Lecture 9: LaTeX

October 23rd, 2018

1 History

1.1 TeX

TeXwas primarily authored by Donald Knuth. Donald started writing The Art of Computer Programming back in 1962, which is a volume of programming algorithms and analyses which is still being worked on today. In the 70's, one of the volumes had to be typeset all over again because an industry standard was moving away from mechanical typesetting (pressing ink onto paper using metal molds of characters) and toward phototypesetting (projecting light through a film-negative of a character onto photographic paper in a light-proof canister).

Motivation for TeX Donald was frustrated by this, and separately but around this same time, he was impressed with having seen for the first time a digital typesetting system. In 1977, he outlined the basic features of TeXand planned to finish it over a sabattical in 1978. The name comes from combining a sequence of mathematical characters Tau, Epsilon, and Chi. The program has undergone several major changes, including rewriting it entirely in TeX'82 to make it a Turing Complete language, which means that the language can be used to simulate *any* computational aspects of any other computer (language).

Stability of TeX The language TeXis now quite stable. It uses an idiosyncratic version numbering system in which updates are signified by adding an additional digit to the end of the current version number, in a way such that the version numbers asymptotically approach π. It's currently in version 3.14159265, and this was updated in January 2014. The design was frozen after version 3 to reflect that no new features will be added, only bugs will be fixed. Even though he's proposed additional feature enhancements, Knuth believes the value in a stable typesetting system outweights further improvements. The final version of TeXwill be released after Knuth passes, at which point the version number will be changed to π exactly and all bugs will become features. See: "The Future of TeXand METAFONT" (Knuth '90).

Bugs and Monetary Awards Donald Knuth has maintained a detailed listing of all bugs ever found (and of course corrected) in TeXsince 1982. There are currently just over 400 listings. Knuth offers monetary awards to individuals who find bugs, starting at one hexadecimal dollar ($2.56) and doubling every year until frozen at its current value of $327.68. Anecdotally, Knuth hasn't lost much money in this endeavor since recipients of these checks enjoy framing them on their walls instead of cashing them.

1.2 LaTeX

Leslie Lamport authored an extension of TeXin the early 80's. It builds off of TeXin a way that is designed to be easier for writers. Essentially, LaTeX can be coarsely described as a collection of TeX macros (e.g. for making chapters, section headers, subsections, etc.). Its current version is LaTeX2$_\epsilon$. It is still under active development, and is less stable than TeX. There is a research version of LaTeX3 being developed. Here is a link to an unofficial reference.

Markup Languages A *markup language* lets us syntatically differentiate between formatting modifiers and content. The name comes from when skilled typographers would "mark up" manuscripts by hand, annotating in the margins what typeface style, font, and size should be applied to each part of the document; after the manuscript was marked up by hand, it would be passed off for actual typesetting. Markup languages are in contrast with What You See Is What You Get programs, e.g. Microsoft Word, which allows the writer to see the end product as they are prototyping.

Relating LaTeX and HTML LaTeX is a document markup language, and is targeted toward high quality print documents, appearing in scientific or academic journals. HTML is also a document markup language, which targets web browsers. The *elements* of HTML are *tags*. TeX commands are also now used to typeset equations on the web, and so even if you don't wish to live in LaTeX, they're still worth learning. E.g. markdown languages support some TeX commands.

