Julia

A Guide on Using Julia Language

The language of the future?

Julia or not Julia

Advantages & Disadvantages

1. Not whitespace-sensitive (or indentation-sensitive) xD
2. Parallel computing (and Macros), Multiple Dispatch, etc.
3. Universal! Can be run inside Python, R, C and ... and vice versa
4. Has all the goods in one place (From R to Matlab & from Python to C)
5. Much faster than Python (Due to the use of Just In Time (JIT) compiler)
6. Solves the two-language problem: You can prototype & put into production the same source code
7. Package development is way easier & usually 100% written in Julia rather than C/C++ & Python combination
8. Smaller community, tutorials and sample codes
9. Harder to debug as it doesn't point you exactly to the problem like Python (See packages slides for a remedy)
10. Less packages (Maybe enough for other tasks than ML & DS. Also, can still use Python/R packages with PyCall/RCall)

Some Stuff to Know

1. In Julia you can use *mathematical symbols* like Σ to name variables in contrast to Python. (Σx = 200 vs. sum_x = 200)
2. It was created with the ambition of combining the *speed* of C, *usability* of Python, the *dynamism* of Ruby, the *mathematical power* of MatLab, and the *statistical power* of R
3. Julia's type system is dynamic, yet takes advantages of static type systems by making it possible to indicate that certain values are of specific types. This allows for example, ***method dispatch*** on the types of function arguments to be deeply integrated with the language

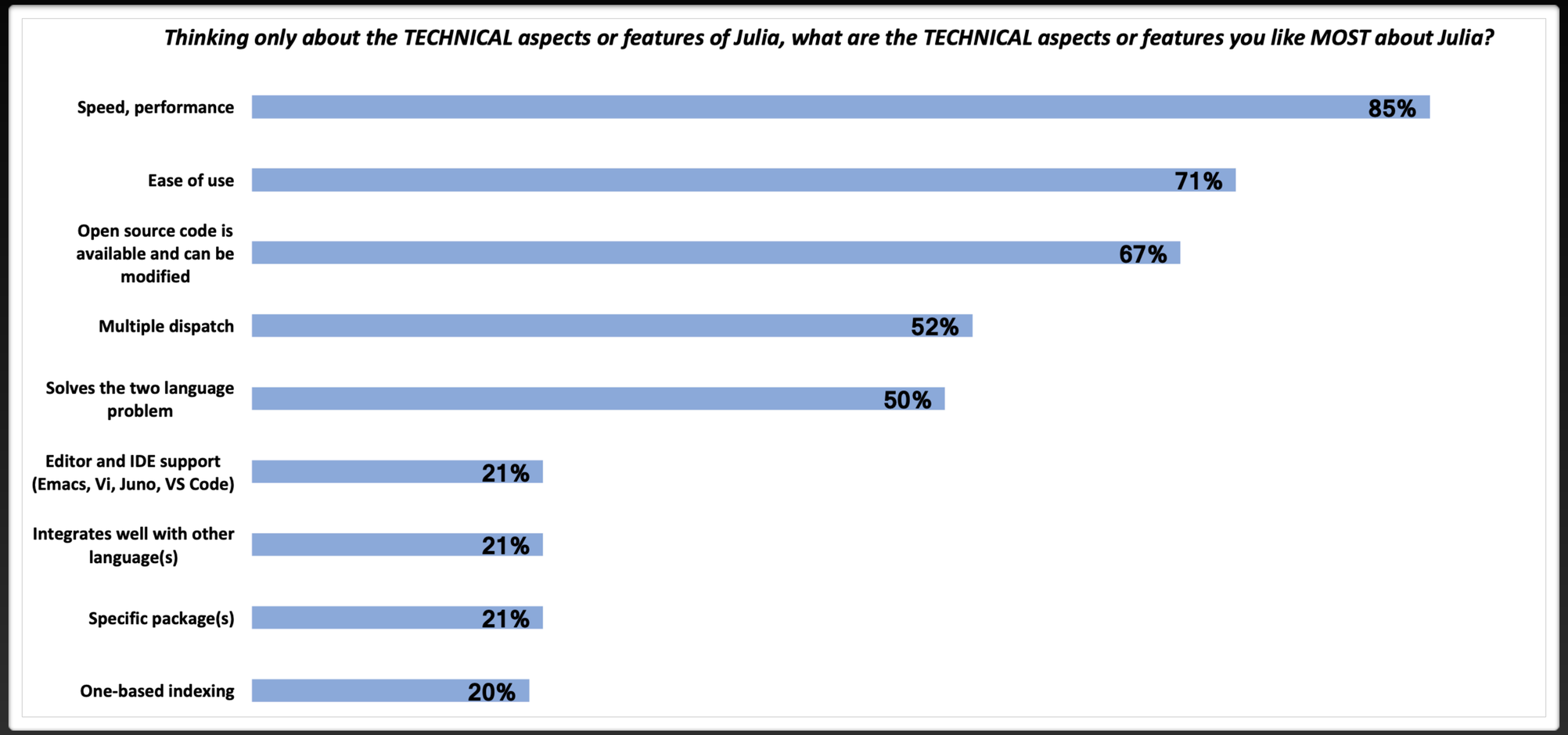
JIT

1. Compilation during execution of a program (at run time), in contrast to most compiled languages that compile byte code to machine code before execution.
2. After the first run (compilation), every other call to the same source code is faster than the first call as it can be observed in the example below:

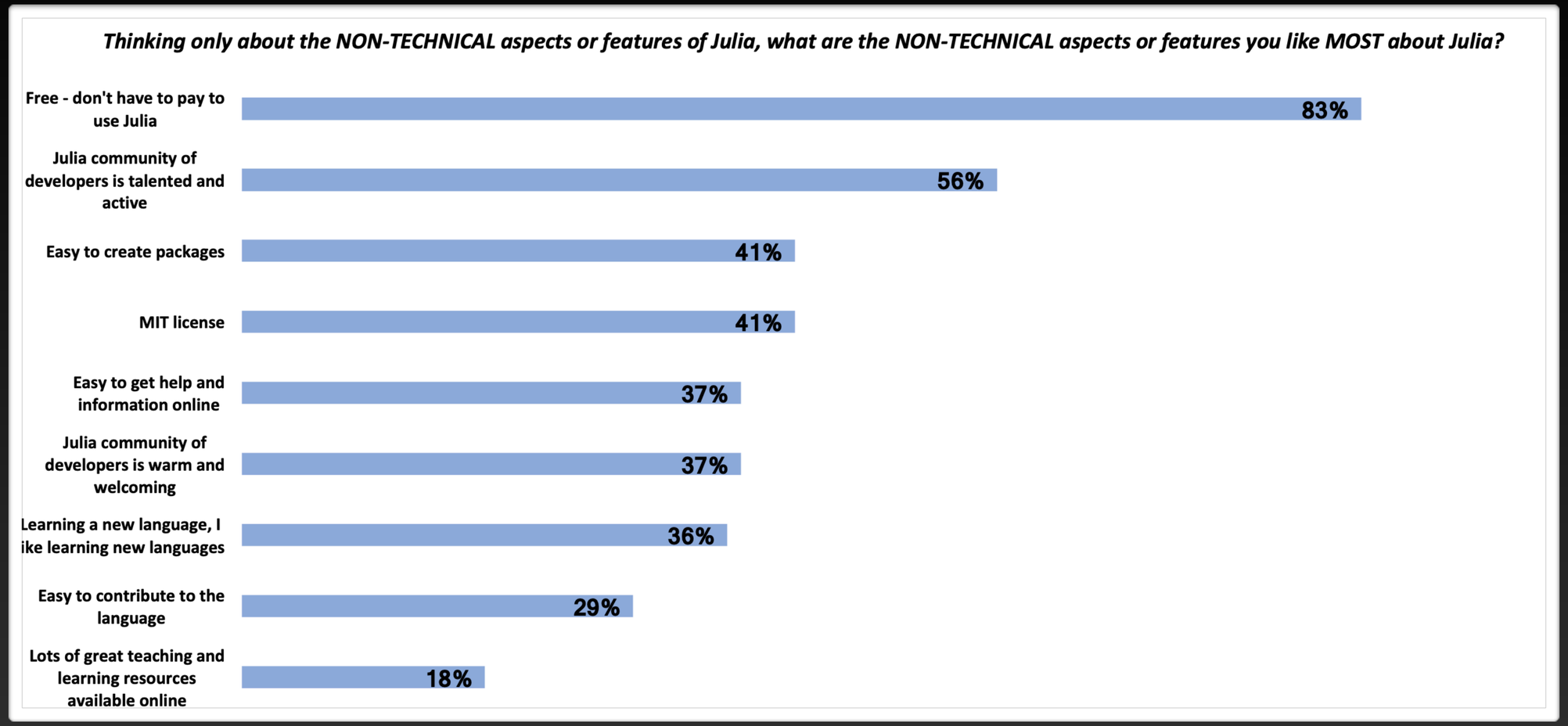
add(20.0, 40.0) => 0.003329 seconds (157 allocations: 10.153 KiB)

add(130.0, 120.0) => 0.000004 seconds (5 allocations: 176 bytes)

