目 錄

表 目 錄

圖 目 錄

LaTeX 與我

A Guide To TeX

吳尙樺

2006.3.7

第 1 章
先來說一些故事

"[T]he Tex research project that I embarked on was driven by two major goals. The first goal was quality: we wanted to produce documents that was not just nice, but actually the best."

"I never intended to have a system that would be universal and used by everybody. I always wanted to write a system that would be used for just the finest books."

Donald E. Knuth[1]

相信一定很多人跟作者一樣, 首次初次接觸功能如此強大的排版軟體。大多數人會感到不知從何下手, 心中徬遑, 感覺很難上手, 一步步退卻, 最後乾脆放棄, 其實他並沒有你想像中那麼難操作。誠如 Title 頁, **A Guide To TEX** , 此書手册給剛接觸 TEX 新手的參考用書。接下來幾章會陸續介紹這個軟體及如何操作, 大家不用急, 一章一章慢慢來, 從最基本的文稿結構, 到進階的表, 圖, 數學式子, 假以時日也會變成進階使用者, 對大部分人而言, 不論是排版報告或者論文, 已經很足夠了。讀者也可以試著編寫屬於自己的使用手册, 就如此手册一樣。編排的當下, 你會遇到很多困難, 例如表格的運用, 圖表的引用等等, 一旦克服, 康莊大道就在你面前。若要變成排版之神, 可以參閱網路上的使用手册, 加上自己慢慢鑽研, 最後一定可以變成 "TEX 與你" , 你玩 TEX 而非 TEX 玩你。作者與大家共勉之。

TEX 是 Donald E. Knuth 教授的精心傑作, 它是個功能非常強大的幕後排版系統, 含有彈性很大, 而且很低階的排版語言。當初, 是因爲 Knuth 教授在寫他的大著 TAOCP(The

[1]TeX 是由 Knuth 教授所發展的排版系統。

Art of Computer Programming) 時, 發覺書商把他書中的數學式子排得太難看了, 於是決定自行開發一個非常適合排數學式子的排版語言, 這就是 TEX 系統的來由。

不僅僅是談到 TEX 一定會提到 Knuth 教授, 只要提到排版, 沒有人可以忽略他的 TEX 所帶來的變革, 影響, 甚至 TEX 已經 20 幾歲了, 仍然深深影響著排版界及排版軟體, 可見這個排版軟體眞的是非同小可。

1.1 LATEX 又是什麼呢?

TEX 是個很低階的排版語言, 如果排版時都要從這種低階語言來控制版面的話, 那將會非常的煩複, 所以, 一些經常要用到的功能, 都會先去定義好 (稱爲巨集, macro), 這樣排版時才會方便, 快速, 直接引用已定義好的巨集裡頭的指令就可以了。

原始的 TEX 已經有了一組 macro, 是 Knuth 教授所寫的, 那就是著名的 Plain TEX, 但仍然不夠方便, 直觀, 於是 Leslie Lamport 又寫了另一組的 macro,[2] 稱爲 LATEX, 主要是把版面配置和文章內容, 適度的分開處理, 只要使用者選定了一種類別, 整本書或整篇文章的結構就是按照這個類別來安排版面, 這樣寫文件的人只要專注於文章內容就可以了, 版面配置就完全交給 TEX / LATEX 去處理。

既然 LATEX 只不過是 TEX 的一大組巨集, 那, 當然原來 TEX 的指令, 大部份也可以用在 LATEX 文稿當中的。而且,LATEX 並不是目前唯一的 TEX macro, 其他如 eplain TEX, ConTeXt, TeXinfo 等都是 TEX macro, 也都有他們自成一套的語法。目前的 LATEX 由 LATEX 3 Project 所維護及發展。[3]

如果談到這裡, 你還是霧煞煞的話, 類比成 HTML markup 標記語言就能大概知道一些概念了, 當然, 這和 HTML 標記語言是可相提而無法並論的。如果連 HTML 也不熟悉, 那也沒關係啦! 這章本來就是在說故事嘛! 只要繼續看後面的內容就行了。

1.2 一般人對 LATEX 的迷思

這裡引用 Peter Flynn 在他的 A beginner's introduction to typesetting with LATEX 一文中所提出來的六大迷思, 並添加個人的一些看法:

[2]Leslie Lamport 於 1985 在 CSL 任職時寫了 LATEX 。目前任職於 DEC Systems Research Center 但已幾乎沒有參與 LATEX 的後續發展了。

[3]http://www.latex-project.org/latex3.html

- LaTeX 只能使用一種字型?

  當然不是, 雖然 TeX 系統預設是使用當初 Knuth 教授所設計的 Metafont, 但目前 OpenType, TrueType, Adobe Type1 字型都可以用在 LaTeX 當中, 只不過, 安裝字型的部份不是那麼的直觀就是了, 但比起其他的排版系統, TeX / LaTeX 所能利用的字型種類, 可以說是最多的。

- LaTeX 只能用於 Unix-like 的作業系統?

  Knuth 教授慷慨的把 TeX 的原始程式碼開放出來, 所以, 只要是有人在使用的作業系統都可以移植過去, 不必擔心版權的問題。像 MS DOS, OS/2, MS Windows, Mac OS, Unix-like 系統都有 TeX 的移植版本, 甚至是 PDA(e.g. the Sharp Zaurus) 都有 TeX / LaTeX 的縱跡。可以說是走到哪兒, 就可以用到哪兒, 而且, 文稿都是互通的, 列印結果也相同。

- LaTeX 已經過時了?

  剛好相反, LaTeX Project 及其他相關 packages 正穩定的研發當中, 尤其和目前新一代的 SGML/XML/HTML 及資料庫系統, 都積極的想辦法銜接起來。對於數學式子的排版, 至今無人能出其右。有興趣的話, 可以參考 news://comp.text.tex 的流量, 及其中 CTAN 對於巨集更新,[4] 上傳的消息發布。

- LaTeX 沒有所見即所得 (WYSIWYG)?

  某種層面上而言, 是的。因爲他本質上是幕後排版系統。但是, 所產生的 dvi/ps/pdf 檔, 可以很精準的顯示你所想要表達的內容, 不曉得這算不算是「所見即所得」? 另外, 一些相關 GUI 配合的軟體, 也會打破幕後排版的定義, 例如 LyX,[5] TeXmacs 等等。[6]

- LaTeX 很難學?

  這個嘛! 我只能說, 有什麼東西是很好學的? 如果只是一般使用, 而不是當個排版專家, 甚至 TeX / LaTeX programmer, 那麼, 幾十分鐘的說明, 就可以「上工」了! 剩下的只是一些細部調整的問題 (不去調整, 也絕對不會離譜)。相對於一般圖形化 Office 類軟體, 要眞正把他的內容熟悉, 恐怕也是不簡單的。另外的問題, 大概是幕前, 幕後系

[4] http://tug.ctan.org/
[5] http://www.lyx.org/
[6] http://www.texmacs.org/

統操作習慣的問題, 甚至是一種第一印象了, 就好像說到電腦, 有不少人的腦子裡就會浮現 MS Windows 的圖象一樣。

- LaTeX 是專爲科學家或數學家而寫的?
  的確, 當初 Knuth 教授是爲了表達精確, 品質優美的數學式子而開發 TeX 的, 但由於 TeX 的彈性, 使得在其他的領域的使用者也爭相使用, 已經不是局限在學術界在使用了。尤其 XML 的興起, 需要一個適合的格式化工具 (formatter) 的配合, TeX / LaTeX 就剛好稱職的做他排版專業的工作。

1.3 本文的重點方向

TeX / LaTeX 的相關議題: 版面的配置, 排版規範, 字型技術, TeX the program 的改進, 繪圖技術, TeX macro 的撰寫, pre/post 處理程式的撰寫, 出版流程 等等, 浩瀚無涯, 可以說窮一輩子也可能研究不完, 所以, 各位在「陷入」這個領域之前, 建議最好有個適當的範圍, 以免「愈陷愈深」終至無法自拔。

所以, 本文的重點是放在「標準」LaTeX 本身, 其他相關的 package/macro 除非必要, 盡量不提及。但是 LaTeX 本身也不是十全十美的, 所以, 有需要的地方也需要一並提及外來的 macro, 無論如何, 重點是放在標準 LaTeX 本身, 請閱讀本文的朋友注意一下這個方向。LaTeX 本身就能解決的, 就不假外求了, 雖然會失去了一些彈性, 但爲免造成 package/macro 滿天飛, 打亂學習陣脚, 剛開始也只好如此了!

