Software for Mathematical Scientists and Educators

Joshua Maglione

January 28, 2024

Contents

Introduction

Communicating mathematics and performing long computations are both vitally important and challenging. Thankfully there is a wide selection of software to make these tasks more manageable. In this module, we will explore software used by everyday mathematicians and scientists. These include biologists, chemists, computer scientists, data scientists, financial analysts, your friends and family, educators, engineers, and physicists.

The goal is to build a foundation by using some of the most ubiquitous software in the field. This will help students throughout their career in and out of university. We will cover four topics in this module:

1. Mathematical Typesetting and LaTeX,
2. Python and Jupyter Notebooks,
3. Introduction to Programming,
4. Symbolic Computation and SageMath.

1 Mathematical Typesetting and LaTeX

With advances in printing and typesetting, the question of how to produce high-quality mathematical symbols and texts is challenging. Without going through the history, we now have essentially two main styles of software to write mathematical formulae, diagrams, and images:

1. What-You-See-Is-What-You-Get (WYSIWYG) and

2. typesetting software (or write-format-preview style).

Software like Microsoft Word, Apple Pages, or LibreOffice Writer are WYSIWYG editors because you see and edit the document as a final product (regardless of whether or not it is the final product). This remove the user from having to remember commands for the document layout, and it has a lower barrier of entry, which is one of its strongest advantages. However, this is not the norm for professionals using many mathematical symbols, and the primary disadvantage to these kinds of software is their sluggish pace—the secondary disadvantage is that many people struggle to pronounce WYSIWYG causing people to avoid such software. I certainly cannot pronounce it properly.

One of the first typesetting software for mathematics is TeX—if not *the* first. It was written by Donald Knuth[1] in 1978 [4], and it is the cornerstone of the more modern software LaTeX written by Leslie Lamport in 1986 [8]. Although TeX is still used today, LaTeX is far more popular. The primary disadvantage to these systems is the higher barrier to entry, but the over-powering advantage is the speed with which one can produce beautiful and high-quality mathematical symbols. Both TeX and LaTeX are free and open-source software.

As alluded to previously, these are typesetting software, so a user writes in a markup language, usually in a `tex` file; then the user compiles the file, and a `pdf` file is produced. There are other options for output, but this will be sufficient for us. Another asset is that one can import third party LaTeX packages to perform more specialized tasks. We will become familiar with basic LaTeX formatting and use some packages to help construct beautiful documents.

For web-based mathematical symbols, MathJax [9] is primarily used. However, there is a new, much faster, alternative called KaTeX [6]. Currently, KaTeX can only do a (proper) subset of what MathJax can do, but KaTeX does enough for everything I have needed for my website—and I have a lot of complicated formulae on my website.

Remark 1.1. Nobody really cares how you pronounce LaTeX, but some of the popular ways are "law-tech" and "lay-tech". This is because Knuth indicated that TeX ought to be pronounced like "tech" [7, Paragraph 2]. The true non-conformists pronounce it "lay-techs". I just prefer to pronounce it as LaTeX.

[1]You can read Knuth's original memo describing TeX. He called the memo the "Preliminary preliminary description of TEX" [7].

1.1 How to get LaTeX

This will not fully cover how to install LaTeX on your own machine, but hopefully it gives you enough information to help you. It not that LaTeX updates so frequently—in fact LaTeX is still on its second edition (formatted as LaTeX 2$_\varepsilon$)—it is that packages tend to update or just need installation.

For Windows machines, I would recommend MiKTeX. This install LaTeX and comes with a package manager to help with package installation. A similar version is available on Mac OS, and it is called MacTeX. Both MiKTeX and MacTeX have graphical user interfaces. For Linux systems, TeX Live is what I recommend, and it usually comes pre-installed. It has its own package manager invoked by the command `tlmgr`.

We will primarily use Overleaf in this module. Overleaf is a website that enables users to interface with LaTeX through cloud-based services. It uses a "freeium" model, so that everyone can use the basic features, which will be sufficient for our module. The major advantage is that one does not have to worry about installing and package management; all of this is done cloud-side. Moreover Overleaf simplifies the workflow slightly by allowing for instant compilation. #NotSponsored

1.2 Workflow and document structure

The basic workflow is perhaps only a little more complicated than how it might be for WYSIWYG software. Here is the basic workflow.