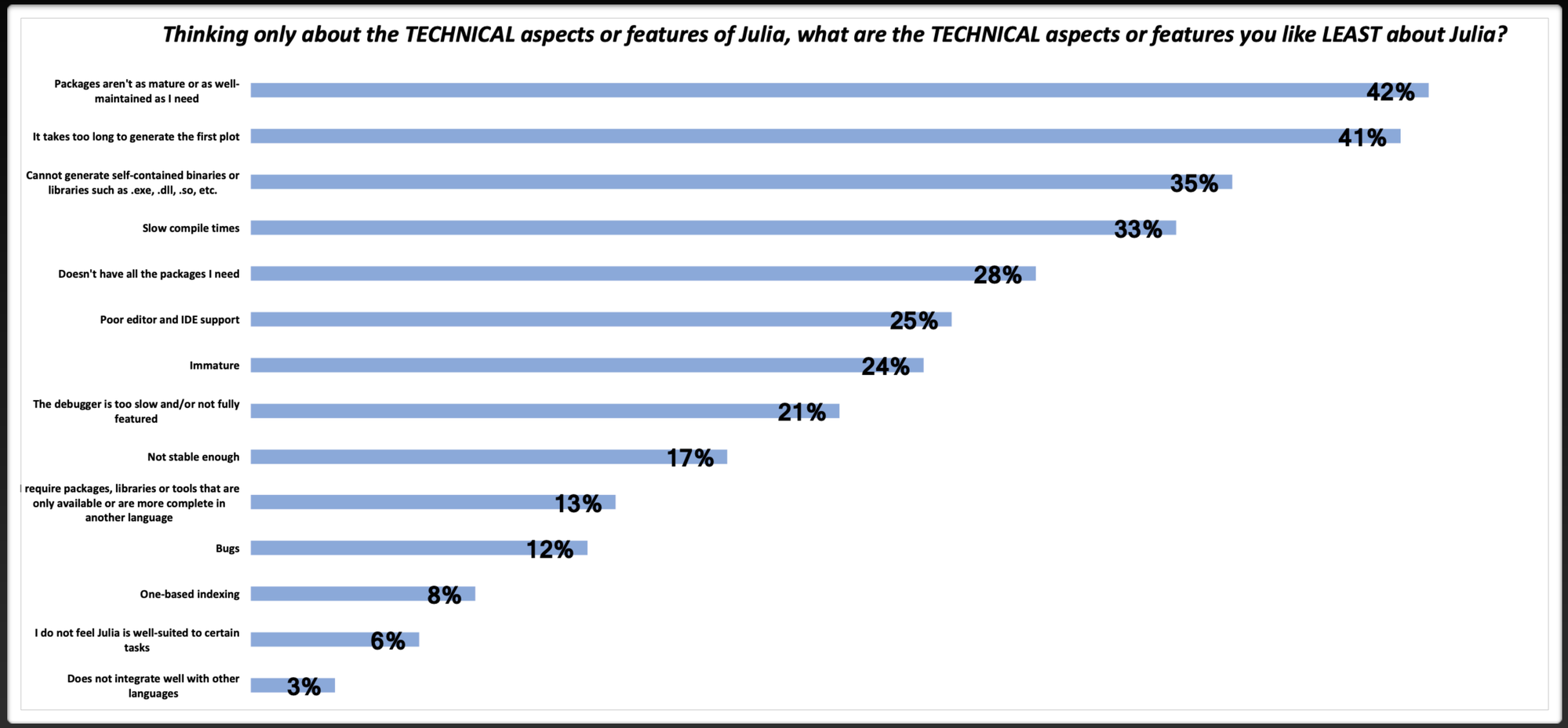
As it is shown above, after the first call with two integers, every other call with any integers will be much faster less memory consuming!



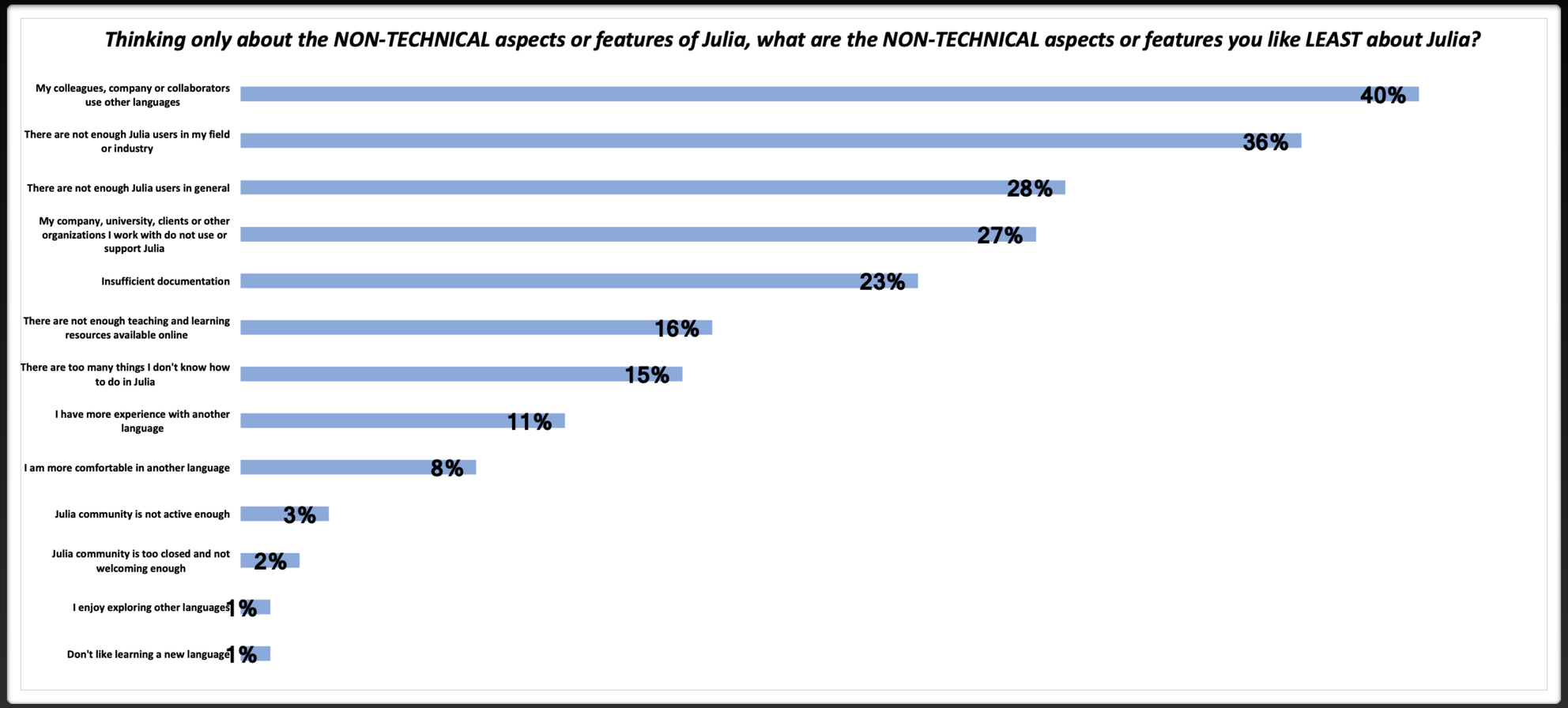
User & Developer Survey 2019, from Julia Computing, Inc.



User & Developer Survey 2019, from Julia Computing, Inc.



User & Developer Survey 2019, from Julia Computing, Inc.



User & Developer Survey 2019, from Julia Computing, Inc.

Installing Julia

Ways to Install

- Using terminal/installer package
- Using Docker
- Using JuliaPro

Installing with Terminal/Package Installer

Windows: 32-bit - 64-bit (Guide)

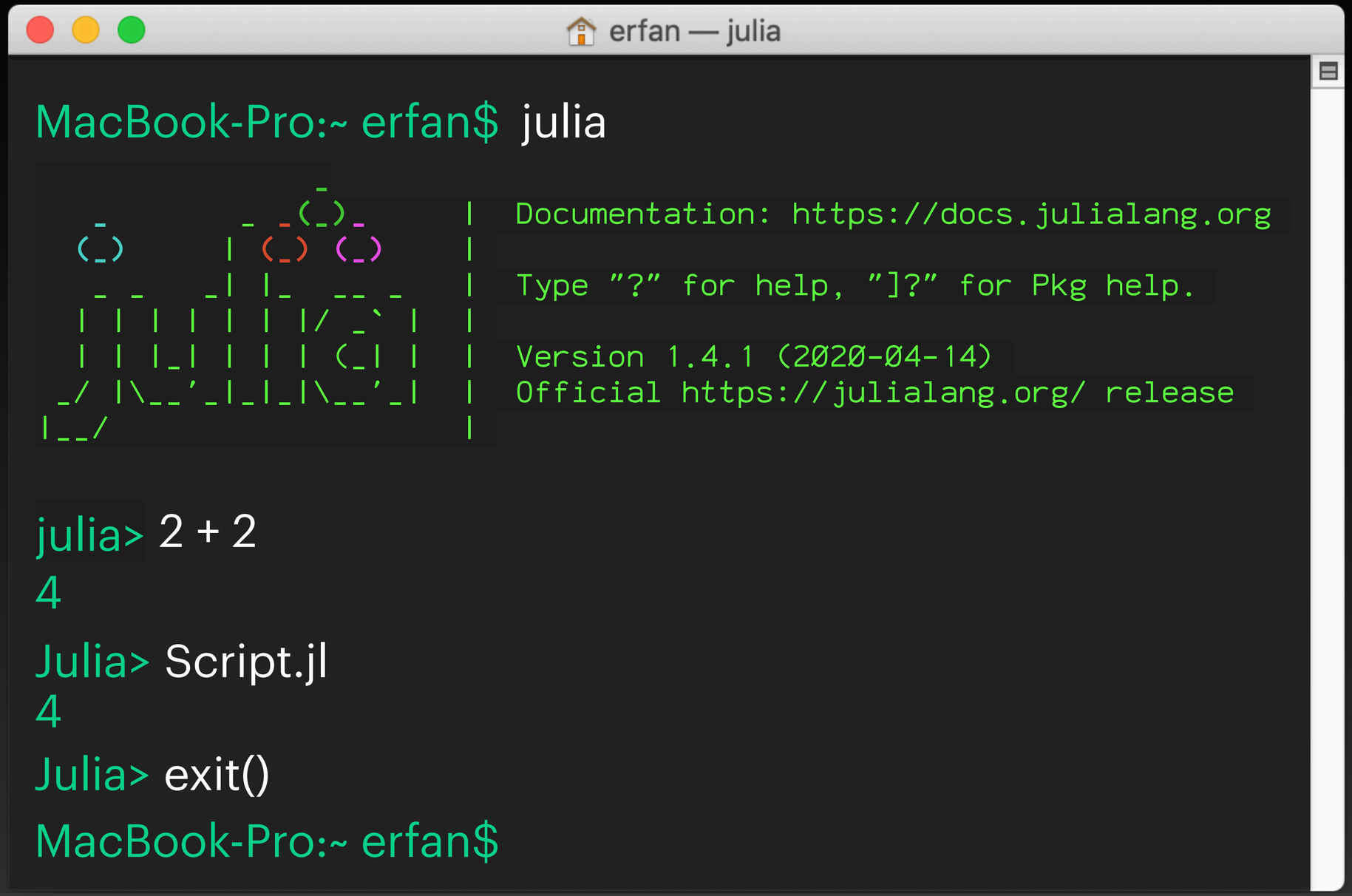
MacOS: 64-bit

Linux: 32-bit - 64-bit

Juno IDE: Download

VSCode Plugin for Julia: Download

PyCharm Plugin for Julia: Download


erfan — julia
MacBook-Pro:~ erfan$ julia
Documentation: https://docs.julialang.org
Type "?" for help, "]?" for Pkg help.
Version 1.4.1 (2020-04-14)
Official https://julialang.org/ release
julia> 2 + 2
4
Julia> Script.jl
4
Julia> exit()
MacBook-Pro:~ erfan$

Installing with JuliaPro

JuliaPro is lightweight, easy to install, comes with many packages, and includes tools like plotting, notebook, etc.

