LaTeX: A Brief Introduction

Andrew Q. Philips*

September 12, 2019

*Assistant Professor, Department of Political Science, University of Colorado at Boulder, UCB 333, Boulder, CO 80309-0333. andrew.philips@colorado.edu. www.andyphilips.com. Earlier versions of this paper came from the political science graduate student "Bootcamp" course held at Texas A&M University in 2015 and 2016.

Contents

1 Introduction

This paper is designed to introduce LaTeX, a typesetting program that can greatly ease production of academic articles, presentation slides, and many other applications. LaTeX (pronounced "lay-tech" or "lah-tech") has grown in use from its creation in 1985, and is already very popular in academia. LaTeX does have a fairly steep learning curve, but it is well worth knowing. In fact, once you create your own basic template, the start-up costs diminish substantially. The purpose of this paper is to show you the capabilities of LaTeX—from there, you can find vast amounts of help documentation online. This is a quick overview that you may find helpful to revisit if you come upon this problem in your own work (for instance, making a table or needing to create a list of references).

LaTeX differs from Microsoft Word in many ways. The largest is the user-interface. In Word you see the end result as you type and change the formatting—such a text compiler is known as a WYSWYG (What You See is What You Get). In LaTeX on the other hand, the user types directly into a panel similar to notepad, not worrying about formatting or appearance. Instead, preferences are programmed as commands that can change the appearance. Think of it as similar to writing in a R script or Stata .do file; you type the commands in the script, but you need to "run" the file in order to see the output in the results window.

For some examples of the capabilities of LaTeX, I might desire bold fonts, colors, or huge fonts in my text—and no button is needed for changing this formatting. LaTeX even works with bibliography packages in order to compile the citations used in your papers using a painless process. If you have ever tried to do this in Word, you know it can be frustrating.

There are many other things that make LaTeX ideal over WYSWYGs. Kerning, or the spacing of combinations of letters that become forced apart due to their shape (e.g., AV, AY, AT, PA,...) is performed automatically in LaTeX but not in MS Word by default. LaTeX also creates "real" capitals (capitals are a different type shape rather than just re-sized), and provides the ability to use characters from other languages rather easily (e.g. ô ,

á, ç, å). Given that you may work with and cite researchers with these special characters in their names, this is a big help. Mathematical symbols are incredibly common, and it is very easy to write characters like δ, Δ, or ε with LaTeX. It also has an advanced algorithm for dealing with white-space and line breaks, has lots of compatible fonts, and a substantial community of users who help each other online.[1] Moreover, figures, tables and equations are auto-updated as you add/remove them and fit 'naturally' in the text; if you tried to do this in Word you may have experienced something like Figure 1.

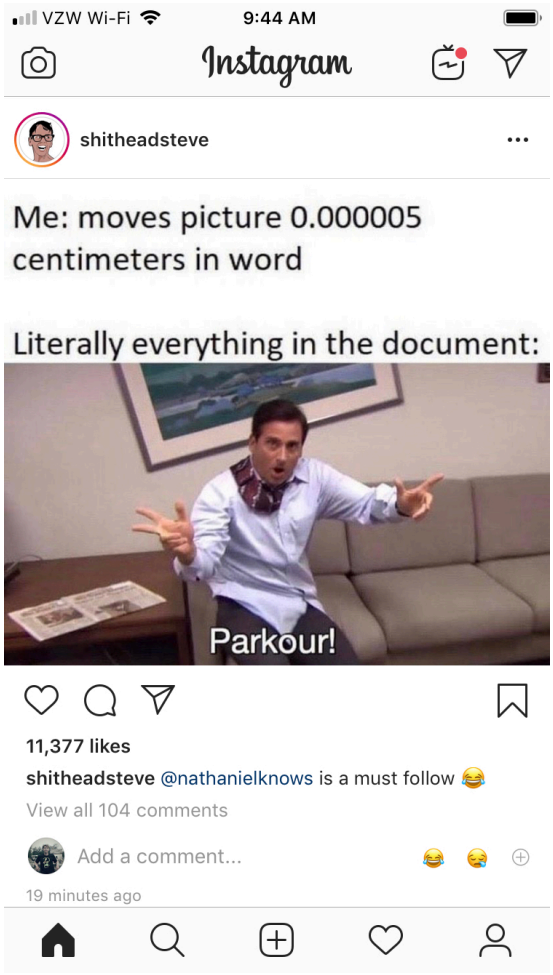


Figure 1: Parkour in Word

The best part about LaTeX is it is free. Anyone can download it from the main website: http://latex-project.org/. In addition, there are copious amounts of help online; often, the only troubleshooting you need to know is the right search term to type into Google!

LaTeX is not without critiques. As discussed above, the start-up costs are quite high

[1] Some benefits of LaTeX over WYSWYG's can be found at http://nitens.org/taraborelli/latex.

compared to WYSWYG editors. Moreover, experimental evidence suggests that even expert LaTeX users have higher error rates and are able to write less text in a given amount of time than novice Word users (Knauff and Nejasmic 2014).[2] However, LaTeX users report much higher levels of satisfaction with their typesetting package than Word users. Equations can be written faster in LaTeX. In addition, once you have constructed a particular template (e.g. for a document, table, or figure), all you really need to do is copy and paste.

Given that many of your coauthors (e.g., professors, advanced graduate students) will use LaTeX—and some might even require that you use it—I would suggest taking the time to learn it. I write nearly all my syllabi, papers and presentations using LaTeX. It is better to take the time to learn this early in grad school so you can take full advantage of it. It can even help with fairly complicated series of documents that need to be compiled together, like books or dissertations.

The rest of this document proceeds as follows. We will first discuss basic template setup and typesetting. Next, we will go over the maths capabilities in LaTeX. Then we will discuss inserting tables and figures. We will cover bibliographies next. Finally, we will discuss Beamer, which is akin to the LaTeX version of Microsoft PowerPoint.

2 Creating your First LaTeX Document

2.1 Downloading LaTeX

As we will discuss below, there are essentially three components needed to run any TeX on your computer; (1). a TeX distribution, (2). an application needed to compile and edit documents in TeX, and (3). packages needed to customize your document.[3]

If it is not already on your computer, TeX must be downloaded from online. The

[2] This study was not without its critiques. For a good summary, see here.

[3] These will need to be updated occasionally. Applications can be updated via the application themselves. Packages can be updated, but this differs slightly based on your operating system (see here for more details). The large TeX distribution itself only gets a major update once a year, and needs to be updated by re-downloading the entire distribution as described above.

easiest way to obtain it is by going to the core website http://latex-project.org/, and clicking on "Obtaining Latex". If you have a Mac, click on the MacTeX link. This brings you to the MacTeX website, where you need to download the most current distribution. This installs a number of programs on your computer under a TeX folder in Applications. The file is quite big and can take some time to download.

If you have a Windows computer, you will need to install proTeXt, which is based off of MiKTeX, and available here. Then you can run the setup.exe extension to install it to your computer.

You'll also need to have an application needed to compile and edit documents (step 2 above). If you have Windows, you might try the following:

- TeXstudio is a free open source application: http://texstudio.sourceforge.net/
- TeXworks is another common free open source application that many prefer for its simplicity: http://www.tug.org/texworks/
- WinEdt is a paid application
- There are a number of new online editors where you don't have to actually install TEX on your own computer, and you can live-update and collaborate with others (the latter of which you typically have to pay for), for example, using ShareLaTeX or Overleaf
- If you're a hard-core text/code editor user (Emacs, Vim, Sublime Text, Visual Studio, TextMate...), many of these apps can compile TEX code as well.

