A high-performance compiler for a subset of R

Ansgar Steland⋆

RWTH Aachen University
⋆Contact author: steland@stochastik.rwth-aachen.de

Keywords: Concurrency, Compiler, Parallel Computing, Statistical Computing, Simulation, Threads

As the de-facto standard for statistical computing and programming, R is heavily used for computationally demanding tasks ranging from the analysis of huge and high-dimensional data sets arising in genetics, econometrics or environmetrics to large-scale stochastic simulation. Despite the progress made, the performance of R can be rather poor when it comes to statistical computations with numbers, i.e. *number crunching*, which is a consequence of the general concept behind R. This particularly applies when computations involve new algorithms which do not reduce to some calls to build-in functions. Another issue is that R has no simple concept for parallel computing which would allow us to exploit the present day multi-core computers effectively.

The common approach to overcome this drawback is to write the time-critical parts (functions) in C and to invoce them from R. This approach often provide optimal speedups, but it has several drawbacks. First, it is no solution for R users not trained in a language such as C. Second, developing C implementations of complex algorithms arising in present day statistical research is often difficult, time consuming, thus expensive, and also error-prone. Third, one has to waive all the elegant and powerful concepts R offers, particularly matrix calculus and lists, or map them to rather cryptic C function calls and pointer arithmetics. The project P allows for an intermediate way where crucial language features are available and the code runs substantially faster, since it is compiled. Indeed, the compiler results in tremendous speedups in many cases.

The compiler implements a dialect of a subset of the S/R language enlarged by various extensions yielding a language called P. The available subset covers the most important data types for mathematical computing (double precision reals, vectors, matrices and lists) and basic operations for them. Loops, if-then-else, while, do-while and functions (with recursion) are available as well. P runs on common 32bit as well as 64bit platforms including OS X, Linux and Windows, without the need to have installed additional programs or libraries. Since the compiler can be embedded in R, source code of P and R can be mixed in one source file. Alternatively, a portable binary file can be generated which is then executed by the runtime system. Although P differs to some extent from R, porting R functions dealing with mathematical computations usually requires only a few trivial changes. Further, porting C code is supported by a `cstyle` environment which, among other issues, changes the indexation of vectors and matrices from $1..n$ to $0 \dots n-1$.

P also implements various extensions of the R/S language, e.g. pointers allowing for call by reference, which can speed up programs substantially and is important when handling large data structures. More importantly, we implemented a powerful and easy to use approach for parallel computing. Indeed, many computations in statistics can easily be parallelized. P has a `thread` statement which allows us to create concurrent shared memory threads. Local threads initiated by a function can share local variables or can use them as private ones, e.g. as loop variables, and can even recursively call the function which created them. Hence the powerful divide-and-conquer approach for parallelization is at our disposal, which can greatly simplify the parallelization of existing programs. Additional extensions (e.g. `inf`) simplify coding certain statistical algorithms and mathematical formulae as arising, e.g., in sequential analysis.

The project is work in progress of the author (during his spare time), but is now rather settled and no longer what is called a *proof-of-concept* in computer science.

References

Steland, A (2010). P: A compiled language, *under preparation.*