Windows: Download

MacOS: Download

Linux: Download

Project

- ASTInterpreter2
- atom-ink
- atom-julia-client
- atom-indent-detective
- DiffEqUncertainty
- DiffEqDevTools
- Gadfly
- Atom
- junowebsite
- PlotThemes

untitled

```
1  using FFTW
2  function profile_test(n)
3      for i = 1:n
4          A = randn(100,100,20)
5          m = maximum(A)
6          Afft = FFTW.fft(A)
7          Am = mapslices(sum, A, dims=2)
8          B = A[:,:,5]
9          Bsort = mapslices(sort, B, dims=1)
10         b = rand(100)
11         C = B.*b
12     end
13 end
14
15 profile_test(1)  # run once to trigger compilation
16 @profiler profile_test(10)
17 d
```

- λ dct(A [, dims]) FFTW
- λ dct!(A [, dims]) FFTW
- k do
- λ div(x, y) Base
- λ dec Base
- λ done Base
- λ diff(A::AbstractVector) Base
- λ dump(x; maxdepth=8) Base
- λ detach(command) Base
- λ divrem(x, y) Base

Performs a multidimensional type-II discrete cosine transform (DCT) of the array `A`, using the uni...

Workspace | Documentation

Filter

Main

	a	Float64[5]
n	ans	3.75...
λ	profile_test	profile_test

Plots | Profiler

Terminal

```
0.8451939136646798
0.704274825552696
0.45520469644229644
0.7767370648076366
0.96420718131274

julia> sum(ans)
7456176817800486
```

00:00:14
Julia

untitled* 24:23 CRLF UTF-8 Julia master Fetch 2 files 12 updates Spaces (4) Main

General Programming

DataStructures
LightGraphs
Atom
JuliaWebAPI
IJulia
Nettle
DSP
NearestNeighbors
Parameters
ParserCombinator
Libz
BenchmarkTools
Rebugger
Debugger

General Math

Calculus
DataFrames
StatsBase
Distributions
HypothesisTests
GLM
OnlineStats
DifferentialEquations
SymPy
KernelDensity
Zygote

Optimization

Optim
Roots

Databases

JDBC

Building UIs and Visualization

PyPlot
Interact
LaTeXStrings
Formatting
Images
Plots
GR
UnicodePlots
ImageMagick
StatPlots
PGFPlots

Deep Learning and Machine Learning

Knet
Clustering
DecisionTree
MLBase
Flux
TensorFlow
Metalhead
ScikitLearn

Interoperability with Other Languages

RCall (Interoperability with R)
JavaCall (Java)
PyCall (Python)
Conda (Python dependencies)
JuliaInXL (Microsoft Excel)

