Julia in the Jupyter Notebook environment (required before the workshop)

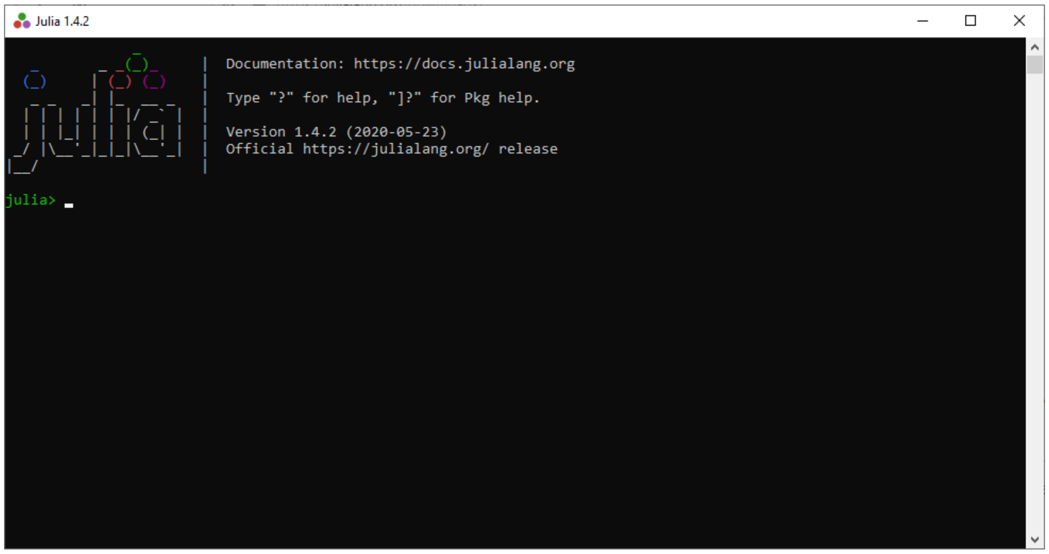
Julia is a modern, open source programming language which has been designed from the ground up for modern scientific computing. It claims to take the best features and advantages from other programming languages such as Python, Matlab, R and C, and incorporate it into one computer language. You can read more about Julia here: https://julialang.org/

To make coding easier, it is common to use an **integrated developer environment** (**IDE**), which has tools make writing and debugging code easier. For example, Python uses an IDE called Spyder, Matlab has its own integrated IDE, and R uses RStudio. A nice IDE for Julia is **Atom** (a fancy text editor) with the **Juno** package (an extension to make it into a Julia IDE), but this will be too complicated to install for a one-off workshop. Instead, we will use **Jupyter Notebook**, which is a convenient web-based solution used to share executable code that can be wrapped in documents, so it is ideal for explaining and learning how to use and write code. You can read more about Jupyter Notebook here: https://jupyter.org/ (but do not install anything from that website).

For this Julia workshop, you will need to execute Julia code and you will perform this by installing Julia on your own computer. You will then import **IJulia**, a package which contains the Jupyter Notebook for Julia. You must have Julia and the relevant packages running before class. Please get in touch if cannot get these instructions to work (email j.potas@unsw.edu.au with the subject heading "help me install Julia").

1. Install Julia and IJulia for the first time

- Download the right version of Julia (current stable release) for your operating system (do not install JuliaPro): https://julialang.org/downloads/
- Once downloaded and installed (do not change the default install location), run Julia (find it from your start menu [Windows] or applications folder [Mac]), it should produce a Julia terminal window (called a REPL) that looks like this:



```
               _
   _       _ _(_)_     |  Documentation: https://docs.julialang.org
  (_)     | (_) (_)    |
   _ _   _| |_  __ _   |  Type "?" for help, "]?" for Pkg help.
  | | | | | | |/ _` |  |
  | | |_| | | | (_| |  |  Version 1.4.2 (2020-05-23)
 _/ |\__'_|_|_|\__'_|  |  Official https://julialang.org/ release
|__/                   |

julia> 
```

- From the Julia REPL, you will need to install the IJulia and Conda packages. For this:
    - Type a closing square bracket ( ]) to get into the package manager *Pkg* mode as follows:

```
julia> ]
```

