LaTeX Starter Guide

Michael DeCrescenzo*

Updated September 11, 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]

Contents

*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

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 `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 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).

2.1 Source files

LaTeX "source" documents are text files that use the file extension `.tex`.

[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

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

At the beginning of every `.tex` file, you declare a document class. The document class defines a broad set of document behaviors. 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_name}` and `\end{environment_name}` tags.

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

```
\documentclass{article}

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

2.2 Compilation

In the old days, you could compile a `.tex` file using shell commands. Today, your code editing software will have a button or a keyboard shortcut that will allow you to build the document into a `.pdf` file.

2.3 Packages

LaTeX contains a number of additional functionalities, but not all are loaded by default. To extent LaTeX's default behavior, you can load additional.

Packages are declared in the *preamble* of your source file, which is the part of the file before the `document` environment (but after `\documentclass{}`). You call them using `\usepackage{package-name}`.

Some packages allow you customize their options with further commands. These commands must be included after the package is declared (i.e. after the `\usepackage` command).

3 Titles and Abstracts

Creating titles in LaTeX is easy. Within the `document` environment, write...

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

LaTeX will format the title for you (according to the parameters set by the document class [or any other packages you loaded]).

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 and Sections

It is common to organize a document with sections. Available section levels are `section`, `subsection`, and `subsubsection`. Sections are numbered by default. To suppress section numbering, include a `*` after the section keyword.

```
\subsection{Subsection}

Some subsection text!

\subsection*{Subsection}

Text in a section without a number.

\subsubsection{Subsubsection}

Even lower text!
```

4.1 Subsection

Some subsection text!

Subsection

Text in a section without a number.

4.1.1 Subsubsection

Even lower text!

4.2 Other types of sections

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

Paragraphs Use `\paragraph{paragraph-name}` to create named paragraphs, if you'd like. They are kind of weird, though.

5 The text block: spacing, breaks, alignment, etc.

5.1 Text margins

For some mysterious reason, you will be required to use one-inch margins for almost all of your work. To obey this stylistic requirement, you can use the `geometry` package. Calling the package like so will set one-inch margins for the entire document:

```
\usepackage[margin=1in]{geometry}
```

. . . but more options are out there.[4]

5.2 Line Spacing

For another mysterious reason, you will be required to double-space almost all of your work even though it looks kinda bad. To obey this stylistic requirement, use the `setspace` package, which allows control over line spacing (also called *leading*). Adding `\doublespacing` to the body of the document will double space any succeeding text. Well, technically it's not *full* double spacing—more like 1.7 spacing. You can set the exact spacing factor using `\setstretch{x}`, where x is the spacing factor. `\setstretch{2}` would be *true* $2x$ spacing. There are also commands for `\singlespacing` (for single spacing) and `\onehalfspacing` (for 1.5x spacing). 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}
```

[4] https://en.wikibooks.org/wiki/LaTeX/Page_Layout

5.3 Text alignment: justification, left, right, center

By default, LaTeX will justify your body text. This means that some of the words in the text block may be hyphenated across lines. Some users find LaTeX's default hyphenation to be a little too aggressive. There are some packages (e.g. `microtype`) that advertise various typographical interventions that may improve hyphenation behavior (among other things), but these packages can be complicated and unnecessary for most LaTeX users.

If you are so inclined, you can align your your document using different alignment environments.

The `flushleft` environment aligns text to the left.

The `flushright` environment aligns text to the right.

The `center` environment can be useful, but mainly for aligning tables and figures.

5.4 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 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 not apply to new lines,
only to new paragraphs.

To insert a page break, you can use `\newpage` or `\clearpage`.[5]

5.5 Indentation

One behavior you will notice is that the first paragraph after a section heading is not indented. LaTeX is opinionated and believes that this typographic convention is better. If this bothers you enough to intervene stylistically, use the `indentfirst` package.

LaTeX will automatically indent subsequent paragraphs, but you can stop that behavior by adding `\noindent` before a new paragraph. As always, it's best to set a document's behaviors globally rather than locally, so it's best to avoid using `\noindent` if you can help it.

To obtain a paragraph style like the one used for this current document (which can be good for problem sets), use the `parkship` package.

