Geophysical Data Processing from a browser: An introduction to Julia and Julia Box for teaching and research

M D Sacchi [1]
Department of Physics
University of Alberta

February 3, 2016

[1] Email: msacchi@ualberta.ca

What is Julia?

- High-level dynamic programming language
- High-level: Easy to understand by humans
- Dynamic: many common programming tasks are executed at run time (no compilation)
- Syntax is similar to Matlab
- Julia can be dowloaded from `http://julialang.org`

What is Julia?

- Main features:
  - parametric types, multiple dispatch, can call Python, etc
  - permits parallel and distributed computing
  - efficient libraries for scientific computing (linear algebra, random number generation, FFTs, *SeismicJulia*)
  - permits direct calling of C and Fortran libraries
  - nice graphics based on Python libraries (PyPlot)
  - good performance (similar to static C)
  - no need to feel ashamed when writing unvectorized codes
    - you can use loops!

Performance

	Fortran	Julia	Python	R	Matlab	Octave	Mathematica	JavaScript	Go	LuaJIT	Java
	gcc 5.1.1	0.4.0	3.4.3	3.2.2	R2015b	4.0.0	10.2.0	V8 3.28.71.19	go1.5	gsl-shell 2.3.1	1.8.0_45
fib	0.70	2.11	77.76	533.52	26.89	9324.35	118.53	3.36	1.86	1.71	1.21
parse_int	5.05	1.45	17.02	45.73	802.52	9581.44	15.02	6.06	1.20	5.77	3.35
quicksort	1.31	1.15	32.89	264.54	4.92	1866.01	43.23	2.70	1.29	2.03	2.60
mandel	0.81	0.79	15.32	53.16	7.58	451.81	5.13	0.66	1.11	0.67	1.35
pi_sum	1.00	1.00	21.99	9.56	1.00	299.31	1.69	1.01	1.00	1.00	1.00
rand_mat_stat	1.45	1.66	17.93	14.56	14.52	30.93	5.95	2.30	2.96	3.27	3.92
rand_mat_mul	3.48	1.02	1.14	1.57	1.12	1.12	1.30	15.07	1.42	1.16	2.36

Figure: benchmark times relative to C (smaller is better, C performance = 1.0).

From `http://julialang.org`

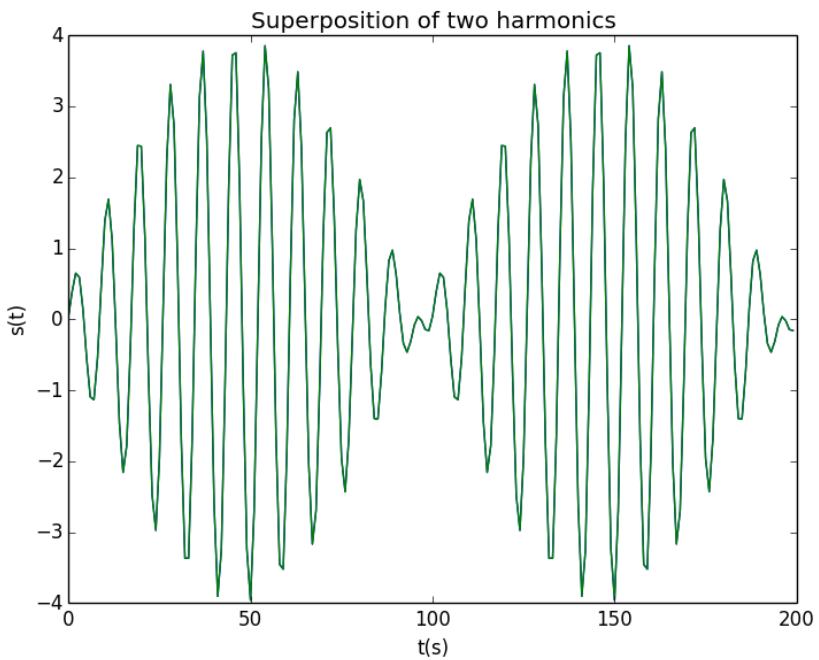
Example 1: foo_1.jl (a Julia script)

```
# foo_1.jl
# Superpositon of two sinusoids

 using PyPlot

  N = 200
 dt = 1
 f1 = 0.11
 f2 = 0.12
  t = (0:1:N-1)*dt

  s = 2*cos(2*pi*t*f1) - 2*cos(2*pi*t*f2+0.2)

  plot(t,s)
  xlabel("t(s)")
  ylabel("s(t)")
  title("Superposition of two harmonics")
```

Example 1: Running your code

Once you have installed Julia, you can run Julia from a terminal and use `include` to run your code

```
julia> include("foo_1.jl")
```



Example 2 : foo_2.jl (a Julia function)

```
1  function foo_2(f1,f2; N=200,dt=1.)
2  # Superposition of two sinusoids of freqs f1 and f2
3  # N:  lenght of signal (defaults to 200)
4  # dt: sampling interval (defaults to 1.)
5
6    t = (0:1:N-1)*dt
7    s = 2*cos(2*pi*t*f1) - 2*cos(2*pi*t*f2+0.2)
8
9   return t,s
10 end
```

Note that after ; are the keyword arguments. They are used to assign default values.