1. Create a `tex` file and write LaTeX markup.
2. Compile the `tex` file with the command `pdflatex`.
3. Sometimes errors are raised and need to be addressed. It is acceptable to cry when this happens; it happens to all of us. If no errors arise, then a `pdf` file is created (or overwritten).
4. View and review the output `pdf` file.

With the exception of the initial creation of the file, all of these steps are repeated often. How often? That depends on you and your situation. I would recommend that, when addressing bugs or typos, that one compile often and review carefully what has changed.

The basic format of a LaTeX document is simple. There are three commands that must be present, and often one tries to adhere to some logical structure due to collaborations or just to readability over longer periods of time. The first command that appears in a functioning `tex` file is the following:

```
\documentclass[<options>]{<style>}
```

where `<style>` is replaced with the name of the desired document class and `<options>` is replaced with optional parameters for the specific document class. For example, this current `pdf` document was built using

```
\documentclass[a4paper, 12pt]{article}
```

Some other popular document classes, besides `article`, are

- `beamer` : for slides,
- `book` : for books,
- `letter` : for letters,
- `standalone` : often used with the package `TikZ` for stand alone pictures.

There are also AMS-inspired article and book classes: `amsart` and `amsbook` and KOMA-script versions: `scrartcl`, `scrbook`, `scrlttr2`, and `scrreprt`. The other two required commands are

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

which encapsulate the main body of the `tex` file. Figure 1.1 shows the basic layout of a `tex` file.

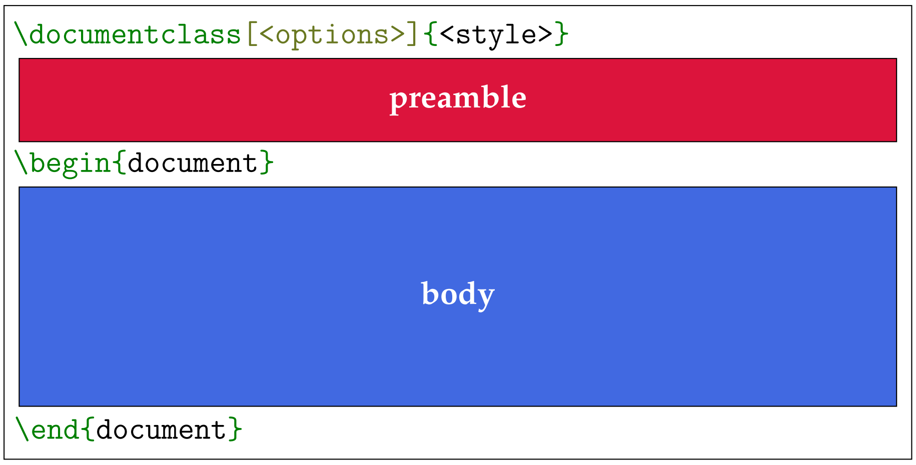

Figure 1.1: Basic structure of a `tex` file.

Although I have never heard anyone refer to the preamble as the head, I think it makes sense—even if it causes me to stop and use my preamble to figure out what is meant by it.

1.3 Source files

LaTeX *source files* are the necessary files needed to compile a `tex` file. In particular, a `tex` file need not be self contained; there are many reasons why this might be the case. For example, one might embed graphics in the form of `jpeg` or `png` files, or one might use BibTeX to dynamically format their bibliography—more on this in Section 1.5.

In the process of compiling a `tex` file to a `pdf` file, LaTeX produces several auxillary files. These serve specific purposes, but their particular uses are outside of the scope we will explore. Because so many additional files are created in the compilation process, it is usually preferred to build a directory for each LaTeX project. For example, the file names in the directory for these lecture notes can be seen in Figure 1.2.

```
LectureNotes
  bibliography.bib               LectureNotes.log
  LectureNotes.aux               LectureNotes.out
  LectureNotes.bbl               LectureNotes.pdf
  LectureNotes.blg               LectureNotes.tex
  LectureNotes.fdb_latexmk       LectureNotes.toc
  LectureNotes.fls               LectureNotes.synctex.gz
```

Figure 1.2: The files in the `LectureNotes` directory.