[5] These two commands have slightly different behaviors, but these differences appear only in certain circumstances. If you really want to know the details, see https://tex.stackexchange.com/questions/45609/is-it-wrong-to-use-clearpage-instead-of-newpage.

6 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 text would appear in the final document % but this text would not

% This entire line would not appear
```

The `comment` package provides a comment environment. This works like a "block comment," allowing you to comment large chunks of text with ease. I have found this 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{insert note}` or `\todo[inline]{insert note}`.

Margin note

Inline note

7 Content control: footnotes, lists, etc.

7.1 Footnotes

To enter text in a footnote, type `\footnote{}` and include whatever text you want in the brackets. This sentence contains a footnote.[6]

```
This sentence contains a footnote.\footnote{Very exciting.}
```

For the sake of readability in your source code, it can be helpful to write footnotes like this:

```
Another example of a footnote.%
  \footnote{Check it out.}
More text.
```

Another example of a footnote.[7] More text. Including the comment symbol is important, as it prevents a space from being inserted between the period and the footnote number.[8]

7.2 Lists

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

[6] Very exciting.
[7] Check it out.
[8] Footnotes go *after* punctuation, in case you were wondering.

An unordered list:

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

```
An unordered list:

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

An ordered list:

1. First item on the list
2. Second item on the list

```
An ordered list:

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

You can create nested lists by placing lists environments within one another.

1. Outer item one
    (a) Inner item one
    (b) Inner item two
2. Outer item two
3. Outer item three

```
  \begin{enumerate}
  \item Outer item one
    \begin{enumerate}
     \item Inner item one
     \item Inner item two
    \end{enumerate}
  \item Outer item two
  \item Outer item three
\end{enumerate}
```

The code can be a little difficult to read. You can make things easier on yourself by being strategic with indentation in the source code.

7.2.1 Customizing lists

Again, you should refrain from taking too much *ad hoc* control over the visual appearance of your document. That being said, I have found some customizations helpful for creating lists.

First, you can customize the character that delineates a list item with square brackets after `\item`. For example:

→ Item text

→ Item text

```
\begin{itemize}
  \item[$\rightarrow$] Item text
  \item[$\rightarrow$] Item text
\end{itemize}
```

Next, sometimes the vertical space between list items is undesirably large. You can use the `noitemsep` argument (from the `enumitem` package) to compress this space.

- Item one
- Item two

Or set the list separation style in the preamble with `\setlist{noitemsep}`.[9]

8 Typeface things

8.1 Type family

The default typeface family used by LaTeX is called "Computer Modern." It comes in Serif, Sans-Serif, and `Monospace` varieties.

There are some packages out there to change the type family, but these are the kind of inessential details that LaTeX doesn't want to concern the user with. You can look these up on your own.

8.2 Type size

Here's the thing about manually resizing your text: try to avoid it. These are the kinds of details that LaTeX can do for you, so we recommend not preoccupying yourself too much with it.

[9] See the package documentation for more separation styles: http://mirror.hmc.edu/ctan/macros/latex/contrib/enumitem/enumitem.pdf

It's fine to set your global text size, which can be done as global option when you specify your document class. You could set the global font size to be 12pt like so.

```
\documentclass[12pt]{report}
```

For most use cases, this is all you need.

If you absolutely must manually resize text on an *ad hoc* basis, you can do so using either inline commands or with environments. The font sizes (and their "names") are...

tiny scriptsize footnotesize small normalsize large Large LARGE huge Huge

Declaring the font size in the document class will change the document's `\normalsize`. Other sizes are relative to `\normalsize`.

Set a font size environment like...

```
\begin{huge}
 huge text
\end{huge}
```

huge text

Use inline sizing if you want only some words to be different

```
Use inline sizing if you want only {\Large some words} to be different
```

8.3 Type 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* are technically different concepts, and LaTeX cares that you know the difference. *Slanting* (`textsl{}`) is also different from *italicizing*, so that is another distinction to keep in mind.

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

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

9 Quirks with inputting characters

9.1 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 backtick character for opening quotes, and use apostrophe or double-quote characters 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.
```

Some LaTeX editors are smart and can be customized to intelligently insert TeX-style quotation marks—for example, this package for Sublime Text.