If you have a Mac:

- TeXstudio is a free open source application: http://texstudio.sourceforge.net/
- TeXworks is another common free open source application that many prefer for its simplicity: http://www.tug.org/texworks/
- I write nearly all of my own TEX on Texpad, which is a paid ($\approx$ \$25) application.

- If you're a hard-core text/code editor user (Emacs, Vim, Sublime Text, Visual Studio, TextMate...), many of these apps can compile TEX code as well.

Below, we'll stick with TeXworks since it's simple and works on everyone's computers.

2.2 Preamble and Packages

Now that we have LATEX installed and opened up a blank window, we can create our first document. Unlike Word, you need define a number of things in a preamble that tells LATEX how you want your document to look. This is most commonly done using the \usepackage{} command. Similar to R, we need to load packages in order for the typesetting engine to use them.

Here is how our preamble will appear (note: anything in this handout that appears as Courier font sandwiched between two horizontal lines—such as the text below—is what I actually typed in LATEX):

```
%
% A simple LaTeX template designed for the Political Science
% Graduate Student Programming "Bootcamp"
% Designed by: Andrew Q. Philips
%
% ------- Create Preamble ------------
\documentclass [12pt]{article}
\usepackage[a4paper]{geometry}
\usepackage{amsmath, amsthm, amssymb, amsfonts}
\usepackage{graphicx,epsfig}
\usepackage{booktabs}
\usepackage{pslatex}
\usepackage{caption}
\usepackage{setspace}
```

\usepackage{hyperref}

\usepackage{multicol, multirow}

What do all these things do? Here we go:

- the % is LaTeX's version of commenting out a line. Anything after the % will not be read by the text compiler; this is an easy way to add notes or to say what a particular package does. In my typesetting package, this appears in red, which makes it easy to spot.
- \documentclass [12pt]{article} says we want an "article" style with 12 point font. This command is always required. Unless you're creating a presentation or writing a book, article is the most common style.
- \usepackage[a4paper]{geometry} defines the type of paper we want using the geometry package. Changing this would change the margins.
- \usepackage{amsmath, amsthm, amssymb, amsfonts} is a suite of math packages that let us use the align environment to write equations, and other helpful math things such as symbols. We will go over this in the "Math and Symbols in LaTeX" section.
- We need the \usepackage{graphicx,epsfig} packages to insert figures into our document of various sorts.
- \usepackage{booktabs} is needed to produce tables that are not ugly (i.e. it gets rid of vertical lines).
- \usepackage{pslatex} lets us use the Times New Roman font.
- \usepackage{caption} lets us label figures.
- \usepackage{setspace} lets us use double-spacing and single-spacing.

- \usepackage{hyperref} lets us create hyperlink references that you can click on to take you to a website. For example, clicking on this link brings you to my website.
- \usepackage{multicol} gives us the multi-column and multi-row environment which is helpful for tables.

This is only a small sample of the packages available to you. But these ones are probably the most important. Typically I just copy and paste one of my existing preambles when creating a new document. As such, it has gotten to be rather large; I have about 30 packages and 15 custom commands. This typically does not present any problems. However, there are times when LaTeX will not compile because two packages do not work well with one another. So in general, pursuing the "Occam's razor" approach to packages is good practice. If you don't know what a package is doing, you may not need it. Rather than deleting it though, you should comment it out using %. I also always try to add a comment after the \usepackage command that tells be a bit about what the particular package is doing.

2.3 Starting the Document

Now that we have the preamble set up, all we have left to do is create the title/author page. This part goes in the preamble, just below what we typed above. Here is a (somewhat simplified) version of what I did for this paper:

```
\singlespace
\title{\textbf{My Title: A Brief Introduction}}
\author{Andrew Q.\ Philips\footnote{Assistant Professor,
  Department of Political Science, CU Boulder.}}
\date{{\normalsize \today}}
```

We first say we want single-spacing for our title and author details by using \singlespace. Next, we create our title in the \title{ } environment. I wanted a bold text, so I wrapped my title in \textbf{ }. In general, you perform operations (e.g., bold, italicize, color) on text by issuing a command with a backslash and then wrapping the desired text in curly brackets: \command{ }. If I had wanted italics, I could of used the \emph{ } command instead.

Next, I type my name under \author{ } section. Note how I use a backslash after the Q., this is so LaTeX understands this is not the end of a sentence and so does not include additional space after the "." (it automatically assumes a "." is the end of a sentence unless you tell it otherwise). Immediately after my name (but before we close the author bracket) we add the command to produce a footnote. Moving ahead, anything that we put between \footnote{ } in our document gets put in a footnote at the bottom of the page.[4] Last, we say that we would like the date shown, and LaTeX knows today's date if we type \today. We asked for this in \normalsize, which is one of the font sizes.

Next we start the document by the following commands:

```
\begin{document}
\maketitle
\newpage
\tableofcontents
\doublespace
\newpage
\section{Introduction}
Here is where we start typing.....

Here is more typing.

\end{document}
```

[4]For instance, this is a footnote.

All documents start with \begin{document} and must conclude with \end{document}. Next we make the title we just created using \maketitle. Then on a new page (by using the \newpage command) we generate the table of contents. Not all papers have a table of contents, so you could just comment this out with % if you would like. Next we declare a double-space environment, and start a new page. Last, we give the beginning of our paper a large section title called "Introduction". Then we start typing our paper. The last thing we need to do is close our paper with the \end{document} command. Similar to all documents starting with \begin{document}, all must end with \end{document}. In fact, we could type after \end{document} and none of the text would get compiled into the document.

If all of this above seems complicated, it's because it is. Remember that we typically never need to change the preamble other than small text changes (names, titles). Structuring the main body of the text should make much more sense, and it will be where you spend most of your time writing.

2.4 Sections, Subsections, and Sub-Subsections

As shown in the last section, we defined a new section by using the command \section{ }. We can create a smaller title nested within section by typing \subsection{ }, and the smallest title is \subsubsection{ }. Note that the default is to number the section, subsections and subsubsections (as shown in this paper). This is nice because if we make a table of contents, LaTeX will hyperlink us to that spot in the text if we click on the section.[5] However, if we wanted to not have these numbers, we would type \section*{ }, \subsection*{ }, or \subsubsection*{ }. The asterisks are LaTeX's way of knowing to not show the numbered sections.

[5] This can also be done with external links as well as citations in the text.