The only files necessary to compile in Figure 1.2 are `LectureNotes.tex` and `bibliography.bib`. The rest of the files are produced by the compilation process.

1.4 An introduction to LaTeX syntax

We will not cover all of the LaTeX syntax here. When we discuss mathematical symbols and formulae in Section 1.6, we will cover more. There are some symbols that LaTeX redefines. For example, `%` indicates that the rest of the line should be ignored by the compiler:

```
% This is a comment and will be ignored by the
% LaTeX compiler.
```

Commands in LaTeX start with the `\` symbol. We have already seen examples of this with the following commands:

```
\documentclass[a4paper, 12pt]{article}
\begin{document}
\end{document}
```

Arguments are input using the `{` and `}` symbols, and multiple arguments would require multiple sets of `{` and `}`. Optional parameters are input using the `[` and `]` symbols, but multiple optional parameters are listed within the single use of `[` and `]` separated by commas. Not every command requires input arguments; for example, to format LaTeX as LaTeX, one uses the command `\LaTeX`.

One often organizes an article into sections, subsections, subsubsection, etc. Similarly for books into parts, chapters, sections, etc. These can be easily managed using standard commands. For example in an `article` document class, one can use

```
\section{<title>}
\subsection{<title>}
\subsubsection{<title>}
```

to produce the desired outcome. For parts and chapters, analogous commands exist, but errors might be raised if not in a document class that supports them. For example, `article` does not support `\chapter{<title>}`.

Many document classes also have their own way of formatting elements like the title, author, and date of a document. To access this, usually one includes the following in their preamble.

```
\title{<title>}
\author{<author>}
\date{<date>}
```

One can type `\date{\today}` to display today's date, provided the document is compiled today. To display the title, author, and date, include the following command in the body of the document.

```
\maketitle
```

1.5 Dynamic bibliographies and BibTeX

Another major advantage to LaTeX is its ability to dynamically format bibliographies and references. There are a number of different ways to format bibliographies in LaTeX, but we will discuss specifically BibTeX, written by Oren Patashnik and Leslie Lamport in the 1980s.

Have you ever written a document, and after having gotten halfway through, say, you decide to add another reference? Depending on the style adopted in your document, this might have caused you lots of additional work. Not one would you have to potentially re-order your bibliography, but you might have to change several references to account for this. Having a dynamically built bibliography avoids all of this pain.

To use BibTeX, one simply includes two lines wherever the references should go—usually at the end of the document:

```
\bibliography{<bib file name>}
\bibliographystyle{<style>}
```

Thus, in order to use BibTeX one needs an additional source file: a `bib` file. A `bib` file collects bibliography entries that BibTeX can read. Figure 1.3 provides an example.

```
@book{Macdonald1998,
    title={Symmetric functions and Hall polynomials},
    author={Macdonald, Ian Grant},
    year={1998},
    publisher={Oxford university press}
}
```

Figure 1.3: A sample BibTeX entry.

In the first line of Figure 1.3, `@book` tells BIBTEX that the bibliography entry is a book. The braces indicate the block of code designated for this particular entry. And the `Macdonald1998` is the *tag*—more on this soon. The lines in between the braces give bibliographical information. As the example illustrates: we have the book's title, author, year of publication, and publisher. There are more potential entries one could include as well. The entry will be formatted relative to the specified style in the output `pdf` file.

When one wants to cite a particular reference in the body of their `tex` file, one simply writes

```
\cite[<options>]{<tag>}
```

and a citation for that particular entry (relative to the tag) matching the given style is produced. Using the example in Figure 1.3, we might write the following sentence in the body of our `tex` file:

```
By \cite[Theorem 3.14]{Macdonald1998}, the proof is complete.
```

There are many different styles for BIBTEX. Here are a few examples.

