



Consortium des Équipements  
de Calcul Intensif  
en Fédération Wallonie-Bruxelles

# Introduction to Scripting and Interpreted Languages

damien.francois@uclouvain.be  
Novembre 2023



## Goal of this session:

“Advocate the use of scripting languages  
(interpreted languages) alongside Fortran and C++  
and help you choose the most suitable for your needs”

# Agenda

1. Interpreters vs compilers
2. Octave, R, Python, Julia
3. Graphical User Interfaces & Literate programming
4. Additional Packages/Libraries/Modules
5. What to do when it is too slow
6. Using several of them at the same time

# Interpreters vs Compilers

- A **compiler** reads the whole (text) code and produces a separate “binary” file that can be executed by the CPU.

C/C++, Fortran, Java, Go, Rust, Haskell, ...

- An **interpreter** reads each line of code and executes it by calling the corresponding functionalities in its own code.

Bash, Python, PHP, Javascript, Ruby, ...

# Interpreters vs Compilers

The ugly truth...

- Many interpreters will pre-compile the code
- Some compilers compile not to CPU-specific machine instructions but to bytecode that is interpreted
- The bytecode interpreters sometimes re-compile the bytecode just before execution (JIT compiling)
- Interpreters exist for C and C++
- Compilers exist for Python
- The interpreter can be compiled or himself interpreted

# Interpreters vs Compilers (simplified)

## Compilers

- can apply code-wise powerful optimization
- practically have no run-time overhead

→ **Speed**

## Interpreters

- allow easy code introspection
- offer high-level language constructs and tools

→ **Ease of use**

# Interpreted languages

- Easier to **learn**
  - Many implementation details hidden
  - Can try and test code portions rapidly and easily
- Easier to **exchange**/reuse
  - The scripts are cross-platform by nearly design
  - Often built-in package management
- Faster development
  - More **convenient programming** and shorter programs
    - Offers many simplifications and shortcuts – no need to micromanage memory
    - Built-in support for mundane tasks (handle files, dates, plots, NAs, NaNs, ...)
    - Often rich standard library
  - **Easier to debug** and profile
    - GUI

1.



Why those four?



# Why those four?

- All very much used in **scientific applications**

R (S/SPPlus): strong for statistics

Octave (Matlab): strong for engineering

Python Scipy/Numpy (Canopy,Anaconda): strong for data science

Julia: strong for computational sciences

- All **free** and free.
- All now **performance**-oriented
- Fun fact: Most started as wrappers for Fortran code!

# Why those four?

By contrast,

Ruby, Perl: smaller bioinformatics-only community

Javascript, PHP, Bash, TCL, Lua: totally different goal

Matlab, IDL, Mathematica: not free

2.

# QuadQuickstart

Start the interpreter to follow along:

```
m\ spider Octave  
m\ spider SciPy-bundle  
m\ spider R  
m\ spider Julia
```

# Assignment



```
octave:1> a=1, b=2  
a = 1  
b = 2  
octave:2> a  
a = 1  
octave:3> b  
b = 2
```



```
>>> a, b = 1, 2  
>>> a  
1  
>>> b  
2
```



```
> a<-1; b<-2  
> a  
[1] 1  
> b  
[1] 2  
> 1->a; b=2
```



```
julia> a, b = 1, 2  
(1, 2)  
  
julia> a  
1  
  
julia> b  
2
```

# Operators



```
octave:11> c=[a-b, a*b, a/b, a^b]
c =
  -1.00000    2.00000    0.50000    1.00000
octave:12> c.*c
ans =
  1.00000    4.00000    0.25000    1.00000
octave:13> c*c'
ans = 6.2500
```



```
>>> import numpy as np
>>> c=np.array((a-b, a*b, a/b, a**b))
>>> c * c
array([1.   , 4.   , 0.25, 1.   ])
>>> c @ c
6.25
```



```
> c=c(a-b, a*b, a/b, a^b)
> c
[1] -1.0  2.0  0.5  1.0
> c*c
[1] 1.00 4.00 0.25 1.00
> c%%c
      [,1]
[1,] 6.25
```



```
julia> using LinearAlgebra

julia> c=(a-b, a*b, a/b, a^b)
(-1, 2, 0.5, 1)

julia> c.*c
(1, 4, 0.25, 1)

julia> c·c # type \cdot<TAB>
6.25
```

# Sequences



```
octave:15> b=1:9  
b =  
  
    1    2    3    4    5    6    7    8    9
```



```
>>> import numpy as np  
>>> b = np.arange(1,10)  
>>> b  
array([1, 2, 3, 4, 5, 6, 7, 8, 9])
```



```
> b <- 1:9  
> b  
[1] 1 2 3 4 5 6 7 8 9
```



```
julia> b = 1:9  
1:9  
  
julia> collect(b)  
9-element Vector{Int64}:  
 1  
 2  
 3  
 [...]
```

# Matrices



```
octave:16> a=reshape(1:9, 3, 3)
a =
```

```
 1  4  7
 2  5  8
 3  6  9
```



```
>>> a = np.arange(1,10).reshape(3,3)
>>> a
array([[1, 2, 3],
       [4, 5, 6],
       [7, 8, 9]])
```



```
> a=array(1:9, dim=c(3,3))
> a
      [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
```



```
julia> a=reshape(1:9,3,3)
3×3 reshape(::UnitRange{Int64}, 3, 3)
with eltype Int64:
 1  4  7
 2  5  8
 3  6  9
```

# Matrices



```
octave:3> zeros(3,2)
ans =
```

```
  0  0
  0  0
  0  0
```

```
octave:4> ones(2,3)
```



```
>>> np.zeros((3,2))
array([[0., 0.],
       [0., 0.],
       [0., 0.]])
>>> np.ones((3,2))
array([[1., 1.],
       [1., 1.],
       [1., 1.]])
```



```
> matrix(0,3,2)
      [,1] [,2]
[1,]    0    0
[2,]    0    0
[3,]    0    0
> matrix(1,2,3)
      [,1] [,2] [,3]
[1,]    1    1    1
[2,]    1    1    1
```



```
julia> zeros(3,2)
3×2 Matrix{Float64}:
 0.0  0.0
 0.0  0.0
 0.0  0.0
```

```
julia> ones(2,3)
2×3 Matrix{Float64}:
 1.0  1.0  1.0
 1.0  1.0  1.0
```



# Slicing



```
octave:18> a(2,3)
ans = 8
octave:19> a(1,:)
ans =

    1    4    7

octave:20> a(:,1)
```



```
>>> a[2,1]
8
>>> a[1,:]
array([4, 5, 6])
>>> a[0,:]
array([1, 2, 3])
>>>
```



```
> a[2,3]
[1] 8
> a[1,]
[1] 1 4 7
> a[,1]
[1] 1 2 3
```



```
julia> a[2,3]
8

julia> a[1,:]
3-element Vector{Int64}:
 1
 4
 7
```

# Slicing



```
octave:27> b(1:2:end)
ans =
```

```
1 3 5 7 9
```

```
octave:28> b(2:2:end-1)
ans =
```

```
2 4 6 8
```



```
>>> b[::2]
Array([1, 3, 5, 7, 9])
>>> b[1::2]
array([2, 4, 6, 8])
```



```
> b[seq(1,9,2)]
[1] 1 3 5 7 9
> b[seq(2,8,2)]
[1] 2 4 6 8
```



```
julia> collect(b[1:2:end])
5-element Vector{Int64}:
 1
 3
 5
 7
 9
```

```
julia> collect(b[2:2:end-1])
```

# Slicing



```
octave:36> b>5
ans =

    0    0    0    0    0    1    1    1    1
```

```
octave:37> find(b>5)
ans =
```

```
    6    7    8    9
```



```
>>> b>5
array([False, False, False, False,
       False,  True,  True,  True,  True])
>>> np.where(b>5)
(array([5, 6, 7, 8]),)
```



```
> b>5
[1] FALSE FALSE FALSE FALSE FALSE
    TRUE  TRUE  TRUE  TRUE
> which(b>5)
[1] 6 7 8 9
```



```
julia> b.>5
9-element BitVector:
 0
 0
 [...]
 1
 1
```

```
julia> findall(b.>5)
```

# Loops



```
octave:38> for i=1:5
> i*i
> end
ans = 1
ans = 4
ans = 9
ans = 16
ans = 25
```



```
>>> for i in range(1,6):
...     print(i*i)
...
1
4
9
16
25
```



```
> for(i in 1:5) {
+ print(i*i)
+ }
[1] 1
[1] 4
[1] 9
[1] 16
[1] 25
```



```
julia> for i=1:5
        println(i*i)
        end
1
4
9
16
25
```

# Conditionals



```
octave:40> if 2>1 ;  
> disp("OK")  
> end  
OK
```



```
> if (2>1) {  
+ print('OK')  
+ }  
[1] "OK"
```



```
>>> if 2>1:  
...     print("OK")  
...  
OK
```



```
julia> if 2>1 ;  
        println("OK")  
        end  
OK
```

