An introduction to Rust

Foreword

This session is somewhat a bias introduction from me because I love `rust`. Here is the backstory: I was first introduced to `rust` in 2022, mostly because of the controversy around it, and I did part of the advent-of-code (AOC) in rust. At the time, I could not see much point of learning and using `rust`, mostly because I do not know where to use `rust`. I went on a year without using and following `rust`, then I did most of AOC 2023 in `julia`. On December 24, 2023, I came across an article about web assembly (`wasm`), which talks about how `wasm` can be used to distribute code on the web which run on client-side with almost native performance, and how one of the main use case for `rust` is to compile to `wasm`. I unfortunately and willingly sacrifice my 25th day of AOC to learn about `rust` and `wasm`, and it was awesome. For the next couple of months I was exploring `rust` a lot, and it was such a fun experience.

In case you do not realize how interesting it is, let me break it down for you. If you have supervised students or collaborate with others on a code project before, especially with someone using a MacOS, the most difficult part of the job could be getting them to install the dependencies correctly. Nowaday most python package can be install with `pip install [package]`, but sometimes people need to build a package dependencies from source, and that is a highway to nightmare if the code is not supported by a huge community. Being able to write in a programming language like `rust`, compile it to `wasm` and run it directly on a browser without any installation is a complete game changer. No install needed, just open the browser and boom, the code runs. Say in a dire situation like needing to factor some prime numbers in a party, you can just pull out your phone, open the browser and check whether a number is a prime. Isn't that cool?

Beside, I really had a great time coding rust in general. It takes some time to get used to the language, but once you are more familiar with rust, it is a great language to go hard core with. In this lab, we are going to go through the following topics:

Key Concepts

Rust is a compiled language

Rust is strongly and statically typed

Borrow checker for memory management

Basic Syntax

Variables

Functions

Control flow

Borrow checker

We will write a simple MCMC algorithm this time

Here we go again to write yet another insertion sort.

An introduction to Rust

Step 0: Install rust

Step 1: Clone the class repository

Step 2: Implement the Metroplis-Hastings algorithm

Step 3: Test the algorithm

Putting the MCMC algorithm on the Web

Best practices

Development tips

Noteworthy libraries