- `plain` : This is one of the original BibTeX styles. It formats entries in a simple, straightforward manner and sorts them alphabetically by author.
- `alpha`: This style creates labels from the author's name and the year of publication. It's handy for author-year citations.
- `abbrv`: Similar to the plain style, but it abbreviates first names and journal names. It's useful for more compact bibliographies.
- `IEEEtran`: Used for formatting bibliographies in the style of the Institute of Electrical and Electronics Engineers (IEEE).
- `acm`: For formatting according to the style guidelines of the Association for Computing Machinery (ACM).
- `apa`: Adheres to the formatting guidelines of the American Psychological Association. Requires additional packages like `apacite`.
- `nature`: Mimics the citation style of the journal 'Nature'. This style is often used in scientific publications.
- `chicago`: This style follows the guidelines of the Chicago Manual of Style. Useful for humanities and social sciences.
- `mla`: Suitable for documents adhering to the Modern Language Association format, commonly used in the humanities.
- `amsplain`: Used for formatting in the style of the American Mathematical Society, particularly suitable for mathematics and related disciplines.

The `bib` file does not need to be ordered. BibTeX will take care of ordering all of the bibliography entries properly. Therefore, the `bib` file can simply be a dump for bibliography entries. All a user needs to know are the relevant tags for the references to cite.

Using BibTeX modifies the basic workflow described in Section 1.2. After running `pdflatex` one needs to run `bibtex` twice, and then `pdflatex` again. Often more crying ensues. Overleaf and other similar programs can run all of this with one button, so we will not dwell on this.

Remark 1.2 (Warning)**.** BibTeX can be challenging for newcomers to work with for many reasons. Some of the pain is alleviated by Overleaf and other programs, but this still does not make it easy.

1.6 Mathematical symbols and formulae

LaTeX provides two primary methods to output mathematical symbols: inline and as a displayed line. To display maths inline, one initiates with `$`, ending with the same symbol. One can also start with `\(` and end with `\)`. Thus, `$e^x$` is displayed as e^x. The Greek alphabet is accessed by commands like `\pi` for π and `\Pi` for Π.

Exponents are indicated by `^` and subscripts by `_`. Without additional braces, both `^` and `_` change only the next character. Note the difference in `$e^{2i\pi}$` as $e^{2i\pi}$ as compared to `$e^2i\pi$` as $e^2i\pi$. The same holds for subscripts. One can use both exponents and subscripts; for example `$M_{ij}^4$` is displayed as M_{ij}^4. By using braces one can nest these operations. For example x^{x^x} is `$x^{x^x}$` and $x_i^{x_j^2+x_k^2}$ is `$x_i^{x_j^2 + x_k^2}$`.

Note that LaTeX adjusts the height of each line relative to its content. For this reason and others, some might prefer to display more complicated expressions in its own displayed line. To accomplish this in LaTeX one uses `$$` or `\[`, ending with either `$$` or `\]`, respectively (and not mixing). For example,

```
\[
    x_i^{x_j^2 + x_k^2}.
\]
```

produces

$$x_i^{x_j^2+x_k^2}.$$

A few more examples of displayed mathematics include the following.

```
\[
    \int_a^b f'(t) dt = f(b) - f(a).
\]
```

produces

$$\int_a^b f'(t)dt = f(b) - f(a).$$

And

```
\[
    \lim_{n\to \infty} \sum_{k=1}^n \frac{1}{k} \rightarrow \infty.
\]
```

yields

$$\lim_{n\to\infty} \sum_{k=1}^n \frac{1}{k} \rightarrow \infty.$$

Notice that the previous example looks differently when typed inline:
`$\lim_{n\to \infty} \sum_{k=1}^n \frac{1}{k} \rightarrow \infty$`
is displayed as $\lim_{n\to\infty} \sum_{k=1}^n \frac{1}{k} \rightarrow \infty$, not as elegant I would say.

Remark 1.3. There is a heated debate whether or not one should use `$$` or the pair `\[` and `\]`. It is very controversial, and by now you see where I stand. (This is of course a joke—perhaps some people are loyal to a particular style, but it is completely pointless. Bring on the haters.)

1.7 LaTeX Environments

One of the general blueprints of LaTeX markup is an *environment*. A LaTeX environment has the following basic structure:

```
    \begin{<environment>}       % Explicit start of environment
        %
        % Content
        %
    \end{<environment>}         % Explicit end of environment
```

One could say that a `tex` file is essentially a LaTeX environment using `document`, but people do not say that. That would be ridiculous; let's not promote that.