    - Your julia> prompt should change to the following: (@v1.4) pkg>. Then type add IJulia followed by **Enter**. Once that has completed, you can repeat the process for add Conda. Note: you will need to wait for each installation, which can take a while:

```
(@v1.4) pkg> add IJulia

(@v1.4) pkg> add Conda
```

- Once finished, hit **backspace** to exit Pkg mode, to bring the julia> prompt back
- Next, we will open a Jupyter Notebook session. First, you must import the IJulia package. Julia's import command is called **using**. Type the following:

```
julia> using IJulia
```

...wait until the julia> prompt returns (the first time may take a while).

- From the imported IJulia package we can run a *notebook()* session, then follow the prompt to install Jupyter via Conda as follows:

```
julia> notebook()

install Jupyter via Conda, y/n? [y]: y
```

Once that is done (i.e. after a Juypter Notebook session has opened in your browser), you should have properly configured Julia and Jupyter. At this point, quit Julia (just close the REPL window) and initiate a new session as described in the next section.

2. Start Julia and launch a Juypter session

- Run Julia (find in Start or Application menus where other programs are kept)
- Import the IJulia package:

```
julia> using IJulia
```

- Run notebook(). However, this time you will need to specify the directory where you put your Julia workshop files:
    - Download the **JuliaWorkshop2020.ipynb** and **neuro_data.jld2** files from Moodle and place them into a folder where you will conduct the workshop
    - Copy the directory location of this folder from the folder's address bar and place it inside the brackets of the Notebook() function (as shown below). This points Jupyter Notebook to the directory location containing the Julia workshop files. Note that for

Windows and Mac, the way a directory is written might differ; for PC it should look something like this:

```
julia> notebook(dir="C:\\ Documents\\OneDrive\\Julia workshop 2020")
```

PC users, note the use of two “\\” not “\” between directory levels.
Mac users try “/” instead.

That should open your default browser and launch a Jupyter Notebook session where you can see inside the directory you specified.

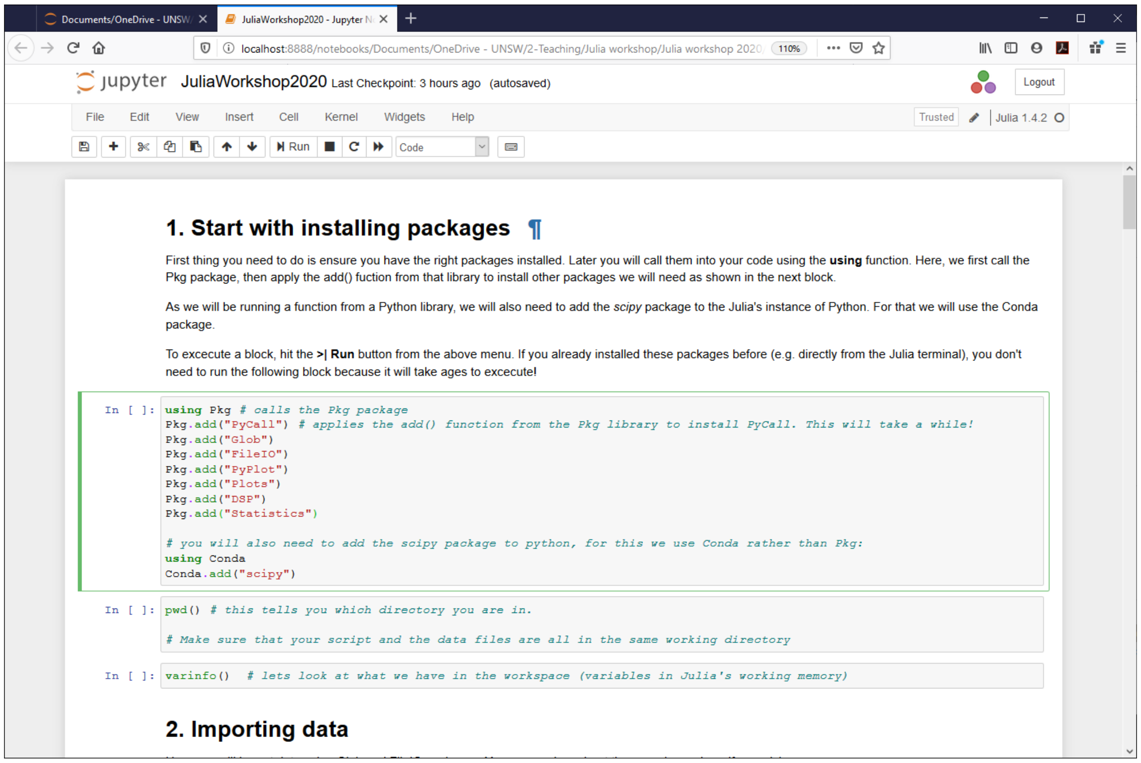
3. Install the required packages for the workshop

