LaTeX Starter Guide

Michael DeCrescenzo*

Updated September 6, 2018

Abstract

This document describes and exemplifies many important LaTeX concepts and best-practices. It discusses document setup, important commands and environments, text formatting, math, figures and tables, and so on. This document (a `.pdf` file) is distributed alongside the source code used to create it (a `.tex` file), allowing new LaTeX users to see how LaTeX code becomes printed content. The source code, along with other workshop materials, can be found on the workshop's Github page.[1]

1 How to read this document

This document is intended to teach LaTeX by example. The text contains a discussion of LaTeX commands, but for instructional purposes it will also be valuable to download, inspect, and modify the source code[2] yourself.

Examples of LaTeX code will appear in the text. These examples are printed with a `monospace` font. If we want to view a lot of code at once, we will see code presented in a code block. Code blocks are styled to make them easier to read:

```
This is where
a lot
of code
would be written at once
```

It is important to distinguish the code that builds the document from the example code that is printed in the final document. The code that builds the document isn't directly visible

*Ph.D. Candidate, Political Science, University of Wisconsin–Madison. This handout borrows heavily past workshop materials created and passed down by Matthew Holleque, Sarah Niebler, and Dave Ohls, Sarah Bouchat, Nat Olin, Richard Loeza, and José Luis Enríquez Chiñas, and Michael Masterson.

[1] `https://github.com/mikedecr/latex-workshop-2018`

[2] I advise that you preserve an original copy of the source code, for future reference. The file can be copied from here: `https://raw.githubusercontent.com/mikedecr/latex-workshop-2018/master/handout/latex-handout-2018.tex`

to you—it is interpreted by the TEX compilation engine to create the final document. The example code that appears in the final text is simply printed, not executed.[3]

2 Essentials

LATEX works like this.

1. **Source code.** You start with a plain text file (with a `.tex` extension), which serves as the source code for your final document. The source contains the text of your document (unstyled) and code to "mark up" the text. The markup code provides logical structure and stylistic guidelines for the text to follow.
2. **Compilation.** The source code is compiled by the TEX engine. The engine interprets the markup code to learn which text serves what purpose, and it follows those instructions to build the final document. The compilation process runs into errors if the compiler can't interpret the source file.
3. **Output.** Successful compilation returns an output document, usually a `pdf` file (usually a *pretty* `pdf` file).

LATEX "source" documents are `.tex` files.

They require you to declare a document class at the very top of the file. The most common class is `article`, but other options include `paper`, `book`, `memoir`, `report`, and so on. Details about these other classes can be found online.

The content of the document lies in the `document` environment. Environments are delineated with `\begin{environment}` and `\end{environment}` tags.

The most basic document you could create, therefore, would look like the following:

```
\documentclass{article}

\begin{document}
  Hello, world!
\end{document}
```

LATEX contains a number of additional functionalities, but not all are loaded by default. You can load packages for LATEX capabilities using the `\usepackage{package-name}` command. You would place package-loading commands in the *preamble*, which is the part of the `.tex` document between the `\documentclass` and the `document` environment.

Some packages allow you to tweak additional settings with further commands. These commands must be included *after* the `\usepackage` command.

[3] We discuss how to print code without executing it later in this document.

3 Titles and Abstracts

Creating titles in LaTeX is easy.

```
\title{Title here}
\author{Author name here}
\date{Type a date or use the \today command}
\maketitle
```

LaTeX will take care of the formatting, so you don't have to waste your time with it.

For articles and conference papers, you can also include abstracts.

```
\begin{abstract}
  Here's what this paper is going to be about.
\end{abstract}
```

4 Chapters, Sections, Subsections, ...

It is common to organize a document with sections. Available section levels are `section`, `subsection`, and `subsubsection`.

```
\section{Section}
\subsection{Subsection}
\subsubsection{Subsubsection}
```

If you put a * after the section command, it will suppress section numbering.

```
\subsection{Subsection with a number}
\subsection*{Subsection without a number}
```

The above commands would create the following subsection titles:

4.1 Subsection with a number

Subsection without a number

The `book` and `report` classes also allow you to use `\part` and `chapter` structures.