# Strings



```
octave:18> n = str2double("10")
n = 10
octave:19> ischar(n)
ans = 0
octave:20> isfloat(n)
ans = 1
octave:21> printf('Number=%d\n', n)
Number=10
```



```
>>> n = float("10")
>>> type(n)
<class 'float'>
>>> print("Number=%d" % n)
Number=10
>>> print("Number={:.0f}".format(n))
Number=10
>>> print(f"Number={n:.0f}")
Number=10
```



```
> n <- as.numeric("10")
> cat(sprintf("Number=%d\n", n))
Number=10
> library(glue)
> glue("Number={n}")
Number=10
```



```
julia> n=parse(Int, "10")
10
julia> print("Number=$n")
Number=10
julia> using Printf
julia> @printf("Number=%.0f", n)
Number=10
```

# Special numbers



```
octave:22> nan
ans = NaN
octave:23> inf
ans = Inf
```



```
>>> np.nan
nan
>>> np.inf
Inf
>>> None
>>>
```



```
> NaN
[1] NaN
> Inf
[1] Inf
> NA
[1] NA
> NULL
NULL
```



```
julia> NaN
NaN

julia> Inf
Inf

julia> nothing
```

# More complete list

## Hyperpolyglot

### Numerical Analysis & Statistics: MATLAB, R, NumPy, Julia

a side-by-side reference sheet

**sheet one:** [grammar and invocation](#) | [variables and expressions](#) | [arithmetic and logic](#) | [strings](#) | [regexes](#) | [dates and time](#) | [tuples](#) | [arrays](#) | [arithmetic sequences](#) | [2d arrays](#) | [3d arrays](#) | [dictionaries](#) | [functions](#) | [execution control](#) | [file handles](#) | [directories](#) | [processes and environment](#) | [libraries and namespaces](#) | [reflection](#) | [debugging](#)

**sheet two:** [tables](#) | [import and export](#) | [relational algebra](#) | [aggregation](#)

[vectors](#) | [matrices](#) | [sparse matrices](#) | [optimization](#) | [polynomials](#) | [descriptive statistics](#) | [distributions](#) | [linear regression](#) | [statistical tests](#) | [time series](#) | [fast fourier transform](#) | [clustering](#) | [images](#) | [sound](#)

[bar charts](#) | [scatter plots](#) | [line charts](#) | [surface charts](#) | [chart options](#)

	<a href="#">matlab</a>	<a href="#">r</a>	<a href="#">numpy</a>	<a href="#">julia</a>
<a href="#">version used</a>	MATLAB 8.3 Octave 3.8	3.1	Python 2.7 NumPy 1.7 SciPy 0.13 Pandas 0.12 Matplotlib 1.3	0.4
<a href="#">show version</a>	\$ matlab -nojvm -nodisplay -r 'exit' \$ octave --version	\$ R --version	sys.version np.__version__ sp.__version__ mpl.__version__	\$ julia --version
<a href="#">implicit prologue</a>	none	install.packages('ggplot2') library('ggplot2')	import sys, os, re, math import numpy as np import scipy as sp import scipy.stats as stats import pandas as pd import matplotlib as mpl import matplotlib.pyplot as plt	
<a href="#">grammar and invocation</a>				
	<a href="#">matlab</a>	<a href="#">r</a>	<a href="#">numpy</a>	<a href="#">julia</a>
<a href="#">interpreter</a>	\$ cat >>foo.m 1 + 1 exit \$ matlab -nojvm -nodisplay -r "run('foo.m')" \$ octave foo.m	\$ cat >>foo.r 1 + 1 \$ Rscript foo.r \$ R -f foo.r	\$ cat >>foo.py print(1 + 1) \$ python foo.py	\$ cat >>foo.jl println(1 + 1) \$ julia foo.jl
<a href="#">repl</a>	\$ matlab -nojvm -nodisplay \$ octave	\$ R	\$ python	\$ julia
<a href="#">command line program</a>	\$ matlab -nojvm -nodisplay -r 'disp(1 + 1); exit' \$ octave --silent --eval '1 + 1'	\$ Rscript -e 'print("hi")'	python -c 'print("hi")'	\$ julia -e 'println("hi")'
<a href="#">block delimiters</a>	function end if elseif else end while end for end	{ }	offside rule	



3.

## Graphical User Interfaces

Editing, debugging, accessing the doc, made easy

## Literate programming

Authoring dynamic documents with code in them

# Octave

The image displays the Octave software interface, which is divided into several panes. On the left, the 'File Browser' shows the current directory as 'C:\Users\Mike Croucher\octave\_test' and lists a file named 'mysinc.m'. The 'Command Window' on the right contains the following text:

```
warning: function .\info.m shadows a core library function
GNU Octave, version 3.8.0
Copyright (C) 2013 John W. Eaton and others.
This is free software; see the source code for copying on
There is ABSOLUTELY NO WARRANTY; not even for MERCHANTABILITY
FITNESS FOR A PARTICULAR PURPOSE. For details, type 'warr
Octave was configured for "i686-pc-mingw32".
Additional information about Octave is available at http://
e.org.
Please contribute if you find this software useful.
For more information, visit http://www.octave.org/get-info
Read http://www.octave.org/bugs.html to learn how to submit
rts.
For information about changes from previous versions, type
>> cd octave_test/
>> edit mysinc.m
>> mysinc
>> a=rand(2000);
>> b=rand(2000);
>> tic;c=a*b;toc
Elapsed time is 5.44231 seconds.
>> |
```

At the bottom of the Octave window, the 'Workspace' pane shows a table of variables:

Name	Class	Dimension	Value	Storage Class
R	double	33x33	[11.314 10.966, ...	
X	double	33x33	[-8 -7.5000 -7, ...	
Y	double	33x33	[-8 -8 -8 -8, ...	
Z	double	33x33	[-0.083953 -0.0, ...	
a	double	2000x2000	[0.17387 0.4938, ...	
b	double	2000x2000	[0.89777 0.4025, ...	
c	double	2000x2000	[501.94 497.51, ...	

On the right, the 'Editor' window shows the code for 'mysinc.m':

```
1 [X,Y] = meshgrid(-8:5:8);
2 R = sqrt(X.^2 + Y.^2);
3 Z = sin(R) ./ R;
4 mesh(X,Y,Z)
```

Below the code, a 3D plot titled 'Figure 1' is displayed. The plot shows a surface with a central peak and a surrounding valley, rendered in a color gradient from blue to red. The axes are labeled with values from -10 to 10. The plot is shown in a window titled 'Figure 1' with a 'File Edit' menu.

# Rstudio (also for Python and Julia with some adaptations)

The screenshot displays the RStudio environment with the following components:

- Source Editor:** Contains R code for loading data, summarizing it, and creating a scatter plot.
- Console:** Shows the execution output of the code, including summary statistics for 'x', 'y', and 'z' variables, and the execution of the plot creation commands.
- Workspace:** Lists the loaded data object 'diamonds' (53940 observations) and the 'p' plot object.
- Plots Panel:** Displays a scatter plot titled 'Diamond Pricing' showing Price vs. Carat, with points colored by Clarity.

```
1 library(ggplot2)
2 source("plots/formatPlot.R")
3
4 view(diamonds)
5 summary(diamonds)
6
7 summary(diamonds$price)
8 aveSize <- round(mean(diamonds$carat), 4)
9 clarity <- levels(diamonds$clarity)
10
11 p <- qplot(carat, price,
12            data=diamonds, color=clarity,
13            xlab="Carat", ylab="Price",
14            main="Diamond Pricing")
15
```

Console Output:

```
      x      y      z
Min.  : 0.000 Min.  : 0.000 Min.  : 0.000
1st Qu.: 4.710 1st Qu.: 4.720 1st Qu.: 2.910
Median : 5.700 Median : 5.710 Median : 3.530
Mean   : 5.731 Mean   : 5.735 Mean   : 3.539
3rd Qu.: 6.540 3rd Qu.: 6.540 3rd Qu.: 4.040
Max.   :10.740 Max.   :58.900 Max.   :31.800
> summary(diamonds$price)
  Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  326   950   2401   3933   5324  18820
> aveSize <- round(mean(diamonds$carat), 4)
> clarity <- levels(diamonds$clarity)
> p <- qplot(carat, price,
+           data=diamonds, color=clarity,
+           xlab="Carat", ylab="Price",
+           main="Diamond Pricing")
>
> format.plot(p, size=24)
>
```

Workspace Data:

Variable	Value
diamonds	53940 obs. of 10 variables
aveSize	0.7979
clarity	character [8]
p	ggplot [8]

Plots Panel Legend:

Clarity	Color
I1	Red
SI2	Orange
SI1	Yellow
VS2	Green
VS1	Cyan
VVS2	Blue
VVS1	Purple
IF	Pink

