



# Improvements of the RIR bytecode toolchain

Jan Ječmen<sup>1</sup>, supervised by Petr Máj<sup>1</sup>

<sup>1</sup>Faculty of Information Technology, Czech Technical University in Prague

Fork me on GitHub

## Introduction

This thesis implemented changes to the RIR project in order to enhance its performance.

- ▶ RIR is an alternative to the internal bytecode toolchain of the R language [1]
- ▶ RIR is intended to research speculative optimizations of R, for which a performant baseline must be established

## Work done

- ▶ Added fastpaths for frequently used operations, e.g.
  - ▶ relational operators
  - ▶ colon operator (generates integer sequences in R)
  - ▶ superassignment (R operator that assigns into parent environment)
- ▶ Tweaked the RIR compiler to better take advantage of compilation context (used to optimize away some non-local jumps in loops)
- ▶ Refactored the RIR interpreter to speed up instruction dispatching

## Conclusions & Future work

- ▶ Bytecode-intensive benchmarks sped up, cutting the deficiency vs. GNU R by about a half
- ▶ Future research:
  - ▶ Further improvements of RIR (e.g. call mechanism)
  - ▶ Static analyses of R, speculative optimizations and deoptimizations
- ▶ Work can be downloaded from [github.com/reactorlabs/rir](https://github.com/reactorlabs/rir)

## References

- [1] Luke Tierney. *A Byte Code Compiler for R*. 2016. URL: <http://homepage.divms.uiowa.edu/~luke/R/compiler/compiler.pdf> (visited on 07/29/2017).
- [2] Leo Oswald. *r-shootout. R version of language shootout game*. 2014. URL: <http://r.cs.purdue.edu/hg/r-shootout> (visited on 07/29/2017).

## Evaluation

The following figure shows the differences in speed brought by this thesis (the values are scaled relative to the speed of the standard GNU R implementation) on the Shootout benchmarks [2].

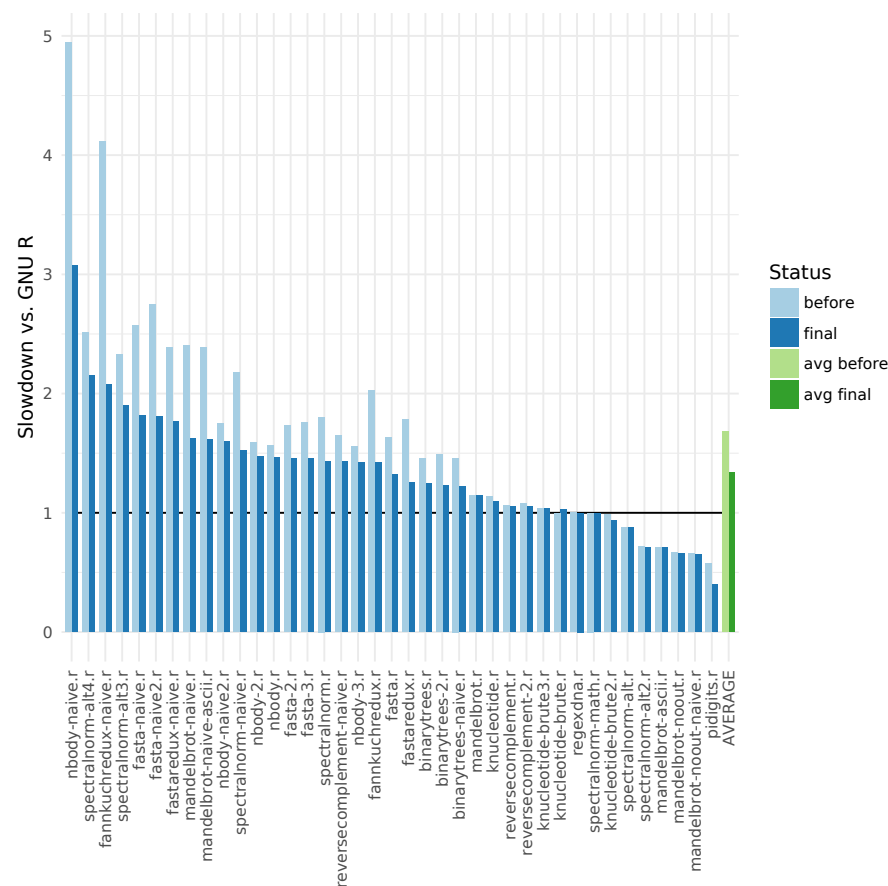


Figure 1: Results achieved on the Shootout benchmarks.