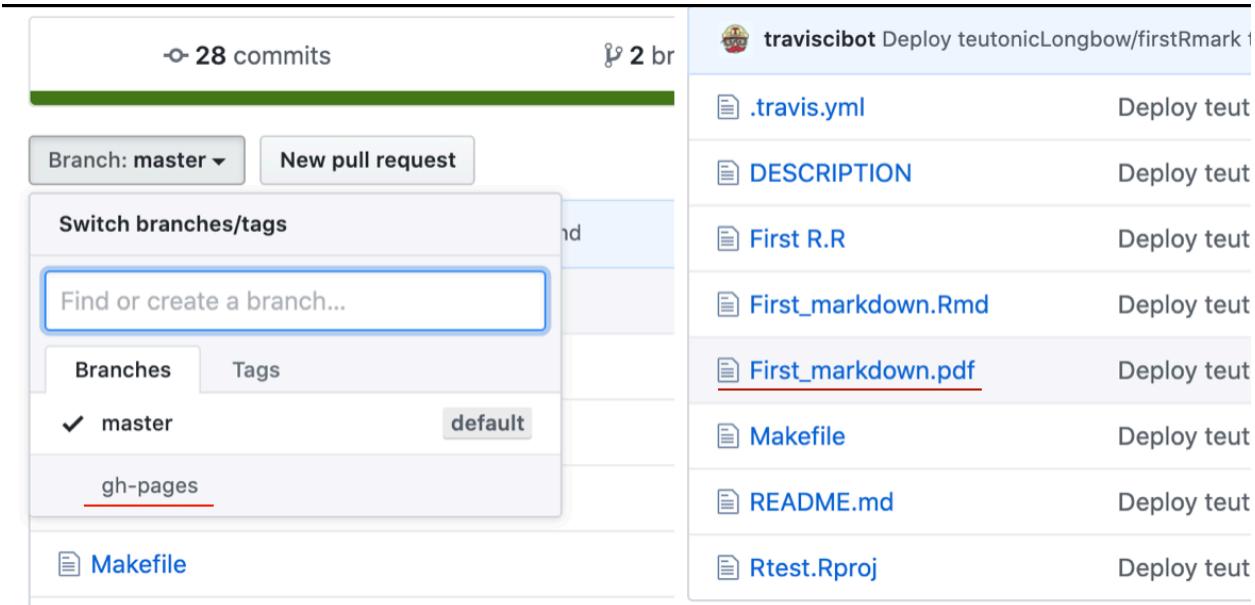


Figure 1.6: 查看輸出檔案

### 1.3.10.1 下載並且上傳中文字體

先決定你文件的中文字要使用什麼字體，在網路上搜尋其.ttf 檔，例如[點擊這裡下載微軟的標楷體](#)，下載後將其上傳至 github

### 1.3.10.2 修改你的.Rmd 文件

在 github 點選你的.Rmd 文件，在網頁右側找到鉛筆圖示，點擊它就能編輯文件，在文件內容最上方找到 output 行，將其換成：

```
header-includes:
  - \usepackage{xeCJK}
  - \setCJKmainfont{kaiu.ttf}
  - \setCJKmonofont{kaiu.ttf}
output:
  pdf_document:
    latex_engine: xelatex
```

把 kaiu.ttf 換成你上傳的中文字體檔，完成後下拉按下 Commit changes，這樣就可以在 R Markdown 使用中文了

# Chapter 2

## R Markdown 文件編輯

本章節解釋如何新增 R Markdown 文件，以及基本編輯說明

### 2.1 R Markdown 優點

融合程式碼與文件撰寫為一體，呈現程式碼並呈現編譯結果，可以輕鬆美化文件，支援多個程式語言如 Python, R, SQL, javascript, CSS, C 等

### 2.2 R Markdown 格式轉換原理

如圖2.1所示，編譯 R Markdown 文件，先經由 knitr 套件將內嵌程式碼編譯，以此建立一個包含程式碼與輸出的 Markdown 文件，接著用 pandoc 將其轉成各種使用者選擇的輸出格式，由於 Pandoc 無法直接將 Markdown 轉成 Pdf，所以會先將其轉成 tex 檔後再經由 LaTeX (或是 RStudio 上的 tinytex) 轉成 Pdf



Figure 2.1: R Markdown 編譯流程

### 2.3 R markdown 常用文法解說

新增與儲存 R Markdown 文件，請見[這裡](#)

#### 2.3.1 基本文法說明

##### 2.3.1.1 新增一般段落文字

你在文件輸入的任何字，只要不夾雜特殊符號，通常會直接呈現

如果你要換行，按下 Enter 還不夠，你要在句子結尾留下兩個空白，再按下 Enter 才能換行

如果你要新增一個段落，在段落下一行新增空白行，然後再下一行新增段落文字。與換行相比，換行會緊貼在下方，而且會遵照上一行的縮排格式；而新段落會於零縮排的位置開始，而且與上一段落會有些微行距

綜合以上觀念，試著編譯這段程式碼，結果如圖2.2所示，留意問號後面有兩個空白鍵，如果要加入中文，請參考[這裡](#)

```
---
```

```
title: "First R Markdown"
output: pdf_document
---

Hello R Markdown!

Are you
kidding me?
Nothing can pass my bow!
```

## First R Markdown

Hello R Markdown!

Are you kidding me?  
Nothing can pass my bow!

Figure 2.2: 你的第一個 R Markdown 文件

### 2.3.1.2 R Markdown 的各種字體

如果要把文字改成斜體字，在其前後用 \* 符號夾住，注意於 html 文件可以顯示斜體中文，pdf 文件預設無法顯示斜體中文

如果要把文字改成粗體字，在其前後用 \*\* 符號夾住，注意於 html 文件可以顯示粗體中文，pdf 文件預設無法顯示粗體中文

要在文章加入上標字，就在目標位置用 ^ 夾住上標文字

要在文章加入下標字，就在目標位置用 ~ 夾住下標文字

如果要讓一段文字被橫線穿過，就在目標位置用 ~~ 夾住文字

如果要在文章使用特殊符號，建議在其前方加上 \，以避免無法預期的錯誤，以下為一些例子：

- “\\*” -> \*
- “\\_” -> \_
- “\\” -> \

綜合以上觀念，試著編譯這段程式碼，結果如圖2.3所示

```
---
```

```
title: "R Markdown fonts"
output: pdf_document
---

this is *it* and **bf** word
```

```
Do^a^ barrel~roll~!  
\* Well, ~~notreally~~ .
```

## R Markdown fonts

this is *it* and **bf** word

Do<sup>a</sup> barrel<sub>roll</sub>!

\* Well, ~~notreally~~ .

Figure 2.3: R Markdown 文件可用字體

### 2.3.1.3 新增歷史名言區塊

引用歷史名言，空出五行，第一行和第五行留白，其餘三行的開頭輸入 >，接著於第二行輸入歷史名言，於第四行加入人名

試著編譯以下程式碼，結果如圖2.4所示

```
---  
title: "Famous words"  
output: pdf_document  
---  
In a word of famous mathmatician:  
  
> Eureka!  
>  
> --- Archimedes  
  
Such an inspiring spirit!
```

## Famous words

In a word of famous mathmatician:

Eureka!  
— Archimedes

Such an inspiring spirit!

Figure 2.4: 引用歷史名言

#### 2.3.1.4 新增章節與各類子標題

新增章與節（大標題與副標題等），上一行要留白，這行以一至六個 # 起頭，然後空一格，再輸入標題名稱，這一至六個 # 分別對應到 h1 至 h6 標題

試著編譯以下程式碼，結果如圖2.5所示，留意此處加了一句設定，使預設為 article 格式的 pdf 轉換成 report 格式，這是一般正式文件常用的格式，讀者可以嘗試將這行設定移除並再編譯一次，觀察有何不同

```
---
```

```
title: "Chapter and section"
output: pdf_document
documentclass: report
--  

# first chapter
About DGR  

  
## Checkpoint city  

  
### It's a beautiful city...  

  
### ...that checkpoint city!  

  
## Second Checkpoint  

It's Christmas!  

  
# Chapter number 2  

Do someting else...
```

# first chapter

About DGR

## Checkpoint city

It's a beautiful city...

...that checkpoint city!

## Second Checkpoint

It's Christmas!

Figure 2.5: 新增章與節標題

#### 2.3.1.5 新增目錄

把最上面 output 的選項換行然後進行縮排（例如 pdf\_document），在後面加上一個英文冒號後，再下一行縮排兩次並加入 toc: true 就可以加上目錄，目錄的項目會自動加上內文超連結。

你可以加上 `toc_depth` 選項來決定目錄顯示的小節最多有幾層 (pdf 此選項的預設值為 2)，你也可以加入 `number_sections: true` 來為你的各種標題加上編號。

試著編譯下面程式碼，結果如圖2.6所示

```
--  
title: "Add table of contents"  
output:  
  pdf_document:  
    toc: true  
    toc_depth: 3  
    number_sections: true  
documentclass: report  
--  
# first chapter  
About DGR  
  
## Checkpoint city  
  
### It's a beautiful city...  
  
### ...that checkpoint city!  
  
## Second Checkpoint  
It's Christmas!  
  
# Chapter number 2  
Do someting else...
```

# Contents

<b>1</b>	<b>first chapter</b>	<b>2</b>
1.1	Checkpoint city . . . . .	2
1.1.1	It's a beautiful city... . . . .	2
1.1.2	...that checkpoint city! . . . .	2
1.2	Second Checkpoint . . . . .	2
<b>2</b>	<b>Chapter number 2</b>	<b>3</b>

Figure 2.6: 新增目錄

### 2.3.1.6 新增條列項目

新增條列項目，上一行留白，這行 \* 起頭，然後空一格，再輸入第一個條列項目內容。

如果要新增下一個條列項目，就緊貼在下一行用 \* 開頭，空一格，然後輸入條列項目內容，在條列項目內換行會與目前條目對齊（而不是從零縮排位置開始）。

如果要新增條列子項目，下一行使用 Tab 或是兩個空白鍵縮排後，用同樣方法輸入條列項目內容，新增子子項目就在子項目下一行用兩個 Tab 或是四個空白鍵縮排後新增，以此類推，你可以使用-或 + 來取代 \*。

若要結束條列項目，就在最下方新增空白行，以新增段落的方式處理

試著編譯下面程式碼，結果如圖2.7所示，留意只有 steak 後面有兩個空白，觀察 garlic 換行的對齊位置以及 pineapple 的前置符號

```
title: "Unordered list items"
output: pdf_document
documentclass: report
---

What I have in mind:

* apple
* mango
  - steak
    garlic added
  - hamburger
    with onion
    + yogart
  - pineapple

Pretty good!
```

What I have in mind:

- apple
- mango
  - steak
    - garlic added
  - hamburger with onion
    - \* yogart
- pineapple

Pretty good!

Figure 2.7: 新增條列項目

### 2.3.1.7 新增表格

新增表格，上面要空一行，表格的每一行開頭與結尾都是 |，每行的各列內容以 | 隔開，| 位於 Enter 鍵上方。

第一行輸入各列標題

第二行決定對齊方法，----代表預設對齊方法；:---代表靠左對齊；---: 代表靠右對齊；:--: 代表置中對齊，你可以改變中間-的數量來讓這個原始碼的表格部分更好讀。

第三行及以後輸入表格內容

如果要結束表格，下一行不要以 | 起頭就好

試著編譯下面程式碼，結果如圖2.8所示

```
--  
title: "Add table"  
output: pdf_document  
documentclass: report  
--  
Rarities:  
  
|normal|rare| epic |legendary!  
|---:|---|---|:-----:  
|white |blue|purple| yellow|  
| 3   | 3  | 2    | 2      |  
Now that's something.
```

Rarities:

normal	rare	epic	legendary!
white	blue	purple	yellow
3	3	2	2

Now that's something.

Figure 2.8: 新增表格

#### 2.3.1.8 新增超連結、圖片、長橫線與頁尾註解

在文章內加入超連結，新增一對中括號並緊貼其後新增一對小括號，於中括號內輸入套用超連結的文字，於小括號內輸入超連結網址

新增圖片，空出三行，第一行和第三行留白；第二行輸入英文的驚嘆號，緊貼著一對中括號再緊貼著一對小括號，中括號內寫圖片標題，小括號加入圖片的相對路徑。如果你的圖片和你的.Rmd 檔在同一個資料夾內，那只需要寫檔名就可以了，如果 RStudio 成功找到圖片，就會在文件編輯區內自動預覽

新增長橫線，上一行留白，這行輸入 \*\*%，下一行可以直接輸入新文字

新增頁尾註解，在想要加入註解的位置插入 ^，並且緊貼一對中括號，於中括號內新增註解內容，註解內容與本文會自動用長橫線隔開。頁尾註解出現的位置視輸出格式而定：html 註解內容在最後方；pdf 則在註解該頁的頁尾

綜合以上觀念，本例將 Tapa.png 存放在與本文件同目錄下的 images 資料夾內，結果如圖2.9所示，注意本例在註解內新增超連結

```
--  
title: "Tapa puzzle"  
output: pdf_document  
documentclass: report  
--
```

Below is a [classic Tapa] (<https://www.gmpuzzles.com/blog/tapa-rules-and-info/>) puzzle<sup>1</sup>[Puzzle page [here] (<https://www.gmpuzzles.com/blog/2019/08/tapa-by-murat-can-tonta-2/>)]:

\*\*\*

! [This is a classic Tapa puzzle] (images/Tapa.png)

Can you solve it?