# Spyder

The screenshot displays the Spyder Python IDE interface. The main window is titled "Editor - C:\Users\Nick\Documents\School\spyder\special2.py". The code editor contains the following Python code:

```
1 # -*- coding: utf-8 -*-
2 """
3 Spyder Editor
4
5 This temporary script file is located here:
6 C:\Users\Nick\.spyder2\.temp.py
7 """
8
9 from numpy import *
10 from scipy import *
11 from scipy import eye
12 from scipy.integrate import odeint
13 import pylab
14
15 #Load data file
16 free_response = loadtxt("free_response.lvm")
17
18 #delete first few lines, adjust time vector back to zero
19 free_response = delete(free_response, linspace(0,20,20),0)
20 free_response[:,0]=free_response[:,0]-min(free_response[:,0])
21
22 #take numerical derivative
23 time = free_response[:,0]
24 pos = free_response[:,1]
25 vel = diff(pos)/diff(time)
26 time = delete(time,-1)
27 accel = diff(vel)/diff(time)
28
29 #resize vectors so they match up nicely
30 time = delete(time,-1)
31 vel = delete(vel,-1)
32 pos = delete(pos, [pos.size-1, pos.size-2], None)
33
34 #Least-squares fit to find parameters
35 #A is matrix with velocity and position
36 #b is vector of acceleration
37 A = vstack((vel,pos))
```

The Object Inspector panel on the right shows the function `delete(arr, obj, axis=None)` from the `numpy.lib.function_base` module. It provides a description: "Return a new array with sub-arrays along an axis deleted." and lists parameters: `arr` (array\_like), `obj` (slice, int or array of ints), and `axis` (int, optional). The Returns section indicates it returns an `ndarray`.

The Console panel at the bottom shows the IPython 1 environment with the following output:

```
Python 2.6.6 (r266:84297, Aug 24 2010, 18:46:32) [MSC v.1500 32 bit (Intel)]
Type "copyright", "credits" or "license" for more information.

IPython 0.10.1 -- An enhanced Interactive Python.
? -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help -> Python's own help system.
object? -> Details about 'object'. ?object also works, ?? prints more.

Welcome to pylab, a matplotlib-based Python environment.
For more information, type 'help(pylab)'.

In [1]: |
```

The status bar at the bottom indicates: Permissions: RW, End-of-lines: CRLF, Encoding: UTF-8, Line: 19, Column: 1.

# Julia for Visual Studio Code (also for R & Python)

The screenshot displays the Visual Studio Code interface with the Julia extension. The Explorer sidebar on the left shows the project structure, including the Julia workspace and a package named 'ans'. The main editor window shows a Julia script with the following code:

```
9     ...end
10     ...return maxiter
11 end mandel (generic function with 1 method)
12
13 for i in 1:10
14     ...println(i)
15 end
16
17 map
18     map
19     map!
20     mapfoldl
21     mapfoldr
22     mapreduce
23     mapslices
24     theme(:dark)
25 plot(x, y) |Plot{Plots.GRBackend() n=1}
```

A documentation popup for the `map` function is visible, showing its signature `map(f, c...) -> collection` and a description: "Transform collection `c` by applying `f` to each element. For multiple collection arguments, apply `f` elementwise." It also includes an example: `julia> map(x -> x * 2, [1, 2, 3])` resulting in `3-element Array{Int64,1}: 2`.

On the right, a plot window titled "Julia Plots (2/2)" shows a red sine wave. The x-axis ranges from approximately 3 to 9, and the y-axis ranges from 0.0 to 1.0. A legend in the top right corner identifies the line as "y1".

The bottom of the interface shows the Terminal panel with the Julia REPL prompt `julia>` and the status bar at the bottom indicating the current file is `sp/isdeflocals*` and the environment is `julia env: v1`.

# RMarkdown and KnitR

The image shows a screenshot of the RStudio interface. On the left is the source editor for a file named 'chunks.Rmd'. The code is as follows:

```
1 R Code Chunks
2 =====
3
4 With R Markdown, you can insert R code
5 chunks including plots:
6
7 ```{r qplot, fig.width=4, fig.height=3,
8 message=FALSE}
9 # quick summary and plot
10 library(ggplot2)
11 summary(cars)
12 qplot(speed, dist, data=cars) +
13   geom_smooth()
```

On the right is the 'Preview HTML' window, which displays the rendered output of the R code chunks. The title is 'R Code Chunks'. Below the title, it says 'With R Markdown, you can insert R code chunks including plots:'. The first chunk is a code block containing:

```
# quick summary and plot
library(ggplot2)
summary(cars)
```

The output of this chunk is a summary table for the 'cars' dataset:

##	speed	dist
##	Min. : 4.0	Min. : 2
##	1st Qu.:12.0	1st Qu.: 26
##	Median :15.0	Median : 36
##	Mean :15.4	Mean : 43
##	3rd Qu.:19.0	3rd Qu.: 56
##	Max. :25.0	Max. :120

The second chunk is a code block containing:

```
qplot(speed, dist, data = cars) + geom_smooth()
```

The output of this chunk is a scatter plot of 'dist' versus 'speed' with a smoothed regression line. The x-axis is labeled 'speed' and ranges from 5 to 25. The y-axis is labeled 'dist' and ranges from 0 to 100. The plot shows a positive correlation between speed and distance, with a blue smoothed line and a grey shaded confidence interval.



# Jupyter notebooks (also for Julia & R)

The screenshot displays a Jupyter Notebook interface with the following components:

- File Browser:** Shows a file named 'Untitled.ipynb'.
- Console:** Contains a list of commands such as 'Clear Cells', 'Execute Cell', and 'Interrupt Kernel'.
- Editor:** Displays Python code for plotting histograms and loading data. The code includes:

```
from matplotlib import pyplot as plt
plt.style.use('bmh')

def plot_beta_hist(a, b):
    plt.hist(beta(a, b, size=10000), histtype='stepfilled',
             bins=25, alpha=0.8, normed=True)
    return

plot_beta_hist(10, 10)
plot_beta_hist(4, 12)
plot_beta_hist(50, 12)
plot_beta_hist(6, 55)
```
- Figure:** A histogram plot showing the distribution of beta values for different parameters (10, 12, 55).
- Code:** A second code block shows the execution of a script:

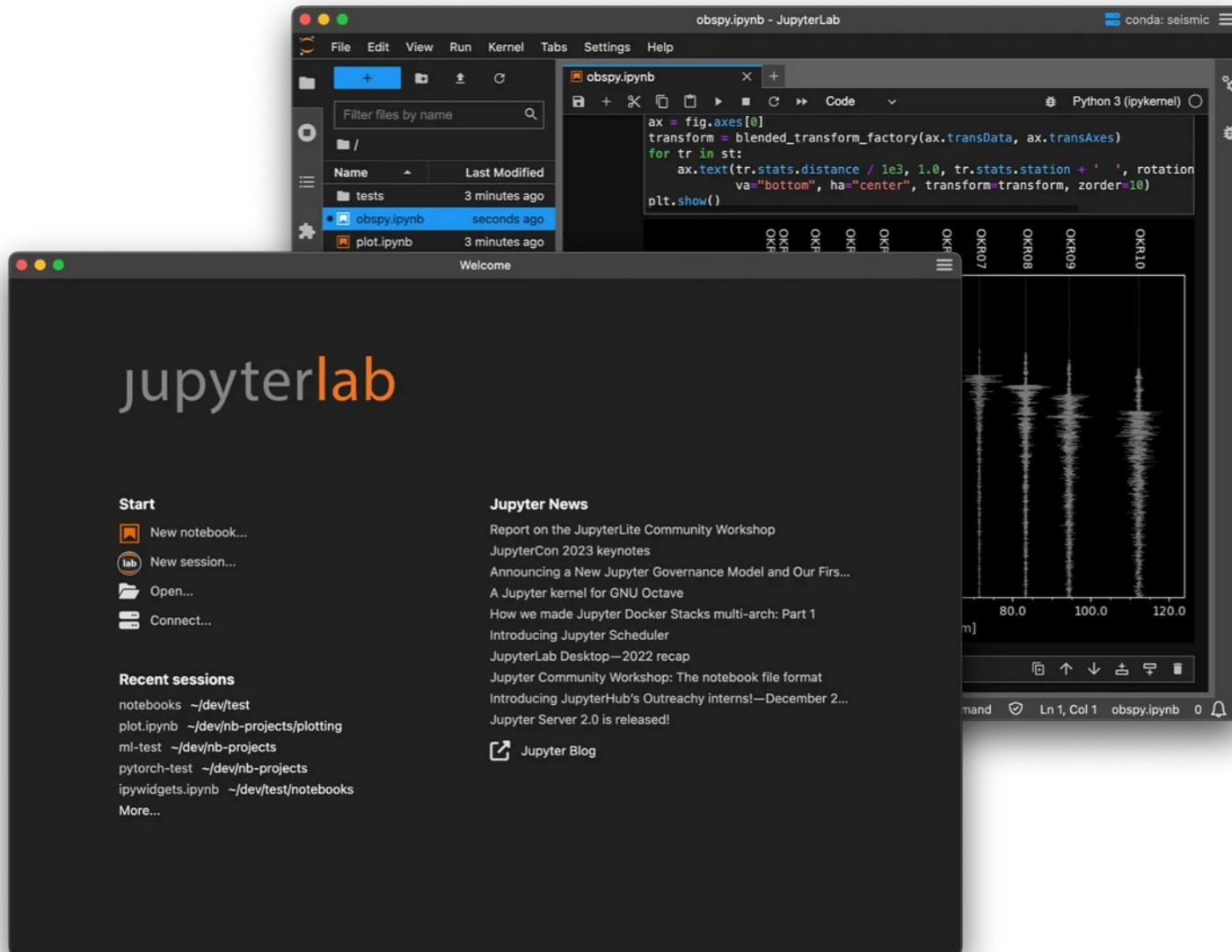
```
%run ~/Downloads/mri_with_eeg.py

loading eeg /Users/fperez/usr/conda/lib/python3.5/site-packages/mat
plotlib/mpl-data/sample_data/eeg.dat
```
- Figure:** A plot showing an MRI brain scan and an EEG waveform. The MRI scan is labeled 'MRI density' and the EEG waveform is labeled 'time (s)'. The EEG waveform shows several peaks labeled PG9, PG7, PG5, and PG3.
- Code:** A third code block shows the loading of the eeg data:

```
numSamples, numRows = 800, 4
eegfile = cbook.get_sample_data('eeg.dat', asfileobj=False)
print('loading eeg %s' % eegfile)
data = np.fromstring(open(eegfile, 'rb').read(), float)
data.shape = numSamples, numRows
t = 10.0 * np.arange(numSamples, dtype=float)/numSamples
ticklocs = []
ax = subplot(212)
xlim(0, 10)
xticks(np.arange(10))
dmin = data.min()
dmax = data.max()
dr = (dmax - dmin)*0.7 # Crowd them a bit.
y0 = dmin
y1 = (numRows - 1) * dr + dmax
ylim(y0, y1)

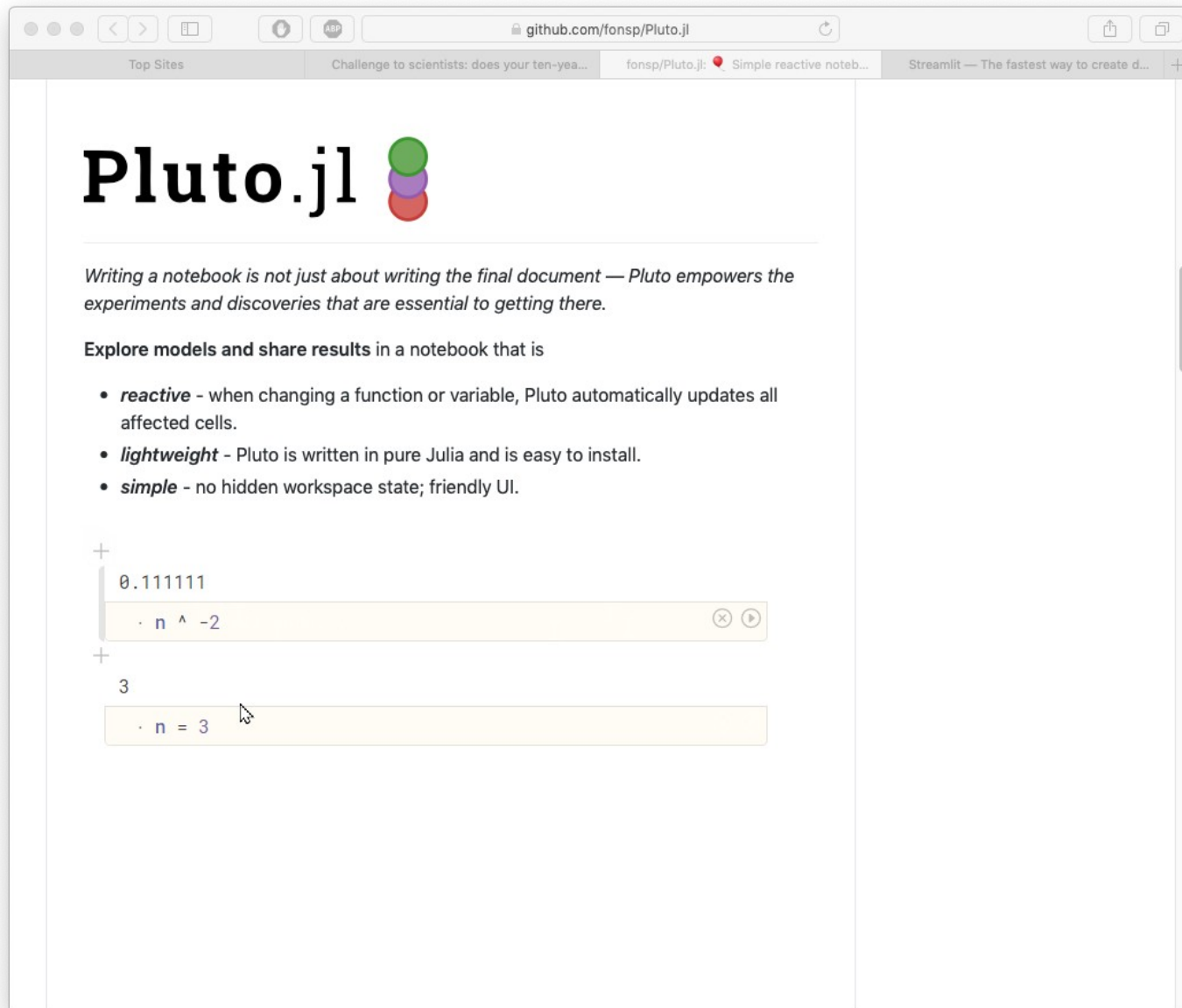
segs = []
for i in range(numRows):
```

# Jupyterlab desktop





# Julia notebooks



The screenshot shows a web browser window with the URL `github.com/fonsp/Pluto.jl`. The page features the Pluto.jl logo, which consists of the text "Pluto.jl" followed by three stacked circles in green, purple, and red. Below the logo is a paragraph of text: "Writing a notebook is not just about writing the final document — Pluto empowers the experiments and discoveries that are essential to getting there." This is followed by a sub-heading "Explore models and share results in a notebook that is" and a bulleted list of features: "reactive", "lightweight", and "simple". At the bottom of the page, there are two interactive code cells. The first cell shows the expression `n^-2` and the result `0.111111`. The second cell shows the expression `n = 3` and the result `3`. The browser's address bar and several tabs are visible at the top of the window.

## Pluto.jl

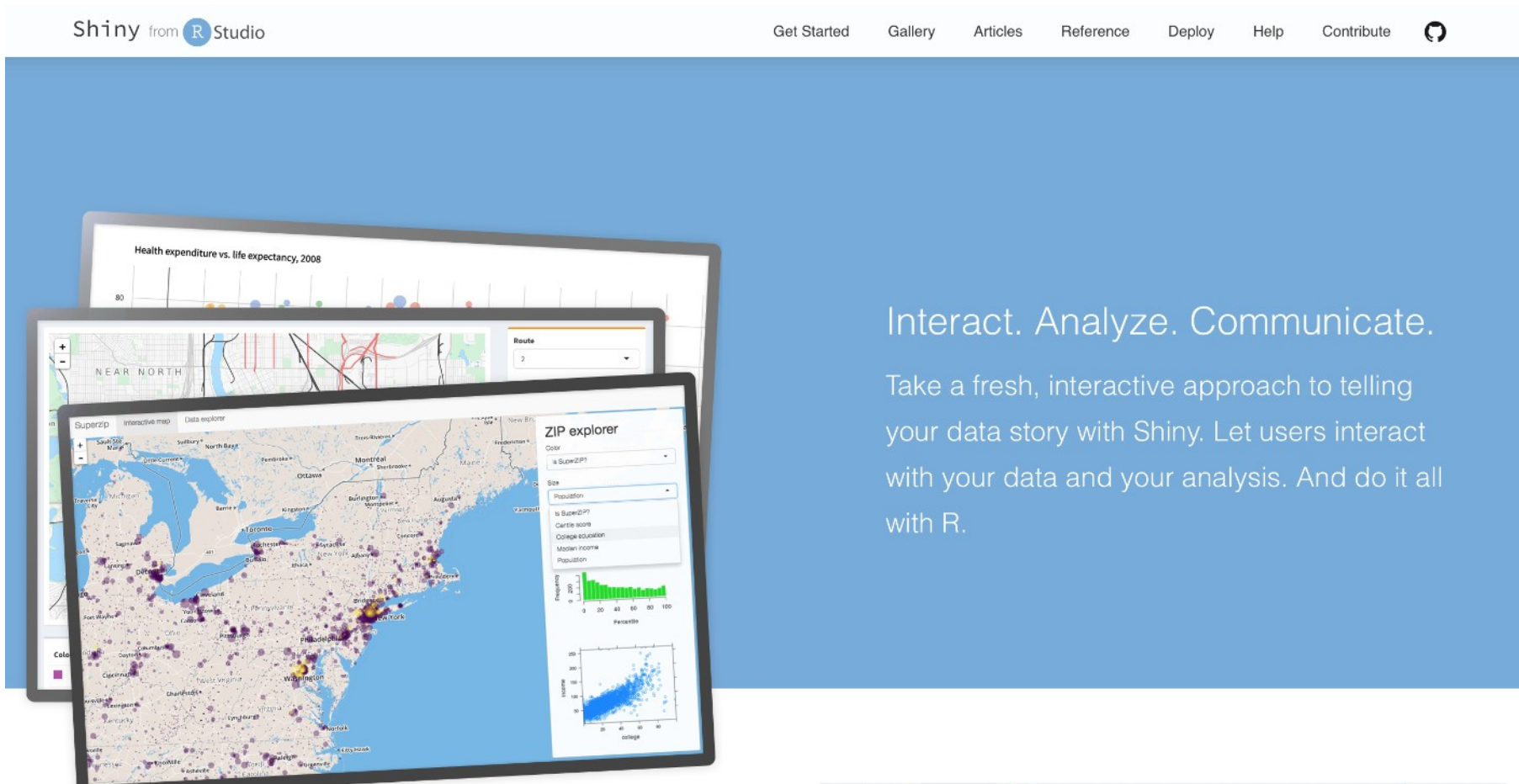
Writing a notebook is not just about writing the final document — Pluto empowers the experiments and discoveries that are essential to getting there.