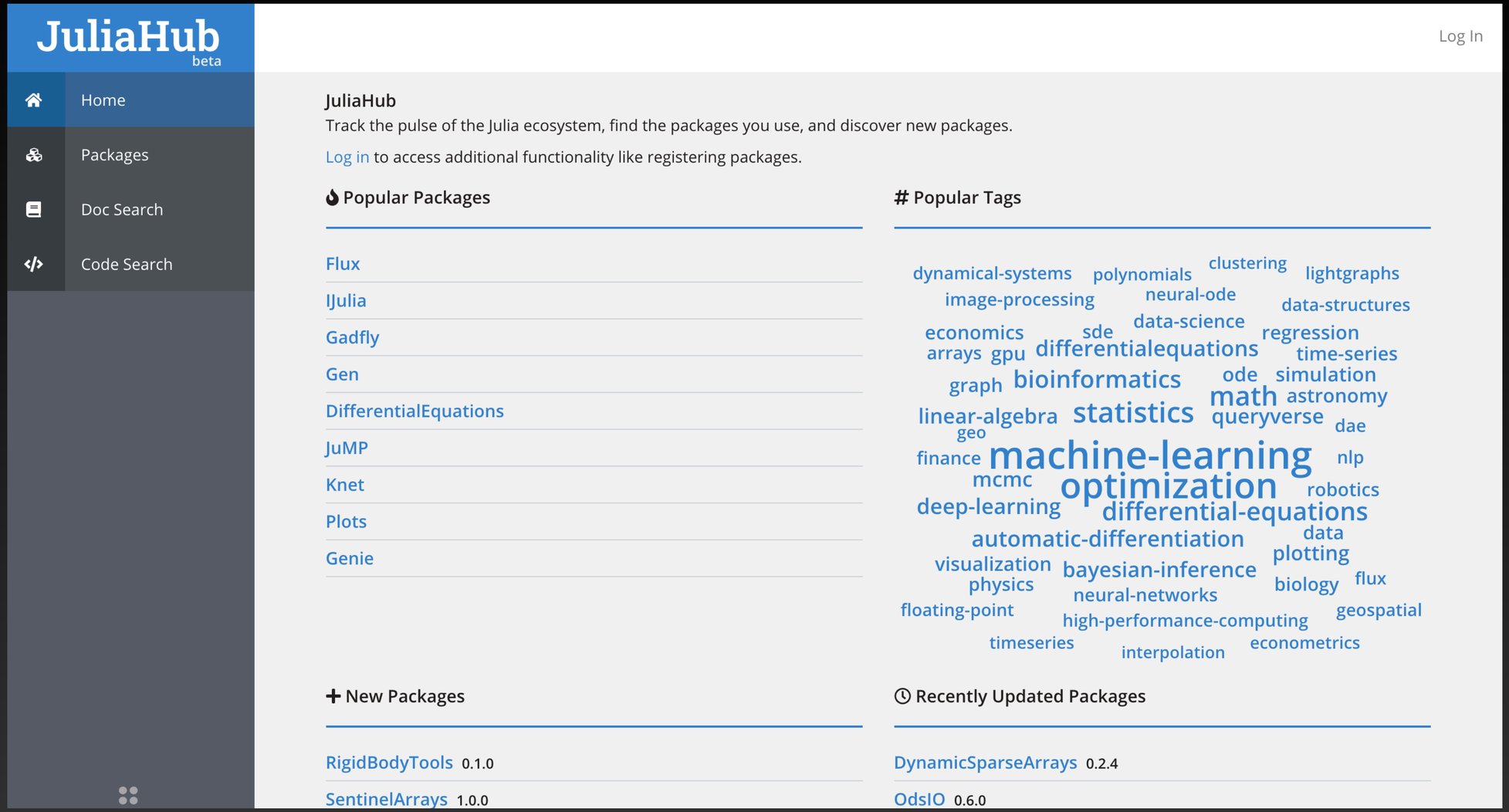
File and Data Formats

JSON
JLD2
CSV
LightXML
StaticArrays
ProtoBuf
CuArrays

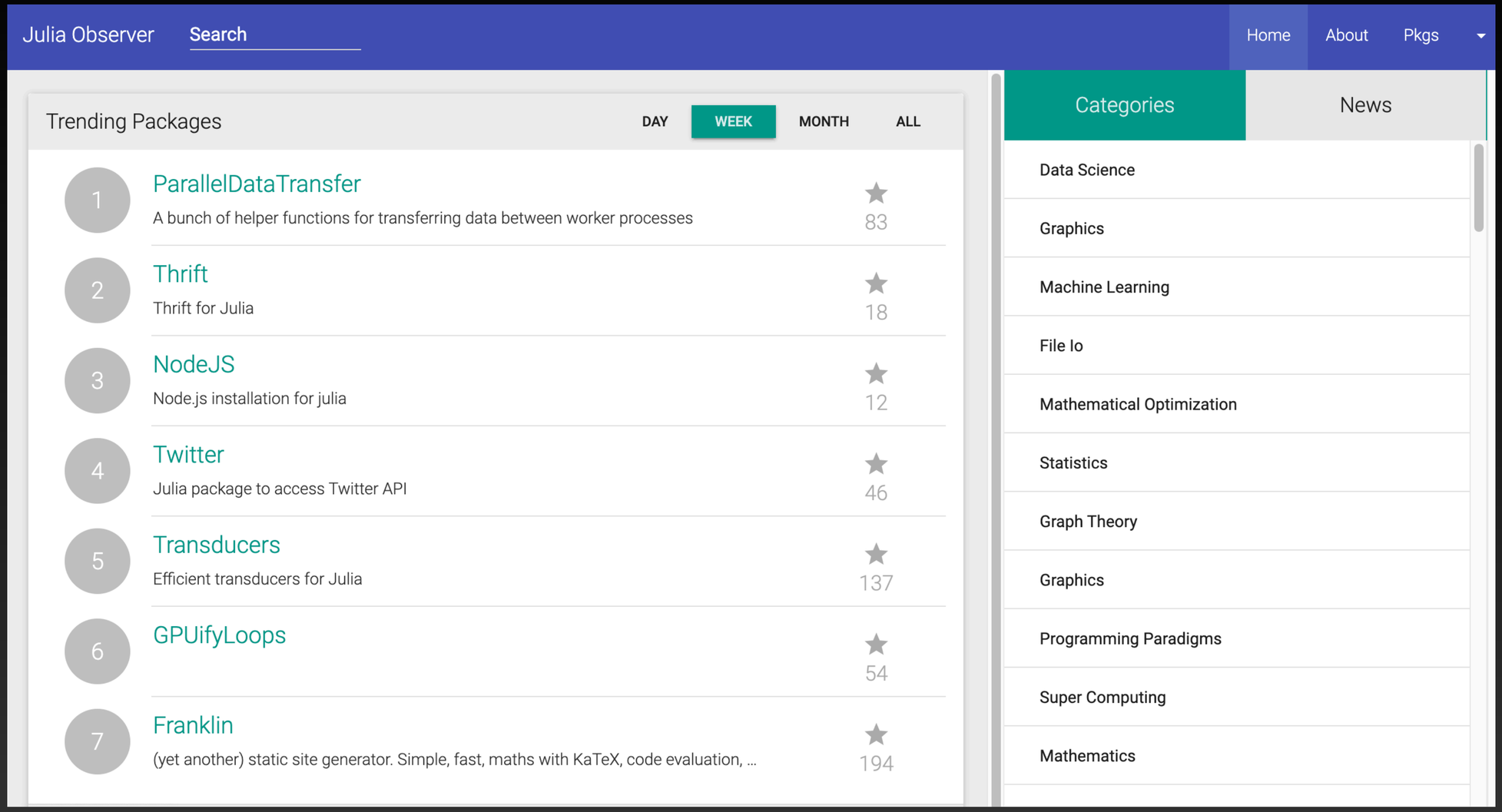
Economics and Finance

QuantEcon
BusinessDays
Bloomberg
Blpapi (Bloomberg connector)
Miletus

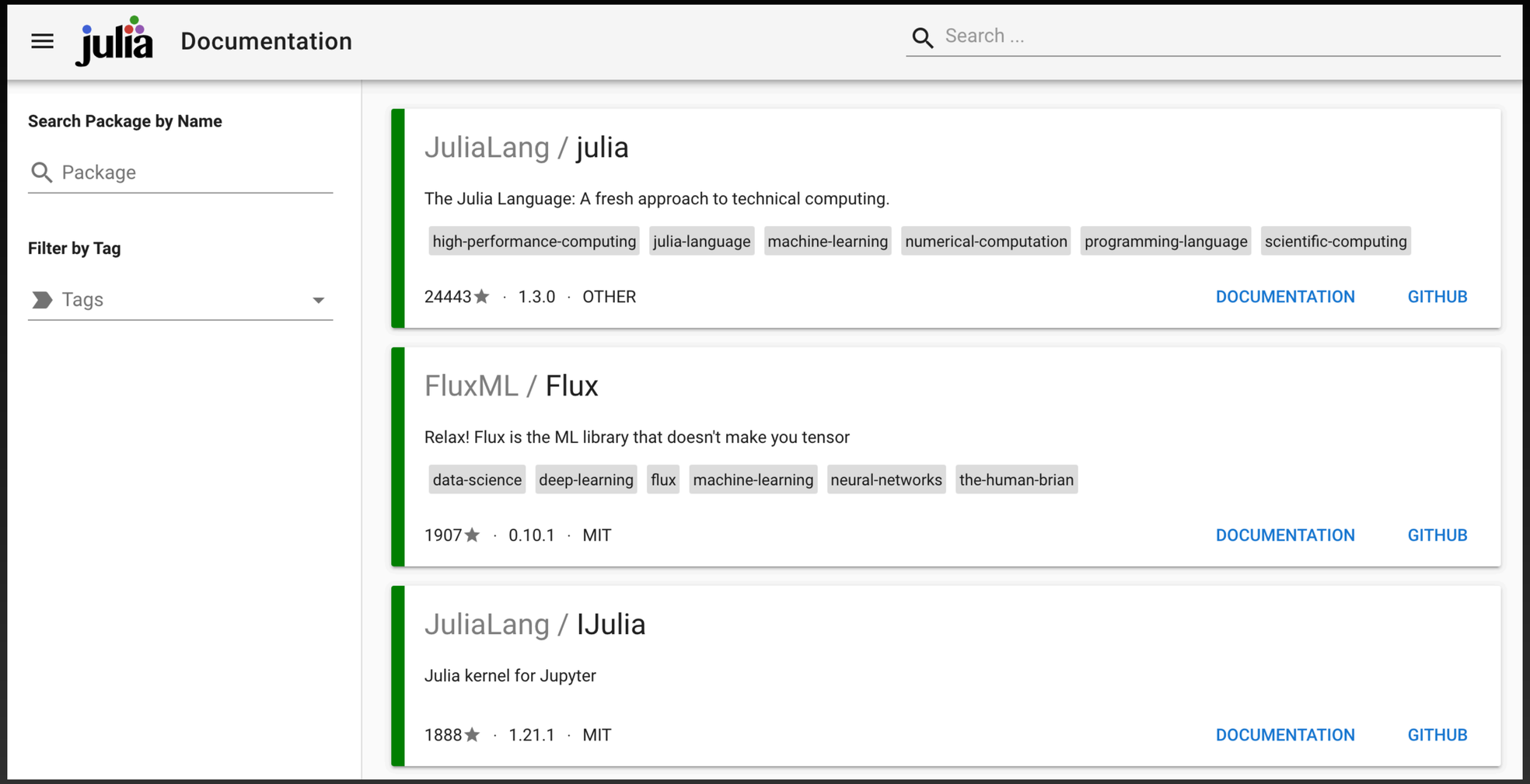
Finding Packages


JuliaHub
beta
Log In
Home
Packages
Doc Search
Code Search
JuliaHub
Track the pulse of the Julia ecosystem, find the packages you use, and discover new packages.
Log in to access additional functionality like registering packages.
Popular Packages
Flux
IJulia
Gadfly
Gen
DifferentialEquations
JuMP
Knet
Plots
Genie
Popular Tags
dynamical-systems
polynomials
clustering
lightgraphs
image-processing
neural-ode
data-structures
economics
sde
data-science
regression
arrays
gpu
differentialequations
time-series
graph
bioinformatics
ode
simulation
math
astronomy
linear-algebra
statistics
queryverse
dae
geo
finance
machine-learning
nlp
mcmc
optimization
robotics
deep-learning
differential-equations
automatic-differentiation
data
plotting
visualization
bayesian-inference
physics
biology
flux
neural-networks
floating-point
geospatial
high-performance-computing
timeseries
interpolation
econometrics
New Packages
RigidBodyTools 0.1.0
SentinelArrays 1.0.0
Recently Updated Packages
DynamicSparseArrays 0.2.4
OdsIO 0.6.0