而且, 可能的情形下, 會往一般用途的方向去介紹, 而不僅僅專注在數理排版。數學式子雖是 TeX / LaTeX 的拿手把戲, 但不代表一般用途就不適合, 相反的, 現在有許多商業公司正把 TeX / LaTeX 用於一般商業出版上。

1.4 cwTeX 排版系統

cwTeX 的功能是將中文字轉換成 TeX 格式。它本身並無排版的能力, 只是把文稿內之中文字轉換成 TeX 的格式, 再交由 LaTeX 排版。在一般的電腦術語中, cwTeX 稱爲前階處理程式 (preprocessor)。cwTeX 程式與 LaTeX 系統構成一可排版中文之系統, 簡稱爲 cwTeX 排版系統。

1.5 Word 與 LaTeX 的比較

比較排版軟件, 一是要看它能做什麼, 二是看使用是否方便。

從基本的排版功能上看, Word 和 LaTeX 相差不大: 頁面設置, 紙張大小, 版心尺寸, 字型字號, 行距段距, 段首縮進, 居中對齊, 頁眉頁脚, 頁碼編排, 脚注尾注, 多級項目編號, 章節自動編號, 目錄自動生成, 交叉引用, 生成索引等等這些功能二者都差不多。LaTeX 在管理參考文獻方面要強一些, 另外 LaTeX 可以很方便地做 "邊注 (margin note)" 。LaTeX 和 Word 都能做圖文混排, 由於 Word 是所見即所得, 在調整圖形的位置及大小時要方便一些。以上這些功能對作者來說是夠用了。

用 LaTeX 排版的感覺就好像用文本編輯器寫 HTML 文件 (或者只用文本編輯器寫 GUI 程序), 一方面你可以精確地控制每個細節和總體結構, 另一方面在排圖形表格較多的文檔時明顯不如可視化工具方便。而用 Word 的感覺恰好相反, 你不能直接看到數據的底層表示, 只能通過上層的鼠標鍵盤操作來控制文檔。在 Word 中把一段文字設成藍色, 又把它設成紅色, 最後改回黑色, 那麼據我觀察 Word 多半不會聰明到去掉這些相互抵消的屬性, 而會留下類似 HTML 中 文字這樣的冗餘標記。

LaTeX 和 Word 都不適合處理過於複雜的版面, 例如報紙, 招貼畫, 複雜的分欄等等, 畢竟那是專業排版軟件的生存空間。個人感覺 Word 和 LaTeX 適合作者用來排版技術書籍和文章, 而 LaTeX 排的版面總體上看更樸素一些, 但細微處處理得更好。

給 Word 添加字體不費吹灰之力, 只要把字體文件 (font files) 拷貝到 $Windows\backslash Fonts$ 就行了; 爲 LaTeX 安裝新字體可是件麻煩事, 不是幾步就能搞定的。

Word 在多人協作方面能力更強一些, 例如可以追蹤修訂。另外 Word 排表格比 LaTeX 要方便一些。

從排版效果看,Word 和 LaTeX 可謂各有所長。(La)TeX 排數學公式的能力天下第一,Word 自然是沒法比。在處理英文 (西文) 文獻時, LaTeX 知道在單詞的音節之間斷開分行, 排出來的版面顯得比 Word 勻稱。在處理中文版面方面,Word 比 LaTeX 體貼一些, 特別是中文標點和拼音的處理。

排版軟件的選擇是個見仁見智的問題。作者不是 LaTeX 專家, 也不是 Word 專家, 犯不著跟自己較勁說非得用哪個軟件不可。哪個順手就用哪個, 不方便就換另一個唄。工具爲人所

用, 人不要爲工具所累。對於一般的技術文檔 (雜誌稿件等) 我一般用 Word 排, 長文檔 (畢業論文) 或者有數學公式的文檔 (某些實驗報告) 用 LaTeX 排。請保持一個開放的心態, 不做軟件的擁躉。

第 2 章
TEX / LATEX 語法概說

這一章要談的是, 和一般的純文字文稿及其他 markup 式文件系統在語言上的一般差異性。爲了讓觀念上能夠更清楚, 以下所述主要是要在命令列執行的, 致於編輯器上方便的按鍵及巨集, 這裡就不多談, 一方面是每個人使用的編輯器不一樣, 二方面是要先把黑盒子拿掉, 整個處理流程才會有概念。

當然, 由於完全還沒有開始實際寫文稿來測試, 所以, 這章是紙上談兵, 不必動手, 用看的就好。但, 別急, 我們會在第 3 章開始實際玩看看, 請別忘了, 到時要再回頭來複習一下這些資料。

而且, 前面已經說過, 這篇文章主要是著眼於 LATEX 所附上的巨集, 一些其他方便的套件引用, 將會在最後或另文再來談。其實, 不引用任何外來特殊套件, 讓 LATEX 本身去處理, 最起碼也就不會太離譜, 要講求美觀, 微調, 個人是認爲先把基礎弄起來再說, 有些套件的複雜程度, 會令人頭疼, 你是不是眞有這個需要, 值得考慮。而且, 很多時候自認爲不錯的「微調」, 其實常常會不合排版的慣例。TEX / LATEX 的語法, 可以是很簡單明白, 也可以是相當的複雜, 這是 TEX 系統本身的彈性所導致。

2.1 LATEX 的特殊專用符號

在 TEX / LATEX 的世界, 原始文稿都是純文字檔, 任何一種編輯器都可以打開來編輯, 觀看。而排版指令通常是由反斜線 (\, backslash) 所開頭來引導。註解則是由百分號 (%) 來引導。例如, 以編輯器編輯下列文字:

```
This is my first \LaTeX\ typesetting example.
```

編譯後會變成以下的結果:

This is my first LaTeX typeset ting example.

其中的 \LaTeX 就是 LaTeX 的一個指令, 會顯示 LaTeX 這個特殊的圖示。

由於, 西方國家的語系, 通常字母、符號的最大容量只有 256 個 (2^8), 因此, 許多現有的符號必須拿來當做控制指令, 才能符合排版的多樣化需求。以下的符號, 接觸 TeX / LaTeX 的朋友, 可能都得時時留意, 不要未經處理就直接寫進文稿裡頭去了。

通常, 編輯器的語法顏色會幫助判斷語法是否正確, 但不是都能完美無缺, 有時還是會漏掉, 這時別忘了查看一下 *.log 檔案, 例如: 編譯 your.tex 檔的話, 他的 log 檔就是 your.log。

表 2.1: 以指令方式輸入之英數符號

符號	作用	文稿上使用	LaTeX 的替代命令
\	下排版命令	$\backslash$	\ textbackslash
%	註解	\%	NA
#	定義聚集	\#	NA
~	產生一個空白或防止分割字串	\~{ }	\ textasciitilde
$	進入 (離開) 數學模式	\$	\ textdollar
_	數學模式中產生下標字	_{ }	\ textunderscore
^	數學模式中產生上標字	\^{ }	\ textasciicircum
{	標示命令的作用範圍	\{	\ textbraceleft
}	標示命令的作用範圍	\}	\ textbraceright
<	數學模式中的小於符號	$ < $	\ textless
>	數學模式中的大於符號	$ > $	\ textgreater
|	OT1 編碼, 數學模式中才能正確顯示	$\|$	\ textbar
&	表格中的分隔符號	\&	NA

2.2 LaTeX 排版上的一些規範或慣例

除了上面所談到的特殊符號外, 也有一些規範或慣例要遵守, 有些是比較硬性的規定, 有些則只是慣例, 可能不同的國家, 語言會有不同的慣例, 暫時先把他當成是 LaTeX 的遊戲規則就成了。

2.2.1 一般性的遊戲規則

1. LaTeX 的指令都是大小寫有別的, 由 \ 開頭, 後接由字母組成的字串或單一的非字母字元。其中由 [ ] 中括號括住的是選擇性參數, 可以省略, 由 {} 大括號括住的是不能省略的參數, 當然, LaTeX 的指令不一定會有參數, 但絕大部份都會有參數, 只不過把他給省略使用預設值罷了。

2. LaTeX 文稿中, 空一個英文空白和空多個英文空白的作用是一樣, LaTeX 會認作一個英文空白。

3. 平常我們編輯純文字檔, 按個 Enter 鍵, 就代表換行, 但實際排版出來, 一行的寬度是按照排版版面的設定, 也就是說, 你在文稿中按 Enter, 不代表排版後就是從這裡斷行, LaTeX 會依一行應有的寬度經過整體計算後自動補成一整行後再來斷行, 而且會在中間自動補足一個空白。這在英文很自然, 稱爲字 (word) 間空白, 但中文則不一樣, 在編輯器中編輯中文, 隨意按 Enter 的結果, 會造成文章中的中文間出現空白。這會在本文中適當的時機, 提出解決的方法。

4. 編輯器中, 多按幾次 Enter 就多空出幾行, 但在 LaTeX 文稿裡, 多個空白行, 和一個空白行是一樣的作用, LaTeX 會把他認作是一個空白行。而這個空白行, LaTeX 同時也會認作是新段落的開始, 所以 LaTeX 是以空白行來分隔各個段落。

5. LaTeX 預設每個章節的第一個段落的第一行是不內縮 (noindent), 從第二個段落開始才會內縮 (indent)。當然, 這是可以更改的, 往後會再提及。

6. LaTeX 的指令, 是從反斜線後第一個字母開始, 到第一個非字母符號爲止 (包括空白、標點符號及數字)。因此:

```
This is my first \LaTeX typesetting example.
```

這樣的話, 實際結果, 因爲 \LaTeX 後的空白是屬於指令的一部份, 空白將不會被解釋, 這樣會印成:

This is my first LaTeXtypesetting example.

這種結果, LaTeX 和 typesetting 連在一起了。要避免的話, 就要指定指令的作用範圍, 例如以下的大括號。或就眞的加個空白, 例如 \ , LaTeX 碰到 \ 就會形成完整的指令, 其後的空白就會被眞正解釋爲空白了:

```
This is my first {\LaTeX} typesetting example.
This is my first \LaTeX{ } typesetting example.
This is my first \LaTeX\ typesetting example.
```

所以, 正常印出來應該是:

This is my first LaTeX typesetting example.

