Performance Analysis for R: Towards a Faster R Interpreter

Helena Kotthaus*, Ingo Korb, Markus Künne, Peter Marwedel

Department of Computer Science 12, TU Dortmund University
*Contact author:helena.kotthaus@tu-dortmund.de

Keywords: Profiling, Performance Analyses, Machine Learning, Tools

The *R* language has a large set of dynamic features which allow for rapid development of new algorithms for statistical applications. However, this flexibility comes at a price: *R* is considered to be a rather slow language that needs a large amount of memory during runtime. Our goal is to resolve these performance problems. An indispensable prerequisite for this is having a more detailed view into the *R* interpreter's internals. For this reason we have developed a profiling tool [3] which enables a detailed analysis of runtime behavior and memory consumption of *R* programs.

As a basis for our profiling tool we have chosen the TraceR framework. TraceR was originally developed at Purdue University for *R* 2.12 [2]. Besides porting it to the current version of *R*, we also improved its usability and analysis capabilities. Since TraceR is directly integrated with the *R* interpreter, we can generate more detailed data compared to other existing profiling tools, such as Rprof. Rprof operates only at the *R* function level and as such does not provide information about the internals of the *R* interpreter or native code. Our new profiling tool, in contrast, can profile the execution time spent in C/Fortran code supplied by *R* packages, or timing characteristics of memory management tasks like garbage collection. Additionally, our tool is able to record the function call hierarchy. This is similar to the context stack feature of Rprof, but independent of the sampling rate. Moreover, it is capable of also recording internal operations of the *R* interpreter. The level of detail of this control flow information is highly configurable and can be changed interactively. The collected data is used to create a call graph which is annotated with other measurement results from our profiling tool like the invocation count.

Even though we originally developed our profiling tool for analyzing the bottlenecks of machine learning *R* programs [1], we believe that the feedback which can be generated with our tool is valuable to guide changes in the original *R* interpreter, as well as to support the development of alternative *R* interpreters. In this talk we will present our profiling tool and how to apply it to analyze the runtime behavior and the memory consumption of *R* programs.

References

[1] Helena Kotthaus, Michel Lang, Jörg Rahnenführer and Peter Marwedel: Runtime and memory consumption analyses for machine learning R programs. Statistical Computing, Schloss Reisensburg, Germany, 2013

[2] The Reactor Project: `http://r.cs.purdue.edu`, Purdue University, 2012

[3] Ingo Korb, Helena Kotthaus, Markus Künne: TraceR, `https://github.com/allr/tracer`, TU Dortmund University, 2014