Below is a classic Tapa puzzle<sup>1</sup>:

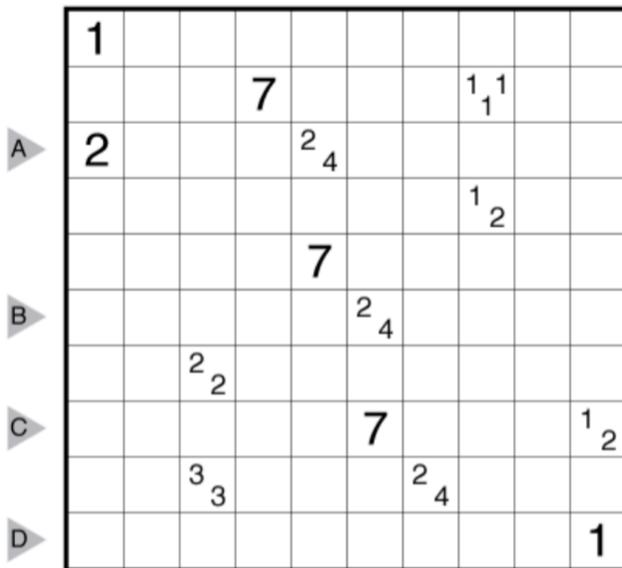


Figure 1: This is a classic Tapa puzzle

Can you solve it?

---

<sup>1</sup>Puzzle page here

Figure 2.9: 超連結、圖片、長橫線與頁尾註解

## 2.3.2 內嵌程式碼相關說明

### 2.3.2.1 短程式碼相關

在段落內插入不會進行編譯的短程式碼，將程式碼頭尾用‘包夾’，‘位在 Esc 鍵附近，通常與 ~ 同位置  
插入會執行的的 r 短程式碼類似，差別在第一個‘要立刻插入 r 和一個空格，目前此功能只支援 R 語言  
你可以預先宣告的變數值，在文件最上方由兩個三橫線包夾的區域內加入 params 條目，此條目內的每一行都要縮排，形式為變數名稱：變數值，取用方式是‘r ... params\$ 變數名稱 ...’

試著編譯下面程式碼，結果如圖2.10所示

```
---
```

```
title: "Short codes"
output: pdf_document
documentclass: report
params:
  n: 7
  d: 17
---
`message("Hello R!")` is the most common
first code in R.

My current R version is `r getRversion()`

I declared two variables:
n is `r params$n` and d is `r params$d`.
```

message("Hello R!") is the most common first code in R.

My current R version is 4.0.0

I declared two variables: n is 7 and d is 17.

Figure 2.10: 插入短程式碼

### 2.3.2.2 html 與 latex 文法作用

R Markdown 的程式註解形式與 html 的程式註解形式相同，即包夾在 <!--和--> 之間的文字，R Markdown 編輯時都會直接忽視，可以用來協助人了解程式碼的作用

在 R MArkdown 使用 LaTeX 語法，會影響對 pdf 的輸出，但是不會影響 html 的輸出

在 R Markdown 使用 html 標籤語法，標籤內文字在 html 會被標籤影響，但在 pdf 只是普通文字

綜合以上觀念，試著編譯這段程式碼，結果如圖2.11所示，讀者可以點選 Knit 鍵旁邊的選單，改成其他輸出格式試試看

```
---
```

```
title: "html and latex"
```

```

output: pdf_document
documentclass: report
---
Mr.Chu?

<!--Equivalent to **bf-->
\textbf{What the what?}

I mean... Chu Ko Nu!

<!--Equivalent to *it-->
<em>What a dad joke!</em>

```

Mr.Chu?

**What the what?**

I mean... Chu Ko Nu!

What a dad joke!

Figure 2.11: html 和 latex 程式碼作用

### 2.3.2.3 長程式碼相關

如果要新增較長的程式碼（無論編譯與否），點選編輯版面上方的 insert，然後選擇你的程式碼語言，在“‘包夾的區域內寫你的程式

```

```{r}
message("hello R Markdown!")
```

```

編譯結果為

```
message("hello R Markdown!")
```

```
## hello R Markdown!
```

注意：有些支援的程式語言並沒有顯示在 insert 選單內，如 javascript, CSS, C 等，但還是可以藉由更改設定來呈現

#### 2.3.2.3.1 設定前置符號

comment 設定編譯結果前置符號，預設值是 ##

```

```{r comment = 'CC'}
for (i in 1:3) {
  print('*')
}
```

```

編譯結果為

```
for (i in 1:3) {  
  print('*')  
}
```

```
CC [1] "*"  
CC [1] "*"  
CC [1] "*"
```

### 2.3.2.3.2 設定是否顯示原始碼

echo 決定是否顯示原始碼，預設值是 TRUE

```
```{r echo = FALSE}  
for (i in 1:3) {  
  print('*')  
}  
```
```

編譯結果為

```
## [1] "*"  
## [1] "*"  
## [1] "*"
```

### 2.3.2.3.3 設定長程式碼錯誤時的處理方法

error 在此長程式碼發生錯誤時，會顯示錯誤訊息並繼續編譯文件 (TRUE)，或是在你的介面顯示錯誤訊息並中斷編譯 (FALSE)，預設值為 FALSE

```
```{r error = TRUE}  
for (i in 1:3) {  
  print('*')\ # 這裡有錯誤  
}  
```
```

編譯結果為

```
for (i in 1:3) {  
  print('*')\ # 這裡有錯誤  
}
```

```
## Error: <text>:2:13: unexpected input  
## 1: for (i in 1:3) {  
## 2:   print('*')\  
##
```

### 2.3.2.3.4 設定是否編譯長程式碼

eval 決定是否編譯此程式碼，預設值為 TRUE

```
```{r eval = FALSE}  
# 這是錯誤程式，因為沒有進行編譯  
# 所以沒有錯誤訊息，也沒有編譯結果  
for (i in 1:3) {  
  print('*')\ # 這裡有錯誤  
}  
```
```

編譯結果為

```
# 這是錯誤程式，因為沒有進行編譯  
# 所以沒有錯誤訊息，也沒有編譯結果  
for (i in 1:3) {  
    print('*')\ # 這裡有錯誤  
}
```

### 2.3.2.3.5 設定編譯程式碼後，是否顯示原始碼與結果

include 在編譯此程式碼後，決定是否呈現原始碼與編譯結果，預設值為 TRUE

```
```{r include=FALSE}  
# 這些會執行，但是會隱藏  
a <- 3  
print("OK")  
```\n\n```{r}  
for (i in 1:a) { # a 值有定義  
# 定義後也有訊息輸出  
# 只是藏起來了而已  
    print('*')  
}  
```
```

編譯結果為

```
for (i in 1:a) { # a 值有定義  
# 定義後也有訊息輸出  
# 只是藏起來了而已  
    print('*')  
}  
  
## [1] "*"  
## [1] "*"  
## [1] "*"
```

### 2.3.2.3.6 多重設定疊加方式

疊加的選項用英文逗點隔開

```
```{r comment = '*', error = TRUE}  
for (i in 1:3) {  
    print('*')\br/>}  
```
```

編譯結果為

```
for (i in 1:3) {  
    print('*')\br/>}  
  
* Error: <text>:2:13: unexpected input  
* 1: for (i in 1:3) {  
* 2:     print('*')\br/>*
```

## 2.4 pdf 文件加入中文注意事項

如果輸出格式是 html\_document，最上方由兩條三橫線所包夾的區域，不用修改也能在文件使用中文；如果是 pdf，這個格式使用 LaTeX，而且預設是無法處理中文的

如果要解決此問題，就要修改文件內容最上面的 output 條目，有兩種做法，第一種簡單快速：

```
output:  
  pdf_document:  
    includes:  
      header-includes: \usepackage{xeCJK}  
    latex_engine: xelatex  
  CJKmainfont: 標楷體
```

xeCJK 是 xelatex 編譯中文使用的套件，如果系統編譯顯示不存在，那就要先去 LaTeX 安裝此套件，如果你使用的是 RStudio 上的 tinytex，它會自動幫你安裝尚未安裝的 latex 套件，最後一列是設定字體，使用電腦上已安裝的字體，請至字體庫找到想要的字體名稱，以改寫最後一行

第二種做法適合 CI:

```
header-includes:  
  - \usepackage{xeCJK}  
  - \setCJKmainfont{kaiu.ttf}  
  - \setCJKmonofont{kaiu.ttf}  
output:  
  pdf_document:  
    latex_engine: xelatex
```

基本上採用這裡的做法，下載的中文字體檔和.Rmd 檔要放在同一個資料夾內，把兩行 kaiu.ttf 換成你的中文字體檔就可以了

## 2.5 R Markdown Workflow

- 確保文件具有好標題、相關檔名、以及簡短描述分析目標的第一段文字
- 在最上方由兩條三橫線包夾的區域內新增開始研究的日期，格式為 YYYY-MM-DD :

```
date: 2020-5-20
```

- 就算你花很長時間研究的資料碰上了死路，不要刪掉，簡短描述失敗原因並留在文件裡，這有助於未來分析不會再次遭遇一樣的失敗
- 一般而言，資料最好在進入 R 以前就先處理，如果你需要紀錄一小段資料，使用 tibble::tribble()
- 如果你發現了資料檔錯誤，不要直接修改，而是使用程式修正資料，並說明理由
- 在你結束以前，記得要編譯文件，這樣能讓你腦子裡還有程式碼時解決任何問題
- 如果你希望程式碼長期下來都能運作（例如一年），你需要追蹤程式碼使用的套件版本
- 如果你會建立很多很多資料分析文件，建議使用獨立計畫儲存，並使用良好的命名方法，方便組織以利於在未來取用

## 2.6 線性模型解說以 Boston 資料集為例

### 2.6.1 讀取資料

```
library(MASS) # Boston 資料集來源  
library(car) # VIF 函數來源  
  
## Loading required package: carData
```

```
#mydata <- read.table(資料檔, header = TRUE, na.strings = "?")
#mydata <- read.csv(資料檔, header = TRUE, na.strings = "?")
```

最後兩行為一般現實情況使用，header 詢問第一行是否為參數名稱，na.strings 確認要把資料集內當成 NA 的符號

## 2.6.2 空值處理

```
my_Boston <- na.omit(Boston) # 移除含有空值的資料
```

## 2.6.3 確認資料集概況

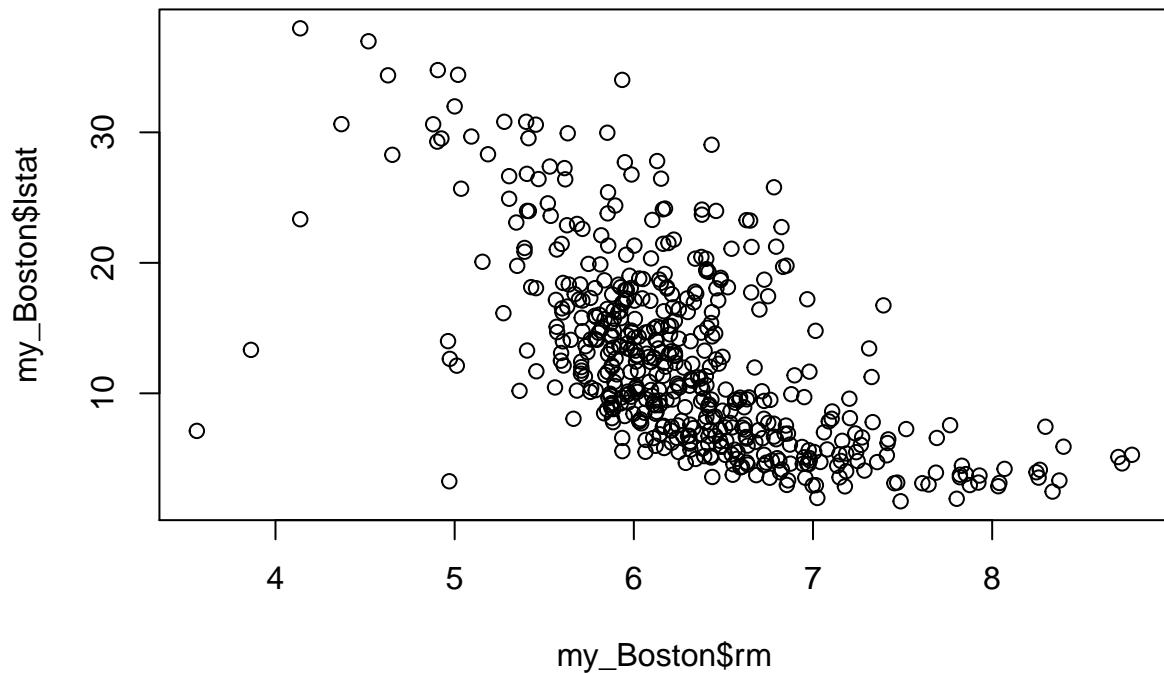
```
summary(my_Boston) # 提供資料集各變數統計值
```

```
##      crim            zn            indus            chas
##  Min. : 0.00632  Min. : 0.00  Min. : 0.46  Min. :0.00000
##  1st Qu.: 0.08205 1st Qu.: 0.00  1st Qu.: 5.19  1st Qu.:0.00000
##  Median : 0.25651 Median : 0.00  Median : 9.69  Median :0.00000
##  Mean   : 3.61352 Mean   : 11.36  Mean   :11.14  Mean   :0.06917
##  3rd Qu.: 3.67708 3rd Qu.: 12.50  3rd Qu.:18.10  3rd Qu.:0.00000
##  Max.   :88.97620 Max.   :100.00  Max.   :27.74  Max.   :1.00000
##      nox             rm            age            dis
##  Min. :0.3850  Min. :3.561  Min. : 2.90  Min. : 1.130
##  1st Qu.:0.4490 1st Qu.:5.886  1st Qu.: 45.02  1st Qu.: 2.100
##  Median :0.5380 Median :6.208  Median : 77.50  Median : 3.207
##  Mean   :0.5547 Mean   :6.285  Mean   : 68.57  Mean   : 3.795
##  3rd Qu.:0.6240 3rd Qu.:6.623  3rd Qu.: 94.08  3rd Qu.: 5.188
##  Max.   :0.8710 Max.   :8.780  Max.   :100.00  Max.   :12.127
##      rad             tax            ptratio          black
##  Min. : 1.000  Min. :187.0  Min. :12.60  Min. : 0.32
##  1st Qu.: 4.000 1st Qu.:279.0  1st Qu.:17.40  1st Qu.:375.38
##  Median : 5.000 Median :330.0  Median :19.05  Median :391.44
##  Mean   : 9.549 Mean   :408.2  Mean   :18.46  Mean   :356.67
##  3rd Qu.:24.000 3rd Qu.:666.0  3rd Qu.:20.20  3rd Qu.:396.23
##  Max.   :24.000 Max.   :711.0  Max.   :22.00  Max.   :396.90
##      lstat            medv
##  Min. : 1.73  Min. : 5.00
##  1st Qu.: 6.95 1st Qu.:17.02
##  Median :11.36 Median :21.20
##  Mean   :12.65 Mean   :22.53
##  3rd Qu.:16.95 3rd Qu.:25.00
##  Max.   :37.97 Max.   :50.00
```

