The Theory and Statistics Research Laboratory's Introduction to LaTeX: A Short Course

Prepared for the starlab by J.P. Olmsted

Updated: September 12, 2011

Contents

Introduction to this Short Course

About. This document was prepared by the author during his tenure as the starlab Fellow for the incoming graduate student cohort in the Political Science department at the University of Rochester. Previous versions of the short course and the corresponding materials are available at the starlab's website (`http://www.rochester.edu/college/psc/thestarlab/main/index.php`). This document is an adaptation of Arthur Spirling's version. In addition to the supplemental files used for the various exercises included in this version, this document is available from both the starlab's website and the author's website (`http://www.rochester.edu/college/gradstudents/jolmsted/`).

Comments. If you have any comments, questions, or concerns regarding this document please contact Jonathan Olmsted (`jpolmsted@gmail.com`).

Part I
An Introduction to Using LaTeX

1 What is LaTeX (lay-teck) or LaTeX, but not latex or Latex?

- LaTeX is an open document preparation system used by many students, academics, and publishers whose content involves technical topics (*e.g.* math professors, engineers, students of Kevin Clarke).
- LaTeX is a markup language which means that the content and structure are described by "tags." If you go to the University of Rochester's website (`www.rochester.edu`) and view the HTML source for the main page it will look (incredibly) different from how that source code is rendered by your browser. LaTeX is much more similar to HTML source than it is to a word processor (*e.g.* MS Word, OpenOffice Writer, Google Documents). When you work with a LaTeX file, you describe how the file should look, but this is not displayed for you in real-time.

2 Why Would You Use LaTeX?

LaTeX has a number of distinct advantages:

1. A LaTeX document is very pleasing to the eye without the need for any customization.
2. As a graduate student, you mainly produce content and are neither a publisher nor a graphic designer. LaTeX separates content from appearance.
3. LaTeX's ability to incorporate math is un-rivaled.
4. People in this discipline who do things like what you will do will expect it.
5. You will never get locked out from the fruits of your intellectual labor because of licenses or old software. The software is open and the files you create are "flat text" files. If someone has a computer, they can read your LaTeX files.
6. The markup code for LaTeX math is one of the canonical ways of representing math in plain-text environments (*e.g.* email, instant messaging).

3 How Does LaTeX Work?

The process of creating a LaTeX document is quite different from creating one in a word processor. Although most of the technical details can safely be ignored, some understanding of the different components comprising a functioning LaTeX system is helpful and very useful when asking for assistance.[1]

The steps to creating a document are:

1. Have an idea!
2. Create a plain text document containing the content in a text editor.
3. Submit the plain text document to the LaTeX binaries.
4. LaTeX pulls in any additional files that are asked for—but they have to be asked for—and creates, among other files, the document you will print or email.
5. View the result/rendering of your LaTeX document.
6. Be happy.

Notice that this is different from the word processor workflow. In that case, Steps 2, 4, and 5 are basically integrated. Step 3 is unnecessary. Step 6 may or may not happen.

Step 2 In the star lab, the Windows machines have the WinEdt text editor. You could use any number of other text editors, but WinEdt is nice because it integrates a lot of the other tools seamlessly. We typically denote LaTeX files with the `.tex` file extension. Although it isn't necessary in some cases, most software and operating systems will make your life easier if you are willing to name LaTeX files what they expect you to name them.

Step 3 You can achieve this either using the WinEdt interface or issuing a command at the command prompt (in most cases, this is no real choice). TeX is different from LaTeX. The differences aren't important here, but know that the TeX buttons will likely not work.

Also, LaTeX-ing your document can proceed through one of two different branches. I will focus on the `pdflatex` branch and not the `latex` branch. `pdflatex` will convert your `.tex` file into a `.pdf` document. Included images must be `.jpg`, `.png`, or `.pdf` files. `latex` will convert your file into a `.dvi` file which you'll eventually convert to a `.pdf`. Included images must be `.eps` or `.ps` files. There are some advantages to the pdf-less `latex` branch, I prefer the `pdflatex` branch and we will walk through this one together.

[1] This section is meant to be accompanied by the workflow diagram available at `http://www.rochester.edu/college/gradstudents/jolmsted/files/teaching/LaTeX/LaTeX_Workflow.svg`.

Step 4 Sometimes various markup in the LaTeX file will require contributed packages which tell the LaTeX binaries how the content should be rendered. However, making sure these packages are placed in the right spot on your system and up-to-date can be involved. LaTeX package managers simplify this. On the machines in the star lab MikTeX is used. By and large, you can and should ignore this fact in the beginning of your LaTeX experience. What is important is to realize the following:

$$\text{LaTeX} \neq \text{WinEdt} \neq \text{MikTeX}.$$

Step 5 In the star lab, Adobe programs are available to view (and edit) `.pdf` files. This can be achieved through the WinEdt interface.

Step 6 No software necessary.