1.7.1 The `equation` environment

The displayed lines of math from Section 1.6 can be viewed as an environment. One can even do this in an environment format. For example,

```
    \begin{equation}
        \sum_{n=1}^\infty \frac{1}{n^s}
            = \prod_{p \text{ prime}} \frac{1}{1 - p^{-s}}
    \end{equation}
```

yields

$$\sum_{n=1}^\infty \frac{1}{n^s} = \prod_{p \text{ prime}} \frac{1}{1 - p^{-s}} \tag{1.1}$$

Remark 1.4. As a digression, notice in the previous example that there is a reference number to the right of the equation, namely (1.1). If we provide a label for the equation, we can reference that label anywhere in the document and it will produce the correct reference number. For example

```
    \begin{equation}\label{eqn:Euler-decomp}
        \sum_{n=1}^\infty \frac{1}{n^s}
            = \prod_{p \text{ prime}} \frac{1}{1 - p^{-s}}
    \end{equation}
```

yields

$$\sum_{n=1}^\infty \frac{1}{n^s} = \prod_{p \text{ prime}} \frac{1}{1 - p^{-s}} \tag{1.2}$$

and we can reference it with `\ref{eqn:Euler-decomp}` producing 1.2. Equations usually have special formatting—note the parentheses—so to account for this, there is a special way to reference equations: `\eqref{eqn:Euler-decomp}` yields (1.2).

1.7.2 The `align` environment

Another mathematical environment is `align`. *This uses a very common package that should be used every time. More about this in Section 1.8.*

```
\begin{align}
    \sum_{n=1}^{\infty} \frac{1}{n^2}
    &= 1 + \frac{1}{4} + \frac{1}{9} + \cdots \\
    &= \frac{\pi^2}{6} .
\end{align}
```

yields

$$\begin{align} \sum_{n=1}^{\infty} \frac{1}{n^2} &= 1 + \frac{1}{4} + \frac{1}{9} + \cdots \tag{1.3} \\ &= \frac{\pi^2}{6}. \tag{1.4} \end{align}$$

A few aspects to note. First observe that each line has a reference number; see Remark 1.4 for a brief explanation on how to take advantage of these reference numbers. (To actually reference that particular remark, I am using label and ref commands. When does the rabbit hole end?!) The standard way to turn off these reference numbers, for example if they are unused or unwanted, is to append the environment name with `*`. Thus, `equation*` and `align*` will not produce reference numbers on the side.

1.7.3 The `center` environment

The `center` environment center justifies the text within the environment. For example

```
\begin{center}
    This text is centered.
\end{center}
```

produces

This text is centered.

1.7.4 the `tabular` environment

The `tabular` environment is used for creating tables in LaTeX. It allows one to specify the alignment, formatting, and content of each cell within the table. For example, we can create a center-justified table with the following code:

```
\begin{center}
\begin{tabular}{|l|c|r|}
    \hline
    \textbf{My} & \textbf{Favorite} & \textbf{Table} \\
    \hline
    Look, & Ma & no \\
    \hline
    hands! & he & shouted \\
    \hline
    before & crashing & (more data) \\
    \hline
\end{tabular}
\end{center}
```

It produces the following table:

My	**Favorite**	**Table**
Look,	Ma	no
hands!	he	shouted
before	crashing	(more data)

The argument `{|l|c|r|}` indicates that the table should have four vertical lines; the cells in the first column should be left-justified; the cells in the second column should be center-justified; and the cells in the third column should be right-justified. The `\hline` command is used to draw horizontal lines between rows. The symbol `&` is used to separate cells within a given row, and the symbol `\\` is used to create new rows in the table.

1.7.5 The `enumerate` and `itemize` packages

The `enumerate` environment is employed to create ordered lists. It enables the user to specify items within the list, automatically numbering them in the desired format—it's magical moment when you do not have to renumber items in your list. The following

```
\begin{enumerate}
    \item First item,
    \item Second item,
    \item Another item,
    \item I am losing count,
    \item but it does not matter because I do not,
    \item have to,
    \item count!
\end{enumerate}
```

produces the following numbered list:

1. First item,
2. Second item,
3. Another item,
4. I am losing count,
5. but it does not matter because I do not,
6. have to,
7. count!