5 Aligning Left, Right, and Center

You can align your your document using different alignment environments.

You can align text to the left with the `flushleft` environment.

You can align text to the right with the `flushright` environment.

You can align text to the center with the `center` environment.

6 Spacing

Line Spacing

You can use the `setspace` package to allow for double spacing (and other space factors). Just add `\usepackage{setspace}` to the preamble, and add `\doublespacing` to the body of the document. The `setspace` package also allows `\singlespacing`, `\onehalfspacing` for 1.5 spacing, and custom spaces using `\setstretch{x}`. These commands also have environment versions, in case you want to take finer control of the document.

```
\begin{singlespace}
  This text would be single spaced.
\end{singlespace}

\begin{doublespace}
  This text would be double spaced.
\end{doublespace}
```

Vertical Space

You can use the `\vspace{}` command to insert blank space. You put an amount of space inside the `{}`. Let's say you wanted to leave a one-inch space before the next line. `\vspace{1in}` would do it.

Check it out.

While taking fine control of vertical space within a document is *possible*, it is generally against the LaTeX philosophy to take such fine control. It is smarter to set your desired

global options in the document's preamble, leaving the document body itself free of tedious stylistic manipulation.

6.1 Line Breaks, Page Breaks, and Blank Pages

There are several ways to to create a new paragraphs and line breaks.

New paragraphs are created by leaving a blank line between two blocks of text in your `.tex` file.

You can also force a single line break by typing `\\` or `\newline` at the end of a line. New lines are not the same as paragraph breaks; indentation rules will apply to new paragraphs but not new lines.

To insert a page break, you can use either the `\newpage` or `\clearpage` commands.

7 Indentation

LaTeX will automatically indent new paragraphs, but you can stop that behavior by adding `\noindent` before a new paragraph.

A document's overall indentation behavior is something you want to set with global parameters in the preamble rather than with `\noindent` commands throughout your entire document. For example, this document uses `\usepackage{parskip}` to (a) add space between paragraphs and (b) prevent indentation in new paragraphs. These `parskip` package defaults can be overridden if you include more commands in the preamble after loading the package (see package documentation).

8 Commenting

You can use the % to write comments in the code. This way, you can leave notes that don't show up in the document.

```
% This would not show up in a normal document.
```

The `comment` package provides a comment environment.

```
\begin{comment}
  You can write

  long

  comments

  that span multiple paragraphs

  and none of the text would show up in the final document.
\end{comment}
```

I have found the `comment` package to be quite handy when writing and revising papers.

Bonus: the `todonotes` package also provides cool way to leave notes to yourself in a document using `\todo{}`. You can modify the appearance of the bubbles in the package options. Useful

9 Font Sizes

Font sizes can be done with inline commands or in environments. The sizes (and size "names" are...

tiny scriptsize footnotesize small normalsize large Large LARGE huge Huge

Inline sizing can be done like so.

```
Inline sizing can be done {\huge like so}.
```

Or you can use an environment.

```
\begin{tiny}
  Or you can use an environment.
\end{tiny}
```

10 Font Styles

Inline font styling (emphasis, bolding) is handled with inline commands as well. Emphasizing is handled with `\emph{}`, bolding with `\textbf{}`.

Although most typefaces and document classes don't show the difference, *italicizing* and *emphasizing* can sometimes look different. *Slanting* (`textsl{}`) is also different from *italicizing*, so that is another distinction to keep in mind.

You can write fixed-width font using the "teletype" font, accomplished using `\texttt{}`. `This is teletype.`

Small Capitals can be accomplished with `\textsc{}`.

11 Verbatim

You can use the `verbatim` environment to print text *exactly* how it appears. This is useful for writing LaTeX commands without executing them. It can also be a nice way to display output from other programs (like R).

```
This text here created inside the verbatim environment.
```

You can do `inline verbatim` as well.

```
You can do  \verb+inline verbatim+ as well.
```

12 Special Characters

Certain characters have special meaning in LaTeX. In order to use them, you have to include escape characters (a backslash) or type a special command.

