A TEXnical Skills Workshop: Tips and Tricks

How to Make LATEX Your Friend

Geoff Vooys

2021 May 11

Structure of the Talk

The Outline

Over the course of the next hour we'll learn how to do LaTeX stuff together! Here's the rough order we'll be doing stuff:

- Pre-requisites and setup stuff.
- Learn how to type math in nice ways (this includes the underrated `\operatorname{}` command) and hints for typesetting matrices.
- Learn how to make your own macros and commands.
- Learn how to set up bibliographies using BibTeX, as well as how to change stuff around as necessary. I will not be covering BibLaTeX(so if you want to talk about Biber just wait unitl the Q&A).

Structure of the Talk

The Outline

- Learn how to set up figures in a document (and all the chaos around this — by the way, it kinda sucks).
- Learn how to reference various lemmas, theorems, propositions, figures, and equations you've set up in ways that make sense. This is to save you work later and do things like hyperlink to page and proposition numbers!
- Depending on time: Learn about how to do stuf in Beamer and make slide shows like this with LaTeX.

Setting Up

Hi There!

This talk is going to be interactive and I'd like you to practice stuff as we go along! For this we'll need to be able to compile some source TEX as we go, so please make sure you have a way to do this. There will be some small take home exercises and a text file of interest to you at my git repository that I've hyperlinked to this sentence. Here are your options for TeX:

Setting Up

Hi There!

This talk is going to be interactive and I'd like you to practice stuff as we go along! For this we'll need to be able to compile some source TEX as we go, so please make sure you have a way to do this. Here are your options for TeX:

- If you want to do LATEX on your own computer I recommend using MikTEX (`https://miktex.org/`). For typesetting IDEs I recommend the Windows option TEXStudio (`https://www.texstudio.org/`).

Setting Up

Hi There!

This talk is going to be interactive and I'd like you to practice stuff as we go along! For this we'll need to be able to compile some source TEX as we go, so please make sure you have a way to do this. Here are your options for TeX:

- If you want to do LATEX on your own computer I recommend using MikTEX (`https://miktex.org/`). For typesetting IDEs I recommend the Windows option TEXStudio (`https://www.texstudio.org/`).
- You can use `https://www.overleaf.com/` if you want to use a browser based LATEX interpreter.

Setting Up

Code Snippits

In this talk I'll occassionally be writing LaTeX code on slides for stuff to do and tricks and the like. This is done in blocks like this

```
\usepackage{amsmath}
\usepackage{stmarysrd}% For power series brackets
```

so as to better mimic raw TeX code.

Why We Learn LaTeX

The Reasons

While LaTeX has **a lot** going for it as a superior document creation/preparation tool and as a markup language, it has a severe drawback: It is complicated to learn. For most of us there is a very good reason to fight through the learning curve:

- Precise, clean, and good-looking typesetting of figures, graphics, mathematics, and mathematics-adjacent material.

Why We Learn LaTeX

What a .tex File Looks Like

Before we dive right in, let's set up some language to use to discuss a .tex file. I've prepared a dummy file for you all hereTODOTODO MAKE A LINK but here's a quick run-down in case you don't want to download it:

```
\documentclass{foo}% Declares that you're writing a
document of type foo.

% This is your preamble. Packages, settings, and commands
you use/need go here.

\begin{document}% This is how you tell TeX to begin the
document you're typesetting.
% The writing and work you do goes here.
\end{document}% This is how you tell TeX where the document
to be typeset ends.
```

Why We Learn LaTeX

Where We're Going and a First Step

We'll begin our journey by learning the least insane of these tasks first: Typesetting mathematical formulae and expressions. For this make sure that you have a .tex document open with the code

```
\documentclass{article}


\begin{document}
% your article/code/math goes here
\end{document}
```

typed out and ready to go.

Math Typesetting in LaTeX

How to Begin Typesetting Math

We all want to typeset mathematical expressions and formulae (I can say with probability 1 that we've all needed to type a summation

$$\sum_{n=0}^{\infty} a_n x^n$$

at some point), but how does LaTeX know to read this as math? Because of the math packages we'll load in our preamble!