Notices the `\item` command is used to delineate individual entries within the list. An `enumerate` without an `\item` is like a goose around her goslings—easily angered, it's best to steer clear from them, trust me.

One can skip or simply change the numbering using `\setcounter`, but I will not go into the details here. There are plenty of examples on the Internet.

You can nest enumerate. For example:

```
\begin{enumerate}
    \item Look at me.
    \item \begin{enumerate}
        \item I can make
        \item nested enumerates.
    \end{enumerate}
    \item So cool.
\end{enumerate}
```

produces

1. Look at me.
2. (a) I can make

   (b) nested enumerates.
3. So cool.

The `itemize` environment works in much the same way as `enumerate` except that produces unordered lists. The default is bullet points.

1.8 Some LaTeX packages

LaTeX packages are essential tools that enhance the functionality of LaTeX. Packages are sets of additional commands and features that extend the capabilities of the basic LaTeX system, often for specialized tasks. Packages are, therefore, designed to address specific needs, making LaTeX highly versatile and adaptable to various document requirements.

To use a LaTeX package, you need to include it in the preamble of your document. Recall from Section 1.2 that the preamble is the part of the document after `\documentclass[<options>]{<style>}` and before `\begin{document}`. To include a package, one first needs it to be installed in the local machine—or one can use Overleaf and avoid thinking about this. Then in the document (often just after the `\documentclass[<options>]{<style>}`), one writes

```
\usepackage[<options>]{<package name>}
```

Sometimes the `<options>` are relevant, and sometimes they are not. For example, the first few lines of this document look like

```
\documentclass[a4paper, 12pt]{article}
\usepackage{amsmath}
\usepackage{amsthm}
\usepackage{amssymb}
\usepackage{enumerate}
\usepackage{hyperref}
\usepackage[margin=3cm]{geometry}
```

We will mention a few packages that might be particularly relevant. We will give brief summaries of their capabilities, but those interested should check the documentation—or probably more realistically, type exactly what you are looking for in Google and go from there.

1.8.1 The `amsmath` package

One of the most useful packages for mathematical output is `amsmath` [1]. It provides numerous environments and commands for formatting equations, aligning mathematical expressions, and handling mathematical symbols. One of the environments provided was discussed in Section 1.7: `align`.

The `amsmath` package provides environments for matrices: `matrix`, `bmatrix`, `pmatrix`. The following code

```
\[
    \begin{pmatrix}
        1 & 2 \\
        3 & 4 \\
    \end{pmatrix}
\]
```

produces

$$\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$$

One separates elements in a given row using the `&`, and one uses `\\` to start a new row. It additionally offers commands `\dfrac` and `\tfrac`, which are like `\frac`, the former is in display style and the latter in inline style.

1.8.2 The `amsthm` package

The `amsthm` package is specifically designed to facilitate the typesetting of theorems, lemmas, definitions, and similar structures in a document. It is part of the collection of LaTeX packages offered by the American Mathematical Society (along with `amsmath`), offering enhanced customization and consistency in the presentation of mathematical theorems and related structures.

The key features of the package are

1. Theorem-like Environments: Allows you to define new theorem-like structures (like Theorem, Lemma, Corollary, Proposition, Definition, Example, Remark).
2. Customization: Provides options to customize the style of theorem-like environments, such as italicized or normal text, numbering, and shared or independent counters.
3. Proof Environment: Offers a dedicated proof environment for typesetting proofs, automatically adding a QED symbol at the end.

One needs to first define the theorem-like environment. There are two main commands for doing this:

```
\newtheorem{<name>}{<printed output>}[<numbered within>]
\newtheorem{<name>}[<numbered like>]{<printed output>}
```

As an example, we define the following in our preamble.

```
\newtheorem{theorem}{Theorem}[section]
```

Now we can write theorems in a dedicated environment using

```
\begin{theorem}
    The equation $x^n + y^n = z^n$ has no solution in
    the positive integers for $n \geq 3$.
\end{theorem}
```

This yields the following.