7. 中英文混合的時候, 通常, 英文字前後都會留個空白, 以便和中文區隔開來, 只是這個空白要多大, 這就沒有固定的慣例, 通常留個英文空白也是可以, 要講究的話, 等談到中文排版相關議題時再來討論, 目前就養成習慣, 英文單字前後留個英文空白。

2.2.2 針對標點符號的遊戲規則

1. 中英文的引號不一樣, 這裡請特別注意, 許多人常常搞錯。中、英文引號不管單雙都要分左右。英文的話, 左邊的引號是 grave accent, 是鍵盤左上方 Esc 或 F1 下方有波形號的那一個鍵; 右邊的是 apostrophe, 也就是鍵盤左邊 Enter 鍵隔壁的那個鍵。雙引號的情形是鍵入兩次的左單引號及兩次的右單引號, 而不是用¨這個一次完成兩個點的 ditto marks。所以, 實際上在鍵入文稿時是:

```
``This sentence.¨
¨This sentence.¨這是錯誤示範!
```

排版出來的情形是:

> “This sentence.”
> ”This sentence.” 這是錯誤示範!

2. LaTeX 會在英文文章的一個句子結束和另一個句子開始的中間, 自動調整成較大一點的空白, 這可以增加文章的易讀性。所謂一個句子結束, 例如: 句點 (.), 問號 (?), 驚嘆號 (!), 及冒號 (:), 這當然是指英文的半形標點符號, 不是中文的全形標點符號。

   現在的問題是, 如果這些標點符號後面不是另一個句子的開始的時候, LaTeX 無法去判斷這種情形, 這時得由我們自己自行判斷, 處理了。例如英文縮寫字:

   ```
   I am Mr. Sean G.J. Wu, G.J. is a abbreviation of my name.
   I am Mr.~Sean G.J. Wu, G.J. is a abbreviation of my name.
   I am Mr.\ Sean G.J. Wu, G.J. is a abbreviation of my name.
   ```

   其中 Mr.\ Sean 的寫法, 和 Mr.~Sean 幾乎是一樣的, 都是強迫插入一個比較小的正常單字間空白, 差別在於後者也另外表示不可以從這裡換行, 通常用在人名的時候, 讓他們不致中斷, 一般在人名的排版, 包括他的頭銜, 職稱, 是不中斷成兩行而分開的。而且整個文句較長的話, 以後者較恰當, 才不會因爲斷行被分成兩半, 這個 ~ 符號也因此在 TeX 的專有名詞, 就稱爲 tie, 把他們綁住的意思。排版出來的時候會變成:

   ```
   I am Mr. Edward G.J. Lee, G.J. is a abbreviation of my name.
   I am Mr. Edward G.J. Lee, G.J. is a abbreviation of my name.
   ```

   請放大去仔細比較一下結果就知道了。第二行的才是正確的, Mr. 和 Sean 之間的空白是正常單字間空白, 比第一行的句子結束空白要小一點點。其他有使用到縮寫字的場合, 例如:‘Dr.’, ‘etc.’, ‘e.g.’, ‘i.e.’, ‘vs.’, ‘Fig.’, ‘cf.’, ‘Mrs.’, 這些都不是代表句子結束, 所以, 要插入一個正常空白。

   那 G.J. 後面爲什麼沒有插入正常空白? 那是因爲, J 是大寫的, 這時 LaTeX 不會去誤認爲是句子結束, 通常句子結束時的句點前的那個字母是小寫的。Well, 有沒有覺得有點道理? :-)

3. 刪節號中文英也是不同, 英文是三點, 如果碰到句點的話, 則是四點。中文的話是六點, 碰到中文句點很容易就分得清楚。但是英文這個三點, 不是就打個三個句點了事, 這樣的點太密集, 可以使用 \ldots 或 \dots 指令, 例如:

```
I'm not a good man ..., but a good husband .... 錯誤示範!
I'm not a good \ldots\ man \ldots, but a good husband \ldots.
I'm not a good \dots\ man \dots, but a good husband \dots.
```

排版出的來結果是:

I'm not a good man ..., but a good husband .... 錯誤示範!
I'm not a good man . . . , but a good husband . . . .
I'm not a good man . . . , but a good husband . . . .

4. 破折號。在英文, 相當於破折號的可能有三種:

   ◇ hyphen

   這是最短的 dash, 通常就是鍵入 - 就行了, 例如 father-in-low, 這樣會表現成 fater-in-low。

   ◇ en-dash

   這是最常用的破折號, 是鍵入兩個 hyphen。例如 1991 - - 2003 年, 這會表現成 1991–2003 年。

   ◇ em-dash

   這是最長的 dash, 由三個連續的 hyphen 組成, 應該是最相近於我們中文所說的破折號。例如 I am - - - a good man. 會表現成 I am—a good man.。至於這個三個連續的 hyphen 前後是否要留空白, 都有人使用, 並沒有硬性的慣例, 但爲了和中文的破折號配合 (中文破折號前後, 通常不留空白), 個人通常是不留空白的。

   ◇ 眞正的減號

   這應不能算是破折號, 而是實際的減號或負號, 這要進入數學模式, 例如: 負五,

要寫成 -5 , 然後表現出來是 -5。這也常常會有人搞錯, 不能直接鍵入一般的負號那個鍵來充數, 這是因爲 TEX / LATEX 的數學式子的用字和間隔處理, 和一般內文不同的關係。

2.3 LATEX 的文稿結構

2.3.1 環境 (environment)

上一節所談的都是指令, 雖然也可以由大括號來定作用範圍, 但如果是一整段, 甚至是一整篇文章都要作用時, 那指令可能就不很適合了, 因此, LATEX 也有一種巨集結構, 稱爲環境 (environment), 主要是讓作用範圍能擴大至較大的範圍。

所有的環境, 都是起於 \begin{環境名稱}, 止於 \end{環境名稱} , 這兩個指令之間的文稿都會被作用, 而且, 環境之內還可以套用其他不同的環境。

LATEX 文稿的內文, 其實就是包在一個 \begin{document} 和 \end{document} 這個 document 環境當中。

2.3.2 最簡單的 LATEX 文稿結構

以下就是所有 LaTeX 必需具備的文稿大結構:

```
\documentclass{article}
這裡是 preamble 區
\begin{document}
這裡是本文區
\end{document}
```

\documentclass{article}, 這是在告訴 LATEX 使用哪一種類別, 我們目前使用的是 article 類別, 關於類別會在下面幾章討論。preamble 區, 則是下一些會影響整個文稿的指令, 及引用巨集套件的地方, 當然, 完全不引用巨集, 也不使用影響全文的指令的話, preamble 區就是空白, 不寫任何東西。本文區, 就是我們實際上寫文章的地方。

2.3.3 preamble 區可以放些什麼?

這裡可以引用巨集, 而且會影響整篇文稿的指令, 例如一些事先定義好的指令, 想在整篇文稿

中使用, 就可以置放在 preamble 區。

2.3.3.1 巨集的引用

本文主要是標準 LaTeX , 但前面已提到, 會有些巨集套件不得不要引用, 底下就來說明如格引用套件。這些套件都是一般 TeX 系統都會附上的。指令及環境要如何開頭都介紹過了, 現在來看看引用巨集要怎麼開頭。

```
\documentclass{article}
\usepackage{color}
\begin{document}
\textcolor{blue}{This is blue color.}
\end{document}
```

編譯一下, 看看結果是什麼? 這裡使用的就是 color package, 裡頭是由 TeX / LaTeX macro 所寫成一個巨集套件。一般簡單的我們就稱爲巨集 (macro), 複雜一點的就稱爲巨集套件 (package), 其實, 裡頭都是一樣的, 只不過大小及有沒有整理成一個系統的差別。

LaTeX 裡頭有什麼現成的套件可以使用, 每個散佈的 TeX 系統所收集的可能都會有所不同。大概沒有人可以精通所有現存的 LaTeX 巨集套件, 因爲實在是太多了。

2.3.3.2 影響整篇文稿的指令

會影響整篇文稿的指令, 通常也是放在 preamble 區, 例如:

```
\linespread{1.36}
\parindent=0pt
```

\linespread 是在控制上下行的行距, 這裡就是將行距變成原來的 1.36 倍。至於什麼是行距呢? 就 是這一行的基線 (baseline) 到下一行的基線的距離, 通常英文文章不必去調整他的行距, 但中文得適當加大行距以利閱讀。

\parindent 是調整段落內縮的程度, 這裡調整成 0 , 也就是說各段落都不內縮的意思, 也可以調整成其他的值, LaTeX 就會依這個值去內縮。當然, 不去設定的話, LaTeX 就會依他的預設值去內縮。

2.3.4 章節結構

本文區當然是我們寫文章的主要地方, 及一些微調。在 LaTeX 的文稿裡頭, 章節標題的形成都是由同樣的指令來控制的, 這樣有一個好處, 臨時插入章節標題及其內文時, 我們不必去理會標題編號及目錄的問題, 也不必去理會要用什麼字型, 及字型大小要多大, LaTeX 會自動計算處理, 字型大小也會和內文使用的字型大小互相配合調整, 使用者就專心在內文構思, 寫作即可。以下由列表來瞭解整個章節結構:

表 2.2: 段落深度表

深度標號	指令	作用及注意事項
−1	\ part{}	這是最大的結構, 我們中文通常稱爲「部」。
0	\ chapter{}	章。在 article 類別裡頭沒有章。
1	\ section{}	節。
2	\ subsection{}	小節。
3	\ subsubsection{}	次小節。
4	\ paragraph{}	段落。
5	\ subparagraph{}	小段落。

章節標題的內容就是直接寫入指令的大括號裡頭就可以了, LaTeX 在排版時會自動使用粗體、加入章節編號及納入目錄裡頭。

至於第一欄的深度標號 (secnumdepth), book / report 類別的深度標號是 2 , article 的是 3 。這是什麼意思呢? 就是說 book / report 類別的文稿, 在 *subsection*{ } 以後(subsection 本身仍會編號), 章節就不再編號了; 同樣的, 在 article 類別的文稿, 在 *subsubsection*{ } 以後就不編號了。但仍然會獨立出一單獨行來表示這個是標題。不編號了的章節內容, 當然也就不納入目錄裡頭了。這當然是可以更改的, 只要更改 LaTeX 的 secnumdepth 這個變數的值就可以了, 這個往後會提及如何更改 LaTeX 的預設值。

