Interoperability using R

Joining forces instead of separating

Anton Antonov (@tonytonov)
tonytonov@gmail.com

2019-04-14

Interoperability

Malone [2014] (emphasis mine – AA)

Language interoperability is the capability of two different programming languages to *natively interact* as part of the same system.

Interoperability

Malone [2014] (emphasis mine – AA)

Language interoperability is the capability of two different programming languages to *natively interact* as part of the same system.

Challenges:

- Object and memory models are different

Interoperability

Malone [2014] (emphasis mine – AA)

Language interoperability is the capability of two different programming languages to *natively interact* as part of the same system.

Challenges:

- Object and memory models are different
- Cross-debugging

Interoperability

Malone [2014] (emphasis mine – AA)

Language interoperability is the capability of two different programming languages to *natively interact* as part of the same system.

Challenges:

- Object and memory models are different
- Cross-debugging
- Performance overhead

Interoperability

Malone [2014] (emphasis mine – AA)

Language interoperability is the capability of two different programming languages to *natively interact* as part of the same system.

Challenges:

- Object and memory models are different
- Cross-debugging
- Performance overhead
- Cognitive effort due to increased complexity

Interop directions

Breakdown by interop scheme:

- R$\rightarrow$: use R inside other language

Interop directions

Breakdown by interop scheme:

- R$\rightarrow$: use R inside other language
- $\rightarrow$R: use other language inside R

Interop directions

Breakdown by interop scheme:

- R$\rightarrow$: use R inside other language
- $\rightarrow$R: use other language inside R
- Magic (next-gen VMs)

Historical perspective

- Matloff [2011], The Art of R Programming, Chapter 15

Historical perspective

- Matloff [2011], The Art of R Programming, Chapter 15
- Two examples: C/C++ from R and R from Python

Historical perspective

- Matloff [2011], The Art of R Programming, Chapter 15
- Two examples: C/C++ from R and R from Python
- First example: manual compilation, then `dyn.load`, then `.C`

Historical perspective

- Matloff [2011], The Art of R Programming, Chapter 15
- Two examples: C/C++ from R and R from Python
- First example: manual compilation, then `dyn.load`, then `.C`
- 1000x speedup for a simple time series prediction task

Historical perspective

- Matloff [2011], The Art of R Programming, Chapter 15
- Two examples: C/C++ from R and R from Python
- First example: manual compilation, then `dyn.load`, then `.C`
- 1000x speedup for a simple time series prediction task
- Second example: `rpy`, now `rpy2` for Python 3

How to use R

- Standard input/output

How to use R

- Standard input/output
- As a shared library (`R.dll`, `libR.so`, etc.)

How to use R

- Standard input/output
- As a shared library (`R.dll`, `libR.so`, etc.)
  - Library can be built from R source: https://github.com/wch/r-source/

How to use R

- Standard input/output
- As a shared library (`R.dll`, `libR.so`, etc.)
  - Library can be built from R source: https://github.com/wch/r-source/
  - Also called R API

How to use R

- Standard input/output
- As a shared library (`R.dll`, `libR.so`, etc.)
  - Library can be built from R source: https://github.com/wch/r-source/
  - Also called R API
  - Low-level, callable from C (or FORTRAN!)

R as API

Some examples from R Development Core Team [2011]:

R as API

Some examples from R Development Core Team [2011]:

- Any R object is of type `SEXP`, which is a pointer to a structure with `typedef SEXPREC`

R as API

Some examples from R Development Core Team [2011]:

- Any R object is of type `SEXP`, which is a pointer to a structure with `typedef SEXPREC`
- `order(..., na.last, decreasing)` is

R as API

Some examples from R Development Core Team [2011]:

- Any R object is of type `SEXP`, which is a pointer to a structure with `typedef SEXPREC`
- `order(..., na.last, decreasing)` is
  `void R_orderVector (int* indx, int n, SEXP arglist, Rboolean nalast, Rboolean decreasing)`

R as API

Some examples from R Development Core Team [2011]:

- Any R object is of type `SEXP`, which is a pointer to a structure with `typedef SEXPREC`
- `order(..., na.last, decreasing)` is
  `void R_orderVector (int* indx, int n, SEXP arglist, Rboolean nalast, Rboolean decreasing)`
- Random number generation, distributions

R as API

Some examples from R Development Core Team [2011]:

- Any R object is of type `SEXP`, which is a pointer to a structure with `typedef SEXPREC`
- `order(..., na.last, decreasing)` is
  `void R_orderVector (int* indx, int n, SEXP arglist, Rboolean nalast, Rboolean decreasing)`
- Random number generation, distributions
- Mathematical functions and constants

