目前，市面上有各種主流的排版軟體，很多排版軟體同時也都有文字編輯器的功能。在 Windows 上，有 Microsoft(MS) Office; 在 macOS 上，有 Pages; 在 ubuntu 上，則有 OpenOffice。既然有這麼多編輯程式，為什麼還需要再學另一個陌生的排版軟體？

這篇文章，將建立在作者¹的親身經驗，相較於網路上的 LaTeX 說明書，希望透過更多實際案例，同時結合經濟領域所需，提供大家一個不同的觀點。

一、什麼是 LaTeX?

LaTeX (發音: lay-teck)是一套由 Leslie Lamport 於 80 年代所開發的排版格式。而他所使用的語言: TeX，是美國史丹佛大學 Donald Ervin Knuth 教授於 1989 年開發完成的語言。使用 LaTeX，相較於 Word 的「所見所得」編輯模式更像在寫程式，透過指令，讓文字能服從特定的排列方式。當然，透過指令的優點，就是能讓使用者達到自訂性最大，讓大家能掌握每一個細節的設定。

相信大家對 Word 中的自動完成都不陌生，以下的「第二」和「第三」點，就是使用 Word 的自動完成所產生。

- 一、 經濟學原理
- 二、 個體/總體經濟學
- 三、 產業經濟學

試問，標號與標題之間的空白距離要如何調整？另外，要如何使每行行距都不同？相信大家一定有很多方法達到指定的效果，但卻都要付出極大的時間成本。但如果使用 LaTeX，你將能輕而易舉得達成以下效果：

一、國際貿易	大三	劉文獻教授
二、總體經濟學	大三	王瑜琳教授
三、產業經濟學	大三	崔曉倩教授
四、賽局理論	大三	唐孟祺教授
五、高等統計學	大四	劉鋼教授

大家可注意每行不固定的行距及三欄文字的對齊。在進行編輯時，你無需設定欄數、間隔數，你只需負責輸入文字內容，其餘排版 LaTeX 會幫你調整，並自動產生標號。最後，你可再依照需求，進行細節的調整，例如將特定行距增大 0.035 毫米，或將中間的年級，整列向右偏移 0.05 公分。

¹ 徐皓哲，中正大學經濟學系，103 級。

二、L^AT_EX 排版支援

LaTeX 當初建立的宗旨，就是為了要能「完美排版」文件。他除了可編輯中文(繁體、簡體)、英文，更可以畫圖、製表，甚至繪製五線譜和音符。基於他強大的對齊功能，LaTeX 已變成編輯 CV、Résumé 的最佳排版方法。

以下是 L^AT_EX 排版的主要優點:

- 相容性
- 大型文件排版
- 編輯數學式
- 巨集指令

相容性

Word 有分為舊版的 *.doc* 和新版的 *.docx* 文件；在平台支援上，也分成 MS Word for Windows 和 MS Word for Mac。一份文件，不管是在兩種版本或兩個作業系統平台上，都會有不同的呈現；一份在 Windows 上完成的文件，拿到 Mac 上用最新的 Word 開啓，原本設定好的表格和對齊的文字都會產生變動。

相較 LaTeX，不管是將寫好的編碼從 Mac 或是 Windows 開啓並進行編譯，最後將能得到完全相同的文件內容。這個道理就如你把在 Mac 上編譯的 [Java](#) 程式拿到 Windows 上跑，只要有裝所需的「編輯器」(JVM)，那麼程式就能跨平台執行。

大型文件排版

報告、論文，都是大家在大學及研究所中常見的文件類型。這些文件，少則3-5頁，多則可到60-100頁。文件中，除了夾帶圖片外，常常更附有大型表格。一般的排版軟體，在處理多頁數時，會因為要快取內容的即時預覽，導致當你捲動文件內容時，會發生卡頓、程式短時間無回應、甚至崩潰的情形發生，這些都是「前台編輯」容易發生的問題。

LaTeX 屬於「後台編輯」，因此在還沒編譯時，都不會產生「PDF文件」。所有的圖片，都是以路徑的方式呈現，並透過參數來設定到時的排版位置、大小和效果(例如: 陰影)。因此，在全部都還是編碼的情況下，編輯器就

不會有需要即時預覽的問題。

在一份報告中，假設第一節位於第3頁，第九節位於第23頁。現在，你突然想將這兩節的內容在 Word 上互換，這時不僅中間每一小節在跨頁時設定好的分段都會受到影響，在進行整個區域的移動時，不僅內容剪下貼上後的排版會跑掉，圖片長距離的搬移也容易出现錯誤，尤其當那些圖片不是以「與文字排列」的方式呈現時，更是令人頭疼。

在論文中，都會將圖片註解、數學算式逐張逐條編號。假設目前有15張圖，13條式子，今天你想在第4和第5張和第11, 12張之間，各增加一張圖；或想要將第4, 8兩式位置互換，然後刪去第2條式子，你會發現，每當你動一次，你就必須把部分的圖和算式全部重新編號。