2.5 Itemize and Enumerate

Lists (i.e. bulletpoints) are sometimes important when creating documents. In LaTeX these are called "itemize" lists. We can create itemized lists the way I just did by using the itemize command. For example, typing the following in my TeX document:

```
\begin{itemize}
    \item Here is one item
    \item Here is another item
    \item Here is another item
    \item Here is another item
\end{itemize}
```

---

produces the following list:

- Here is one item
- Here is another item
- Here is another item
- Here is another item

We use \begin{itemize} to start the itemize environment. For each new bullet-point, we need to type \item and then our text. At the end we stop the environment by typing \end{itemize}. If we wanted to create numbers, we would replace the itemize with enumerate:

```
\begin{enumerate}
    \item Here is one item
    \item Here is another item
```

```
    \item Here is another item
    \item Here is another item
\end{enumerate}
```

And the result would be numbered:

1. Here is one item
2. Here is another item
3. Here is another item
4. Here is another item

We can also nest enumerates or itemizes within each other—for instance typing:

```
\begin{enumerate}
    \item Here is one item
        \begin{enumerate}
            \item This item is nested!
            \item This item is nested!
                \begin{enumerate}
                    \item This item is double-nested
                \end{enumerate}
        \end{enumerate}
    \item Here is another item
    \item Here is another item
        \begin{enumerate}
            \item This item is nested!
            \item This item is nested!
                \begin{enumerate}
```

```
                \item This item is double-nested
            \end{enumerate}
        \end{enumerate}
    \item Here is another item
\end{enumerate}
```

Gives us output that appears as the following:

1. Here is one item
    (a) This item is nested!
    (b) This item is nested!
        i. This item is double-nested
2. Here is another item
3. Here is another item
    (a) This item is nested!
    (b) This item is nested!
        i. This item is double-nested
4. Here is another item

As is clear from this example, it can start to get pretty confusing where itemizes and enumerates begin and end. Therefore (as with any coding program), it is a good idea to tab nested commands to keep them organized.

2.6 Typesetting

After we have set the preamble and written our text, the next step is to compile our LaTeX file into a PDF. This is done using the typeset command up in the menu bar. Remember

that we may need to press compile a few times in order to get all the equation, figure, and table numbers to compile properly. If you are typesetting for the first time, this may take awhile—LaTeX needs to read all of the graphs and preamble commands. Typically, all typesetting programs have a console that shows what it is thinking and will let you know if it gets stuck on something and fails to compile. If you are having trouble, make sure that the compiling engine is set to pdfLaTeX.

Unfortunately, the error message you get if your file does not compile is not typically very helpful. In my experience, if it is hard to diagnose where the error is occurring, sometimes it helps to remove all the text and add it half of it back in, see if it compiles, add half of the remaining text back in, compile,...etc. This way you can isolate out the trouble paragraph or section. By far, the most common error that I have seen is forgetting to add a closed curly bracket.

3 Math and Symbols in LaTeX

One of the biggest advantages of LaTeX is its ability to write complex mathematical formulas relatively easily. In this section we will go over some of the most common ones as well as how to present equations. Consider a relatively simple equation:

$$E = mc^2 \tag{1}$$

```
\begin{align}
E=mc^{2}
\end{align}
```

How did I make this? We first used the "align" environment to tell LaTeX that we had

an equation.[6] We then typed the equation, using the carrot symbol to get the superscript. Anything that we put inside the curly brackets after the carrot gets superscripted, and anything left outside does not. By default, LaTeX will assume the first character after a superscript or subscript is meant to be superscripted/subscripted, but if you want more characters you will need to use the curly brackets. For instance, I can make the 2 turn into $(2+5x)$ through the following:

$$E = mc^{(2+5x)} + 1 \tag{2}$$

```
\begin{align}
E=mc^{(2+5x)}+1
\end{align}
```

3.1 Referencing

Notice also how LaTeX automatically numbers our equations. We can reference these equation numbers by using the "label" and then "ref" commands as follows:

$$E = mc^{(2+5x)} + 1 \tag{3}$$

```
\begin{align}\label{eq:myfirstequation}
E=mc^{(2+5x)}+1
\end{align}
```

---

[6] Note that we could have used the "equation" environment instead of align. To the best of my knowledge, the only difference is that the equation environment allows you to break up the equation if it runs across the entire page, although see this link for a debate.

Now anytime we type:

```
\ref{eq:myfirstequation}
```

---

in our text (not in the align environment) we get a label for Equation 3. This is live-updated; if we were to delete an equation or move it somewhere else, LaTeX will recognize that the order numbering has changed and update the numbering.[7] This is a huge time-saver, especially in long papers, since you no longer have to go through and change equation numbers in the text. We just have to make sure that each new equation gets a new label, so having a labeling system like eq1, eq2, eq3...is helpful.

Note too that all equations are referenced by \ref{eq:something} . The eq: part is necessary to let the program know this is an equation, but the "something" part can be any label you want to give it. Below we will see that labeling figures or tables works in the same way, but instead of eq: we need fig: or tab:, respectively.

3.2 Subscripts

As with superscripts, a subscript command let the typesetting program know you want everything within the following brackets after the underscore in subscripts. It looks like the following:

$$E_t = m_t c_t \tag{4}$$

```
\begin{align}\label{eq:mysecondequation}
E_{t}=m_{t}c_{t}
\end{align}
```

[7] Sometimes if your paper has not compiled in some time, or you had an error last time, you will get ?? instead of a number. Just keep re-compiling and it will take care of itself.

See how everything in Equation 4 has been subscripted by t. We could always combine both super- and subscripts:

$$E_t = m_t c_t^{(2+5x)} \tag{5}$$

```
\begin{align}
E_{t}=m_{t}c_{t}^{(2+5x)}
\end{align}
```

Note that the order of the carrot and underscore after c does not matter, but the curly bracket rule is still in place; all brackets must be closed and everything within the bracket gets used by the preceding command. And, all superscript commands and subscript commands should come right after the letter or symbol (e.g., no m _{t}).

3.3 Fractions

Fractions are easy to create. The first set of curly brackets is the numerator, the second, the denominator:

```
\begin{align}
\frac{1}{2}
\end{align}
```

will produce the fraction:

$$\frac{1}{2} \tag{6}$$

As with the superscripts and subscripts, we can add more complicated commands into the numerator or denominator, just remember those curly brackets:

```
\begin{align}
\frac{\frac{1}{3}}{2^{x}}
\end{align}
```

$$\frac{\frac{1}{3}}{2^{x}} \tag{7}$$

3.4 Greek Symbols

In scientific writing, symbols (commonly Greek) are often used. They are relatively simple to add, as long as you remember the name. Capitalizing the Greek letter will call its upper-case Greek counterpart, if applicable. For instance:

$$\alpha + \beta + \gamma + \delta + \varepsilon + \mu + \theta + \Delta + \Sigma \tag{8}$$

```
\begin{align}
\alpha + \beta + \gamma + \delta + \epsilon + \mu + \theta + \Delta + \Sigma
\end{align}
```

---

are some of the most common greek letters. Note how Delta is capitalized in its second occurrence in order to get the upper-case version. We can add things like superscripts and subscripts directly to these symbols, e.g., β_s.

A quick Google search can help you find all the others, as well as other symbols to use in LaTeX—like $\leftarrow$, $\neq$, $\forall$, or ∞. For instance, some helpful websites for symbols are:

- https://www.sharelatex.com/learn/List_of_Greek_letters_and_math_symbols
- http://omega.albany.edu:8008/Symbols.html

Depending on your typesetting distribution, you may even have hotkeys that you can press to include these symbols right into your text (i.e. MiKTeX in Windows). If not, you will have to look up/remember these symbols each time. Thankfully, many of these commands are intuitive...i.e. the \Delta command makes a Δ. Note that we can include these in fractions, add superscripts/subscripts to them, and so on—in other words, Greek letters can be treated just like any other character in the equation.

3.5 Equations in the Text

So far we have given each equation a new line and its own number using the align environment. Other times we may want to reference or create a specific math symbol or equation in the text, such as β, or $c^{(2+5x)}$. To do so, we enclose them by using the $:

For instance, the sentence such as β, or $c^{(2+5x)}$ can be written as:

For instance, the sentence such as β, or $c^{(2+5x)}$ can be written as:

Failing to add the $ or curly brackets to either side is one of the most common reasons why LaTeX fails to compile (it will issue an error in the console like "missing $ inserted"), so be sure to write these correctly.