第 3 章
排版: 學會控制空間

這一章要談的是排版。一篇文章, 從無到有, 除了文字之外, 圖或者表才是排版的重點。但不急, 只要你學會如何控制空間, 你已經快要學會排版了。

3.1 換行

每行後面加上 \\ , 就表示要強迫換行, 且會在你加入的地方強迫換行。\\ 之後也可加上參數, 如: \\ [0.3 cm] 。此即換行距離控制在 0.3 cm

3.2 縮排

- 在第一行之前加入 **\noindent** 來指示 LaTeX 不要去縮排。但是這只作用在下指令的地方, 其他該縮排的地方還是會縮排。
- 在 preamble 區加入 **\parindent=0pt** , 這表示讓全文的縮排爲 0pt , 當然, 這就表示全文都不要縮排了。

3.3 加入章節標題

在 LaTeX 裡頭, 要加入章節標題實在是太容易了, 也不必去管字體的大小及置放的位置, 盡管加上去就對了! LaTeX 會替我們安排一切。表 2.2 當中的指令就是加入章節標題的指令。

3.4 加入 title page 資訊

這是指內頁的第一頁, 在 LaTeX 裡頭, 我們就稱爲 title page 。在 LaTeX 的標準格式裡, 他包括了標題(title)、作者名字(author)、日期(date)及感謝詞(thanks) 。要注意的是, 在 report / book 類別, title page 是自成一單獨頁的, 但在 article 類別裡, 他是和本文連起來的。

```
\documentclass{article}
\title{ABC}\\
\author{吳尙樺}\\
\date{2006.3.7}\\
\begin{document}
這裡是本文區
\end{document}
```

我們可以發現, 這一頁是不編頁碼的, 從下一頁開始才是第一頁。作者可以有多個, 使用 **\and** 指令來連接。日期不一定要有, 如果沒有 **\date{\today}** 這個指令, 那還是有日期, 但只能固定在今天。如果內容過長, 他會自動折行, 但也可以手動加 \\ 來強迫換行, 不管如何換行, 整個句子是居中排列的。**\maketitle** 是下在本文區的開頭, 如果不下這個指令, 那編譯時不會有什麼錯誤, 只是就沒有 title page 了。

3.5 加入目錄 (Table of Contents)

加入目錄 (Table of Contents) 對 LaTeX 而言, 更是輕而易舉的事情, 只要在本文開頭加個 **\tableofcontents** 指令就成了! 這裡千萬要注意的是, \tableofcontents 要加在 \maketitle 的後面, 否則目錄會印在 title page 之前。而且要編譯兩次。第一次產生 ???.toc , 然後第二次編譯再跟據這個 toc 檔, 眞正編入目錄。

3.6 加入摘要 (abstract)

這不一定會有, 如果要加入的話, 可使用 abstract 環境, 在這個環境中的文章, 左右會縮排。要注意的是, 只有 article / report 類別才有 abstract, book 類別不能使用這個環境。

report 類別的摘要自成一頁, 不編頁碼, 且不會編入目錄中, 這和一般的論文格式可能會不一樣, 使用時請注意。article 的類別則仍然是和本文相連的, 會出現在文章標題之後。

```
\documentclass{article}
\title{ABC}\\
\author{吳尚樺}\\
\date{2006.3.7}\\
\begin{document}
\maketitle
\begin{abstract}
這裡是摘要內容。
\end{abstract}
\tableofcontents
\chapter{這裡是 chapter }
\section{這裡是 section }
\end{document}
```

abstract 和 summary 在較正式的論文是有區分的, 通常 abstract 在文前; summary 則在文後。但目前一般性的文章則沒有這樣區別, 通通當成「摘要」。通常, 摘要裡頭是不用註解、無交互參照也不使用公式圖表的。

3.7 加入註解: 腳註 (Footnote), 邊註 (Marginal note)

在 LaTeX 裡頭, 註解可有兩種方式, 一種是脚註 (footnote), 一種是邊註 (marginal note)。通常 LaTeX 的脚註預設是由阿拉伯數字在編號, 置於頁底部。在沒有部 (part) 的情形下, report / book 類別, 編號每章會從頭起算, article 類別則會連續, 而且, 會使用 footnotesize 的字體印出。邊註則不編號, 字體是正常大小。脚註使用 **footnote**{} 指令即可, 邊註使用 **marginpar**{} 。

3.8 調整字族、字型系列、字形的指令

以下的表一目了然。

	字型	標準指令	宣告式指令（環境）	舊用法
字	textup	\textup{textup}	{\upshape textup}	
形	*italic*	\textit{italic}	{\itshape italic}	{\it italic}
	slant	\textsl{slant}	{\slshape slant}	{\sl slant}
	SMALL CAPS	\textsc{small caps}	{\slshape small caps}	{\sc small caps}
系	medium	\textmd{medium}	{\mdseries medium}	
列	**boldface**	\textbf{boldface}	{\bfseries boldface}	{\bf boldface}
字	roman	\textrm{roman}	{\rmfamily roman}	{\rm roman}
族	sans serif	\textsf{sans serif}	{\sffamily sans serif}	{\sf sans serif}
	typewriter	\texttt{typewriter}	{\ttfamily typewriter}	{\tt typewriter}

圖 3.1: 調整字族、字型系列、字形的指令

3.8.1 相對字型大小的調整

指令	實際例子	效果	實際的大小（點數）
\tiny	{\tiny tiny}	tiny	5pt
\scriptsize	{\scriptsize scriptsize}	scriptsize	7pt
\footnotesize	{\footnotesize footnotesize}	footnotesize	8pt
\small	{\small small}	small	9pt
\normalsize	{\normalsize normalsize}	normalsize	10pt
\large	{\large large}	large	12pt
\Large	{\Large Large}	Large	14.4pt
\LARGE	{\LARGE LARGE}	LARGE	17.28pt
\huge	{\huge huge}	huge	20.74pt
\Huge	{\Huge Huge}	Huge	24.88pt

圖 3.2: 相對字型大小的調整

3.8.2 絕對字型大小的調整

通常字型的大小, 使用上一節所說的相對字型大小來調整會比較方便, 而且對於整個版面的配合也會比較恰當, 例如行距也會跟著做適當的調整, 如果自行用絕對字型大小的方法來調整字型大小的話, 常常會造成行距不一致的情形, 因此, 如非必要, 應盡量避免。

但有時候就是需要做這樣的調整, 例如本文封面的字型大小, 縱使是 LaTeX 預設的最大字型也覺得稍小了點, 這就要另外引入 package 調整了。

這裡我們使用 type1cm package 來調整。當然, 得使用 Type 1 字型, 才可以達到無段放大、縮小的目的, 而這個 package 也是配合 Type 1 字型使用的。個別放大的 pk 點陣字, 這裡就不討論了, 目前絕大部份的 Computer Modern 字型都已有 Type 1 的 free 版本, 而且各個 TeX distribution 都會附上, 使用上會較方便。以下是 type1cm 的使用方法:

```
\usepackage{type1cm}
這裡是文字內容。
\fontsize{字型大小}{行距大小}\seclectfont
```

其中的「字型大小」就是所要指定的大小, 通常以 pt 爲單位, 當然, 要使用其他單位也是可以。「行距大小」也是要一併指定, 不可省略。最後的 **\seclectfont** 是讓他發生作用的意思, LaTeX 有些關於字型的較低階指令, 要下 \seclectfont 後才會作用, 例如 \fontsize{}{}

第 4 章
空間與位置

4.1 版面大小

我們對於所能控制的一整張紙的範圍都可以稱爲版面。當然, 我們的內文(body) 並不是佔滿整張紙的範圍, 上下左右都會留有一定的空白。小時候在宣紙上練習寫毛筆, 老一輩的都會要我們留「天地」, 這就是指內文四周的空白, 除了視覺上的理由, 大概也是人生的哲理吧? :-)

在編輯上, 也有人稱內文 (body) 的部份爲「版心」或「版口」, 四周的空白部份, 則稱爲「版邊」。突破版心、版邊的設計, 就稱之爲「出血」, 例如, 以背景圖佈滿整張紙當做是背景的場合, 以這個背景圖而言, 就無所謂版邊了。但這在 LaTeX 通常是不會有這種情況出現, 除非特意去指定內文和紙張大小同樣範圍。

當然, 在內文以外的空白, 也並非全是空白, 他包含了頁足 (footer), 頁眉 (header) 及邊註 (marginal note) 的部份, 記載關於頁數、註解等資訊。這裡所謂的紙張大小, 指的是 paperwidth 和 paperheight 所圍成的範圍, 並非實際上手上拿到的紙張大小, 實際在手上的紙張通常會略大於我們這裡的所謂紙張, 所以, 正式列印時, 還需做微調或截切才會是眞正的這裡所謂的紙張大小 (版面大小)。

如果不指定紙張的話, LaTeX 預設會使用美式 letterpaper 的大小, 如要使用歐、日式的 a4paper 的話, 要另行指定。我們可以稍微看一下 LaTeX 預設是如何安排版面空間的。其中 Header(頁眉)、Footer(頁足) 及邊註的空間是不含括在內文 Body 裡頭的, 這裡是只是單面的圖, 如果是雙面的話, 那偶數頁和奇數頁的邊註是要左右對換的, 也就是說這個圖是奇數頁, 偶數頁的話, 邊註是在左邊。

