

Julia/IJulia/JuMP Installation Guide

Purpose

This document aims to get you set up with




1. **Julia** - the programming language.
2. **IJulia** - a popular environment for writing Julia code in.
3. **JuMP** - a package for Julia that lets you express optimization models, *and* **Coin-OR Clp and Cbc** - open-source solvers for linear and mixed-integer linear programming.
Ipopt - open-source solver for nonlinear programming.
4. **Learning Julia**

Julia (<http://julialang.org/>) is a relatively new programming language that is aimed at “technical computing” - the kind of computing that most of you do every day. There are a lot of reasons “why Julia”, but rather than write them out again, check out <https://github.com/stevengj/julia-mit> for a good explanation. We’ll be using Julia through the **IJulia** notebook interface, which lets you mix code, documentation, mathematics, graphics, etc. This requires Python to be installed first, and is optional - you can use Julia directly through its [REPL](#) interface if you’d like.

JuMP (<https://github.com/JuliaOpt/JuMP.jl>) is a modeling language for optimization problems. It lets you translate the mathematical statement of an optimization problem into a form the computer can work with, but is still easy for humans to work with. It is developed by students at the MIT Operations Research Center, so we can provide support for you if you want to use it in your work. If you want to get involved in its future development, please get in touch!

1. Julia

You can download Julia from the official Julia downloads page: <http://julialang.org/downloads/>. You should download the “**current release**” (**v0.3.1**). (If you have an version of Julia already installed which is older than v0.3.0, please update it.) Accept the default setup options.

-  If you are on Windows you should probably choose the 64-bit version. If you have a very old Windows computer (>4 years old) there is a chance you will have a 32-bit version of Windows. To check, go to the “System Information” page (start typing “system information” in the Start menu and it should show up).
-  If you have an old Mac and haven’t upgraded to OS X 10.7 or higher, you won’t be able to run Julia. You can check this by clicking the Apple logo in the top-left of your screen and clicking “About This Mac”
-  Up-to-date versions of Julia are available for some distributions. Please read instructions on the download page. If you are on Ubuntu, **do not** install the version of Julia that is in the main repository - it is out-of-date.

2. IJulia

IJulia is a browser-based environment for writing and running Julia code. It lets you combine code, text, graphics, and equations all in one place. You may have heard of IPython, which is something similar for the Python programming language. In fact, both IJulia and IPython share the same infrastructure, so we need to install Python/IPython first).

Note: If you are on the MIT campus, **do not use the MIT Guest wifi network**. Please connect to “MIT” or “MIT Secure” - downloads can have issues on MIT Guest.

Windows

- Install Anaconda Python from <http://continuum.io/downloads>. This will also install IPython.
- The Anaconda installer window gives options *Add Anaconda to the System Path* and also *Register Anaconda as default Python version of the system*. Be sure to check these boxes.
- Open Julia by double-clicking the icon or from the Start menu.
- A window with a `julia>` prompt will appear.
- Type `Pkg.add("IJulia")` and wait for the installation to finish.

Apple

OSX actually comes with a version of Python already installed. However, it tends to be out-of-date so it is preferable to use a newer version. If you feel comfortable installing IPython with the in-built version of Python, or a version of Python installed some other way, there is no problem with doing so. These instructions will assume you want to use Anaconda Python:

- Install Anaconda Python from <http://continuum.io/downloads>. This will also install IPython.
- After it is installed, open Julia from your Applications.
- A window with a `julia>` prompt will appear.
- Type `Pkg.add("IJulia")` and wait for the installation to finish.