Julia Hub

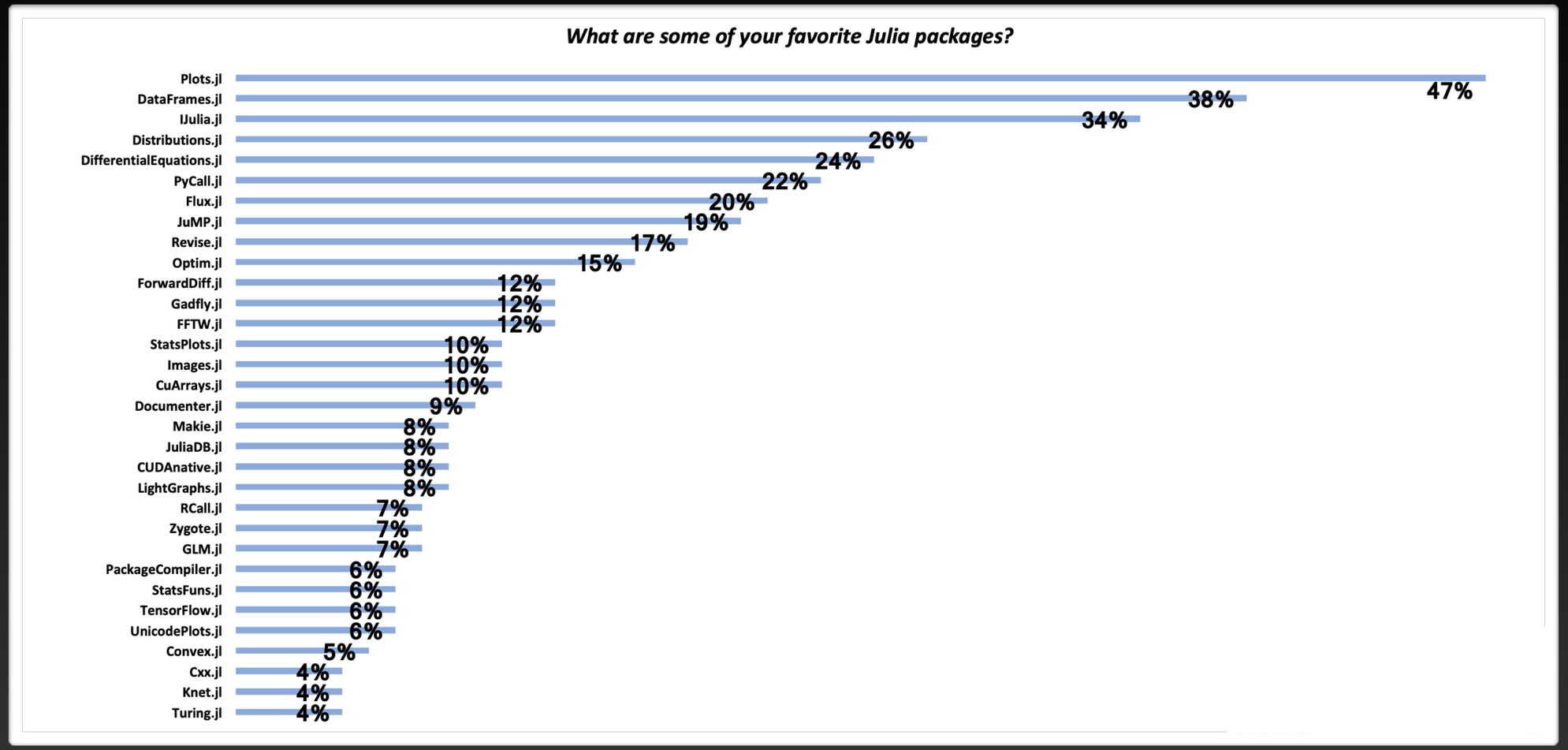


Julia Observer



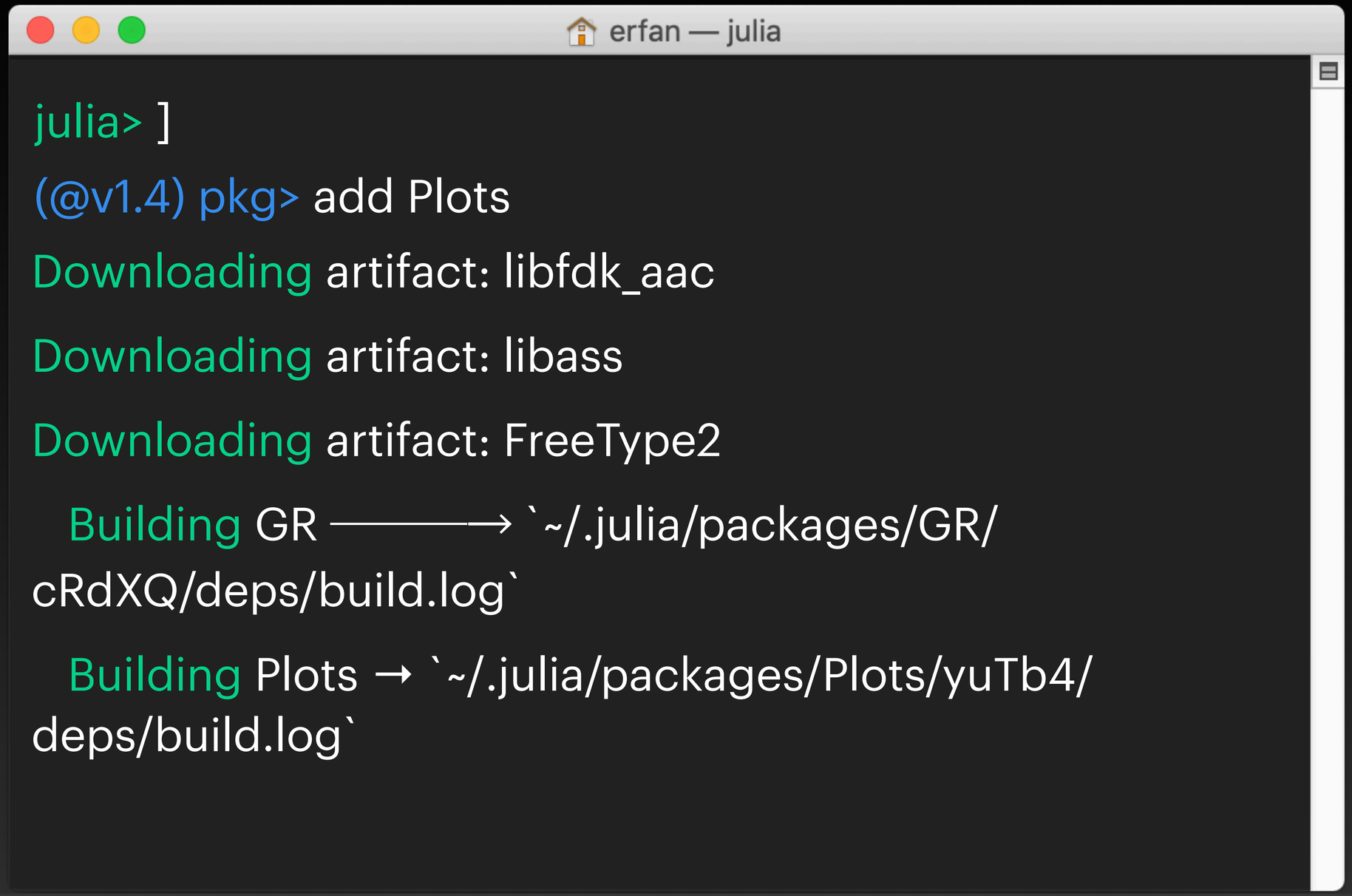
Julia Packages

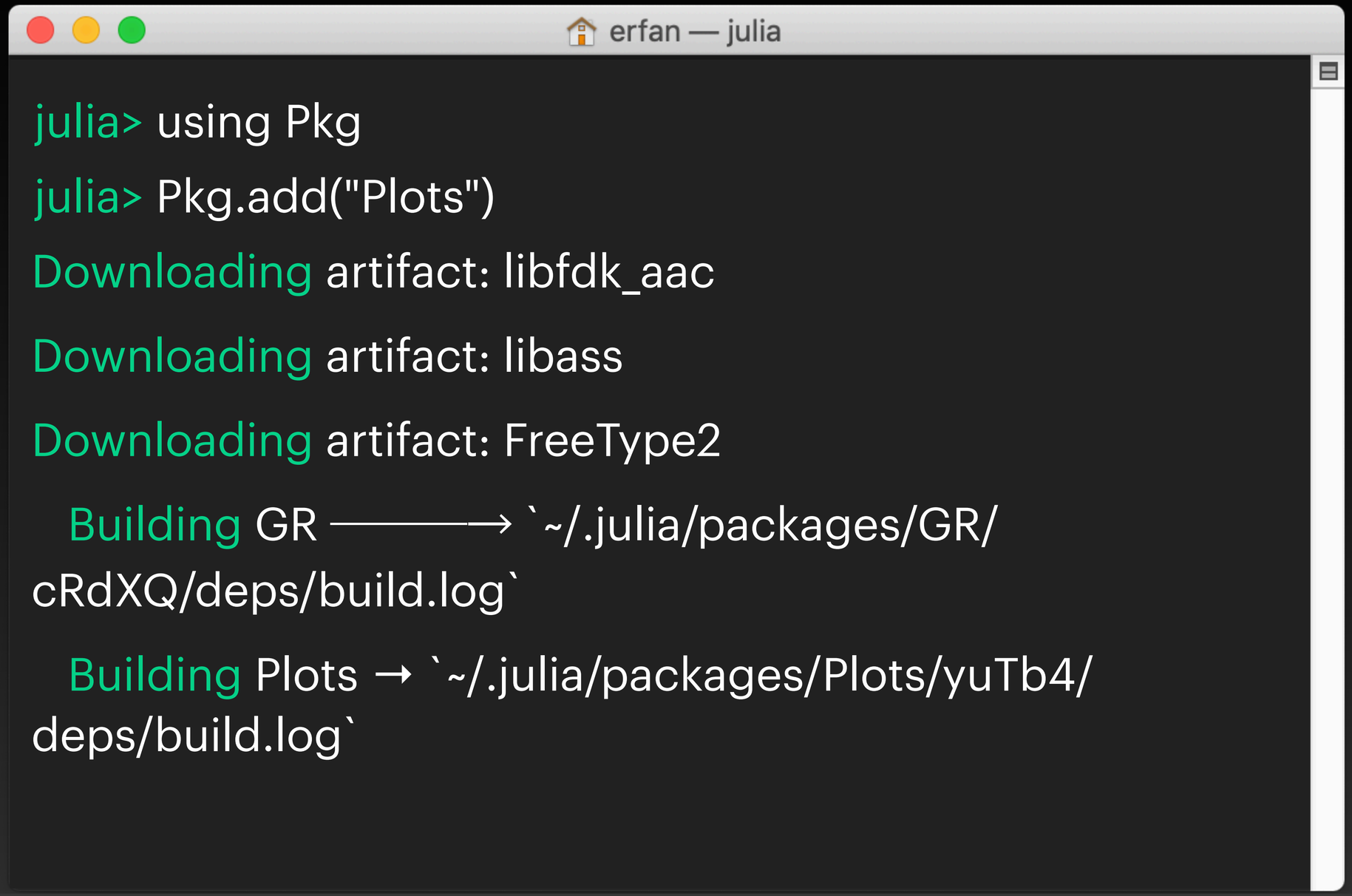
Popular Packages Among Julia Users



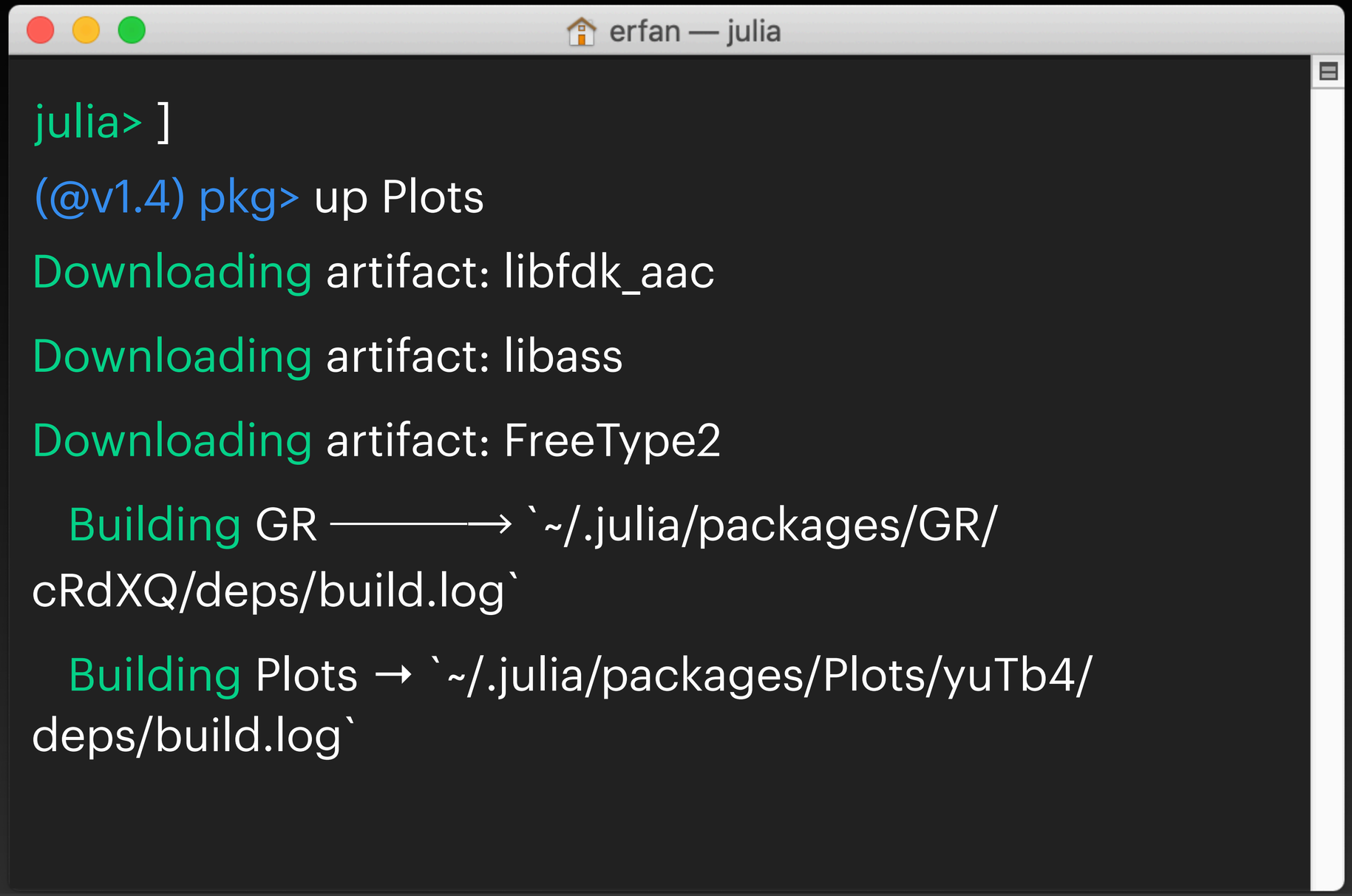
User & Developer Survey 2019, from Julia Computing, Inc.