4 A Simple Task

1. Go to `www.rochester.edu/college/gradstudents/jolmsted/teaching/LaTeX/`.
2. Save the file `FakeFile.tex` to the desktop.
3. Open this file in WinEdt.
4. Run `pdflatex` on the `.tex` file and view it.
5. Run `latex` on the `.tex` file and view it.
6. Notice how many extra files this process creates.

You have successfully created your first LaTeX document and simultaneously learned to always keep LaTeX files/projects in separate directories because the output will create clutter. Consider this a windfall!

Part II
Basic Content in LaTeX

5 LaTeX's Interpretation of Plain Text

- The entire `.tex` file you write will contain only characters on your keyboard. That means, somehow, your using characters on your keyboard you need to represent: a, A, α, $\mathbb{A}$, à, $\mathtt{A}$, $_a$, and Ä.
- Your entire document will be composed of the following characters (including documents utilizing ):

```
a b c d e f g h i j k l m n o p q r s t u v w x y z
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
0 1 2 3 4  6 7 8 9
! @ # $ % ^ & * ( ) _ +
{ } [ ] | \ l ; : ’ " < , > . ? /
```

Special Characters

- The following characters are reserved or *special*:

  `% # $ ^ & _ { } ~ \`

- If you type "`%`" into WinEdt and `pdflatex` your file, you will not get a percent sign.
- In order to generate these symbols you need to use special commands. Although detailed descriptions of these symbols is beyond the purpose of this section, accept the following brief comments:

  `%`: used for including comments and preventing LaTeX from interpreting file contents

  `#`: used to define LaTeX commands (or macros)

  `$`: used to input LaTeX math

  `^`: used for superscripts in math mode

  `&`: used for "tab stops" and alignment and tables

  `_`: used for subscripts in math mode

  `{ }`: used to issue arguments to LaTeX commands

  `~`: used to represent a special kind of space

  `\`: used to start every LaTeX command

Commands

- Using LaTeX efficiently often requires one be familiar with various LaTeXcommands (or macros). In general, this comes only with practice. Commands start with a "\" and they are case-sensitive. Commands can be used to generate particular glyphs or alter the content in some way.
- One command is "`\LaTeX{}`" which produces "LaTeX". Similarly, "`\textbf{election}`" produces "**election**".
- You'll never learn a significant proportion of all the LaTeX commands that people use. However, you'll eventually memorize the ones you use over and over and know how to look up the rest.
- The mandatory arguments to a macro are passed inside curly braces and the option arguments to a macro are passed inside square brackets. The result is something like

  `\command[optional1=1, optional2=2]{mandatory=Always}.`

Whitespace and Spacing

- 2 or more carriage returns, "\\", and "`\linebreak`" break lines
- "`~`" is a non-breaking space
- "`\par`" will create a new paragraph for the succeeding text, regardless of surrounding whitespace
- "`linebreak`" and "`pagebreak`" are appropriately named
- multiple consecutive spaces are interpreted as one

So,

```
Keith         Poole \\




      once
           proclaimed \\ at~a~breakfast
here in Rochester,\par ``I am unfireable!''
```

becomes

Keith Poole

once proclaimed
at a breakfast here in Rochester,
"I am unfireable!"

6 .tex Input File Structure

- LaTeX `.tex` files have a particular structure. If the file you attempt to compile has an error, your document will either fail to be produced or it will not be compiled in accordance with your intentions.

- First, you must declare the document class:

```
\documentclass[letterpaper, 10pt]{article}
```

or

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

and this is part of the *preamble*. For your purposes, "article" is sufficient for the time being.

- Second, the remainder of the preamble contains explicit calls to outside packages if you require their functionality, the creation of new macros, and setting document properties. After the `\documentclass` command we might see the following lines:

```
\usepackage{fancyhdr}

\pagestyle{fancy}
\cfoot{\thepage}
\title{A Document}
```

which provides the functionality from the "`fancyhdr`" package, sets the "`pagestyle`" to "fancy", places the page number in the center of the footer, and sets the title to "A Document". This is also part of the *preamble*.

- Third, the main content is typed within the "`document`" environment, as in:

```
\begin{document}

%% place content here

\end{document}
```

- Lastly, any characters occurring after the close of the environment will be ignored by LaTeX. This includes special characters and commands.

In its entirety, we have something like

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

\title{An Essay on Rob}

\begin{document}

\maketitle

Rob is a cheery chap whose trumpeted lips are sometimes chapped.

