A Brief Introduction to LaTeX

Jimmy Kelliher

October 19, 2022

Table of Contents

1 What Is LaTeX?

LaTeX is a document preparation program. Unlike Microsoft Word, which is a what-you-see-is-what-you-get word processor, LaTeX functions by compiling unformatted text and commands into a prepared document. LaTeX was originally designed to standardize the procedure for writing mathematical publications. However, in the past few decades, an increasing number of academic disciplines have adopted LaTeX as the standard for writing theoretical papers and even textbooks.

That's great that it's popular, but why should I use LaTeX? This is a valid question! Different people have different reasons for using this program, but to me, LaTeX is appealing because it allows for precision and consistency in communicating mathematical ideas. As we will see, LaTeX allows us to create professional-looking equations, tables, and graphics. Being able to write abstract mathematical notions in a clear and effective way can aid us in our research and in the classroom. Of course, talk is cheap, so let's dive right in and see what LaTeX has to offer!

2 Commands and Environments

2.1 Commands

As with any typesetting language, we can make text **bold** or *italic*. We can also make text blue or red. The symbols &, %, $, #, _, {, }, ~, ^, and \ have special uses in LaTeX, so we have a unique means of calling them in a paragraph. The first seven of these symbols can be called with the escape character \, but the remaining three symbols have their own special command.

The most important symbol above is the \ symbol. When this symbol is followed by text without a space between, LaTeX interprets the expression as a command. For example,

- the command `\textbackslash` prints the \ symbol;
- within a math environment, `\log` outputs the function log, the natural logarithm; and
- to instantiate a new environment, we enclose objects between the commands `\begin{...}` and `\end{...}`, as we will soon see.

These commands are all *functions*, and functions live in *packages*. When you execute a command, LaTeX will search the packages you loaded for the definition of that command; if you make a typo or if you call a command that doesn't exist, LaTeX will fail to compile and instead notify you that the command is undefined.

Another notable symbol of those above is the $ symbol. By enclosing an object between two $ symbols, we are telling LaTeX to interpret the object as a mathematical one. For example, we can define a function $f : X \times Y \to \mathbb{R}$, which looks like

```
$f : X \times Y \to \mathbb{R}$
```

when written as code in our TeX document. Note that `\times` is a command that outputs the set product operator and `\mathbb{...}` is a command that returns the blackboard notation of its argument (here, the argument `R` outputs to the usual notation for the real line). In this document and in the live demo thereafter, we will see many more commands that are useful in writing mathematical prose. Before that, though, we should talk about *environments*.

2.2 Environments

If commands are the meat of this typesetting sandwich, then environments are its buns. Environments impose a special structure for a specific task. For example, the align environment outputs multi-line equations, and it does so by interpreting symbols like & and \\ in a particular way to codify the alignment of the equations. Alternatively, the graphics environment enables the use of the `\includegraphics` command, which reads in and outputs an image from the specified directory.

3 Mathematical Environments

3.1 Creating Single-line Equations

In-line mathematical objects are great, but sometimes we will want to highlight an important result or separate a longer mathematical expression. To do so, we can instantiate the math environment by enclosing a mathematical object with `\begin{equation}` and `\end{equation}`. As an example, let's look at the distribution function of our favorite random variable,

$$\Phi(x) = \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi}} e^{-\xi^2/2}\, d\xi. \tag{1}$$

Above, we have written Greek letters, fractions, exponents, and even limits of integration. LaTeX does a great job of resizing and rearranging things to make our object readable. Note that LaTeX has numbered the above equation. If we do not desire this numbering, we can instead enclose our mathematical objects between `\[` and `\]`. Let's explore a few more examples below. With LaTeX we can write limits such as

$$\lim_{n\to\infty} \left(1 + \frac{x}{n}\right)^n = e^x.$$

We can also write logical quantifiers and operators on sets such as

$$\forall \text{ sequences of sets, } (A_n)_{n\in\mathbb{N}}, \bigcup_{n=k}^{\infty} A_n = A_k \cup \bigcup_{n=k}^{\infty} \Big(A_n^c \cap A_{n+1}\Big), \quad \text{and}$$

$$x \in \Big\{\xi \in X : f(\xi) \leq a\Big\} \quad \Longleftrightarrow \quad x \in f^{-1}\Big((-\infty, a]\Big).$$

Ultimately, if you can dream the mathematical notation, you can write it in LaTeX.