To conclude, LaTeX makes it really easy to write complex mathematical formulas like:

$$Z_{it} = \alpha_{it} + \delta X_{it} + \frac{\gamma X + e^{3X+\varepsilon}}{(1-\sqrt{c_{it}^2})} \tag{9}$$

Just remember to keep track of those curly brackets:

```
\begin{align}
Z_{it} = \alpha_{it} + \delta X_{it} + \frac{\gamma X +
 e^{3X + \epsilon}}{(1-\sqrt{c_{it}^2})}
\end{align}
```

Or, a complex example from one of my lectures. Here, we use the split command to split this equation across multiple lines:

$$\begin{aligned} \ln L(\mu,\sigma^2|y,k_{Il},k_{Iu}) = & \sum_{y_i\notin\mathscr{I}} \ln\left(\sigma^{-1}\phi\left(\frac{y_i-\mu}{\sigma}\right)\right)+ \\ & \sum_{y_i\in\mathscr{I}} \ln\left(\Phi\left(\frac{k_{Iu}-\mu}{\sigma}\right)-\Phi\left(\frac{k_{Il}-\mu}{\sigma}\right)\right) \end{aligned} \tag{10}$$

```
\begin{equation}
    \begin{split}
    \text{ln}\ L(\mu,\sigma^2| y,k_{Il},k_{Iu}) = & \sum_{y_i\notin\mathcal{I}}\text{ln}
    \Bigg(\sigma^{-1}\phi\left(\frac{y_i-\mu}{\sigma} \right) \Bigg) +\\
    & \sum_{y_i\in\mathcal{I}} \text{ln} \Bigg(\Phi\left(\frac{k_{Iu}-\mu}{\sigma}\right)-
    \Phi \left(\frac{k_{Il}-\mu}{\sigma}\right)\Bigg)
    \end{split}
\end{equation}
```

The easiest way to find the specific name to write complex symbols or commands (like writing a fraction or square root) is to search online for it. Although all of the brackets and parentheses may look daunting, they follow a set of simple rules that make writing mathematical formulae much easier than in other word processors.

3.6 Symbols that are Off-Limits

You may have noticed how important the \ is to LaTeX. If we try to put one in our document by itself, it may not compile.[8] The _ (underscore), the $, the %, the ^, the &, and the #, if included, won't compile. That is because these are key parts of the LaTeX language. For instance, as we saw above, the dollar sign was used to start and end an equation in the text. If you still want to use these special characters (called macro-parameter characters in LaTeX), you will have to put a backslash, \, in order to use them (i.e. \$). In my experience, writing an illegal symbol in the text one of the other most common reasons why LaTeX fails to compile.

4 Inserting Tables and Figures

In the previous section we saw how to insert equations into our paper. In this section we will discuss how to add tables and figures into LaTeX. This is particularly important since much of quantitative social science involves presenting results through the use of tables and figures.

4.1 Figures

Inserting a figure into LaTeX is relatively straightforward. Assume we already created and saved a figure using R or Stata. A good extension to use is as a PDF.[9] Also make sure

[8] Technically, if it precedes a blank space it will compile, but if it abuts a letter or something it might not.

[9] There seems to be some contention as to the best type of file extension to use for graphics. I found a forum that found that, "For photos, screenshots and such bitmap graphics, you could use their native format (to preserve quality) if it's jpeg or png. Otherwise convert those pictures to png or at least to

that we have the packages \usepackage{graphicx,caption} in the preamble. We can then add a figure to our document. For instance, Figure 2 shows a bar graph I made for one of my papers. Here is how it is added into my LaTeX document:

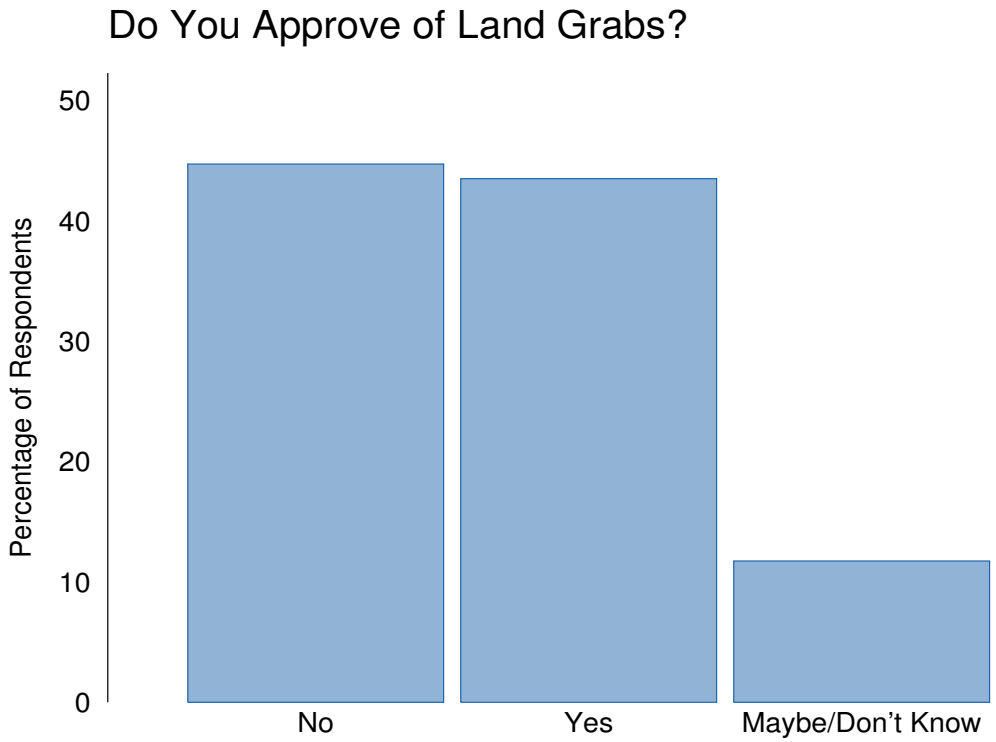


Figure 2: This is Only an Example Figure

```
\begin{figure}[htpb]\centering
 {\includegraphics[width=150mm,height=105mm]{myfigure.pdf}}
\caption{This is Only an Example Figure}
\label{fig:myfirstfigure}
\end{figure}
```

We first have to call the figure by using \begin{figure}. The [htpb] is a LaTeX command

lossless jpg...For drawings, plots and diagrams you better use a vector image format, because it can be scaled without quality loss. Possible vector formats are pdf and ps." Keep in mind that not all extensions work in all compiling types. I think that in general, the best to use are .pdf or .eps.

that lets the graphs float around in order to fit them on the page the best.[10] Next, we tell LaTeX to center the figure using \centering. Then, we (inside a set of brackets) type \includegraphics to define the pdf we want to use. Within square brackets we type the command [width=150mm,height=105mm]; this defines the width and height of the figure. Depending on the figure, you may have to tweak this a bit to make the graph look good.[11] Next we specify the filename within another set of curly brackets, this one is called "myfigure.pdf". Then we label the figure using \label{fig:myfirstfigure} so that we can reference it later. Last, we use \end{figure} to end the figure creation.

One more note; we called the number for Figure 2 using \ref{fig:myfirstfigure} in the text. So it is the same as using \ref for the equations as we saw in the previous section, only now fig comes before the colon instead of eq.