9.2 Hyphens and dashes

Your professors care about the following distinctions, and you should too. 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`)

9.3 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.

$ % & _ ` { } ~ \

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

9.4 Verbatim

You can use the `verbatim` environment to print text *exactly* how it appears. This can be useful for writing LATEX commands (or other bits of computer code) without executing them. This is how we have included so many examples of LATEX code throughout this document.[10] 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.
```

10 Figures, Tables, etc.

10.1 Graphics

Graphic capabilities are enhanced by including the `graphicx` package.

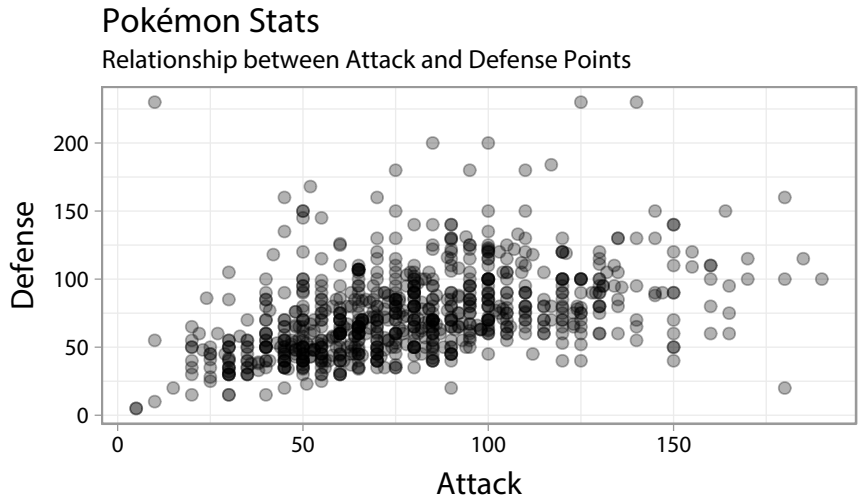
The fundamental component of including a graphic or figure is the `\includegraphics{path}` command, where `path` is the relative path to an image file.

Specify the size of the image using height or width. Here, the command...