3.2 Creating Multi-line Equations

The most natural extension of a single-line equation is a multi-line one. Often when writing proofs, it is important to write out each step of your logic in a systematic way so that your audience completely understands your argument.

$$\begin{aligned} x \in (A \cup B)^c &\iff x \notin A \cup B && (2)\\ &\iff x \notin A \wedge x \notin B && (3)\\ &\iff x \in A^c \wedge x \in B^c && (4)\\ &\iff x \in A^c \cap B^c && (5) \end{aligned}$$

In the align environment, we enclose our object with `\begin{align}` and `\end{align}`, and we also equip our object with more structure: `&` tells LaTeX where to create vertical line breaks, and `\\` tells LaTeX where to create horizontal line breaks. We actually see this paradigm of using `&` and `\\` for vertical and horizontal line breaks, respectively, in matrices and tables, as well. Let's consider another example on the next page.

Proposition 3.2.1. For all $a, b \in \mathbb{R}$ and for all $n \in \mathbb{N}$, it follows that

$$(a+b)^n = \sum_{k=0}^{n} \binom{n}{k} a^k b^{n-k}.$$

Proof. We proceed by induction. For the base case, let $n = 1$ and observe that

$$\begin{aligned}(a+b)^1 &= a+b \\ &= \binom{1}{0} a^0 b^{1-0} + \binom{1}{1} a^1 b^{1-1} \\ &= \sum_{k=0}^{1} \binom{1}{k} a^k b^{1-k}.\end{aligned}$$

Thus, the base case holds. For the inductive step, suppose that the claim holds for n. Then

$$\begin{aligned}(a+b)^{n+1} &= (a+b)(a+b)^n \\ &= (a+b) \sum_{k=0}^{n} \binom{n}{k} a^k b^{n-k} && \text{(by hypothesis)} \\ &= \sum_{k=0}^{n} \binom{n}{k} a^{k+1} b^{n-k} + \sum_{k=0}^{n} \binom{n}{k} a^k b^{n+1-k} && \text{(by distributivity)} \\ &= \sum_{k=1}^{n+1} \binom{n}{k-1} a^k b^{n+1-k} + \sum_{k=0}^{n} \binom{n}{k} a^k b^{n+1-k} && \text{(by reindexing)} \\ &= \binom{n+1}{0} b^{n+1} + \sum_{k=1}^{n} \left(\binom{n}{k-1} + \binom{n}{k} \right) a^k b^{n+1-k} + \binom{n+1}{n+1} a^{n+1} \\ &= \sum_{k=0}^{n+1} \binom{n+1}{k} a^k b^{n+1-k}.\end{aligned}$$

The final line follows from the well established combinatoric result wherein

$$\binom{n}{k-1} + \binom{n}{k} = \binom{n+1}{k}$$

for all $k, n \in \mathbb{N}$ such that $k \leq n$. That is, the case for n implies the case for $n+1$, and hence the result holds by mathematical induction. $\square$

In the above example, we enclose our object with `\begin{align*}` and `\end{align*}` to suppress any numbering, and we instead employ the `\tag{...}` command to manually label equations based on the progression of our argument. Writing proofs in this way reduces ambiguity and ensures that you haven't overlooked any potential gaps in your logic.

4 Itemized Lists

4.1 Creating Enumerated Lists with the Enumerate Environment

As much as I love a good proof, I acknowledge that academic papers are much more than mathematical arguments. Sometimes we'll need to write lists, make tables, and include visualizations to best communicate with our audience. Toward that aim, let's first look at the enumerate environment.

1. This is a *numbered* list.
2. LaTeX takes care of the numbering for us.
3. Elements of a list are indented by default.

Above, we have enclosed the elements of our list with `\begin{enumerate}` and `\end{enumerate}`, and each item in our list is identified by the `\item` command.