Installing Packages

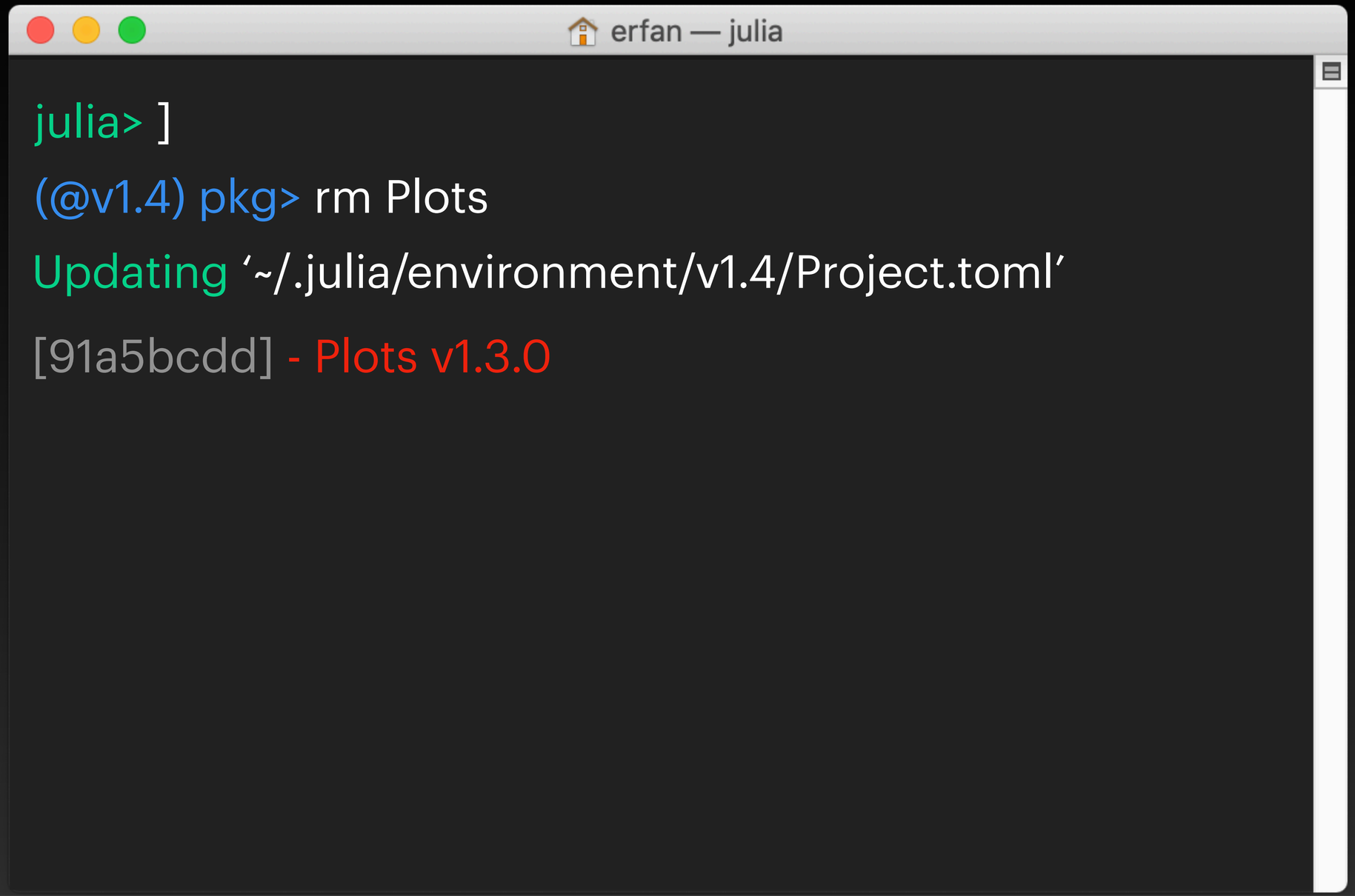

erfan — julia
julia>]
(@v1.4) pkg> add Plots
Downloading artifact: libfdk_aac
Downloading artifact: libass
Downloading artifact: FreeType2
Building GR ————→ `~/.julia/packages/GR/cRdXQ/deps/build.log`
Building Plots → `~/.julia/packages/Plots/yuTb4/deps/build.log`


erfan — julia
julia> using Pkg
julia> Pkg.add("Plots")
Downloading artifact: libfdk_aac
Downloading artifact: libass
Downloading artifact: FreeType2
Building GR ————→ `~/.julia/packages/GR/cRdXQ/deps/build.log`
Building Plots → `~/.julia/packages/Plots/yuTb4/deps/build.log`

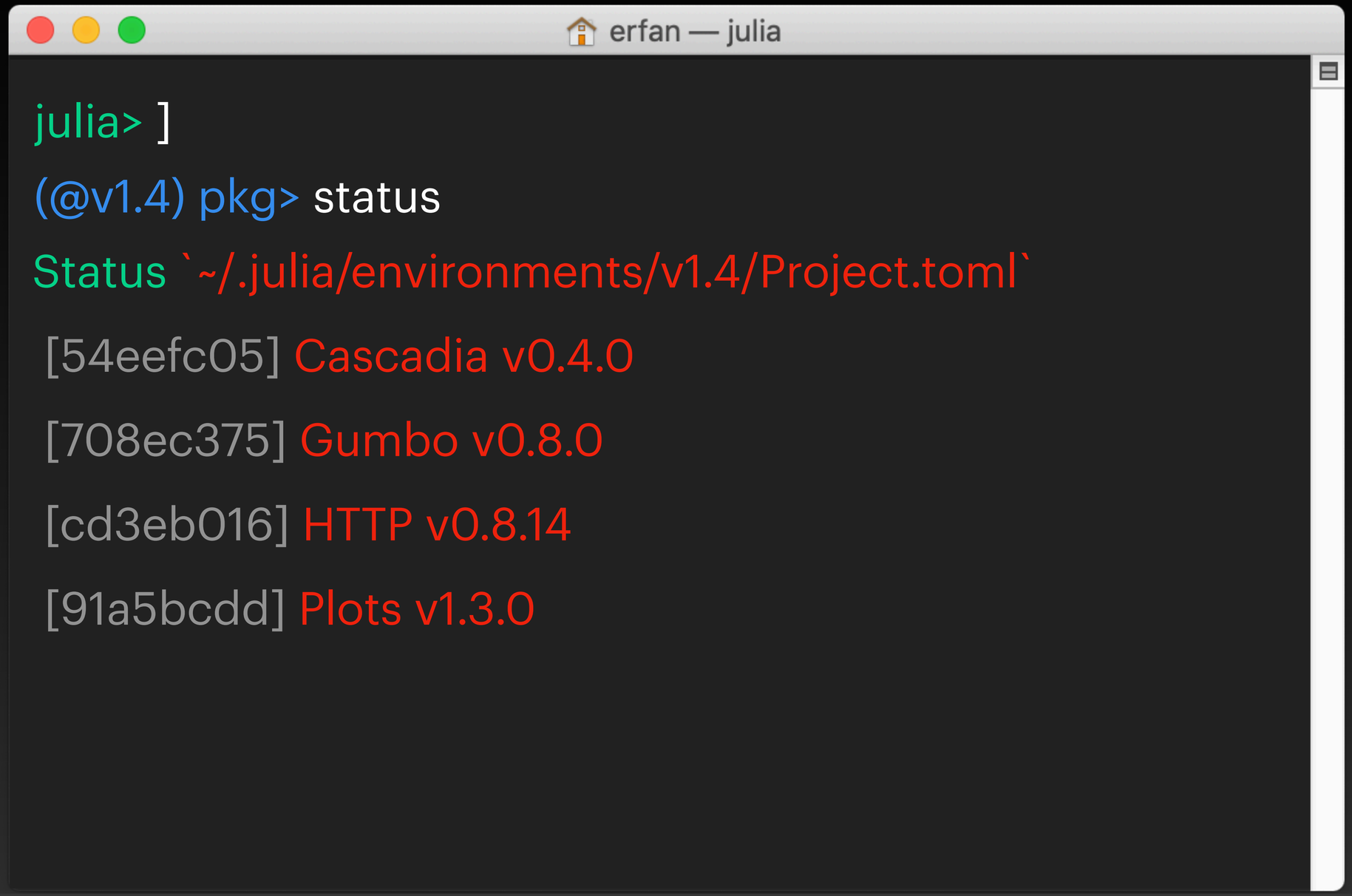
Updating Packages


erfan — julia
julia>]
(@v1.4) pkg> up Plots
Downloading artifact: libfdk_aac
Downloading artifact: libass
Downloading artifact: FreeType2
Building GR ————→ `~/.julia/packages/GR/cRdXQ/deps/build.log`
Building Plots → `~/.julia/packages/Plots/yuTb4/deps/build.log`

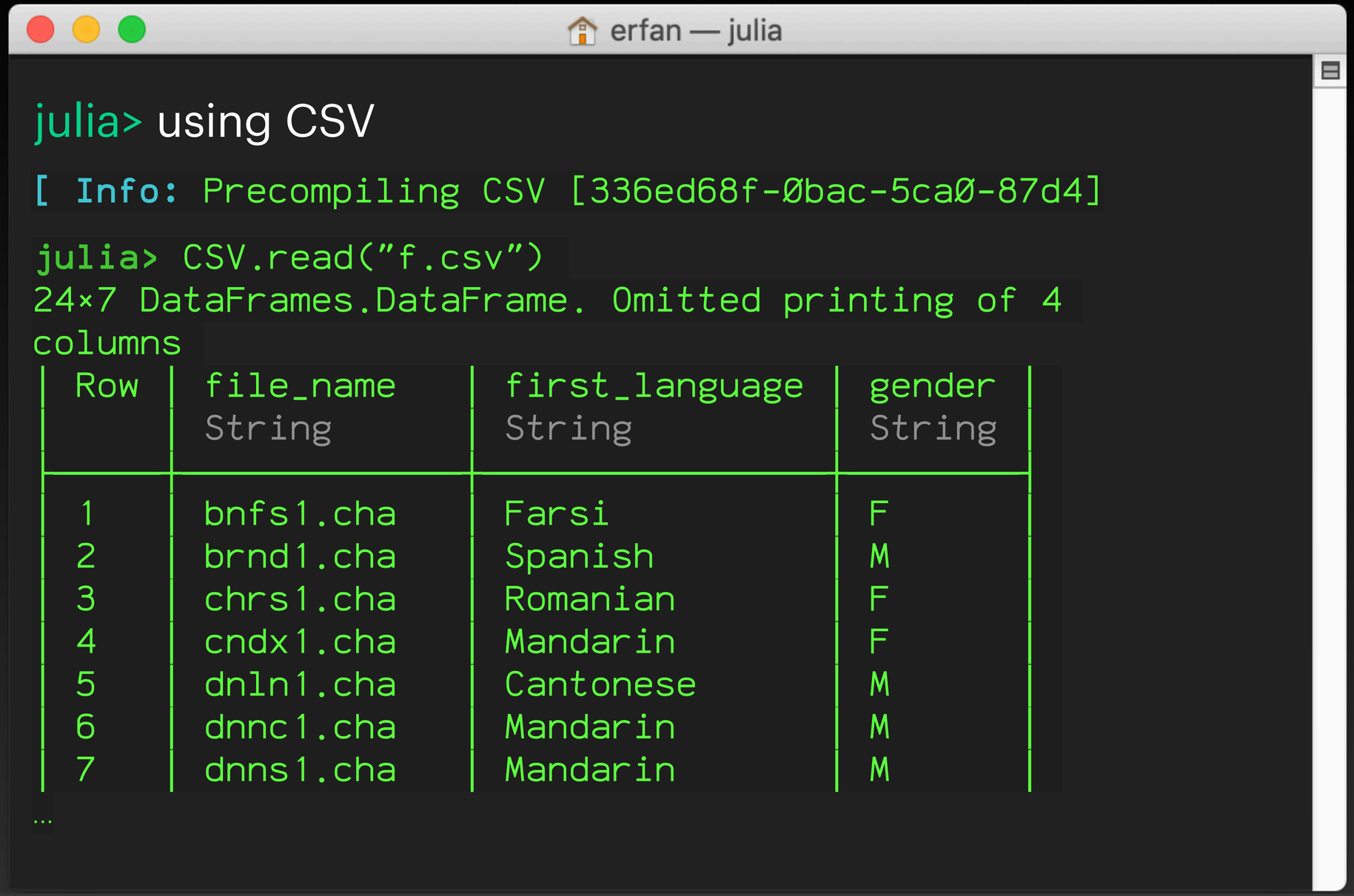
Removing Packages


erfan — julia
julia>]
(@v1.4) pkg> rm Plots
Updating '~/.julia/environment/v1.4/Project.toml'
[91a5bcdd] - Plots v1.3.0