```
\includegraphics[width = .9\textwidth]{imgs/specs-gen-vio-box.png}
```

specifies that we want to go into the `imgs` folder and include the image called `attack-defense.pdf`. We specify the width to be 90 percent of the width of the text block, but we could have specified a width in inches or centimeters. The height is scaled automatically with the width.

[10] Well, not exactly. This document primarily uses the `listings` package for verbatim commands. This is done purely for stylistic reasons (the gray background in the code blocks). So this is a rare "do as I say, not as I do" moment in this LATEX workshop.



10.2 Floats

One great feature of LaTeX is that figures can be allowed to "float" to an appropriate position in the text. This lets you write in the code that a figure should go approximately in some place, but the exact position is determined by LaTeX. We recommend the `float` package for enhanced float capabilities.

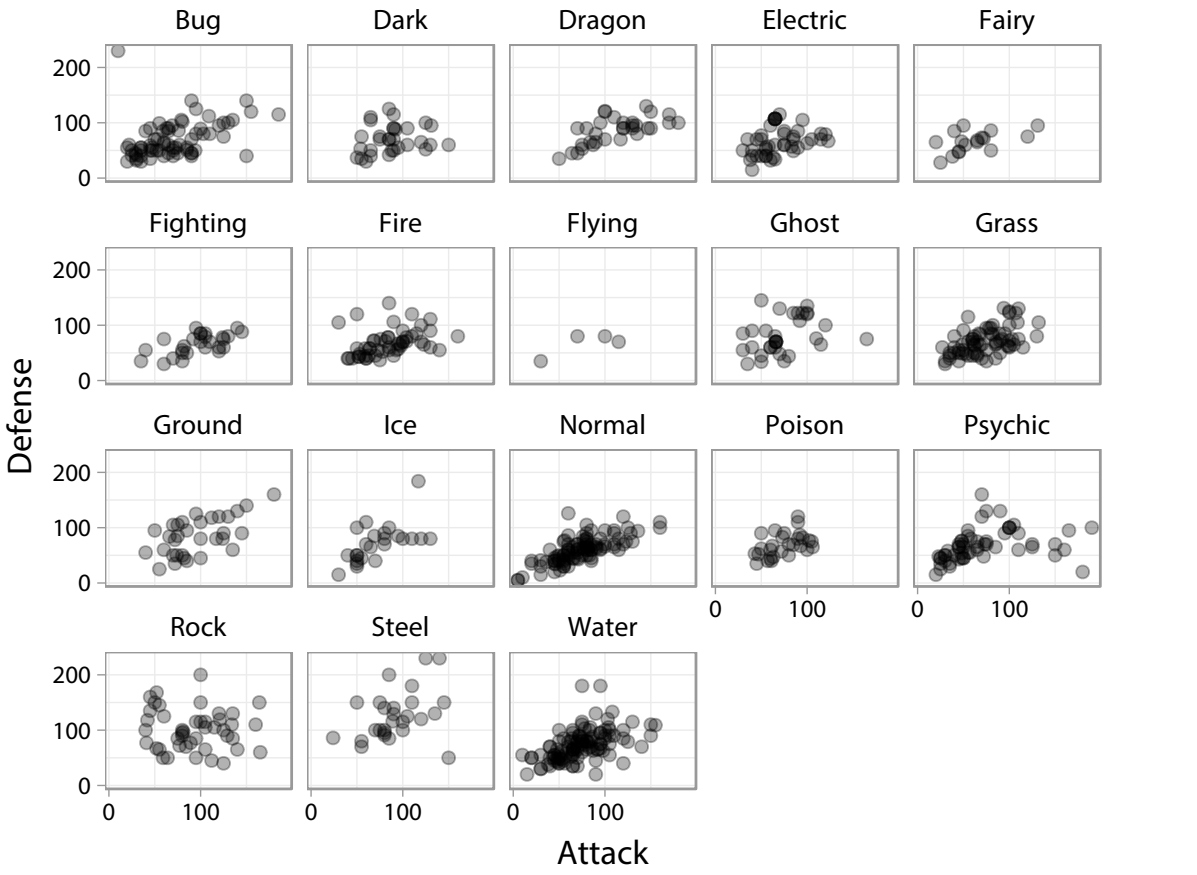
Place a figure in a floating environment using the `figure` environment. Floats also allow you to specify captions and labels. If we put the above image into a float, we can add more information about the figure:

```
  \begin{figure}[ht]
  \centering
  \caption{Distribution of Pok\'emon Stats}
  \label{fig:poke-stats}
  \includegraphics[width = \textwidth]{imgs/attack-defense-type}
  Data source: \url{https://www.kaggle.com/rounakbanik/pokemon}
\end{figure}
```

A few other bits of syntax and notes:

- `[ht]` locates the float 'here' on the page or, if 'here' doesn't work, then on the top of the next page. Alternately, can use `[t!]` ('top') or `[b!]` ('bottom'). The exclamation point overrides LaTeX's normal rules for placing figures.
- `\caption` inserts a caption. You 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.

Figure 1: Distribution of Pokémon Stats



source: https://www.kaggle.com/rounakbanik/pokemon

Labeling floats with the `\label{labelname}` command immediately after the caption allows you automatically to reference the float later using, e.g., `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.[11]

10.2.1 Additional figure capabilities

Quick things:

- The `rotating` package contains the `sidewaysfigure` environment, in case you want to rotate a wide figure to fit vertically on the page.
- You can place multiple images within the same float, but the float will only have one caption and one label. If you want to have *multiple* captions and multiple labels,

[11] Try doing that with Microsoft Word!

however, you can use `subcaption` package for the `subfloat` environment.

10.3 Tables

Tables are created using the `tabular` environment. Furthermore, place tabulars within the `table` floating environment to allow for floating placement, captions, labels, and so on.

This code produces the table below:

```
\begin{table}[!ht]
\begin{center}
\caption{Selected Films} \label{tab: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:

- The `table` environment is what floats in your document. This environment contains a `tabular` environment that is centered horizontally on the page by the `center` environment.
- We've given the table a `caption` and a `label` for referencing Table 1 in the text.
- `{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, see documentation for details).
- Cells in the same row are divided by `&` symbols.
- `\\` creates a line break at the end of each row in the table.
- `\hline` inserts a horizontal line.