Linux

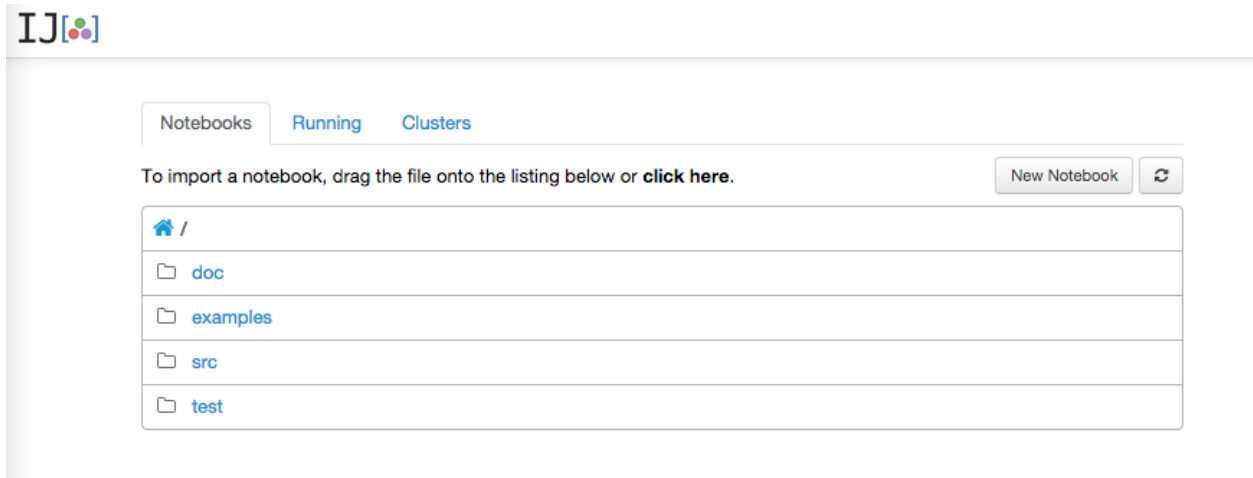
If you are using Linux, you probably like doing things yourself. Python is almost surely already installed. Installing IPython with `pip` is as easy as `pip install ipython`. After that, open Julia and run `Pkg.add("IJulia")` at the `julia>` prompt.

After installing IJulia

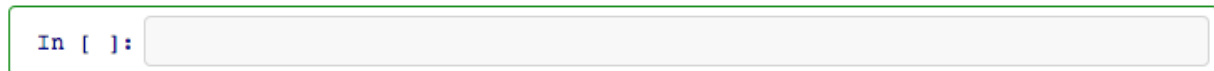
The easiest way to check if your installation was successful is to open Julia and run

```
using IJulia
IJulia.notebook()
```

A web browser window should open, with something that looks like:



(the list of files will be different). Click “New Notebook” and a new tab will open. In the “In []” box:



type `1+1` and press shift and enter/return at the same time. The result should appear like



If that doesn't work, please consult the “Troubleshooting” section of <https://github.com/stevengj/julia-mit> first, and if that doesn't work please contact the TA.

3. JuMP, Coin-OR Clp/Cbc, Ipopt

- Open an IJulia notebook like described above.
- In the first cell enter the following commands:

```
Pkg.add("JuMP")  
Pkg.add("Clp")  
Pkg.add("Ipopt")
```

Then press shift-enter to run the cell.
- This will install the JuMP package, the Clp package and solver, the Cbc package (which is a dependency of the Clp package), and Ipopt solver. It might take a while as it has to download the code and solver (on 🐧 Linux it will build them from source).
- Test it worked by writing in the next cell

```
using JuMP  
m = Model()
```

```
@defVar(m, x[1:2], Bin)
@setObjective(m, Max, 3x[1] + 2x[2])
@addConstraint(m, x[1] + x[2] <= 1)
solve(m)
print(getValue(x))
```

- You should see something like

```
In [7]: using JuMP
m = Model()
@defVar(m, x[1:2], Bin)
@setObjective(m, Max, 3x[1] + 2x[2])
@addConstraint(m, x[1] + x[2] <= 1)
solve(m)
print(getValue(x))

[1] = 1.0
[2] = 0.0
```

4. Now familiarize yourself with Julia

- We only have an hour of class time, so the focus will mostly be on modeling optimization problems with JuMP. We'll assume you are familiar with the basics of Julia to save time, and **we have no class time for getting your installation working.**
- The good news is that if you have used a language like MATLAB or Python before then you pretty much know the Julia basics already!
- There are lots of resources out there to learn Julia:
 - Julia video tutorial: <https://www.youtube.com/watch?v=vWkgEddb4-A>
 - Julia + IJulia cheatsheet: <http://math.mit.edu/~stevenj/Julia-cheatsheet.pdf>
 - Learn X in Y minutes, Julia edition: <http://learnxinyminutes.com/docs/julia/>
 - And of course, the detailed Julia manual: <http://docs.julialang.org/en/latest/>