$ % & _ {} ~ ^ \

```
\# \$ \% \& \_ \{\} $\tilde{}$ $\grave{}$ \textbackslash
```

$300 is 50% of my take-home pay.

```
\$300 is 50\% of my take-home pay.
```

Some of these special characters affect the characters immediately after them. For example, my old roommate (José Luis Enríquez Chiñas) must have a difficult type TeXing his name:

```
Jos\'e Luis Enr\'iquez Chi\~nas
```

13 Quotation Marks

Quotes are a little funky in LaTeX because it takes the direction of the quote mark literally. To get normal looking quotation marks, use the accent key for opening quotes and apostrophes or double quotes for closing quotes.

"Political Science is fun!" Looks wrong.

“Political Science is fun!” Looks much better.

```
"Political Science is fun!" Looks wrong.

``Political Science is fun!''  Looks much better.
```

14 Dashes

Your professors care about the following distinctions. They are also important to get right using LaTeX.

One dash `-` is a hyphen. Example: cold-blooded (`cold-blooded`).

Two dashes `--` is an en-dash. Example: University of Wisconsin–Madison
(`University of Wisconsin--Madison`)

Three dashes `---` is an em-dash. Example: Nietzsche—who greatly overueses em-dashes—once wrote...
(`Nietzsche---who greatly overueses em-dashes---once wrote\dots`)

15 Footnotes

To enter text in a footnote, type `\footnote{}` and include whatever text you want in the brackets.[4] Footnotes go *after* punctuation, in case you were wondering.

16 Lists

LaTeX has separate `itemize` and `enumerate` for creating unordered and ordered lists, respectively.

An unordered list:

- First item on the list
  - First item on the sublist
  - Second item on the sublist
- Second item on the list

```
An unordered list:

\begin{itemize}
  \item{First item on the list}
  \begin{itemize}
    \item{First item on the sublist}
    \item{Second item on the sublist}
  \end{itemize}
  \item{Second item on the list}
\end{itemize}
```

An ordered list:

1. First item on the list
   (a) First item on the sublist
   (b) Second item on the sublist
2. Second item on the list

[4] Very exciting footnote text.

```
An ordered list:

\begin{enumerate}
  \item{First item on the list}
  \begin{enumerate}
    \item{First item on the sublist}
    \item{Second item on the sublist}
  \end{enumerate}
  \item{Second item on the list}
\end{enumerate}
```

You could customize the character beside a list item with square brackets after `\item`. For example, `\item[/] Item text` would create list items like

/ Item text

... although as with many things in LaTeX, it's efficient for you *note* to sweat the small stuff like this.

17 Math Basics

One of the most attractive and powerful features of LaTeX is its ability to typeset complex mathematical notation via "math mode". Math mode can be invoked inline by wrapping an expression in dollar signs ($). For example, `$e=mc^2$`, is rendered as $e = mc^2$. Alternatively, a variety of commands allow you set equations apart from the main body of your text. For example:

```
\begin{equation}
y = \beta_1 + \alpha
\end{equation}
```

Renders as:

$$y = \beta_1 + \alpha \tag{1}$$

If you would like to suppress the equation numbering, you can add an asterisk to the end of the command (`\begin{equation*}...\end{equation*}`). One nice shorthand: `\[...\]` is equivalent to `\begin{equation*}...\end{equation*}`.

17.1 Fractions

To write fractions, simply use `\frac{}{}` and place the numerator of the fraction in the first set of brackets and the denominator of the fraction in the second set of brackets. You can also embed fractions within fractions. For example, $\frac{1}{2}$ or $\sum \frac{x_i - \bar{x}}{n-1}$. Producing these fractions is straight forward, but sometimes the nested code blocks can be difficult to read.:

```
$\frac{1}{2}$
$\sum\frac{x_{i}-\bar{x}}{n-1}$
```

17.2 Greek Letters

To write Greek letters, simply write a backslash and the name of the letter after the slash. For example, `$\beta$` and `$\sigma$` renders as β and σ. Note that capitalizing the first letter of the Greek letter name gives you a capital Greek letter. So, `$\Delta$` renders as Δ, but `\delta` is δ. For some Greek letters, such alpha, the capital is the same as the Roman. In cases like these, you just use the Roman—there is no `$\Alpha$`, only A.