Additional notes:

- Most textual and mathematical style commands (bold, emphasis, equations, etc.) work within tables.

- It's usually best practice to align text to the left, numbers to the right (or numbers with `dcolumn`)
- Sometimes people mess with the text size in a table if it needs to fit a lot of information. Use with caution.
- Tables can look quite ugly in your source code. Sometimes it can be helpful to turn text wrapping OFF in your text editor to view them more cleanly. Or, you could use careful indentation to keep the code organized.

View the source code to examine the following table about detectives.

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 with center alignment.

10.3.1 Booktabs

There exists a table package called `booktabs` that is widely regarded to be *the* way to make prettier tables in LaTeX. The package is designed to facilitate a handful of stylistic improvements on the typical LaTeX table, but with an interface that feels similar. See the documentation for more details.[12] The basic idea is that the table should include:

- More spacious horizontal rules (no double rules)
- No vertical rules

Compare the following table created using ordinary LaTeX and then `booktabs`. (Consult source file for code.)

[12] http://mirrors.ibiblio.org/CTAN/macros/latex/contrib/booktabs/booktabs.pdf

Table 3: "Suck It, Trebek" Trivia Team Details ("vanilla" LaTeX table)

Name	Year	Topic Strength
Richard	NA	Comics, Wresting
José Luis	7	Comics, TV, Movies
Mike	5	NA
Micah	5	American History
Erin	4	Science, Sports
Jordan	4	British History
Rachel	6	Sports, Music

Table 4: "Suck It, Trebek" Trivia Team Details (`booktabs` table)

Name	Year	Topic Strength
Richard	NA	Comics, Wresting
José Luis	7	Comics, TV, Movies
Mike	5	NA
Micah	5	American History
Erin	4	Science, Sports
Jordan	4	British History
Rachel	6	Sports, Music

Notice how the `booktabs` table just looks, feels, smells better.

In the code, the main differences you will encounter are the use of `\toprule`, `\midrule`, and `\bottomrule` (instead of `\hline`), and (should you need it) the use of `\cmidrule` instead of `\cline`. See documentation for more details.

10.3.2 Other table advice

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

Avoid creating tables yourself. Statistical software contain tools for creating tables in LaTeX code. Hand-typing tables, while *possible*, is prone to error and makes your workflow less reproducible. Creating tables algorithmically is a much better practice. Packages for LaTeX tables include `texreg`, `xtable`, and `stargazer` for R; `outtex` for Stata. My choices among R packages are `xtable` for summary statistics and marginals, `texreg` for regression tables.

And **use booktabs**. It just looks way better.

11 Math

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 = \alpha + \beta x
\end{equation}
```

Renders as:

$$y = \alpha + \beta x \tag{1}$$

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*}`.

Note: equations can be labeled like figures and tables. We could refer to Equation 2 like so:

$$y_i = X_i\beta + \varepsilon_i \tag{2}$$

```
We could refer to Equation~\ref{eq:mtx-reg} like so:

\begin{equation} \label{eq:mtx-reg}
  y_{i} = X_{i}\beta + \varepsilon_{i}
\end{equation}
```

11.1 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. For example, $\frac{1}{2}$ or $\frac{1}{n} \sum x_i$.

```
For example, $\frac{1}{2}$ or $\frac{1}{n} \sum x_{i}$.
```

It is possible to embed fractions within fractions.

11.2 Greek

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. 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 Greek symbol is the same as the Roman symbol. 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.

11.3 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—you must write `\{...\}` to get {...}. 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 $\max(x)$, $\ln(x)$, $\lim_{x \to \infty}$, $\int$, and ∂ are also LaTeX commands. You can read more about these and other commands here: http://en.wikibooks.org/wiki/LaTeX/Mathematics.

11.4 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 $\bar{\beta} \neq \hat{y}$.

11.5 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 want to put multiple characters into the superscript but forget the brackets (`$e^-z\gamma$`), only the first character will be in the superscript ($e^- z\gamma$).

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 these commands look a little better, we can add the command `\limits` after the `\sum` command: `$\sum\limits_{i=0}^{100} i$`. From this we get: $\sum\limits_{i=0}^{100} i$. We can also accomplish this by including `\displaystyle` before the expression: `\displaystyle \sum_{i=0}^{100} i` gives us $\displaystyle \sum_{i=0}^{100} i$. This can also improve limit expressions ($\lim\limits_{x\to\infty}$) and integrals ($\int\limits_x^y$). 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.

11.6 Multiple, aligned equations

To align multiple equations together, 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.

The following code...

