julia

A fresh approach to technical computing

Carlos Becker

Why yet another language?

We love prototyping, but …

- We are greedy
  - We want blazing-fast code!
- We are picky
  - We hate debugging, re-compiling, debugging…
  - We hate segfaults (also known as <<WTF?>>)
- We are lazy
  - Fast prototyping

    why C++ when we can use Python or Matlab-like syntax?

julia

Just-in-Time compilation

Matlab-like syntax

Compiled to machine code

Open source!!

Why yet another language?

MATLAB

```
function [x_samp,y_samp] = gibbs2(n,thin)

   x_samp = zeros(n,1);
   y_samp = zeros(n,1);

   y = 0;
   x = 0;

   for i=1:n

      for j=1:thin
         x=(y^2+4) * randg(3,1,1);
         y=1/(1+x) + randn/sqrt(2*x+2);
      end

      x_samp(i) = x;
      y_samp(i) = y;
   end

end


tic; gibbs2(50000, 1000); toc
```

MATLAB's JIT cannot optimize these calls

128 seconds

julia

```
function gibbs2(n, thin)

   x_samp = zeros(n,1)
   y_samp = zeros(n,1)

   x = 0.0
   y = 0.0

   for i=1:n

      for j=1:thin
         x=(y^2+4) * rand(Gamma(3))
         y=1/(1+x) + randn()/sqrt(2*x+2)
      end

      x_samp[i] = x
      y_samp[i] = y
   end

   return x_samp, y_samp
end

@time gibbs2(50000, 1000)
```

4.5 seconds

(c++ version w/GSL: 8.1 seconds)

[Darren Wilkinson's blog]

[http://timsalimans.com/the-power-of-jit-compilation/]

Why yet another language?

MATLAB

Let's give it another try…

```
function [x_samp,y_samp] = gibbs2(n,thin)

    x_samp = zeros(n,1);
    y_samp = zeros(n,1);

    y = 0;
    x = 0;

    for i=1:n

        % PRE-allocate random numbers
        gammarands = randg(3,thin,1);
        normrands = randn(thin,1);

        for j=1:thin
            x=(y^2+4) * gammarands(j);
            y=1/(1+x) + normrands(j)/sqrt(2*x+2);
        end

        x_samp(i) = x;
        y_samp(i) = y;

    end
end
```

7.2 seconds

[http://timsalimans.com/the-power-of-jit-compilation/]

julia

```
function gibbs2(n, thin)

   x_samp = zeros(n,1)
   y_samp = zeros(n,1)

   x = 0.0
   y = 0.0

   for i=1:n

      for j=1:thin
         x=(y^2+4) * rand(Gamma(3))
         y=1/(1+x) + randn()/sqrt(2*x+2)
      end

      x_samp[i] = x
      y_samp[i] = y
   end

   return x_samp, y_samp
end

@time gibbs2(50000, 1000)
```

4.5 seconds

(c++ version w/GSL: 8.1 seconds)

[Darren Wilkinson's blog]

Why yet another language?

Just-in-Time compilation

- Functions compiled to native code on-the-fly

Multiple dispatch

- Arguments' type determines which method to invoke

  eg. `myfunc(Double, Double)` and `myfunc(Int, Int)`
  should have different native code implementations

And more cool stuff

- Clean code
- Call C/Fortran functions directly
- Easily call Python functions
- Designed for parallelism and distributed computation
- Many libraries (Packages) already available (*more on this later*)
- and more ...

Hands on!
(Switch to IJulia)

Packages

Packages

Julia comes with an inbuilt Package Manager

- `Pkg.add(“PackageName”)` to add a package
    - Automatically checks out latest version, downloads, compiles and installs it !

- Summary of available packages at http://julia.readthedocs.org/en/latest/packages/packagelist/

- Generally well documented code, open source

- (btw, most of Julia's code itself is written in Julia)

Packages

Some neat examples:

- DataFrames: Tools for using statistical data / tables
- DimensionalityReduction: PCA, ICA, NMF, ...
- Distributions: Probability distributions and related functions
- MixtureModels, CRF , MCMC

- Wrappers
  - CUDA, OpenCL
  - Pandas
  - PyCall, PyPlot: Matplotlib plots in Julia
  - PySide: use QT through PyCall and PySide
  - OpenGL
  - LIBSVM
  - CoreNLP: interface to Stanford's CoreNLP toolkit
  - Mosek / NLopt / Gurobi / CPLex
  - LightXML

- I/O extensions
  - Images, ImageView
  - MAT: write/read MATLAB files in Julia
  - MATLAB: call matlab from Julia

- Datasets
  - MNIST
  - Rdatasets

.. and many more...

Getting started: Learning Julia

Getting Started

- Download latest version from http://julialang.org (I use v0.3)
- If on **Linux**, just checkout the git repo and compile it yourself
- If you have questions, search the web or ask in the **mailing list** (http://julialang.org/community/)
- If using Sublime, do not miss the Sublime-IJulia integration plugin

- To keep in mind: work in progress:
  - you may want to get v0.3 to get the last improvements, instead of v0.2
  - On Mac/Windows, packages that need external libraries may be tricky to get to work. Hopefully will be fixed soon.

Getting Started

- There are a few tutorials "à la hands on"
  - Learn Julia in Y minutes: http://learnxinyminutes.com/docs/julia/
  - Forio's tutorials: http://forio.com/products/julia-studio/tutorials/
    (check out the "Save the Apollo 13 Astronauts!" tutorial!)
  - Tutorial Videos at MIT: http://julialang.org/blog/2013/03/julia-tutorial-MIT/

- Plus the official docs: http://docs.julialang.org/en/latest/manual/

- Check out Noteworthy differences from Matlab/Python/R from the Julia manual

Thank you, and happy prototyping!