Status of Packages


erfan — julia
julia>]
(@v1.4) pkg> status
Status `~/.julia/environments/v1.4/Project.toml`
[54eefc05] Cascadia v0.4.0
[708ec375] Gumbo v0.8.0
[cd3eb016] HTTP v0.8.14
[91a5bcdd] Plots v1.3.0

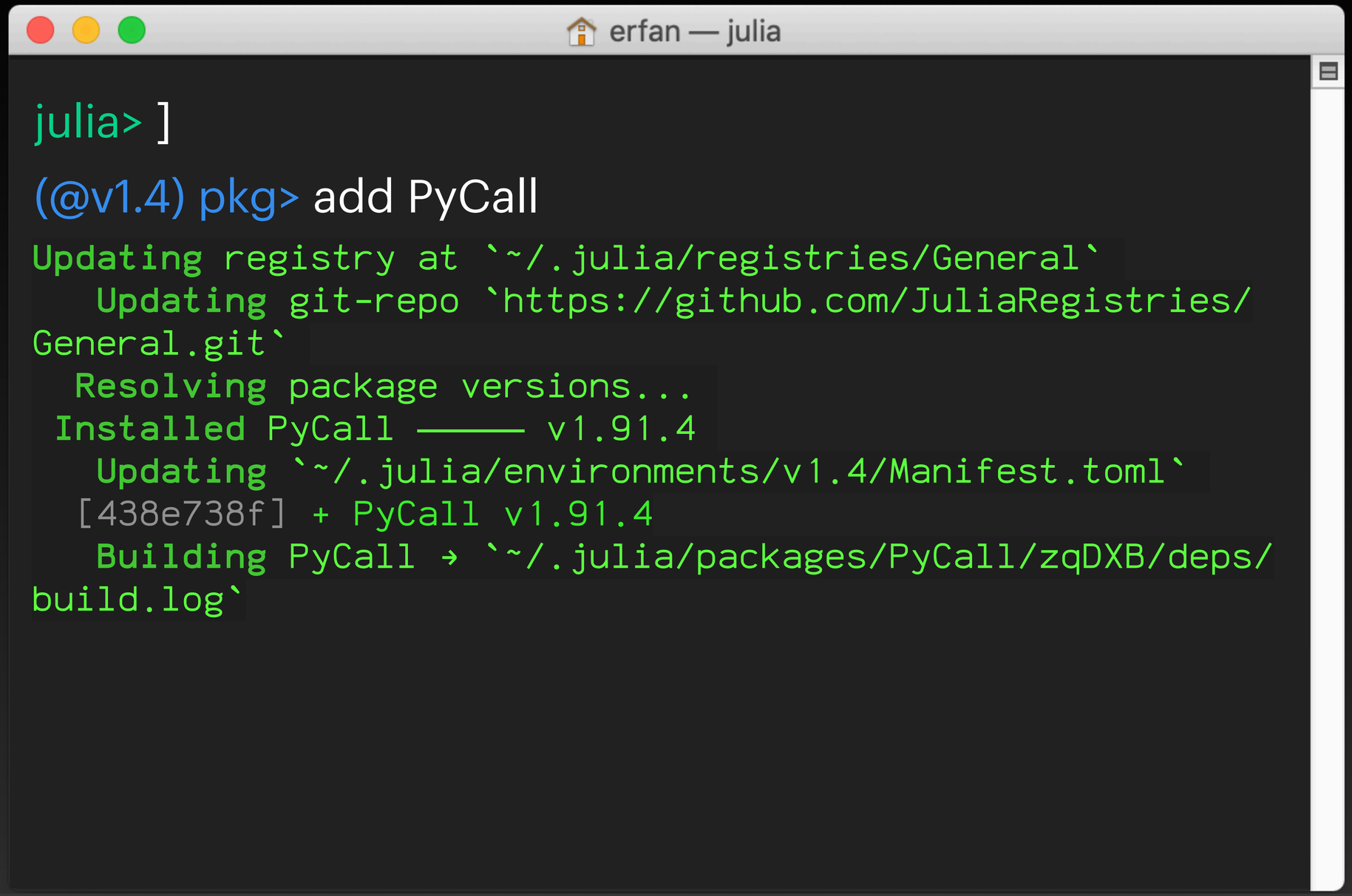
Using Packages

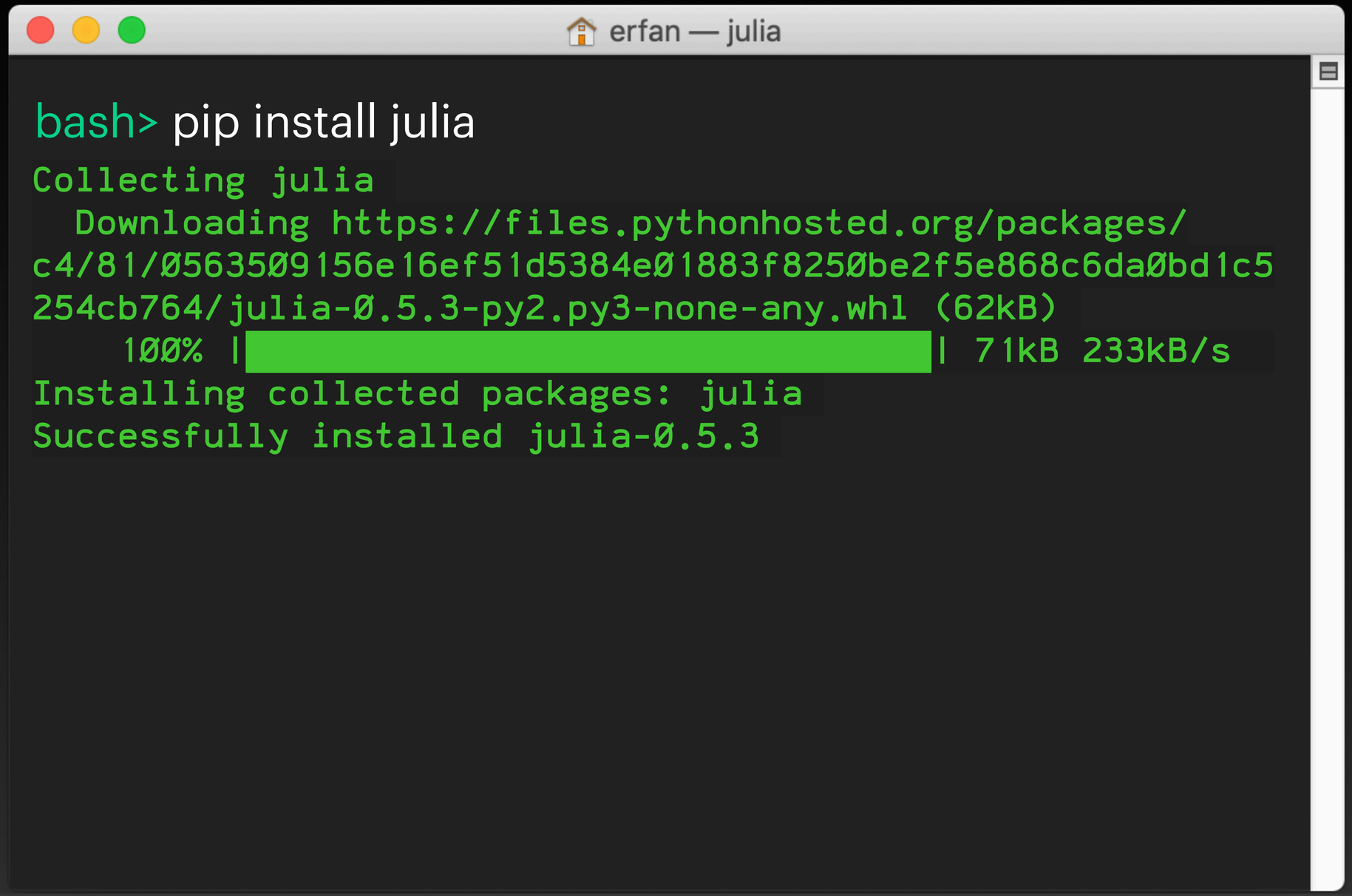

erfan — julia
julia> using CSV
[Info: Precompiling CSV [336ed68f-0bac-5ca0-87d4]
julia> CSV.read("f.csv")
24×7 DataFrames.DataFrame. Omitted printing of 4 columns
Row	file_name	first_language	gender
	String	String	String
1	bnfs1.cha	Farsi	F
2	brnd1.cha	Spanish	M
3	chrs1.cha	Romanian	F
4	cndx1.cha	Mandarin	F
5	dnln1.cha	Cantonese	M
6	dnnc1.cha	Mandarin	M
7	dnns1.cha	Mandarin	M
…

Stylistic Conventions & Tips

- Names of variables are in **lower case**. (Letter (A-Z or a-z), underscore, or a subset of Unicode code points greater than 00A0 are allowed.)
- Word separation can be indicated by underscores (‘_’) (But using it is discouraged unless the name would be hard to read otherwise)
- Names of Types and Modules begin with a **Capital Letter** and word separation is shown with **Upper Camel Case** instead of underscores.
- Names of functions and macros are in **lower case**, without underscores.
- Functions that write to their arguments have names that end in **!**. These are sometimes called "mutating" or "in-place" functions because they are intended to produce changes in their arguments after the function is called, not just return a value. (Check sample codes)