When there are multiple versions of the same Greek letter, you can use the `var` prefix to use the alternate. For example, `$\epsilon = \varepsilon$` will render as $\epsilon = \varepsilon$.

A complete list of commands for the Greek alphabet is available at `http://jblevins.org/notes/greek`.

17.3 Operators and Other Symbols

Numerous symbols can be used in the math environment. Commonly used symbols include inequalities, set notation, and operators. Many of these are straight forward (e.g. equals, greater than, less than), but many are not. We review a few commonly used symbols and operators below.

To write a summation sign or a product sign, the commands are `\sum and \prod`, which produce $\sum$ and $\prod$ respectively.

For inequalities, the commands are `\neq`, `\ge`, `\le`, and these produce $\neq$, $\geq$, and $\leq$ respectively.

Common set notation commands include `\mathbb{R}` ($\mathbb{R}$), `\in` ($\in$), `\not \in` ($\notin$), `\mid` ($\mid$). Because curly braces are interpreted by the LaTeX compiler, you need to escape them with a backslash whenever you use them in math mode. So, `{...}` will either not render or cause an error. Instead, use `\{...\}`. This will render as {...}. Putting it all together, we can write `$\mathbb{A} \in \{1,2,3\}$`, which will render as $\mathbb{A} \in \{1, 2, 3\}$.

You may also want to be aware that operators and functions like $\max_{x_L}$, $\ln(x)$, $\lim_{x \to +\infty}$, $\int$, and ∂ are also LaTeX commands. You can read more about these and other commands here: `http://en.wikibooks.org/wiki/LaTeX/Mathematics`.

17.4 Hats and Bars

To place a hat or a bar over any math character, simply place the text in brackets after the writing `\bar{}` or `\hat{}` commands. `$\bar{\beta} \neq \hat{y}$` gives us $\bar{\beta} \neq \hat{y}$.

17.5 Superscripts and Subscripts

To write a superscript, write `^{}` with the text of the superscript in the brackets. If you just have one character in the superscript, you do not have to include the brackets. For example, `$x^3$` renders as x^3 or `$e^{-z\gamma}$` renders as $e^{-z\gamma}$.

If you forget the brackets, as in `$e^-z\gamma$`, the superscript will render as $e^-z\gamma$. (I wrote my own keyboard shortcuts for Sublime Text to always insert curly brackets after a `^` to prevent these kinds of mistakes.)

To write a subscript, write `_{}` with the text of the subscript in the brackets. Again, if you have just one character in the subscript, you do not have to include the brackets. For example, `$x_1$` renders as x_1 and `$x_{ij}$` renders as x_{ij}.

Some commands allow you to use the subscript and superscript environments together to place notation on top of or underneath other notation. For example, `$\sum_{i=0}^{100} i$` renders as $\sum_{i=0}^{100} i$.

To make it look a little better, we can add the command `\limits` after the `\sum` command: `$\sum\limits_{i=0}^{100} i$`. From this we get:

$$\sum_{i=0}^{100} i$$

Including `\displaystyle` before the $\sum$ accomplishes a similar effect. This can also improve limit expressions ($\lim_{x\to\infty}$) and integrals ($\int_x^y$). Display style will also slightly increase the size of your expressions in in-line math, which can prevent complicated math from getting shrunk too small in the middle of a block of text.