```
#?Boston # 由於為套件內資料，此指令可以查看資料詳細內容
```

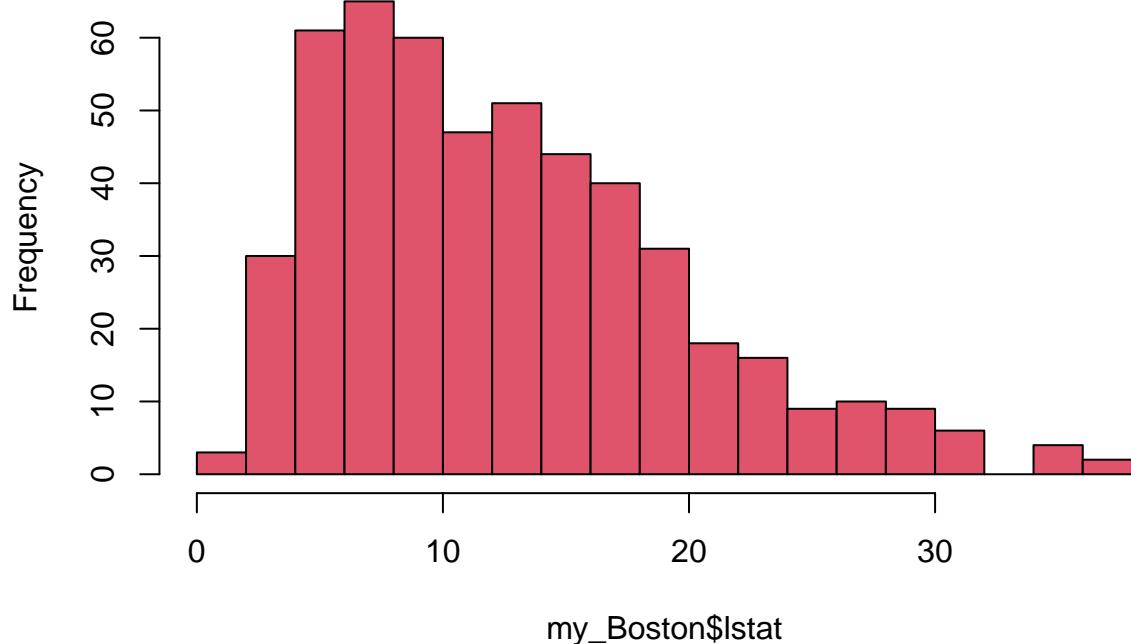
本例目標為預測 medv 和其他變數的關係

```
plot(my_Boston$rm, my_Boston$lstat) # 畫出兩變數的散佈圖
```

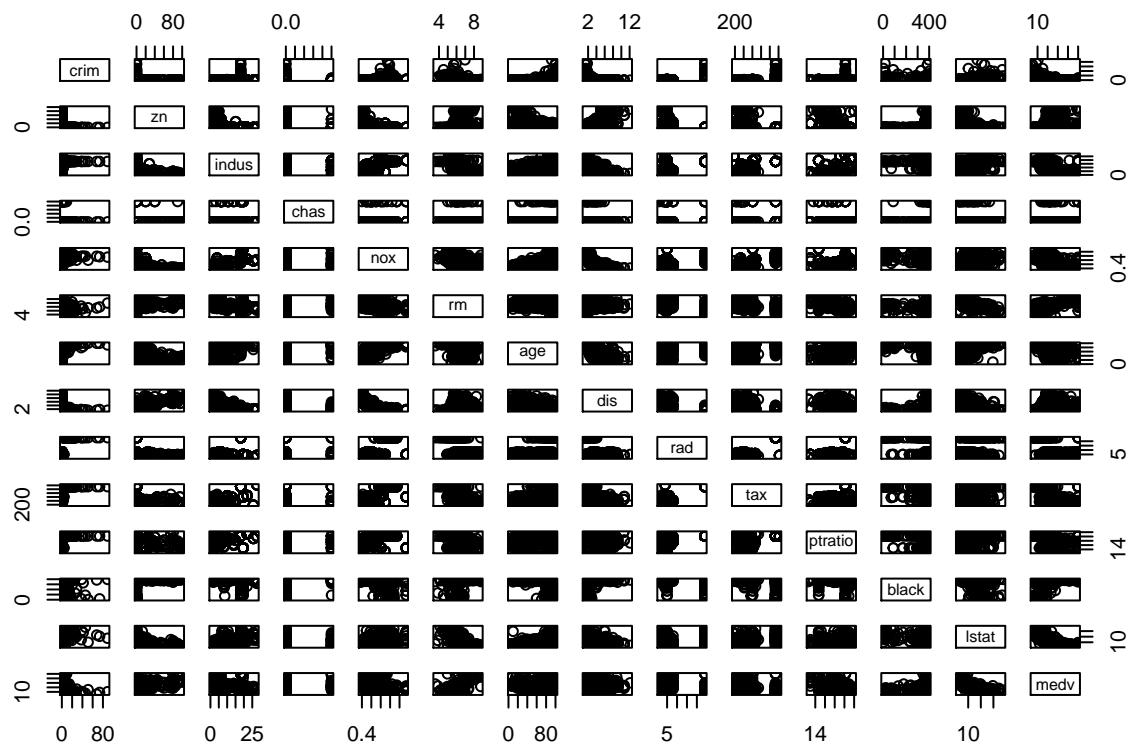


```
hist(my_Boston$lstat, col = 2, breaks = 15) # 畫出單一變數長條圖
```

### Histogram of my\_Boston\$lstat



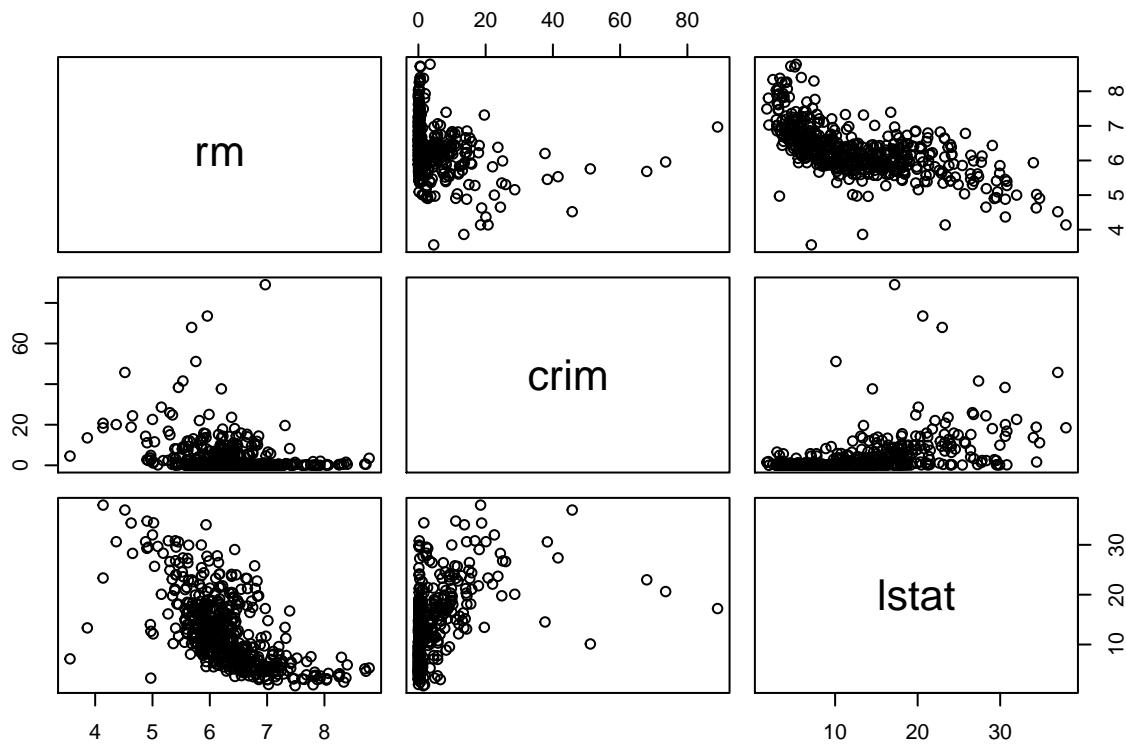
```
pairs(my_Boston) # 畫出資料集所有可能的雙變數散佈圖
```



```

pairs(~ +rm + crim + lstat,
      data = my_Boston
) # 畫出資料集部分的雙變數散佈圖

```



```
cor(my_Boston) # 計算資料集所有的雙變數相關係數
```

```
##          crim      zn      indus      chas      nox
## crim  1.0000000 -0.20046922  0.40658341 -0.055891582  0.42097171
## zn    -0.20046922  1.00000000 -0.53382819 -0.042696719 -0.51660371
## indus  0.40658341 -0.53382819  1.00000000  0.062938027  0.76365145
## chas   -0.05589158 -0.04269672  0.06293803  1.000000000  0.09120281
## nox    0.42097171 -0.51660371  0.76365145  0.091202807  1.00000000
## rm     -0.21924670  0.31199059 -0.39167585  0.091251225 -0.30218819
## age    0.35273425 -0.56953734  0.64477851  0.086517774  0.73147010
## dis    -0.37967009  0.66440822 -0.70802699 -0.099175780 -0.76923011
## rad    0.62550515 -0.31194783  0.59512927 -0.007368241  0.61144056
## tax    0.58276431 -0.31456332  0.72076018 -0.035586518  0.66802320
## ptratio 0.28994558 -0.39167855  0.38324756 -0.121515174  0.18893268
## black  -0.38506394  0.17552032 -0.35697654  0.048788485 -0.38005064
## lstat   0.45562148 -0.41299457  0.60379972 -0.053929298  0.59087892
## medv   -0.38830461  0.36044534 -0.48372516  0.175260177 -0.42732077
##          rm      age      dis      rad      tax      ptratio
## crim  -0.21924670  0.35273425 -0.37967009  0.625505145  0.58276431  0.2899456
## zn     0.31199059 -0.56953734  0.66440822 -0.311947826 -0.31456332 -0.3916785
## indus -0.39167585  0.64477851 -0.70802699  0.595129275  0.72076018  0.3832476
## chas   0.09125123  0.08651777 -0.09917578 -0.007368241 -0.03558652 -0.1215152
## nox   -0.30218819  0.73147010 -0.76923011  0.611440563  0.66802320  0.1889327
## rm     1.00000000 -0.24026493  0.20524621 -0.209846668 -0.29204783 -0.3555015
## age   -0.24026493  1.00000000 -0.74788054  0.456022452  0.50645559  0.2615150
## dis    0.20524621 -0.74788054  1.00000000 -0.494587930 -0.53443158 -0.2324705
```

```

## rad      -0.20984667  0.45602245 -0.49458793  1.000000000  0.91022819  0.4647412
## tax      -0.29204783  0.50645559 -0.53443158  0.910228189  1.000000000  0.4608530
## ptratio   -0.35550149  0.26151501 -0.23247054  0.464741179  0.46085304  1.0000000
## black     0.12806864 -0.27353398  0.29151167 -0.444412816 -0.44180801 -0.1773833
## lstat    -0.61380827  0.60233853 -0.49699583  0.488676335  0.54399341  0.3740443
## medv     0.69535995 -0.37695457  0.24992873 -0.381626231 -0.46853593 -0.5077867
##           black      lstat      medv
## crim     -0.38506394  0.4556215 -0.3883046
## zn        0.17552032 -0.4129946  0.3604453
## indus    -0.35697654  0.6037997 -0.4837252
## chas      0.04878848 -0.0539293  0.1752602
## nox       -0.38005064  0.5908789 -0.4273208
## rm        0.12806864 -0.6138083  0.6953599
## age       -0.27353398  0.6023385 -0.3769546
## dis       0.29151167 -0.4969958  0.2499287
## rad       -0.44441282  0.4886763 -0.3816262
## tax       -0.44180801  0.5439934 -0.4685359
## ptratio   -0.17738330  0.3740443 -0.5077867
## black     1.000000000 -0.3660869  0.3334608
## lstat    -0.36608690  1.0000000 -0.7376627
## medv     0.33346082 -0.7376627  1.0000000

```

## 2.6.4 建立模型

```

lm.fit1 <- lm(medv ~ ., data = my_Boston) # 使用所有變數建立線性模型
summary(lm.fit1) # 查看模型大綱

```

```

##
## Call:
## lm(formula = medv ~ ., data = my_Boston)
##
## Residuals:
##      Min      1Q      Median      3Q      Max 
## -15.595  -2.730  -0.518   1.777  26.199 
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) 3.646e+01  5.103e+00  7.144 3.28e-12 ***
## crim        -1.080e-01  3.286e-02 -3.287 0.001087 ** 
## zn          4.642e-02  1.373e-02  3.382 0.000778 *** 
## indus       2.056e-02  6.150e-02  0.334 0.738288    
## chas        2.687e+00  8.616e-01  3.118 0.001925 ** 
## nox        -1.777e+01  3.820e+00 -4.651 4.25e-06 ***
## rm          3.810e+00  4.179e-01  9.116 < 2e-16 ***
## age         6.922e-04  1.321e-02  0.052 0.958229    
## dis        -1.476e+00  1.995e-01 -7.398 6.01e-13 *** 
## rad         3.060e-01  6.635e-02  4.613 5.07e-06 ***
## tax        -1.233e-02  3.760e-03 -3.280 0.001112 ** 
## ptratio    -9.527e-01  1.308e-01 -7.283 1.31e-12 *** 
## black       9.312e-03  2.686e-03  3.467 0.000573 *** 
## lstat     -5.248e-01  5.072e-02 -10.347 < 2e-16 ***
## ---      
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
```