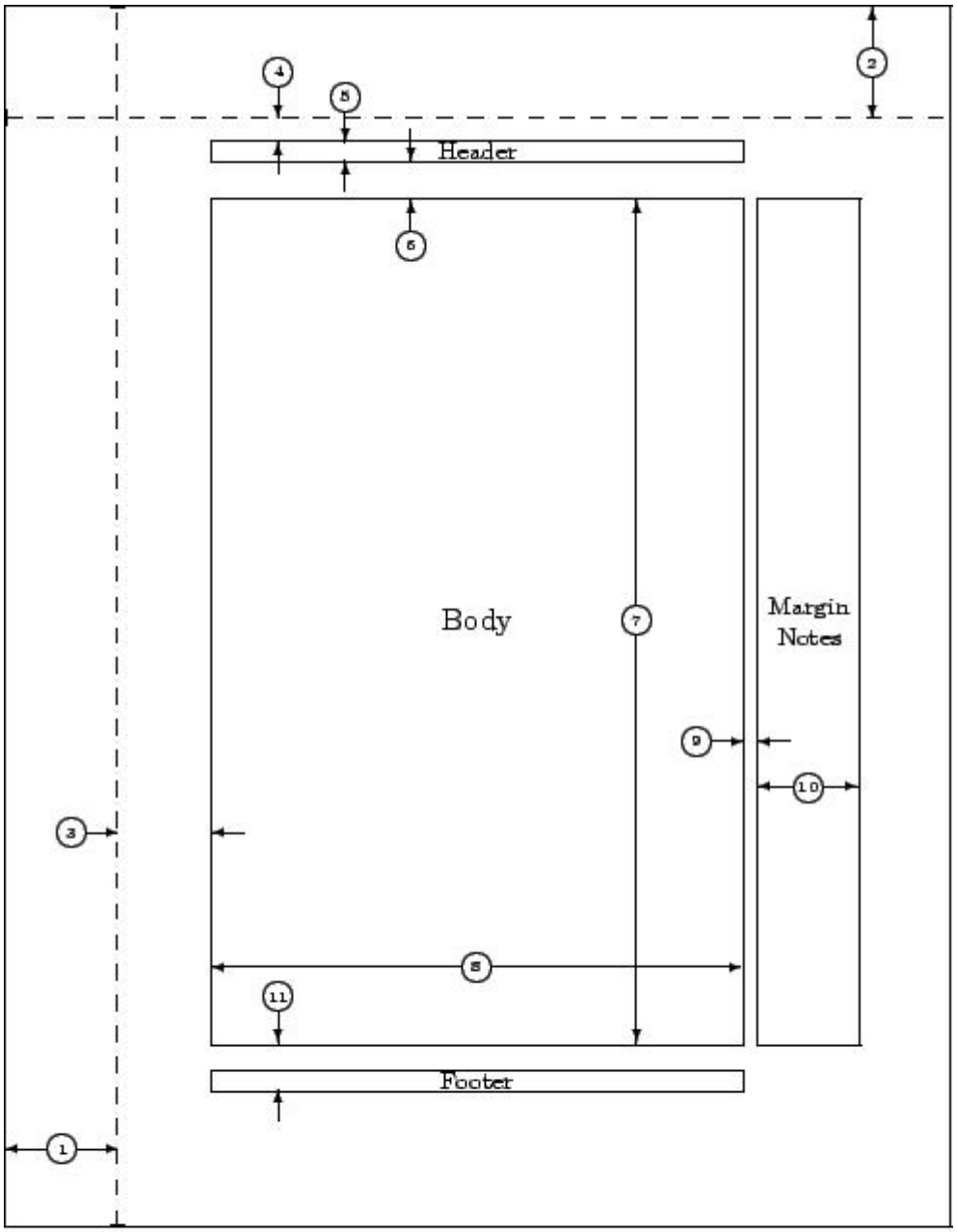
Header
Body
Margin
Notes
Footer
1 one inch + \hoffset
2 one inch + \voffset
3 \oddsidemargin = 62pt
4 \topmargin = 16pt
5 \headheight = 12pt
6 \headsep = 25pt
7 \textheight = 550pt
8 \textwidth = 345pt
9 \marginparsep = 11pt
10 \marginparwidth = 65pt
11 \footskip = 30pt
\marginparpush = 5pt (not shown)
\hoffset = 0pt
\voffset = 0pt
\paperwidth = 614pt
\paperheight = 794pt

圖 4.1: 版面配置圖

表 4.1: 版面設定指令表

指令 (值)	意義	指令 (值)	意義
\ paperwidth	紙張的寬度	\ topmargin	頁眉上方的空白
\ paperheight	紙張的高度	\ marginparwidth	邊註的寬度
\ textwidth	內文 (body) 的寬度	\ marginparsep	邊註與內文的距離
\ textheight	內文 (body) 的高度	\ marginparpush	兩邊註間距
\ headheight	頁眉 (header) 長度	\ oddsidemargin	內文左邊的空白大小
\ headsep	頁眉與內文間的距離	\ hoffset	微調版面在實際紙張的左右位置
\ footskip	內文底至頁足底之距離	\ voffset	微調版面在實際紙張的上下位置

4.1.1 紙張大小

紙張	大小	紙張	大小
a4paper	21x29.7cm	letterpaper	8.5x11in
a5paper	14.8x21cm	legalpaper	8.5x14in
b5paper	17.6x25cm	executivepaper	7.25x10.5in

\documentclass[12pt,a4paper]{report}

所以, 這篇文章使用的是 a4paper, 內文字型的大小是 12pt。方括號的參數是選項, 可以省略, 如果省略的話, 預設值就是 **10pt/letterpaper**。

4.2 調整橫向空間

表 4.2: 調整橫向空間的指令

指令 (值)	意義
\ hspace{單位}	向右空出某個度量單位的空白, 如果是負數, 則是向左
\ dotfill	作用和 \hfill 相同, 只是空白變成點
\ hfill	讓左右兩旁的文字往兩邊擴張至一個行寬爲止
\ hrulefill	作用和 \hfill 相同, 只是空白變成一橫線
\ quad	空出一個 em 單位的空白
\ centering	此指令以後的文字將會居中排列, 左右沿將不切齊
\ qquad	空出二個 em 單位的空白
\ raggedright	此指令以後的文字將會居左排列, 右沿將不切齊
\ thinspace	空出1/12個 em 單位的空白
\ raggedleft	此指令以後的文字將會居右排列, 左沿將不切齊
\ enspace	空出1/2個 em 單位的空白
\ centerline{}	將大括號內的文字居中排列

4.2.1 調整橫向空間的環境

\begin{center}...\end{center}	讓這個環境內的內容置中
\begin{flushleft}...\end{flushleft}	讓這個環境內的內容靠左
\begin{flushright}...\end{flushright}	讓這個環境內的內容靠右
\begin{raggedright}...\end{raggedright}	讓這個環境內的內容靠左, 右沿將不切齊
\begin{raggedleft}...\end{raggedleft}	讓這個環境內的內容靠右, 左沿將不切齊

進入環境, 和上一節提到的指令, 兩者有什麼不同呢? 最大的不同是, 這可以方便的指定一個範圍的文句讓他作用, 而不會影響環境以外的文句。其次, 進入環境, 縱使和上下行連在一起, 沒有空出空白行, 他也會自動的在上、下行空出個空白行出來, 使用指令的話則不會。

咦!這裡怎麼又有個 raggedright 及 raggedleft ? 原來他也是可以當環境來使用。由於這兩個指令會使以下的內容的左、右沿不切齊, 因此使用上要非常小心, 除非本來就想讓內文的左、右沿不切齊, 否則, 最好是使用有範圍限制的方式。當然, 如果這兩個指令是用在某個

其他環境範圍內, 他的作用也將僅限於這個環境內, 不會影響這個環境外的文句。

4.3 調整縱向空間

表 4.3: 調整縱向空間的指令

指令 (值)	意義
\ vspace{單位}	向下空出某個單位的空白(行), 負數則是向上
\ bigskip	產生 12pt(11-12pt) 的垂直空白 (行)
\ medskip	產生 6pt(5-7pt) 的垂直空白 (行)
\ smallskip	作產生 3pt(2-4pt) 的垂直空白 (行)
\ vfill	和 \hfill 類似, 作用是將某段落向上頂, 或往下擠
\ parskip 單位=	調整全文每個段落間的距離爲某個單位

其中的 \bigskip, \medskip, \smallskip 並非固定的, 他們會視上下文脈絡的需要自動做微調, 以達到一整頁較一致的空間配置。\vspace 如果是出現在一頁的第一行或最後一行時, 將會失去作用, 這時可以加個星號, \vspace*{單位}。

爲了維持版面的一致性, 使用縱向空間調整的指令時要特別留意, 例如章節標題上下的空間、各段落間的空間, 進入環境前後所空出的空間, 這都有一個固定值, LaTeX 會自動去調整, 不必由使用者自行動手, 除非是封面這種單獨頁。所以, 使用縱向空間調整指令時, 要非常注意整體的一致性, 這也是排版上的一個很重要的原則。

4.4 條列環境

條列環境也是屬於一種空間的控制, 他把一些文字按一定的方式來排列, 條列環境中一些起頭的符號、文數字或字串, 我們稱之爲項目標籤 (item label), 利用這些不一樣的排列位置及不一樣的項目標籤起頭來敍述文句, 就可以達到醒目的作用。這是以章節分隔以外, 相當常用讓內容一目了然的方法, 建議多多利用。請千萬記得, 環境中還可以有環境, 而且以下三種的條列方式可以混合交叉使用。

4.4.1 項目式條列環境 (itemize)

