Intro to the Julia programming language

Brendan O'Connor
CMU, Dec 2013

They have very good docs at: **http://julialang.org/**

I'm borrowing some slides from: http://julialang.org/blog/2013/03/julia-tutorial-MIT/

Julia

- A relatively new, open-source numeric programming language that's both <u>convenient</u> and <u>fast</u>
- Version 0.2. Still in flux, especially libraries. But the basics are very usable.
- Lots of development momentum

Why Julia?

Dynamic languages are extremely popular for numerical work:

- Matlab, R, NumPy/SciPy, Mathematica, etc.
- very simple to learn and easy to do research in

However, all have a “split language” approach:

- high-level dynamic language for scripting low-level operations
- C/C++/Fortran for implementing fast low-level operations

Libraries in C — no productivity boost for library writers

Forces vectorization — sometimes a scalar loop is just better

slide from ?? 2012
Bezanson, Karpinski, Shah, Edelman

“Gang of Forty”

Matlab Maple Mathematica SciPy SciLab IDL R
Octave S-PLUS SAS J APL Maxima Mathcad
Axiom Sage Lush Ch LabView O-Matrix PV-WAVE
Igor Pro OriginLab FreeMat Yorick GAUSS MuPad
Genius SciRuby Ox Stata JLab Magma Euler Rlab
Speakeasy GDL Nickle gretl ana Torch7

slide from March 2013
Bezanson, Karpinski, Shah, Edelman

Numeric programming environments

Core properties

	Dynamic and math-y?	Fast?
C/C++/ Fortran/Java	−	+
Matlab	+	−
Num/SciPy	+	−
R	+	−

Older table: http://brenocon.com/blog/2009/02/comparison-of-data-analysis-packages-r-matlab-scipy-excel-sas-spss-stata/

- Dynamic vs Fast: the usual tradeoff
- PL quality: more subjective. can you define more than 1 function in a file? do you have a module system? do operators and functions work in a consistent way?
- Julia aims to have all of them
- Ecosystem
- To the extent there's still a CS/Stats divide (or engineering/stats divide), you see it in R versus Matlab.

Numeric programming environments

Core properties

	Dynamic and math-y?	Fast?
C/C++/ Fortran/Java	−	+
Matlab	+	−
Num/SciPy	+	−
R	+	−
Julia	+	+

Matlab-style syntax, REPL

Close to C speeds

Older table: http://brenocon.com/blog/2009/02/comparison-of-data-analysis-packages-r-matlab-scipy-excel-sas-spss-stata/

Tuesday, December 17, 13

- Dynamic vs Fast: the usual tradeoff
- PL quality: more subjective. can you define more than 1 function in a file? do you have a module system? do operators and functions work in a consistent way?
- Julia aims to have all of them
- Ecosystem
- To the extent there's still a CS/Stats divide (or engineering/stats divide), you see it in R versus Matlab.

Numeric programming environments

Core properties

	Dynamic and math-y?	Fast?	PL quality?
C/C++/ Fortran/Java	−	+	+
Matlab	+	−	−
Num/SciPy	+	−	+
R	+	−	−
Julia	+	+	++

Matlab-style syntax, REPL

Close to C speeds

Optional static types
Multiple dispatch
Lisp-style macros

Older table: http://brenocon.com/blog/2009/02/comparison-of-data-analysis-packages-r-matlab-scipy-excel-sas-spss-stata/

Tuesday, December 17, 13

- Dynamic vs Fast: the usual tradeoff
- PL quality: more subjective. can you define more than 1 function in a file? do you have a module system? do operators and functions work in a consistent way?
- Julia aims to have all of them
- Ecosystem
- To the extent there's still a CS/Stats divide (or engineering/stats divide), you see it in R versus Matlab.