List 1: 共變異數 和 變異數

```

1 \begin{equation}
2 Cov(X,Y)=E\Big[\big(X-E(X)\big)\big(Y-E(Y)\big)\Big]
3 \end{equation}
4
5 \begin{equation}
6 Var(X+Y)=Var(X)+Var(Y)+\underbrace{Cov(X,Y)+Cov(Y,X)}_{2Cov(X,Y)}-2Cov(X,Y)
7 \end{equation}

```

在 LaTeX 中，就如 List 1 所示，進行編輯時我們只負責公式內容，公式在整份文章中的編號，則交給 LaTeX 決定。上面的編碼經過編譯後，就會呈現如下方所示：

$$Cov(X, Y) = E\left[(X - E(X))(Y - E(Y))\right] \quad (1)$$

$$Var(X + Y) = Var(X) + Var(Y) + \underbrace{Cov(X, Y) + Cov(Y, X)}_{2Cov(X, Y)} - 2Cov(X, Y) \quad (2)$$

編輯數學式

經濟系擁有最強大的工具就是數學。從大家在大二所學的統計學到大三的計量經濟學，或是將來的理論推導，甚至拿金融資料、產業資料進行實證分析，都需要很多數學的幫助。因此，如何以最好的方式呈現數學，就成了非常重要的問題。

MS Word, OpenOffice 都有自己內建的方程式編輯器。同時，更有第三方軟體如: [MathType](#)，讓使用者在編輯文件時，能夠插入方程式。

大家在大二時，會看到非常著名的菲利浦斯曲線(Phillips Curve)，以下兩者分別是在 Word 和 LaTeX 中的方程式:

- Word:

$$\pi_t = \kappa x_t + \beta E_t \pi_{t+1} + u_t$$

- LaTeX:

$$\pi_t = \kappa x_t + \beta E_t \pi_{t+1} + u_t$$

在 Word 中排版時，雖然也有不用離開鍵盤的方程式輸入模式，但卻不能改變方程式字型，而且數學符號選擇性非常少，因此大部分都是直接使用 MathType，先在編輯器的面板上輸入數字、選擇希臘字母、上標下標，再將方程式插入 Word。但是，這些公式都不帶有書籤的功能，因此在文章中，並不能自動交叉引用這些方程式。

以下是這條曲線方程式在 LaTeX 中的編碼:

List 2: 菲利浦斯曲線

1 \pi_t=\kappa x_t+\beta E_t\pi_{t+1}+u_t

也許，方程式較短時，在面板上直接選符號比較方便，但當方程式變長時，或者有很多數學夾雜在文字當中時，一直開關 MathType 編輯面板反而會拖慢編輯速度，尤其在面對以下方程式: 同時要對齊還要調整括號大小時，使用面板點選的速度相較之下，就會明顯慢很多:

$$\hat{\beta}_{W2SLS} - \beta = \left[\underbrace{\left(\frac{1}{n} \sum_{i=1}^n \frac{x_i z'_i}{\sigma^2(z_i)} \right)}_{\xrightarrow{p} E\left(\frac{x_i z'_i}{\sigma^2(z_i)}\right)} \underbrace{\left(\frac{1}{n} \sum_{i=1}^n \frac{z_i z'_i}{\sigma^2(z_i)} \right)^{-1}}_{\xrightarrow{p} E\left(\frac{z_i z'_i}{\sigma^2(z_i)}\right)^{-1}} \underbrace{\left(\frac{1}{n} \sum_{i=1}^n \frac{z_i x'_i}{\sigma^2(z_i)} \right)}_{\xrightarrow{p} E\left(\frac{z_i x'_i}{\sigma^2(z_i)}\right)} \right]^{-1} \cdot \left[\underbrace{\left(\frac{1}{n} \sum_{i=1}^n \frac{x_i z'_i}{\sigma^2(z_i)} \right)}_{\xrightarrow{p} E\left(\frac{x_i z'_i}{\sigma^2(z_i)}\right)} \underbrace{\left(\frac{1}{n} \sum_{i=1}^n \frac{z_i z'_i}{\sigma^2(z_i)} \right)^{-1}}_{\xrightarrow{p} E\left(\frac{z_i z'_i}{\sigma^2(z_i)}\right)^{-1}} \underbrace{\left(\frac{1}{n} \sum_{i=1}^n \frac{z_i (x'_i \beta + \epsilon_i)}{\sigma^2(z_i)} \right)}_{\xrightarrow{p} E\left(\frac{z_i (x'_i \beta + \epsilon_i)}{\sigma^2(z_i)}\right)} \right] - \beta$$

而且 MathType 在 Mac 上，並沒有跟 Word 和 Pages 完全相容。但不可否認，用 LaTeX 打長方程式時，要特別小心打錯字的問題。

以下則是數學和文字混合在一起的情況，這是大家熟悉的極限定義:

For all $\epsilon > 0$, there exists $\delta \in \mathbb{N}$, such that for all $n > \delta$, $|x_n - x| < \epsilon$. Then the sequence x_n of real numbers converges to some limit $x \in \mathbb{R}$.