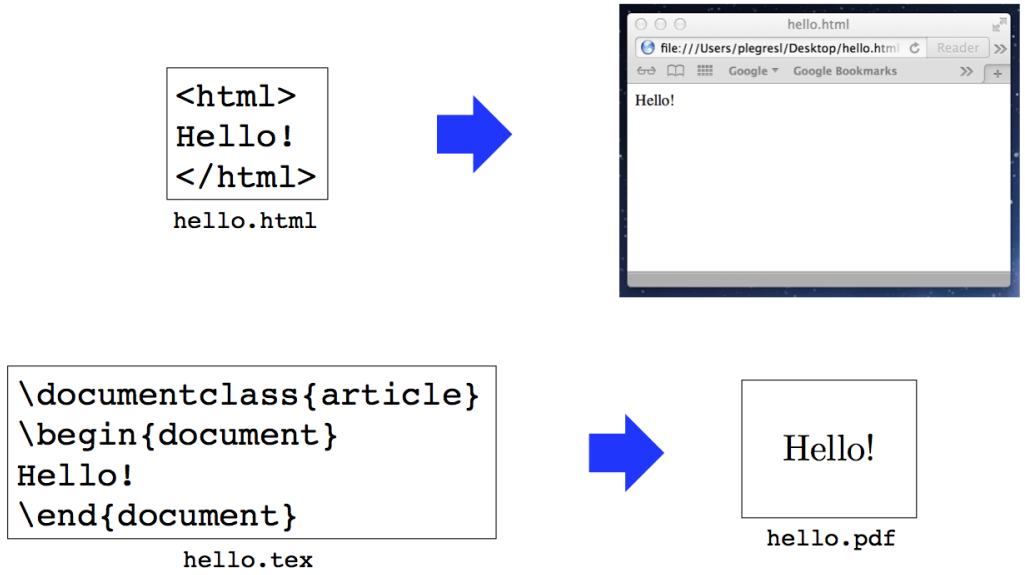


Figure 1: We show raw source code used to generate a simple HTML document and correspondingly for a LaTeXdocument. In either case, we must delimit our body of text in some way, either by `<html>` and `</html>` or `\begin{document}` and `\end{document}`.

2 Setup and Usage

2.1 Installation

We provide different recommendations based on the operating system.

- Windows: `http://miktex.org/about`
- Mac OSX: `https://tug.org/mactex/`
- Ubuntu: `$ sudo apt-get install texlive`
- Fedora: `$ sudo yum install texlive-scheme-medium`

Note: on Linux distributions, you may have to install other `texlive` packages to get the full TeXLive distribution. Ubuntu has the package `texlive-full`. Fedora has the package collection `texlive-scheme-full`. These are large downloads. Be prepared to wait. Don't install TeX at the last moment!

2.2 Compiling a `.tex` file into a PDF

We write our LaTeX markup in `.tex` files, similar to how we write Python code in `.py` files. When we want to compile the source code into a PDF, we need to use a separate program, e.g. `pdflatex`.[1] In general, we compile a `.tex` document into a `.pdf` file via the following.

```
$ pdflatex myfile.tex    # myfile.pdf is produced.
```

We can even use a GNU makefile to drive TeX.

Emacs and Vim Recall our extensible editors? Emacs supports AUC-TeX, which easily allows us to compile LaTeX into a pdf without ever having to leave our editor. We just use the key-strokes `C-c C-c`. Here's an installation of Emacs (complete with AUCTeX, and other goodies like ESS). Note that installing emacs is totally separate from installing TeX. Similarly for Vim, there is `Vim-LaTeX`.

GUIs Of course, some may enjoy a graphical interface. Options include TeXShop, TeXWorks, and TeXMaker.[2]

[1] TeXLive is already installed on `rice.stanford.edu`, and this includes `pdflatex`. The primary access point for CME 211 will be the command line program `pdflatex`.

[2] Note that there are also programs like Overleaf, an online tool for writing TeXdocuments, compiling, and sharing with others; you can get a temporary educational account with free access, but again they're happy to charge you $15/month thereafter. They advertise features like being able to Sync with Dropbox and Github, and provide full tracking history. Do these features sound familiar to other free tools that we've learned how to use, like Git?

2.3 Hello world

See: `tex/hello.tex`:

```
\documentclass{article}
\begin{document}
Hello
\end{document}
```

Typesetting instructions (from the same directory as the lecture notes are contained in):

```
$ cd tex
$ pdflatex hello.tex
```

This creates `hello.pdf`. To be explicit, for our class, and if we wanted to use Emacs on Rice (which is already installed with AUCTeX)

```
ssh <sunetid>@rice.stanford.edu
# Login and Navigate to your favorite working directory.
emacs hello.tex
# Create file just as above. Execute: C-c C-c. This  compiles a pdf.
```

Then, one could go to https://afs.stanford.edu/ to view the pdf document that was created on `rice`. If we wanted to perform the entire process locally, we could install LaTeX on our machine and forego having to `ssh` into `rice` and then fetch our `.pdf` via an online browser (or perhaps `scp`).