18 More Complicated Math

Now, for more complicated math, you can use the `align` environment, which will align a set of equations along an operator of your choice. To pick the operator on which you want to align your list of equations, place a `&` in front of it. For multiple lines, you need to place two backslashes at the end of each line of equations, and there should be no empty lines anywhere in the `align` environment. Take a look at the following, which will not compile correctly:

```
\begin{align}

\ln(L) &= n \ln(\alpha) + n \ln (x) - n \ln (x)\\
\frac{d \ln(L)}{d\alpha} &= \frac{n}{\alpha} + n \ln(x) - 0\\
\frac{n}{\alpha} + n \ln(x) &= 0\\
\frac{n}{\alpha} &= -n \ln(x)\\
```

```
n &= \alpha (-n \ln(x))\\
\alpha &= \frac{n}{-n \ln(x)}\\
\alpha &= \frac{-1}{\ln(x)}\\
\end{align}
```

This, however will compile correctly:

```
\begin{align}
\ln(L) &= n \ln(\alpha) + n \ln (x) - n \ln (x)\\
\frac{d \ln(L)}{d\alpha} &= \frac{n}{\alpha} + n \ln(x) - 0\\
\frac{n}{\alpha} + n \ln(x) &= 0\\
\frac{n}{\alpha} &= -n \ln(x)\\
n &= \alpha (-n \ln(x))\\
\alpha &= \frac{n}{-n \ln(x)}\\
\alpha &= \frac{-1}{\ln(x)}\\
\end{align}
```

It renders as:

$$
\begin{aligned}
\ln(L) &= n\ln(\alpha) + n\ln(x) - n\ln(x)\\
\frac{d\ln(L)}{d\alpha} &= \frac{n}{\alpha} + n\ln(x) - 0\\
\frac{n}{\alpha} + n\ln(x) &= 0\\
\frac{n}{\alpha} &= -n\ln(x)\\
n &= \alpha(-n\ln(x))\\
\alpha &= \frac{n}{-n\ln(x)}\\
\alpha &= \frac{-1}{\ln(x)}
\end{aligned}
$$

You can add additional space (or subtract space) between lines by specifying skip lengths in square brackets. Example:

```
\begin{align*}
  a &= 1 \\
  a &= 1 \\[6pt]
  a &= 1 \\[12pt]
  a &= 1
\end{align*}
```

. . . renders as. . .

$$a = 1$$
$$a = 1$$

$$a = 1$$

$$a = 1$$

19 Figures

The fundamental component of including a graphic or figure is the `\includegraphics` command. Here, the syntax `\includegraphics[width=10cm]{extensions.png}` specifies that it should include the figure with filename *extensions.png*[5] with a width of 10 centimeters (it will automatically adjust the height of the figure along with this).

```
\section{Figures}

  % primarily relying on the graphicx package

  The fundamental component of including a graphic or figure is the \verb+
  \includegraphics+ command.  Here, the syntax \verb+\includegraphics[
  width=7cm]{extensions.png}+ specifies that it should include the figure with
  filename \textit{extensions.png}\footnote{If the filename isn't in the same
  directory as the .tex file, you'll need to specify the full file path.
  Generally, it's simpler to just make sure all files called are in the same
  folder.} with a width of 7 centimeters (it will automatically adjust the
  height of the figure along with this).
```

To create figures with accessories, it is often useful to put it in a 'float' using the *figure* environment, which is generally good practice for figures. This allows captions and other elements to be included and automatically associated with the figure.

Sizing can be done as a fraction of the width of the text using `[width = x\textwidth]` where `x` is the proportion of the text width that the graphic fills.

The following code inserts a graphic of Pokémon statistics:

```
\begin{figure}[!ht]
  \centering
  \includegraphics[width = \textwidth]{imgs/specs-gen-vio-box.png}
  \caption{Distribution of Pok\'emon Stats}
  \label{fig:poke-stats}
\end{figure}
```

A few other bits of syntax and notes:

- `[ht]` locates the float 'here' on the page or on the top of the next page if there isn't enough space. Alternately, can use `[t!]` ('top') or `[b!]` ('bottom'). The exclamation

[5] If the filename isn't in the same directory as the .tex file, you'll need to specify the full file path. Generally, it's simpler to just make sure all files called are in the same folder.

