What Julia Can and Can't Already Offer Statisticians

John Myles White

JSM 2016

This Talk

- Some of the main sub-problems in statistical computing
- Julia's current solutions to these sub-problems
- Where is Julia going?

I/O

```
iris <- read.csv("iris.csv")
```

Julia's State of the Art

- DataFrames.jl most popular I/O library
- CSV.jl most promising I/O library

Operations on Data Tabular

```
iris %>%
    group_by(Species) %>%
    summarize(
        Sepal.Length = mean(Sepal.Length),
        Sepal.Width = mean(Sepal.Width),
        Petal.Length = mean(Petal.Length),
        Petal.Width = mean(Petal.Width)
    )
```

Julia's State of the Art

- DataFrames.jl most popular tabular data library
- Tables.jl and jplyr.jl most promising (in-development) libraries

EDA and Data Visualization

```
ggplot(
    iris,
    aes(
        x = Sepal.Length,
        y = Sepal.Width,
        color = Species
    )
) +
    geom_point()
```

Julia's State of the Art

- Gadfly.jl and Plots.jl most popular data visualization libraries
- Big divide between statistical graphics and other needs
  - See GLVisualize.jl for a *very* different approach

Table-to-Array Transformations

```
iris_ext <- transform(
    iris,
    Is.Setosa = as.numeric(Species == "setosa")
)

X <- model.matrix(
    Is.Setosa ~ Sepal.Width + Sepal.Length,
    iris_ext
)

y <- iris_ext$Is.Setosa
```

Julia's State of the Art

- DataFrames.jl provides very basic model matrix support
- Upcoming changes to DataFrames.jl will improve support
- Table-to-array transformations will eventually be pulled out

Array Operations

```
M <- matrix(rnorm(100), nrow = 10, ncol = 10)
t(M)
svd(M)
```

Julia's State of the Art

- Julia's core library is dominated by array functionality
- Much of Julia's syntactic sugar is focused on array operations
- Julia's array support may already be superior to R's

Statistical Modeling

```
fitted_lm <- lm.fit(X, y)
```

Julia's State of the Art

- GLM.jl most popular linear regression library
- Big divide between statistical modeling and machine learning
  - See GLMNet.jl for a *very* different approach

Overall Review

- I/O: ~50% of R's usability
  - Less optimized, fewer supported formats
- Operations on Data Tabular: ~25% of R's usability
  - Much worse performance than dplyr and worse interface
- EDA and Data Visualization: ~50% of R's usability
  - Fewer features and worse performance

Overall Review

- Table-to-Array Transformations: ~25% of R's usability
  - Fewer features and worse performance
- Array Operations: >100% of R's usability
  - More functionality and much better performance
- Statistical Modeling: It Depends
  - GLMNet.jl often outperforms the Fortran library behind R
  - But many libraries are still missing

Why Such a Mixed Comparison?

- Why is Julia better today than R for arrays, but worse for tables?
- 1. Social factors
  - Original benefactor is a linear algebra expert at MIT
  - Original design was heavily based on Matlab
  - Many contributors come from array-oriented languages
- 2. Technical factors
  - Julia's type-inference system is ill-suited to R idioms
  - But original tabular libraries emulated R closely

How Will Julia Improve?

- Improvements to the core language's compiler
- Idioms will be made more amenable to run-time type-inference

The Road to Julia 1.0

- Vectorization
- Higher-order programming

Vectorization

```
x = randn(1_000_000)
x .= sin.(cos.(x))
```

Higher-Order Programming

```
x, y = randn(1_000_000), randn(1_000_000)
z = map((x_i, y_i) -> x_i^2 / y_i^2, x, y)
@select(tbl, col3 = col1 + col2^2)
```

Questions?