```

## Residual standard error: 4.745 on 492 degrees of freedom
## Multiple R-squared:  0.7406, Adjusted R-squared:  0.7338
## F-statistic: 108.1 on 13 and 492 DF,  p-value: < 2.2e-16

vif(lm.fit1) # 查看 VIF 值

##      crim      zn    indus     chas      nox       rm      age      dis
## 1.792192 2.298758 3.991596 1.073995 4.393720 1.933744 3.100826 3.955945
##      rad      tax   ptratio    black    lstat
## 7.484496 9.008554 1.799084 1.348521 2.941491

```

有幾個數值得注意， $\text{Pr}(|t|)$  值（簡稱 p 值）越小（<5% 或 <1%），此變數與目標的相關性就越高；VIF 值越高（>10 或 >5），此變數與其他預測變數的相關係數越高，因此要移除 p 值過高的變數，以增加模型準確度以及降低模型複雜度，也要移除 VIF 值過高者以增加模型可信度。

```

lm.fit2 <- lm(medv ~ . - indus - age - tax,
               data = my_Boston
) # 使用所有變數，並且移除部分變數建立線性模型
summary(lm.fit2) # 查看模型大綱

```

```

##
## Call:
## lm(formula = medv ~ . - indus - age - tax, data = my_Boston)
##
## Residuals:
##      Min      1Q      Median      3Q      Max
## -16.2609 -2.9888 -0.5083  1.8041 26.2482
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) 34.712342  5.102742  6.803 2.97e-11 ***
## crim        -0.104843  0.033132 -3.164 0.001650 ** 
## zn          0.036634  0.013412  2.731 0.006532 ** 
## chas        2.967868  0.860830  3.448 0.000614 *** 
## nox        -20.314416  3.472292 -5.850 8.92e-09 *** 
## rm          3.977104  0.407731  9.754 < 2e-16 ***
## dis         -1.429370  0.186922 -7.647 1.08e-13 ***
## rad          0.128761  0.040788  3.157 0.001692 ** 
## ptratio     -1.014914  0.129006 -7.867 2.30e-14 *** 
## black        0.009700  0.002701  3.591 0.000363 *** 
## lstat       -0.528147  0.047930 -11.019 < 2e-16 *** 
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.79 on 495 degrees of freedom
## Multiple R-squared:  0.7342, Adjusted R-squared:  0.7288
## F-statistic: 136.7 on 10 and 495 DF,  p-value: < 2.2e-16

vif(lm.fit2) # 查看 VIF 值

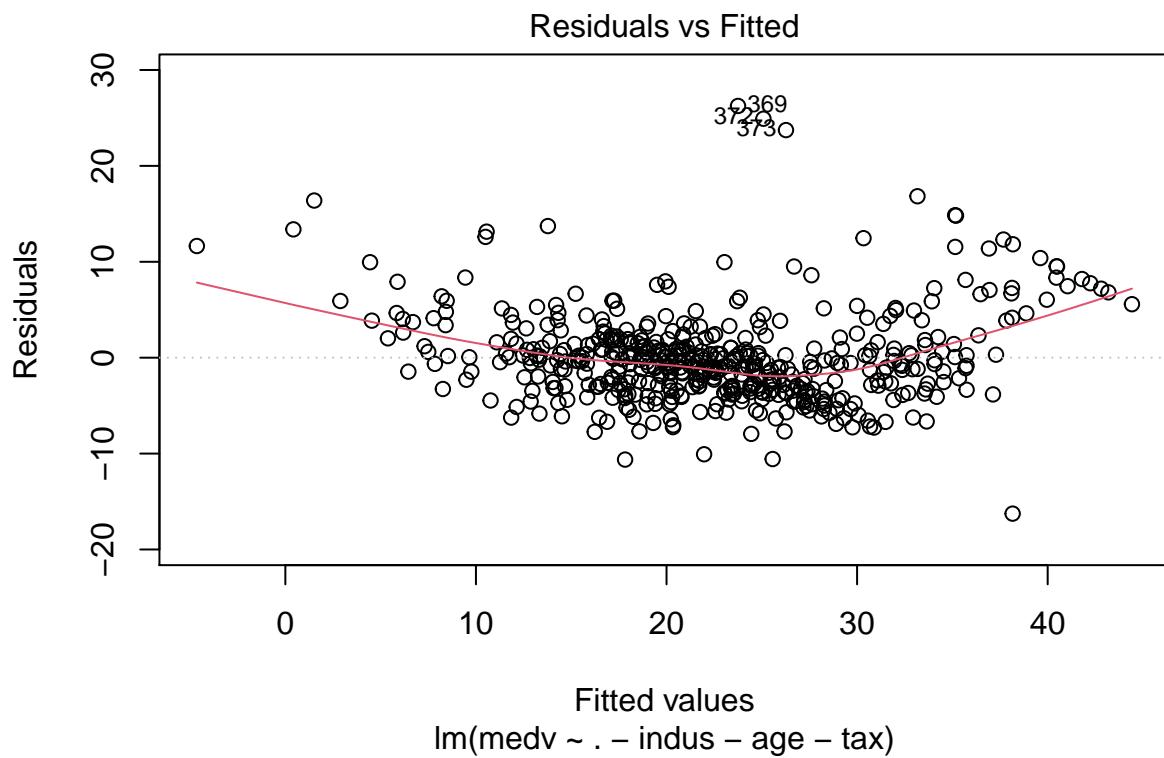
```

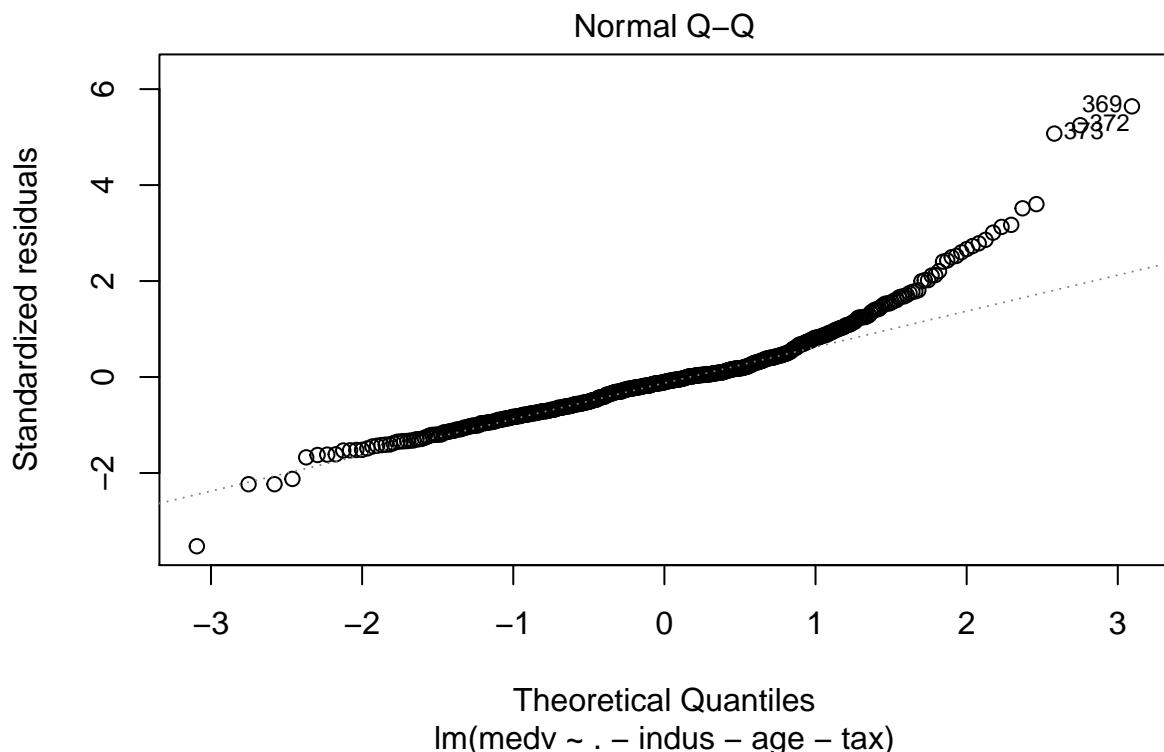
```

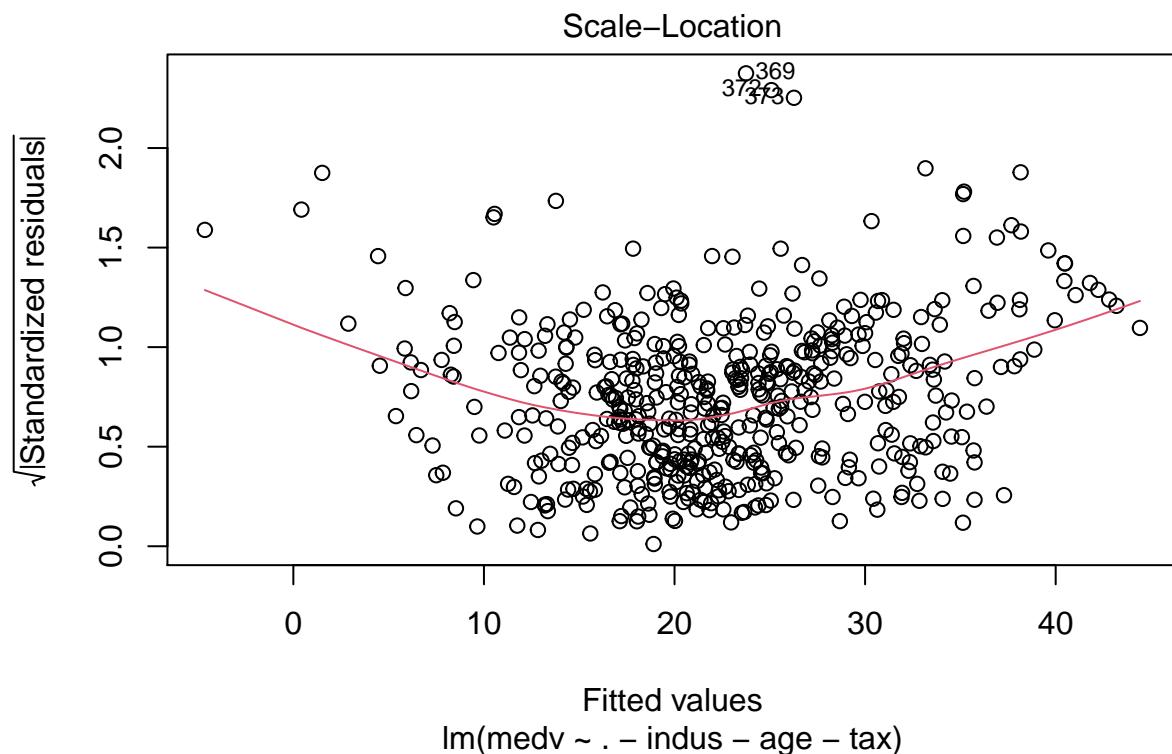
##      crim      zn    chas      nox       rm      dis      rad   ptratio
## 1.787963 2.154054 1.052428 3.564036 1.806735 3.410587 2.776775 1.717222
##      black    lstat
## 1.338982 2.579040