3 Overview of LaTeX

There's a lot of formatting options in LaTeX, and it can feel overwhelming at first.

3.1 Document Formatting

3.1.1 Document Class - Specifying Page Layout

A document class defines the layout for a page. Options include:

- `article` - general purpose class for publications, short reports, etc.
- `proc` - proceedings (based on the article class)
- `report` - for longer reports with several chapters, small books, and theses.
- `book` - for real books.
- `slides` - for presentation slides. See as well: beamer.
- `letter` - writing letters.

For each of these classes, there are options that can be applied. E.g. we can choose the font, the paper size, whether to use one or two columns, one or two-sided outputs, landscape mode, etc.

Custom Classes A class can be defined using a `.cls` file. Various organizations will also distribute customized document classes for various purposes. These can be useful if you plan to submit your work to a particular vendor or publisher. E.g.

- SIAM LaTeX: https://www.siam.org/journals/auth-info.php.
- Stanford PhD thesis template: https://library.stanford.edu/research/bibliography-management/latex-and-bibtex.

3.2 Introduction to Custom Formatting

3.2.1 White space

White space is normalized so `1` to `n` spaces are treated the same.

```
\documentclass{article}
\begin{document}
There       are       lots       of       spaces       here.
\end{document}
```

spaces.tex

There are lots of spaces here.

Figure 2: The space operator acts the same whether used once or consecutively.

If you want to add *additional* spacing, there's many different ways to do this. You could, for example, use either to indicate additional separation, or you could use `\hspace{length}`, where `length` is an argument such as inches, millimeters, or points. E.g. `\hspace{2pt}`, or `\hspace{1cm}` are valid.

3.2.2 Paragraphs

Paragraphs are delimited by multiple line breaks.

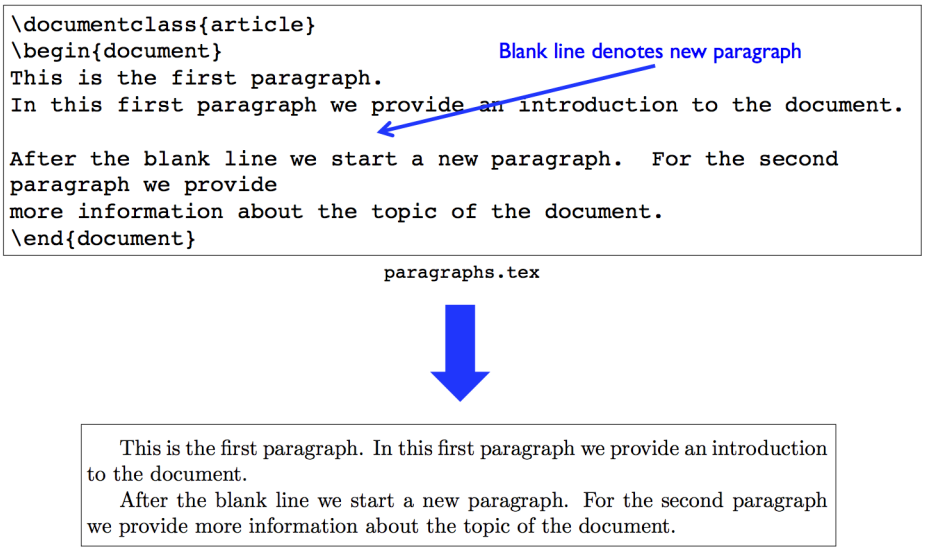


Figure 3: Multiple line breaks are required in order to indicate a paragraph break.

3.2.3 Bulleted list

We use `itemize` for lists, which can of course be nested.