Math Typesetting in LaTeX

How to Begin Typesetting Math

Make sure to load the following packages in your preamble whenever you want to type math by doing the following:

```
\documentclass{foo}

\usepackage{amsmath}
\usepackage{amsfonts}
\usepackage{amssymb}% These three packages tell us math is happening!

\begin{document}
\end{document}
```

Typesetting Math: The Basics

When typesetting math, there are two different modes of making LaTeX output math: Inline (Math) Mode and Display Mode. Here is a quick description of the two:

- Inline Mode: This is math that is done within a line and internal to a sentence or line of text. Something like "for all $x \leq 5$" or "whenever $a < 0$ the ring $\mathbb{R}[x]/(x^2 + a)$ does a thing." This math should usually be short and not very tall.
- Display Mode: This is math that is done one a line all by itself, away from the text. This is usually long, complicated, and tall math like:

$$\sum_{n=0}^{\infty} \left(\frac{x^n}{n!}\right)^2, \qquad \prod_{\substack{p \geq 2 \\ p \text{ prime}}} \frac{1}{1 - p^{-s}}, \qquad \int_X \int_Y f(x, y)\, \mathrm{d}\nu(y) \mathrm{d}\mu(x).$$

Typesetting Math: The Basics

Some handy commands we'll also want to know for typing math are given below:

- The command

  `\mathbb{foo}`

  gives you a letter in blackboard bold font (so long as foo is a capital letter). For example: $\mathbb{A}, \mathbb{C}, \mathbb{R}, \mathbb{E}, \mathbb{X}, \mathbb{Z}$.
- To give terms superscripts and subscripts, when in math mode type either

  `foo_{boo}, foo^{boo}`

  to get either foo_{boo} or foo^{boo}.

Typesetting Math: The Basics

Some handy commands we'll also want to know for typing math are given below:

- The command

  `\sum`

  gives a large capital sigma for use in a summation. Giving it a subscript gives a lower limit to the sum and giving it a super script gives an upper limit to the sum. Explicitly, the code

  `\sum_{foo}^{boo}`

  gives

$$\sum_{foo}^{boo}$$

Typesetting Math: The Basics

Some handy commands we'll also want to know for typing math are given below:

- To draw an arrow between objects (like when defining function as in $f : X \to Y$) use the command:

  `\to`

- Fractions are given by the code

  `\frac{foo}{boo}`

  and give you the fraction

$$\frac{foo}{boo}.$$

Math Typesetting in LaTeX

Inline Math

Let's begin by getting to know inline math in LaTeX. Here are some facts:

- Enter inline math mode by typing \(and close math mode by typing \). For example:

  ```
  We solve the equation \( x^2 + 2\) over the complex
  numbers \(\mathbb{C}\)
  ```

- Ideally this is for shorter math that fits in a sentence of words.
- This is NOT for tall math! If you have a fraction either write a/b if it needs to be inline or put the fraction in display mode if it needs to be tall.

Math Typesetting in LaTeX

Example

The code that gives you the line that I've typed here with the math $asd^{-1}(1/dsa)$ is:

```
The code that gives you the line that I've typed here with
the math \(asd^{-1}(1/dsa)\) is:
```

Similarly, the code that gives the line "I really like fractions a/b inline but not $\frac{a}{b}$" is:

```
I really like fractions \(a/b\) inline but not
\(\frac{a}{b}\)
```

Display Mode

Let's get to know display mode math in LaTeX.

- Enter display mode by typing \[and close display mode by typing \]. For example, the code

```
My favorite function is not
\[
\zeta(s) = \sum_{n = 1}^{\infty}n^{-s}.
\]
```

gives: My favorite function is not

$$\zeta(s) = \sum_{n=1}^{\infty} n^{-s}.$$

Display Mode

- Alternative ways to enter display mode: By using the environments

```
\begin{equation*}
foo
\end{equation*}
```

or, if you want your equation to have a label/number to your displayed equation,

```
\begin{equation}
foo
\end{equation}
```

Math Typesetting in LaTeX

Display Mode

- Display mode is best used for tall equations (things with a summation or product symbol, fractions, powers of powers of numbers, etc.) and for equations to which you explicitly need to label.

Example

The code that gives the expression

$$\sum_{n=0}^{\infty} \frac{(-1)^n}{n!} = \frac{1}{e}$$

is

```
\[
\sum_{n=0}^{\infty}\frac{(-1)^n}{n!}= \frac{1}{e}
\]
```

Practice Typesetting Math

Example

Find code to write the following:
My favorite scheme is $\mathbb{A}^{\infty}_{\mathbb{C}}$ although $\mathbb{A}^{1}_{\mathbb{C}}$ is nice too. You can define something like

$$f(z) = \sum_{n=0}^{\infty} \frac{z^n}{n!}$$

affine-locally on them!