Explore models and share results in a notebook that is

- **reactive** - when changing a function or variable, Pluto automatically updates all affected cells.
- **lightweight** - Pluto is written in pure Julia and is easy to install.
- **simple** - no hidden workspace state; friendly UI.

```
+  
0.111111  
· n ^ -2  
+  
3  
· n = 3
```

# Shiny




The screenshot displays the Shiny website interface. At the top left, it says "Shiny from R Studio". The navigation menu includes "Get Started", "Gallery", "Articles", "Reference", "Deploy", "Help", and "Contribute". The main content area features three overlapping dashboard examples:

- Health expenditure vs. life expectancy, 2008:** A scatter plot with a y-axis labeled "80" and data points of various colors.
- Interactive map:** A map of the "NEAR NORTH" area with a "Route" dropdown menu set to "2".
- ZIP explorer:** A dashboard with a map of the United States, a sidebar with filters for "Color" (set to "is SuperZIP?"), "Size" (set to "Population"), "is SuperZIP?", "Crime score", "College education", "Median income", and "Population". It also includes a histogram of "Percentage" vs "Percentage" and a scatter plot of "Income" vs "College".

Interact. Analyze. Communicate.

Take a fresh, interactive approach to telling your data story with Shiny. Let users interact with your data and your analysis. And do it all with R.

# Mercury



The image illustrates the Mercury interface, which provides a more interactive and user-friendly experience for running Jupyter notebooks. It is shown in two states: the original Jupyter notebook and the Mercury-enhanced version.

**Jupyter Notebook (Left):**

```
1 from matplotlib import pyplot as plt
2 import random

1 name = "Piotr"
2 points_count = 100

1 print(f"Hello {name}!")
Hello Piotr!

1 plt.figure(figsize=(10, 7))
2 x = [random.gauss(0, 1) for _ in range(points_count)]
3 y = [random.gauss(0, 1) for _ in range(points_count)]
4 _=plt.plot(x, y, '.', markersize=14)
```

**Mercury Interface (Right):**

The Mercury interface, titled "My notebook", features a sidebar with the following controls:

- Provide name:** A text input field containing "Piotr".
- Number of points:** A slider control ranging from 50 to 200, currently set to 100.
- Run:** A green button to execute the code.
- Download:** A blue button to download the notebook.
- Clear tasks:** A red button to clear the execution history.

The main content area displays the output of the notebook, including the text "Hello Piotr!" and a scatter plot of 100 blue points. The plot has a title "Python Notebook" and a subtitle "Hello Piotr!". The x-axis ranges from -3 to 2, and the y-axis ranges from -2 to 2.

# Interact.jl

## Interact

build passing docs latest

Interact.jl allows you to use interactive widgets such as sliders, dropdowns and checkboxes to play with your Julia code:

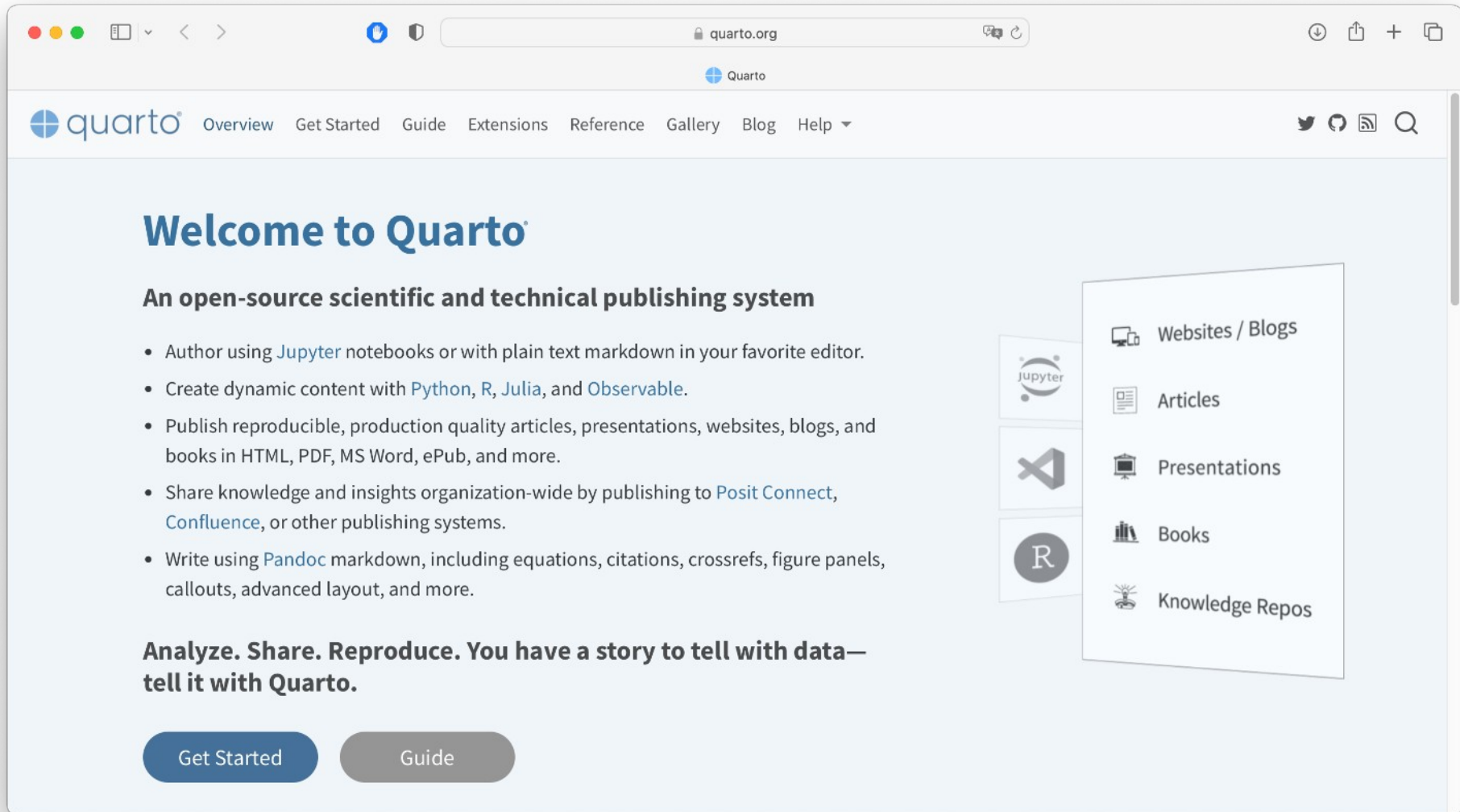


## Getting Started

To install Interact, run the following command in the Julia REPL:

```
Pkg.add("Interact")
```

# Quarto



The image shows a browser window displaying the Quarto website. The browser's address bar shows 'quarto.org'. The website header includes the Quarto logo and a navigation menu with links for Overview, Get Started, Guide, Extensions, Reference, Gallery, Blog, and Help. Social media icons for Twitter, GitHub, and RSS are also present. The main content area features a large heading 'Welcome to Quarto' and a sub-heading 'An open-source scientific and technical publishing system'. Below this is a list of bullet points describing the system's capabilities. To the right, a vertical list of icons represents different content types: Websites / Blogs, Articles, Presentations, Books, and Knowledge Repos. At the bottom, there are two buttons: 'Get Started' and 'Guide'.