Numeric programming environments

	Core properties			Ecosystem		
	Dynamic and math-y?	Fast?	PL quality?	Open-source	Stat libraries (viz, regul/Bayes, regression...)	Engr libraries (optimization, signals...)
C/C++/ Fortran/Java	−	+	+	+	−	−
Matlab	+	−	−	−	~	+
Num/SciPy	+	−	+	+	~	~
R	+	−	−	+	++	−
Julia	+	+	++	+	underway	underway

Matlab-style syntax, REPL

Close to C speeds

Optional static types
Multiple dispatch
Lisp-style macros

Older table: http://brenocon.com/blog/2009/02/comparison-of-data-analysis-packages-r-matlab-scipy-excel-sas-spss-stata/

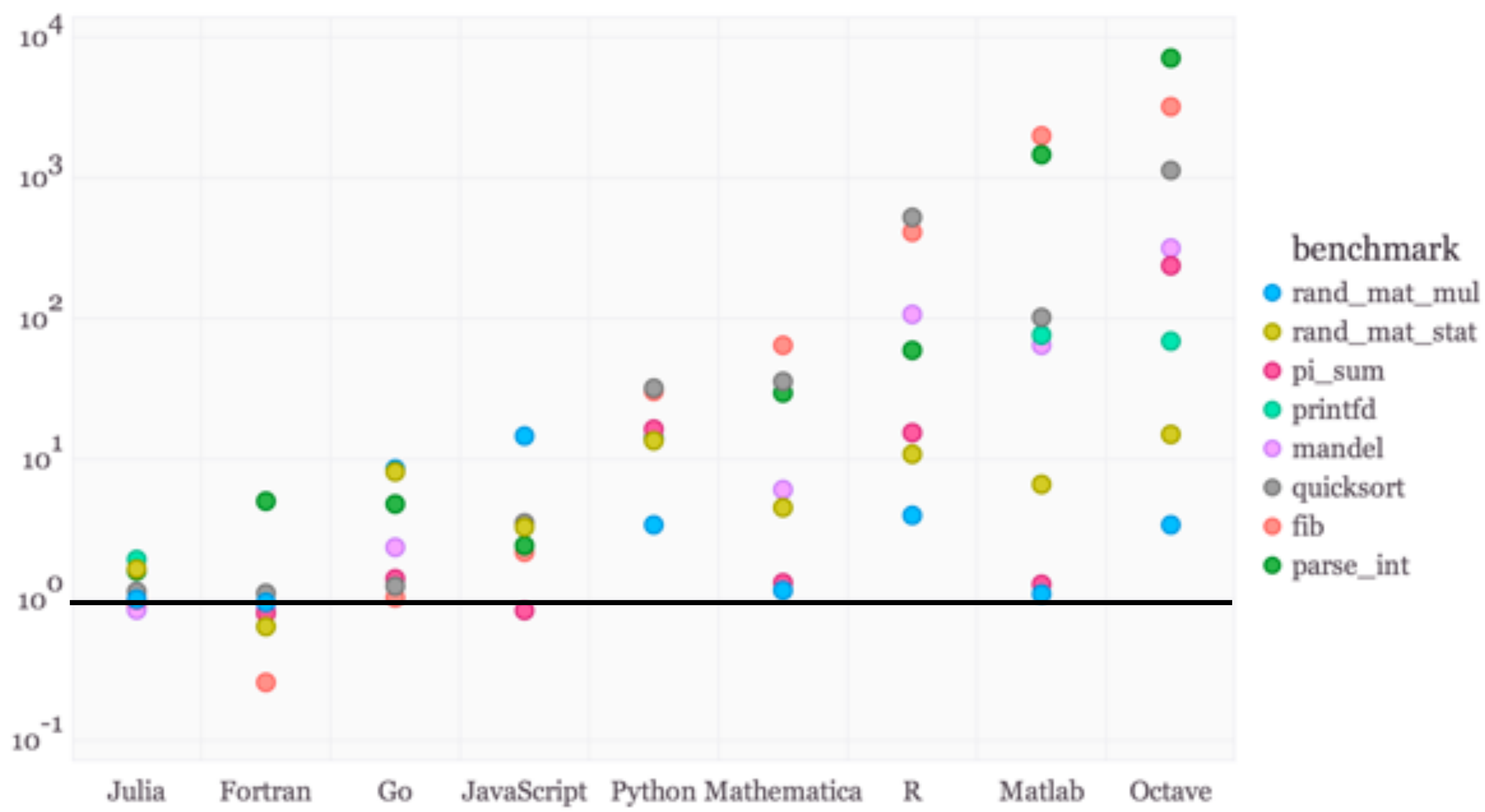
- Dynamic vs Fast: the usual tradeoff
- PL quality: more subjective. can you define more than 1 function in a file? do you have a module system? do operators and functions work in a consistent way?
- Julia aims to have all of them
- Ecosystem
- To the extent there's still a CS/Stats divide (or engineering/stats divide), you see it in R versus Matlab.

