JuliaCall: an R package for seamless integration between R and Julia

Changcheng Li

1 Department of Statistics, Pennsylvania State University at University Park

Summary

R is a widely used software and computing environment for statistics, which provides a variety of statistical techniques and packages (R Core Team, 2018). Many R packages provide interfaces to other computer languages to bring new functionalities to R or to accelerate computations, such as Rcpp (Edelbuettel & François, 2011) and V8 (Ooms, 2019). The package JuliaCall provides an R interface to Julia, which is a computer language for high-performance scientific computing (Bezanson, Edelman, Karpinski, & Shah, 2017). JuliaCall embeds Julia in R, and provides functions to evaluate Julia commands, to call Julia functions, to transmit data objects between R and Julia, and so on. It also provides many utilities for user convenience. For example, JuliaCall gives detailed error messages for the embedded Julia. It also provides Julia package management functions such as installation and loading, and utility functions to get the documentation of Julia functions. JuliaCall can also be used in R Markdown document as the engine of Julia language, see Section 2.77 in Xie, Allaire, & Grolemund (2018).

Some R packages wrap Julia packages based on JuliaCall to provide new functionalities or performance improvements to some existing packages in R. autodiffR (Li, 2018a) provides automatic differentiation to R functions by wrapping ForwardDiff.jl and ReverseDiff.jl. convexjlr (Li, 2018b) is an R package for Disciplined Convex Programming (DCP) providing a high level wrapper for Convex.jl. diffeqr (Rackauckas, 2018) solves differential equations in R using DifferentialEquations.jl. FixedEffectjlr (Loualiche, 2018) estimates large fixed effects models in R by providing an interface to FixedEffectModels.jl.

Besides JuliaCall, some other packages also provide interfaces between R and Julia: XRJulia, RJulia, and the Julia package RCall.jl. The package XRJulia connects to Julia from R (Chambers, 2017). It uses JavaScript Object Notation (JSON) format to transmit data, while JuliaCall copies objects in memory between R and Julia. It has performance disadvantages compared to JuliaCall. Table 1 depicts the times needed to transmit a 500 x 500 matrix full of ones from R v3.5.2 to Julia v1.0.3 using JuliaCall v0.16.4 and XRJulia Github master b6224fa at the time of writing (there is no released version of XRJulia to support Julia v1.0 yet). The times are measured by R package microbenchmark with 1000 evaluation times. The script with the benchmark code and setup instructions can be found in the paper directory in JuliaCall Github repository.

dependency for JuliaCall, and JuliaCall utilizes RCall.jl’s type conversion between R and Julia. JuliaCall integrates well with RCall.jl, and it is the default for JuliaCall to load RCall.jl in the embedded Julia automatically at starting. With JuliaCall and RCall.jl, it is easy to use R from Julia and Julia from R.

Table 1: Time measurements for XRJulia and JuliaCall to transmit a $500 \times 500$ matrix from R to Julia.

<table>
<thead>
<tr>
<th></th>
<th>Time in ms</th>
<th>min</th>
<th>lq</th>
<th>mean</th>
<th>median</th>
<th>uq</th>
<th>max</th>
</tr>
</thead>
<tbody>
<tr>
<td>XRJulia</td>
<td>24.680699</td>
<td>30.65626</td>
<td>39.29180</td>
<td>33.98684</td>
<td>38.83510</td>
<td>454.6484</td>
<td></td>
</tr>
</tbody>
</table>

Users can get stable releases of JuliaCall from CRAN, and the latest development version from JuliaCall Github repository. Documentation can be found in the package as well as on CRAN. Bug reports and other feedback can be submitted to GitHub issue page.

References