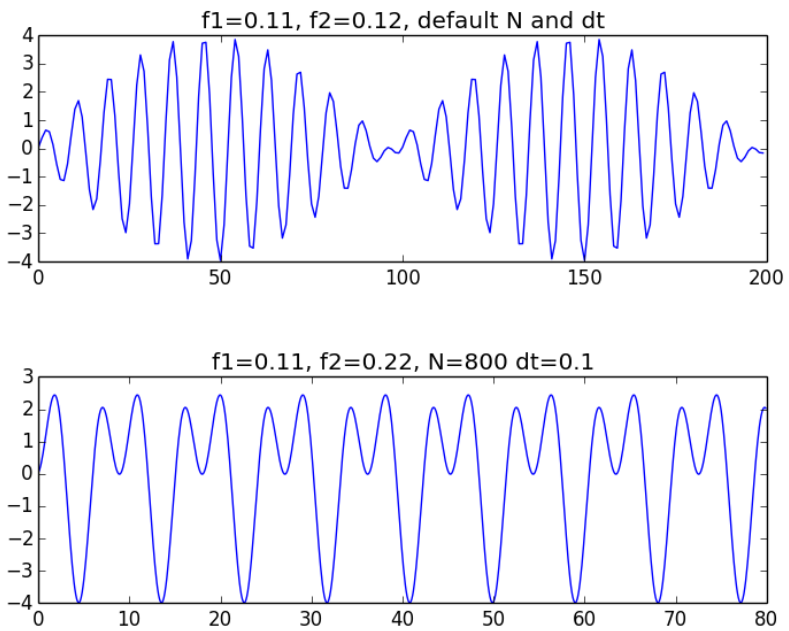
Example 2 : main.jl calls the function foo_2.jl

```
1
2  # main.jl
3
4  using PyPlot
5  include("foo_2.jl")
6
7  # Run foo with default N and dt
8
9  (t1,s1) = foo_2(0.11,0.12)
10 subplot(211)
11 plot(t1,s1)
12 title("f1=0.11, f2=0.12, default N and dt")
13
14 # Run foo with non-default N and dt
15
16 (t2,s2) = foo_2(0.11, 0.22, N=800, dt=0.1)
17 subplot(212)
18 plot(t2,s2)
19 title("f1=0.11, f2=0.22, N=800 dt=0.1")
```

Example 2: Running your code

Once you have installed Julia, you can run Julia from a terminal. Use `include` to run your code

```
1 julia> include("main.jl")
```



Example 3 : Module called library.jl

```
# library.jl
module library
 function foo_2(f1,f2; N=200,dt=1.)
 # Superposition of two sinusoids of freqs f1 and f2
 # N:  lenght of signal (defaults to 200)
 # dt: sampling interval (defaults to 1.)
   t = (0:1:N-1)*dt
   s = 2*cos(2*pi*t*f1) - 2*cos(2*pi*t*f2+0.2)
  return t,s
  end
 function periodogram(s;dt=1.,k_pad=2)
 # Compute the Periodogram of a real time series
  N = length(s); M = k_pad*nextpow2(N)
  Mh = convert(Int64,M/2)+1
  s_pad = [s,zeros(M-N)]
  P  = (dt/M)*abs(fft(s_pad))
  P = P[1:Mh]
  f = (0:1:Mh-1)/(dt*M)     # Freq in Hertz
 return f, P
 end
end
```