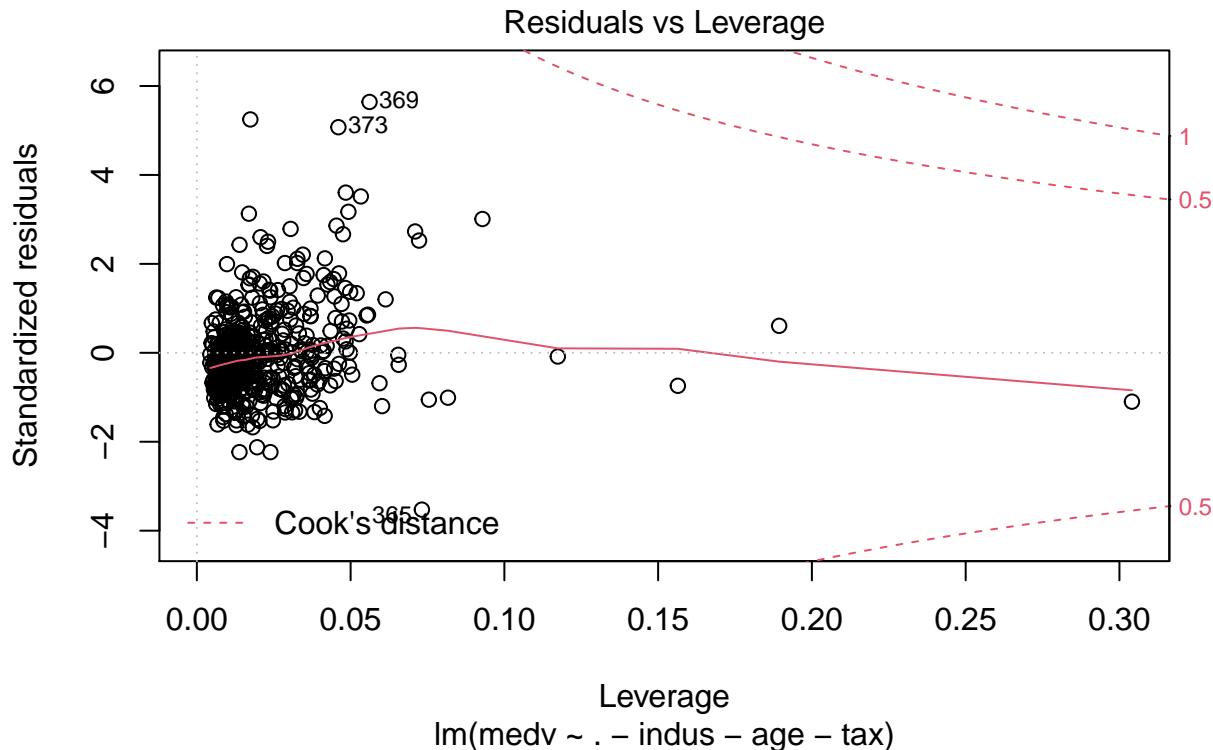
```

```
plot(lm.fit2) # 為此模型繪圖
```









模型大綱中 Adjusted R-squared 代表此資料集有多少目標比例可以藉由此模型解釋，越接近 1 越好，但是注意不要太接近 1，不然會有過度套入—也就是此模型預測現實資料的準確度會很差—的問題；F-statistic 指的是此模型的可信度，越高越好

此外，為了建造準確的模型，注意 Residuals vs. Fitted 的黑點分佈，必須大致是水平、平均分佈的長方形，而不是鐘型、弧形，為了達到此要求，有三種方法

- 增加互動型變數：形如  $a:b$  的變數，因為兩者互動而產生的疊加效果，如果此互動型的 p 值很低，就算 a 與 b 的 p 值很高，還是要留在模型內，此型有助於將弧形轉為水平線
- 增加或扭曲變數：形如  $I(x^2)$ 、 $\log(x)$  的變數，或許與單一變數並不是單純的直線關係，如果高次項的 p 值很低，就算低次項的 p 值很高，還是要留在模型內，此型有助於將弧形轉為水平線
- 將目標變形，形如  $\sqrt{y}$ 、 $\log(y)$ ，或許與目標並不是單純的直線關係，此型有助於將鐘形轉為長方形

還有一點要注意，注意 Residuals vs. Fitted 圖離紅線很遠的點，這些是局外點，也就是與模型誤差最多的資料，旁邊的數字指的是資料位置，記得要移除再試一次，因為這些點會大幅影響誤差和模型，同時有可能成為關鍵資料

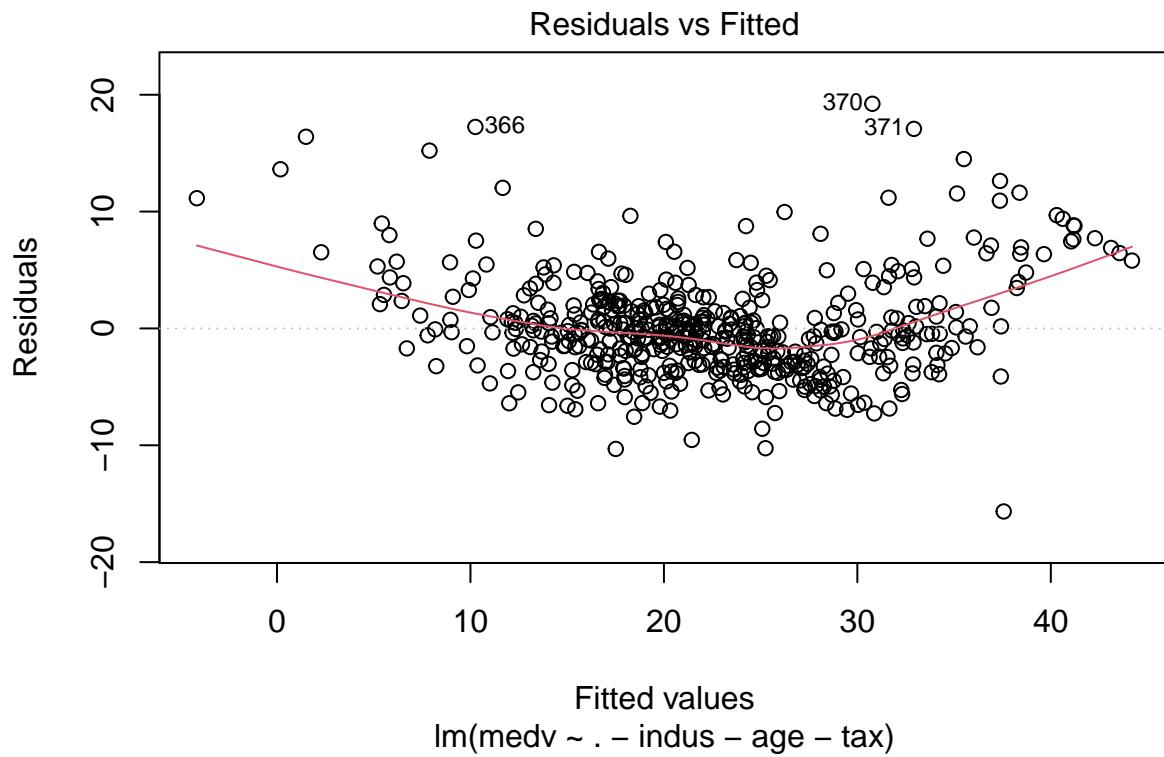
```
lm.fit3 <- lm(medv ~ . - indus - age - tax,
  data = my_Boston[-c(369, 372, 373), ]
) # 移除局外點
summary(lm.fit3)

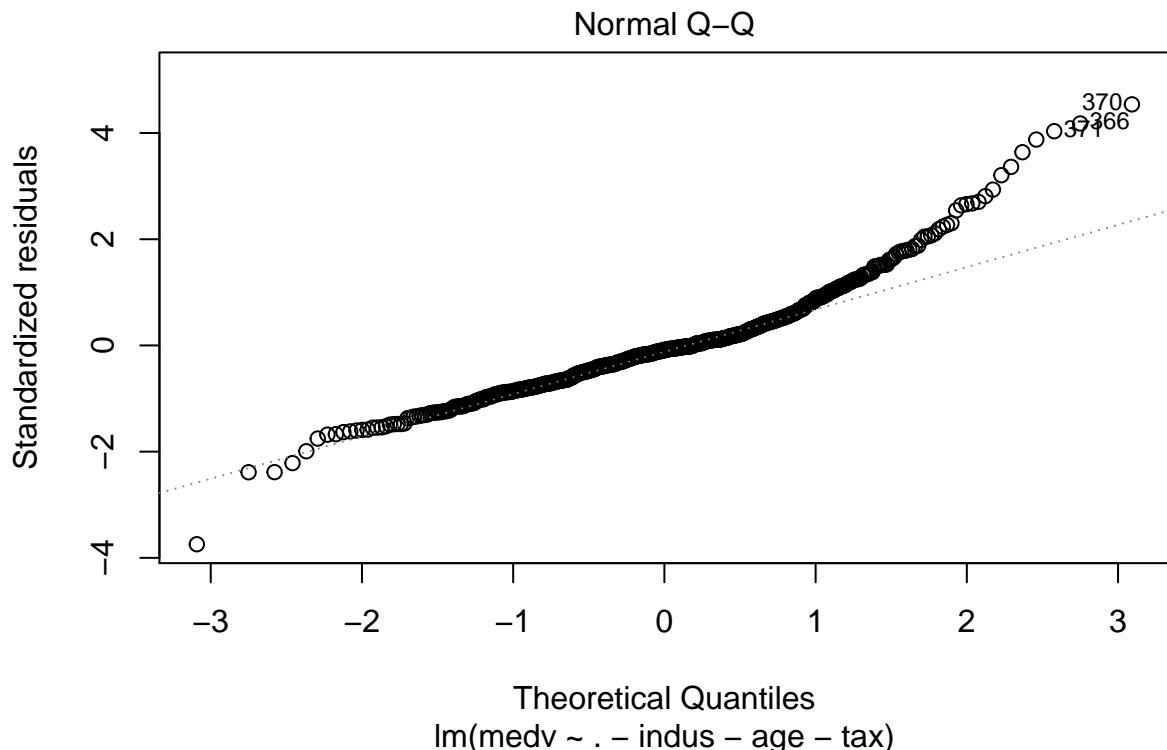
##
## Call:
## lm(formula = medv ~ . - indus - age - tax, data = my_Boston[-c(369,
##   372, 373), ])
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -3.63250  -0.50000  -0.32500  -0.20000  5.00000
```

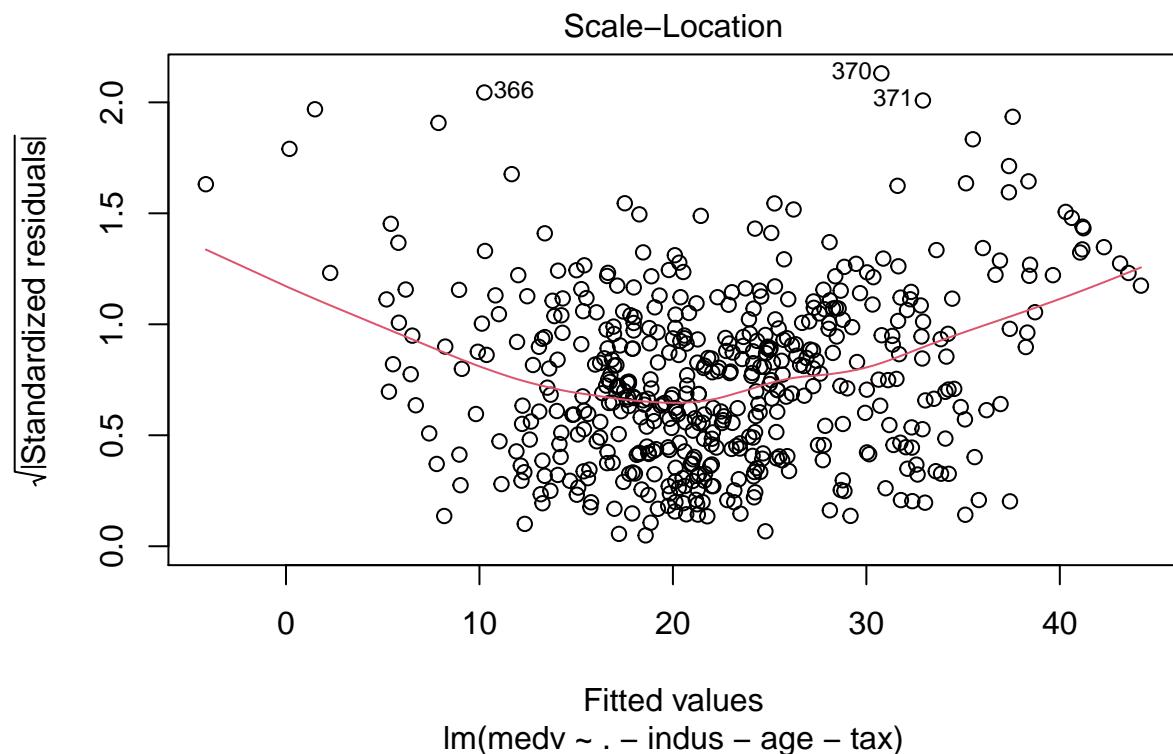
```

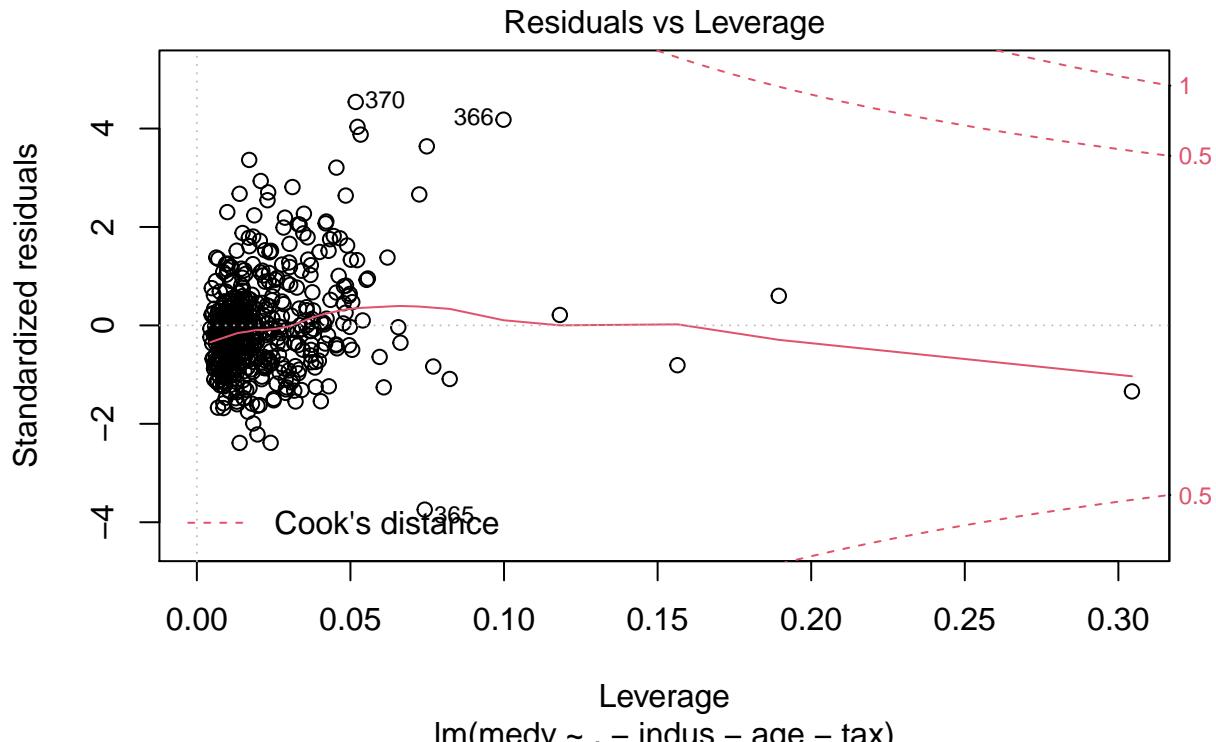
## -15.6641 -2.8410 -0.3751 1.8066 19.2247
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 28.640030  4.680205  6.119 1.92e-09 ***
## crim        -0.095376  0.030100 -3.169  0.00163 **
## zn          0.030425  0.012193  2.495  0.01291 *
## chas        2.405876  0.792192  3.037  0.00252 **
## nox       -19.218002  3.154500 -6.092 2.25e-09 ***
## rm          4.668916  0.377795 12.358 < 2e-16 ***
## dis        -1.229979  0.170831 -7.200 2.27e-12 ***
## rad         0.073739  0.037418  1.971  0.04932 *
## ptratio    -1.025144  0.117171 -8.749 < 2e-16 ***
## black       0.009106  0.002454  3.711  0.00023 ***
## lstat      -0.433398  0.044591 -9.719 < 2e-16 ***
## ---
## Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.349 on 492 degrees of freedom
## Multiple R-squared: 0.7699, Adjusted R-squared: 0.7653
## F-statistic: 164.6 on 10 and 492 DF, p-value: < 2.2e-16
plot(lm.fit3)

```