Theorem 1.1. *The equation $x^n + y^n = z^n$ has no solution in the positive integers for $n \geqslant 3$.*

We can cross-reference by using `\label{<tag>}` and `\ref{<tag>}`. Let's label our theorem:

```
\begin{theorem}\label{Fermat}
    The equation $x^n + y^n = z^n$ has no solution in
    the positive integers for $n \geq 3$.
\end{theorem}
```

This produces

Theorem 1.2. *The equation $x^n + y^n = z^n$ has no solution in the positive integers for $n \geqslant 3$.*

We can reference Theorem 1.2 by `\ref{Fermat}`.

1.8.3 The `graphicx` package

The `graphicx` [3] package is a powerful tool for including and manipulating graphics (or images) within documents. It extends the basic capabilities of LaTeX by providing commands for inserting images, scaling, rotating, and controlling their placement. For example

```
\begin{center}
    \includegraphics[scale=0.2]{imgs/Sherlock.jpg}
\end{center}
```

produces a picture of my dog

1.8.4 What now?

There are over 4000 packages, so I guess we should go through them all one by one. This will surely be on the exam. I recommend, in particular, checking out the following packages—at least see how people might use these packages.

- `hyperref` : Enables references and citations to produce hyperlinks within the `pdf` file. (This file uses it.)
- `listings` : Displays code and pseudo-code in a structure block.
- `siunitx` : For those working often with SI units, this can help manage it all.
- `tikz` : Very powerful image producing package. The possibilities are stunning.

2 Python and Jupyter

Python [12], a high-level programming language, has established itself as one of the most popular and versatile languages. Known for its emphasis on readability and simplicity, Python's design philosophy revolves around easy-to-read code and a syntax that allows programmers to express concepts in fewer lines than possible in many other languages. This readability makes Python particularly appealing for beginners and professionals alike—may you never have to read code that is both bad and ugly!

One of the most significant advantages of Python is its vast ecosystem of third-party packages—called *modules*. This ecosystem includes powerful tools for a wide range of tasks, from web development with frameworks like `Django` and `Flask`, to scientific computing and data analysis with `NumPy`, `SciPy`, `pandas`, and `Matplotlib`. In the field of machine learning and artificial intelligence, Python is the language of choice, thanks to libraries such as `TensorFlow` and `Scikit-learn`.

Python's simple, easy-to-learn syntax, combined with its powerful libraries, makes it an excellent choice for automating repetitive tasks, rapidly prototyping applications, and building scalable, high-performance software. It's widely used in various domains, from web and software development to scientific research, data analytics, and beyond.

In the educational realm, Python's straightforward syntax has made it a favored language for teaching programming concepts. Its popularity in both industry and academia means that students learning Python are gaining a skill that is both academically enriching and practically useful. I mean, I'm using it for this module.

We will working with Python via Jupyter (pronounced like Jupiter) [11], which is an open-source web application that has improved the way data scientists, researchers, and educators work with code, data, and computational narratives. At its core, Jupyter provides an interactive environment where you can write and execute code in a variety of programming languages, most notably Python, but also R, Julia, and many others.

What sets Jupyter apart is its notebook interface, a document that combines live, executable code with rich text elements like paragraphs, equations, figures, and links. This makes Jupyter particularly valuable for creating and sharing documents that contain a mix of explanatory text, mathematical notation, and other media along with the code that performs computations.

Jupyter's versatility lies in its ability to support data visualization, machine learning, statistical modeling, and other data-intensive tasks directly within the notebook. Users can easily visualize data with graphs and charts, making it a powerful tool for exploratory data analysis. The interactive nature of Jupyter notebooks enables immediate feedback by executing code cells independently and viewing the results inline, which is ideal for iterative exploration and debugging.

Moreover, Jupyter has become a staple in educational settings—again, I am using it in my module. Its user-friendly interface is conducive to teaching concepts related to programming, allowing students to see and interact with code in a more engaging way. Educators can create notebooks that serve as interactive textbooks or assignment templates, combining instructional content with code exercises.