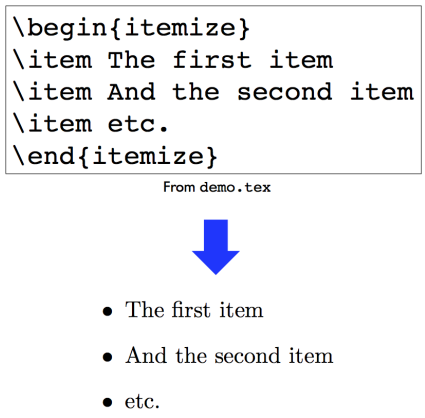


Figure 4: Bulleted lists.

If we wish for enumerated items, we can simply use `\begin{enumerate}` and `\end{enumerate}` to delimit our list instead, still using the same syntax to distinguish elements.

3.3 Special characters

There are several reserved characters in LaTeX. These take on a different meaning from their literal counterparts.

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

3.3.1 Comments

Anything which follows a percent symbol (%) is treated as part of a comment, i.e. the text which follows it is not treated as content of the document.

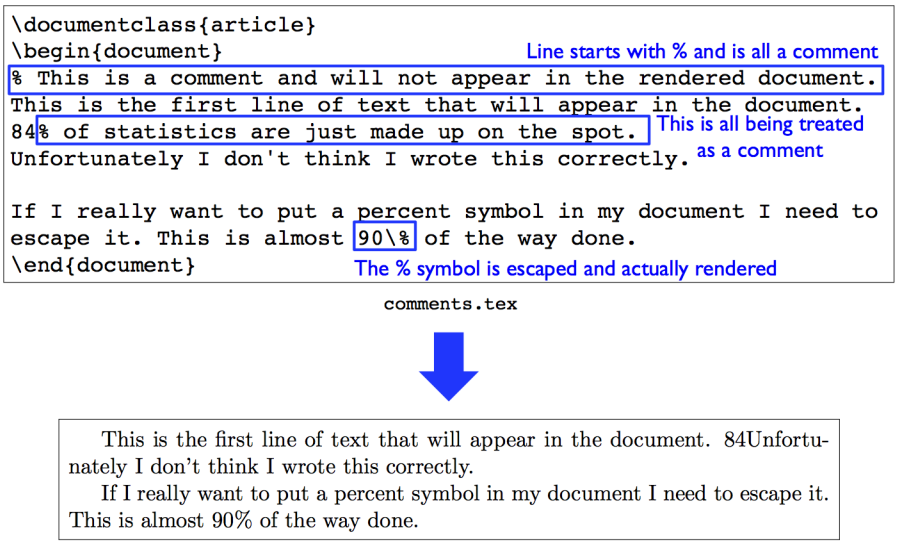


Figure 5: Comments in LaTeX. Note that anything after the percent symbol (%) is treated as a comment. To treat a special character as a literal character, we must escape it using a backslash, i.e. \.

3.3.2 Groups

Pairs of curly brackets denote a group and are typically used to limit the scope of switches:

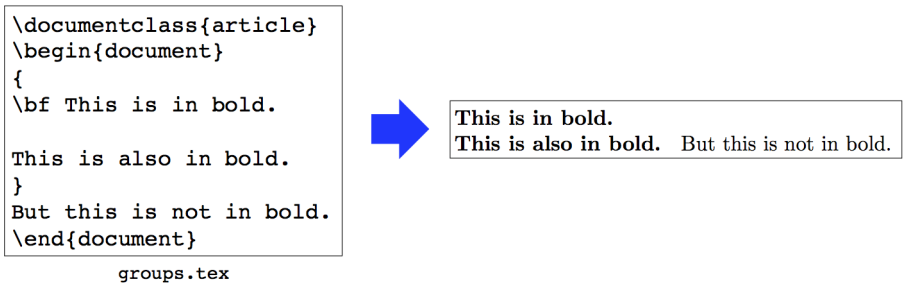


Figure 6: Groups can be delimited by curly braces.

You'll notice above that we used a *command* to switch to boldface text, denoted by `\bf`; this is an example of a font style.

3.4 Commands

The general syntax for a LaTeX command is

```
\commandname[option1,option2,...]{argument1}{argument2}...
```

The square brackets delimit optional input arguments, whereas the curly braces indicate required arguments.