4.2 Advanced Figures and Tips

Sometimes getting the figure width right on the page can be difficult. A quick fix is to use width=0.9\linewidth in the size of the figure section, which makes the width 90 percent of the width of the lines on the page. This is shown in Figure 3. The nice part about specifying only the width is that the height auto-sizes itself, so there is little risk of distorting the figure. I tend to use this command most often when adding figures.

```
\begin{figure}[htpb]\centering
 {\includegraphics[width=0.9\linewidth]{myfigure.pdf}}
\caption{This is Another Example Figure}
\label{fig:myotherfigure}
\end{figure}
```

---

[10] There are others, like [H], that force compile a figure at a certain point in the text. [htpb] is superior, in my opinion, since it lets the figure "float" to the most natural break in the text.

[11] Many books on the graphical presentation of results define the "golden" ratio as a 1.6 to 1 ratio between width and height, respectively. Typically though trial and error works pretty well.

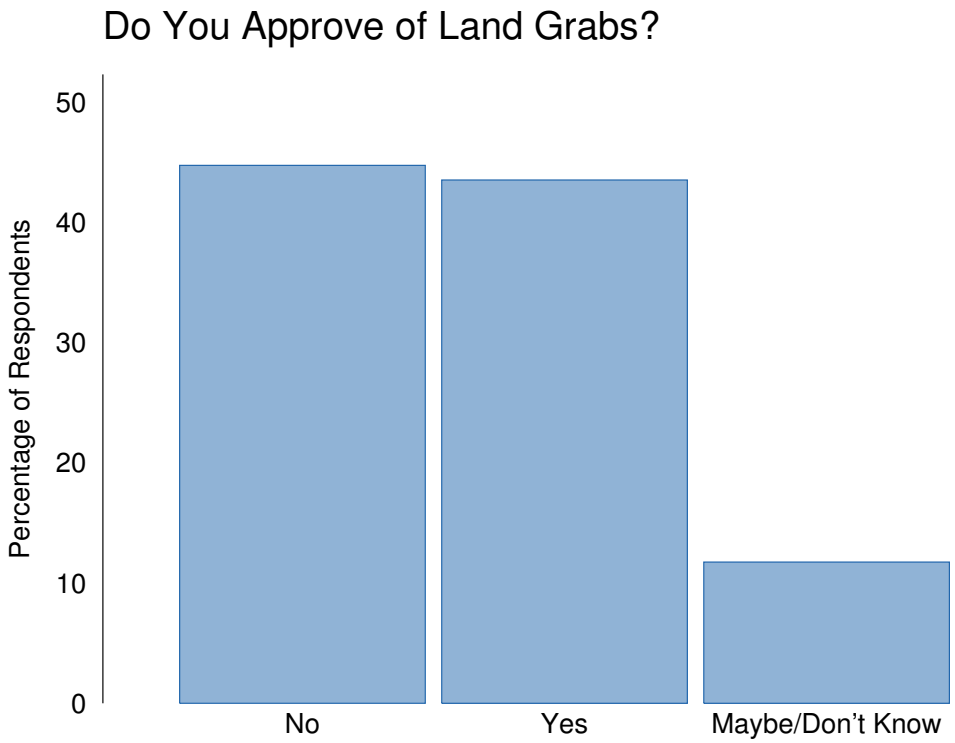


Figure 3: This is Another Example Figure

We may also want to combine figures together. Although you can do this in both Stata and R, by combining figures in LaTeX you can reference each individual figure (labeled as 1a, 1b, 1c...), as well as the overall figure number. We first need to add \usepackage{subcaption} to the preamble. Then, typing the following

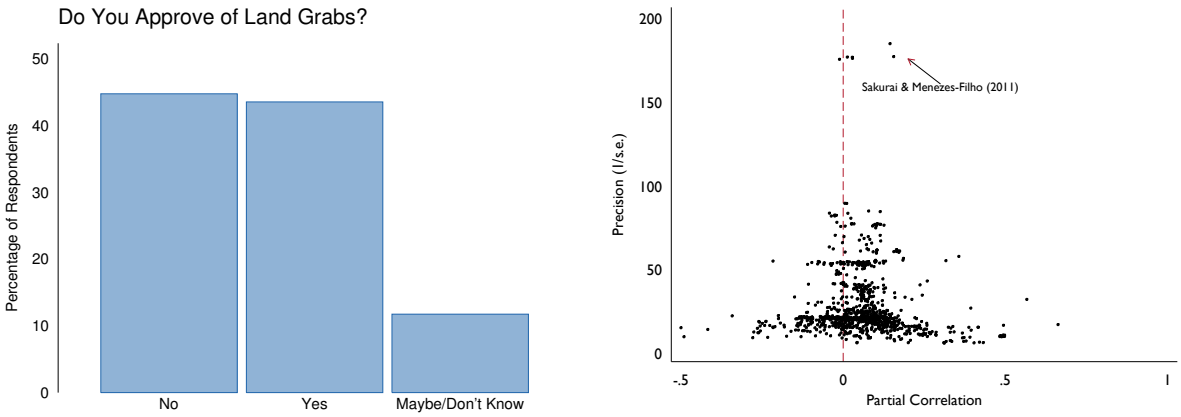
---

```
\begin{figure}[htpb]
\begin{subfigure}{.5\textwidth}
  \centering
  \includegraphics[width=1\linewidth]{myfigure.pdf}
  \caption{This is the First Figure Caption}
  \label{fig:combinedfig1}
\end{subfigure}%    <-- % is IMPORTANT!
\begin{subfigure}{.5\textwidth}
  \centering
  \includegraphics[width=1\linewidth]{mysecondfigure.pdf}
```

```
  \caption{This is the Second Figure Caption}
  \label{fig:combinedfig2}
\end{subfigure}
\caption{This Caption Labels the Entire Figure}
\label{fig:overallfigurelabel}
\end{figure}
```

---

produces Figure 4. We start with \begin{figure}[htpb]. Then we use the command \begin{subfigure}{.5\textwidth} to start one of the subfigures, giving it 50 percent of the total width of the figure. Next we add the figure information just like in a regular figure. It is very important that we put a % after we end the first subfigure. I am not sure why, but without it the figures will be on top of each other rather than next to each other. In the second half we create another subfigure. After we end the second subfigure, we add a caption and label to the overall figure. Now we can reference individual subfigures, such as 4a or 4b, as well as Figure 4.



(a) This is the First Figure Caption (b) This is the Second Figure Caption

Figure 4: This Caption Labels the Entire Figure

Last, for those that would like to draw figures or schematic diagrams in LaTeX, there is a package called TikZ. This is beyond the scope of this paper (and a bit of overkill given that most of these diagrams are easier to make in PowerPoint or Keynote), but if you want to learn more you should check out this link.