這是以符號來起頭醒目的一種條列方式。例如:

```
\begin{itemize}
\item 第一大項, 這裡是第一大項。
\item 第二大項, 這裡是第二大項。
\begin{itemize}
\item 第一小項, 這裡是第一小項。
\item 第二小項, 這裡是第二小項。
\end{itemize}
\item 第三大項, 這裡是第三大項。
\item 第四大項, 這裡是第四大項。
\end{itemize}
```

- 第一大項, 這裡是第一大項。
- 第二大項, 這裡是第二大項。
  - 第一小項, 這裡是第一小項。
  - 第二小項, 這裡是第二小項。
- 第三大項, 這裡是第三大項。
- 第四大項, 這裡是第四大項。

4.4.2 列舉式條列環境 (enumerate)

這是以數目字或字母或羅馬數字來起頭醒目的條列方式。同樣的例子, 改成 enumerate 的話, 會排版成:

```
\begin{enumerate}
\item 第一大項, 這裡是第一大項。
\item 第二大項, 這裡是第二大項。
\begin{enumerate}
\item 第一小項, 這裡是第一小項。
\item 第二小項, 這裡是第二小項。
\end{enumerate}
\item 第三大項, 這裡是第三大項。
\item 第四大項, 這裡是第四大項。
\end{enumerate}
```

1. 第一大項, 這裡是第一大項。
2. 第二大項, 這裡是第二大項。
   (a) 第一小項, 這裡是第一小項。
   (b) 第二小項, 這裡是第二小項。
3. 第三大項, 這裡是第三大項。
4. 第四大項, 這裡是第四大項。

4.4.3 敘述式條列環境 (description)

這是以一個簡短文字敍述來起頭醒目的條列方式, 這些文字要用方括號括住。當然也可用符號替代。

```
\begin{description}
\item[第一大項] 這裡是第一大項。
\item[第二大項] 這裡是第二大項。
\begin{description}
\item[第一小項] 這裡是第一小項。
\item[第二小項] 這裡是第二小項。
\end{description}
\item[第三大項] 這裡是第三大項。
\item[第四大項] 這裡是第四大項。
\end{description}
```

第一大項 這裡是第一大項。

第二大項 這裡是第二大項。

第一小項 這裡是第一小項。

第二小項 這裡是第二小項。

第三大項 這裡是第三大項。

第四大項 這裡是第四大項。

要注意的是, 不管哪一種的條列環境, 每個項目 (item) 的文字敍述會自動折行, 這相當方便, 使用者只要把條列的結構弄妥, 專心打每個項目的內容就成了。而且, 如果使用方括號括住一些字元、字串或符號, 那帶頭的標示將會是這些字元、字串或符號, 如果是列舉式的條列方式, 那麼有方括號的將不被編號, 會自動跳過, 編號順序則會自動順延。

第 5 章
LaTeX 的標準文稿類別

這章主要是在說明 LaTeX 文稿的類別 (document class), 這是 LaTeX 規範文稿整體結構的方法。使用 class 的用意, 就是把版面結構處理和實際文稿分開, 這樣的最大好處就是維持整篇文章排版結構上的一致性, 也使文稿內容更清爽簡潔, 使用者只要專心於文稿內容的寫作即可, 如果 class 定義的好, 也可以達到一文多變化又不變更文稿內容的目的, 只要把引用的 class 換成別的就可以了, 其他的可以不必更動。

目前, LaTeX 有五種標準類別用於一般文件, 可用於一般的書信、雜誌、期刊、報告及論文。但有些期刊、論文會要求一定的結構, 這時得依需求另行訂定。因此, 也有其他的類別存在, 標準類別並不是唯一的。甚至, 也可以自行撰寫自己的文稿類別。當然, 我們一般使用是不需要這麼講究, 這裡只介紹 LaTeX 的標準類別。而且, 如果有和他人交換文稿的需求時, 我們應該盡可能的使用流通性較廣泛的類別。

5.1 LaTeX 類別的宣告

LaTeX 的類別, 要在文稿的一開頭時就宣告 (當然, 其上有註解是沒有關係的), 他的一般格式如下:

```
\documentclass[選擇性參數]{類別}
```

選擇性參數是可以省略的, 但類別名稱則不能省, 一定要指定一個類別。而且只能只有一個類別。

5.2 類別的選擇性參數

選擇性參數可以選擇多個, 各個選項是以逗點分開的。

1. 10pt, 11pt, 12pt

   指定內文一般正常字的大小, 預設是 10pt 。其他點數沒有外來 package 的幫忙不能指定。

2. a4paper, letterpaper, b5paper, executivepaper, legalpaper

   指定紙張大小, 預設是 letterpaper。

3. fleqn

   使數學式靠左對齊, 預設是居中對齊。

4. leqno

   使數學式編號靠左, 預設是靠右。

5. titlepage, notitlepage

   決定 title page 是否獨佔一頁。預設 article 不獨佔一頁, 但 report / book 則會獨佔一頁, 在這裡是可以指定來變更預設行爲, 例如 article 文稿, 指定 titlepage 的話, 那 title page 就會獨佔一頁。

6. onecolumn, twocolumn

   文章單欄或兩欄式, 預設是單欄, 也就是不分欄。

7. twoside, oneside

   是否區分奇偶數頁。預設 article / report 不區分, book 則會區分。一般的書籍, 在裝訂的部份, 他的中央線稱爲書脊, 偶數頁會在打開書籍時的左方, 而且其中內容會偏向書脊的中央 (此時是向右), 反之, 奇數頁會在右, 內容一樣會偏左向書脊, 在 oneside 的情形則不做這樣的區分, 不管奇偶頁都會在紙張的中央部位。

8. landscape

   橫向列印或縱向列印, 預設縱向 (portait)。

9. draft

   草稿式編譯, 這時圖檔將不會被引入, 可加快編譯的速度。不過, 如果編譯是使用向量字型的話, 編譯速度應該是還算很快。但使用 draft 的一個好處是, 過長的地方會標示出來。

10. openright, openany

    這是在控制, 章的開始是否是奇數頁 (right-hand page)。在 book 類別, 預設章會從奇數頁開始, report 類別則不會。article 類別沒有章, 所以, 對此一設定會忽略。

5.3 類別的種類

這裡只列出一般文章使用的類別, 其他特殊巨集或 LaTeX 正式文件所使用的類別就不列出了, 一般使用, 這些類別就足夠了。

類別	一般用途	特性
article	一般短文	無章, 連續頁方式的安排, 無奇偶數頁的區分
report	較長論文	章會起新頁, 預設無奇偶數頁的區分
book	書籍類	章會於奇數頁起新頁, 預設有偶數頁的區分
letter	信件	英文信件格式
slides	幻燈片	幾乎另用外來套件取代
minimal	測試及寫新類別	這是最簡單的類別, 只規定了內文的寬、高, 正常字

當然, 這些用途並不是固定不變的, 得看使用者的安排, 不想多花時間、精神的話, 那就依 LaTeX 預設的格式去使用, 至少就不會太離譜。其中 minimal class 是用來測試用的, 或者寫新的 class 用的, 他完全沒有版面的安排, 例如, 沒有章節的結構, 各種間距也沒有定義, 預設的字型是正常字, 沒有其他的變化, 幾乎所有的變化要自行去定義。

第 6 章
表格的處理

這是屬於一般人覺得比較困難, 但卻是很重要的部份, 讓我們多花點時間研究。LaTeX 的表格, 因爲是抽象邏輯的思考方式來製作表格, 對一般使用者而言, 比較不容易轉換成直觀印象。我們先從 LaTeX 本身表格的結構理解起, 這樣使用其他的輔助工具時也會比較得心應手, 甚至沒有其他工具, 只要把握住表格的大結構, 製作表格就不會摸不著頭緒了。

6.1 表格的種類

表格的使用, 在文章上常常是必備的要件, 他有歸納及醒目的作用, 當然, 表格太多也是會喧賓奪主。通常, 我們中文的使用習慣, 表 格就是大方框內有小方框, 文字置於小方框內, 甚至某些小方框內還有斜線在分隔。爲了排版上的方便及視覺表現上的美觀、清楚, 在國際上大部份較正式的論文已不使用縱線、斜線, 表格通常由橫線來做區隔, 甚至完全沒有線條, 使用空間區隔的方式。這種趨勢幾乎在二十幾年前就已開始普遍, 只是國內的文件似乎還是很喜歡有縱、斜線在表格之中, 好像沒有一些框線層層包住就不像表格。如非特殊的表現上的需求, 我們應該朝簡化表格本身的方向走, 將重點置於表格的內容及表格的邏輯結構安排, 漂漂亮亮的表格外觀加上不當的內容配置, 個人覺得這是個失敗的表格製作。

另外, 等粗的雙線條, 可能也是得盡量避免, 通常粗細不等的外框雙線條有裝飾的作用, 因此, 如果文件是較正式的論文, 那就可能要避免, 如果是海報、DM 或要讓人們填寫的表格之類的, 那又是另外一回事, 這時封閉性的方框可能會有需要。這些規範只不過是一些慣例, 並非一成不變的, 得視文件的性質及使用場合來做變化, 一個大原則是, 如果是以文字敍述爲主的文件, 那麼, 表格本身如果比文字內容搶眼太多的話, 或許就要考慮簡化表格本身了。

6.2 tabbing 環境

這是 LaTeX 裡頭最基本的表格形式, 除非自行另外定義、繪製, 他並沒有方便可用的線條指令來區隔, 完全使用空間、位置的配置來顯示表格內容, 這時整個 tabbing 表格在 LaTeX 的地位並不是一個最小單位的 box , LaTeX 不會把整個表格當成一個單位來處理。所以, tabbing 表格是可以跨頁的, 他可以被分成兩半來處理。因此, 要和其他文字、圖表並排排版時, 得另外放進一個 box 中, 讓他自成一個 box 單位, 例如 \parbox 或 minipage 環境裡頭。