Command Example Suppose we want to italicize a selection of text. As opposed to a What You See Is What You Get, e.g. Microsoft Word where we may highlight a selection of text and then click the **bold** icon to display emboldened text, LaTeX uses a special syntax within the document to signify that a sequence of characters shall be boldened.

```
\documentclass{article}
\begin{document}
Some of this text has been \emph{emphasized}.  Other parts have not.
\end{document}
```

emph.tex

Some of this text has been *emphasized*. Other parts have not.

Figure 7: In the above example, we've used an `\emph` command to signify that the text contained as input argument should be italicized.

3.5 Environments

Environments perform an action on a chunk of text; we can think of an environment as delimited by two commands.

```
\begin{environmentname}
Text to be influenced by this environment
\end{environmentname}
```

Everything within the environment is treated as an argument to be acted upon, in a sense. We might ask, why not simply use a single command with an argument which mimics what would otherwise be the body of an environment? The answer gets into technical implementation details, but we can think about environments as being more user-friendly when the argument is intended to be very long, e.g. a paragraph or an algorithm.

3.6 Math and Equations in LaTeX

Even though we may not have learned yet how to use limits, the below code should be reasonably easy to interpret, especially alongside the output.

```
\begin{equation}
\int_0^\infty e^{-x^2} dx=\frac{\sqrt{\pi}}{2}
\label{eq:integral}
\end{equation}
```

Label to be used for referring to this equation from the text

From `demo.tex`

$$\int_0^\infty e^{-x^2} dx = \frac{\sqrt{\pi}}{2}$$

Figure 8: Out of mathematical curiosity, we remind ourselves of error function and standard normals. The example showcases the use of metacharacters ^ and _ to provide superscripts and subscripts; notice that we can use them both concurrently and the result is nicely formatted. The example also demonstrates that we can nest exponents, or use commands as input arguments, e.g. $\sqrt{\pi}$.

Operators as Distinguished From Variables One small error in this example: operators should appear upright in order to differentiate themselves from variables. Although this may seem like a nuanced opinion, it's its in fact a standard by ISO 31. LaTeX is nice in that it affords us such fine grained control: to write a differential operator, we can use `\mathrm{d}x` to produce $\mathrm{d}x$. If we'd like a shorthand, we can place `\def\D{\mathrm{d}}` in our preamble before the `\begin{document}` in our `.tex` file, wherein we have access to a new command anywhere in the body of our TeXdocument, e.g. we can type `\D x` $\rightsquigarrow \mathrm{d}x$.

3.6.1 Personal Favorites for Formatting

I am fairly particular about formatting my mathematics, whether it be for a problem set or project proposal: I want the details to be clear and accessible. A couple tools I find helpful are: (i) the `align*` environment for creating mathematical equations broken across multiple lines which are aligned in a particular way (possibly with comments), and (ii) the `underbrace` command which can be used to explain particular terms. Example from branching theory:

$$\begin{aligned}
G_{n+1}(z) &= \mathbb{E}[z^{X_{n+1}}] && \text{Definition of } G_{n+1}(z) \\
&\dots \\
&= \mathbb{E}\left[\prod_{i=1}^{X_n} \underbrace{\mathbb{E}[z^{Y_i^{(n)}}]}_{=G(z)}\right] && \text{Since } \dots \\
&= G_n\left(G(z)\right) && \text{By definition of } G_n(z) = \mathbb{E}[z^{X_n}]
\end{aligned}$$

This was created using

```
\begin{align*}
G_{n+1}(z) &= \mathbb E[z^{X_{n+1}}] &\textrm{Definition of } G_{n+1}(z) \\
           &\ldots                                                        \\
           &= \mathbb E \left[ \prod_{i=1}^{X_n}
                \underbrace{\mathbb E[z^{Y_i^{(n)}}]}_{=G(z)} \right]
           &\textrm{Since } \ldots                                        \\
           &= G_n \left(G(z) \right)
           &\textrm{By definition of } G_n(z) = \mathbb E[z^{X_n}]
\end{align*}
```

In using an `align*` environment (as opposed to `align`), we are requesting to forego numbering our equations. The `&`'s are used to align equations, where the first & aligns based on its appearance in the first line, and subsequent &'s on each line indicate right-alignment for a comment. We've added a sub-text to our underbrace using a simple _ underscore modifier.