R as API

Some examples from R Development Core Team [2011]:

- Any R object is of type `SEXP`, which is a pointer to a structure with `typedef SEXPREC`
- `order(..., na.last, decreasing)` is
  `void R_orderVector (int* indx, int n, SEXP arglist, Rboolean nalast, Rboolean decreasing)`
- Random number generation, distributions
- Mathematical functions and constants
- Numerical optimization, integration

Rcpp

- An integration between R and C++ that is easy to use

Rcpp

- An integration between R and C++ that is easy to use
- Very popular among package authors to gain performance speedup

Rcpp

- An integration between R and C++ that is easy to use
- Very popular among package authors to gain performance speedup
- *Seamless*: provides access to all R objects

Rcpp

- An integration between R and C++ that is easy to use
- Very popular among package authors to gain performance speedup
- *Seamless*: provides access to all R objects
- Handles compilation and linkage, platform dependency, etc.

Rcpp

- An integration between R and C++ that is easy to use
- Very popular among package authors to gain performance speedup
- *Seamless*: provides access to all R objects
- Handles compilation and linkage, platform dependency, etc.
- Supports all modern features of C++11, C++14, C++17

Rcpp

- An integration between R and C++ that is easy to use
- Very popular among package authors to gain performance speedup
- *Seamless*: provides access to all R objects
- Handles compilation and linkage, platform dependency, etc.
- Supports all modern features of C++11, C++14, C++17
- Backed up by great package extensions: RcppArmadillo, RcppParallel, RcppMLPACK, ...

Rcpp

- An integration between R and C++ that is easy to use
- Very popular among package authors to gain performance speedup
- *Seamless*: provides access to all R objects
- Handles compilation and linkage, platform dependency, etc.
- Supports all modern features of C++11, C++14, C++17
- Backed up by great package extensions: RcppArmadillo, RcppParallel, RcppMLPACK, ...
- Great documentation, gallery, supported by RStudio

Rcpp

- An integration between R and C++ that is easy to use
- Very popular among package authors to gain performance speedup
- *Seamless*: provides access to all R objects
- Handles compilation and linkage, platform dependency, etc.
- Supports all modern features of C++11, C++14, C++17
- Backed up by great package extensions: RcppArmadillo, RcppParallel, RcppMLPACK, ...
- Great documentation, gallery, supported by RStudio
- Rcpp ($\rightarrow$R) is accompanied by RInside (R$\rightarrow$)

Java→R

- Interface to Java, similar to the `.C`/`.Call` C interface

Java→R

- Interface to Java, similar to the `.C`/`.Call` C interface
- R→Java also exists and is called JRI

Java→R

- Interface to Java, similar to the `.C`/`.Call` C interface
- R→Java also exists and is called JRI
- Requires both JRE and JDK to be installed

Java→R

- Interface to Java, similar to the `.C`/`.Call` C interface
- R→Java also exists and is called JRI
- Requires both JRE and JDK to be installed
- Installation is a bit non-trivial (paths, flags, env. vars)

Example: dxFeed API

```
library(rJava)
rjava_obj <- R6::R6Class('rJavaObject',
    public = list(jobject = NULL,
        get_orders = function() {
            .jcall(self$jobject, '[[D',
                   method = 'getOrders', simplify = TRUE)
        },
        initialize = function(symbol) {
            .jinit(classpath = 'path/to/dxfeed-samples
               -3.254.jar')
            self$jobject <- .jnew('com.dxfeed.sample._
               simple_/OrderBookR', symbol)
        }
    )
)
tsla <- rjava_obj$new("TSLA"); Sys.sleep(2)
tsla$get_orders()
```

Example: dxFeed API

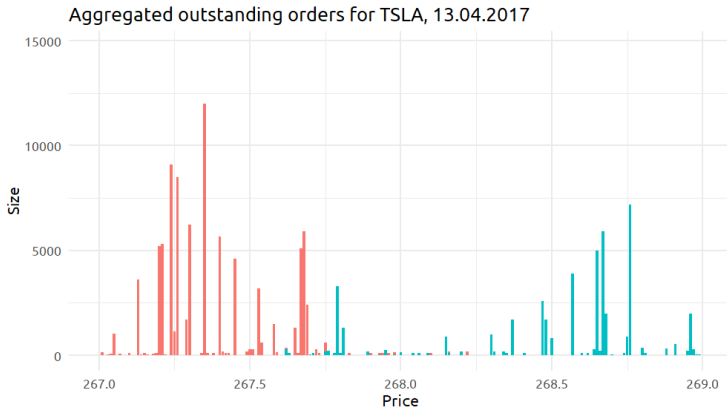

Figure 1: Markets are closed!