```
\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}
```

. . . will render as. . .

$$\begin{aligned}
\ln(L) &= n\ln(\alpha) + n\ln(x) - n\ln(x) && (3)\\
\frac{d\ln(L)}{d\alpha} &= \frac{n}{\alpha} + n\ln(x) - 0 && (4)\\
\frac{n}{\alpha} + n\ln(x) &= 0 && (5)\\
\frac{n}{\alpha} &= -n\ln(x) && (6)\\
n &= \alpha(-n\ln(x)) && (7)\\
\alpha &= \frac{n}{-n\ln(x)} && (8)\\
\alpha &= \frac{-1}{\ln(x)} && (9)\\
& && (10)
\end{aligned}$$

You can suppress equation numbering, again, by typing `align*` (with an asterisk) instead of `align`.

Split a numbered equation across multiple lines using the `split` environment within `align`:

$$\begin{aligned}
y_i &= \hat{y}_i + \varepsilon_i\\
y_i &= \hat{\alpha} + \hat{\beta}x_i + \varepsilon_i
\end{aligned} \qquad (11)$$

The `align` environment is useful enough to replace the `equation` environment, in my opinion.

If your equations are getting too smushed together, 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*}
```

dots renders as. . .

$$\begin{aligned}
a &= 1\\
a &= 1\\[6pt]
a &= 1\\[12pt]
a &= 1
\end{aligned}$$

11.7 Flexible brackets

Sometimes the stuff you want to put inside of parentheses is bigger than the parentheses themselves. For example:

$$(\frac{1}{n})$$

You can create size-flexible brackets by prefacing the bracket symbols with `\left` and `\right`. If you type `\left( \frac{1}{n} \right)`:

$$\left(\frac{1}{n}\right)$$

This trick works with parens, square brackets, curly brackets, etc.

11.8 Arrays and Matrices

Tables, meet math. If you want to create a table in math mode, you can use the `array` environment, which behaves basically like the `tabular` environment. They are useful for piecewise functions:

```
\begin{align*}
  f(x) &= \left\{
            \begin{array}{ll}
              -x & \text{ if $x < 0$} \\
              1 & \text{ if $x = 0$} \\
              x^{2} & \text{ if $x > 0$} \\
            \end{array}
          \right.
\end{align*}
```

$$f(x) = \begin{cases} -x & \text{if } x < 0 \\ 1 & \text{if } x = 0 \\ x^2 & \text{if } x > 0 \end{cases}$$

There are also matrix environments for math mode (with the `amsmath` package), which place a table inside of various kinds of brackets. I tend to use `bmatrix` for square bracket matrices.

```
\begin{align*}
  \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}
\end{align*}
```

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

You can also use various `\dots` commands for matrices, like with this monstrosity:

$$\begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \cdots & 0 & 1 \end{bmatrix}$$

12 Good LaTeX Practices

- **Minimize stylistics interventions**. LaTeX is built on a philosophy that you should be concerned with the content of the document, not be preoccupied with controlling the stylistic details. Try not to overly style your documents. If you want to intervene, it's always better to intervene with global settings (in the preamble), since they can be easily modified later and universally applied through the document.
- **Take content control seriously.** LaTeX can be integrated into the social science workflow in a number of ways. To minimize human error (and save time and effort), export figures and tables directly from Stata or R and then import them with LaTeX. You can also do this with statistics such as p-values so that your document is always up-to-date with your most recent analysis!
- **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 a bracket or an environment, 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 LaTeX document, 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.

13 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.