CXXR and Add-on Packages

Andrew Runnalls[1]

1. School of Computing, University of Kent, UK, A.R.Runnalls@kent.ac.uk

Keywords: R, CXXR, C++, packages, CRAN

CXXR (www.cs.kent.ac.uk/projects/cxxr) is a project to refactor (reengineer) the interpreter of the R language, currently written for the most part in C, into C++. It is hoped that by reorganising the code along object-oriented lines, by deploying the tighter code encapsulation that is possible in C++, and by improving the internal documentation, the project will make it easier for researchers to develop experimental versions of the R interpreter.

The design of CXXR endeavours to reconcile three objectives:

- Above all, to be functionally consistent with standard R, both at the R language level, and at the C/Fortran package interface level.
- For the core of the interpreter to be written in idiomatic, standards-conforming C++, making best use of the C++ standard library, and providing a well documented C++ API on which C++ package writers can build.
- To provide a reasonably simple mechanism for CXXR to be upgraded to parallel the continuing evolution of standard R.

Development of CXXR started in May 2007, then shadowing R 2.5.1; at the time of this abstract it reflects the functionality of R 2.10.1. At useR! 2009 Chris Silles described an offshoot project to introduce provenance-tracking facilities into CXXR, so that for any R data object it will be possible to determine exactly which original data files it was derived from, and exactly which sequence of operations was used to produce it: in other words, an enhanced version of the old S AUDIT facility.

In principle any R add-on package should work without alteration under CXXR, provided it conforms to the `R.h` or `S.h` APIs. (Code using `Rinternals.h` may need alterations, usually minor, as explained in the CXXR documentation.) The primary purpose of this paper—after giving a general update on the CXXR project—is to gauge to what extent this is true in practice, by describing the author's experiences in installing and testing under CXXR a number of the most widely used packages from CRAN. A particular observation will be how CXXR can quickly bring to light memory-protection errors (i.e. incorrect use of `PROTECT()`, `UNPROTECT()` etc.) that may long lie dormant under the standard R interpreter.

The paper will go on to explain how CXXR offers the prospect of making life simpler for package writers incorporating native C/C++ code, and allowing—in a controlled way—closer interaction between package code and the underlying interpreter. For example, the following are already feasible:

- Direct access to the underlying garbage collection system *via* a well-documented and well-encapsulated API.
- In CXXR the `SEXPREC` union is replaced by a C++ class hierarchy. Package writers can extend this class hierarchy as they see fit, rather than needing to use external pointers and finalizers. In particular, new R classes can be wrapped around new C++ classes within the hierarchy.
- Instead of using `PROTECT()` and kindred functions, package writers can use C++ smart pointers which afford memory protection to whatever they point to. This is much simpler and less error-prone than the `PROTECT()` mechanism.

These points will be illustrated by showing how the **ff** package can be reengineered under CXXR. (Admittedly, these facilities come at the expense of compatibility with the standard R interpreter.)

References

Chris Silles and Andrew Runnalls (2009). Provenance Tracking in CXXR, `http://www.agrocampus-ouest.fr/math/useR-2009/slides/Silles+Runnalls.pdf`.

Daniel Adler et al. (2007). ff: memory-efficient storage of large data on disk and fast access functions, `http://cran.r-project.org/web/packages/ff/index.html`.