4.2 Creating Bulleted Lists with the Itemize Environment

A numbered list is an example of an ordered list. Sometimes, however, we want to make a list with no particular ordering. For this, we consider the itemize environment.

- This is a *bulleted* list.
- We can even include in-line code in a list, such as
  - $f : X \to \mathbb{R}$,
  - $\Gamma(z) \equiv \int_0^\infty t^{z-1} e^{-t}\, dt$, or
  - $\ell^\infty(T) \equiv \{f : T \to \mathbb{R} \mid \sup_{t \in T} |f(t)| < \infty\}$.
- Wow, we even can make lists within a list!

In the example above, we have a nested list. With LaTeX, making nested lists is as simple as calling the itemize environment within an itemize environment. We can even mix and math ordered and unordered lists in a single nested list.

5 Tables and Matrices

5.1 Creating Tables with the Tabular Environment

With LaTeX, I would summarize making tables as cumbersome, but rewarding. Let's consider a very simple table to start.

Table 1: Mean Glucose Level by Sub-population

	Obese	Non-obese
Exposed	125	105
Non-exposed	130	110

Much like the align environment, we specify new columns of our table via `&` and new rows of our table with `\\`. To center our table, we further enclose the entire tabular environment within a center environment. Note that for some objects, this approach to centering might fail; when this happens, it's usually best to take a trip to Stack Exchange!

Now let's consider a more complicated table with in-line math expressions, shaded rows, and a caption in addition to the title.

Table 2: Model Fit for Outcome $\log(C_{\max})$ for Type III Sums of Squares

Effect	DF	SSR	MSS	F-statistic	p-value
Sequence	1	0.0149	0.0149	0.0192	0.8909
Subject(Sequence)	23	17.7439	0.7715	5.8212	0.0000
Treatment	1	0.0043	0.0043	0.0327	0.8581
Period	1	0.0608	0.0608	0.4588	0.5049
Residuals	23	3.0481	0.1325		

Note: effect $\beta_{i(j)}$ is measured as a random effect in the above specification; all other effects are measured as fixed effects.

Though it certainly has uglier code, the fundamental syntax is the same. Do note that while the first table can be executed without calling any packages, the second table relies on several of the packages in our preamble.

5.2 Creating Matrices within a Math Environment

Matrices and tables are closely related. The key difference is that tables are defined within the tabular environment, whereas matrices exist within the math environment, and hence their cells are all interpreted to be mathematical objects by default.

$$\begin{pmatrix} 1 & \text{Hello, World!} & e \\ \pi & \sin(\beta) & \limsup_{n\to\infty} x_n \\ \frac{1}{2} & \int_\Omega X\, dP & 2^{2^{2^2}} \end{pmatrix}$$

Above, we have enclosed our matrix object within `\begin{pmatrix}` and `\end{pmatrix}`, which is further enclosed by `\[` and `\]` to let LaTeX know we're working with a mathematical object. The argument `pmatrix` refers to a matrix bounded by **p**arentheses. For determinants of matrices, we can instead consider passing the argument `vmatrix` for **v**ertical bars. Below we give an example of a matrix within the align environment.

$$\begin{aligned}\det(\lambda I - A) &= \begin{vmatrix} a_{11} - \lambda & a_{12} \\ a_{21} & a_{22} - \lambda \end{vmatrix} \\ &= (a_{11} - \lambda)(a_{22} - \lambda) - a_{12}a_{21} \\ &= \lambda^2 - (a_{11} + a_{22})\lambda + (a_{11}a_{22} - a_{12}a_{21}) \\ &= \lambda^2 - \operatorname{diag}(A)\lambda + \det(A)\end{aligned}$$

