

Programación en Julia: Herramientas para el aprendizaje

CADI

Héctor Medel

Benjamín Pérez

Tecnológico de Monterrey

January 22, 2023

Toma de asistencia

Objetivo

Aprender de manera práctica el uso de Julia y Pluto.jl como herramienta para la enseñanza de cursos en STEM.

Fechas y horarios

- ▶ Sesiones sincrónicas: Del 23 al 27 de enero (Lunes a Viernes) de 09:00 a 13:00 hrs.
- ▶ Actividades asincrónicas: Del 23 al 27 de enero (Lunes a Viernes) de 14:00 a 16:00 hrs.
- ▶ Modalidad: Virtual.

Políticas para acreditar el curso

- ▶ **Asistencia** de al menos el 80% del taller.
- ▶ A lo largo de la semana se encargarán aproximadamente **3 tareas**. Las tareas serán entregadas en equipos de 3 integrantes.
- ▶ La correcta solución de las tareas deberá ser entregada a más tardar el **viernes 27 de enero a las 23:59 hrs.**

Temario del curso

1. Presentación e instalación
2. Crash course de Julia
3. Editores y notebooks
4. Instalación de Pluto
5. PlutoUI, creación de notebooks interactivas
6. Uso práctico con ejemplos
7. Graficación
8. Integración de paquetes
9. Uso de SymPy.jl
10. Interactividad en sitios web


Agenda

LUNES	MARTES	MIÉRCOLES	JUEVES	VIERNES
Introducción/motivación	Markdown y presentaciones	Ejemplos	Notebooks de pluto en sitios web	Presentaciones por equipo
Instalación	Manejo de Pluto y PlutoUI	Sympy y otras funciones	Dudas	
Crash course de Julia (Actividad de repaso)	Ejemplos			
Actividad 1	Continuación de actividad 1			
Tarea 1	Tarea 2		Tarea 3	

Motivación


Instalación de Julia

- ▶ Para descargar Julia ingresa al sitio <https://julialang.org/downloads/>
- ▶ Descarga el archivo correspondiente a tu sistema operativo (en windows v1.6.7).



[Download](#)[Documentation](#)[Learn](#)[Blog](#)[Community](#)[Research](#)[JSoC](#)[Sponsor](#)

Download Julia

 Star 41,229

Please star us on [GitHub](#). If you use Julia in your research, please [cite us](#). If possible, do consider [sponsoring us](#).

Current stable release: v1.8.4 (December 23, 2022)

Checksums for this release are available in both [MD5](#) and [SHA256](#) formats.

Windows [help]	64-bit (Installer), 64-bit (portable)	32-bit (Installer), 32-bit (portable)
macOS x86 (Intel or Rosetta) [help]	64-bit (.dmg), 64-bit (.tar.gz)	
macOS ARM (M-series Processor) [help]	64-bit (.dmg), 64-bit (.tar.gz)	
Generic Linux on x86 [help]	64-bit (glibc) (GPG), 64-bit (musl) ^[1] (GPG)	32-bit (GPG)
Generic Linux on ARM [help]	64-bit (AArch64) (GPG)	
Generic FreeBSD on x86 [help]	64-bit (GPG)	
Source	Tarball (GPG)	Tarball with dependencies (GPG) GitHub

Almost everyone should be downloading and using the latest stable release of Julia. Great care is taken not to break compatibility with older Julia versions, so older code should continue to work with the latest stable Julia release. You should only be using the long-term support (LTS) version of Julia if you work at an organization where implementing or certifying upgrades is prohibitively expensive and there is no need for new language features or packages. See this description of “[Risk Personas](#)” for more detail on who should be using what versions of Julia based on their risk tolerance. See this blog post on [Julia's Release Process](#) for more information on different kinds of releases.

Instalación de Julia

- ▶ Por ahora, es recomendable dejar la configuración por defecto durante el proceso de instalación.
- ▶ Dependiendo de tu OS, sigue las instrucciones del instalador.



Download Julia

Star 41,229

Please star us on [GitHub](#). If you use Julia in your research, please [cite us](#). If possible, do consider [sponsoring us](#).

Current stable release: v1.8.4 (December 23, 2022)

Checksums for this release are available in both [MD5](#) and [SHA256](#) formats.

Windows [help]	64-bit (installer), 64-bit (portable)	32-bit (installer), 32-bit (portable)	
macOS x86 (Intel or Rosetta) [help]	64-bit (.dmg), 64-bit (.tar.gz)		
macOS ARM (M-series Processor) [help]	64-bit (.dmg), 64-bit (.tar.gz)		
Generic Linux on x86 [help]	64-bit (glibc) (GPG), 64-bit (musl) ^[1] (GPG)	32-bit (GPG)	
Generic Linux on ARM [help]	64-bit (AArch64) (GPG)		
Generic FreeBSD on x86 [help]	64-bit (GPG)		
Source	Tarball (GPG)	Tarball with dependencies (GPG)	GitHub

Almost everyone should be downloading and using the latest stable release of Julia. Great care is taken not to break compatibility with older Julia versions, so older code should continue to work with the latest stable Julia release. You should only be using the long-term support (LTS) version of Julia if you work at an organization where implementing or certifying upgrades is prohibitively expensive and there is no need for new language features or packages. See this description of “[Risk Personas](#)” for more detail on who should be using what versions of Julia based on their risk tolerance. See this blog post on [Julia's Release Process](#) for more information on different kinds of releases.