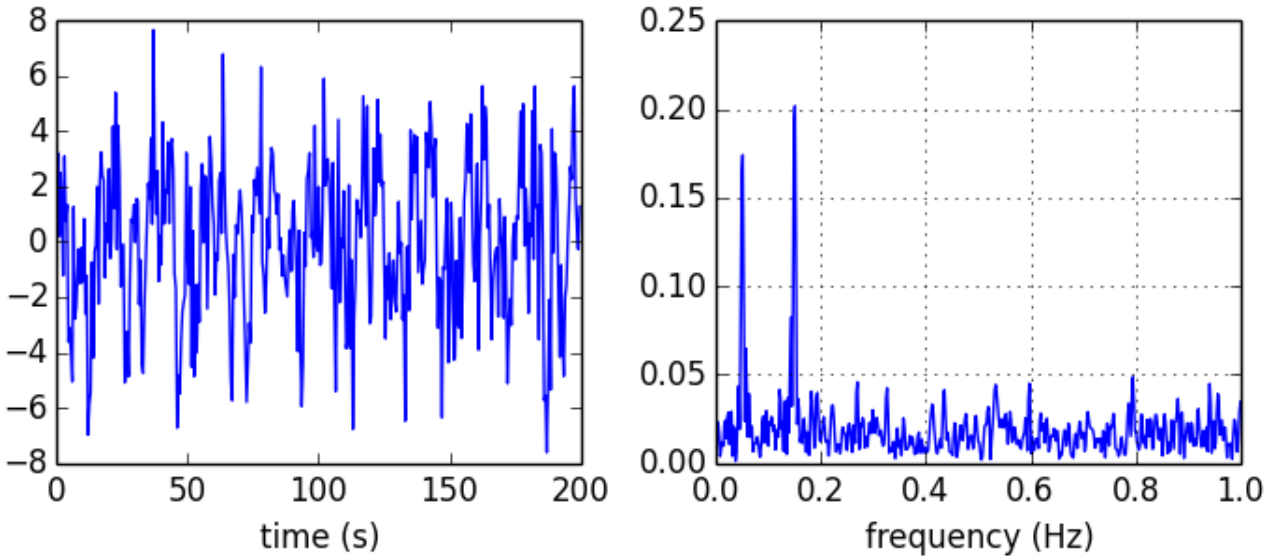
Example 3 : I am using prog.jl to call functions in library.jl

```
# prog.jl
using PyPlot
include("library.jl")

(t,s) = library.foo_2(0.05,0.15,dt=0.5,N=400)

s = s + 2.*randn(size(s))       # add noise

(f,P) = library.periodogram(s,dt=0.5)

subplot(221);plot(t,s);xlabel("time (s)")

subplot(222);plot(f,P);xlabel("frequency (Hz)")
grid("on")
```

then, we run prog.jl

```
julia> include("prog.jl")
```

Example 3 : You should get the following



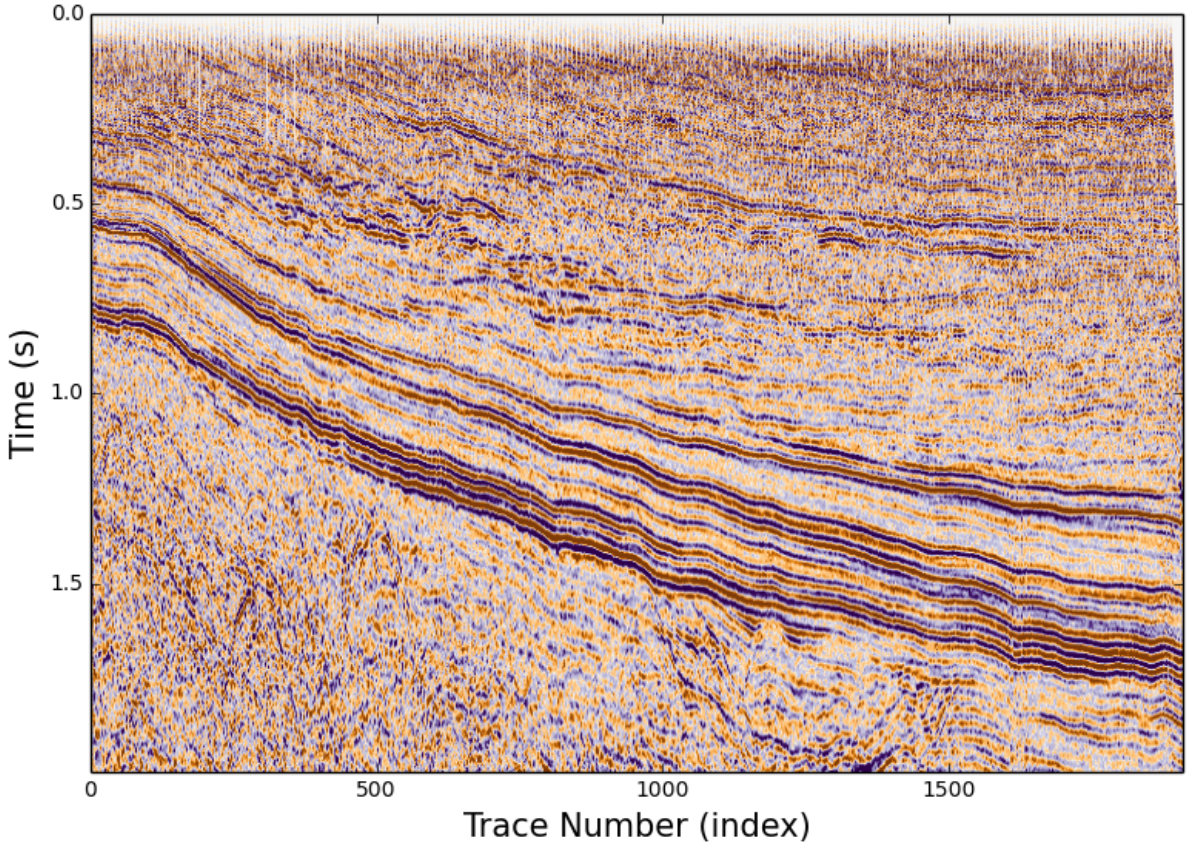
Example 4 : Using SeismicJulia

```
# aaron.jl

  using PyPlot,Seismic


#  download("http://certmapper.cr.usgs.gov/nersl/NPRA/seismic
    /1979/616_79/PROCESSED/616_79_PR.SGY","616_79_PR.SGY");
   download("http://seismic.physics.ualberta.ca/data/mc_no.su","
    mc_no.su");


  SegyToSeis("mc_no.su","mc_no.seis",format="su")

  d,h,e = SeisRead("mc_no.seis")

  SeisPlot(d,e,cmap="PuOr",wbox=9)
```