\end{document}

$ breaking \ rules # $ without }{ effect
```

Go ahead and compile this document after you type it out.

7 Output File Structure

Headings

- LaTeX accepts the definition of a document hierarchy and displays headings appropriately. The *article* class accepts the following headings whose order indicates how far down they are in the order: `\part{}`, `\section{}`, `\subsection{}`, `\subsubsection{}`, `\paragraph{}`, and `\subparagraph{}`. If an "`*`" is placed after the name as in "`\part*{}`" then the heading will not be numbered. Otherwise it will. The one mandatory argument is the text of the heading.
- Any headings after the `\appendix` command will be altered to reflect that they do not belong to the main body of text.

Add a `section` and a `subsection*` heading to your document. Name them after your two favorite colors. Compile the code.

8 Environments

- All of the content that will be displayed in your final document is contained in at least one environments are denoted by the `\begin{env}` and `\end{env}` commands. We have already seen the `document` environment.
- We will discuss various environments (and there are many) during the remainder of the week, but the basics are all the same. If you begin an environment, you must end it. When nesting environments, the most recently opened environment must be closed before an earlier environment can be closed.
- The `enumerate` environment creates ordered lists. The `itemize` environment creates unordered lists. The `table` environment is used to encapsulate table-like content.

At the end of your content begin and end an environment by the name of `tiny`. Inside this environment type your favorite thing to shout. Use proper latex quotes (*i.e.* `` ``...'' and not "..."). Compile the document.

9 Error Debugging

You will, invariably, make errors. However, you'll probably never be the first person to make any particular error. The four most common errors are

1. you type a command incorrectly
2. you forget to `usepackage` the package that supplies a macro which LaTeX interprets as if you typed a command incorrectly
3. you don't `end` what you `begin` or do so in the right order
4. you have un-balanced braces/parenthesis/brackets depending on what LaTeX is looking for.

Place the `tiny` environment inside a `center` environment. Now, change one of the `begin` commands to `benign`. Compile it. This is not benign! Look at the error message. Correct this mistake and swap the two `end` statements. Compile the document now. Look at the error message.

10 Solving Problems and Getting Help

- Don't panic.
- Check the usual suspects.
- Google a description of your problem in addition to "latex ctan." Including "latex" without "ctan" is a bad, bad, bad, bad, bad idea!
- Comment out potential sources of the problem until you identify it. Slowly add things back in until you have removed only the offending portion. Can this be fixed?
- Consult one of the LaTeX books in the star lab.
- Consult one of the many great LaTeX sites out there.[2]
- Ask the star lab fellow!

[2] Google either "latex ctan wikibook" or "latex comprehensive symbol". These two are great.

Part III
Creating Full Documents in LaTeX

11 Starting Point

- This tutorial builds off of the previous one which introduced LaTeX commands.
- By the end of the last tutorial we had working documents with environments and headings that were quite pleasing to the eye.
- We will quickly create a very basic document to pick up where we left off.
- Open WinEdt and create a new LaTeX `*.tex` file where you would like it.
- Create a basic document with the following code and check that it compiles.

```
\documentclass{article}

% This is just a comment.
\begin{document}

\section{The Content}

\subsection{Short Content}

\subsection{Long Content}

\appendix

\section{Some Book-keeping}

\end{document}
```

- Go ahead and download the two dummy text files from `www.rochester.edu/college/gradstudents/jolmsted/teaching/LaTeX/`. Place these two files inside the same directory as your new LaTeX `.tex` file.
- As it stands, our compiled document is quite bare. Add the following LaTeX commands:
  - Add `\input{./loremipsum_short.txt}` on a new line directly after `\section{Short Content}`. As long as the downloaded file was placed in the appropriate place, you can safely ignore what this command does.

  - Add `\input{./loremipsum.txt}` on a new line directly after `\section{Long Content}`. Enter this line, again, on a new line. As long as the downloaded file was placed in the appropriate place, you can safely ignore what this command does.

- Compile this document again.

12 Fonts

In LaTeX we can make some very basic changes to the font in with ease. These changes can be made with one of two approaches. The *environment* approach or the *command* approach. In the first, whatever font we change will be inside an environment. In the second, whatever font we change will be the mandatory argument to a command.

Font Styles

When consider **bold face** or Small Caps, we are changing font styles. The basic and most common changes are here:

Style	Environment	Command
normal/default	`\begin{textnormal} ...\end{textnormal}`	`\textnormal{...}`
bold	`\begin{bf} ...\end{bf}`	`\textbf{...}`
italics	`\begin{it} ...\end{it}`	`\textit{...}`
SmallCaps	`\begin{sc} ...\end{sc}`	`\textsc{...}`
underline	:-(	`\underline{...}`
Roman	`\begin{textrm} ...\end{textrm}`	`\textrm{...}`
Sans Serif	`\begin{textsf} ...\end{textsf}`	`\textsf{...}`
`teletype`	`\begin{tt} ...\end{tt}`	`\texttt{...}`

- Place the `\input` command under the "Short Content" section inside a font style environment of your choice. Compile the document.

- Before this environment, but under the "Short Content" section type and complete the following sentence: "If I had a boat, I'd ...".

- Change the style of the font for that sentence and only that sentence by using the command approach. This time, choose a different style from the environment change you made above.

Sizes

Style	Environment	Command
tiny	`\begin{tiny} ...\end{tiny}`	`\tiny{...}`
scriptsize	`\begin{scriptsize} ...\end{scriptsize}`	`\scriptsize{...}`
footnotesize	`\begin{footnotesize} ...\end{footnotesize}`	`\footnotesize{...}`
small	`\begin{small} ...\end{small}`	`\small{...}`
normalsize	`\begin{normalsize} ...\end{normalsize}`	`\normalsize{...}`
large	`\begin{large} ...\end{large}`	`\large{...}`
Large	`\begin{Large} ...\end{Large}`	`\Large{...}`
LARGE	`\begin{LARGE} ...\end{LARGE}`	`\LARGE{...}`
huge	`\begin{huge} ...\end{huge}`	`\huge{...}`
Huge	`\begin{Huge} ...\end{Huge}`	`\Huge{...}`

- Place the `\input` under the "Long Content" inside a font size environment of your choice, but choose a size that is small than the normal size. Compile your document.

13 List Environments

There are three main list environments that you will use: `itemize`, `enumerate`, and `description`.

`itemize` The `itemize` environment allows you to make bullet-ed lists without conveying any order. These can be nested to create any structure desired. However, complex nesting often results in user-mistakes.

`enumerate` The `enumerate` environment allows you to make numbered lists which convey a precise order. The number is automatic. These can be nested to create any structure desired and the "numbering" system used at each level reflects how far in the tree the entry is.

`description` The `description` environment is slightly different. In fact, I am using it here. Instead of each entry being marked with an automatic bullet or number, the user must provide the "term" being described.

- For either `itemize` or `enumerate` your code looks like this, where "`environment`" is swapped out for the correct environment:

```
\begin{environment}
\item An item
\item Another item
\item Still more items!
\end{environment}
```

- For the `description` environment your code is slightly different:

```
\begin{description}
\item[An item] description 1
\item[Another item] description 2
\item[Still more items!] description 3
\end{description}
```

- Create a list in the "Some Book-keeping" section. In this list you should have two items—two of your favorite books. If you are using the `description` environment include the author's name as the description if you know it. In this case, make sure the book title is in square brackets.

14 Metadata: titles, dates, authors, abstracts

\maketitle

- Although our content is already impressively long, the document is lacking all metadata. To address this we will formally introduce the `\maketitle` command
- In the preamble of the file (that is, before `\begin{document}`, but after `\documentclass`), enter the following code:

```
\title{My First Opus}
\date{\today}
\author{Kevin Clarke}
```

- Compile the document. Does anything change?
- Now, add `\maketitle` directly after `\begin{document}`, but on a new line.
- Compile again.
- Comment out the line with the `\date` directive. Re-compile. What happens?
- Uncomment that same line, delete the directive's single mandatory argument and compile. What happens?
- In order to use the `\maketitle` command, a title must be defined. The author and date, however are optional.

\thanks

- Kevin is a humble man and acknowledges the contribution that non-Kevin entities play in his success. To add this to the title we use `\thanks`.

- Include the following line at the end of "Clarke" within the author directive as part of the mandatory argument: `\thanks{The author thanks Rochester winters for being sufficiently ridiculous that everyone is forced to stay inside and work.}`
- You can use `thanks` to include institution information or contact information, too.

`titlepage`

- Because this opus is so substantial, it can be overwhelming to assault the reader with content on the first page. In this case, we'd like to have just a title page.
- Place the `\maketitle` directive in an environment by the name of `titlepage`

`abstract`

- Even with a title, we might want to convey the content of our document in more detail.
- In this case, we'd use an abstract.
- After `\maketitle` but within the `titlepage` environment, create a new environment (which is completely nested) called `abstract`. Within the `abstract` environment, write two sentences about what we have covered in this tutorial, so far. Include the word LaTeX, properly typeset.
- Compile the document.

15 Fancy (and not Fancy) Headers

- By default, the only content in the header or footer of our document is the page number.
- We can change this by using a package called `fancyhdr`.
- Tell LaTeX to use this package? What is the command? Where does it go?
- In the header, also include `\pagestyle{plain}`. Compile the document. What changes?
- Now use `\pagestyle{fancy}`, instead. Compile the document. What changes?
- You can change the contents of the headers and footers by using commands of the form `\rfoot{}` for `r`, `l`, and `c`; and for `foot` and `head`.

- Place the state or non-US country in which you were born somewhere in the footer. Compile the document.
- Notice that the first page, the title page, has no header. Even when we use fancy headers and get the pretty content on that first inch of the page, any page with `\maketitle` is unchanged, by default. We can change this by including `\thispagestyle{fancy}` immediately after `\maketitle`.

Part IV
Including Mathematics in LaTeX Documents

16 Additional Packages

- In order to avoid a lot of redundant and complicated code, it is helpful to load two particular packages each time you create a document with mathematics in it.
- A bare-bones document requires only a `\documentclass{}` line, followed by the `\begin{document}` and `\end{document}` lines.
- We will now add `\usepackage{amsmath, amsthm, amssym, amsfonts}` to the *preamble* of the document. This tells LaTeX to recognize certain commands defined in these packages.
- We'll never use most of the functionality they add, but we'll often use some of it!
- Create the following empty LaTeX document in WinEdt in which we'll practice mathematical commands.