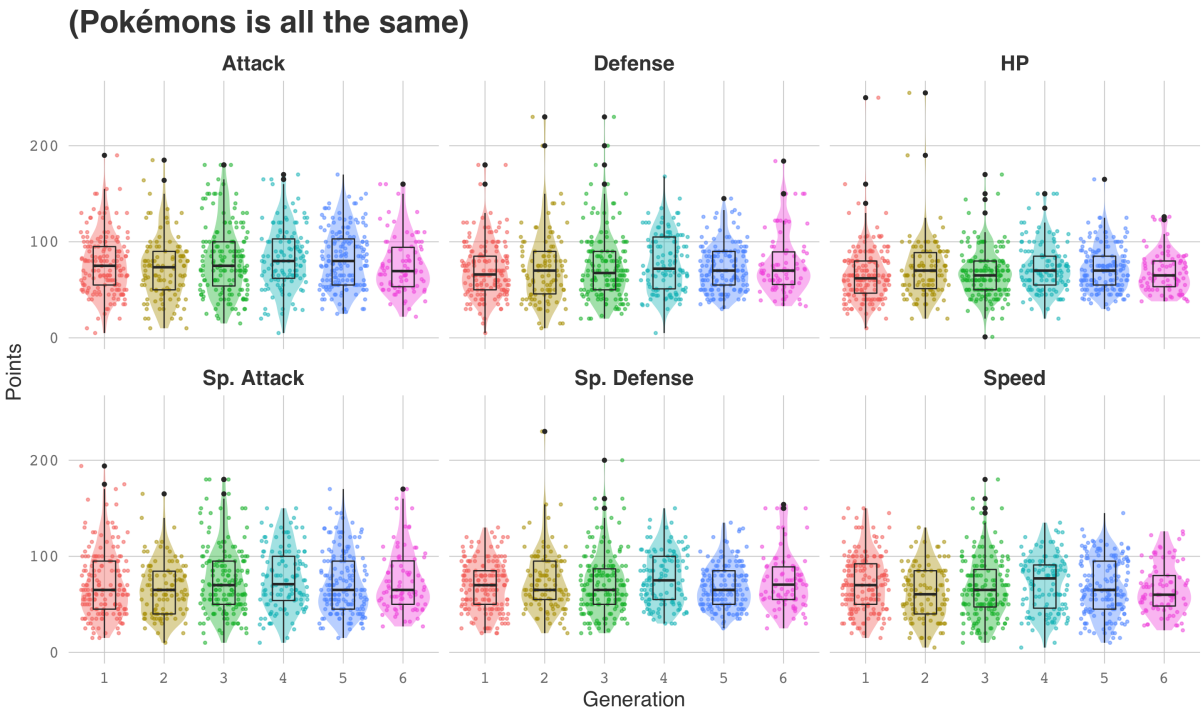

Figure 1: Distribution of Pokémon Stats

point overrides LaTeX's normal rules for placing figures.

- `\caption` inserts a caption. Alternately, can use `\caption*` to inset an unnumbered caption.
- Several file formats will work, but in general, you should include "vector" graphics (such as `.pdf`s), since they are infinitely scalable without losing image detail. Stata and R can produce `.pdf` graphics with no problem at all.

You can also put multiple figures within the same command to get side-by-side figures, three across, a grid, etc. Include multiple `\includegraphics` commands and set the sizes appropriately to make it look how you want.[6] You could caption multiple figures in the same floating environment with the `subcaption` package.

Labeling your figures and tables with the `\label{labelname}` command immediately after the caption allows you automatically to reference them later using `Figure~\ref{labelname}` (the tilde inserts a space that won't break across multiple lines). I could reference Figure 1 using `Figure~\ref{fig:poke-stats}`. The beauty of LaTeX is that figure number is handled automatically, so you never have to re-number anything if you shuffle the order of figures.[7]

[6] If you want a grid but don't want the figures to be large enough to force that to happen automatically, you can accomplish that using the tabular environment discussed below.

[7] Try doing that with Microsoft Word!

20 Tables

Tables are created using the `tabular` environment, generally within the `table` floating environment to allow for accessories (as with figures, this is good practice but not strictly necessary if you have no accessories).