Big 3 usual numeric languages

Languages that Google has spent a zillion dollars to make fast

	Fortran gcc 4.8.1	Julia 0.2	Python 2.7.3	R 3.0.2	Matlab R2012a	Octave 3.6.4	Mathematica 8.0	JavaScript V8 3.7.12.22	Go go1
fib	0.26	0.91	30.37	411.36	1992.00	3211.81	64.46	2.18	1.03
parse_int	5.03	1.60	13.95	59.40	1463.16	7109.85	29.54	2.43	4.79
quicksort	1.11	1.14	31.98	524.29	101.84	1132.04	35.74	3.51	1.25
mandel	0.86	0.85	14.19	106.97	64.58	316.95	6.07	3.49	2.36
pi_sum	0.80	1.00	16.33	15.42	1.29	237.41	1.32	0.84	1.41
rand_mat_stat	0.64	1.66	13.52	10.84	6.61	14.98	4.52	3.28	8.12
rand_mat_mul	0.96	1.01	3.41	3.98	1.10	3.41	1.16	14.60	8.51

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



Why is it fast?

- Language design and smart use of LLVM
- [notebook]
- Don't have to vectorize everything!
  - Matlab/R/NumPy have taught us wrong
    - And it's a bad paradigm for structured cases, e.g. in NLP
  - e.g. Wasteful temporary allocations
    a+b+c+d


```
f(x) = x + 5
code_native(f, (Int,))
```

Other stuff

- Multiple dispatch
- Parallelism
- Metaprogramming (homoiconic, macros...)
- Calling C


```
julia> f(x::Int, y::String) = 10
julia> f(x::String, y::String) = 20
julia> f(3,"asdf")
10
julia> f("qwer","asdf")
20
```

Community

- Many developers, active mailing lists & responsive github issues
- Package system (200+ currently)

ApproxFun Arduino ArgParse ASCIIPlots AWS Benchmark BinDeps BioSeq Blocks BloomFilters BSplines Cairo Calculus Calendar Cartesian Catalan Cbc ChainedVectors ChemicalKinetics Clang Clp Clustering ClusterManagers Codecs Color Compose ContinuedFractions Cosmology Cpp CRC32 Cubature CUDA Curl DataFrames DataStructures Datetime Debug DecisionTree Devectorize DICOM DictUtils DimensionalityReduction DiscreteFactor Distance Distributions Docker DoubleDouble DualNumbers DWARF ELF Elliptic Example ExpressionUtils FactCheck FastaIO FileFind FITSIO FunctionalCollections FunctionalUtils Gadfly GARCH Gaston GeneticAlgorithms GeoIP GetC GLFW GLM GLPK GLPKMathProgInterface GLUT GnuTLS GoogleCharts Graphs Grid GSL Gtk Gurobi GZip Hadamard HDF5 HDFS Homebrew HopfieldNets HTTP HTTPClient HttpCommon HttpParser HttpServer HyperLogLog HypothesisTests ICU IJulia Images ImageView ImmutableArrays IniFile Ipopt IProfile Iterators Ito JSON JudyDicts JuliaWebRepl JuMP KLDivergence kNN Languages LazySequences LibCURL LibExpat LIBSVM LightXML Loess Loss MarketTechnicals MAT Match MathProgBase MATLAB MATLABCluster MCMC MDCT Meddle Memoize Meshes Metis MinimalPerfectHashes MixedModels MixtureModels MLBase MNIST Monads Mongo Mongrel2 Morsel Mustache Named NetCDF Nettle NHST NIfTI NLopt NPZ NumericExtensions ODBC ODE OpenGL OpenSSL Optim Options PatternDispatch Phylogenetics PLX Polynomial ProfileView ProgressMeter ProjectTemplate PTools PyCall PyPlot PySide Quandl QuickCheck RandomMatrices RDatasets RdRand Readline Regression REPL REPLCompletions Resampling Rif Rmath RNGTest RobustStats Roots SDE SDL SemidefiniteProgramming SimJulia Sims SliceSampler SMTPClient Sodium SortingAlgorithms Soundex SQLite Stats StrPack Sundials SVM SymPy Terminals TextAnalysis TextWrap TimeModels TimeSeries Tk TOML TopicModels TradingInstrument Trie Units URIParser URITemplate URLParse UTF16 UUID ValueDispatch Vega WAV WebSockets WinRPM Winston WWWClient YAML ZipFile Zlib ZMQ