```

lm.fit4 <- lm(medv ~ . - indus - age - tax + I(rm^2),
  data = my_Boston[-c(369, 372, 373), ]
) # 經過嘗試，增加高次項
summary(lm.fit4)

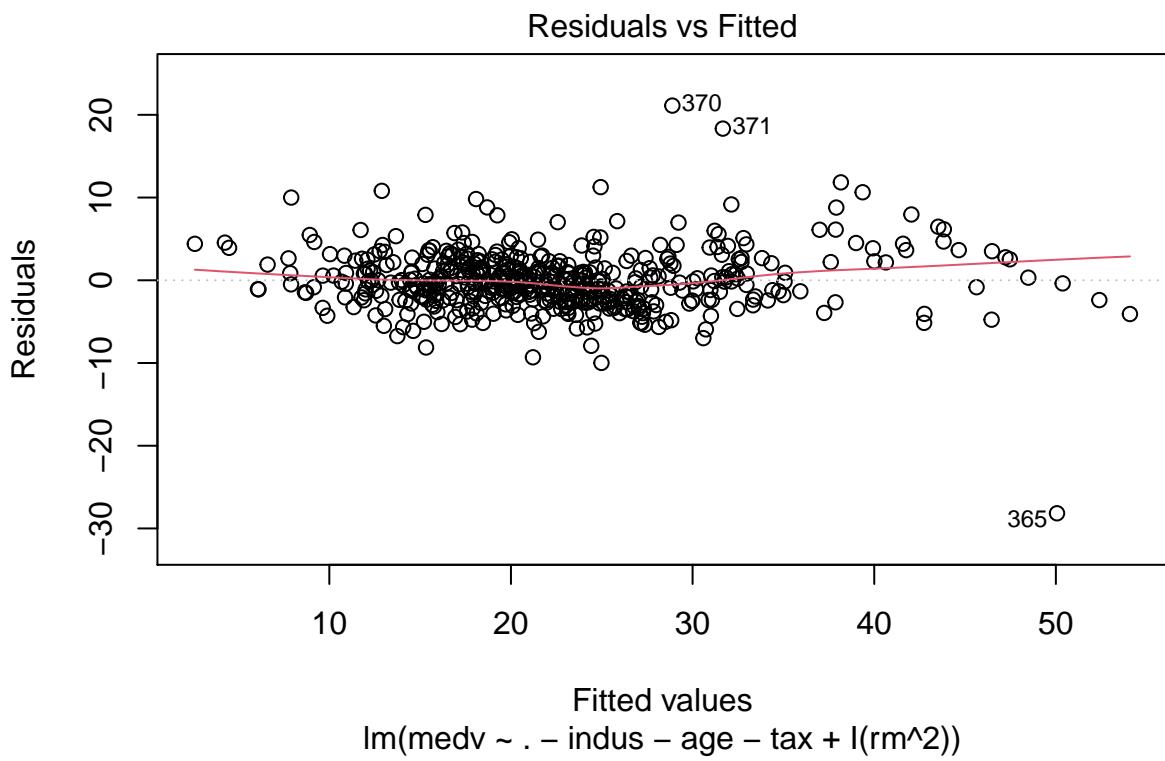
##
## Call:
## lm(formula = medv ~ . - indus - age - tax + I(rm^2), data = my_Boston[-c(369,
##   372, 373), ])
##
## Residuals:
##      Min        1Q    Median        3Q       Max
## -28.1633  -2.1124  -0.1689   1.8064  21.1150
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 129.671088   8.111429 15.986 < 2e-16 ***
## crim        -0.119797   0.025401 -4.716 3.14e-06 ***
## zn          0.019304   0.010295  1.875 0.061376 .
## chas        2.031042   0.667502  3.043 0.002470 **
## nox        -18.061706   2.657161 -6.797 3.10e-11 ***
## rm         -28.494774   2.348941 -12.131 < 2e-16 ***
## dis        -0.926868   0.145395 -6.375 4.23e-10 ***
## rad         0.048122   0.031555  1.525 0.127905
## ptratio     -0.804577   0.099859 -8.057 5.98e-15 ***
## black       0.006917   0.002072  3.339 0.000904 ***

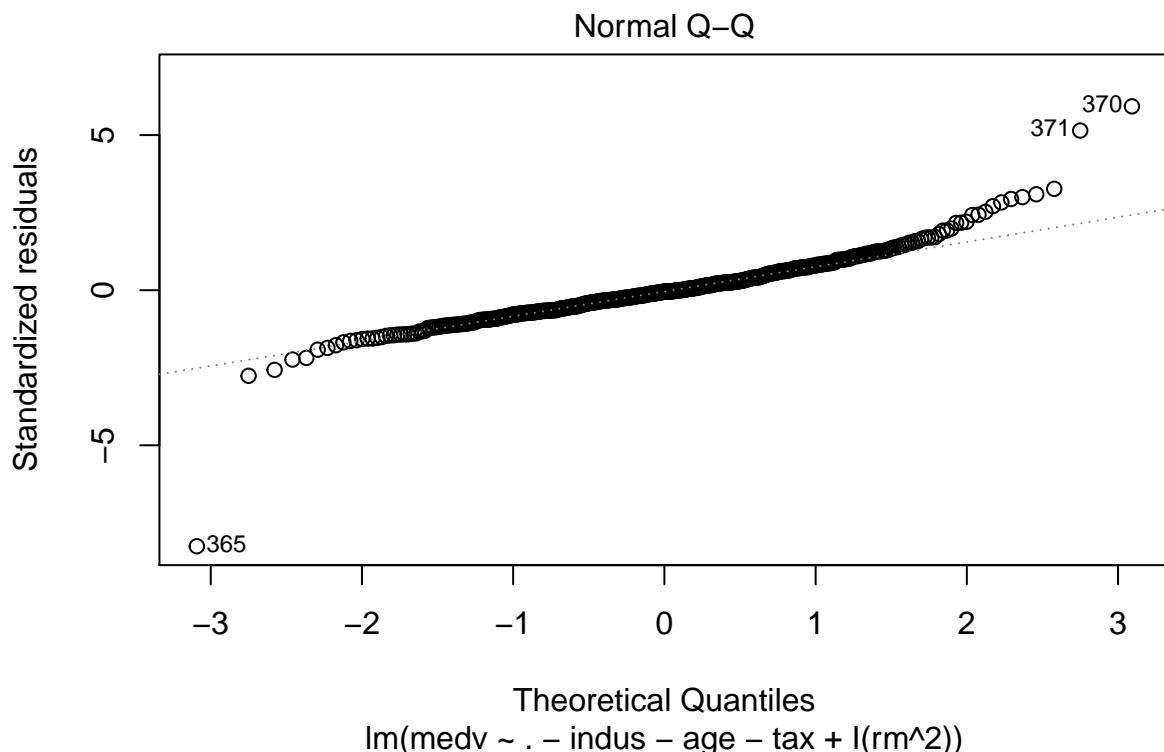
```

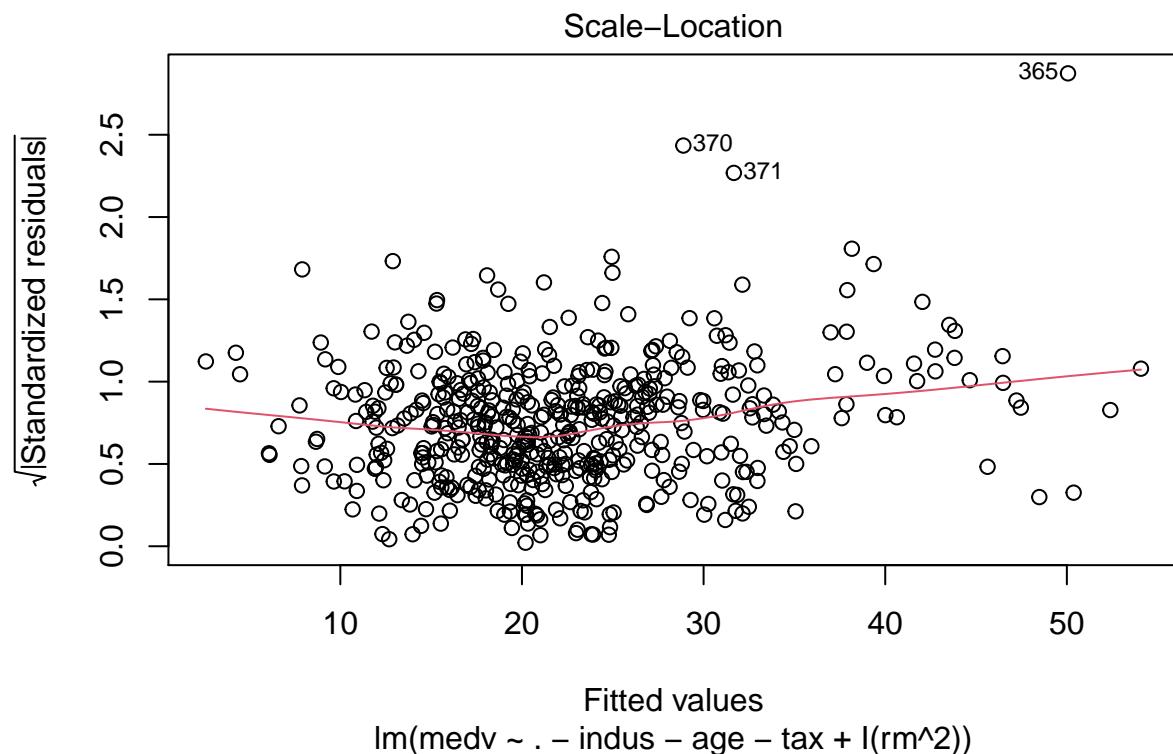
```

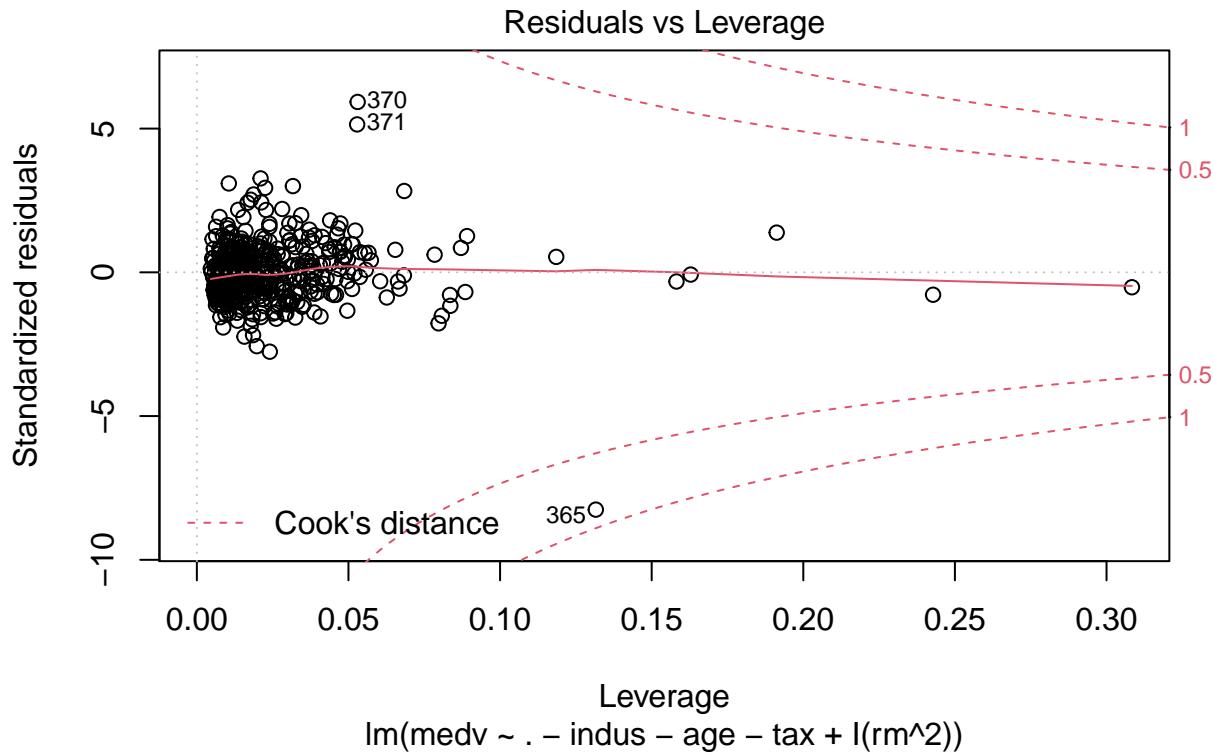
## lstat      -0.458099   0.037583 -12.189 < 2e-16 ***
## I(rm^2)    2.578384   0.180941 14.250 < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.661 on 491 degrees of freedom
## Multiple R-squared:  0.8372, Adjusted R-squared:  0.8336
## F-statistic: 229.6 on 11 and 491 DF,  p-value: < 2.2e-16
plot(lm.fit4) # 此時已接近水平線

```









要如何知道新模型的效果真的比較好？有個方法

```
anova(lm.fit3, lm.fit4) # 比較兩個模型
```

```
## Analysis of Variance Table
##
## Model 1: medv ~ (crim + zn + indus + chas + nox + rm + age + dis + rad +
##                    tax + ptratio + black + lstat) - indus - age - tax
## Model 2: medv ~ (crim + zn + indus + chas + nox + rm + age + dis + rad +
##                    tax + ptratio + black + lstat) - indus - age - tax + I(rm^2)
##   Res.Df   RSS Df Sum of Sq    F    Pr(>F)
## 1     492 9303.9
## 2     491 6581.9  1      2722 203.06 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