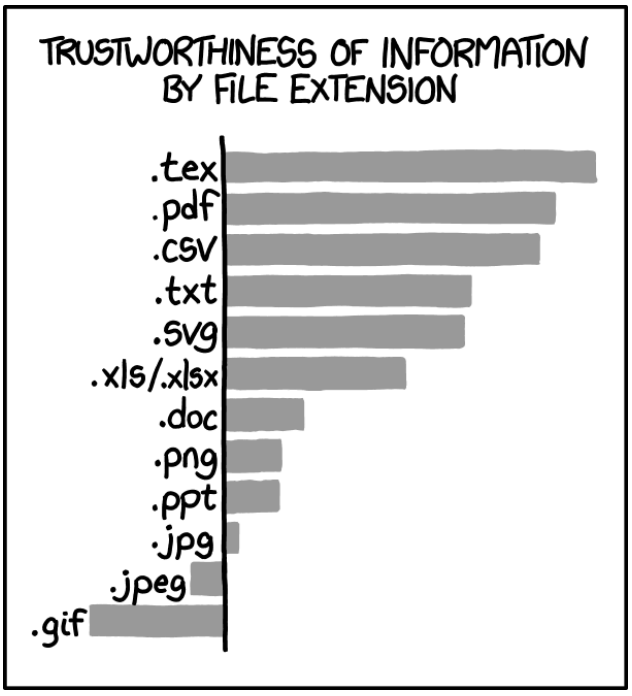
There are yet more bracketing options for matrices, but I digress for now.

We can even write matrices as in-line code: $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$. Of course, just because we *can* do something, that doesn't mean we *should* do it. When LaTeX tries to write something large like a matrix or summation within a paragraph, it will often create a more compact version of the object: $\sum_{k=1}^{n} k = \frac{n(n+1)}{2}$. While not always recommended, we can suppress this feature by employing the `\displaystyle` command: $\displaystyle\sum_{k=1}^{n} k = \frac{n(n+1)}{2}$.

6 Visualizations

6.1 Importing Images with the Figure Environment

Pulling in images is easy with the figure environment from the graphics package. We can even caption our figures to provide a source or description.



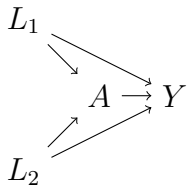
Source: Randall Munroe via xkcd.com.

Omitting the `*` in `\caption*` will automatically label the figure as Figure 1, much like we saw in the equation and align environments. Suppressing this feature depends on your needs, but it is nice that LaTeX will automatically renumber figures in the event that we add or remove one. As always, be sure to correctly specify the path to where your image lives!

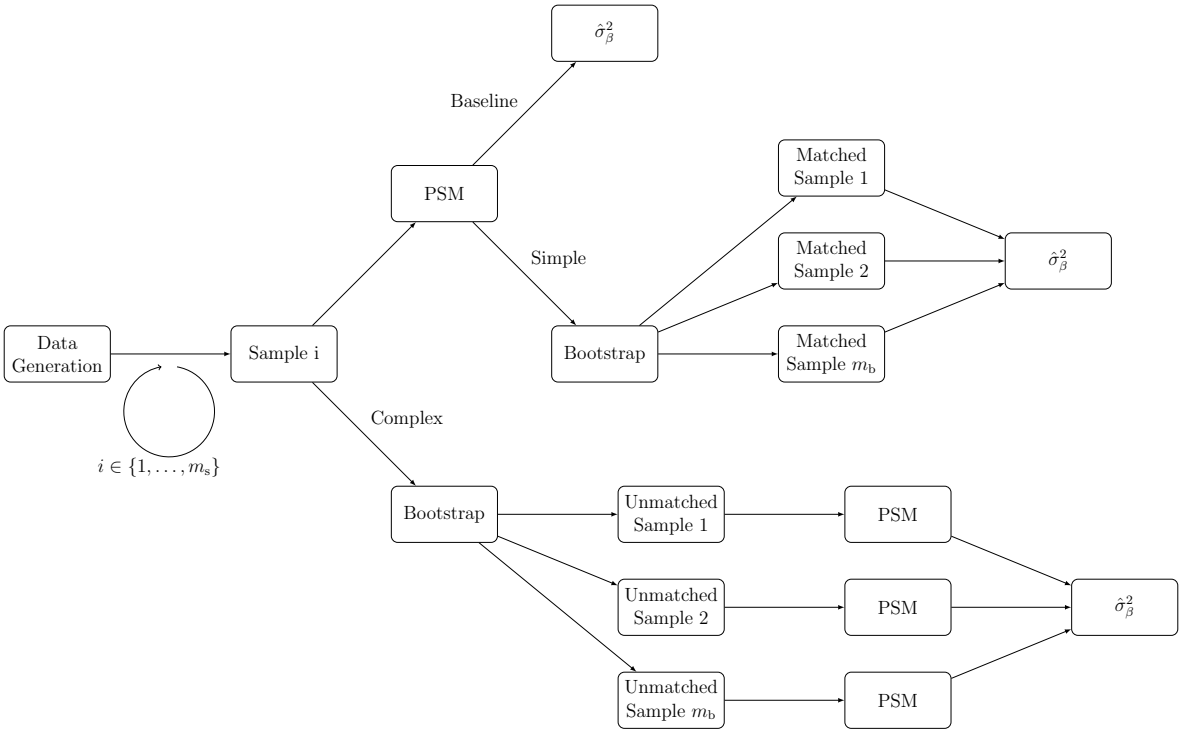
6.2 Creating Diagrams with the Tikzpicture Environment