3.7 Latex packages

Many LaTeX environments are defined in packages. To include a package use the `\usepackage` command in the document preamble. The `tex/demo.tex` document uses a few:

```
\usepackage{graphicx}
\usepackage{algorithm2e}
```

These *must* be placed *before* the `\begin{document}` command.

- `graphicx` provides the `\includegraphics[scale=0.5]` command for figures
- `algorithm2e` provides an environment for displaying algorithms

You can even define your own packages; it's quite similar to creating a Python module. Simply place preamble into a `.sty` file, and this can be then used as a package. To use a style file in a single project, just place the `.sty` file in the corresponding project directory. To use the style file across projects, place the style file in your TEXhome directory. Can you guess how I ensure the lectures are somewhat consistent in style?

3.8 Figures

Figures can be tricky to learn at first: the following example uses multiple environments or commands, some of which can seem redundant at first glance.

```
\begin{figure}[htb]   Placement control (more on this later)
\begin{center}
\includegraphics[width=0.85\linewidth]{maze1.pdf}
\caption{Maze}        Scale the image to 85% of
\label{fig:maze1}       the line width
\end{center}
\end{figure}
```

From `demo.tex`

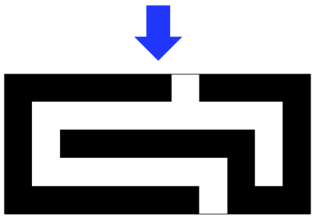


Figure 1: Maze

Figure 9: The *figure* environment ensures that images aren't split across pages, and allows captions and references. The `\center` environment ensures that we vertically align the figure on the page. We use the `\includegraphics` command to import an actual image, where here we've specified to shrink the image somewhat. Lastly, the `\label` command allows us to create a cross-reference for later use.

3.9 Tables

These can also be a bit overwhelming to learn at first, where similar to our example on figures it may seem there are redundant commands used.

```
\begin{table} [htb]
\begin{center}
\caption{Dataset Characteristics}
\begin{tabular}{|c|c|c|}   Three centered columns with vertical lines around them
\hline
Dataset number & Reference length & Number of reads \\  \hline
1              &   100            &   60            \\  \hline
2              &  1000            &  600            \\  \hline
3              & 10000            & 6000            \\  \hline
\end{tabular}
\label{table:datasets}
\end{center}
\end{table}
```

From `demo.tex`

Table 1: Dataset Characteristics

Dataset number	Reference length	Number of reads
1	100	60
2	1000	600
3	10000	6000

Figure 10: The `table` environment declares a `floating` object, i.e. one that cannot be split across pages. The `caption` command is effectively used as an input to our `table` environment. What's new here is the use of `tabular` to define a table environment. The required arguments for this environment are the number of columns in the table, and how they shall be aligned or delimited. I.e. we use one of {`l`,`c`,`r`} to denote left, central, or right alignment within a column; using a pipe character `|` signifies to delimit the columns by a vertical partitioner. In contrast to the number of columns (fixed by a required input argument to the `tabular` environment), the number of *rows* is determined by the body of the environment. We use the & character to delimit columns within a row, and two adjacent backslashes delimits the end of a row.

3.9.1 Controlling Placement

By default figures, tables, etc. will "float" around to where they best fit. But, you can also specify preferences about placement. Floating environments take a parameter in square brackets: `\begin{figure}[?]`. The options are:

- `h` for "float here"
- `t` for "top of page"
- `b` for "bottom of page"
- `H` for "put here, don't float"

Good figure placement often requires some experimentation. Advice: write the document first. Make it look nice second. Things will change as you add more text and figures.