Practice Typesetting Math

Example

Here is some sample code:

```
My favorite scheme is \(\mathbb{A}_{\mathbb{C}}^{\infty}\)
although \(\mathbb{A}_{\mathbb{C}}^{1}\) is nice too. You
can define something like
\[
f(z) = \sum_{n=0}^{\infty}\frac{z^n}{n!}
\]
affine-locally on them!
```

The Last Math Typesetting Topic: left/right Delimiters

Example

Can you figure out how to make the output

$$(\sum_{n\geq 1} \frac{1}{n^2}) = \frac{\pi^2}{6}$$

look nice? Note those lame parentheses on the sides of the sum: can we fix them? The answer won't surprise you!

The Last Math Typesetting Topic: left/right Delimiters

Left/Right Delimiters

Sometimes when doing math (in display mode) we have really tall operators and stuff that go inside parentheses, brackets, braces, and other such things. Without doing some ninjitsu, the outer wrappers will not be of an apporpariate size to contain the tall stuff within them. Here are examples:

$$\left(\int_X f\,\mathrm{d}\mu\right), \quad \left[\prod_{n\in\mathbb{N}}\left(1+\frac{1}{n+1}\right)^n\right], \left\{x\in\mathbb{C} \;:\; \left|\frac{z^2+2}{4}\right| < 2\right\}$$

The Last Math Typesetting Topic: left/right Delimiters

Left/Right Delimiters

To fix these issues we use the commands

```
\left* foo \right*
```

where in this case the $*$ symbol is used to denote whatever wrapper you're using.

- Left/Right commands MUST be used in pairs. If you don't your code will crap out.
- Left/Right can be used on pretty much any wrapper and even with exotic commands like:

  ```
  \lvert\rvert, \lVert\rVert, \lbrace\rbrace
  ```

The Last Math Typesetting Topic: left/right Delimiters

Example

With proper left/right delimiter use the stupid examples earlier now look like this:

$$\left(\int_X f \, \mathrm{d}\mu \right)$$

$$\left[\prod_{n \in \mathbb{N}} \left(1 + \frac{1}{n+1} \right)^n \right]$$

$$\left\{ x \in \mathbb{C} \ : \ \left| \frac{z^2 + 2}{4} \right| < 2 \right\}$$

The Last Math Typesetting Topic: left/right Delimiters

Example

Here is the code for each example:

```
\left(\int_X f\, \mathrm{d}\mu\right)
```

```
\left[ \prod_{n \in \mathbb{N}}\left(1 +
\frac{1}{n+1}\right)^n\right]
```

```
\left\lbrace x \in \mathbb{C}\; : \; \left\lvert
\frac{z^2+2}{4}\right\rvert < 2 \right\rbrace
```

The commands \, and \; are simply spacing commands (they add a certain amount of whitespace between things) and are not strictly necessary.

The Last Math Typesetting Topic: left/right Delimiters

Exercise

Write your own code to make the following expression pretty:

$$\{z \in \mathbb{C} \ : \ (\sum_{n=0}^{\infty} \frac{z^{2n}}{n!})(\sum_{n=0}^{\infty} \frac{z^{n}}{n!}) = e^{z^2+z}\} = \mathbb{C}$$

Typesetting Matrices and Operator Names

How to Typeset Matrices

With probabliity near 1, as mathematicians and statisticians we will likely need to typeset a matrix at some point. The good news is that this is not hard: There are really clean environments for just this purpose!

Typesetting Matrices and Operator Names

How to Typeset Matrices

- If you want to typeset a matrix with parentheses like

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

use the command

`\begin{pmatrix}foo \end{pmatrix}`

- If you want to typeset a bracket matrix like

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

use the command

`\begin{bmatrix}foo \end{bmatrix}`

Typesetting Matrices and Operator Names

How to Typeset Matrices

- If you want to typeset a matrix with vertical lines like

$$\begin{vmatrix} a & b \\ c & d \end{vmatrix}$$

use the command

```
\begin{vmatrix}foo \end{vmatrix}
```

- In any case, when typesetting matrices use a symbol & to add new columns and use a double backslash \\ to add a new row.