如果 F 值越高而且 p 值越低，表示後者模型效果比前者好

經過一番調整與修正，這是目前建造的最佳線性模型

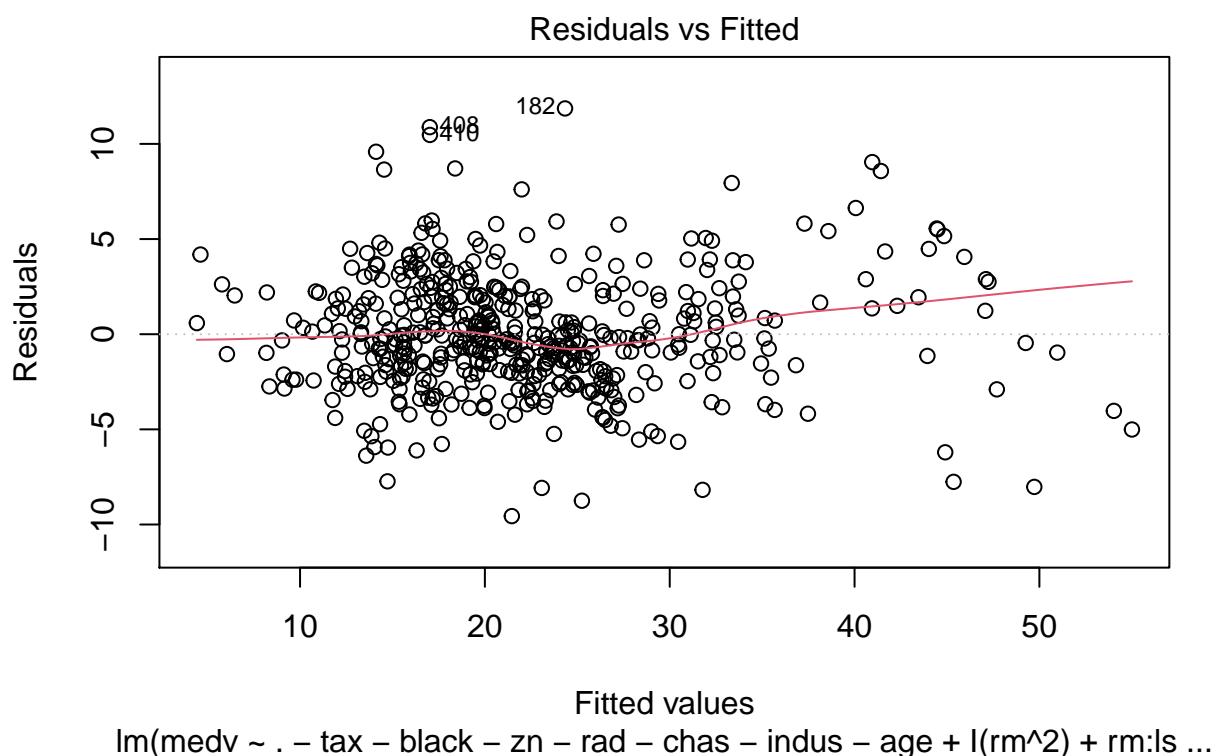
```
lm.fit <- lm(medv ~ . - tax - black - zn - rad -
              - chas - indus - age + I(rm^2) + rm:lstat,
              data = my_Boston[-c(365, 370, 371, 372, 373, 369), ]
) # 加入另一個互動項，並移除更多不相關變數
summary(lm.fit)
```

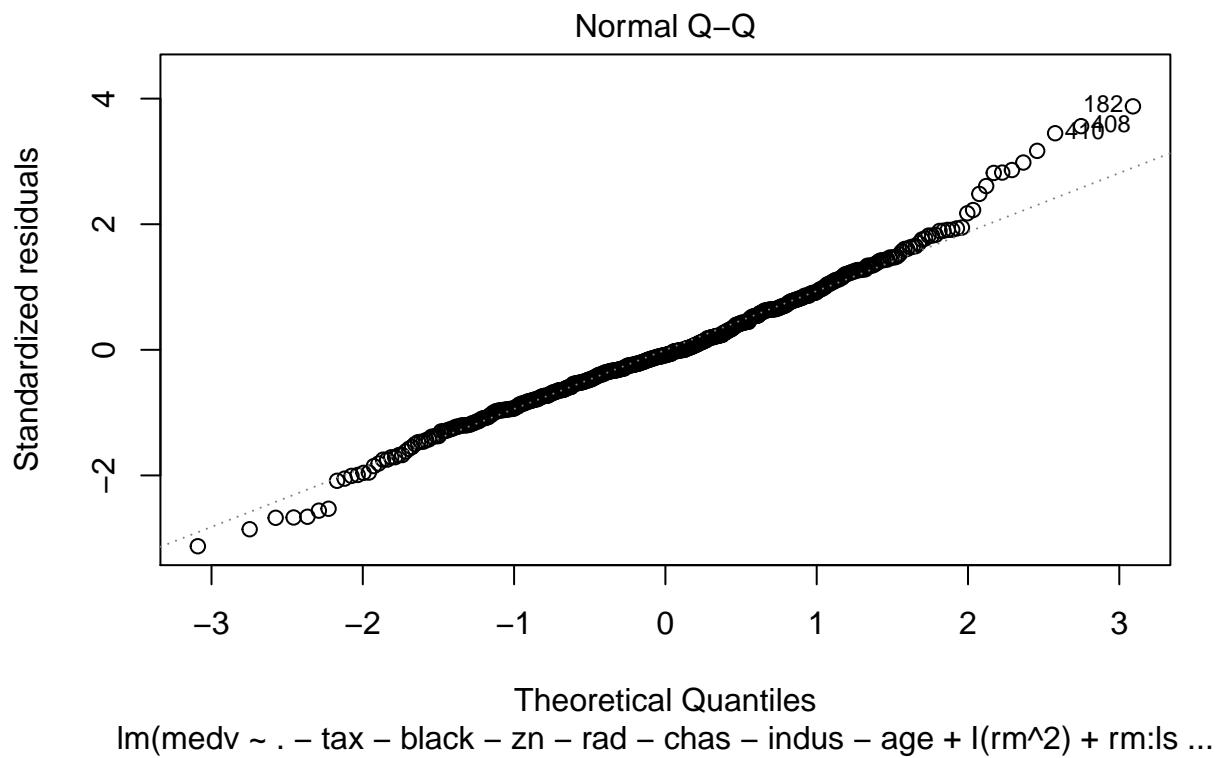
```
##
## Call:
## lm(formula = medv ~ . - tax - black - zn - rad - chas - indus -
```

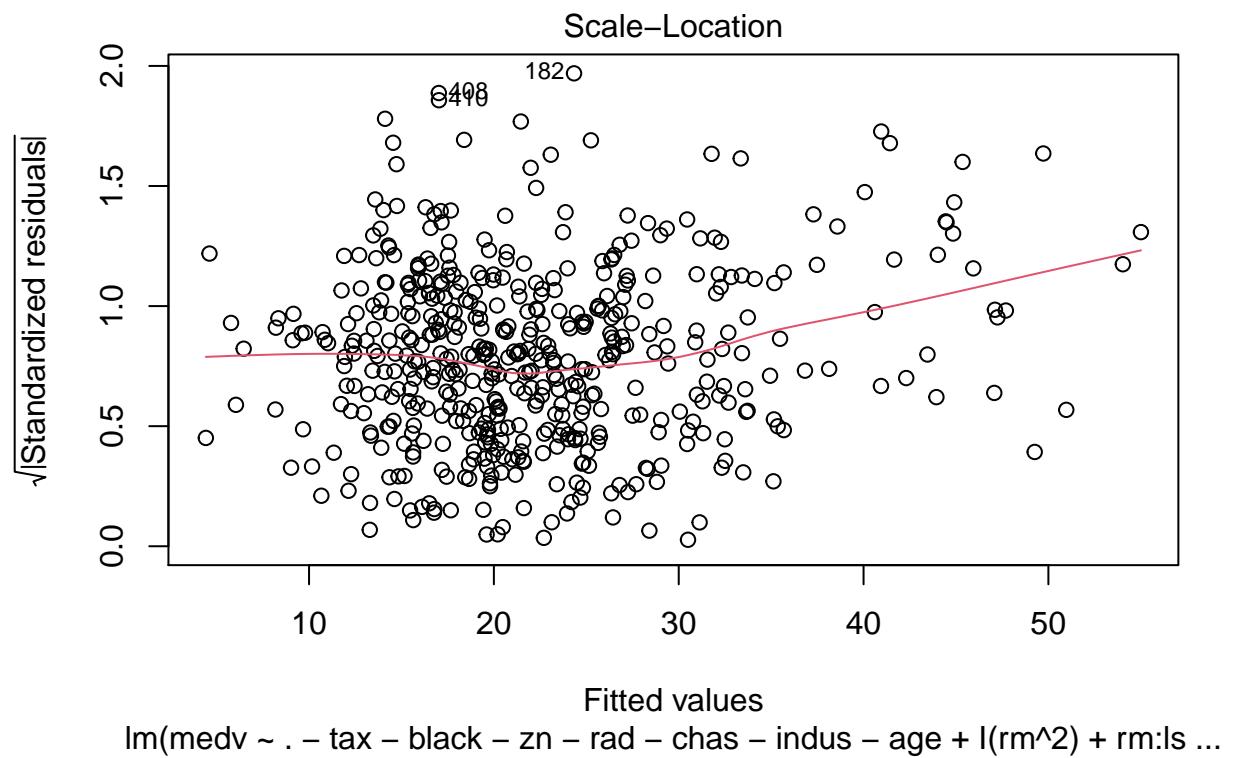
```

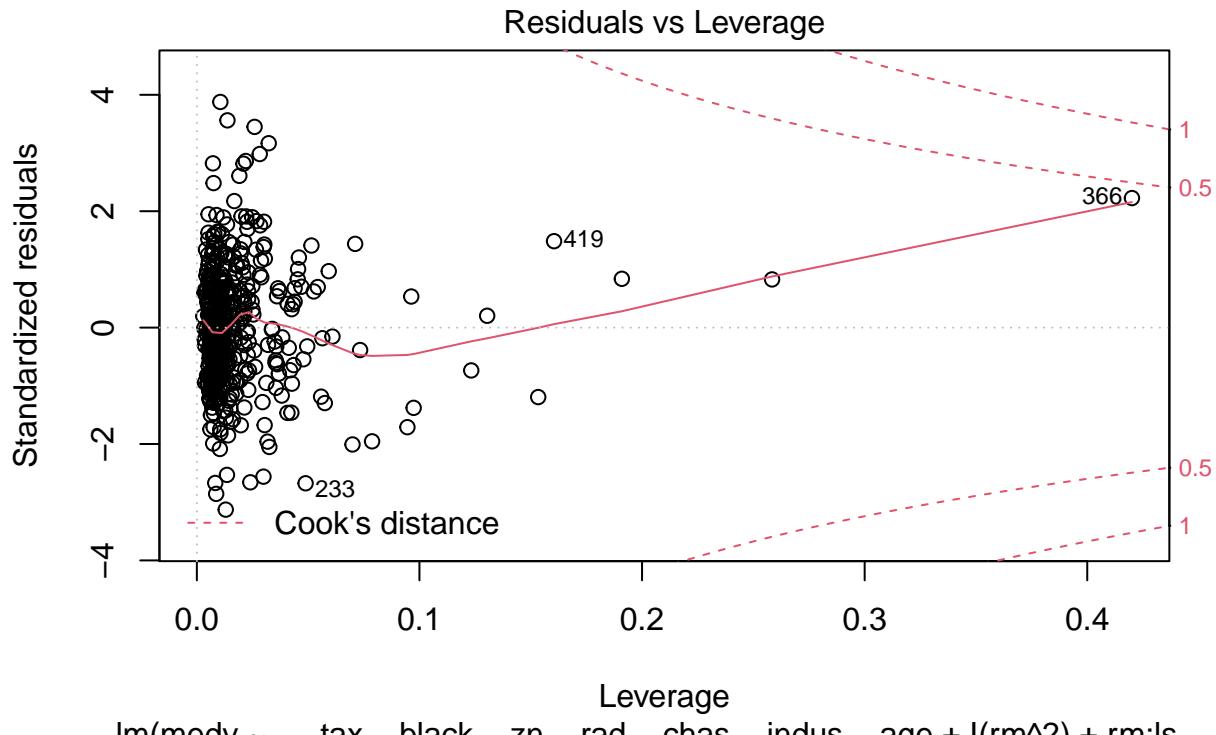
##      age + I(rm^2) + rm:lstat, data = my_Boston[-c(365, 370, 371,
##      372, 373, 369), ])
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -9.5582 -1.9372 -0.2571  1.9227 11.8638
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 86.75059  10.77013   8.055 6.08e-15 ***
## crim        -0.13820   0.01880  -7.352 8.23e-13 ***
## nox         -12.89127   2.08698  -6.177 1.37e-09 ***
## rm          -18.61758   2.99217  -6.222 1.05e-09 ***
## dis          -0.67371   0.10572  -6.373 4.28e-10 ***
## ptratio      -0.67652   0.07478  -9.047 < 2e-16 ***
## lstat        0.92054   0.19315   4.766 2.48e-06 ***
## I(rm^2)      2.03884   0.21352   9.549 < 2e-16 ***
## rm:lstat    -0.23420   0.03297  -7.103 4.33e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.076 on 491 degrees of freedom
## Multiple R-squared:  0.8806, Adjusted R-squared:  0.8786
## F-statistic: 452.6 on 8 and 491 DF,  p-value: < 2.2e-16
plot(lm.fit)

```









```
my_Boston[c(365, 370, 371, 372, 373, 369), ] # 查看局外點
```

```
##      crim zn indus chas   nox     rm    age     dis rad tax ptratio black lstat
## 365 3.47428 0 18.1 1 0.718 8.780 82.9 1.9047 24 666 20.2 354.55 5.29
## 370 5.66998 0 18.1 1 0.631 6.683 96.8 1.3567 24 666 20.2 375.33 3.73
## 371 6.53876 0 18.1 1 0.631 7.016 97.5 1.2024 24 666 20.2 392.05 2.96
## 372 9.23230 0 18.1 0 0.631 6.216 100.0 1.1691 24 666 20.2 366.15 9.53
## 373 8.26725 0 18.1 1 0.668 5.875 89.6 1.1296 24 666 20.2 347.88 8.88
## 369 4.89822 0 18.1 0 0.631 4.970 100.0 1.3325 24 666 20.2 375.52 3.26
##      medv
## 365 21.9
## 370 50.0
## 371 50.0
## 372 50.0
## 373 50.0
## 369 50.0
```

