Introduction to Julia

Foreword

`julia` is a language which I have a love-hate relationship with. If the first programming language you learned is `python`, I think `julia` offers a fresh take on what you can do with computers while having the interactiveness of `python`. It has a lot of modern features built into the language, such as its just-in-time (JIT) compilation, multiple dispatch, and metaprogramming capabilities. It also comes with its own package manager, which is quite nice to use. This makes `julia` a great "advance" language for data scientists to learn after `python`. However, the `julia` ecosystem is not nearly as mature as `python`, as a lot of its packages are maintained by small communities, and some time they lead to down some dead ends.

Nonetheless, `julia` is a fun language to play with. `julia` often offers more flexibility and performance than `python`, and its ecosystem has a lot of interesting research codes which are often not found in other ecosystem. So in this lab, we are going to go through

Outline

Introduction to Julia

Key Concepts

Julia has a JIT compiler

Julia has multiple dispatch

For the people who learn `python` as their first programming language, and perhaps engaged in some projects related to `python`, you may find `julia` quite odd in the sense that **it does not have classes**.

Julia has a package manager

Basic Syntax

Variables

Functions

Control Flow

Writing an insertion sort algorithm

Packaging code

Building documentation

Documenter.jl

Writing tests

Best practices

Type stability

Write functions

Development tips

Noteworthy libraries