巨集指令

在 L^AT_EX² 中，有許多巨集套件(package)如 [TiKz](#) 可以使用。這些巨集能強化環境及增加許多讓使用者能進行細部微調的功能。例如: `package{enumerate}`能使標頭編號更加多元、`package{graphicx}`能針對圖形和顏色等相關設定進行調整。使用者除了可以使用“`\newcommand`”來針對不同頁面自訂效果外，也可以寫自己的巨集(macros)，為自己所需要的排版方式，量身訂做適合的套件。

所有的 L^AT_EX文件，都是根據以下初始編碼進行擴展。

List 3: 文件基本格式

```

1 \documentclass[12pt]{article}
2 \begin{document}
3
4 Hello World!
5
6 \end{document}

```

List 3 就是一個 LaTeX 文件的基本編碼格式，編譯後就會產生一個PDF檔，並且印有一行文字: Hello World!。編碼一開始必須宣告文件類別(class)，例如: article, report，並且給定整體文件的預設字級(font size)，所有內文都必須放在`\begin{.}`和`\end{.}`之間。

使用L^AT_EX排版，能讓你專注於文章的內容，不需顧慮排版。如果大家有用Word寫過文章，一定會發現文章寫完後，還要從頭到尾檢查文字間有無多餘的空格及引用格式是否正確、一致。但其實，你只需使用 BibTeX 巨集套件，將文獻作者、出處等填入，剩下就交給 LaTeX 自動生成。而引用格式，可以根據期刊需求更改，例如社會科學常用的 [APA 引用格式](#)。

在撰寫英文文章時，LaTeX 也會自動幫你將每行文字數目做最佳分配，並在斷字時，自動將單字從正確的音節位置分行:

“...promote students’ abilities in critical thinking and logical reasoning and **de-velop** their analytical and problem-solving skills. Our aim is to increase students’ competitiveness in their future careers.”

這篇文章和大家考研究所經常看到的參考書《提綱挈領學統計》，都是 L^AT_EX的排版作品。

²L^AT_EX(Windows): [MikTeX](#)(T_EX) + [Texmaker](#)(編輯器) + [cwTeX](#)(繁中) 或 [cTeX](#) (簡中)
L^AT_EX(Mac): [MacTeX](#)(T_EX) + [Texmaker](#)(編輯器) + [cwTeX](#)(繁中)

三、成本效益分析

LaTeX 是一個免費、跨平台(作業系統)相容的排版方式。其寫作模式，基於指令編輯，會促使大家在撰寫時，以結構嚴謹的方式進行寫作。輸出成PDF文件後，章節也會被編入選單，讓其他讀者在瀏覽時，能輕鬆的透過書籤來翻閱你的文章。同時，LaTeX生成的文件，也比一般 Word 做出來的更帶有清晰的結構，因為每張圖、表、公式都帶有書籤，因此很容易就能建立總體清單，在引用時也能輕而易舉的將圖、表與內文連結。

另外，因為編碼文件非常乾淨，因此在一般的記事本、終端機中的 `nano` 上就可編輯，而且自動化方便，可以輕而易舉的與 `Python` 結合。Word 雖為圖形介面，但很多時候卻不知道要去哪找需要用的功能。

最重要的，LaTeX 編輯出來的文件，就是美。

但是，在這些強大功能的背後，卻須付出代價。L^AT_EX 是一款學習曲線(learning curve)非常陡的系統，不只上手不容易，只要一有bug，編譯就會失敗。加上編輯不同語言，都需要額外的巨集套件³，而這些套件的安裝設定又要花費一翻功夫。雖然網路上有許多教學，但大多步驟都十分複雜難懂。幸好這些步驟，都只需做一次就一勞永逸。

所有的安裝和設定都解決後，接下來就要熟悉 T_EX編碼和指令。雖然 YouTube 上有一些相關的教學，但大多都是英文教學。大家也許可以輕而易舉的找到 L^AT_EX的指令手冊，但相信照著手冊上來寫 code，結論就像看著 Stata help 來寫 `Stata` code 一樣痛苦。因此，最快的學習方式，還是直接拿別人寫好的「成品」直接看，從那些可以成功編譯的編碼學習，並且再去做更改，這樣在完成5-10份文件後，就能成功上手。

雖然，不會 L^AT_EX也能靠 Word 完成所有任務，但多學一種排版方法，沒有任何壞處，只要撐過辛苦的前期，往後就能編輯出漂亮的文件。這也像軟體工程師一樣，雖然現在，在 `GitHub` 上 `Swift` 已經成了最大宗、流行的程式語言，但大家都還會再去學 `Java`、`Python`等其他語言。

與其羨慕那些漂亮的排版，不如相信自己也可以做得到。

³編輯中文，需要使用 `cxTEX` 或 `XqLATEX`。但因 `XqLATEX`在即時預覽時，會出現錯誤，故本文推薦 `cxTEX`。