Typesetting Matrices and Operator Names

Example

The matrix

$$\begin{pmatrix} 1 & 1 & a \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}$$

is typeset by the code

```
\begin{pmatrix}
1 & 1 & a \\
0 & 1 & 1 \\
0 & 0 & 1
\end{pmatrix}
```

Typesetting Matrices and Operator Names

The Operatorname Command

Sometimes we want to make words or letter combinations like "trace, var, op, id, true" and others look like text and behave as if they were functions in math mode. Using the text command doesn't do this (sadly), so we need another command: The operatorname command! The command

```
\operatorname{foo}(x)
```

will output foo(x).

Typesetting Matrices and Operator Names

Example

Each of the following operators were typeset with operatorname:

$$\operatorname{trace}(a_{ij}) = \sum_{i=1}^{n} a_{ii}, \quad \operatorname{id}_X(x) = x, \quad \operatorname{Ad}_h(g) = hgh^{-1}.$$

The code is

```
\operatorname{trace}(a_{ij}) = \sum_{i=1}^{n}a_{ii}

\operatorname{id}_{X}(x) = x

\operatornameAd_h(g) = hgh^{-1}
```

Typesetting Matrices and Operator Names

Exercise

Find code to typeset the following sentence:
When writing math fractions like a^x/b and terms like $\sum_{i=0}^{k} \operatorname{trace}(A_i)$ are best done in math mode like

$$\frac{a^x}{b}, \sum_{i=1}^{k} \operatorname{trace}(A_i),$$

especially so that we avoid nonsense like having a matrix

$$A_i = \begin{pmatrix} 1 & x^i \\ 0 & 1 \end{pmatrix}$$

inline.

A Take-Home Question for Testing Your Understanding

Take Home Exercise

In the file Math Exercises, I have produced a TeX document that has some ugly math and poor life choices in it. Try to fix it up using what is outlined in this section! I'll post my solutions (your milage may differ, of course) at the end of the week.

What are Macros and Commands?

Well What are They? Hurry up, Geoff!

Given how often we have to write certain things (like $\mathbb{R}, \mathbb{C}, \mathbb{N}, \mathbb{Z}, \mathbb{Q}$, to be cheeky about it, as well as in my case $\mathfrak{C}, \mathscr{C}, \mathscr{A}, \underline{\text{Set}}, \underline{\text{Cat}}, \mathfrak{Cat}, \mathbf{Sch}$, and other madness) it can be extremely tedious to type every instance of everything out compeltely all the time. This is where macros come in: They're shorthand terms we write for ourselves that output some command or operator for us!

What are Macros and Commands?

How Do I Macro?

The easiest way to learn how to macro is to play some good old Starcraft 2. Jokes aside, here's how to write a macro for yourself:

1. Go to your preamble in your code (before the begin{document} line) but after you've loaded your packages (this way you can use anything you want to load).
2. Type the code

   ```
   \DeclareMathOperator{\command}{definition}
   ```

   This creates a command that you call by typing

   ```
   \command
   ```

   that outputs the definition (I'm going to display it in display mode):

$$\text{definition}$$

What are Macros and Commands?

Example

The command

```
\DeclareMathOperatorname{\R}{\mathbb{R}}
```

creates a command that outputs an $\mathbb{R}$ character whenever we type the command

```
\R
```

What are Macros and Commands?

Example

Here's a more complicated example. The macro

```
\DeclareMathOperator{\SumZeroInf}{\sum_{\mathnormal{k}=
0}^{\infty}}
```

creates a summation sign like this whenever you call it:

$$\SumZeroInf$$

What are Macros and Commands?

Exercise

Create code to produce macros for the following objects:

$$\prod_{p \in \mathbb{P}} \frac{1}{1 - p^{-s}}$$

$$\left(\int_X |f|^p \, \mathrm{d}\mu \right)^{1/p}$$

Note that the product and integral can be typeset with code

```
\prod
\int
```

you may need a mathnormal command or two, the d in the dx term is in the mathrm font, and the absolute values were typeset with \lvert on the left and \rvert on the right.

What are Macros and Commands?

Exercise

Here is my code for these macros:

```
\DeclareMathOperator{\EulerProdZeta}{\prod_{\mathnormalp
\in
\mathbb{P}}\frac{1}{1 - \mathnormal{p}^{-\mathnormal{s}}}}

\DeclareMathOperator{pNormf}{\left(\int_{\mathnormal{X}}
\lvert\mathnormal{f}\rvert \mathrm{d}\mu
\right)^{1/\mathnormal{p}}}
```

What are Macros and Commands?

