CMOR 420/520
Computational Science

Jesse Chan, 8/26/24

Motivation: why take this course?

- Scientific computing and scientific software are different from general-purpose programming.
- Performance and design are important, especially for large-scale and long-term projects.
- Scientific computing tools are as important (sometimes more) than the programming languages you'll use
- Foundations for high performance and parallel computing.

Assumptions / prerequisites

- You should be familiar with linear algebra
- You don’t need to have prior experience with C, C++, or Julia.
- Some programming familiarity (variables, loops, functions) would be helpful.

Administrative stuff:

- See Canvas page for notes and assignments (no website). Lectures will also be hybrid/recorded on Canvas.
- Grade based on assignments/projects only.
- Late policy: automatic 10% deduction each day.

Computing

- You will need access to a programmable computer (e.g., not an iPad or Chromebook). *Please let me know if you do not have access to one.*
- You are encouraged to bring a computer to class to experiment alongside the lectures.

What do we need to download/buy?

- Nothing, lecture materials, lecture recordings, and codes will be posted to Canvas.
- Demo codes will be available online (Github or Canvas).
- Information can also be found in online documentation or searching (forums, StackOverflow, etc).

Grading

- It is ***time-consuming*** to grade coding assignments. You will lose points if it is not easy for us to grade your code!
- You are responsible for ensuring that the graders and I can easily compile, run, and understand your code.
- Clean up and organize your code before turning it in!
    - Make your variable names clear and descriptive.
    - Add comments explaining complex or un-intuitive chunks of code.
    - Remove unused or unnecessary code, separate concerns, make it modular.

Generative AI policies

- Generative AI tools (ChatGPT, Gemini, Copilot, etc) are generally better at programming than many other scientific tasks (math, logic).
- You may use generative AI tools for assignments, but please note in your writeup which tool you used and provide the prompt.
  - You are still responsible for making sure that the code generated compiles and runs properly, is readable, and correct.

What are we supposed to learn?

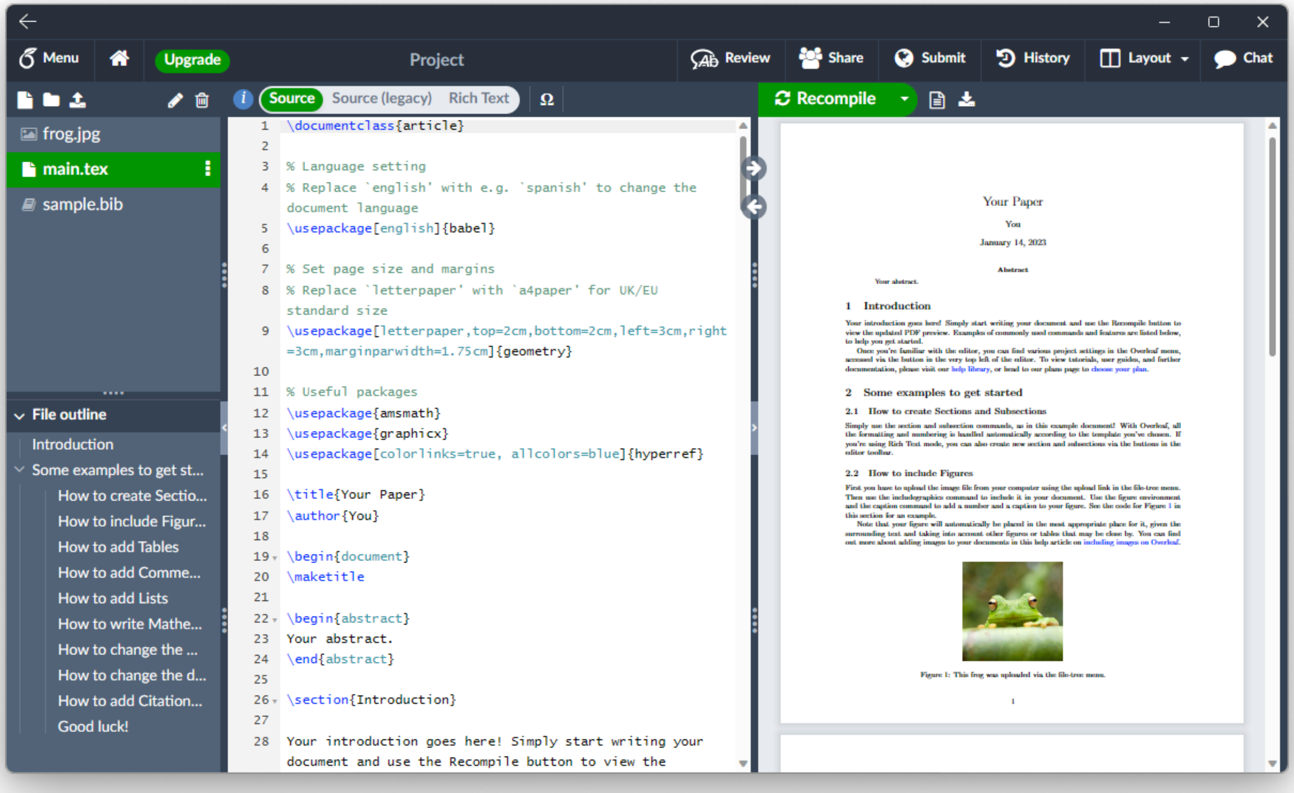
- Tools:
  - LaTeX for typesetting papers
  - The Linux (*nix) operating systems
  - Git version control
- Scientific computing in:
  - C programming language
  - C++ programming language