3.10 Referencing Labels

We may use `\label` and `\ref`.

```
With LaTeX you can create equations
as shown in Equation~\ref{eq:integral}:
%                          Reference the equation by label
\begin{equation}
\int_0^\infty e^{-x^2} dx=\frac{\sqrt{\pi}}{2}
\label{eq:integral}  Give the equation a unique label
\end{equation}
```

From `demo.tex`

Figure 11: Using a `ref` to mark a point of interest, and `label` to cite it. One great advantage here when compared with conventional word processors is that all of our linking is done dynamically. E.g. if we choose, we can use a table of contents or table of figures, and the page-numbering and table/figure numbers will be automatically updated whenever we update the document. This can save against some nasty errors wherein we may insert a new figure into the body of our document but forget to update the table of contents.

The first pass of LaTeX will produce an unresolved reference:

Unresolved reference

formatting. With LaTeX you can create equations as shown in Equation ??:

$$\int_0^\infty e^{-x^2} dx = \frac{\sqrt{\pi}}{2} \tag{1}$$

Figure 12: An example of what an unresolved reference looks like within a rendered TEX document.

The second pass of LaTeX will actually resolve the reference.

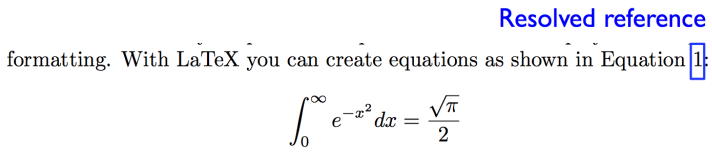


Figure 13: An example of what the corresponding resolved reference might look like in our rendered example.

3.11 Algorithms

Another fantastic use of TEX is in describing sub-routines and algorithms. I preference the `algorithm2e` package; it is quite flexible and formats nicely.

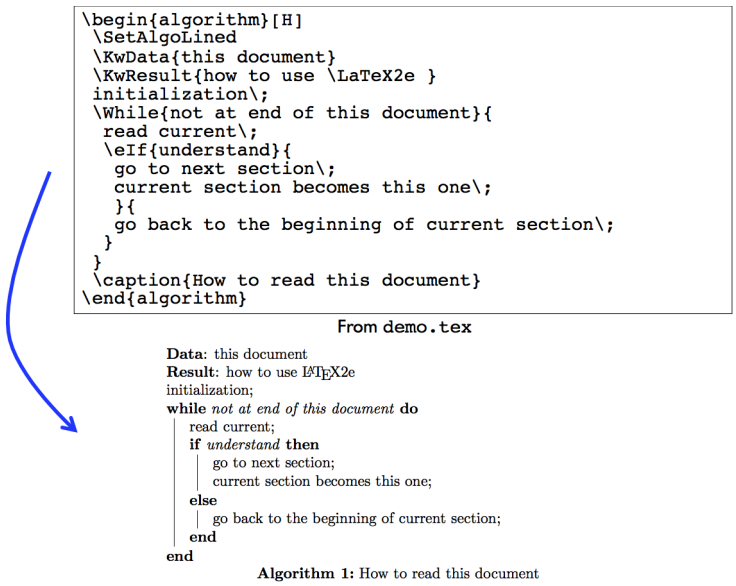


Figure 14: An example of the `algorithm` environment provided by `alborithm2e` package. Notice that we can declare our input data and output results, which are essential for those unfamiliar with the sub-routine to understand its high-level purpose. We also have access to control-flow structures, and we can use all of our usual TEX symbols or mathematical formatting. Not featured in the above example: I like to use `\tcp{comment}` to include inline comments using the C++ style of `//`.

4 Bibliographies

4.1 BibTex

This is a companion program for managing citations of papers, books, websites, etc. We start by creating a `.bib` file. E.g. see `tex/references.bib`:

```
@article{Ronaghi:2001:Pyrosequencing,
        Author = {Mostafa Ronaghi},
        Journal = {Genome Research},
        Pages = {3--11},
        Title = {Pyrosequencing Sheds Light on DNA Sequencing},
        Volume = {11},
        Year = {2001}}

@misc{CME211:2013:FinalProjectPart1, Unique label
        Author = {CME211},
        Howpublished = {http://coursework.stanford.edu},
        Month = {November 15,},
        Title = {Final Project: Part 1},
        Year = {2013}}
```

references.bib

Figure 15: We demonstrate the formatting of a `.bib` file. For more details, see the BibTex file format specification.