In the realm of collaboration and sharing, Jupyter notebooks can be easily converted to a range of formats like HTML, PDF, and slideshows, facilitating the dissemination of research findings or educational material. Platforms like GitHub and JupyterHub further enhance collaboration, allowing multiple users to work on shared notebooks.

For the remainder of the module, we will be using Jupyter notebooks.

2.1 How to run Jupyter

Thankfully there are a number of ways to run Jupyter on your machine. I will describe a few ways and give advantages and disadvantages for all the methods. If you do not know what to do, and do not want to install anything on your machine, you can work in browsers; see Sections 2.1.3 and 2.1.4.

2.1.1 Manual

As always, one can install Jupyter on the machine—to do this, one needs Python; see the Jupyter website [11] for more details. One needs to use `pip` for this method.

The major advantages are that you have full control of what you install; you do not need a special account; you do not need to use a browser or an internet connection.

The major disadvantages are that you have to install everything yourself using the terminal and the Python package installer `pip`. [This is easy if you are already comfortable with the terminal.]

2.1.2 Anaconda

There is a (rather bulky) manager called Anaconda [2] that will install Python and Jupyter Notebooks, among other things as well, on your machine. This is a rather popular option, and I recommend this method if you want it on your local machine, provided you do not feel comfortable with the terminal.

The major advantages are that you have all the components on your own machine, so you do not need to work in a browser nor do you need an internet connection. Everything is installed through the Anaconda manager.

The major disadvantages are that you have to install several extra programs (over 5 GB).

2.1.3 Colab

Google has developed the Colaboratory [5], which is a browser-based Jupyter notebook. In order to execute code in a notebook, one needs a Google account. Notebooks get saved into the account's Google Drive.

The major advantages are that Google takes care of everything for you. Everything runs smoothly and easily.

The major disadvantage is that you have to have or link your Google account. I am not sure what information they are collecting

2.1.4 Cocalc

The makers of SageMath have built a website called Cocalc [13] where one can create, edit, and execute Jupyter notebooks in the browser. Accounts are free, but without a license, you can only execute code on shared resources among, presumably, everyone else using the software in the world. Thus, it is very slow. [I am looking to see if the university has a cocalc server, and if not, whether we could get one.]

The major advantages are that SageMath takes care of everything for you, and you do not need to install anything.

The major disadvantages are that you have to have an account; it is free, but I do not know what data is collected of you. Without a paid license, executing code is very slow.

2.1.5 Binder

The makers of Jupyter notebook have developed Binder [10], which is a way of running Jupyter notebooks in the browser. However, it seems this is only for running notebooks and not creating an editing notebooks. Binder is suitable to follow along in the lecture, but when you need to write your own notebooks, you will need to use something else.

The major advantages are that you do not need to install anything, and you do not need an account for this.

The major disadvantage is that this is for reading and executing only. You cannot create and edit notebooks.

I hope to see you in the Jupyter notebooks.

References

[1] American Mathematical Society. amsmath – AMS mathematical facilities for latex, 2024. https://ctan.org/pkg/amsmath.

[2] Anaconda Inc. Anaconda, 2024. anaconda.com.

[3] David Carlisle. graphicx – Enhanced support for graphics, 2024. https://ctan.org/pkg/graphicx.

[4] Susan D'Agostino. The computer scientist who can't stop telling stories, 2020. Retrieved on 03-Jan-2024.

[5] Google. Welcome to Colaboratory, 2024. colab.research.google.com.

[6] Khan Academy. KaTeX, 2024. katex.org.

[7] Donald Knuth. Preliminary preliminary description of TEX. TEXDR.AFT, 1977.

[8] Leslie Lamport. *LaTeX: A Document Preparation System*. Addison-Wesley Longman Publishing Co., Inc., USA, 1989.

[9] MathJax Consortium. MathJax, 2024. mathjax.org.

[10] Project Jupyter. Binder, 2024. mybinder.org.

[11] Project Jupyter. Jupyter, 2024. jupyter.org.

[12] Python Software Foundation. Python, 2024. python.org.

[13] SageMath Inc. Cocalc, 2024. cocalc.com.