```
\documentclass{article}

\usepackages{amsmath, amsthm, amssymb, amsfonts}

% This is just a comment.

\begin{document}

\end{document}
```

17 Math Modes

- In order to display mathematical content in LaTeX, we will use a new set of commands that we wouldn't need for, say, writing a letter. However, LaTeX doesn't allow you to issue these commands just anywhere.
- Rather, you need to be either in math mode or in a mathematical *environment* where the math mode is implied.
- There are two math modes which don't utilize special *environments*: *inline* math and *display math*.
- These two are analogous to the way we include quotes in our own written work.

In-line Sometimes the quotes are treated like any other text and printed in the same sized font without any special indentation.

Display Sometimes the quotes begin on new lines, are indented, and quite distinct from the other text.

- As a note, some commands and characters can only be entered in math mode. However, many, like the word "computer" can be entered in normal mode and math mode. Yet, if you think "computer" is different from "*computer*", then you must beware which mode you use.
- After this section, it will be assumed that all math command sequence are typed in math mode. Otherwise, they will almost always fail. The math mode openings and closings are omitted.

In-line Math

- The *in-line* math mode is opened by typing a single dollar sign (`$`) and then closed by typing a single dollar sign.
- The content is placed in between the two `$` symbols.
- When we type `$55/i = \pi^{-0.3}$`, we get $55/i = \pi^{-0.3}$. Any lines immediately following the in-line text are entirely unaffected, including this one. And this one, too.

Display Math

- The *display* math mode is opened by typing `\[` and closed by typing `\]`. Another approach is to use double dollar signs as in the in-line mode (`$$ ...$$`). However, the square bracket approach is considered "better".
- Again, the content to be displayed in the math mode is placed between the opening and closing key sequences.
- Now, when we type `\[55/i = \pi^{-0.3}\]`, we get

$$55/i = \pi^{-0.3},$$

  which is the same as when we type `$$55/i = \pi^{-0.3}$$` and we get

$$55/i = \pi^{-0.3}.$$

- Clearly the appearance of the *display* math is different from the *in-line* math. However, sometimes the differences go beyond indentation and size. Both $\int_0^\infty \sum_{n=1}^\infty \frac{1}{n}\, dx$ and

$$\int_0^\infty \sum_{n=1}^\infty \frac{1}{n}\, dx$$

have the same math mode code, but they are placed in the two different modes. In this examples, the limits of integration are the most obvious difference in how these expressions are rendered.

18 Superscripts and Subscripts

- To add a superscript to an expression use a hat, "`^`", after the term with the exponent. So, `X^3` gives X^3.
- If your exponent has more than one character to be rendered in the case of X^{3+y}, then we must surround the entire exponent in *curly braces* (`{ }`). For this previous expression we'd type `X^{3+y}`. In this sense, the *curly braces* group terms. When in doubt, use curly braces to set the scope of your exponent. Had we omitted the curly braces, we'd see $X^3 + y$ which is quite different.
- The principle for subscripting is identical, but we use the underscore, "`_`", instead. Typing `X_{3+y}` gives X_{3+y}.
- Super– and subscripting can be recursive. However, one must be very careful with braces in this case. We can typeset e^{x_i} with `e^{x_{i}}`. Now, the braces around the `i` term are optional. Remove them, what happens to the rendering? Remove the braces around the x_i term. Compile the document. What happens to the rendering?
- Both superscripts and subscripts can be used simultaneously. In Social Choice, we often care about the median voter's preference, denoted x_i^m. This can be achieved with either `x_i^m` or `x_{i}^{m}`. The latter is more typing, but forces you to be explicit about the scope which tends to reduce the number of errors.

19 Operators

$+$ `+`	$\vee$ `\vee`	$\forall$ `\forall`
$=$ `=`	$\wedge$ `\wedge`	$\exists$ `\exists`
$-$ `-`	$\bigvee$ `\bigvee`	$\subset$ `\subset`
$/$ `/`	$\bigwedge$ `\bigwedge`	$\subseteq$ `\subseteq`
$<$ `<`	$\cup$ `\cup`	$\subsetneq$ `\subsetneq`
$>$ `>`	$\cap$ `\cap`	$\supset$ `\supset`
$\leq$ `\leq`	$\bigcup$ `\bigcup`	$\supseteq$ `\supseteq`
$\geq$ `\geq`	$\bigcap$ `\bigcap`	$\supsetneq$ `\supsetneq`

$\int$ `\int`	$\Leftrightarrow$ `\Leftrightarrow`	$/$ `\not`
$\sum$ `\sum`	$\leftrightarrow$ `\leftrightarrow`	$\neg$ `\neg`
$\prod$ `\prod`	$\Leftarrow$ `\Leftarrow`	$\div$ `\div`
∂ `\partial`	$\leftarrow$ `\leftrightarrow`	∞ `\infty`