Take Home Exercise

Write macros to produce a limit as $x \to \infty$ and a limit as $t \to \infty$ as below:

$$\lim_{x \to \infty}$$

$$\lim_{t \to \infty}$$

Command Time!

Now What?

We've all seen macros now, and got some chance to get used to them. However, in the last example it can be the case that we may want a short way to write down a p-norm or put stuff between norm terms without having to type \int, \lvert, \lVert, etc., all the time. Equivalently, you may want to make a limit where you only type in the variable and where it's going instead of some \lim underscore garbage all the time. This is where commands come in!

Command Time!

What is a Command and How do I Make Commands?

Essentially commands are macros that have variables in them that you define! In some sense macros are nullary commands (for all one or two universal algebraists out there). You define them as follows:

1. Go to your preamble again.
2. In your preamble type the following:

```
\newcommand{\command}[n]{foo{#1}{#2}...{#n}}
```

In the example above the syntax is:

```
\newcommand{\Command Name}[Number Of Arguments]thing
{#1}{#2}...{#n}
```

Command Time!

Example

This stuff is hard, so let's see an example! I wrote a command to do limits with one variable (I assume all limits are limits at ∞). The code is:

```
\newcommand{\limAtInfty}[1]{\lim_{{# 1}\to \infty}}
```

By typing in

```
\limAtInfty{t}
```

I get:

$$\lim_{t \to \infty}$$

Command Time!

Example

I now have realized there are limits that aren't at infinity. I've written a command to do limits at endpoints that I can specify! The command is

```
\newcommand{\limAtWhere}[2]{\lim_{{#1}\to {#2}}}
```

and by typing

```
\limAtWhere{t^{x+1}}{5}
```

I get

$$\lim_{t^{x+1}\to 5}$$

Command Time!

Exercise

Write a command which takes one input and gives you the output

$$\|\#1\|.$$

After writing this test it on the input

```
\frac{x}{4}
```

to see if it formats things correctly.

Command Time!

Exercise

Write a command which takes one input and gives you the output

$$\|\#1\|.$$

After writing this test it on the input

```
\frac{x}{4}
```

to see if it formats things correctly. My code was:

```
\newcommand{Norm}[1]{\left\lVert{#1}\right\rVert}
```

Command Time!

A Harder Exercise

WRite a command that takes two inputs and outputs

$$\left(\int_X |\#1|^{\#2}\,\mathrm{d}\mu\right)^{1/\#2}$$

To see if it formats things correctly test it on inputs #1 = \frac{1}{x^2} and #2=p.

Command Time!

A Harder Exercise

WRite a command that takes two inputs and outputs

$$\left(\int_X |\#1|^{\#2}\,\mathrm{d}\mu\right)^{1/\#2}$$

To see if it formats things correctly test it on inputs #1 = \frac{1}{x^2} and #2=p.
Here is my code:

```
\newcommand{\pNorm}[2]{\left(\int_{\mathnormal{X}}
\left\lvert {#1}^{#2}\right\rvert
\, \mathrm{d}\mu\right)^{1/{#2}}}
```

Command Time!

Take Home Exercise

Write a command that takes three inputs and outputs:

$$\overset{\#1}{\underset{\#2}{\#3}}$$

You might want to use the overset and underset commands for this. There's also the file Macro Exercises for extra practice.

Our Friend the BibTeX

Why BibTeX?

As academics contributing to the academic conversation, it is absolutely crucial that we be able to cite and reference what we use and read from the works of others. Manually typing bibliographies is a nightmare (especially because of the various differences and abbrieviations in journal names). Luckily, there is a bibliography managment tool in LaTeX that can do a lot of this for us!

Our Friend the BibTeX

How to BibTeX

The best thing about BibTeX is that you don't have to load any packages to use it! Here's what you need to do instead:

1. Create an external bibliography file like

   `MyBibliography.bib`

   The beauty is that the file MyBibliography is where all the bibliography magic, minus a couple commands, lives.

2. At the end of your document (before the \end{document}) you declare your bibliography style and make sure that TeX knows where to find your bibliography (note below that I assume you loaded the package amsrefs):

```
\nocite{*}
\bibliographystyle{amsplain}
\bibliography{MyBibliography}
```

Our Friend the BibTeX

The Bibliography File

As I said, the magic of BibTeX is that the bibliography information is in a separate .bib file that controls the actual bibliography. The .bib file is essentially just a text file with some formatting restrictions that tells LaTeX what to look for and how to scrape the information. The good thing is this means all your information and bibliographic needs are compiled and written consistently!

