Thoughts on R development and the future

Duncan Temple Lang
UC Davis

Topics

- Extensibility of the kernel/core to facilitate experiments
- Compiler tools in R to allow different compilation approaches & experiments.
- High-level DSLs for big data analysis.
- Social process of developing & integrating alternative implementations into the community.
- Desired Features.

Sustainability

- R has been amazingly successful (both technically and community-wise).
- Could we have done better?
- Luxury for “statistics” to own its own interpreter, system, language.
- R-core spends a lot of time implementing facilities in other systems (UTF8, parallelism).
- Delay in availing of this new functionality.

- Increasingly more important to integrate other communities (ML, PL) and not just “statisticians”.
- Foster existing community, and new opportunities & relevance.
- Especially important when statistics doesn’t have good computational students.

Extensibility

- Important limitation is that it is hard to make changes and have them distributed with R.
  - focus on user space - packages, not kernel.
- Sociology of accepting enhancements/patches
- Unfulfilled opportunities for others to either participate or compete with new systems.

Compilation Tools in R

- As an alternative to having a byte-code compiler tightly coupled with a VM, explore LLVM
- RLLVM package provides functions to create IR directly with R calls,
  - either compiling R code or some other DSL.
- Let LLVM do all the work and generate native code
  - for CPU, GPU and different targets (JavaScript).
- Goal is to allow others to explore things within current R.

- RLLVMCompile is a very simple-minded translator of R expressions into LLVM IR elements.
- Then compile and optimize.
- E.g. 2D Random Walk
- Written in very naieve way for R (no vectorization)

R function

```
rw2d1 =
function(n = 100) {
    xpos = ypos = numeric(n)
    for(i in 2:n) {
      delta = if(runif(1) > .5) 1 else -1
      if (runif(1) > .5) {
        xpos[i] = xpos[i-1] + delta
        ypos[i] = ypos[i-1]
      }
      else {
        xpos[i] = xpos[i-1]
        ypos[i] = ypos[i-1] + delta
      }
    }
    list(x = xpos, y = ypos)
  }
```

Timings

	Time	Speedup
Interpeted	302.488	1.00
Byte Compiled	203.226	1.48
Vectorized	1.549	195.27
Rllvm	0.641	471.90

- (Aug 2012, R 2-16-devel)

- User specifies types for variables
  - potentially annotate the function with these via TypeInfo package
  - or type inference
  - type information beneficial for other purposes.
- Can indicate whether there are NAs or not.
- Whether data is mutable or not

Potential

- Introduce new data types, e.g. trees, bignums, big arrays.
- Generate wrappers to 3rd party code (or use dynamic FFI)
- Analyze code to identify dead variables, garbage collect
- Perhaps recognize potential for memory reuse across segments of scripts.
- Recognize data distribution patterns so transfer subsets to different nodes and execute multiple operations.
- CodeDepends package helps to identify code flow in R.

DSLs

- Instead of users writing procedural code, perhaps they can declare things about the data analysis and have that be compiled/interpreted.
- Combine model + fitting algorithm + parallelism strategy + sub-sampling
- Opportunity because we are in a quite specific domain.
- Say what you want, not low-level computations that lose the big picture.

- R formula language
  - Very different abstraction from model/design matrix
  - Model description object. Unconnected with data & fitting method.
  - Combine model with fitting algorthim
  - Can predict new data, update model, etc.
  - FastLab - Alexander Gray (Georgia Tech)
- Similarly, extended formula language for lattice/trellis plots
  wireframe(y ~ x1 + x2 | z, data)

- Bayesian tools this approach
  - BUGS (Bayesian MCMC) uses this approach.
  - NIMBLE (Paciorek, DeValpine, DTL)
  - Stan (Gelman et al.)
- PMML represents models (and results, etc.)

Big Data DSLs

- Sampling language
  - to describe complex sampling schemes for sub-samples, bootstrap, etc.
  - Perhaps survey package already has this.
- Language for indicating how to distribute data and computations.
- Goal is to allow descriptions of computations to be used elsewhere and in future systems.
- Don't have to be languages, just high-level descriptions as objects.

- Goal is to allow people to create composite algorithms without programming, i.e. reuse different steps.
- Users can still program with general purpose language, but rewarded for not.
- Implementors of the pieces can use high-level descriptions that are compiled, or use general language.

Integrating New Implementations

- Some projects outside of the R community have created modified R implementations that are not maintained.
- CXXR has very nice features, but minimal uptake.

Desiderata

- More/better facilities for developing software
  - optional type specification
  - interface/contract
- Provenance and Reproducability
- Caching and updating results.
- Streaming data/block updating algorithm paradigm
- Approximate results
- Embedding in other systems (databases, languages Web browsers)
- Security
- Compile to stand-alone applications

- Need to seriously consider a plan to adopt/integrate/combine/coexist different implementations, enhacements.
- Sustain and maintain the computing environment for community.
  - partner long-term volunteers with shorter term researchers.
- Try to plan for the inevitable changes that will continue to come - both technical and social.