Then, we run

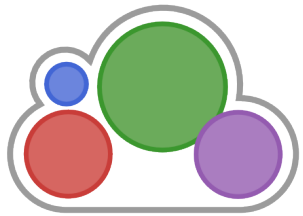
```
julia> include("aaron.jl")
```

Example 4 : Using SeismicJulia



SeismicJulia is a package created by Aaron Stanton to read/write and process 5D seismic volumes `https://github.com/SeismicJulia/Seismic.jl`

What is Juliabox?



- `https://www.juliabox.org`
- a cloud environment hosted by Amazon (AWS) for Julia (and Python)
- uses IJulia to create Notebooks (we can host our own ”local cloud”)
- interactive notebook coding
- easy to use, you only need your UofA CCID
- you don’t need to install software on your computer
- you can concentrate on your assignments and/or research rather than on IT stuff
- expensive computer labs for teaching could become obsolete
- your lab can be a Starbucks coffee shop: you only need an internet connection and a computer with a browser
- A $308.99 Chromebook (Dell Canada -11 inches) will do it :)

Example: Teaching UG Seismic Signal Processing with IJulia

- I used JuliaBox to create the Notebook and then *nbviewer* to render it as a web document.
- Check this notebook:
  `http://nbviewer.jupyter.org/gist/msacchi/92dd380fb28793888a3b`
- Students should work directly with the notebook to reproduce results under different set of parameters.
- Students should submit their homework using Notebook with a similar structure (text, equations, programs, results, etc)

Example: Teaching iterative reduced-rank image reconstruction with IJulia

- I used JuliaBox to create the Notebook and then *nbviewer* to render it as a web document.
- Check this notebook:
  `http://nbviewer.jupyter.org/gist/msacchi/676fb34d8336b5509625`

Current status: Research

- Sam Kaplan (Chevron) pointed us to Julia about a year ago. Sam is adopting Julia for Seismic R & D
- Aaron developed SeismicJulia. Codes for his PhD project (Elastic Wave Equation Migration) are written in Julia. Aaron is our Julia SGL (Supreme Guiding Leader). Heavy parts are in C called by Julia.
- Wenlei developed a PP-PS registration package in Julia
- Linan developed a Julia RTM package with calls to C
- Fernanda, Juan Ignacio, Linan and Tianqui are also adopting Julia for their research projects.
- Gian is using Python to call C for his E-FWI project.
- Group seems to be moving to Julia — Python — C

Current status: Teaching

- MDS plans to adopt Julia and JuliaBox for GEOPH 426 or GEOPH 431 & 531 (depending on course assignment)
- Notebooks (Python and/or Julia) could be used to teach all UG geophysics courses and 2nd year Computational Physics
- We should introduce UGs to data analytics and scientific computing early on their career and make them more flexible to face current job prospects
- Stay tuned for series of hands-on seminars during the summer or beginning of next term