## Welcome to Quarto

**An open-source scientific and technical publishing system**

- Author using [Jupyter](#) notebooks or with plain text markdown in your favorite editor.
- Create dynamic content with [Python](#), [R](#), [Julia](#), and [Observable](#).
- Publish reproducible, production quality articles, presentations, websites, blogs, and books in HTML, PDF, MS Word, ePub, and more.
- Share knowledge and insights organization-wide by publishing to [Posit Connect](#), [Confluence](#), or other publishing systems.
- Write using [Pandoc](#) markdown, including equations, citations, crossrefs, figure panels, callouts, advanced layout, and more.

**Analyze. Share. Reproduce. You have a story to tell with data—tell it with Quarto.**

[Get Started](#) [Guide](#)

- Websites / Blogs
- Articles
- Presentations
- Books
- Knowledge Repos

4.

## Extensions

Packages – Libraries – Modules

# Octave Forge



Octave-Forge is a central location for the collaborative development of packages for [GNU Octave](#).

The Octave-Forge packages expand Octave's core functionality by providing field specific features via Octave's package system. For example, image and signal processing, fuzzy logic, instrument control, and statistics packages are examples of individual Octave-Forge packages ([browse the full list of packages](#)).

### Installing packages

You can find the list of packages by clicking on the *Packages* link at the top. To install a package, use the `pkg` command from the Octave prompt by typing:

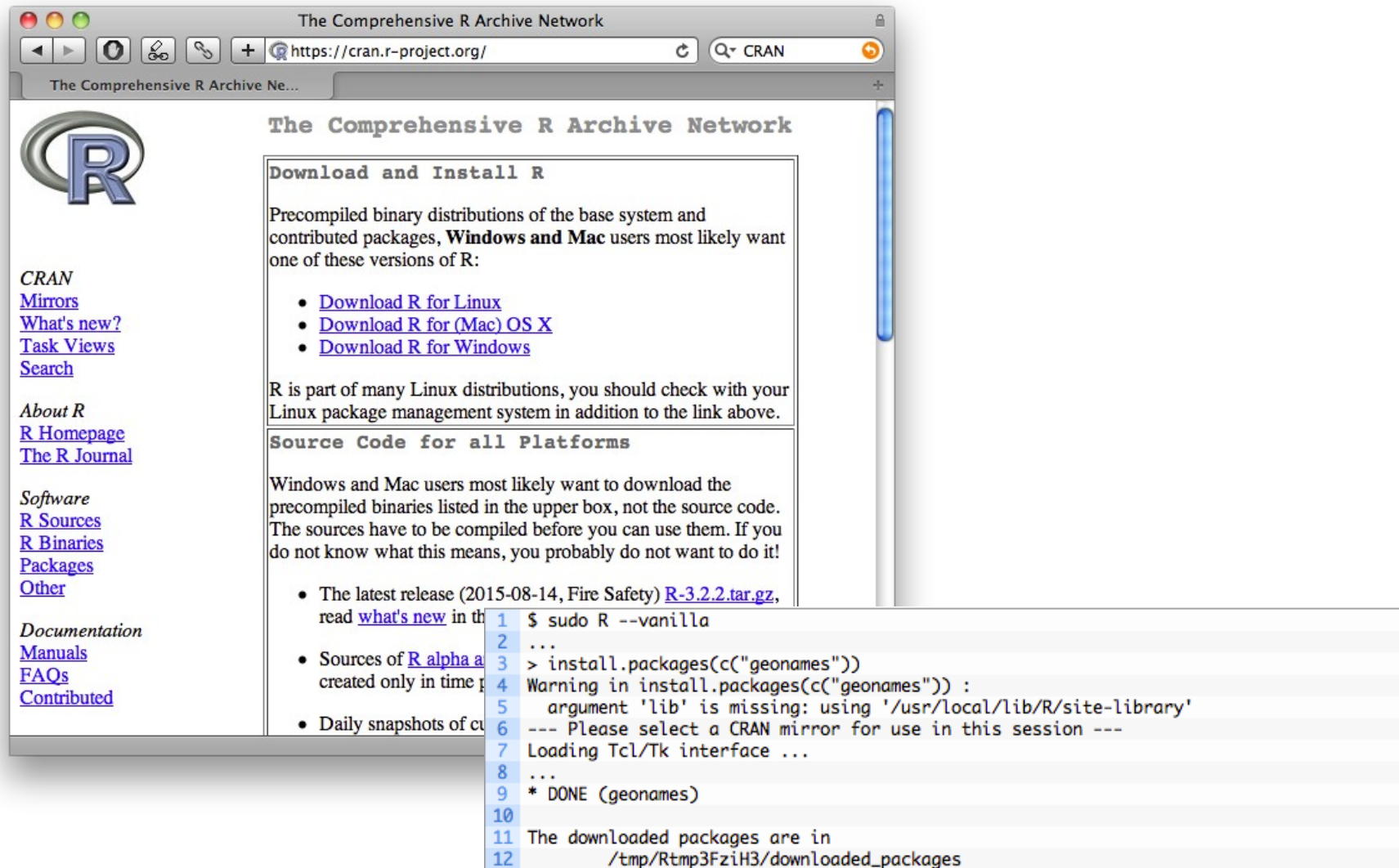
```
pkg install -forge package_name
```

where *package\_name* is the name of the package you want to install.

```
>> pkg install -forge image
warning: creating installation directory C:\Octave\Octave-4.0.0\share\octave
warning: called from
  install at line 30 column 5
  pkg at line 405 column 9
For information about changes from previous versions of the image package, r
>> pkg list
Package Name | Version | Installation directory
-----+-----+-----
          image |    2.4.0 | C:\Octave\Octave-4.0.0\share\octave\packages\image
```



# CRAN



The screenshot shows a web browser window displaying the CRAN website. The browser's address bar shows the URL `https://cran.r-project.org/` and the search bar contains the text "CRAN". The website content includes the CRAN logo, navigation links for "CRAN", "Mirrors", "What's new?", "Task Views", "Search", "About R", "R Homepage", "The R Journal", "Software", "R Sources", "R Binaries", "Packages", "Other", "Documentation", "Manuals", "FAQs", and "Contributed". The main content area is titled "The Comprehensive R Archive Network" and contains sections for "Download and Install R" and "Source Code for all Platforms".

The "Download and Install R" section states: "Precompiled binary distributions of the base system and contributed packages, **Windows and Mac** users most likely want one of these versions of R:"

- [Download R for Linux](#)
- [Download R for \(Mac\) OS X](#)
- [Download R for Windows](#)

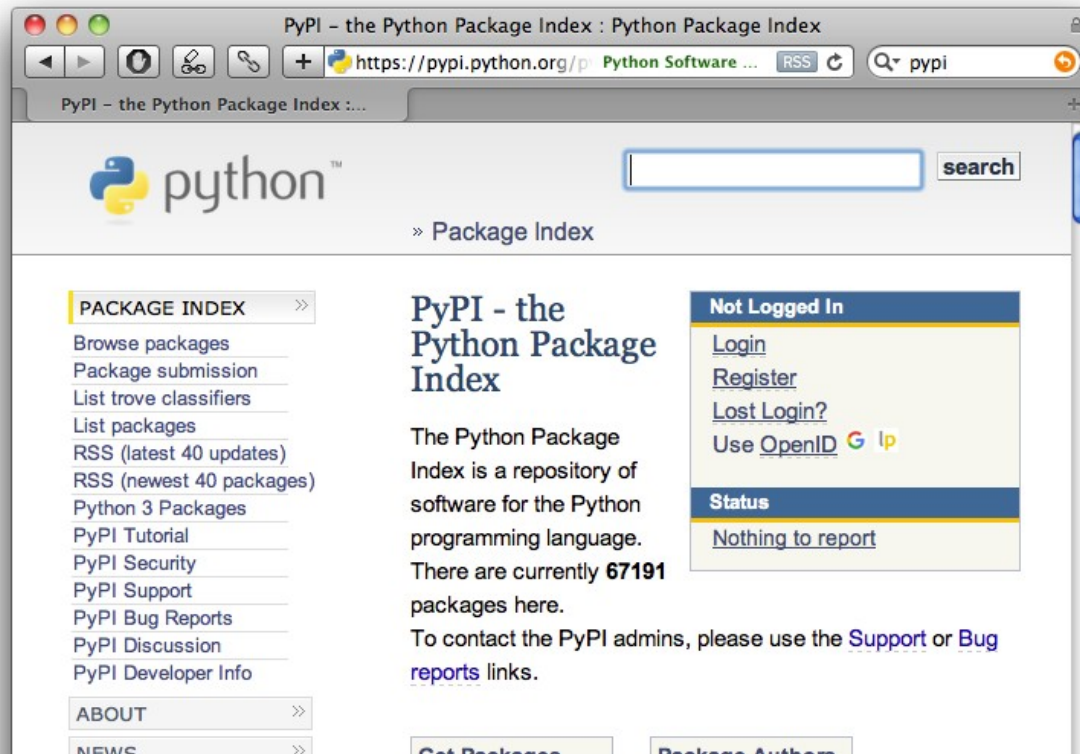
The "Source Code for all Platforms" section states: "Windows and Mac users most likely want to download the precompiled binaries listed in the upper box, not the source code. The sources have to be compiled before you can use them. If you do not know what this means, you probably do not want to do it!"