This code produces the table below:

```
\begin{table}[!ht]
\begin{center}
\caption{Selected Films}\label{film}
\begin{tabular}{l|c | c |  c | D{.}{.}{4}}
           & Acting & Writing   & Effects  & \multicolumn{1}{c}{Plot} \\ \hline
Inception  & good   & okay      & great    & 90.52   \\ \hline
The Artist & great  & none      & B\& W    & 100.0   \\ \hline
Birdemic   & bad    & atrocious & clip art & 3.1     \\ \hline
Twilight   & none   & none      & sparkly  & .111    \\
\end{tabular}
\end{center}
\end{table}
```

Table 1: Selected Films

	Acting	Writing	Effects	Plot
Inception	good	okay	great	90.52
The Artist	great	none	B& W	100.0
Birdemic	bad	atrocious	clip art	3.1
Twilight	none	none	sparkly	.111

What the syntax elements are doing:

- `{l|c|c...}` specifies the number of columns, their alignment (c=center, l=left, r=right), and vertical lines (|) between them.
- `D{.}{.}{4}` aligns cells in that column on the decimal point (requires the `dcolumn` package).
- `&` specifies breaks between cells within the table.
- `\\` creates a line break at the end of each line of the table.
- `\hline` inserts a horizontal line.
- `\caption` inserts a caption, just like with figure

Additional notes:

- Most normal in-text style and mathematical commands (boldface, italics, equations, etc) work fine within tables.
- If a table needs to include a lot of information, and the font doesn't need to be huge to be readable, you can make the whole thing 'small' or 'footnotesize' to fit.
- Toggling 'wrap' on/off in WinEDT will change the display settings with your code to see unbroken lines - very useful when editing wide tables or ones with lots of text

modifying commands in the input that take up space. In TexShop, this can be toggled under 'Source'-'Wrap Lines'

Table 2: Detective Comparison

	L&O: SVU		Literature		Other	
	Benson	Stabler	Holmes	Hardy Boys	Dick Tracy	Batman
Loose Cannon	6	10	9	0	3	11
Det. Ability	8	5	10	3	4	11
Effectiveness	80%	70%	95%	100%	40%	5%
Tenure (yrs)	14+	12	23	1	66	75

Some new syntax introduced:

- `\cline{2-7}` inserts horizontal lines across columns 2 through 7.
- `\multicolumn{2}{c}{text}` merges adjacent cells to have a multiple-column cell, in this case spanning 2 columns and aligned in the center.

Item	Unit Cost	Qty	Cost
Buying coffee to claim a table for the morning	$3.00	60	$150.00
Library books	$0.00	45	$0.00
Spinal realignment (needed after carrying library books up and down Bascom Hill)	$150.00	1	$150.00
Stress balls	$4.50	3	$13.50
Buying coffee to claim a table for the afternoon	$3.00	60	$150.00
Office supplies (pens, pencils, highlighters, tags, index cards, binders, cookies, ice cream)	(assorted)		$60.00
Celebratory after-prelims steak (Tornado Room)	$60.00	1	$60.00
		Total:	$642.50
		Finishing Prelims:	priceless

2 more handy new commands:

- `p{3.5in}` fixes the width of the column, allowing you to wrap text.
- `\rule{0cm}{.5cm}` creates a line with the specified width (here 0cm) and height (here .5cm). This can be used to create an actual line or, as it is used here, to adjust the vertical spacing of cells (by making a line with 0 width, so it doesn't actually draw anything).

You can create sideways tables with the appropriately named `sidewaystable` package.

Some additional advice: Minimize the extent to which you hand-create tables. R and Stata can generate LaTeX code as output using the `stargazer`, `texreg`, `xtable` (for R), or `outtex` (for Stata) packages. Hand-typing tables introduces human error and makes your

workflow less reproducible for anyone who wants to replicate your analysis, so relying on algorithms for creating tables is good practice.

