



Introducción al lenguaje Julia



Francisco Gárate

Jornadas Medialab-Prado

13-14, febrero 2018

¿Quién es Julia?




- ▶ Julia es un lenguaje de programación libre, de código abierto (licencia MIT) y con librerías amigables.
- ▶ Creado en el MIT por Jeff Bezanson, Alan Edelman, Stefan Karpinski y Viral Shah en 2009, y publicado en 2012.
- ▶ Diseñado especialmente para entornos científicos y matemáticos.
- ▶ *"Walks like Python. Runs like C"*. Se lee igual que Python o Octave, pero se comporta casi igual de rápido que C.
- ▶ Lenguaje diseñado para correr en paralelo y en la nube fácilmente.
- ▶ Documentación: <https://docs.julialang.org>

Instalar Julia

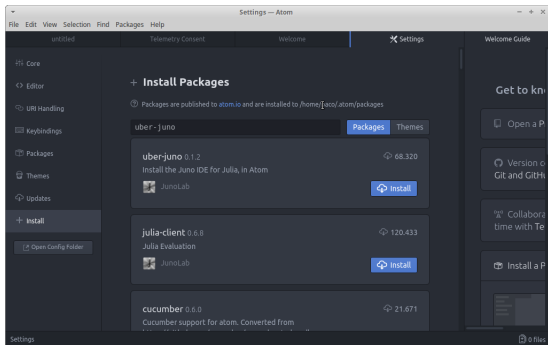


- ▶ Descargar instalador: <http://julialang.org/downloads/>
- ▶ En linux: `apt-get install julia`
- ▶ Julia incluye un terminal interactivo llamado REPL:
 - ▶ `;` La siguiente instrucción se ejecuta como si fuera una consola de linux
 - ▶ `?` Busca el texto a introducir en los manuales de ayuda

```
Terminal - paco@pbell: ~  
Archivo  Editar  Ver  Terminal  Pestañas  Ayuda  
paco@pbell:~$ julia  
 | A fresh approach to technical computing  
Documentation: http://docs.julialang.org  
Type "?help" for help.  
  
Version 0.4.5 (2016-03-18 00:58 UTC)  
x86_64-linux-gnu  
  
julia> |
```



- ▶ Soporte para: Vi(m), emacs, Eclipse, Sublime Text...
- ▶ Recomendación: Atom + Juno (<http://junolab.org>)

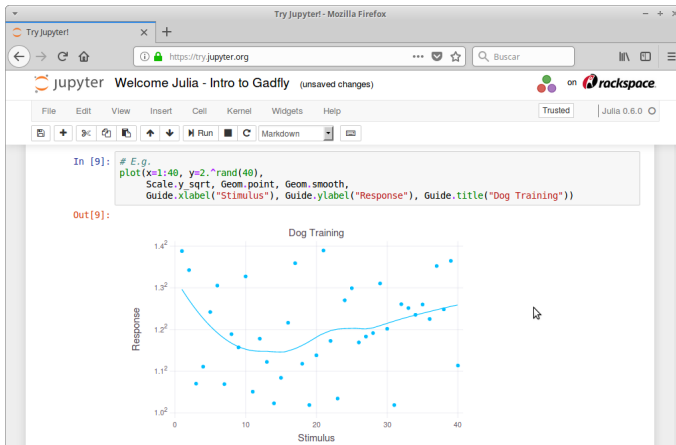


desde Atom: **Ctrl ,** o **Cmd ,**, Install, 'uber-juno'

Prueba Julia (sin instalar)



- ▶ Jupyter Notebook: IJulia (similar a IPython):
<https://try.jupyter.org>
- ▶ www.juliabox.com Incluye +100 paquetes



Paquetes (Packages)



- ▶ Repositorio oficial: <http://pkg.julialang.org>
- ▶ Existen actualmente 1.703 paquetes registrados y testeados:
<http://pkg.julialang.org/pulse.html>
- ▶ Para instalar: `Pkg.add("Distributions")`
- ▶ Desinstalar: `Pkg.rm("Distributions")`
- ▶ Para ver el estado de tus paquetes: `Pkg.status()`



- ▶ `println("Hola Mundo")`
- ▶ Desde shell: `julia holamundo.jl`
- ▶ Operadores `+`, `-`, `*`, `/`
`a = 4.0`
`b = 7.0`
`c = (b - a) * (b + a) / a`
- ▶ Funciones matemáticas familiares:
`result = exp(-2.33) * cos(22 * 180 / pi)`
- ▶ Funciones incluidas de serie. Por ejemplo:
`rand()` (*genera números aleatorios entre 0 y 1*)
`std(v)/mean(v)`
- ▶ Condicionales, listas, loops, rangos, etc.. al estilo Python

¿Por qué Julia?