The terminal window shows the following commands and output:

```
1 $ sudo R --vanilla
2 ...
3 > install.packages(c("geonames"))
4 Warning in install.packages(c("geonames")) :
5   argument 'lib' is missing: using '/usr/local/lib/R/site-library'
6 --- Please select a CRAN mirror for use in this session ---
7 Loading Tcl/Tk interface ...
8 ...
9 * DONE (geonames)
10
11 The downloaded packages are in
12   /tmp/Rtmp3FziH3/downloaded_packages
```

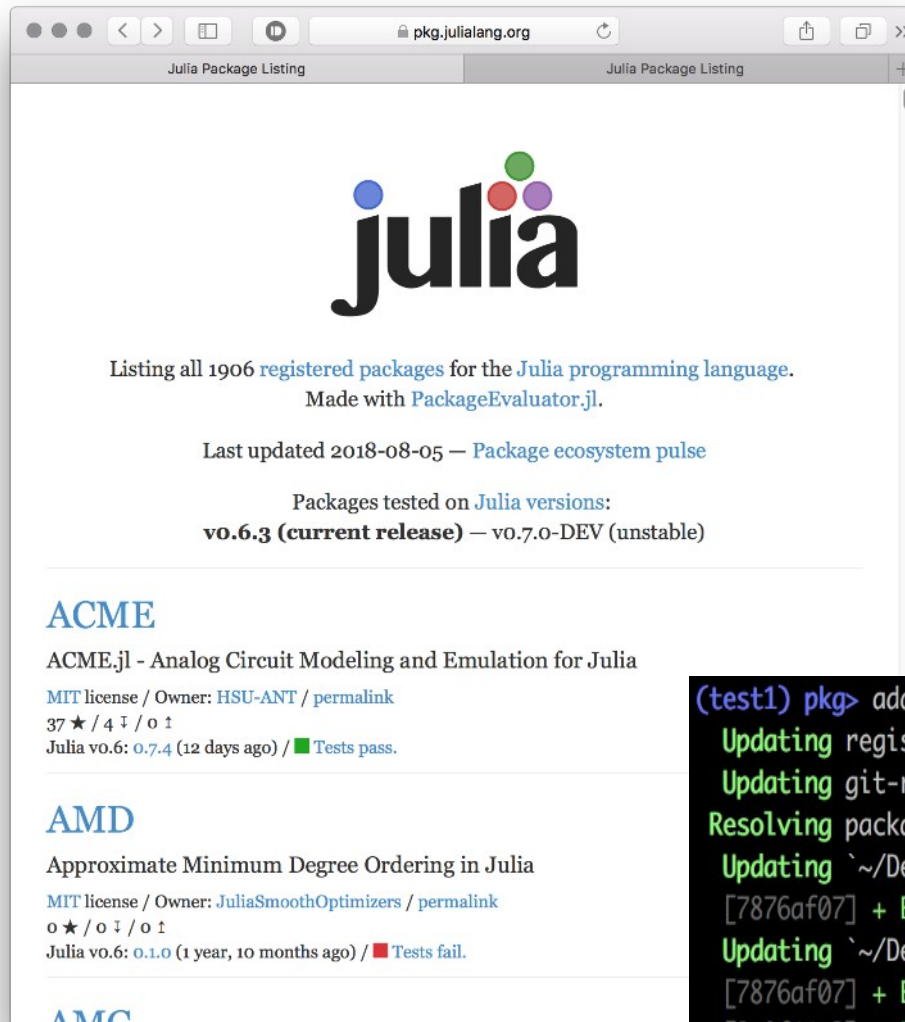


# PyPI



```
[root@localhost ~]# pip install virtualenv
/usr/lib/python2.6/site-packages/pip-7.1.0-py2.6.egg/pip/_vendor/requests/packages/urllib3/util/ssl_.py:90: InsecurePlatformWarning:
A true SSLContext object is not available. This prevents urllib3 from configuring SSL appropriately and may cause certain SSL connections to fail. For more information, see https://urllib3.readthedocs.org/en/latest/security.html#insecureplatformwarning.
InsecurePlatformWarning
Collecting virtualenv
/usr/lib/python2.6/site-packages/pip-7.1.0-py2.6.egg/pip/_vendor/requests/packages/urllib3/util/ssl_.py:90: InsecurePlatformWarning:
A true SSLContext object is not available. This prevents urllib3 from configuring SSL appropriately and may cause certain SSL connections to fail. For more information, see https://urllib3.readthedocs.org/en/latest/security.html#insecureplatformwarning.
InsecurePlatformWarning
Downloading virtualenv-13.1.0-py2.py3-none-any.whl (1.7MB)
100% |#####| 1.7MB 201kB/s
Installing collected packages: virtualenv
Successfully installed virtualenv-13.1.0
[root@localhost ~]#
```

# Julia package ecosystem



Listing all 1906 registered packages for the Julia programming language.  
Made with [PackageEvaluator.jl](#).

Last updated 2018-08-05 — [Package ecosystem pulse](#)

Packages tested on Julia versions:  
**vo.6.3 (current release)** — vo.7.0-DEV (unstable)

## ACME

ACME.jl - Analog Circuit Modeling and Emulation for Julia

MIT license / Owner: [HSU-ANT](#) / [permalink](#)  
37 ★ / 4 ↓ / 0 ↑  
Julia vo.6: 0.7.4 (12 days ago) / ■ Tests pass.

## AMD

Approximate Minimum Degree Ordering in Julia

MIT license / Owner: [JuliaSmoothOptimizers](#) / [permalink](#)  
0 ★ / 0 ↓ / 0 ↑  
Julia vo.6: 0.1.0 (1 year, 10 months ago) / ■ Tests fail.

## AMC

Pkg comes with a REPL. Enter the Pkg REPL by pressing ] from the Julia REPL. To get back to the Julia REPL, press backspace or ^C.

```
(test1) pkg> add Example
Updating registry at `~/julia/registries/General`
Updating git-repo `https://github.com/JuliaRegistries/General.git`
Resolving package versions...
Updating `~/Desktop/hobby/julia/test/test1/Project.toml`
[7876af07] + Example v0.5.1
Updating `~/Desktop/hobby/julia/test/test1/Manifest.toml`
[7876af07] + Example v0.5.1
[2a0f44e3] + Base64
[8ba89e20] + Distributed
```

# 5. General tips when it is slow

- Program thoughtfully:
  - Use vectorized functions
  - Avoid loops
  - Preallocate
  - Force type
  - Avoid copy-on-write
- Link to fast libraries (C/C++, Fortran, Java)
- Write low-level parts in C or Fortran
- Compile – jit
- Go parallel

## 6. Bridges (use ... in ... )

Python	→ R	<a href="http://rpython.r-forge.r-project.org/">http://rpython.r-forge.r-project.org/</a>
Octave	→ Python	<a href="https://pypi.python.org/pypi/oct2py">https://pypi.python.org/pypi/oct2py</a>
R	→ Python	<a href="http://rpy.sourceforge.net/">http://rpy.sourceforge.net/</a>
Octave	→ R	<a href="https://cran.r-project.org/web/packages/RcppOctave">https://cran.r-project.org/web/packages/RcppOctave</a>
Python	→ Octave	<a href="https://github.com/daniel-e/pyoctave">https://github.com/daniel-e/pyoctave</a>
R	→ Octave	<a href="http://www.omegahat.org/ROctave/">http://www.omegahat.org/ROctave/</a>
R	→ Julia	<a href="https://github.com/Non-Contradiction/JuliaCall">https://github.com/Non-Contradiction/JuliaCall</a>
Julia	→ R	<a href="https://github.com/JuliaInterop/RCall.jl">https://github.com/JuliaInterop/RCall.jl</a>
Python	→ Julia	<a href="https://github.com/JuliaPy/pyjulia">https://github.com/JuliaPy/pyjulia</a>
Julia	→ Python	<a href="https://github.com/JuliaPy/PyCall.jl">https://github.com/JuliaPy/PyCall.jl</a>

So..

