RCaller: A Java package for interfacing R

Mehmet Hakan Satman[1] and Kopilov Aleksandr[2]

1 Department of Econometrics, Istanbul University **2** Department of Mathematical Optimization and Modeling, BIA-Technologies

Summary

In this paper, version 3.0 of the open source library RCaller is introduced. RCaller is a software library which simplifies performing data analysis and statistical calculations in Java using R. The details are hidden from users including transferring data between platforms, function calls, and retrieving results. Addition to the previous revisions, RCaller 3.0 implements the scripting API of Java in which the R function calls and data transfers are performed in a standard way as in the way of calling other scripting languages in Java. Besides implementation of new features, RCaller has many performance improvements in the new release of 3.0 (Satman & Curcean, 2016).

State of the field

Nowadays, the interest on statistical computation and data analysis increased the popularity of some programming languages such as R, Python, and Julia. As a result of popularity, hundreds of packages are developed for these targeted languages. On the other side, many mainframe languages are adopted for more specific tasks in server-side or the general-purpose use in several platforms. Interaction between those languages became a major requirement due to the lack of modern statistical tools written in the general-purpose languages such as Java. Renjin (Bertram, 2013) is a great effort for interfacing R from within Java using a different perspective: Writing R from scratch in Java! GraalVM (Niephaus et al., 2019) is an other admirable attempt for interfacing Python and R in Java with its polyglot design. Many tools and software packages are developed for interfacing R and Java. Each one stands out with its different features including speed, installation procedures, learning speed, and ease of use.

Statement of need

RCaller is a Java library for interfacing R from within Java (Satman, 2014). R is a popular programming language and a programming environment with hundreds of packages written in C, C++, Fortran, and R itself (R Core Team, 2020). These huge collection of computation tools are not directly accessible for the other languages, especially for Java. RCaller supplies a clean API for calling R functions, managing interactions, and transfering objects between languages. There are other options in the literature including Rserve (Urbanek, 2003) and rJava (Urbanek, 2009) which are based on TCP sockets and JNI (Java Native Interface), respectively. RCaller provides a set of easier calling schemes without any dependencies. Previous works showed that the performance of the library is suitable for more cases and studies with moderate datasets can be handled in reasonable times (Satman & Curcean, 2016). After being introduced in paper (Satman, 2014), support for DataFrame objects, R start-up options, automatic Rscript

1

executable locator, and Java Scripting API (JSR 223) are implemented besides minor library changes and performance improvements. R has many well-tested and matured packages for automatic time series model selection, clustering, segmentation and classification, non-linear and robust regression estimations, data and text mining, linear and non-linear programming, generating plots, multivariate data analysis, and function optimization besides other research tools. RCaller brings all of the functionality that R serves in Java. RCaller is also used as a core of RCallerService (Kopilov, 2020), free microservice solution for running R-scripts from any other languages using HTTP.

Java scripting interface

RCaller also implements the scripting API of Java (JSR 223) after version 3 and above. In other words, the engine behaves like a Java implementation of R. With this feature of RCaller, calling R from Java is seemingly the same as in the other JSR 223 implementations such as JavaScript, Python, Groovy, and Ruby. However the performance is not directly comparable with native counterparts (Rhino, Jython, JGroovy, JRuby, etc.) since the target language is not reimplemented in Java. Here is an example of sorting a Java array in R side and handling the result in Java:

```java
ScriptEngineManager manager = new ScriptEngineManager();
ScriptEngine en = manager.getEngineByName("RCaller");

try {
    double[] a = new double[] {19.0, 17.0, 23.0};
    en.put("a", a);
    en.eval("sortedA <- sort(a)");
    double[] result = (double[])en.get("sortedA");
    System.out.println(result[0]);
} catch (ScriptException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
```

RCaller, as a scripting engine in Java, creates an R process, encodes Java objects to XML, runs commands in R side, gets back the results as XML, and parses the result to Java objects. Since a single R process is created and used for consecutive calls, XML parsing is the only calculation overhead.

Acknowledgements

References

Bertram, A. (2013). Renjin: A new r interpreter built on the jvm. *The R User Conference, useR! 2013 July 10-12 2013 University of Castilla-La Mancha, Albacete, Spain*, *10*, 105.

Kopilov, A. (2020). *RCallerService — microservice for running r-scripts by http*. https://github.com/Kopilov/RCallerService

Niephaus, F., Felgentreff, T., & Hirschfeld, R. (2019). Towards polyglot adapters for the graalvm. *Proceedings of the Conference Companion of the 3rd International Conference on Art, Science, and Engineering of Programming*, 1–3. https://doi.org/10.1145/3328433.3328458

R Core Team. (2020). *R: A language and environment for statistical computing*. R Foundation for Statistical Computing. https://www.R-project.org/

Satman, M., & Curcean, P. (2016). *RCaller 3.0: An easy tool for abstraction of java and r connectivity*. https://github.com/jbytecode/rcaller/blob/master/doc/rcaller3/rcaller3.pdf

Satman, M. H. (2014). RCaller: A software library for calling r from java. *Journal of Advances in Mathematics and Computer Science*, 2188–2196. https://doi.org/10.9734/bjmcs/2014/10902

Urbanek, S. (2009). How to talk to strangers: Ways to leverage connectivity between r, java and objective c. *Computational Statistics*, *24*(2), 303–311. https://doi.org/10.1007/s00180-008-0132-x

Urbanek, S. (2003). A fast way to provide r functionality to applications. *Proceedings of Dsc*, *2*.