Some amusing information: You can edit .bib files in nearly any program. Some things manage the information better than others (it would be helpful if people who use such programs could post their favorites in chat), but if you're really masochistic you can even edit these things in Notepad.

Our Friend the BibTEX

The Bibliography File

The .bib file is made up of as many (or as few) individual entries as you need. There is a large list of different formats an entry can take; while some are more esoteric, the most common are misc, book, and article entry types today. In any case, the entries will look like this (with or without some changes):

```
@ENTRY{RefTag,
title = {TITLE},
author = {AUTHORS},
year = {YEAR},
pages = {PAGE NUMBERS},
publisher = {PUBLISHER},
volume = {VOLUME},
}
```

Our Friend the BibTEX

The Bibliography File

Here are some universal things to keep in mind:

- If there is math mode in a title or accents/fixed capitalization on letters anywhere, you need to write braces {} around those terms. Otherwise BibTEX will throw out the formatting and just pretend everything is text.
- When listing multiple authors it is best to order them like so:

  `LAST NAME 1, FIRST NAME 1 and LAST NAME 2, FIRST NAME 2 and ...`
- Before trying to do all the entries manually, search for your reference here at mathsci net. The site I linked has complete BibTEX entries for a lot of things!
- There are no constraints on what is allowed for Reference Tags save that they have to exist and be unique to each entry.

Our Friend the BibTEX

Exercise

The nice thing about BibTEX is that it's pretty straightforward to set up. I've included two files (Bibliography.bib and BibExercise.tex) for you to fix and set up so that the bibliography will compile properly. Try playing around with it and getting it to compile! Also, you'll need to wait until we talk labels to fix this file.

Oh, because I forgot to say it elsewhere, to actually cite the things in your .bib file, just type \cite{BibKey} where BibKey is some key you've given an entry in your .bib file.

Inserting Figures: The Nightmare Begins

Why It Begins

In math and stats presenting data, pictures, and other visual information is extremely important. I may want to draw a bunch of circles, present some graphs, show regressions, or other things that are really important for people to see in order to understand your work. Unfortunately, adding figures is a nightmare, but I can help guide you so that the spoopiness is minimized.

Inserting Figures: The Nightmare Begins

It Begins by Setting Up

To create a figure you'll need to load some packages (especially if you're including images that are saved on your computer after doing experiments). For this you'll need to load the following package in your preamble:

```
\usepackage{graphicx}
```

If your images are in a different directory you also need the command below in your preamble:

```
\graphicspath{foo}
```

Inserting Figures: The Nightmare Begins

Including Images

To include pictures or other images it's best to do so wihtin a figure environment (for reasons). In order to do this we use the command

```
\begin{figure}
\begin{center}
foo
\end{center}
\end{figure}
```

Inserting Figures: The Nightmare Begins

Including Images

To include images in the figure, you use the command \includegraphics as in

```
\includegraphics{foo}
```

where foo is the file name of the image.

Inserting Figures: The Nightmare Begins

Example

In my working directory I have a file named DatBoi I want to make a figure. By including the code

```
\begin{figure}
\begin{center}
\includegraphics{DatBoi}
\end{center}
\end{figure}
```

Inserting Figures: The Nightmare Begins

Example

I get the figure

Inserting Figures: The Nightmare Begins

Wait What?

Note that in the example above our picture was way too big! Luckily we can control the height of pictures directy. When typing the \includegraphics command before we type the braces {} by including a height or width command in brackets [].

```
\includegraphics[width = WVal, height = HVal]{foo}
```

- The values we can give the width and height can take cm or pt (point) values. Experimentation is key to find what works best!
- You can also scale the image by a factor by using scale = VAL commands.

Inserting Figures: The Nightmare Begins

Example

In my working directory I have a file named DatBoi I want to make a figure. By including the code

```
\begin{figure}
\begin{center}
\includegraphics[height = 5cm]{DatBoi}
\end{center}
\end{figure}
```

Inserting Figures: The Nightmare Begins

Example

I get the figure

Inserting Figures: The Nightmare Begins

Other Things with Figures

When making figures it's also important to give captions so that we can explain what a figure does (especially since figures often don't go where we expect — more on this later). To make captions just insert the command \caption{} after you include the image in your figure as in the code below:

```
\begin{figure}
\includegraphics{foo}
\caption{A caption describing foo}
\end{figure}
```

Inserting Figures: The Nightmare Begins

Example

In my working directory I have a file named DatBoi I want to make a figure. By including the code

```
\begin{figure}
\begin{center}
\includegraphics[height = 5cm]{DatBoi}
\caption{Hey look! It's Dat Boi!}
\end{center}
\end{figure}
```

Inserting Figures: The Nightmare Begins

Example

I get the figure

Figure: Hey look! It's Dat Boi!