$\sim$ `\sim`	$\Rightarrow$ `\Rightarrow`	$\emptyset$ `\emptyset`
$\approx$ `\approx`	$\rightarrow$ `\rightarrow`	$\in$ `\in`

20 Greek Letters

- Greek letters are uncomfortably common in the work we do, so you will get used to including them in your documents.
- Although there are neither ζ nor χ keys on your keyboard, we can include these symbols in the math environment in a very intuitive way. Type `\name-of-letter` or `\Name-of-letter` for the lower-case and upper-case version of the Greek letter.
- So, `\delta` and `\Delta` give δ and Δ, respectively.
- Not every letter has a special upper-case version provided. Although `\beta` gives β `\Beta` is undefined.
- Not every letter has a special character, regardless of case. For example, the Greek omicron is just like the Roman 'o', so `o` gives o which will be sufficient.
- With Greek letters a particular issue arises often, although it isn't specific to Greek letters. LaTeX commands do not need a leading space before the `\`. Yet, they do need a trailing space so that LaTeX knows the name of the command is over. So, `X\beta` ($X\beta$) is rendered identically to `X \beta` ($X\beta$) because LaTeX has its own way of interpreting whitespace. However, `X\beta + \epsilon` ($X\beta+\epsilon$) works whereas `X \beta+ \epsilon` will not because `\beta+` is an undefined control sequence (i.e. we just made up that command).

21 Other Letter-y Things

Sometimes we need to present change the presentation of standard letters or symbols in math mode. Here are several common examples. The way these work is that whatever symbols are to have their face changed are passed to these commands as arguments. Notice how the "math" faces remove white space

Modification	Command	Example	Common Uses
Normal Math Face	—	$ABCXY$	most maths
Roman Face	`\textrm{}`	ABC XY	text within equation
Bold Math Face	`\mathbf{}`	$\mathbf{ABCXY}$	vectors and matrices
Blackboard Math	`\mathbb{}`	$\mathbb{ABCXY}$	special sets of numbers
Calligraphic Math	`\mathcal{}`	$\mathcal{ABCXY}$	arbitrary sets

22 Special Text-like Mathematical Expressions

Certain functions, operators, and constructs pop up in math frequently which are basically abbreviates for words. The cosine function—as in $\cos(\pi) = 1$—and the limit of an expression—as in $\lim_{x\to\infty} \frac{1}{x}$—are two such examples. If we just typed `$cos$` (cos) or `$lim$` (lim) we get results quite different. An incomplete list of the expressions for which the control sequences are defined is below.

det `\det`	min `\min`	cos `\cos`	exp `\exp`
lim `\lim`	inf `\inf`	sin `\sin`	Pr `\Pr`
max `\max`	sup `\sup`	tan `\tan`	arg `\arg`

23 Delimiters

Many times, we need to express groupings through the use of delimiters. For example, $(100 - 100)^{100}$ is quite different than $100 - 100^{100}$. For parentheses we can just enter `(x)`. For curly braces we enter `{x}`. And, lastly, for square brackets we use `[x]`. These produce (x), $\{x\}$, and $[x]$, respectively. In longer expressions, though, the results can look funny. In

$$[\sum_{x=1}^{3}\{(\int_0^x (e^y\ dy))\}],$$

the size of the delimiters is off. However, in

$$\left[\sum_{x=1}^{3}\left\{\left(\int_0^x (e^y\ dy)\right)\right\}\right],$$

the problems seem to be gone. If we type `\left(`, `\left\{`, or `\left[` and close the expression off with the approprate `\right)`, `\right\}`, or `\right]`, the sizes are determined by the size of the internal expression as opposed to being constant.

24 Accents

Often times we need to modify slightly a character it denote that the construct has slightly changed. For example, we want

- not the true parameter β but the estimate $\hat{\beta}$,
- not the random variable y, but the sample mean $\bar{y}$, or
- not the singleton x, but a vector $\vec{x}$.

A partial list of these accents is below.

$\acute{a}$ `\acute{a}`	$\dot{e}$ `\dot{e}`	$\tilde{\imath}$ `\tilde{\imath}`
$\bar{b}$ `\bar{b}`	$\ddot{f}$ `\ddot{f}`	$\vec{\jmath}$ `\vec{\jmath}`
$\check{c}$ `\breve{c}`	$\grave{g}$ `\grave{g}`	$\widehat{xyz}$ `\widehat{\xyz}`
$\check{d}$ `\check{d}`	$\hat{h}$ `\hat{h}`	$\widetilde{xyz}$ `\widetilde{\xyz}`

25 Assignment!

Questions

Write LaTeX code for the following expressions. Confirm that your compiled document matches these.

1. $$\neg\left((T \vee F) \wedge F\right) \vee (T)$$
2. $$e^{ix} = \cos x + i \sin x$$
3. $$y_i = \beta_0 + \beta_1 x_i + \beta_2 x_i^2 + \epsilon_i$$
4. $$\left(\sum_{x=1}^{\infty}(1/x)\right) - \left(\sum_{x=1}^{\infty}(1/x)^2\right)$$

Answers

1.
```
\[
\neg\left(\left(T \vee F\right) \wedge F\right) \vee \left( T \right)
\]
```
2.
```
\[
e^{ix} = \cos x + i \sin x
\]
```

3.