4.3 Tables

Tables are a bit more difficult to create by hand than most things in LaTeX. Thankfully, often we can get our statistical packages, like Stata, to do the heavy lifting for us (we saw how to do this in Stata and R). Here is a super-basic table:

Table 1: The Title of My First Table

Column name 1	Column name 2
4.5	3.6
3.2	1.3

We create Table 1 by the following:

```
\begin{table}[htbp]
\caption{The Title of My First Table}
\begin{center}
\begin{tabular}{c c}
Column name 1 &  Column name 2 \\\hline
4.5  &  3.6  \\
3.2 & 1.3  \\\hline
\end{tabular}
\end{center}
\label{tab:myfirsttable}
\end{table}%
```

As with creating figures, we first define the environment through \begin{table}[htdp], using the [htdp] environment to let our table "float" through the pages. We then define the \caption{}, or title. Next we start a centering environment. The next part of creating a table is to start a \begin{tabular}{c c} environment. The c's in curly brackets mean that we want two columns in our table, centered. Some other possibilities include l for

left-aligned or r for right aligned. Note that if we were to expand the table—add more columns for instance—we would need to add the correct number of columns here, each column represented by a “c” or “l” or “r”.

After, we enter in the data for the table. Each entry in the table is separated by a &. Note too that \\ is the command to end the line of the table, and after this command we start a new line, which ends up being a new row in the column. Note that unlike the number of columns, which we did have to specify in the \begin{tabular}{c c} environment, we can just keep adding rows to the table.

At the end of the table, we use \hline to create the horizontal lines separating the variable names. And that is it...we could make as many new rows of numbers as we wanted. For more complicated Table functions (i.e. multiple columns, multiple rows, slimming down columns, etc...see online help).

5 Bibliographies

One of the most difficult parts of writing an academic paper is keeping track of (and creating) all of the citations needed. Thankfully, LaTeX is pretty good at handling this. It uses something called a BibTeX typeset to create references. Unless you are hand-writing in authors at the end of your document (don’t do this please!), most people get some sort of .bib file program.

What is a .bib file? It stores citation information written in a particular way using the BibTeX language:

```
@article{greenwade93,
  author  = "George D. Greenwade",
  title   = "The {C}omprehensive {T}ex {A}rchive {N}etwork ({CTAN})",
  year    = "1993",
  journal = "TUGBoat",
```

```
    volume  = "14",
    number  = "3",
    pages   = "342--351"
}
```

If it looks clunky and confusing...it is. Thankfully, there are a number of free bibliography tools. These programs help look up citations on places like Google Scholar, and then show the content above via a graphical user interface (GUI). The program then saves the file as a .bib, and then we can compile the references in our LaTeX document by making calls to the .bib file.

Graduate students and others that I know primarily use two bibliography programs: BibDesk, or JabRef. I use BibDesk, but it is only available for Mac—JabRef is available for all platforms since it utilizes Java. We will focus on JabRef since it is similar cross-platform.[12] First download the program and open it.[13] It should look something like Figure 5.

Next select a new BibTeX database. Then, under search click "Web search". Select Google Scholar and search for a name. I searched for "Anthony Downs". Then click on any/all relevant checkboxes of works by Downs that you want to cite, and click "OK". A menu pops up that shows what we added. Click "OK" again and now the main screen will appear as in Figure 6.

Of course, sometimes we may not find an article/book we are searching for. Or we may want to enter or change the entry. To create a new entry, click the "Plus" button. Next, select which entry type you want (choose "Article" for this example). Then, fill out the form that pops up with the relevant information. You should not have to fill out entries by hand very often since most work is available on the web.

[12] In fact, there even appears to be a Java applet to run JabRef in your web browser.

[13] I had particular difficulties doing this on my Mac; after downloading the .zip file from SourceForge and clicking on it, I got a "this package is damaged and cannot be opened" window pop up. To override this (the package is not damaged), go to System Preferences, and under Security & Privacy change it so that your computer allows downloaded apps from "Anywhere".

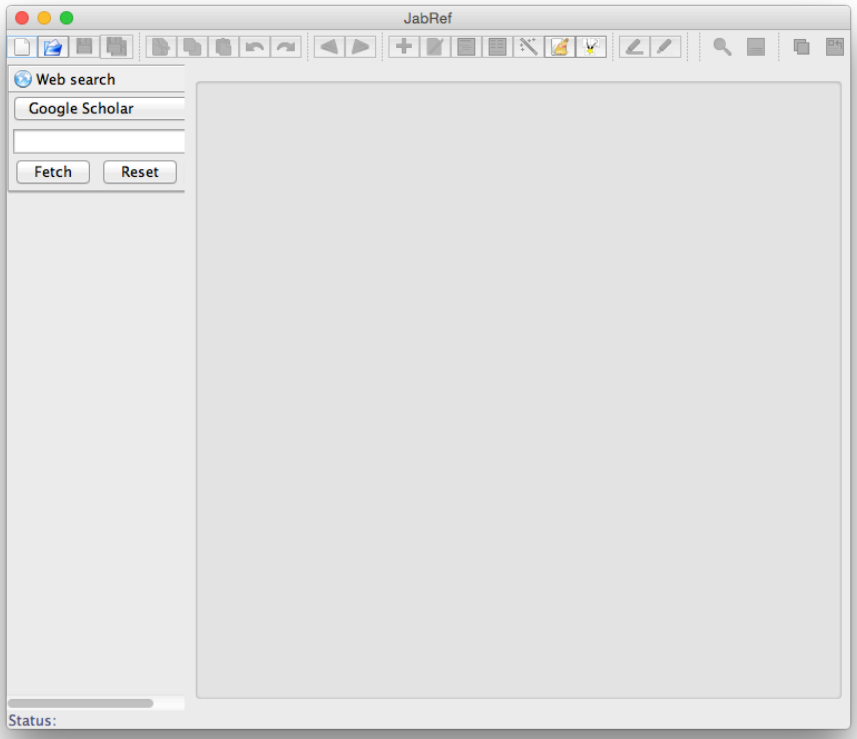


Figure 5: A Fresh JabRef

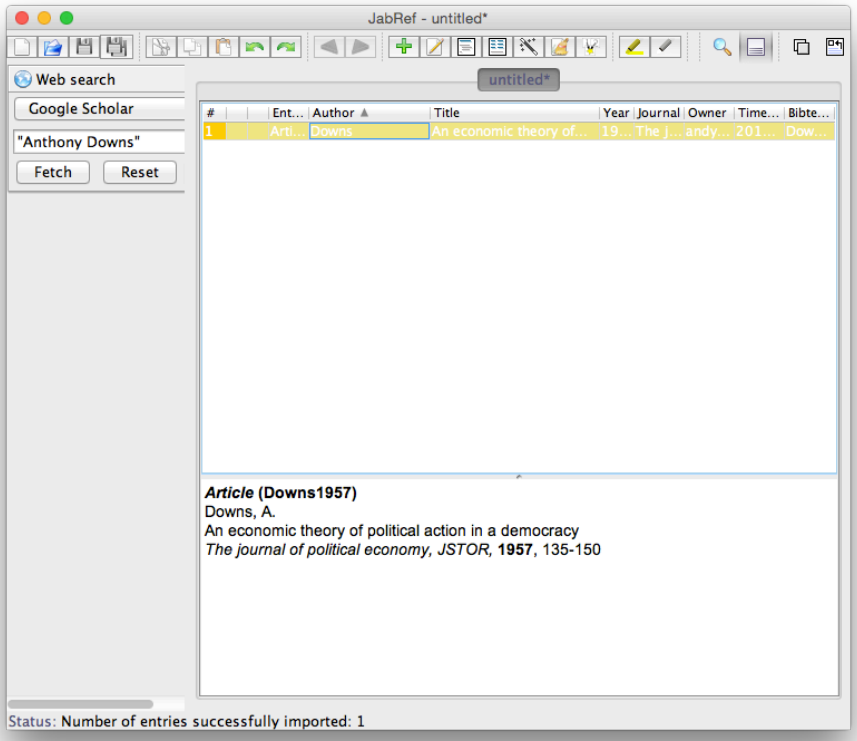


Figure 6: JabRef with One Author

At the bottom of every entry is something called a Bibtexkey. This is very important, since it serves as a unique identifier for each entry. If you have the same Bibtexkey for two or more entries, JabRef will complain. What is a good entry key? The default is to list the first author and the year of publication; for instance Downs1957. In fact, if you click the magic wand button on the menu bar, an entry key is automatically generated. You can, of course, use whatever entry key you like, but it is probably a good idea to keep them short and consistent.