Inserting Figures: The Nightmare Begins

Last Topics for Inserting Figures

While we've seen how to include figures and caption them, we still need to see how to position figures. Often LaTeX will try its best to find a spot where the figure goes and fits on a page, but it will frequently become utter chaos. We won't do too much with this today save for your take home exercise, but here are the placement commands you can use in a figure as in the code below:

```
\begin[PLACEMENT OPTIONS]{figure}
foo
\end{figure}
```

Inserting Figures: The Nightmare Begins

Last Topics for Inserting Figures

The placement options are:

- h: This tells LaTeX to do its best to put the figure as close to the exact place relative to where you typeset the figure relative to the paragraphs around the figure as possible.
- !: This tells LaTeX to supress finding "good" floats for determining where figures should go.
- t: This puts the figure towards the top of the page.
- b: This puts the figure towards the bottom of the page.

Theorem and Theorem-Like Environments

We're Typesetting Books and Articles Now!

As we do things in math (and stats? I'm displaying my ignorance here) we'll need to typeset things within theorem or proposition or lemma or definition blocks. These blocks should have numbers and labels (and proofs) so that we can refer to them and their results with cute little lines like "by Theorem 4.20 we find the height function rises." Let me first show you how to make these.

Theorem and Theorem-Like Environments

Theorem Environments

To make a theorem environment you need to define it in your preamble. This can be done via the code:

```
\newtheorem{COMMAND}[NumberControl]{NameToAppear}
```

You then call this new theorem environment by typing

```
\begin{COMMAND}[Optional Stuff]
foo
\end{COMMAND}
```

Theorem and Theorem-Like Environments

Theorem Environments

In the theorem environment COMMAND I suggested above, the NameToAppear is what appears when you call the environment. So beginning command would give you an environment like what is below. Note that by default theorem style environments italicize text like the word "foo" below.

NameToAppear

foo

Theorem and Theorem-Like Environments

Numbering Theorems

When you create a theorem environment, by default it just counts how many theorems/propositions/COMMAND/etc. environments you've used in your document. It is good style to have many things number together (number theorems, propositions, lemmas, examples, and definitions together means you know roughly how far into a document a thing is; otherwise it's a crap shoot). Making this work properly just involves using that number control part when we define our environments!

Theorem and Theorem-Like Environments

Numbering Theorems

When defining multiple theorem environments, filling the number control with the name of an environment you've already defined tells LaTeX to count the new environment as if it was the thing you substituted. For example, in the code

```
\newtheorem{theorem}{Theorem}
\newtheorem{lemma}[Theorem]{Lemma}
\newtheorem{Conjecture}{Conjecture}
```

every instance of lemma is counted as if it were a theorem, while Conjecture is counted separately.

Theorem and Theorem-Like Environments

Numbering Theorems

Finally, we can tell LaTeX to count our theorems by order of section, subsection, or just in order of use. This is done by the command in your preamble:

```
\numberwithin{ENVIRONMENT}{Counter}
```

It tells LaTeX to count each instance of ENVIRONMENT so long as the value of Counter is fixed. Some values you can feed to counter are:

- section,
- subsection,
- chapter.

Note that you can also tell equations to be numberwithin'd sections as well!

Theorem and Theorem-Like Environments

Example

Here is a preamble that will define three theorem environments with two counted independently of one and numbered by sections and chapters, respectively.

```
\newtheorem{theorem}{Theorem}
\newtheorem{lemma}[theorem]{Lemma}
\newtheorem{FalseClaim}{False Claim}

\numberwithin{theorem}{section}
\numberwithin{FalseClaim}{chapter}
```

Theorem and Theorem-Like Environments

Definition Style Theorem Environments

Theorem environments are great, but there is a problem: If you define a definition environment naïvely you get an italicized definition, which is not ideal. However, we can fix this by declaring that our theorem style is "definition" before making more new theorem environments. This is perfect for making examples and defintions and exercises (oh my)!