Run Julia in Python

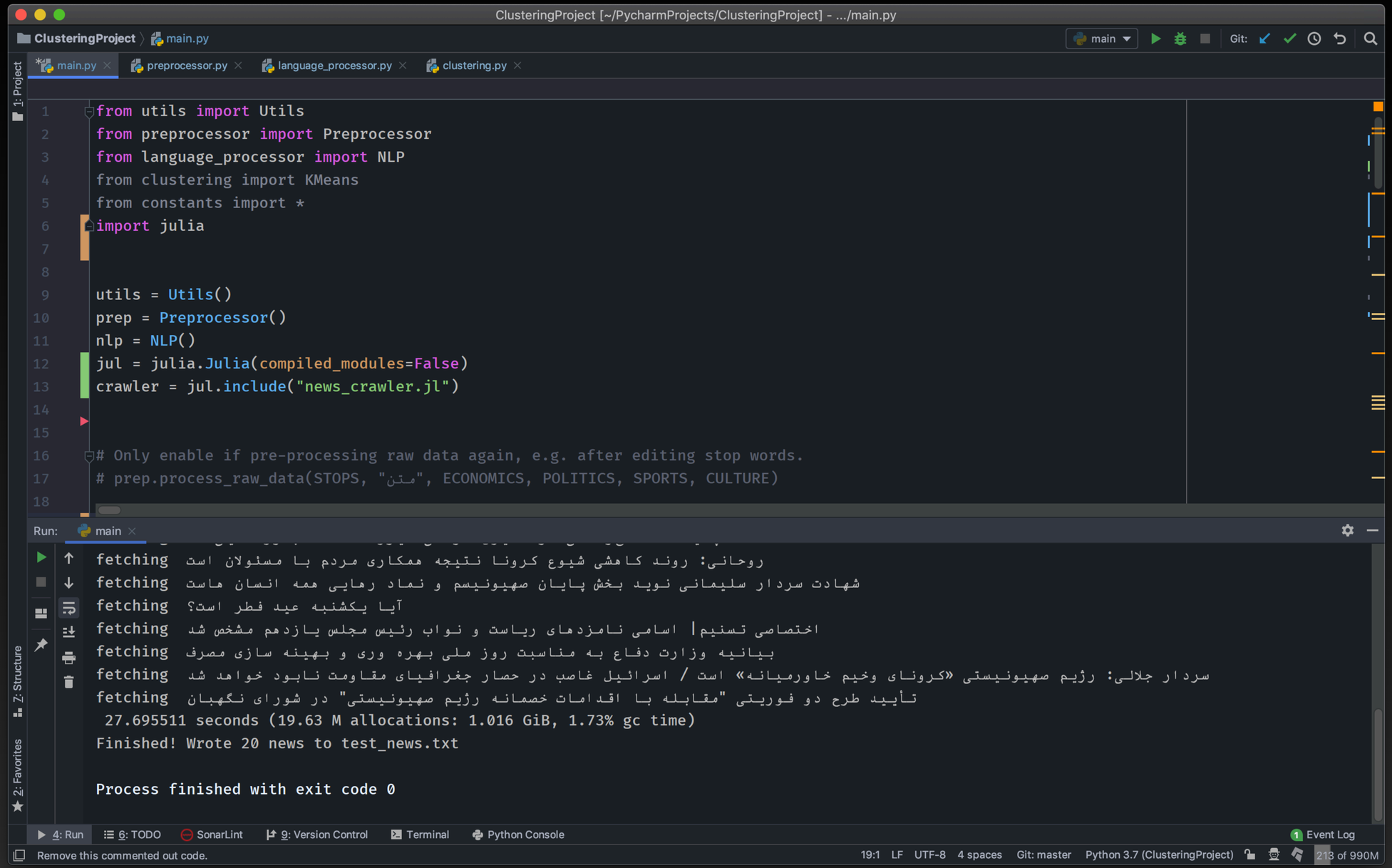

erfan — julia
julia>]
(@v1.4) pkg> add PyCall
Updating registry at `~/.julia/registries/General`
Updating git-repo `https://github.com/JuliaRegistries/General.git`
Resolving package versions...
Installed PyCall ———— v1.91.4
Updating `~/.julia/environments/v1.4/Manifest.toml`
[438e738f] + PyCall v1.91.4
Building PyCall → `~/.julia/packages/PyCall/zqDXB/deps/build.log`


erfan — julia
bash> pip install julia
Collecting julia
Downloading https://files.pythonhosted.org/packages/
c4/81/0563509156e16ef51d5384e01883f8250be2f5e868c6da0bd1c5
254cb764/julia-0.5.3-py2.py3-none-any.whl (62kB)
100% | 71kB 233kB/s
Installing collected packages: julia
Successfully installed julia-0.5.3

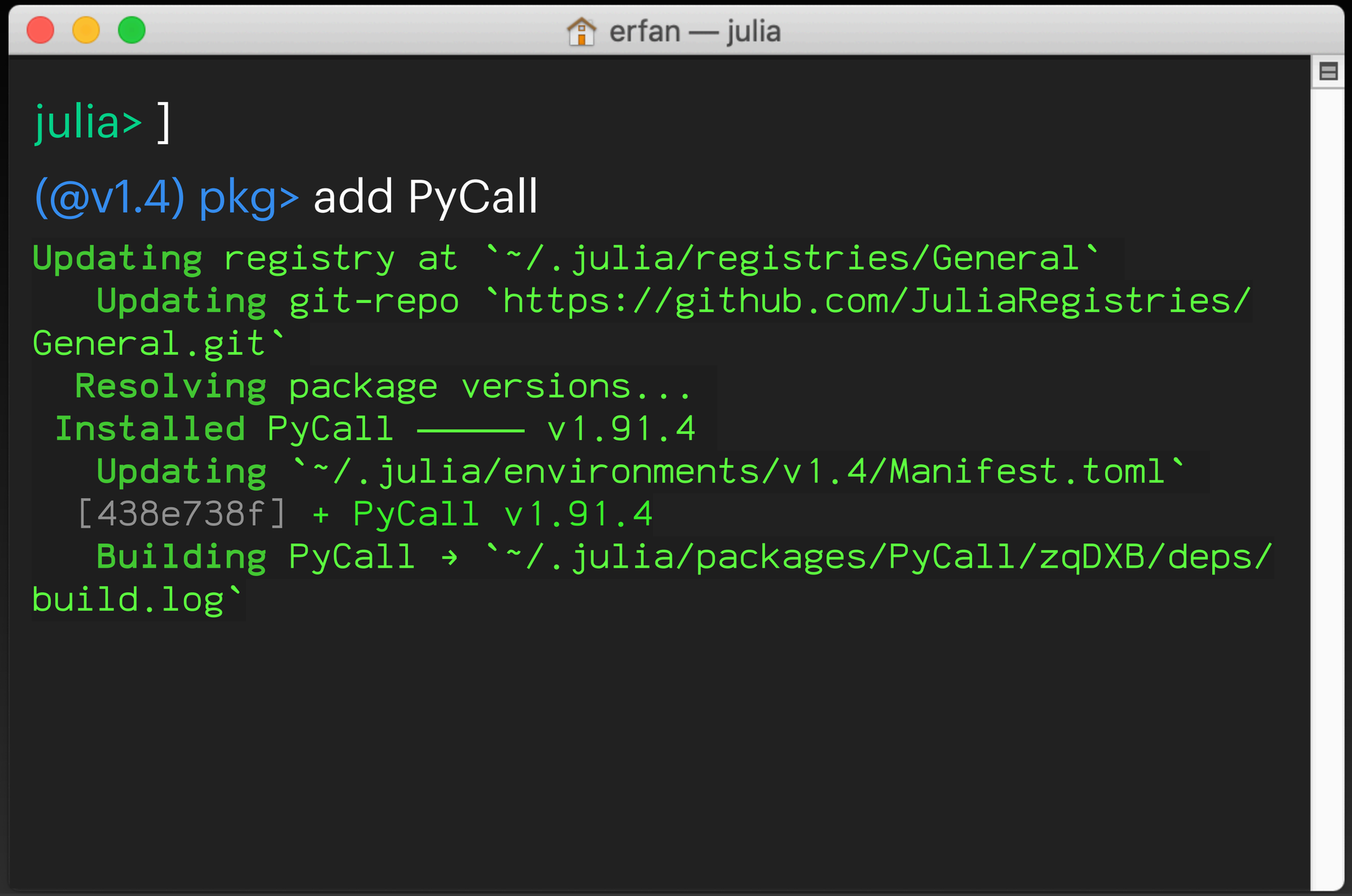
Run Julia in Python

After installing the packages, add the following lines to your Python program:

```
>>> import julia
>>> jul = julia.Julia(compiled_modules=False)
>>> jl_module = jul.include(“jl_module.jl”)
```


ClusteringProject [~/PycharmProjects/ClusteringProject] - .../main.py
ClusteringProject
main.py
main
Git:
main.py
preprocessor.py
language_processor.py
clustering.py
1: Project
from utils import Utils
from preprocessor import Preprocessor
from language_processor import NLP
from clustering import KMeans
from constants import *
import julia
utils = Utils()
prep = Preprocessor()
nlp = NLP()
jul = julia.Julia(compiled_modules=False)
crawler = jul.include("news_crawler.jl")
# Only enable if pre-processing raw data again, e.g. after editing stop words.
# prep.process_raw_data(STOPS, "متن", ECONOMICS, POLITICS, SPORTS, CULTURE)
Run:
main
fetching روحانی: روند کاهشی شیوع کرونا نتیجه همکاری مردم با مسئولان است
fetching شهادت سردار سلیمانی نوید بخش پایان صهیونیسم و نماد رهایی همه انسان هاست
fetching آیا یکشنبه عید فطر است؟
fetching اختصاصی تسنیم| اسامی نامزدهای ریاست و نواب رئیس مجلس یازدهم مشخص شد
fetching بیانیه وزارت دفاع به مناسبت روز ملی بهره وری و بهینه سازی مصرف
fetching سردار جلالی: رژیم صهیونیستی «کرونای وخیم خاورمیانه» است / اسرائیل غاصب در حصار جغرافیای مقاومت نابود خواهد شد
fetching تأیید طرح دو فوریتی "مقابله با اقدامات خصمانه رژیم صهیونیستی" در شورای نگهبان
27.695511 seconds (19.63 M allocations: 1.016 GiB, 1.73% gc time)
Finished! Wrote 20 news to test_news.txt
Process finished with exit code 0
4: Run
6: TODO
SonarLint
9: Version Control
Terminal
Python Console
Event Log
Remove this commented out code.
19:1 LF UTF-8 4 spaces Git: master Python 3.7 (ClusteringProject)
213 of 990M
2: Favorites
Z: Structure

Run Python Packages in Julia


erfan — julia
julia>]
(@v1.4) pkg> add PyCall
Updating registry at `~/.julia/registries/General`
Updating git-repo `https://github.com/JuliaRegistries/General.git`
Resolving package versions...
Installed PyCall ———— v1.91.4
Updating `~/.julia/environments/v1.4/Manifest.toml`
[438e738f] + PyCall v1.91.4
Building PyCall → `~/.julia/packages/PyCall/zqDXB/deps/build.log`

Run Python Packages in Julia

After installing the packages, add the following lines to your Julia program:

```
>>> using PyCall
>>> @pyimport requests
>>> response = requests.get(“url_to_fetch”)
```

Important Tools & Packages

Important Tools & Packages

- Revise.jl: Allows you to modify code and use the changes without restarting Julia (in the same session). Source
- Debugger.jl: A full-fledged debugger for Julia. Source (For Juno check this)
- Plots: Visualization tool for Julia. Source (If you find it too slow, precompile it using PackageCompiler.jl (Source) or use Makie.jl (Source) instead.

Further Resources

Further Resources

- Jupyter Notebooks & Sample Codes for This Lecture
- Julia Cheatsheet: A Quick Overview of Julia
- How to Add Julia to Jupyter Notebook
- Noteworthy Differences from Python
- PyCall: Run Julia in Python
- PyJulia: Run Julia in Python