Trabajando en Julia – Real Evaluate Print Loop (REPL)

- Al correr el archivo ejecutable de Julia, se abrirá una ventana similar a la siguiente.

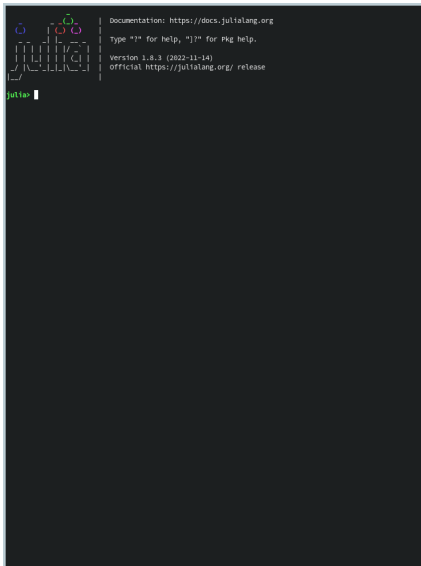
A screenshot of the Julia REPL (Real Evaluate Print Loop) window. The window has a dark background. In the top left corner, there is a small logo consisting of four colored circles (blue, green, red, yellow) arranged in a square. To the right of the logo, the text "Documentation: https://docs.julialang.org" is displayed. Below this, the text "Type '?' for help, ']' for pkg help." is shown. Further down, the text "Version 1.8.3 (2022-11-14)" and "official https://julialang.org/ release" are visible. At the bottom left, the prompt "julia>" is shown in green, followed by a white cursor. The rest of the window is empty.

```
Documentation: https://docs.julialang.org
Type '?' for help, ']' for pkg help.
Version 1.8.3 (2022-11-14)
official https://julialang.org/ release

julia> |
```

Trabajando en Julia – Real Evaluate Print Loop (REPL)

- Sigamos en la terminal.



```
Documentation: https://docs.julialang.org
Type "?" for help, "?>" for pkg help.
Version 1.8.3 (2022-11-14)
official https://julialang.org/ release

julia> |
```

Probemos los siguientes comandos

```
julia> 6 * 7
42
julia> ans
42
julia> ans + 10
52
```

Si por alguna razón no queremos que se despliegue el resultado, agregamos ; al final.

Podemos **asignar** un valor a una variable

```
julia> a = 6 * 7  
42  
julia> b = "Hola"  
"Hola"
```

Algunos comandos básicos en el REPL

- ▶ Flecha hacia arriba/abajo nos ayudan a navegar en el historial de comandos ejecutados.
- ▶ Borrar pantalla CTRL+L
- ▶ Interrumpir la ejecución de un comando CTRL+C

Accesar a la [documentación/ayuda](#)

Cuando ingresamos el caracter ? en el REPL, notemos que cambia de la siguiente manera

```
help?>
```

Busquemos ayuda acerca de la función coseno.

Accesar a la [documentación/ayuda](#)

Cuando ingresamos el caracter ? en el REPL, notemos que cambia de la siguiente manera

```
help?>
```

Busquemos ayuda acerca de la función coseno.

```
help?> cos
```

```
search: cos cosh cosd cosc cospi acos acosh acosd sincos sincosd sincospi
```

```
cos(x)
```

```
Compute cosine of x, where x is in radians.
```

```
See also [cosd], [cospi], [sincos], [cis].
```

Manejo de paquetes (pkg)

Cuando ingresamos el caracter] en el REPL, notemos que cambia de la siguiente manera

```
(@v1.8) pkg>
```

Esto es conocido como el modo pkg. Dentro de este entorno es como instalamos (y compartimos) librerías y paquetes.

Manejo de paquetes (pkg)

Podemos trabajar de manera más ordenada si definimos “proyectos” donde instalaremos los paquetes. Para lo anterior **crearemos** una carpeta para las actividades de la semana. En mi caso la ruta es `/home/ben/.../Dell/cadi/`
Una vez creada la carpeta, dentro de Julia hacemos lo siguiente:

```
julia> cd("PATH")  
(@v1.8) pkg> activate .  
(cadi) pkg>
```

Los paquetes que instalemos ahora, serán instalados para el proyecto `cadi`. Esto nos permite trabajar de manera más limpia.