R-style data analysis

- Plotting
  - **Gadfly** (*ggplot* grammar of graphics-style): http://dcjones.github.io/Gadfly.jl/
  - **PyPlot**: interface to Python's matplotlib

- **DataFrames** http://juliastats.github.io/DataFrames.jl/
  - Split-combine-apply, missing values, etc.

demo

Statistics - a few libraries

- In-progress overview: https://github.com/JuliaStats/Roadmap.jl/issues/1
- **Distributions**: sampling, moments, MLE, conjugate updates
- **GLM**: linear mixed-effects regressions models
- **MCMC**

Optimization juliaopt.org

- **JuMP** - An algebraic modeling language for optimization problems
- **Optim.jl** - Implementations of standard algorithms in pure Julia
- Interfaces to external solvers

Julia Solver Interfaces										
Solver	Julia	JuMP	LP	MILP	QCQP	MIQCQP	SDP	NLP	MINLP	Other
COIN Cbc/Clp	Cbc.jl/Clp.jl	✔	✔	✔						
GNU GLPK	GLPK.jl	✔	✔	✔						IP Callbacks
Gurobi	Gurobi.jl	✔	✔	✔	✔	✔				IP Callbacks
Ipopt	Ipopt.jl	✔1	✔2		✔2			✔		
NLopt	NLopt.jl		✔2		✔2			✔		

JuMP library

```
using JuMP

m = Model()
@defVar(m, 0 <= x <= 2 )
@defVar(m, 0 <= y <= 30 )

@setObjective(m, Max, 5x + 3*y )
@addConstraint(m, 1x + 5y <= 3.0 )

solve(m)
```

- Calls out to external solvers
- Macros and metaprogramming make it easier to develop specialized languages

MCMC library

- Metaprogramming gives expression parsing, supports autodiff for Hamiltonian MC

```
ex = quote
        vars ~ Normal(0, 1.0)
        prob = 1 / (1. + exp(- X * vars))
        Y ~ Bernoulli(prob)
end

m = model(ex, vars=zeros(nbeta), gradient=true)

# run random walk metropolis
mcchain01 = run(m * RWM(0.05) * SerialMC(1000:10000))

# run Hamiltonian Monte-Carlo
mcchain02 = run(m * HMC(2, 0.1) * SerialMC(1000:10000))
```


this is L2-regularized logreg
demo of quoting...
:(x + y * z)
dump(:(x + y * z))

	Core properties			Ecosystem		
	Dynamic and math-y?	Fast?	PL quality?	Open-source	Stat libraries (viz, regul/Bayes, regression...)	Engr libraries (optimization, signals...)
C/C++/ Fortran/Java	−	+	+	+	−	−
Matlab	+	−	−	−	−	+
Num/SciPy	+	−	+	+	−	~
R	+	−	−	+	++	−
Julia	+	+	++	+	underway	underway

The core language and standard library work very well right now.

The greater ecosystem of libraries is not yet mature, but advancing (frighteningly) rapidly. Comparability to R will surely take years.

- Dynamic vs Fast: the usual tradeoff
- PL quality: more subjective. can you define more than 1 function in a file? do you have a module system? do operators and functions work in a consistent way?
- Julia aims to have all of them
- Ecosystem
- To the extent there’s still a CS/Stats divide (or engineering/stats divide), you see it in R versus Matlab.

Links

- http://julialang.org/
- http://julialang.org/blog/2013/03/julia-tutorial-MIT/
- http://datacommunitydc.org/blog/2013/07/a-julia-meta-tutorial/
- Some stuff here I literally found last night from mailing list discussions
  https://groups.google.com/forum/#!forum/julia-stats
  https://groups.google.com/forum/#!forum/julia-users
  https://groups.google.com/forum/#!forum/julia-dev