21 Arrays and Matrices

The tabular environment can also be used to draw arrays, matrices, and other similar objects using much of the same syntax as tables. A few samples:

$$B_1(q) = \begin{cases} 0 & \text{if} \quad q < \frac{1}{4} \\ [0,1] & \text{if} \quad q = \frac{1}{4} \\ 1 & \text{if} \quad q > \frac{1}{4} \end{cases}$$

$$\text{Matrix} = \left[\begin{array}{cccc} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{array} \right]$$

$$\begin{bmatrix} 4 & 5 \\ 2 & 0 \\ 1 & 7 \end{bmatrix} \times \begin{bmatrix} 8 & 4 & 0 \\ 6 & 3 & 4 \end{bmatrix} = \begin{bmatrix} 62 & 31 & 20 \\ 16 & 8 & 0 \\ 50 & 25 & 28 \end{bmatrix}$$

- `\[` and `\]` start and close the object.
- `\left**` and `\right**` define the boundaries of the object. These can be curly brackets `{`, parentheses `(`, square brackets `[`, or empty `.` (period).
- Matrix environments such as `bmatrix` provide similar functionality.

22 Good LaTeX Practices

- **Re-use code**. Copy and paste from old code you or someone else created, and modify as needed. No need to reinvent the wheel every time. Make a blank `.tex` file that has a preamble using your most-used external packages, keep it somewhere safe, and start all fresh papers with that template.
- **Include comments**. Particularly as you're still learning, it's a good idea to write comments in your code to explain why you're doing things or how you're doing it. It's also a nice way to include content that you're not sure if you'll use—put it in a comment and you still have it later if you want it back in.
- **Whenever you open something, close it immediately**. Whether you're beginning and ending environments or opening and closing parenthesis or brackets, always put the end in right away, then fill in the content in between. This will greatly cut down on syntax errors from forgetting to close things once you've gotten distracted by the substance. Some software will do this for you, so find handy software!
- **Compile early and often**. Bugs can be hard to find, so if you write a long document and it won't compile it is often frustrating and time-consuming to figure out where it's getting tripped up. Compile often—anytime you do anything new or complicated—to make sure it's working; that way when errors occur you'll know where they are.
- **Debug in the order the errors appear.** When debugging a LaTeXdocument, make sure that you start by fixing errors as early in the document as you can. Errors tend to cascade—failing to close an environment early in the document can result in multiple other problems later in the document.
- **Comment-out lines to debug**. If you're getting errors and can't figure out why, try commenting-out complicated parts here and there to find the source that's tripping it up.
- **-** $\neq -\neq$ —. Don't confuse your dashes, en dashes, and em dashes. You have a powerful typesetting system at your fingertips, make use of it! For those who can't remember when to use your en dash (–) and when to use your em dash (—), Wikipedia has your back (`http://en.wikipedia.org/wiki/Dash`).

23 Useful Resources

- The Comprehensive TeX Archive Network (`http://www.ctan.org`). In particular see the 'Not So Short' guide at `http://ctan.tug.org/tex-archive/info/lshort/english/lshort.pdf` with tons of information on beginning and advanced topics.
- The LaTeX Wikibook (`https://en.wikibooks.org/wiki/LaTeX`). This is how I (Michael DeCrescenzo) learned LaTeX.

- TEX Exchange (`http://tex.stackexchange.com/`). This site collects user questions related to working with LATEX. Other users offer answers and can "upvote" the best answers. The large user base results in very high quality answers to both common and obscure questions. Don't be afraid to post a question if you're unable to figure out a given issue.
- Crash Course in LATEX (`http://haptonstahl.org/latex/`). A very nice user-friendly website guide to LATEX, created by a Steve Haptonstahl, a (former) political methodologist.
- Google (`https://www.google.com/`). If all else fails (or, perhaps, before searching through pages and pages of documentation) just search for what you're looking for — there will be a user group discussion or a guide posted somewhere that explains how to do it.
- Wikipedia's List of LATEX editors (`http://en.wikipedia.org/wiki/Comparison_of_TeX_editors`). There are various alternative LATEX editors that are of varying degrees of quality and price. Many are better than the default editors that come with various TeX distributions.
- Detexify (`http://detexify.kirelabs.org`). Need to include a symbol but don't know what it's called or how to enter it into LaTex? Draw the symbol using this tool, and the program will return the command to draw it. For example, draw that "trident-looking thing" and the tool will tell you how to make a ψ.
- Overleaf (`https://www.overleaf.com/`). Cloud-based LATEX editor which allows documents to be edited by multiple individuals at the same time and which compiles in real-time.