Instalemos un paquete

Ejecutemos las siguientes líneas dentro del modo pkg

```
(cadi) pkg> add BenchmarkTools
```

Instalemos un paquete

Ejecutemos las siguientes líneas dentro del modo pkg

```
(cadi) pkg> add BenchmarkTools
```

Para salir del modo pkg, presionamos la tecla BACKSPACE

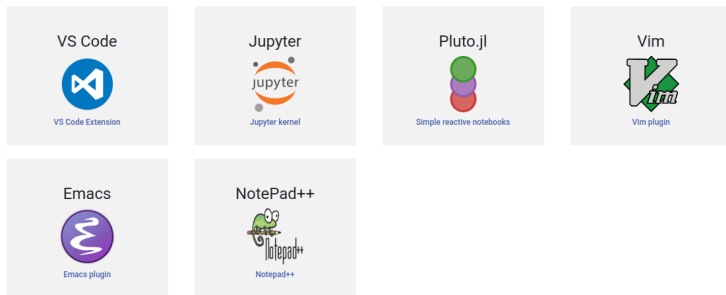
```
julia>
```

```
julia> using BenchmarkTools
```

```
julia> @benchmark rand(1000)
```

Por ahora hemos interactuado con Julia vía el REPL...

Existen diversos **IDEs** y **Editores**, por ejemplo



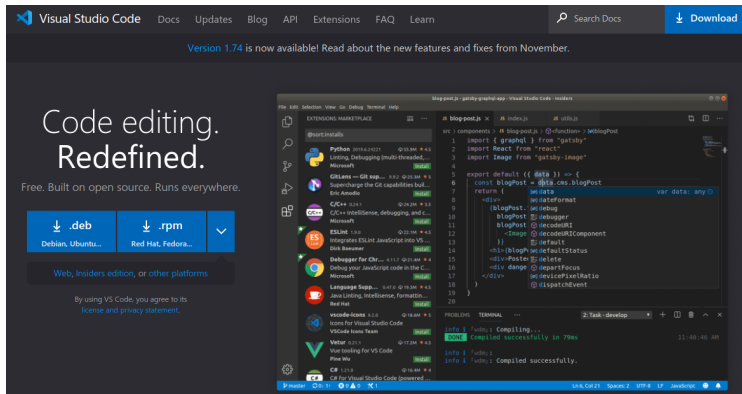
Veremos la instalación de VSCode y Pluto.jl

¿Qué es VSCode?

- ▶ Editor de código multiplataforma.
- ▶ Soporta varios lenguajes, entre ellos Julia.

Instalación de VSCode

- ▶ Para descargar VSCode ingresa al sitio <https://code.visualstudio.com/>
- ▶ Descarga instala el archivo correspondiente a tu sistema operativo.



Extensión de Julia

- ▶ Dentro de VSCode, instalaremos la extensión para Julia.
- ▶ Abre el menú de extensiones que se encuentra en la barra vertical de la izquierda.
- ▶ En el cuadro de búsqueda escribe `julia`, e instala la extensión.

¿Qué es Pluto.jl?

- ▶ Entorno de programación para Julia tipo [notebook](#).
- ▶ Código [interactivo/reactivo](#).

Instalemos Pluto.jl

Dentro de la terminal instalaremos el paquete como lo hicimos para el caso del paquete Plots. Es decir

```
(cadi) pkg> add Pluto
```

Instalemos Pluto.jl

Dentro de la terminal instalaremos el paquete como lo hicimos para el caso del paquete Plots. Es decir

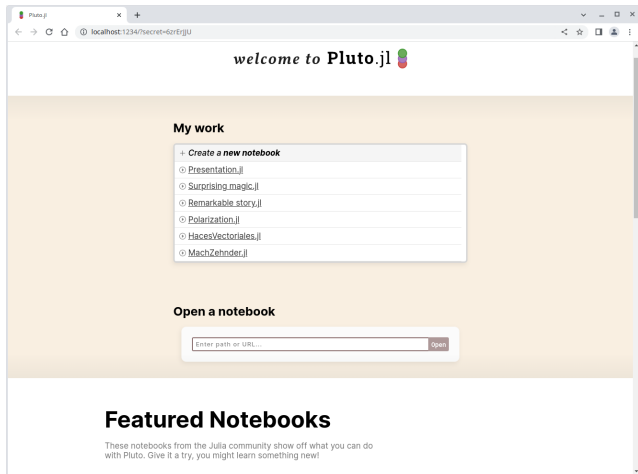
```
(cadi) pkg> add Pluto
```

Posteriormente, salimos del entorno Pkg, cargamos Pluto y lo ejecutamos.

```
julia> using Pluto  
julia> Pluto.run()
```

Instalemos Pluto.jl

Lo anterior abrirá una ventana de nuestro navegador



Actividad en equipos

- ▶ Instalar Julia.
- ▶ Instalar VSCode y el plugin de Julia.
- ▶ Instalar el paquete de gráficas Plots.
- ▶ Instalar Pluto.