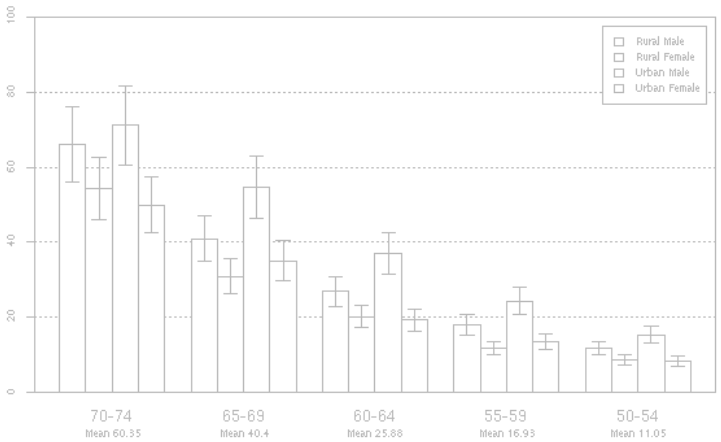
REVOLUTION ANALYTICS

Data Analytics and Dynamic Languages



Lee E. Edlefsen, Ph.D.
VP of Engineering

Overview

- This is my perspective on the use of dynamic languages (interpreters) for data analytics (statistics)
- I am a long-time user and commercial developer of dynamic languages for data analysis, but I am also a hard-core C++ programmer

Outline

- A bit about my experience
- Good/bad things about dynamic languages for data analysis, with special focus on R
- Why statistics/data analytics requires its own language
- The importance of parallel and distributed computing for data analysis
- What I think would be the ideal language for statistics

Dynamic languages I have used and/or developed

- **APL** – tried using to teach econometrics with it in early 1980's
- **MATLAB** – tried using a very early version to teach econometrics in the early 80's
- **Gauss** – designed, developed, and commercialized this matrix-oriented statistical system in mid-80's

Dynamic languages I have used and/or developed -- 2

- **Axum** – designed and developed this commercial technical graphics package, and wrote a C-like interpreter for doing data transformations
- **S-PLUS** – was VP of Development for S-PLUS (at MathSoft) in late 90's. S-PLUS is the commercial version of S.

Dynamic languages I have used and/or developed -- 3

- **ExaStat** – designed and developed this C++ based data analysis system that can be both interpreted (using CINT) and compiled
  - Threading is built in for most array and matrix computations
  - Includes a framework for automatically parallelizing and distributing a broad class of statistical algorithms
- **R** -- currently VP of Engineering at Revolution Analytics, which focuses on R

Features that make dynamic languages so nice for data analysis

- Command line for quick experimentation
- Ability to work directly with arrays, sets of arrays, and matrices
- Environment for inspection of objects
- Availability of functions for doing common operations
- Natural syntax that corresponds to subject matter – don't have to be a hard core programmer to get things done

More nice features

- Reduced need to write loops
- Don’t have to worry about specifying data types
- No compile/link/load time
- No worries about memory allocation, pointers, headers, linking errors, loading problems

Problems with most dynamic languages for data analysis

- Speed, especially compared with the best compiled code
- Loops tend to be especially slow
- Problems scaling with data size
- Often requires translation into compiled code before use in production environment
- Insufficient range of data types
- Lack of support for parallel and distributed computing

Especially nice features of R

- Common language used by practitioners around the world
- Much of new statistics over the past 10-15 years has been implemented in R
- Easy ability to download and install new packages; thousands are available
- Excellent interface to compiled languages
- Functions available for doing almost anything data-related

Particular problems with R

- Slow, especially for loops over data (about 100,000 times slower than C++ for simple loops)
- Memory hog; multiple copies of “read-only” data can be made during a simple analysis
- Not enough data types
- Encourages bad coding practices – especially use of globals
- Not thread-safe

Why statistics needs its own language(s)

- A natural and familiar syntax is important
- It is necessary to have “data set” objects with column and row names, data descriptions, mixed data types
- Easy, fast I/O of these objects
- Missing value handling is crucial and must be built-in to almost all functionality
- Proper handling of categorical data is also critical

The importance of parallel and distributed computing in data analysis

- Our ability to collect and store data is rapidly and greatly outpacing our ability to analyze that data
- To analyze all this data we must use multiple cores and multiple hard drives
- This means we need software that distributes computations across cores and computers and puts the results back together as if the work had been done on one core

My view of the ideal statistical language

- Based on the R syntax, but fixing the main problems
- Implemented using LLVM or something similar
- Extended range of data types
- Allow passing objects by reference
- Perhaps allow type specifications to enable increased speed of loops over data
- Have built-in support for parallel and distributed computations