```
\[
y_i = \beta_0 + \beta_1 x_i +
\beta_2 x_i^2 + \epsilon_i
\]
```

4.

```
\[
\left(\sum_{x=1}^{\infty} (1/x) \right) -
\left(\sum_{x=1}^{\infty}
  (1/x)^2 \right)
\]
```

Part V
LaTeX Miscellany

26 Tables

- We can create tabular content in a LaTeX document either within a float environment or not. We will describe what a float is in a moment.
- Either way, tabular content like the following is created with the `tabular` environment.

Student	Alma Mater	Favorite Color
Tyson Chatagnier	Texas A&M	Denim
Jonathan Klingler	Notre Dame	Green
Gary Hollibaugh	UCSD	Orange and Aquamarine
Jonathan Olmsted	UDel	French Blue
Peter Haschke	UNC-Asheville	Titian Red
Lukas Pfaff	Iowa State University	Cornflower blue[3]

- The `tabular` environment takes a mandatory argument that is enclosed in curly braces. In the case of the above table where each column is center justified, we'd use `\begin{tabular}{c c c} \end{tabular}`. If we wanted the columns to be right-justified or left-justified we could use `r` or `l`.
- If we wanted to add a vertical line on either the left or right or in-between two columns we'd use the argument `{| c | c | c |}`.
- Horizontal lines are given by `\hline`.
- Tab breaks are caused by `&` and line breaks by `\\`.
- The full code for the above table is:

```
\begin{center}
  \begin{tabular}{c c c}
    \hline
    \hline
    Student & Alma Mater & Favorite Color \\
    \hline
    Tyson Chatagnier & Texas A\&M & Denim \\
    Jonathan Klingler & Notre Dame & Green \\
    Gary Hollibaugh & UCSD & Orange and Aquamarine \\
    Jonathan Olmsted & UDel & French Blue \\
    Peter Haschke & UNC-Asheville & Titian Red \\
    Lukas Pfaff & Iowa State University & Cornflower blue
    \footnote{This is not necessarily true, it is,
      however, just a corn joke!'}\\
    \hline
    \hline
  \end{tabular}
\end{center}
```

- The line breaks and whitespace in the above code are poorly selected, but this results from the need to fit the code in such a small block. Usually, the code for LaTeX tables can be several times wider than anything else in your LaTeX document.
- There are much more complex environments than `tabular` that you will have to use once your tabular content grows. They are similar, but have their own documentation to help you through.

27 Pictures

- LaTeX has the ability to draw arbitrary types of objects and schematics within a LaTeX document in a native language. However, this is seldom used and probably less advantageous than including external files.
- Download `Image1.pdf` and `Image2.png` from `www.rochester.edu/college/gradstudents/jolmsted/`
- In your preamble, add the `graphicx` package (i.e. `\usepackage{graphicx}`)
- Where you'd like the picture to show up, type `\includegraphics{./Image1}` without a file extension and be sure to use the appropriate path to the image file, relative to the LaTeX document. The file extension is not necessary.
- The `scale` optional argument is useful to change dimensions on the fly without distorting the aspect ratio.

- Try the following code in a LaTeX document environment:

```
\begin{center}
  \fbox{
    \fbox{
      \includegraphics[scale=.25]{./Image1}
    }
    \fbox{
      \includegraphics[width=3in,
      height=1in,
      angle=90]{./Image2}
    }
  }
\end{center}
```

- The result is something like

- Notice that we didn't have to specify the file extension. Notice the order in which the `angle` and then `height`/`width` arguments are applied. What does `\fbox{}` do?

28 Floats

- Although we can create tables with the `tabular` environment and we use `\includegraphics{}` to insert external image files, these commands

are seldom placed inside a document without entering them in a special environment.

- Typically, these kinds of content are placed in *floats* which act like containers for tables and figures. LaTeX, according to a set of rules, figures where these containers should be placed which depends on the content before the float, the content after the float, and the LaTeX code within the float.
- If nothing else, the big adjustment required by users placing content in floats is realizing that the placement of a table or figure is up to LaTeX and "jury-rigging"/"jimmy-rigging"/"jerry-rigging" the placement is ill-advised.
- The float for tables is `\begin{table}[htpb] \end{table}`. The optional `[thpb]` argument provides LaTeXwith some instructions on where to place the table.
- The float environment for figures is `\begin{figure}[htpb] \end{figure}`. Notice, again, the optional argument.
- In actuality, there need not be anything inside these float environments, or it could easily be regular LaTeX markup.
- `\caption{}`, placed somewhere in the float, allows a title of the content to be placed and automatically numbered.
- `\label{}`, placed immediately after the `\caption{}` command gives an identified to the object by which it can be referred for directing readers to Figure 1 or Table 4 without hard-coding the float order.
- Try:

```
\begin{figure}
  \begin{center}
    \fbox{
      \includegraphics[width=3in,
      height=1in,
      angle=120]{./Image2}
    }
    \caption{A Hero of a Man} \label{f:Riker}
  \end{center}
\end{figure}
```

- Now, see Figure 1 on page 27 to view the output. We were able to reference the figure number automatically using `\ref{f:Riker}` which matches our `\label`. We reference the page number automatically with `\pageref{}`.

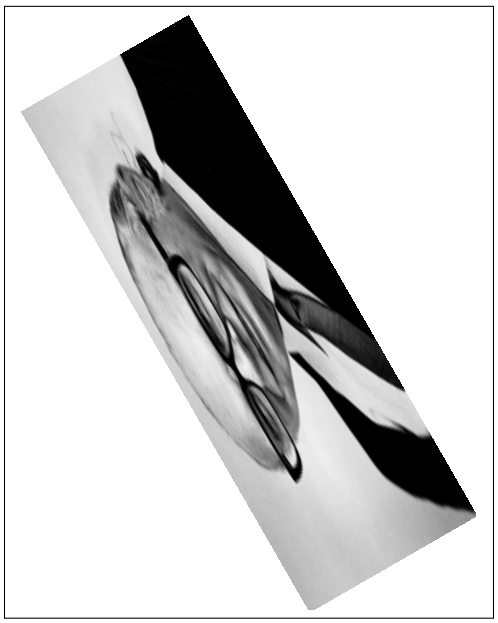

Figure 1: A Hero of a Man

More Math Environments

- There are a number of math environments that become useful when one is typesetting mathematical notation beyond very basic experessions.
- As in a previous tutorial, add `\usepackage{amsmath, amsthm, amssymb, amsfonts}` to the preamble if not already there.

Fraction

There are instances where 3/4 would look better as $\frac{3}{4}$ and this works equally well for longer expressions

$$\frac{\tan\left(\cos\left(\sin(X)\right)\right)}{\int_{\mathbb{R}} f(x)\ dx}.$$

It is the command `\frac{}{}` which provides this. The numerator is the first argument and the denominator is the second.

Equation

The equation environment is a very common way to typeset equations when reference numbers are being used. So, for example, `\begin{equation} 4=x^2 \end{equation}`

gives

$$4 = x^2. \tag{1}$$

Now, it is not necessary to place the equation environment in a math mode. In this sense, the math mode is implied. There is also a variant such that `\begin{equation*} 4=x^2 \end{equation*}` suppresses the equation numbering,

$$4 = x^2.$$

Equation numbers can be labeled and referenced as was done in the figure environment. This is a single equation environment and if you try to enter line breaks such that you could force another line, it will fail. In that case, I find the `align` approach being the easiest because it is flexible.

Align

The `align` environment allows you to enter multiple lines and include alignment stops. There is a un-numbered version, `align*`, too. The code

```
\begin{align}
   \sum_{x=1}^{4} \frac{1}{x} &
   &= \frac{1}{1} + \frac{1}{2}  + \frac{1}{3} + \frac{1}{4} & \\
   &= \frac{25}{12} &\\
   & &>2
\end{align}
```

produces a three line `align` environment.

$$\sum_{x=1}^{4} \frac{1}{x} = \frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} \tag{2}$$

$$= \frac{25}{12} \tag{3}$$

$$> 2 \tag{4}$$

Notice that the numbering is cumulative. The use of `\label{}` and `ref{}` works identically here.

Array

The `array` environment provides a unified way of representing vectors and matrices. The environment begins in the standard `\begin{array}{cc} \end{array}` way, but the mandatory `{cc}` argument specifies there are two center-justified columns. Changes to this argument proceed identically to the argument in the creation of tables. On important point is that the array environment does not create its own math mode environment, so we must put it inside one when we use it. We get

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix},$$

the two dimensional identity matrix from

```
\[
 \left[
\begin{array}{cc}
  1 & 0 \\
  0 & 1
\end{array}
\right],
\]
```

Notice the comma in the source code. It must be in the display math environment so that it is placed adjacent to the matrix and not in the text. By changing the number of rows, columns, and the delimiters, most matrix-like objects can be represented with this environment.

Cases

The `cases` environment is both extremely useful, but quite narrow in application. Although it is designed to be used only to represent piece-wise functions, it is a marked improvement over the alternative which would be to "hack" the `array` environment. Like the `array` environment, though, `cases` must be used within math mode. So,

$$\mathbf{1}_{\mathcal{X}}(x) = \begin{cases} 1, & x \in \mathcal{X} \\ 0, & \text{otherwise} \end{cases}$$

is the result of the code

```
\[
\mathbf{1}_{\mathcal{X}}(x) =
\begin{cases}
  1, & x \in \mathcal{X} \\
  0, & \textrm{otherwise}
\end{cases}.
\]
```

Notice how I use whitespace and line breaks to organize the code although LaTeX won't interpret it.

29 Words of Wisdom

- It is considered good practice to keep text file contents within the first 80 characters. This may seem weird or hard, but this, along with use of the `%` and comments will make the input file more human-readable.
- As you learn LaTeX don't worry about trying to make LaTeX look a certain way. Tell LaTeX about your content and its structure. Let LaTeX worry about the details of appearance.

- Google is your friend.
- Comment out error-laden parts of code. Add things back in one at a time until you've identified the source of your syntactical mistakes.
- Let WinEdt help you. It highlights the source file according to rules. If the rules are broken, the highlighting will appear other it should and this is a visual cue that something is wrong.
- Because LaTeX seldom interprets whitespace in too generous a way, use it as an organizational tool.