Fast to learn  
Fast to code

# Challenge

- Look at the files in  
/CECI/proj/training/scripting/exercice2  
on any CÉCI cluster

```
[dfr@lemaitre3 exercice2]$ pwd
/CECI/proj/training/scripting/exercice2
[dfr@lemaitre3 exercice2]$ ls
res-10.txt  res-18.txt  res-25.txt  res-32.txt  res-3.txt   res-47.txt  res-54.txt  res-61.txt  res-69.txt  res-76.txt  res-83.txt  res-90.txt  res-98.txt
res-11.txt  res-19.txt  res-26.txt  res-33.txt  res-40.txt  res-48.txt  res-55.txt  res-62.txt  res-6.txt   res-77.txt  res-84.txt  res-91.txt  res-99.txt
res-12.txt  res-1.txt   res-27.txt  res-34.txt  res-41.txt  res-49.txt  res-56.txt  res-63.txt  res-70.txt  res-78.txt  res-85.txt  res-92.txt  res-9.txt
res-13.txt  res-20.txt  res-28.txt  res-35.txt  res-42.txt  res-4.txt   res-57.txt  res-64.txt  res-71.txt  res-79.txt  res-86.txt  res-93.txt
res-14.txt  res-21.txt  res-29.txt  res-36.txt  res-43.txt  res-50.txt  res-58.txt  res-65.txt  res-72.txt  res-7.txt   res-87.txt  res-94.txt
res-15.txt  res-22.txt  res-2.txt   res-37.txt  res-44.txt  res-51.txt  res-59.txt  res-66.txt  res-73.txt  res-80.txt  res-88.txt  res-95.txt
res-16.txt  res-23.txt  res-30.txt  res-38.txt  res-45.txt  res-52.txt  res-5.txt   res-67.txt  res-74.txt  res-81.txt  res-89.txt  res-96.txt
res-17.txt  res-24.txt  res-31.txt  res-39.txt  res-46.txt  res-53.txt  res-60.txt  res-68.txt  res-75.txt  res-82.txt  res-8.txt   res-97.txt
[dfr@lemaitre3 exercice2]$ cat res-1.txt
# Result file for experiment
[main]

parameter=0.01
result=0.15492

[meta]
time=531244[dfr@lemaitre3 exercice2]$
```

- We will pretend they are the result of running 100 jobs that take an input parameter and produce output a result (.INI file)
- Copy that directory into your home dir

# Challenge

- Find for which value of 'parameter' is 'result' the lowest.
- Course of action:
  - Read all files and parse them (you might need to install additional packages/libraries/modules)
  - Build two arrays one of parameter values and the other one for result values
  - Remove problematic values (plotting might help here)
  - Find minimum

# Challenge

- Pseudo code:
  - “Activate” extension for `.ini` files if necessary
  - Initialize two arrays to hold the values
  - For-loop 1-99 :
    - Read file (using a ready-made extension)
    - Store values in corresponding arrays
  - Remove from array values that show too large a difference between consecutive values (slicing)
  - Find index of minimum value in one array and the corresponding value in the other array



# Possible solution

```
# Result file for experiment
[main]
s.numeric("10")
parameter=0.1
result=0.13637
v(glove)
[meta]
time=1407642
```

→ INI file format

Libraries/packages/modules exist:

- <https://nl.mathworks.com/matlabcentral/fileexchange/17177-ini2struct>
- <https://cran.r-project.org/web/packages/ini/index.html>
- <https://docs.python.org/3/library/configparser.html>
- <https://juliapackages.com/p/inifile>

# Possible solution



```
nb_res=99;
p=zeros(nb_res,1);
r=zeros(nb_res,1);

for i = 1:nb_res;
    res = ini2struct(sprintf("res-%d.txt", i));
    p(i)=str2double(res.main.parameter);
    r(i)=str2double(res.main.result);
end
r(diff(r)>0.1)=nan;
plot(p,r)
[i, j]=min(r);
i, p(j)
```



```
import configparser
import numpy as np
import matplotlib.pyplot as plt

nb_res = 99

p = np.zeros(nb_res)
r = np.zeros(nb_res)

for i in range(nb_res):
    f = configparser.RawConfigParser()
    f.read("res-{}i.txt".format(i=i+1))
    p[i] = float(f.get('main', 'parameter'))
    r[i] = float(f.get('main', 'result'))

plt.plot(p, r, '-')
r[np.where(np.diff(r) > .1)] = np.nan
print(np.nanmin(r))
print(p[np.nanargmin(r)])
```



```
library(ini)
nb_res <- 99

p <- numeric(nb_res)
r <- numeric(nb_res)

for (i in 1:nb_res) {
    f <- read.ini(sprintf('res-%d.txt', i))
    p[i] <- as.numeric(f$main$parameter )
    r[i] <- as.numeric(f$main$result )
}

plot(p,r, 'l')
r[diff(r) > 0.1] <- NA
print(min(r, na.rm=T))
print(p[which.min(r)])
```



```
using IniFile
using Plots

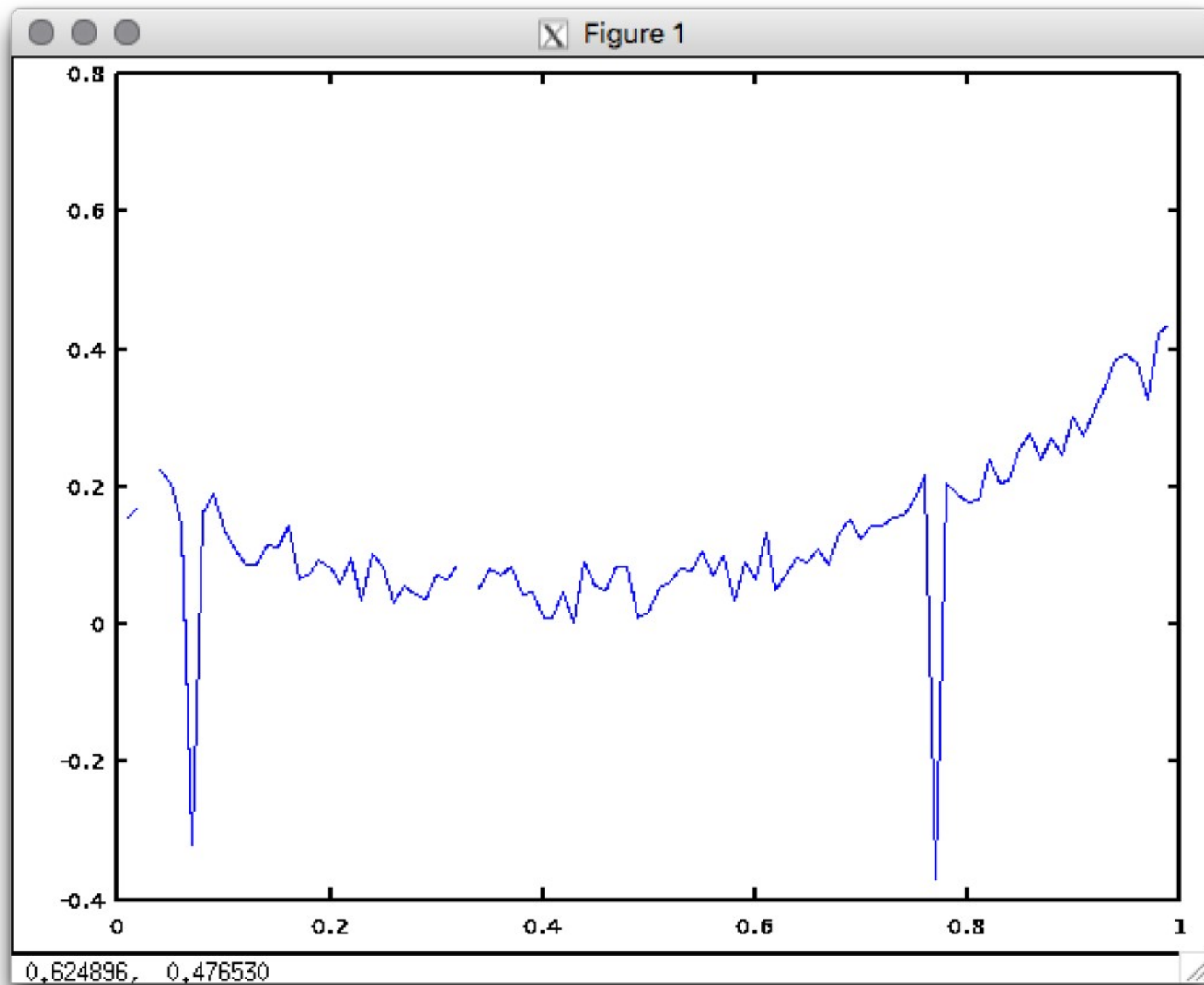
nb_res = 99;

p = Array{Float64}(undef, nb_res)
r = Array{Float64}(undef, nb_res)

for i in 1:nb_res
    ini = read(Inifile(), "res-{}i.txt");
    p[i] = parse(Float64, get(ini, "main", "parameter"))
    r[i] = parse(Float64, get(ini, "main", "result"))
end

r[findall(abs.(r[1:end-1] - r[2:end]).>.1)] .= NaN
r[findall(isnan.(r))] .= Inf
#plot(r)
show(findmin(r))
```

# Challenge



# Summary

Octave, R, Python\* and Julia\*

Much programmer-friendly than C/C++/Fortran

Still able to use fast compiled code

Focus on the unsolved problems

Try all and choose one

\* upcoming dedicated session!