Julia Installation

Gregory Farage, Śaunak Sen

```
gfarage@uthsc.edu / sen@uthsc.edu
Division of Biostatistics
Department of Preventive Medicine
University of Tennessee Health Science Center
Memphis, TN
```

Julia is a modern programming language gaining popularity among data scientists, engineers, and researchers due to its speed, flexibility, and ease of use. Julia offers a Read-Eval-Print Loop (REPL), which provides an interactive environment for running Julia code. With the REPL, users can quickly test their code snippets, inspect variables, and quickly iterate on their ideas. The REPL also includes a package manager for installing and managing third-party packages. The REPL is the most basic and direct way to interact with Julia.

Most users will also want an Integrated Development Environment IIDE) to manage Julia code and projects. Several several options are available, including Intellij, VS Code with the Julia extension, and Jupyter with the Julia kernel. Each IDE has its strengths, and they all provide valuable features such as code highlighting, autocomplete, and debugging tools to help users write and debug Julia code more efficiently.

Of course, you can also work with just a plain text editor such as Emacs, Vim, SublimeText or NotePad. Most offer Julia syntax highlighting; for example the ESS (Emacs Speaks Statistics) package offers Julia syntax highlighting for Emacs.

Before diving into the world of Julia, you would need to set up your environment with the necessary development tools and packages and libraries.

In this introductory workshop to the Julia language, we will present our examples in the **Jupyter** IDE, and we recommend you to also follow us by creating you own Jupyter notebook and playing with it, to take the advantage of fast experimenting in Jupyter. Visual Studio Code also offers native support for working with Jupyter Notebooks.

Let's now look at how we can set up the Julia environment.

Installation instructions

Step 1: Download the current stable release Julia (**v1.8.5**) for your **specific operating system (OS)** from the Julia homepage here.

Step 2: Follow the platform-specific instructions to install Julia on your OS:

- Windows
- macOS
- Linux

It is highly recommended to opt-in to add Julia to PATH during the installation process, which will allow you to conveniently start a Julia session by simply typing `julia` into the command line.

Step 3: Check if Julia has been successfully installed in your OS, start the Julia session from you console.

To exit from Julia, either type `exit()` in the terminal or press `Ctrl + D`.

Julia in Jupyter IDE (recommended):

To use the Jupyter IDE with the Julia programming language, we need to establish the connection between Julia and Jupyter - to create the language-specific **kernel** to interact with the Jupyter interfaces.

The Julia package for creating a kernel in Jupyter is called `IJulia.jl`. (more details about the package `IJulia.jl` here). If Jupyter IDE is not already installed, `IJulia.jl` will give the option to to install Jupyter in a private path to Julia. Alternatively, we can install Jupyter independenlty form Julia by following the Jupyter official documentation or the Jupyter lab manual.

Follow the steps below to start a Jupyter notebook with the Julia kernel.

Step 1: Install the package `IJulia.jl`

Inside a Julia session (you can check if a `julia>` prompt appears in the terminal window), install the `IJulia.jl` package either by typing

```
using Pkg
Pkg.add("IJulia")
```

or by first entering the package environment (by typing `]`) and then typing <-- REMOVE

```
(@v1.8) pkg> add IJulia
```

Step 2: Create the Julia kernel in Jupyter

- **For the first time Jupyter users**, the simplest way to establish the kernel is through the package, which will have the (minimal) Jupyter interface for Julia installed and create the kernel for you. Simply typing the following to start the Jupyter

```
using IJulia
notebook() # for jupyter notebook
jupyterlab() # for jupyter lab
```

The first time running the command will prompt you whether Jupyter should be installed. Press enter to tell IJulia to install the minimal Python/Jupyter distribution (via Miniconda) that does not interfere with your other Jupyter environments (in **PATH**) but is private to Julia.

(More details about running IJulia for creating Julia kernels)

- **If you already have jupyter** in your computer and have used it before, you still need to create the Julia kernel in Jupyter using `IJulia.jl`, by

```
using Pkg
Pkg.build("IJulia")
```

This will tell your Jupyter where to find Julia in your computer.

Note: If you install a new Julia binary (or do anything to change the location of Julia binary in your computer), you need to run this again to update the IJulia installation.

After these two steps, Jupyter notebook/lab can be launched from the terminal by entering `jupyter notebook` or `jupyter lab`.

Julia in Visual Studio Code (optional):

For **Visual Studio Code** users, the Julia-VS-Code IDE is free to download and is highly maintained for Julia users. The environment also comes with preloaded libraries. Julia-VS-

Code, in some way, is analog to RStudio with R. But we recommend to stick with the notebook if possible for this workshop.

(Detailed instructions for setting up Julia environment in VS Code)

Step 1: Install VS-Code IDE:

Download and install the latest stable version of VSCode compatible with your system, from the VSCode homepage.

Step 2: Install Julia extension in VS-Code:

- Open the VS-Code application on your computer.
- Select `View` -> `Extensions` from the screen top or simply press `Shift + Command + X` for macOS to open the **Extensions View**.
- Search for `julia`. Click on green `Install` botton to download and install the extension.
- For more information about using Jupyter Notebooks in Visual Studio Code, please refer to the link here.

Note: it is recommended to re-open VS-Code after adding the Julia extension.