Before you come to class, you will need to install several packages. There are several ways to do this:

a) Execute the first block of script within the Julia workshop notebook

The first method requires you to open the Julia workshop notebook and execute the first code block (a script):

- Open a Julia session of Jupyter Notebook at the Julia workshop directory (as instructed above)
- click on the file: **JuliaWorkshop2020.ipynb**. This will open a Julia session in a new browser tab that will contain the 9 steps of the workshop which looks as follows:



Documents/OneDrive - UNSW
JuliaWorkshop2020 - Jupyter N
localhost:8888/notebooks/Documents/OneDrive - UNSW/2-Teaching/Julia workshop/Julia workshop 2020
110%
jupyter JuliaWorkshop2020 Last Checkpoint: 3 hours ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Julia 1.4.2
Run
Code

1. Start with installing packages ¶

First thing you need to do is ensure you have the right packages installed. Later you will call them into your code using the **using** function. Here, we first call the Pkg package, then apply the add() fuction from that library to install other packages we will need as shown in the next block.

As we will be running a function from a Python library, we will also need to add the *scipy* package to the Julia's instance of Python. For that we will use the Conda package.

To excecute a block, hit the **>| Run** button from the above menu. If you already installed these packages before (e.g. directly from the Julia terminal), you don't need to run the following block because it will take ages to excecute!

```
In [ ]: using Pkg # calls the Pkg package
        Pkg.add("PyCall") # applies the add() function from the Pkg library to install PyCall. This will take a while!
        Pkg.add("Glob")
        Pkg.add("FileIO")
        Pkg.add("PyPlot")
        Pkg.add("Plots")
        Pkg.add("DSP")
        Pkg.add("Statistics")

        # you will also need to add the scipy package to python, for this we use Conda rather than Pkg:
        using Conda
        Conda.add("scipy")

In [ ]: pwd() # this tells you which directory you are in.

        # Make sure that your script and the data files are all in the same working directory

In [ ]: varinfo()  # lets look at what we have in the workspace (variables in Julia's working memory)
```

2. Importing data

- Execute the first code block (outlined in a green box in the image above). To do this, select the first block and hit the **>| Run** button (found directly below the **Cell** menu heading).
- How do you know it has finished running? Wait for the * to disappear in the input (code) box and turn into a number as follows:

In [*] → In [1]

- When it has finished, you can quit Jupyter:
    - Go to the previous tab which shows the directory tree and click the "Quit" button on the upper right corner.

b) Install packages directly from a Julia terminal

The second method does the exact same thing as a) above directly from the Julia terminal (no need to open the workshop notebook). Note that you can also directly type each line of code from the workshop notebook directly in a Julia terminal, but that takes too long and you have to wait for each line to be finished before you can do the next one.

The easier (three line) option is to use the package manager by typing "]", then "add", then list all the desired packages you wish to add as follows (separated with a space only):

```
(@v1.4) pkg> add PyCall Glob FileIO PyPlot Plots DSP Statistics JLD2
```

Once that is done (i.e. after the (@v1.4) pkg> prompt has returned), you need to add the extra Python scipy library:

- Hit "backspace" button to exit Pkg mode
- Type: using Conda, then enter
- Type: Conda.add("scipy"), then enter (no space between add and the bracket), as follows:

```
julia> using Conda

julia> Conda.add("scipy")
```

Once that is finished (wait for the julia> prompt to return), you are ready for the workshop!! Feel free to go through the workshop by yourself if you are curious!