As R is becoming increasingly more popular and widely used, two great challenges have emerged: performance and big data.

Increasingly more computation time is being spent in the R code as opposed to the numerical libraries. This has performance penalties and analysts often end up rewriting the hot spots of their R code to C/C++, which is time consuming and error prone. The current implementation of R has been around for about 2 decades (some parts of it nearly 4) and it would be very hard to extend it with today’s state-of-the-art optimizations such as those present in the C/C++ compilers. With hot spots being in the R code, the lack of parallelism in the R code is becoming a performance issue as well: current multi-core systems cannot be efficiently employed. Adding multi-threading to the R language would be hard within the current implementation.

R is being used for increasingly larger data. The data size limitations imposed by the use of 32-bit integers in the present R interpreter for encoding vector offsets are becoming a bottleneck on todays machines with large amounts of RAM. Data analysis these days and in the near future, however, needs to be done also on much larger data that would ever fit onto a single machine. Such data is typically stored in a cluster/cloud, often heavily cached in RAM of many nodes or even fully included in RAM of the nodes. Could R be made run in the cloud, evaluating parts of R expressions on the nodes where the data is?

We aim to attack these problems with a new R engine built on top of a Java virtual machine. The benefits we get from Java are good integrated support for multi-threading, a modern garbage collector, and a better integration with the cloud and databases. Choosing Java instead of say C++ brings also a number of challenges. A big challenge is accessing well proven numerical libraries implemented in C/Fortran, such as LAPACK/BLAS, but also the Rmath library and other numerical codes present in R. Accessing them from Java incurs installation burden and for short-running operations has a performance overhead. Converting them to Java is difficult and the resulting code is likely to be slower for large data, as has been reported for the automatically converted codes of LAPACK/BLAS. A similar challenge is the use of R packages, parts of which are again implemented in C or Fortran.

We will explain the status of the project, FastR, currently on small benchmarks. On these we have seen speedups between 2x and 15x over the latest version of the R interpreter. We will provide some thoughts about where to go from there.