Julia in VS Code #1
Getting started

Chris Griffiths, Eva Delmas

November 18, 2020

This document covers the following:

- Installing Julia and Visual Studio Code
- Adding extensions - i.e. setting up your computer
- Julia's package manager: installing and loading packages

First off, check out the Introductory videos on the visual studio webpage. These will help explain some of the terminology used below. They'll also show you how to change the colour theme and icon symbols (by far the most important part of any coding tutorial...)

1 Download and install Julia and VS Code

1. Julia

Follow the platform-specific instruction on this page to dowload and install Julia (current stable release).

2. Visual Studio Code

Go to this page to dowload and install VSCode.

2 Setting up VSCode

Note - VSCode extensions are higher level packages that allow you to use different coding languages, edit your VS code themes and icons, or provide helpful applications like spell checker or Bracket Pair Colorizer (colours pairs of brackets)

- Open VSCode
- Install the Julia extension in VSCode (only need to do this once)

  - navigate to the marketplace (fourth symbol down in the activity bar - vertical panel on the left)
  - search for Julia
  - install Julia, this extension provides support for the Julia programming language.
  - install Julia Formatter (this will help you write clean code that are easy to read)
  - there are **many** extensions that can make your life easier, explore the marketplace for extensions that can help you with pdf, csv files, etc. You can do this later, when you need them.

2.1 Projects

A good practice to adopt is to always work in a contained project environment, because we usually want different contexts (package versions, working directory, location of the data folder, etc.) for different projects.

- Start by creating a toy folder, name it as you want (we will use JuliaTuto for the examples, simply replace this name by the one you chose).

To open VSCode in a project (e.g. here JuliaTuto):

- click on the explorer symbol (top symbol on the activity bar) and click Open Folder
- Navigate to the JuliaTuto folder and click open - this becomes the location of your working directory (same as an RProject). Your directory will appear as a vertical pane on the left hand side of the screen.

You can then create a new file (a script) using (cmd-N, File>New File or left click>New File in the directory) and save it with .jl extension (tells VS code you want to use the Julia language)

Note: you can also right-click on the folder (in Finder, Windows file explorer or Linux nautilus) and select Open with -> Other -> VSCode. Or if you are a command-line addict, `cd` into the directory and type `code .` to open VSCode (needs to be activated for Mac).

Now, open a new Julia REPL (stands for - read, execute, print and loop) the REPL is like the console in R. It's VSCodes using Julia for a brain do this by typing Alt-J Alt-O or by typing Start REPL in the command palette (accessed using F1, cmd-shift-P or View>Command Palette).

Type `print(Hello world)` in your new script, and execute using Ctrl-Enter.

3 Julia package manager

For an introduction to the package manager, see the documentation. The "Background and design" section provides useful details if you want to understand more about its originality compared to other languages and how it works.

3.1 Projet and Manifest files

We mentionned above how we may want to work within an environment specific to each project. The Julia package manager allows you to do that easily: *Unlike traditional package managers, which install and manage a single global set of packages, Pkg is designed around environments: independent sets of packages that can be local to an individual project or shared and selected by name* (from the documentation).

The environment is stored in two files:

- Project.toml stores the list of package you have installed, with their version
- Manifest.toml does the same thing but for all the packages dependencies

You don't have to create or modify these files, they will be automatically created once you have activated your project, when you install your first package.

1. Using the package manager

There are two ways to use the package manager (`Pkg`)

- from the REPL:
  - open the REPL
  - type `]` -> you see that instead of seeing `julia>` you now see `(JuliaTuto) pkg>`, indicating that all the package that you now install (or update) will be installed (or updated) in this specific project.
  - (exit the REPL package manager by using the backspace)
- using `Pkg`, this is useful when you want to use the package manager from your script:
  - type `import Pkg`
  - now you can use the package manager commands by typing `Pgk.function()`

There are two ways to check that you are actually working into the right environment:

- check/click the 'Julia env:...' on the bottom left of your screen, it should match your folder name
- enter the package manager by typing] in the Julia REPL, you should see '(your-folder-name) pkg>' instead of 'julia>'. Exit it using backspace.

2. Activating a project

- Activate `] activate .` or `Pkg.activate(".")`
  - activating is done automatically in VScode, activating a directory makes it the "active project" which package operations manipulate.

  - note: the dot stands for the current working directory (`pwd` in julia) so you can also type `Pkg.activate("path/to/folder/JuliaTuto")` or `Pkg.activate(pwd())`

3. Working on someone else's project

If you are working on someone else's project (or on your project but from a different computer), you want to use the same package versions, so after activating the directory, you can install all the packages and the dependencies they referred to in their Project and Manifest files by typing either `] instantiate` from the REPL or `Pkg.instantiate()`.

3.2 Package manager

Now that we are all set up, we are going to try to install a package, to check the environment status and to remove the package.

- type `] add Plots` in the REPL (or `Pkg.add("Plots")` in you script and execute using Ctrl-Enter) -> you just added the Plots package (equivalent to Base plots in R) to your project. This is the equivalent to R `install.packages` function.
- type `] st` in the REPL (this is the status command) -> You should see something like:

```
(JuliaTuto) pkg> st
    Status `~/projets/JuliaTuto/Project.toml`
    [91a5bcdd] Plots v1.6.12
```

- to use an installed package (e.g. Plots), type `using Plots` This imports all the functions of the Plots package.
- you can update package versions using either `] up` or `Pkg.update()` (this can be done for a specific package: `] up Plots`, as you have installed the latest version, this will likely return `[no changes]`)
- type `] rm Plots` in the REPL or `Pkg.rm("Plots")` in your script and execute -> you removed the Plots package from your project
- now if you look at your directory status using `st` the list should be empty.

We have tested these tutorials with a specific list of packages. So downloads the Project and Manifest files from [this link: https://github.com/cagriffiths/VS-code-for-Julia][https://github.com/cagri
code-for-Julia] and replace your Manifest and Project with these files. Then activate (if not already done) and instantiate (see below).

4 Script

```julia
#import the functions from the Pkg manager
import Pkg
#if not already done, activate your project
Pkg.activate(".")
#download Project and Manifest
download("https://raw.githubusercontent.com/cagriffiths/VS-code-for-Julia/main/Project.toml",
"Project.toml")
download("https://raw.githubusercontent.com/cagriffiths/VS-code-for-Julia/main/Manifest.toml",
"Manifest.toml")
#install the packages necessary for these tutorials
Pkg.instantiate()
#check the pkg status
Pkg.status()
```

This is what you should see:

```
(JuliaTuto) pkg> st
    Status `~/projets/VS-code-for-Julia/Project.toml`
  [9b49b652] BioEnergeticFoodWebs v1.1.2 #fix_rates_normalization
(https://github.com/PoisotLab/BioEnergeticFoodWebs.jl.git)
  [a93c6f00] DataFrames v0.21.8
  [0c46a032] DifferentialEquations v6.15.0
  [31c24e10] Distributions v0.23.8
  [f03a62fe] EcologicalNetworks v0.3.0
  [c91e804a] Gadfly v1.3.1
  [033835bb] JLD2 v0.2.4
  [91a5bcdd] Plots v1.6.12
  [ce6b1742] RDatasets v0.6.10
  [44d3d7a6] Weave v0.10.6
  [8bb1440f] DelimitedFiles
  [9a3f8284] Random
```