Using on MAC

There are several ways I tested, all which support self-completion and have an integrated editor/runtime environment (as does MATLAB). The first one works on any architecture, (e.g Windows, or Linux as well). The following I tested recently:

1. Remotely, using juliabox. (tested on 01/04/2015 – Now for a FEE!!)
   - Needs access to internet at : juliabox.org. (login with google account, can use
   - Once one has it, it can be run immediately. (it is what I used in class on 01/04/2016).

2. Local Executable, Through JUNO (tested in 2016)
   - can do it stand-alone with JUNO from http://junolab.org/docs/install.html
   - I have installed it through homebrew to maintain consistency with python, but the standalone may be fine.
   - once you downloaded, it is better if you run it from terminal to get the right environment variables. For that, you need the address of the executable.
     - If you place JUNO in Applications the executable should be at /Applications/Juno.app/Contents/MacOS/atom
     - I advise entering a line in ~/.profile "export JUNO=/Applications/Juno.app/Contents/MacOS/atom"
     - Open a new terminal. Enter at the prompt "$JUNO".
   - to add new packages in Julia, (and we will need a few of those), you need to locate the julia executable that came with JUNO.
     - If JUNO is in applications, it should be at:/Applications/Juno.app/Contents/Resources/app/julia/bin/julia
     - I advise entering a line in ~/.profile "export JULIA_JUNO=/Applications/Juno.app/Contents/Resources/app/julia/bin/julia"
     - then you can open the julia of JUNO from a new terminal prompt with "$JULIA_JUNO".
   - Once Julia starts. You can use the Julia package manager entering commands at the julia prompt. (I assume you have a good active internet connection)
     - Pkg.status() should show the installed packages.
     - Pkg.add("JuMP") should install the Julia JuMP package (which you will need), for example. We will need several others
     - Pkg.update() updates the existing packages to new version. It is good to run after each add.
   - As of 01/2016 I use:
     - MATLAB – for interaction with Matlab (e.g for graphing)
     - Gadfly – for minimal plotting.
     - JuMP – for modeling optimization problems.
     - PyPlot – for interaction with Python Plotter
   - Note 01/05/2016: Unfortunately, it crashes due to an error in Require.jl – at least on my laptop. See: http://discuss.junolab.org/t/couldnt-connect-to-julia-error/459/4 When this will be fixed, it would be a valid way to do it.

SOME FIXES

- Sometimes the added packages enter a weird state. Not the most elegant, but an easy way I found to deal with it is to remove the version folder under your $HOME/.julia/v0.x directory. Then you have to reinstall the packages. If that does not work, you can try to erase the whole $HOME/.julia
- As of 01/05/2016, it appears that JUNO does not work properly.
  - There is something off with "require". In particular if either the editor or you run Pkg.update() it will crash everything ...

3. Using Sublime Text + IJulia.(tested on 01/05/2015)
   - Get a Julia Executable from here: http://julialang.org/downloads/
   - Modify the PATH environment variable to have the folder where you installed (this is in either .profile or .bashrc. I use bash and for me the line is "export PATH=$PATH:/path/to/Julia" last among all PATH declarations (for the path, right-click on the package with julia to Show Contents, and find the executable). You can then run julia by just invoking "julia" in a terminal or X11 shell. Alternatively, you can just provide the absolute path.
   - Download Sublime Text 3 from here: http://www.sublimetext.com/3
   - Set up Sublime Text 3 to work with Julia as described here: https://github.com/quinnj/Sublime-IJulia
   - As of /01/05/2016, there is a bug in Sublime Text 3 which does not allow it to work with the last version of IJulia.
   - I hacked it as follows (after doing ALL previous steps):
     - Start Julia.
     - Now downgrade IJulia to use ipython2 as follows:
       - julia- Pkg.checkout("IJulia","ipython2")
       - julia- Pkg.build("IJulia")
     - When you restart Sublime Text, and do "Command-Shift-P" followed by "Open IJulia Console".
     - After a few seconds, the Julia banner should appear. You then have for yourself an integrated Julia Environment.

• Not even sure what I did, but it appears effortless. Installed julia with brew. And followed the instructions.
  • I think:
    • Installed atom
    • then did packages->Settings->Installed Packages and it just worked!
  • For using gaston, need to install gnuplot first. I used homebrew.

5. Using JuliaPro (it was free as of 11 Jan 2020)

• 11 Jan 2020 Downloaded JuliaPro from https://juliacomputing.com/products/juliapro
• Installed it on my Mac Pro old version, asked for package server but accepted old version.
• 13 Jan 2020 : it works very well so far.
  • Transferring from JuMP is however a challenge. The MathProg interface has changed, to MOI and some of the Julia functions were moved to packages and not in base (e.g. "sparse")

Some debugs.

• sometime, particularly after Mac updates, I get problems. I remove all of ~/.julia/v0.6 and reinstall packages.
• Python stuff is beyond fragile. For instance PyPlot did not run out of the box with clean install on 08 Mar 2018 . One solution:
  • run("install_name_tool -change @rpath/libiomp5.dylib @loader_path/libiomp5.dylib $(Pkg.dir("Conda", "deps/usr/lib/libmkl_intel_thread. dylib"))")
  • Conda.update()
  • Pkg.build("PyCall")
• The compile tools I use through Homebrew. Once in a while, they are off (and I used brew update and upgrade quite regularly). Then things like Rmath do not build. An idea:
  • brew reinstall $(brew deps gcc); brew reinstall gcc
• Compile tools; brew does not by default kill old versions of tools, that needs to be done with "brew cleanup gcc". That led to linking errors, as if some dependencies are updated and some not; you may use the old compiler that looks for a dependency that does not exist. I experienced this with Gadfly and Rmath.
• Check which Xcode version you have: /usr/bin/xcodebuild -version