Typesetting with BibTeX Reference This can be annoying at first! In order to get references correct, we must take a couple passes through the document.

```
$ pdflatex demo
$ bibtex demo
$ pdflatex demo
$ pdflatex demo
```

Although any text editor can be used to create, edit, and manage a `.bib` file, some editors will recognize the file extension and enable a BibTex specific mode. There are also applications *specifically* for doing this: BibDesk (MacOSx only), Jabref (any OS), and Mendeley.

4.2 Citations

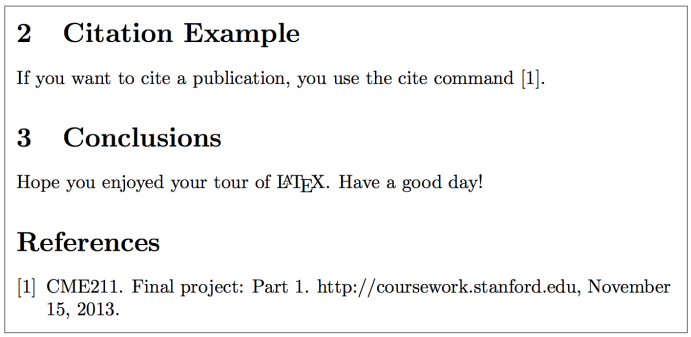
Citations in LaTeX simply use the `cite` command, where the argument specifies which entry in the BibTex file to use.

```
\section{Citation Example}

If you want to cite a publication, you use the cite
command~\cite{CME211:2013:FinalProjectPart1}.
          Label must match the label in the .bib file
\section{Conclusions}

Hope you enjoyed your tour of \LaTeX.  Have a good day!

% Reference section
\bibliographystyle{unsrt}   Specify style (unsrt) and name
\bibliography{references}   of .bib file (e.g. references.bib)

\end{document}
```

From demo.tex

Figure 16: We use `cite` to indicate a reference toward a bibliographic citation. This is different from referencing a labeled object, described earlier in section 3.10!

Resulting PDF:



2 Citation Example

If you want to cite a publication, you use the cite command [1].

3 Conclusions

Hope you enjoyed your tour of LaTeX. Have a good day!

References

[1] CME211. Final project: Part 1. http://coursework.stanford.edu, November 15, 2013.

From `demo.pdf`

Figure 17: We emphasize that the references are dynamic. Want to switch to lexicographic ordering of listings in the bibliography? No need to switch all of our citations, simply update the bibliography style in a single place! Re-compiling our document (a couple times) will yield updated references.

5 A note on LaTeX errors

LaTeX will dump error messages and start a prompt on errors:

```
$ pwd
/Users/nwh/Dropbox/courses/2015-Q4-cme211/lecture-prep/lecture-13-work/tex
nwh-mbpro:tex nwh$ pdflatex demo
This is pdfTeX, Version 3.14159265-2.6-1.40.16 (TeX Live 2015) (preloaded format=pdflatex)
.....
) (./demo.aux) (/usr/local/texlive/2015/texmf-dist/tex/latex/base/omscmr.fd)
! Undefined control sequence.
l.35 \includegraphics[scale=0.5]
                      [width=0.85\linewidth]{../fig/maze.pdf}
?
```

LaTeX is prompting you how to continue. See LaTeX errors and warnings. We may type `x` to e**x**it the program, `q` to **q**uietly carry on with execution, or for example `h` for more **h**elp. In our example, we could type `x` to quit the compilation process, then fix the error in the `.tex` file and try again. In this case, we forgot `\usepackage{graphicx}` in the preamble.

References: Apart from Google,

- Guide to LaTeX by Kopka and Daly: http://proquest.safaribooksonline.com/book/graphic-design/9780321617736
- **Detexify**: a web tool to go from symbol drawing to TeX command: http://detexify.kirelabs.org/classify.html
- LaTeX Wikibook: https://en.wikibooks.org/wiki/LaTeX
- A Gentle Introduction to TeX: http://texdoc.net/texmf-dist/doc/plain/gentle/gentle.pdf
- How to draw in TeX: http://www.texample.net/tikz/