Teniendo R, Matlab, Mathematica, Python...
¿De verdad hace falta otro lenguaje de programación?



¿Por qué Julia?



- ▶ Julia no es popular.
- ▶ No aparece en el ranking de los lenguajes más populares de GitHub ¹ ni en el Top 20 Most Popular de Stackoverflow².
- ▶ Tampoco la NASA³ o el CERN han liberado ninguna librería (de momento).
- ▶ ... pero eso también mola!
- ▶ Veamos las ventajas de Julia frente a otros lenguajes:

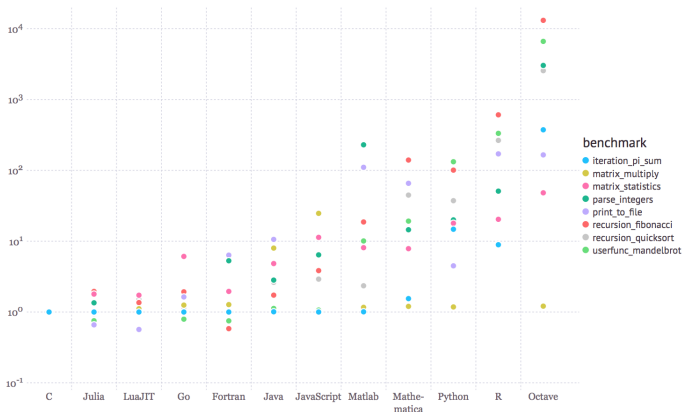


- ▶ Se introducen símbolos al igual que \LaTeX

```
usp-unicode.jl
1  λ = 1.0
2  Σπ = 3.0
3  γ = 2.0
4  \delt
   δ delta
   Δ Delta
8  σ = exp(γ+(0.5+Σπ*ln(yt/xt))/(Σπ))
9
```



- Su compilador JIT (Just-In-Time) basado en LLVM (Máquina Virtual de Bajo Nivel) permite a Julia acercarse, e incluso igualar, a C.



Ventajas

Computación en paralelo de forma nativa



```
parallel_pi.jl — ~/Escritorio/ejemplos — Atom
File Edit View Julia Selection Find Packages Help

parallel_pi.jl
1 addprocs(2) # Add 2 Julia worker processes
2
3 function parallel_pi(n)
4     in_circle = @parallel (+) for i in 1:n # <-- (+) suma el trabajo repartido
5         x = rand()
6         y = rand()
7         Int((x^2 + y^2) < 1.0)
8     end
9     return (in_circle/n) * 4.0
10 end
11
12 parallel_pi(10000)
```

- ▶ Repositorio de modelos para programación en paralelo:
<https://juliaparallel.github.io>
- ▶ Celeste.jl⁴: Librería que ha incrementado x225 la velocidad del análisis de imágenes (dataset de 55 TB) ayudando al NERSC en la catalogación de objetos astronómicos



- ▶ Licencia MIT (compatible con GNU GPL)
- ▶ Robusto ecosistema de herramientas estadísticas, distribución en paralela, optimización y de visualización de datos.
- ▶ Librerías gráficas (como Gadfly) para jugar en primera división.
- ▶ Llamadas a librerías C, Fortran de forma nativa:

```
julia>ccall(:clock, Int32, ())
```

1180595

- ▶ ...y a Python (a través de PyCall.jl)
- ▶ Sintaxis familiares a Python
- ▶ En resumen: Unir facilidad de uso y alto rendimiento es posible.



- ▶ Se buscan voluntarios para:
 - ▶ Dar a conocer el lenguaje Julia
 - ▶ Traducir la documentación oficial al castellano
 - ▶ Crear/Potenciar una comunidad de usuarios de Julia en español (blog, meetup, etc...)
 - ▶ Crear o migrar librerías de R/Python a Julia
- ▶ Mas info: `paco@garpa.net` o
 - ▶ `https://github.com/franciscogarate/Introduccion-al-lenguaje-Julia`



- 1 <https://octoverse.github.com>
- 2 <https://insights.stackoverflow.com/survey/2017#technology>
- 3 <https://code.nasa.gov/?q=julia> (*consulta vacía*)
- 4 <https://juliacomputing.com/press/2016/11/28/celeste.html>