What is LaTeX?

- LaTeX is a mathematical typesetting software used to write reports, books, and papers.
- LaTeX commands are becoming more common (Canvas, Markdown, most times you enter “math mode”)
- Can run LaTeX from Linux command line, or…

LaTeX

- Overleaf is fine
- However, all LaTeX files should be submitted with assignments
- Make sure your writeups compile on Linux too! Overleaf sometimes compiles when Linux doesn't.



```latex
\documentclass{article}

% Language setting
% Replace `english' with e.g. `spanish' to change the document language
\usepackage[english]{babel}

% Set page size and margins
% Replace `letterpaper' with `a4paper' for UK/EU standard size
\usepackage[letterpaper,top=2cm,bottom=2cm,left=3cm,right=3cm,marginparwidth=1.75cm]{geometry}

% Useful packages
\usepackage{amsmath}
\usepackage{graphicx}
\usepackage[colorlinks=true, allcolors=blue]{hyperref}

\title{Your Paper}
\author{You}

\begin{document}
\maketitle

\begin{abstract}
Your abstract.
\end{abstract}

\section{Introduction}

Your introduction goes here! Simply start writing your document and use the Recompile button to view the
```

Your Paper

You

January 14, 2023

Abstract

Your abstract.

1 Introduction

Your introduction goes here! Simply start writing your document and use the Recompile button to view the updated PDF preview. Examples of commonly used commands and features are listed below, to help you get started.

Once you're familiar with the editor, you can find various project settings in the Overleaf menu, accessed via the button in the very top left of the editor. To view tutorials, user guides, and further documentation, please visit our help library, or head to our plans page to choose your plan.

2 Some examples to get started

2.1 How to create Sections and Subsections

Simply use the section and subsection commands, as in this example document! With Overleaf, all the formatting and numbering is handled automatically according to the template you've chosen. If you're using Rich Text mode, you can also create new section and subsections via the buttons in the editor toolbar.

2.2 How to include Figures

First you have to upload the image file from your computer using the upload link in the file-tree menu. Then use the includegraphics command to include it in your document. Use the figure environment and the caption command to add a number and a caption to your figure. See the code for Figure 1 in this section for an example.

Note that your figure will automatically be placed in the most appropriate place for it, given the surrounding text and taking into account other figures or tables that may be close by. You can find out more about adding images to your documents in this help article on including images on Overleaf.

Figure 1: This frog was uploaded via the file-tree menu.

1

Linux terminal

- Command line interface (CLI) operating system
- Navigate and organize directories within a file system
- Install and update packages
- Compile code and run programs
- Shell scripting: automate and chain together commands

Some notes on the Linux terminal

- You will need access to a Linux terminal. For example, you can:
  - download the VirtualBox emulator and install Ubuntu.
  - use Windows Subsystem for Linux (WSL) or native Mac terminal
  - use an ssh client (e.g., PuTTY) and ssh.clear.rice.edu
- You can **not** use cmd.exe or Powershell for this course.
- Online Linux terminals don’t always provide “root access”.
  - Emulators like Cygwin may work; they’ll need to be able to install and run Git, LaTeX compilers, and C/C++ compilers.

Git version control

- Version control: absolutely crucial for managing large projects
- Git allows you to:
  - Back-up codebases and track code changes
  - Switch between different versions (“branches”) of a codebase to test new code.
  - Collaborate with other programmers through hosting services like Github or Bitbucket

Programming languages: C, C++

- Why these languages specifically?
- Python is the most popular scientific programming language in the world; why are we skipping it?

Why this combination of material?

- **Efficiency.** Optimize scientific code without changing the behavior.
    - Most Python libraries use C/C++ for performance-critical code.
    - Introduce core concepts: pointers, memory management, etc.
- Tools for managing large software projects: the Linux terminal, Git/Github for version control and collaboration, Makefiles for compilation workflows.