Interacting with Julia

Jesse Perla, Thomas J. Sargent and John Stachurski

March 5, 2020

1 Contents

• Overview 2
• Using Jupyter 3
• Using the REPL 4
• (Optional) Adding Jupyter to the Path 5

2 Overview

In this lecture we’ll start examining different features of the Julia and Jupyter environments.

3 Using Jupyter

3.1 Getting Started

Recall that the easiest way to get started with these notebooks is to follow the cloning instructions earlier.

To summarize, if on a desktop you should clone the notebooks repository https://github.com/quantecon/quantecon-notebooks-julia, then in a Julia REPL type

using IJulia; jupyterlab()

Hint: Julia will remember the last commands in the REPL, so you can use up-arrow to restart JupyterLab.

Alternatively, if you are using an online Jupyter, then you can directly open a new notebook.

Finally, if you installed Jupyter separately or have added added Jupyter to the Path then cd to the folder location in a terminal, and run

jupyter lab
Regardless, your web browser should open to a page that looks something like this

![Dashboard Image]

The page you are looking at is called the “dashboard”.

If you click on “Julia 1.x.x” you should have the option to start a Julia notebook.

Here’s what your Julia notebook should look like

![Notebook Image]

The notebook displays an active cell, into which you can type Julia commands.

### 3.2 Notebook Basics

Notice that in the previous figure the cell is surrounded by a blue border.

This means that the cell is selected, and double-clicking will place it in edit mode.

As a result, you can type in Julia code and it will appear in the cell.

When you’re ready to execute these commands, hit **Shift-Enter**
3.2.1 Modal Editing

The next thing to understand about the Jupyter notebook is that it uses a modal editing system.

This means that the effect of typing at the keyboard depends on which mode you are in. The two modes are

1. Edit mode
   - Indicated by a green border around one cell, as in the pictures above.
   - Whatever you type appears as is in that cell.

1. Command mode
   - The green border is replaced by a blue border.
   - Key strokes are interpreted as commands — for example, typing b adds a new cell below the current one.

(To learn about other commands available in command mode, go to “Keyboard Shortcuts” in the “Help” menu)

3.2.2 Switching modes

- To switch to command mode from edit mode, hit the Esc key.
- To switch to edit mode from command mode, hit Enter or click in a cell.

The modal behavior of the Jupyter notebook is a little tricky at first but very efficient when you get used to it.

3.2.3 Working with Files

To run an existing Julia file using the notebook you can copy and paste the contents into a cell in the notebook.

If it’s a long file, however, you have the alternative of

1. Saving the file in your present working directory.
2. Executing include("filename") in a cell.

The present working directory can be found by executing the command pwd().
3.2.4 Plots

Note that if you’re using a JupyterHub setup, you will need to first run

```julia
In [1]: using InstantiateFromURL
   github_project("QuantEcon/quantecon-notebooks-julia", version = "0.5.0")
   # instantiate = true) # uncomment to force package installation
```

in a new cell (i.e., Shift + Enter).

This might take 15-20 minutes depending on your setup, as it installs a large set of packages for our use.

Run the following cell

```julia
In [2]: using Plots
   gr(fmt=:png);
   plot(sin, -2π, 2π, label="sin(x)")
```

You’ll see something like this (although the style of plot depends on your installation)

![Plot Example]

**Note:** The “time-to-first-plot” in Julia takes a while, since it needs to compile many functions - but is almost instantaneous the second time you run the cell.

3.3 Working with the Notebook

Let’s go over some more Jupyter notebook features — enough so that we can press ahead with programming.
3.3.1 Tab Completion

Tab completion in Jupyter makes it easy to find Julia commands and functions available. For example if you type `rep` and hit the tab key you’ll get a list of all commands that start with `rep`.

```
In [1]: rep
search: repeat
Out[1]: repeat(A::AbstractArray, counts::Integer...)  
Construct an array by repeating array A a given number of times in each dimension, specified by counts.
```

3.3.2 Getting Help

To get help on the Julia function such as `repeat`, enter `? repeat`. Documentation should now appear in the browser.

```
In [1]: ? repeat
```

3.3.3 Other Content

In addition to executing code, the Jupyter notebook allows you to embed text, equations, figures and even videos in the page.

For example, here we enter a mixture of plain text and LaTeX instead of code:

```
Enter found that
$$ \exp(i \pi) = -1 $$
```

Next we Esc to enter command mode and then type m to indicate that we are writing Markdown, a mark-up language similar to (but simpler than) LaTeX. (You can also use your mouse to select Markdown from the Code drop-down box just below the list of menu items)
Now we *Shift + Enter* to produce this

\[ \exp(i\pi) = -1 \]

### 3.3.4 Inserting unicode (e.g. Greek letters)

Julia supports the use of **unicode characters** such as $\alpha$ and $\beta$ in your code.

Unicode characters can be typed quickly in Jupyter using the `tab` key.

Try creating a new code cell and typing `\alpha`, then hitting the `tab` key on your keyboard.

### 3.3.5 Shell Commands

You can execute shell commands (system commands) in Jupyter by prepending a semicolon.

For example, `; ls` will execute the UNIX style shell command `ls`, which — at least for UNIX style operating systems — lists the contents of the current working directory.

These shell commands are handled by your default system shell and hence are platform specific.

### 3.3.6 Package Operations

You can execute package operations in the notebook by prepending a `]`.

For example, `] st` will give the status of installed packages in the current environment.

**Note:** Cells where you use `;` and `]` must not have any other instructions in them (i.e., they should be one-liners).

### 3.4 Sharing Notebooks

Notebook files are just text files structured in **JSON** and typically end with `.ipynb`.

A notebook can easily be saved and shared between users — you just need to pass around the `ipynb` file.

To open an existing `ipynb` file, import it from the dashboard (the first browser page that opens when you start Jupyter notebook) and run the cells or edit as discussed above.

The Jupyter organization has a site for sharing notebooks called nbviewer which provides a static HTML representations of notebooks.

QuantEcon also hosts the QuantEcon Notes website, where you can upload and share your notebooks with other economists and the QuantEcon community.
4 Using the REPL

As we saw in the desktop installation, the REPL is a Julia specific terminal.

It becomes increasingly important as you learn Julia, and you will find it to be a useful tool for interacting with Julia and installing packages.

As a reminder, to open the REPL on your desktop, either

1. Navigating to Julia through your menus or desktop icons (Windows, Mac), or
2. Opening a terminal and typing `julia` (Linux)

If you are using a JupyterHub installation, you can start the REPL in JupyterLab by choosing

1. Choose “New Launcher”
2. Choose a Julia Console

We examine the REPL and its different modes in more detail in the tools and editors lecture.

5 (Optional) Adding Jupyter to the Path

If you installed Jupyter using Julia, then you may find it convenient to add it to your system path in order to launch JupyterLab without running a Julia terminal.

The default location for the Jupyter binaries is relative to the .julia folder (e.g., "C:\Users\USERNAME\.julia\conda\3\Scripts" on Windows).

You can find the directory in a Julia REPL using by executing

```
] add Conda
using Conda
Conda.SCRIPTDIR
```

On Linux/OSX, you could add that path to your .bashrc.

On Windows, to add directly to the path, type `;` to enter shell mode and then execute

```
setx PATH "$(Conda.SCRIPTDIR);%PATH%"
```