Developing a "Next Generation" Statistical Computing Environment

Ross Ihaka
University of Auckland

Duncan Temple Lang
University of California

Talk Outline

This talk will seek to answer a number of questions about statistical software systems:

- Where are we now?
- Where would we like to be?
- How do we get there from here?
- What are the implications of reaching our goals?

Current Systems

- Statistical Packages
  - *SAS*, *SPSS*, *STATA*, . . .
- Statistical Programming Environments
  - *LISPSTAT*, *R*, *S*, *SPLUS*, . . .
- Programming Environments
  - *MATLAB*, *OCTAVE*, *OmegaHat*, *Python*,. . .
- General Purpose Programming Languages
  - *C*, *C++*, *C#*, *Fortran*, *Java*, *Lisp*, *Scheme*, . . .

Programming Environments

- Programming environments have become increasingly popular over the past 10 to 15 years.
- They are characterised by a having a flexible programming (or scripting) language with a large amount of functionality added through library-like extensions.
- Generally, these environments obtain their flexibility by being based on interpreters.
- Programming environments are easier to use than general programming languages, but there is a performance hit because they are interpreted.

S-Like Programming Environments

- The S family of languages includes *R* and *SPLUS*.
- The family members all have:
  - Call-by-value semantics
  - A whole-object model of computation
  - Vectorised capabilities
  - Generic-function based object systems
  - A very large library of statistical applications
- The languages are implemented as slow interpreters, but critical computations are implemented in C or Fortran.

Problems with the S Family

- The whole-object model limits the size of problem which can be handled. This is compounded by the use of call-by-value semantics which can produce multiple copies of data sets in memory.
- It also leads the use of algorithms which require the presence of entire data sets.
- The emphasis is very much on vector operations. In fact the family does not include the notion of scalar quantities, meaning the overhead of array access is unavoidable. This makes it difficulty to implement many computations efficiently.

The Need for a New System

- The computational model used in the S family has its origins in the mid 1970s.
- Advances in computing technology have kept software using the model useful, but more and more problems lie beyond the scope of what it can handle.
- In order to handle a large range of the current range of problems of interest, the basic computational model needs to be changed.

Obstacles to Evolution

- In the beginning, R provided a useful platform for experimentation and research.
- As a mature system with a large user base, such experimentation is necessarily curtailed.
- While useful work continues, substantive changes (e.g. anything involving a change to the evaluation model) would not be welcomed by the user base.
- Real change is only likely to come through revolution rather than evolution.

Features for a New System

- Speed
- Smaller (relative) footprint
- Foreign function interface
- Parallelism
- Support for reasoning about code

Candidate Systems

- We don't have the resources to build a new system entirely from scratch. We need some giant shoulders to stand on.
- Two candidate systems are:
  - Python
  - Lisp
- Benchmarking shows that (compiled) Lisp is better for array-based computations.
- We also think that Lisp has features which will ultimately make it a better data analysis language.

Lisp as the Base for New System

On the plus side:

- Lisp is a well-established, widely-used system
- There are a multiplicity of high-quality implementations
- There are very good resources explaining Lisp at both the high and low levels.

On the minus side:

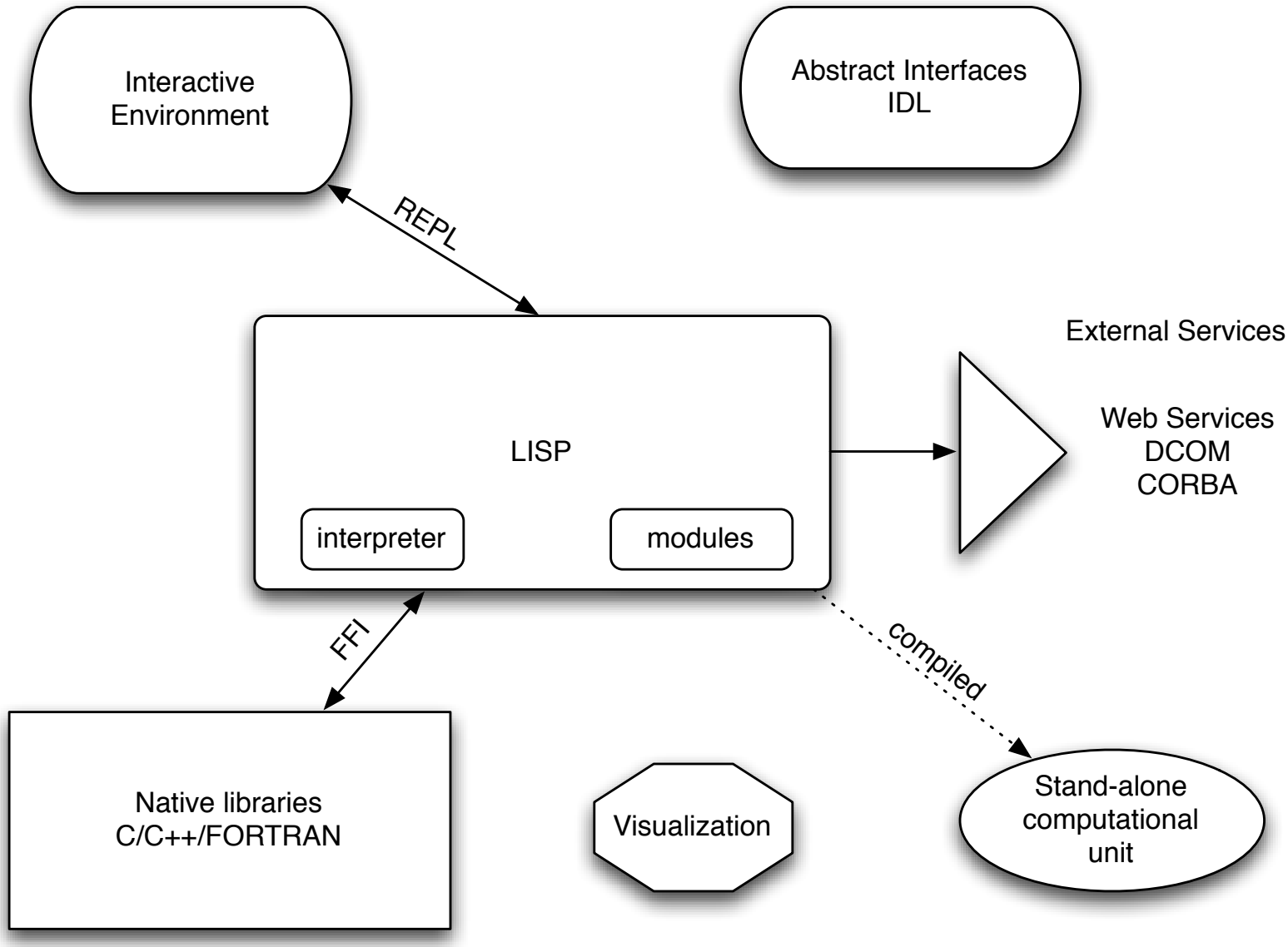
- Lisp has an image problem – it is perceived as a "dead" language.
- Because Lisp is an amalgam of features there are some inelegances to deal with.

Particular Lisp Features of Interest

- Compilation to machine code
- Optional type-declaration
- Pass-by-reference semantics
- A flexible generic-function based object system
- Macros
- Support for computing on the language.

A System Vision

- We envision Lisp lying at the heart of a rich computational environment.
- On top there is an interactive language which is really just a thin syntax layer over the Lisp system.
- Desirable Lisp features such as the FFI, and existing interfaces can be exploited directly.
- Commercial tree-shaking technology can be used to produce small stand alone applications.


Interactive
Environment
Abstract Interfaces
IDL
REPL
LISP
interpreter
modules
External Services
Web Services
DCOM
CORBA
FFI
compiled
Native libraries
C/C++/FORTRAN
Visualization
Stand-alone
computational
unit

Implementation Issues

- Syntax
- Lisp idiosyncrasies
- The object model
- Access to standard technologies

Development Time-Frame

- At present we are very much exploring the feasibility of building a new system.
- We feel that it is prudent to invest a good deal of time in understanding the technology we are proposing to adopt.
- Building a basic computing language can be relatively quickly, but there are other tasks which may take longer.

The Implications of a New Environment

- The environment we envision would be rather richer than current statistical computing environments.
- For statisticians to use the full capabilities of such an environment will require them to rethink the way in which they use computing.
- This will require a higher level of awareness and better education about the computational sciences.
- Doing this may well require changes to educational programs in statistics.

Summary

- This talk represents a progress report on an investigation of how we might go about producing a new statistical computing environment.
- Preliminary experiments have indicated that building a system on top of Common Lisp provides a productive direction to go in.
- We believe that systems based on Lisp will be both more sophisticated and better performing than present systems.