Now we are ready to save our .bib file (the file format that JabRef and other programs use) and put the needed author entries into LaTeX. In the course materials, there is already a .bib file named "bibliography.bib" that has a lot of citations. To cite something, we go to the cite we want. Then we type \cite{ } into our LaTeX document where we want the citation to appear. For instance, \cite{aidt2014voting} produces a citation that appears as: Aidt and Mooney (2014). To make this appear in both our paper and in the references there are a number of steps:

1. First, we need to add the following lines into the preamble:

```
\usepackage{natbib} % need for bibtex
\setcitestyle{aysep={}} % no commas for author-year separation
\bibliographystyle{apsr}
```

The natbib package is required to use bibtex. The next line removes the pesky commas that appear between the authors and years. The third line, \bibliographystyle{apsr}, is the APSR bibliography style. There are a few other bibliography styles, and based off the journal you submit your manuscript to you may have to change this, but these three lines should get you pretty far.

2. Second, the following needs to be added at the end of the document but before the \end{document} line:

```
\newpage
\bibliography{bibliography}
```

The \newpage command puts the references on a new page. The \bibliography{ } command specifies the name of the JabRef .bib file. We need to make sure that the .bib file is in the same location as our LaTeX document. We also need to make sure that we have the correct file name.

3. We need to compile the document in both LaTeX and BibTeX. Make sure you have saved your .bib file. Then proceed as follows:

   (a) First compile LaTeX.

   (b) Then compile as BibTex (you should be able to change the typesetting from LaTeX to BibTex in your typesetting program).

   (c) Now compile as LaTeX at least twice. This part is weird and sometimes appears to take a number of BibTex→LaTeX→LaTeX combinations. You will know if it is successful if you no longer get ? as cites.

The best part of having a .bib file is that it is rather intelligent. For instance, if an author has more than one article in a year, the program automatically adds year-a and year-b to the citations and references...for instance: (Philips, Rutherford and Whitten 2016a,b) was generated by \citep{philips2016dynamic,philips2016sj}. Notice how this citation looks different. by using \citep{ }, we can get citations surrounded by parentheses. By using a comma to separate entries, we can get multiple citations separated by a semi-colon.

There are a number of ways to generate citations to include parentheses, page numbers, and so on. Here are most of them:

- \citet{aidt2014voting} appears as: Aidt and Mooney (2014).

- \citep{aidt2014voting} appears as: (Aidt and Mooney 2014).
- \citep[][pp. 412]{aidt2014voting} appears as: (Aidt and Mooney 2014, pp. 412).
- \citep[e.g.][pp. 412]{aidt2014voting} appears as: (e.g. Aidt and Mooney 2014, pp. 412).

To sum up, bibliography programs such as BibTeX and JabRef greatly increase your efficiency when creating documents. You are less likely to make mistakes when citing authors, and can use the same .bib document so you don't have to keep searching for entries on Google Scholar. You can even link the actual .pdf articles to JabRef, so it serves as a sort of "document warehouse".

6 Beamer

Beamer, like PowerPoint, is used to make presentations. Lots of people like it because it is very no-nonsense; no moving parts, works no matter the LaTeX distribution, and, like with documents in LaTeX, you only have to set up the formatting in the preamble to have it throughout the rest of the presentation. With a few exceptions, it is actually very similar to LaTeX. It saves the document as a .pdf, which can be read by every computer—this is ideal since some computers may not have the capability of reading a PowerPoint .ppt, or may have a depreciated version that cannot handle all the slide animations; in contrast with a Beamer file you just open up the .pdf and go to slideshow mode and everything should work perfectly.

In the course files, you should receive a simple beamer template ("CU-beamer-template") which I created for you. It will be helpful to go through this step-by-step.

Like LaTeX, we have to first set up the preamble, starting with the required document class:

```
\documentclass[mathserif,ignorenonframetext]{beamer}
```

In this case we ask for the beamer document class. Next we use one of the many themes available. I put them all in, but you may want to look them up here to see how they look. I chose \usetheme{Copenhagen} but many of them look similar.

Next we define packages just like we did when creating a document. There are a few beamer-specific ones I added that eliminate a lot of the junk that is the default in beamer (like slide numbers and arrows at the bottom of the slide). Then we create the CU colors using

---

```
% ----------------------------------------
% Define CU colors:
\definecolor{cugold}{RGB}{207,184,124}
\definecolor{cublack}{RGB}{0,0,0}
\definecolor{cudkgray}{RGB}{86,90,92}
\definecolor{cultgray}{RGB}{162,164,163}
```

The text below the CU colors applies it to certain parts of the Beamer file, like how the bottom is in gold and black. Feel free to mess around with these to mix things up a bit. While it is perfectly fine to use different colors (lots of presenters use a simple blue theme), be careful that the colors that you choose are not too bright or too dark for presentations. While that bright yellow you picked out may look great on your laptop, it probably will induce headaches when it is up on the screen.

Next we create the title and author information. Again, this is very similar to the document, except that we include a small mini-title and author information to appear at the bottom of every slide. After, we start the document and create the title page:

```
\title[CU Boulder]{\textcolor{black}{Title Here}}
```

```
\author[\copyright\ Last Name 2017]{
Your Name Here\\ \bigskip
{\small University of Colorado Boulder} }
\date{}

\begin{document}

\frame{\titlepage}
```

If we were to compile the Beamer (which is still compiled in LaTeX, but just a different document class), we should get something that looks like Figure 7.

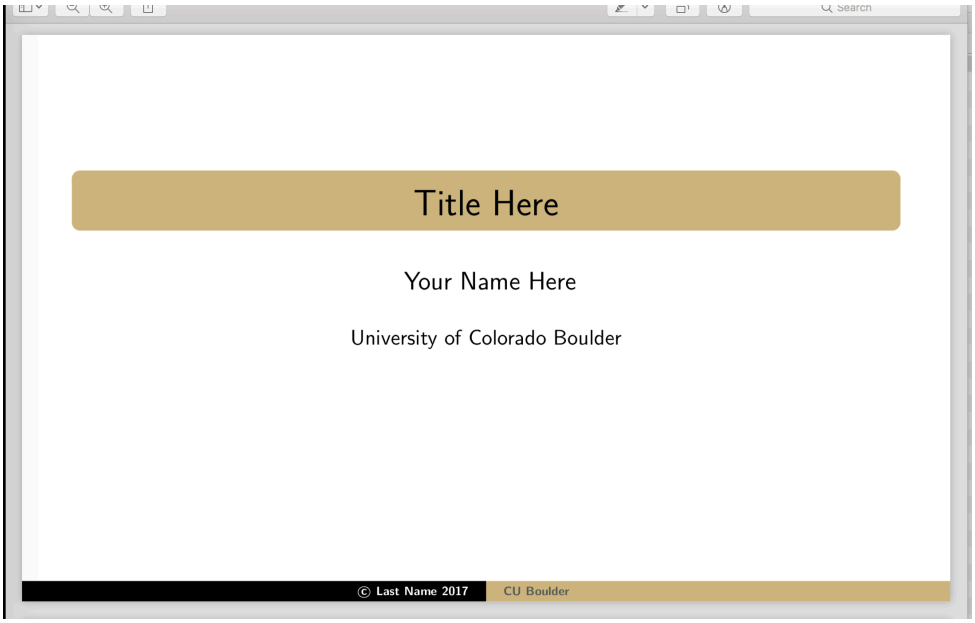


Figure 7: The Title Page of a Beamer Presentation

We can now start creating slides. Typically they appear something like the following:

```
% ----------------------------------------
\begin{frame}\frametitle{Frame title}
```

```
\begin{itemize}\setlength\itemsep{1em}
\item Here's an item\pause
\item Here's another
\end{itemize}
\end{frame}
% ----------------------------------------
```