The TikZ package is a more advanced package that allows users to *create* visualizations. The syntax is pretty technical, so you might want to start with a template from the web as you first get started. However, as you become accustomed to the syntax, you can make very beautiful visualizations with a lot of precision and flexibility.

As a simpler example, let's construct a causal DAG with two confounders.

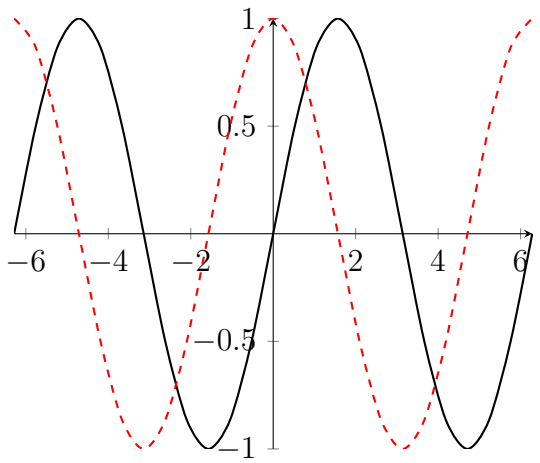


The above diagram is a representation of a mathematical *graph*: a collection of nodes and edges. We can define the relative position of nodes (the easy part), whereupon LaTeX will output their position absolutely on our document (the hard part). Again, while the syntax appears tedious at first, we quickly see how powerful this tool is in making professional-looking output. Now let's consider a much more involved diagram, just for fun!



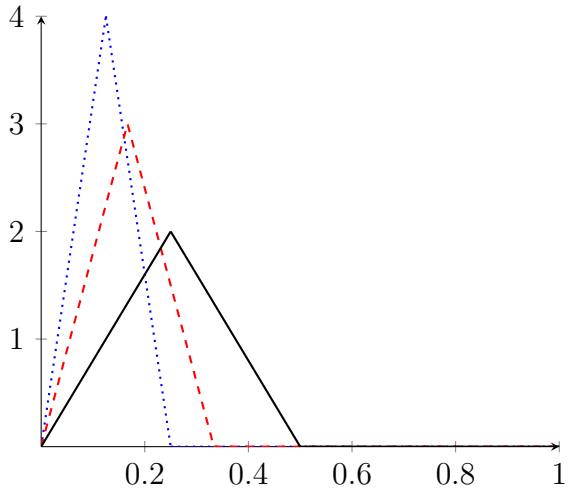
6.3 Creating Graphs with the Tikzpicture Environment

The pgfplots package extends the TikZ package to make creating plots of functions a breeze.



Above, we have plotted familiar sine and cosine functions over a manually specified domain. The syntax is much cleaner than that of the base version of the tikzpicture environment.

By restricting our domain with an `\addplot` call, we can also write piecewise functions, as shown below.



This is by no means an exhaustive characterization of visualizations that you can make in LaTeX! As mentioned earlier, there are hundreds of templates available online to help you get started. Again, this is one of the more technical aspects of this program, so go slow and always read your error messages!