# Chapter 3

## R code style 簡介

好的程式碼架構容易理解與溝通

### 3.1 檔名注意事項

檔名必須有意義，.R 結尾

```
# Good
fit-modols.R
utility-functions.R
# Bad
foo.r
stuff.r
```

如果要依序編譯，開頭用數字排序

```
0-download.R
1-parse.R
2-explore.R
```

### 3.2 變數宣告注意事項

一般變數名稱使用小寫名詞，以底線區隔，設法想出簡潔有力的名稱

```
# Good
day_one
day_1
# Bad
first_day_of_the_month
Dayone
dayone
djm1
```

如果可以，避免變數名稱與現有函數名稱或變數衝突

```
# Bad
T <- FALSE
c <- 10
mean <- function(x) sum(x)
```

### 3.3 插入空白注意事項

在所有二元運算子 (=, +, <-, -, etc) 前後加上一個空格，遇到: , :: 和::: 則不用留空白，在逗號後面加上空格，前面不留以比照英語文法

```
# Good
average <- mean(feet / 12 + inches, na.rm = TRUE)
x <- 1:10
base:::get
# Bad
average<-mean(feet/12+inches,na.rm=TRUE)
x <- 1 : 10
base :: get
```

在左小括弧前加一個空白，如果是呼叫函數的左小括弧就不用

```
# Good
if (debug) do(x)
plot(x, y)
# Bad
if(debug)do(x)
plot (x, y)
```

如果有助於讓宣告符號對齊，可以加入額外的空白

```
list(
  total <- a + b + c,
  mean   <- (a + b + c) / n
)
```

小括弧和中括弧內側不要留白，遇上逗點例外

```
# Good
if (debug) do(x)
diamonds[5, ]
# Bad
if ( debug ) do(x)
x[1,]
x[1 ,]
```

左大括號絕不獨立成行，而且其後面的內容會從下一行開始；右大括號獨立成行，遇到 else 例外，大括號內總是要縮排

```
if (y < 0 && debug) {
  message("Y is negative")
}
if (y == 0) {
  log(x)
} else {
  y^x
}
# Bad
if (y < 0 && debug)
message("Y is negative")
if (y == 0) {
  log(x)
}
else {
```

```
y ~ x  
}
```

如果條件很短，允許放在同一行

```
if (y < 0 && debug) message("Y is negative")
```

如果可以，每行盡量不要超過 80 字元，或是 40 中文字元

縮排使用兩個空白鍵，不要使用 tab 鍵，唯一的例外是函數宣告不只一行，此時把第二行縮排至定義開始的位置

```
long_function_name <- function(a = "a long argument",  
                                b = "another argument",  
                                c = "another long argument") {  
  # 程式碼一樣縮排兩個空白鍵  
}
```

## 3.4 其他注意事項

變數宣告使用 <- 而非 =

```
# Good  
x <- 5  
# Bad  
x = 5
```

加入註解使用註解符號 # 加上一個空格，註解用來說明為什麼這樣做，而不是做了什麼，使用註解的直線 - 或 = 將你的程式碼分成不同區塊

```
# Load data -----  
  
# Plot data -----
```

## 3.5 其他：Google 的 Code Style

Google 使用駝峰字動詞來命名函數，以和物件區隔；如果是私用函數，用一個點作為命名開頭

```
# Good  
DoNothing <- function() {  
  return(invisible(NULL))  
}  
.DoNothingPrivately <- function() {  
  return(invisible(NULL))  
}
```

不要使用 attach() 函數，容易出現錯誤

不要使用 -> 定義變數，不然尋找變數命名位置會更加困難

函數的 return() 不要省略

```
# Good  
AddValues <- function(x, y) {  
  return(x + y)  
}  
# Bad  
AddValues <- function(x, y) {
```

```
x + y  
}
```

在 Google 這種具有大量 R 程式碼的公司，外來套件的函數容易與現有名稱衝突，使用`::` 方法引入外來函數，如果是預設 R 套件的函數就不需要

```
# Good  
purrr::map()
```

所有套件都需要 `packagename-package.R` 套件說明文件

# Chapter 4

## 使用 R Shiny 建立互動式文件

本章節說述如何建立 R Shiny 文件，文件結構說明，以及一個實例

### 4.1 建立 R Shiny 檔案

打開 RStudio，點選右側的 Packages 標籤，點選 install，第二行輸入 shiny 然後安裝  
點選最上面 file -> New File -> Shiny Web App... 輸入資料夾名稱後建造，系統會產稱預設檔，點擊上方 Run App 即可執行

### 4.2 文法大綱

文法大綱在[這裡](#)，每個 shiny 文件都遵照以下格式

```
library(shiny)
ui <- fluidPage(
  # 在這裡寫使用者介面及輸入資料的方法
)
server <- function(input, output) {
  # 在這裡接收輸入轉成輸出送回使用者介面
}
shinyApp(ui = ui, server = server)
```

### 4.3 UI 撰寫

#### 4.3.1 版面配置選項

參考上面連結第二頁右側，先決定版面配置，之後於配置子元素內定義輸入與輸出格式，或是加入更多版面，舉例：

```
sidebarLayout(
  sidebarPanel(
    # 定義輸入與輸出物件
  ),
  mainPanel(
    # 定義輸入與輸出物件
  )
)
```

這個配置設定會把版面切成左三又七兩部份

### 4.3.2 定義使用者輸入物件

所有輸入遵照這個格式：

```
*Input("inputID", "label", ...)
```

其中 \*Input 定義輸入模式，例如 numericInput 讓使用者輸入一個數字，checkboxGroupInput 讓使用者可以複選給定選項，上面的連結第一頁右側提供了多數的 \*Input 範例可供參考；inputID 定義輸入物件的 ID，每個輸入物件的 ID 必須不同，於 server 函數使用 input\$inputID 獲得輸入值；label 則是輸入上方會顯示的文字，可用於告知使用者輸入的相關資訊；之後的參數視一開始的 \*Input 函數而定，可參考上面文件，或是使用 RStudio 右側的 Help 標籤查閱說明

### 4.3.3 宣告輸出版面格式

所有輸出格式的宣告遵照這個格式：

```
*Output("outputID")
```

其中 \*Output 定義輸出模式，例如 plotOutput 宣告輸出格式為統計圖，verbatimTextOutput 宣告輸出為程式碼結果，上面的連結第一頁下面提供了多數的 \*Output 範例可供參考；outputID 定義輸出物件 ID，於 server 函數使用 output\$outputID <- render\* 定義輸出程式

## 4.4 server 撰寫

### 4.4.1 前置作業處理

你可以在決定輸出以前先在此定義其他變數以協助後續程式編輯工作，一般情況用普通的 R 語言宣告就行，但是如果這個變數宣告會用到 UI 的輸入，那就要改用下面這個做法：

```
re <- reactive({  
  # 像是寫函數一樣定義此變數  
})
```

以此例而言，如果要呼叫此變數，使用 re() 而非直接使用變數名稱

### 4.4.2 輸出處理

在 UI 介面宣告輸出格式後，在此處撰寫將輸入轉成輸出的方法

```
output$outputID <- render*({  
  # 你的轉換方法  
})
```

這裏 render\* 取決於 outputID 的輸出格式，例如 tableOutput 物件要使用 renderTable 函數處理輸出

## 4.5 以 Boston 資料集為例

```
library(shiny) # 應格式要求  
library(MASS) # Boston 資料集來源  
library(car) # vif 函數來源  
  
ui <- fluidPage( # 應格式要求  
  titlePanel("Boston Data Analisis"), # 定義標題版面與標題  
  sidebarLayout( # 宣告左三右七的版面配置
```

```

sidebarPanel( # 定義左側版面
  checkboxGroupInput("predictors", # 宣告複選輸入
    "Choose your predictors", # 複選標題
    choices = c(
      "crim" = 1, # 以下定義所有選項
      "zn" = 2,
      "indus" = 3,
      "chas" = 4,
      "nox" = 5,
      "rm" = 6,
      "age" = 7,
      "dis" = 8,
      "rad" = 9,
      "tax" = 10,
      "ptratio" = 11,
      "black" = 12,
      "lstat" = 13,
      "I_rm_2_" = 15,
      "rm_lstat" = 16
    ), # 所有選項宣告結束
    selected = c(
      "crim" = 1, # 以下定義預設選取選項
      "zn" = 2,
      "indus" = 3,
      "chas" = 4,
      "nox" = 5,
      "rm" = 6,
      "age" = 7,
      "dis" = 8,
      "rad" = 9,
      "tax" = 10,
      "ptratio" = 11,
      "black" = 12,
      "lstat" = 13
    ) # 預設選項設定結束
  ), # 複選輸入定義結束
  numericInput("removeRowNum", # 定義整數輸入
    "Choose how many rows to remove (up to 10)", # 整數輸入標題
    value = 0, # 預設值
    min = 0, # 最小值
    max = 10 # 最大值
  ), # 整數輸入定義結束
  conditionalPanel( # 定義條件版面，以下皆是
    condition = "input.removeRowNum > 0", # 根據整數輸入決定是否出現
    numericInput("RRow1", # 定義整數輸入
      "Remove row number 1", # 整數輸入標題
      min = 1, # 最小值
      max = 507, # 最大值
      value = 507 # 預設值
    ) # 整數輸入定義結束
  ), # 條件版面定義結束
  conditionalPanel(
    condition = "input.removeRowNum > 1",

```

```

        numericInput("RRow2",
            "Remove row number 2",
            min = 1,
            max = 507,
            value = 507
        )
    ),
    conditionalPanel(
        condition = "input.removeRowNum > 2",
        numericInput("RRow3",
            "Remove row number 3",
            min = 1,
            max = 507,
            value = 507
        )
    ),
    conditionalPanel(
        condition = "input.removeRowNum > 3",
        numericInput("RRow4",
            "Remove row number 4",
            min = 1,
            max = 507,
            value = 507
        )
    ),
    conditionalPanel(
        condition = "input.removeRowNum > 4",
        numericInput("RRow5",
            "Remove row number 5",
            min = 1,
            max = 507,
            value = 507
        )
    ),
    conditionalPanel(
        condition = "input.removeRowNum > 5",
        numericInput("RRow6",
            "Remove row number 6",
            min = 1,
            max = 507,
            value = 507
        )
    ),
    conditionalPanel(
        condition = "input.removeRowNum > 6",
        numericInput("RRow7",
            "Remove row number 7",
            min = 1,
            max = 507,
            value = 507
        )
    ),
    conditionalPanel(

```

```

        condition = "input.removeRowNum > 7",
        numericInput("RRow8",
            "Remove row number 8",
            min = 1,
            max = 507,
            value = 507
        )
    ),
    conditionalPanel(
        condition = "input.removeRowNum > 8",
        numericInput("RRow9",
            "Remove row number 9",
            min = 1,
            max = 507,
            value = 507
        )
    ),
    conditionalPanel(
        condition = "input.removeRowNum > 9",
        numericInput("RRow10",
            "Remove row number 10",
            min = 1,
            max = 507,
            value = 507
        )
    )
),
# 左側版面定義結束
mainPanel( # 定義右側版面
    tabsetPanel( # 定義標籤版面
        tabPanel(
            "summary", # 標籤一標題
            verbatimTextOutput("summary") # 宣告為程式碼結果輸出
        ), # 標籤一宣告結束
        tabPanel(
            "vif", # 標題二標題
            verbatimTextOutput("vifs")
        ),
        tabPanel(
            "residual",
            plotOutput("residue") # 宣告為統計圖輸出
        ),
        tabPanel(
            "outliers",
            verbatimTextOutput("myoutliers")
        )
    ) # 標籤版面宣告結束
) # 右側版面宣告結束
) # 左三又七版面定義結束
) # UI 版面定義結束

server <- function(input, output) { # 應格式需求
    my_data <- data.frame(
        "I_rm_2_" = Boston$rm^2, # 產生二次項

```

```

"rm_lstat" = Boston$rm * Boston$lstat # 產生互動項
) # 新資料定義結束
full_Boston <- cbind(Boston, my_data) # 將新變數加入原始資料
lm.fit <- reactive({ # 宣告互動物件
  my_Boston <- full_Boston[-c(
    input$RRow1, # 移除選定資料
    input$RRow2,
    input$RRow3,
    input$RRow4,
    input$RRow5,
    input$RRow6,
    input$RRow7,
    input$RRow8,
    input$RRow9,
    input$RRow10
  ), ]
  attach(my_Boston) # 將資料集的變數直接拿來使用
  xnam <- paste0( # 將選擇列名黏合
    colnames( # 將列數轉成列名
      my_Boston[as.double(input$predictors)] # 選擇使用者複選的變數列
    )
  )
  fmla <- as.formula( # 宣告線性模組參數
    paste("medv", "~", paste(
      xnam,
      collapse = " + "
    )))
  )
  nfmla <- paste(
    "lm(formula = medv", "~", paste( # 宣告線性模組運算名稱
      xnam,
      collapse = " + "
    )),
    ",",
    "data = Boston)"
  )
  model1 <- lm(fmla) # 產生顯性模組
  model1[["call"]] <- nfmla # 新增運算名稱
  model1 # 物件輸出
})
output$summary <- renderPrint({
  # 標籤一輸出方法定義
  summary(lm.fit()) # 模型大綱
})

output$vifs <- renderPrint({
  vif(lm.fit()) # vif 值
})

output$residue <- renderPlot({
  par(mfrow = c(2, 2))
  plot(lm.fit()) # 統計圖
})

```

```
output$myoutliers <- renderPrint({  
  Boston[c(  
    input$RRow1, # 查看使用者移除資料  
    input$RRow2,  
    input$RRow3,  
    input$RRow4,  
    input$RRow5,  
    input$RRow6,  
    input$RRow7,  
    input$RRow8,  
    input$RRow9,  
    input$RRow10  
)  
, ]  
)  
}  
  
shinyApp(ui = ui, server = server)
```