Theorem and Theorem-Like Environments

Definition Style Theorem Environments

Here's how to declare definition style theorems. Type something of the following nature:

```
\theoremstyle{defintion}
\newtheorem{def}{Definition}
\newtheorem{example}[defn]Example
```

You can even have noremal theorems defined before you say that you have \theoremstyle{definition} and have both italicized and non-italicized environments!

Theorem and Theorem-Like Environments

Example

The following code gives italicized environments (lemma, theorem) and non-italicized environments (example, definition) which all have the same counter indexed by sections:

```
\newtheorem{thm}{Theorem}
\newtheorem{lemma}[thm]{Lemma}

\theoremstyle{definition}
\newtheorem{defn}[thm]{Definition}
\newtheorem{example}[thm]{Example}

\numberwithin{thm}{section}
```

Theorem and Theorem-Like Environments

Exercise

Write a preamble that introduces three theorem environments, two italicized and one non-italicized, that are all numbered together and indexed by sections and change the numbering so that equations are numbered within a section as well.

Theorem and Theorem-Like Environments

Exercise

Here is my sample code:

newtheorem{thm}{Theorem}

newtheorem{lemma}[thm]{Lemma}

theoremstyle{definition}

newtheorem{defn}[thm]{Definition}

numberwithin{thm}{section}

numberwithin{equation}{section}

Labels and References

Recall

Before we get started remember that if you want to make an equation with a number on it like

$$e^{2\pi\theta} = \cos(2\pi\theta) + i\sin(2\pi\theta) \tag{1}$$

is done by typing

```
\begin{equation}
foo
\end{equation}
```

Labels and References

Labels

We now get to see how to internally reference things so that we never have to manually type in numbers that change (and, if you're using the hyperref package, can be clicked on to send you to the correct place). These are done by setting up and using labels! Luckily, setting up labels is straightforward: Immediately after your command (like a theorem environment or equation) and any options you use in it, you simply add a label:

```
\begin{thm}[optional stuff]\label{DESCRIPTION}
foo
\end{thm}

\begin{equation}\label{DESCRIPTION}
foo
\end{equation}
```

Labels and References

Labels

The most important thing when writing labels is that you make your description something sensible (like naming variables in code, calling something "fart" is funny once but good luck searching through a zoo of "fart1" labels to find the correct thing you're referencing). Here are my suggestions that I use in my own code:

- Begin by adding **what** you're labelling to your label. For instance, if you're labelling a theorem or lemma or equation, tell **future** you that this is a theorem or lemma or equation.
- Next describe roughly where this thing is (so in Section DESCRIPTION or something like that) so that you know roughly where to start looking for the thing if you need to manually check the statement of the thing it's not insane to try and find it.
- Finally, give a short description of what the thing actually says so you and future you know **what** you're actually referencing.

Labels and References

Example

Here is a theorem environment and an equation I've labelled in an imaginary document.

```
\begin{thm}[optional]\label{Thm: Section Preliminaries:
Euler's Thm}
For real \theta the equation
\begin{equation}\label{Equation: Section Preliminaries:
Euler's Eqn}
e^{i\theta}= \cos(\theta) + i\sin(\theta).
\end{equation}
\end{thm}
```

Labels and References

References

We now get to do the most fun thing someone can do with their labels: Referencing them! This is very easy: In your code anywhere just write \ref{foo} where foo is a label you've already defined! For example:

```
\begin{thm}[The Fart Theorem]\label{Thm: fart}
foo
\end{thm}
Using Theorem \ref{Thm: fart} we get...
```

Labels and References

A Warning

I just want to point out the one main thing you actually can't do with labels. You only need to make sure that each label you write is unique.If your labels aren't unique LaTeX will get confused and crap out.

Labels and References

Example

Here's a theorem and equation environment with references:

```
\begin{thm}[optional]\label{Thm: Section Preliminaries:
Euler's Thm}
For real \theta the equation
\begin{equation}\label{Equation: Section Preliminaries:
Euler's Eqn}
e^{i\theta}= \cos(\theta) + i\sin(\theta).
\end{equation}
\end{thm}
Theorem \ref{Thm: Section Preliminaries: Euler's
Thm} is classical and Equation \ref{Equation: Section
Preliminaries: Euler's Eqn} can be proved with power series
tricks.
```

The Joke's On Me

I ran out of time. Beamer kinda sucks but the documentation on overleaf is pretty good. I've also included the source TeX for this talk in the git repo so you can dissect it and use it for your presentations.

Any Questions?

The Last Slide

The End

Thanks for coming and listening everybody!