在 tabbing 環境中, 第一個列 (row) 是以 \ = 來標示 Tab 寬度來區隔欄位 (column), 這個寬度是由欄位裡頭的字串寬度所決定的。後續的每個欄位是由 \ > 這個符號來區隔, 每列尾要自行加上 \\ 來換行, 最後一行可以不必使用 \\ 換行。tabbing 的基本大結構是:

```
\begin{tabbing}
column1 \ = column2 \ = column3
item1 \ > item2 \ > item3
itemA \ > itemB \ > itemC
\end{tabbing}
```

對於欄位內文字的控制, tabbing 較不完備, 雖然 LaTeX 有提供 \' 讓這個符號之前的文字靠左, 及 \` 讓這個符號之後的文字靠右, 但實際運用, 可能不是使用者想要的結果, 因此 LaTeX 的表格, 主要還是以 tabular 環境較爲常用。但 tabbing 環境的好處是, 他不見得一定要用於表格的排版, 例如他也可以表現如條列環境般的另一種表現方式, 而且他可以跨頁排版。

6.3 tabular 環境

這大概是最常使用的表格形式, 可以很方便的畫線框。這種表格, LaTeX 是把整個表格當成一個單位來處理, 就像字母一樣, 因此他在版面的安排上是和一般的字母一般的處理, 所以, 這種表格不經特殊處理, 無法被分割成兩個部份來跨頁。

和 tabbing 環境的不同, 除了可以有線條之外 (tabular 環境, 當然也是可以完全沒有線條), 分隔欄位的符號是 &, 而且, 一定要指定欄內文字的置放位置, 欄內文字超出指定的寬度時, 會自動折行, 還有許多其他更細節的調整。

6.3.1 tabular 表格的基本結構

```
\begin{tabular}[t]{lll}
\ hline
column1 & column2 & column3
item1 & item2 & item3
itemA & itemB & itemC
\ hline
\end{tabbing}
```

其中 [t] 表示 top , 也可以是 b 表示 bottom , 或 c 代表 center , 這要在前後有文字相並排的時候才會顯現作用, 因爲 LaTeX 會把整個 tabular 表格當成一個字母單位, 所以可以和其他文字、圖表並排排版。這些參數的意思是和同行文字的對齊方式, top 是表格頂端和前後文字對齊, bottom 則是表格底部和前後文字對齊, center 則是和表格中央對齊。

換行的方式和 tabbing 環境一樣, 其中的 \hline 是畫一條橫線的意思, 連續兩個 \hline\hline 會畫雙橫線, 他本身會自動換行, 因此不必加上換行符號。其中 \begin{tabular}[t]{lll} 的 lll 是在指定各欄位內容在小方框內的置放位置, l 表示靠左 (left), r 表示靠右 (right), c 表示置中 (center)。在 {lll} 中加上 bar(|) 會畫縱線, 例如 {|l|l|l|} 這樣就會變成傳統的大方框、小方框的表格。而兩個 bar 就會畫雙縱線。

tabular 環境內尚可使用另一個 tabular 環境來製作更複雜的表格, 這在 tabbing 環境是不被允許的。

6.3.2 tabular 環境對欄位的調整

1. p{寬度}

   這裡的 p 指的是段落 (paragraph)。通常用於一個小段落的文字, 指定了寬度後裡頭的文字會自動折行, 而且這個段落的頂端會和其他欄位的頂端對齊。

2. @{文字、符號或指令}

   這可以作用在本欄的各個列, 讓他們都出現某個文字、符號或都在某個指令的作用下。

這個指令另外會同時將欄位間距縮成 0 , 置於首尾的話, 會有讓橫線和文字切齊的作用 (預設不會切齊, 橫線兩端會多出欄位間距的部份)。

3. \multicolumn{欄位數}{左右位置}{文字內容}
   跨欄排版, 例如一小段文字跨兩欄。左右位置可使用 lrc 之一。

4. \cline{a-b}
   畫某部份欄位的橫線, 其中的 a-b 指的就是要畫線的欄位數, 例如 \cline{a-b} 就是畫第二欄至第三欄的橫線。

5. \arrayrulewidth=單位長度
   調整表格線條的粗細, 預設值是 0.4pt 。使用方法:\arrayrulewidth=1.5pt 即可, 但要注意的是要在進入 tabular 環境之前設定好。

6. \tabcolsep=單位長度
   調整兩欄位的左右間距。請注意, 這個值是實際兩欄位間距值的一半, 預設是 6pt。使用方法和 \arrayrulewidth 一樣。

7. \doublerulesep=單位長度
   調整畫雙線時, 這兩線間的間距, 預設值是 2pt。使用方法和 \arrayrulewidth 一樣。

8. \arraystretch
   調整表格的上下行距。請注意, 這要由 \renewcommand 來重設, 因爲在 LaTeX 定義出的一個常數值, 而這個 \arraystretch 只是這些常數值的倍數, 我們要重新改變他才能改變預設倍數。

在 tabular 環境的參數中, 可能是取代原來的參數, 例如 p{寬度}。也可能是置放在原參數的前後, 如 @{文字、符號或指令} 。@{文字、符號或指令} 如果完全沒有加入任何參數, 那麼他的作用只是在去掉左右兩欄間距而已, 大家可以把有關 @{文字、符號或指令} 的部份拿掉, 試著再編譯看看, 仔細比較看有什麼不同。有些專業排版的專家建議把表格前後加個 @{文字、符號或指令} 去除突出來的橫線(實際上就是去除原有左右兩邊間距的部份)。如果 @{} 裡頭不是指令, 而是文字或符號, 那這個文字或符號會加在各欄文字內容的前或後。p{} 指令的使用時機是某一個欄位的文字比較多, 需限定欄位的寬度讓他自動折行。

6.4 表格線條粗細的控制(booktabs)

由前面幾節所述, 可以看得出來 LaTeX 表格巨集的功能稍嫌陽春了點, 對於一些特殊狀況可能會無法處理, 對於表格外觀要求較高的使用者也會感到不足, 雖然也可以自行去定義巨集, 但這樣一來不但可能有可攜性的問題, 而且也不是每個人都有時間去學習 TeX / LaTeX 巨集的寫作。我們試圖來看看有沒有其他的解決方式, 這裡不得不會提到一些外來的巨集套件, 但這些套件的使用相當的普遍, 幾乎可以忽略他的可攜性的問題。

我們前面曾學過 \arraryrulewidth 指令, 可以調整線條的粗細, 但是這無法各別調整線條, 每個在 tabular 表格環境內的線條會調整成一樣的粗細。booktabs 巨集套件可以很方便的達成這個目的。我們來看看這個提供了什麼方便的指令:

指令	功能
\toprule[線條粗細]	畫表格頂端的橫線
\midrule[線條粗細]	畫表格裡頭的橫線
\bottomrule[線條粗細]	畫表格底部的橫線
\cmidrule	指令某個欄位畫橫線, 取代原來的 \cline

使用方法和 tabular 環境差不多, 連環境名稱都一樣, 但可在指令後加個方括號來指定線條的粗細, 不指定的話, toprule 及 bottomrule 都會比中間的其他線條粗一點。上面的表格就是標準例子之一。其中 cmidrule 另有更進一步的功能:

\cmidrule[線條粗細](左右是否去邊){畫線欄位}

其中「畫線欄位」和 \cline 一樣, 指定欄位數即可, 例如 2-3。左右去邊要表明左 (l) 或 / 及右 (r), 也可由大括號指定要去掉多少 (預設 0.5em), 如:(lr{0.7em}){2-3}。

6.5 彩色表格 (colortbl)

彩色表格已經是很普遍, 但千萬要小心喧賓奪主的情況, 也別弄成了大花臉。因此, 淡色系可能會比較合適。

6.5.1 color 巨集套件

這是附在 LaTeX 工具組 graphics package 中的一個巨集, 使用上非常簡單, 只要把 color 巨集在文稿 preamble 區引上就可以使用顏色了。以下是常要用到的控制指令:

指令	功能
\color{顏色}	這會使用文章所有內容都使用這個顏色
\definecolor	定義顏色
\textcolor{顏色}{文字內容}	讓文字內容使用某特定顏色
\pagecolor{顏色}	這是在設定背景顏色, 本頁及其後的頁面會使用這個背景顏色
\normalcolor{顏色}	回復原來的顏色
\colorbox{顏色}{文字內容}	這是方框背景的顏色
\fcolorbox{框色}{框內背景色}{文字內容}	這是方框顏色和其內背景顏色不同

6.5.2 colortbl 的主要指令

指令	功能
\columncolor	這會使用文章所有內容都使用這個顏色
\rowcolor	定義顏色
\arrayrulecolor{顏色}	這是在設定背景顏色, 本頁及其後的頁面會使用這個背景顏色
\doublerulesepcolor{顏色}	回復原來的顏色

在這裡,\columncolor 和 \rowcolor 的參數是一樣的, 他們的共同語法是:

\columncolor[色彩模型]{顏色}[左緣突出長度][右緣突出長度]

我們現在就來看個實例, 這裡頭有四個小例子, 包括: 灰階橫條、部份欄位著色、整個表格在著色背景及單一個表格內方框著色:

段考成績表

	段考	
	英文	數學
陳水扁	61	95
連戰	65	90
馬英九	80	95
呂秀蓮	82	85
李登輝	59	88
李遠哲	99	99

	段考	
	英文	數學
陳水扁	61	95
連戰	65	90
馬英九	80	95
呂秀蓮	82	85
李登輝	59	88
李遠哲	99	99

段考成績表

	段考	
	英文	數學
陳水扁	61	95
連戰	65	90
馬英九	80	95
呂秀蓮	82	85
李登輝	59	88
李遠哲	99	99

	段考	
	英文	數學
陳水扁	61	95
連戰	65	90
馬英九	80	95
呂秀蓮	82	85
李登輝	59	88
李遠哲	99	99

6.6 浮動環境

tabular 表格, LaTeX 都會把他視爲一個獨立的 box , 也就是會把他當成一個字母單位在處理, 他不能被分割, 常常因爲圖表稍大些 LaTeX 就會起新頁去置放, 但這樣一來原本的頁面就會顯得空盪, 整個版面看起來很不自然, 這種情形下, 他們的置放位置就很重要了, 使用浮動環境的話, LaTeX 會繼續文字的部份, 而把圖表置放在下一頁的頂端。通常, 在 LaTeX 的浮動環境下, 圖表通常會置放在一頁的頂端或都是底部, 正常是不置放在一頁中間的位置, 除非強迫指定, 有放不下的情形時, 就會讓他佔一整頁。因此, LaTeX 就得把前後位置經過整體的計算後再來決定圖表應該置放在什麼地方, 這就是所謂的浮動環境。

6.6.1 基本的浮動環境

LaTeX 的浮動環境很簡單, 就是把表格置於 table 環境當中就可以了。在裡頭有 \{caption} 指令可以指定表格的標頭, 而且編譯後會自動標上 Table n: 字樣, 後接 caption 的內容, 那個 n 會自動編號。

一般國際上較正式的文件, caption 置放的位置慣例是「表上圖下」, 也就是說表格的標題是置於表格上方, 圖形則在下方。但 LaTeX 對 caption 的置放位置, 是對於不管圖表皆置於下方的配置, 我們把 caption 置於上方時, caption 和表格的間距將會太小。如果不想手動去調整, 可以找 topcapt package 試試看。

6.6.2 浮動環境選項參數

LaTeX 的浮動環境的配置, 有時會不符和我們實際上的期望, 這時可加入選項參數。

指令	功能
h(here)	置於下指令處位置
t(top)	置於一頁的頂端
b(bottom)	置於本頁底部, 如空間不夠會置於次頁
p(page)	單獨佔一頁, 此頁沒有內文的部份
\suppressfloats[位置]	抑制浮動物件置放於本頁的某處, 他會出現在次頁
!	置於以上選頁之前, 會更強烈要求達到此選項的作用。但對 p 則無作用

第 7 章
圖形的引入

這裡我們使用 graphicx package 來說明, 他會自動引入 graphics package, 這兩個 package, 主要是一些指令的參數用法不同, 由於 graphicx 的參數用法彈性較大, 而且也和 LaTeX 的一些參數的形式較符合, 因此, 我們就以 graphicx 來說明, 引入巨集時就引用 graphicx 就可以了。

圖 7.1: 原始圖檔: 現在的小孩

7.1 引入外來圖檔的方法

使用 graphicx package 的 \includegraphics 指令:

\includegraphics[選項參數]{圖檔名稱}

這樣就行了, 就這麼簡單!不使用浮動環境也是可以的。

7.1.1 includegraphics 指令的選項參數

我們還沒談到 \includegraphics 選項參數的部份。這個選項參數很多, 功能很強大。這些選項參數可以有多個, 各選項間以逗點來分隔, 他的值的設定是使用等號。

1. angle
   旋轉的角度。旋轉指的是逆時針的方向轉的, 除非使用負數的角度。

2. orgin
   旋轉的中心點。

3. width
   這是指圖形的寬度, 會自動伸縮調整, 長度亦會等比例調整。

4. height
   這是指圖形的高度, 會自動伸縮調整, 寬度亦會等比例調整。

5. totalheight
   這是指圖形的總高度, 即 height 再加上 depth 的值。會自動伸縮調整, 寬度亦會等比例調整。

6. scale
   按一定比例縮放, 這沒有單位, 這是縮放倍數。

7.1.2 指定圖檔的搜尋路徑

每個人操作電腦的習性不同, 理所當然無法強求圖片在每台電腦的目錄完全一樣, 爲了因應每換台電腦, 每個圖檔的路徑就要更改一次之情況, 我們可以在 preamble 區新設定一個指令,

圖 7.2: 用 scale 縮0.5倍

指令名稱讀者自己設定, 後面接圖檔目錄。例如:

\newcommand{\imgdir}{D:/mytex/images/}

圖 7.3: 用angle 轉35度, 並調整圖的長寬


(甲) (乙)

圖 7.4: 圖片並排


(丙) (丁)

圖 7.5: 圖片並排

第 8 章
數學式

數學式是 TEX 完美的地方, 不同於目前排版的 Word , 打個數學式要先安裝方程式編輯器, 且輸入極爲不方便, 有了 TEX 以後, 數學式就不再那麼麻煩了。

8.1 隨文數式 (math inline mode)

這是在夾雜在一般文章內的數學式子, 是隨著整個文章段落一起排版的。

1. $ 數學式子 $
   其實, 我們在前面的章節的例子裡, 就已經常常在使用了, 只是沒有詳細說明。由兩個錢字符號 $ 所包圍的內容就會進入隨文的數學模式, 在一般文字段落內要使用到一些數學式子的話, 這是最方便的方法。爲什麼是使用錢字符號? 因爲 Knuth 教授認爲數學是很「昂貴」的!眞正文章中要寫錢字符號時, 要把他 escape, 寫成 \$, 大概是指, 平常不必把錢看得太重的意思吧 (這是我猜的)! :-)

2. \begin{math} 數學式子 \end{math}
   如果數學式子很長, 那麼使用環境的方式亦可。但是這個環境和一般的環境不同的是, 他不會在上下行區隔出來, 而是隨著其他正常文字一起排版的。要非常注意的是, 在這個環境的上下行不要留空白行, 否則會另起段落排版, 那就不是我們所要的隨文數式了。

3. \(數學式子 \)
   這是 \begin{math} 數學式子 \end{math} 省略寫法。

誤	正
f(x,y)=3x+4y	$f(x,y) = 3x + 4y$
f(x, y) = 3x + 4y	$f(x,y) = 3x + 4y$
sin(2x)=-sin x cos x	$\sin(2x) = -\sin x \cos x$
f(x,y) = 3(x+y)y / (2xy-7)	$f(x,y) = 3(x+y)y/(2xy-7)$

8.2 展式數式 (math display mode)

通常獨立的數學式子, 我們不會使用一般文章一樣的做法去換行, 而是讓他進入展式數式的數學模式, 他會獨立成一行, 有需要的話也可以加入編號, 方便在文章中引用。和隨文數式另一個很大的不同是, 展示數式會適當的選用較大的數學符號及字體, 尤其是較複雜的數學式子的時候。例如:

隨文式: Let $f(x) = \sqrt[4]{x+1}$ and $g(x) = \sqrt{9-x^2}$

展式:

Let

$$f(x) = \sqrt[4]{x+1}$$

and

$$g(x) = \sqrt{9-x^2},$$

8.3 各種數學式子的書寫方法

8.3.1 分式

$$f(x,y) = 3(x+y)y/(2xy-7)$$

應改寫成:

$$f(x,y) = \frac{3(x+y)y}{(2xy-7)}$$

範例:

$$\frac{\frac{a}{x-y} + \frac{b}{x+y}}{\frac{x-y}{x+y} + \frac{a-b}{a+b}}$$

8.3.2 上下標

範例:

$$(a+b)^2 = a^2 + 2ab + b^2 \tag{8.1}$$

$$cos(2x) = cos^2 x - sin^2 x \tag{8.2}$$

$$(y^m)^n = y^{mn} \tag{8.3}$$

$${}^a_b Y^c_d \tag{8.4}$$

$$e^{tcos\theta} \tag{8.5}$$

$$\lim_{n\to\infty} \sum_{i=1}^{n} \frac{1}{n} \tag{8.6}$$

$$y_1 = 1/3(x_1 + \omega x_2 + \omega^2 x_3) \tag{8.7}$$

8.3.3 根號

範例:

$$\sqrt{x^2 + y^2} \tag{8.8}$$

$$\sqrt[5]{a + \sqrt{b}} \tag{8.9}$$

8.4 矩陣

範例:

$$A = \begin{pmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{pmatrix} \tag{8.10}$$

$$g(x,y) = \begin{cases} f(x,y), & \text{if } x < y \\ f(y,x), & \text{if } x > y \\ 0, & \text{otherwise.} \end{cases} \tag{8.11}$$

8.4.1 綜合練習

$$\frac{\partial^2 L_i}{\partial \beta_j \partial \beta_h} = -\frac{x_{ij}}{a(\phi)}\left(\frac{\partial \mu_i}{\partial \beta_h}\right) \tag{8.12}$$

$$\widehat{Hat} = \widehat{W}^{1/2} X (X' \widehat{W} X)^{-1} X' \widehat{W}^{1/2} \tag{8.13}$$

$$p(y; \alpha, \beta) = \binom{n}{y} \frac{B(\alpha + y, n + \beta - y)}{B(\alpha, \beta)} \tag{8.14}$$

$$F = \frac{SSR(X_3 | X_1, X_2)/1}{SSE(X_1, X_2, X_3)/(n-4)} \xrightarrow{H_0} F(1, n-4) \tag{8.15}$$

打到這裡足足花了我一整個禮拜, 快瘋了, 終於給我打完了!