---

where the frame environment defines each slide. To create a pause (so that you will have to click through once to see the next part) we can use the \pause command. In addition, itemizing or enumerating are commonly done when using Beamer presentations. Note too that we could add a table or graph right into a slide, just like we can do in LaTeX.

Last, remember that the end of the document needs the \end{document} line or else nothing will compile. After that, we can Typeset the document and it should compile into a .pdf that we can open and use in a slideshow format just like any other document.

To conclude, we use Beamer slides as a simple way of presenting academic work. These documents are relatively easy to create (after you have the basic template) and are highly compatible with LaTeX. Like LaTeX they excel at presenting mathematical formulae, graphs, and tables. Other than the "frame" environments and preambles, you write in it just as you would in LaTeX.

7 CVs

LaTeX is a great tool for helping you create an academic CV, since these are typically full of formatting and indentations that are difficult to handle in WYSIWYGs. For some online templates check out: https://www.sharelatex.com/templates/cv-or-resume or http://www.latextemplates.com/cat/curricula-vitae.

8 Other 'TeX' Languages

Everything we have talked about so far has been done in pdfLaTeX, which is typically the default typesetting engine (what compiles your document when you hit typeset). However, there are a number of others that you may want to consider, depending on your needs. Two of the most common alternatives are XeTeX and LuaTeX. XeTeX is similar to pdfLaTeX, but allows much more flexibility in terms of fonts, mostly through the fontspec package. Changing fonts away from the standard ones in pdfLaTeX is quite difficult, in contrast. XeTeX also supports UTF-8 encoded input, but only provides .pdf output (regular pdfLaTeX allows you to produce a .dvi or .ps file as well). LuaTeX is similar to TeX but uses a different scripting language (known as "Lua"). I nearly always use XeTeX, and sometimes pdfLaTeX. I would especially recommend XeTeX if working on a project that needs special fonts and formatting (e.g., CV or book). ConTeXt is another engine, but I do not know much about it. The following link has a bit more on these differences: http://tex.stackexchange.com/questions/36/differences-between-luatex-context-and-xetex.

9 Resources

The goal of this document was to get you up and running with LaTeX. We have covered downloading and setting up the application, typesetting the document, math symbols, and inserting tables and figures. In addition, we discussed why using a bibliography package with LaTeX is a really good idea. Last, we saw the advantages of using Beamer to make our presentations.

While this document is a helpful overview, it is by no means a complete introduction to LaTeX. Even if you frequently run into problems, there are lots of online sources (and fellow graduate students!) to help you out. Often, simply typing into Google something such as, for instance, "latex footnote to endnote" gets you a good answer. Other than the other links referenced earlier in the document, here are some other ones that I find helpful:

- At WikiBooks there is a broad overview of the capabilities and different parts of LaTeX: https://en.wikibooks.org/wiki/LaTeX.
- There is a TeX stackexchange, just like there are analogous versions for Stata and R: http://tex.stackexchange.com/. I estimate that I answer 90 percent of my problems using this website alone.
- There is a handy table-maker at http://www.truben.no/latex/table/ that takes some of the difficulty out of creating tables.
- This is a really cool site that lets you draw the symbol you want to use in LaTeX: http://detexify.kirelabs.org/classify.html.
- Here is a link to more help on the main LaTeX site: http://latex-project.org/guides/.

9.1 Documents Included in this Introduction

There are a number of LaTeX-specific documents that have been included in the Bootcamp course files. You should have the following:

- A .TeX document, "CU-beamer-template", which is a template you can use to create Beamer slides.
- A .TeX document, "simple latex template", which is a template you can use to create a LaTeX document.
- A .bib document, "bibliography", which has about 300 author entries, so you have somewhere to start when building your own library.

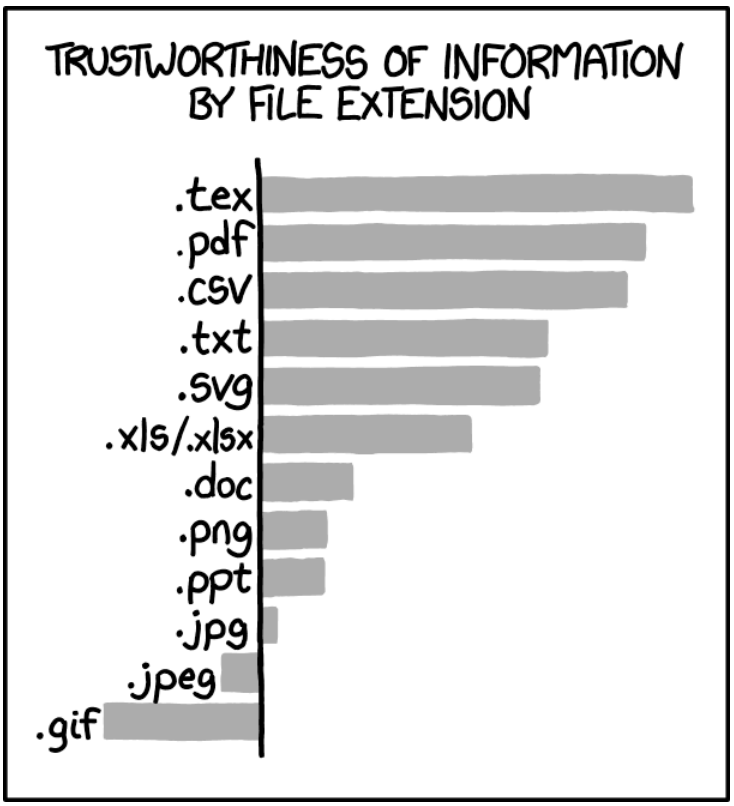


Figure 8: XKCD

References

Aidt, Toke S and Graham Mooney. 2014. "Voting suffrage and the political budget cycle: Evidence from the London Metropolitan Boroughs 1902–1937." Journal of public economics 112:53–71.

Knauff, Markus and Jelica Nejasmic. 2014. "An Efficiency Comparison of Document Preparation Systems Used in Academic Research and Development." PloS one 9(12):e115069.

Philips, Andrew Q, Amanda Rutherford and Guy D Whitten. 2016a. "Dynamic Pie: A Strategy for Modeling Trade-Offs in Compositional Variables over Time." American Journal of Political Science 60(1):268–283.

Philips, Andrew Q, Amanda Rutherford and Guy D Whitten. 2016b. "dynsimpie: A program to examine dynamic compositional dependent variables." Stata Journal .

