Index

September 11, 2018

1 A Deep Introduction to Julia for Data Science and Scientific Computing

1.1 Introduction

This workshop is put together by Chris Rackauckas as part of the UC Irvine Data Science Initiative. This workshop is made to teach people who are experienced with other scripting languages the relatively new language Julia. Unlike the other Data Science Initiative workshops, this workshop assumes prior knowledge of some form of programming in a language such as Python, R, or MATLAB.

We will start by introducing the basic syntax of the language, and quickly move into the details of how Julia is different from other scripting languages and how to exploit Julia's type system + multiple dispatch to be able to achieve C/Fortran-like performance while maintaining the concise syntax of a scripting language. Large parts of the package ecosystem will be explored and some information on implementation details will be highlighted in order for the participants to learn how to design Julia projects.

The ideal participant is anyone who is interested in Julia. There are many groups of people interested in using Julia. One large fraction come with a strong software development background and C/Fortran knowledge, and are looking to learn Julia as a tool to create packages with enhanced productivity while not losing performance. On the other side, there are users who are interested in the growing package ecosystem of Julia and would like to add Julia to their knowledgebase. And then there's everything in between. One major goal of this workshop is to use Julia's language and syntax to bridge the gap between "package users" and "package developers" in the way that Julia has done.

This is very problem-focused. The method is not passive: the goal is to get you using Julia!

1.2 Introduction to the Author

Chris is a PhD student in Mathematics from the Mathematical, Computational, and Systems Biology (MCSB) Gateway program and is an active part of the Julia community. He runs the blog, http://www.stochasticlifestyle.com/, where he write tutorials on using the Julia languages covering many topics, such as high-performance and GPGPU computing, package development, and Julia tips. He is part of the JuliaMath and JuliaDiffEq communities for curating the Julia libraries for mathematics and differential equations respectively. Chris is the developer of many Julia packages, the most prominent being DifferentialEquations.jl, a Julia library which has become a standard solver for many forms of differential equations.

1.3 Notes Before We Get Started

- Please install some version of IJulia/Jupyter to follow along. I also recommend working through longer problems using the Juno IDE.
- The start of the course will be on developing general performant Julia code. After lunch it will be about the package ecosystem. This understanding of Julia will be useful even for "Julia users" (i.e. non package developers) to use packages effectively, but don't worry we will get to packages.
- Please do the problems/projects at your own pace. Not everyone is expected to complete all of the material in the allotted time. **Some of the problems are supposed to be hard!** Instead, this is supposed to be a resource to introduce you to a large amount of Julia, and the material may take awhile to fully be digested. That's okay!
- During the basic introduction, there will be information that is not included in these notebooks. That is intentional. One major hurdle for learning a language is learning how to find out more about the language. Please use the manual, chatrooms, StackExchange, etc. If these aren't working for you, ask for help.

A good primer for the workshop: https://www.youtube.com/watch?v=JNvMs0j3a4E

1.4 Installation and Setup

To get started, see the Tooling, Documentation, and Community notebook.

1.5 Rendered Jupyter Notebooks

1.5.1 Introduction

- Tooling, Documentation, and Community
- Juno, the Julia IDE (Debugging, Progress Bars, etc.): JunoLab
- A Mental Model for Julia
- A Very Quick Introduction to Git/Github for Julia Users

1.5.2 Basics

- A Basic Introduction to Julia
- Why Julia?
- More Details on Arrays and Matrices - How to get Fortran Speeds in Linear Algebra
- The Julia Manual
- The Julia Wikibook
- Noteworthy Differences from Other Languages (Julia's Manual)
- Julia Cheatsheet Reference (with MATLAB and Python)

1.5.3 General Problems

- Basic Problems
- Intermediate Problems
- Advanced Problems
- Stock Modeling Problem

1.5.4 Detailed Usage: A Peek Into "the Rabbit Hole"

- Multiple Dispatch Designs: Duck Typing, Hierarchies and Traits
- Metaprogramming
- Call Overloading
- 7 Julia Gotchas and How to Handle Them
- Array and Iterator Interfaces

1.5.5 Packages to Explore (with Problems!)

This section introduces you to a wide variety of packages for data science and scientific computing in Julia. Many of these pages have example problems for you to have a guided tour through the package basics.

- Overview of the Package Ecosystem

Data Science

- Clustering: Clustering.jl and Distances.jl
- Dimensionality Reduction: MultivariateStats.jl
- Data Visualization: Plots.jl
- Bioinformatics: Bio.jl
- Deep Learning: Flux.jl

Scientific Computing

- Differential Equations: DifferentialEquations.jl
- Solving Nonlinear Equations: NLsolve.jl and Roots.jl
- Graph Algorithms and Analysis: LightGraphs.jl

Both!

- Mathematical Programming / Optimization: JuMP and Optim.jl
- Forward-mode Autodifferentiation (with Optimization): ForwardDiff.jl and Optim.jl

1.5.6 Extra Projects and Problems

- Using External Languages from Julia (Interop)
- Parallelism and HPC
- Package Development, Documentation, and Testing
- Robust Benchmarking: BenchmarkTools.jl and ProfileView.jl

1.5.7 Experiments

Probe around and understand the following results:

- Type-Stability Experiment
- Scoping Experiment

Problem Answers

Answers to the Problems

2 Slide Versions of the Pages

2.0.1 Introduction

2.0.2 Basics

2.0.3 Detailed Usage: A Peak Into "the Rabbit Hole"

2.0.4 Packages to Explore

- Forward-mode Autodifferentiation (with Optimization): ForwardDiff.jl and Optim.jl
- Graph Algorithms and Analysis: LightGraphs.jl
- Bioinformatics: Bio.jl
- Deep Learning: KNet.jl, TensorFlow.jl, MXNet.jl, and Flux.jl

2.0.5 Extra Projects and Problems

- Using External Languages from Julia (Interop)
- Parallelism and HPC
- Work in Progress: JuliaML for Machine Learning
- Package Development, Documentation, and Testing
- Tools for Faster Code: InplaceOps.jl, CatViews.jl, and VML.jl
- Robust Benchmarking: BenchmarkTools.jl and ProfileView.jl

2.0.6 Experiments

Probe around and understand the following results:

- Type-Stability Experiment
- Scoping Experiment

2.1 How These Were Made

This entire repository is made using Jupyter notebooks using the template from the JupyterSite repository. To contribute to these materials, see the Github repository.