Overview

Urbanek [2003]

Rserve is a TCP/IP server which allows other programs to use facilities of R [...] without the need to initialize R or link to the R library.

Overview

Urbanek [2003]

Rserve is a TCP/IP server which allows other programs to use facilities of R [...] without the need to initialize R or link to the R library.

Features:

- client/server, multiple connections, authentication

Overview

Urbanek [2003]

Rserve is a TCP/IP server which allows other programs to use facilities of R [...] without the need to initialize R or link to the R library.

Features:

- client/server, multiple connections, authentication
- native data type conversion

Overview

Urbanek [2003]

Rserve is a TCP/IP server which allows other programs to use facilities of R [...] without the need to initialize R or link to the R library.

Features:

- client/server, multiple connections, authentication
- native data type conversion
- clients for Java, C++, C#, Python, R, Haskell, JavaScript, Ruby, ...

Overview

Urbanek [2003]

Rserve is a TCP/IP server which allows other programs to use facilities of R [...] without the need to initialize R or link to the R library.

Features:

- client/server, multiple connections, authentication
- native data type conversion
- clients for Java, C++, C#, Python, R, Haskell, JavaScript, Ruby, ...
- binary transport, file transfer

Overview

Urbanek [2003]

Rserve is a TCP/IP server which allows other programs to use facilities of R [...] without the need to initialize R or link to the R library.

Features:

- client/server, multiple connections, authentication
- native data type conversion
- clients for Java, C++, C#, Python, R, Haskell, JavaScript, Ruby, ...
- binary transport, file transfer
- Rserve runs as a (remote) separate process!

As a process

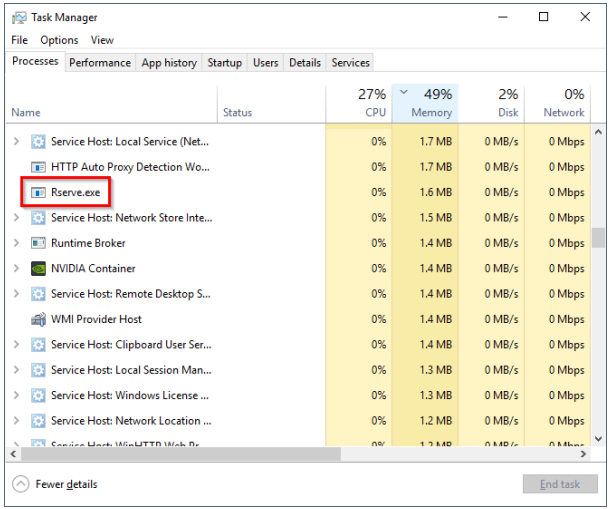

Figure 2: Rserve, ready to serve

Rserve example (Java code)

```
import org.rosuda.REngine.*;
import org.rosuda.REngine.Rserve.*;

public class Main {
    public static void main(String[] args)
            throws RserveException, REXPMismatchException {
        RConnection c = new RConnection();
        REXP x = c.eval("R.version.string");
        System.out.println(x.asString());
        double d[] = c.eval("rnorm(10)").asDoubles();
        for (int i = 0; i < d.length; i++) {
            System.out.println(d[i]);
        }
    }
}
```

Rserve example (output)

```
"C:\Program Files\Java\jdk1.8.0_181\bin\java.exe" (...)
R version 3.5.1 (2018-07-02)
1.1970643462988602
-0.5769699503103762
-0.3684006009955729
0.06546082679467544
1.2032233686962528
-0.04087141177944551
1.7461656480450007
1.8672363049908611
-1.1040409417073296
-0.6567773875872637

Process finished with exit code 0
```

Rserve example (plotting)

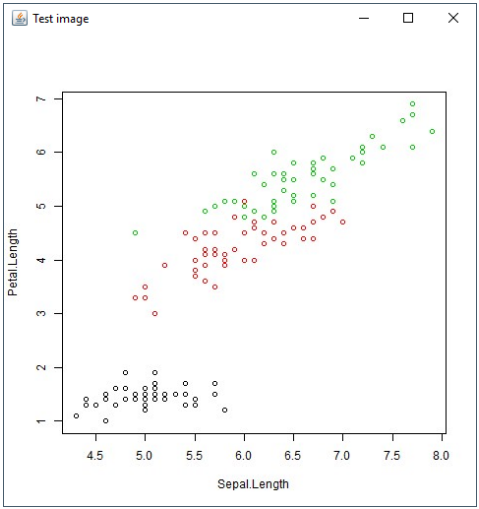

Figure 3: Beloved flowers, now in Java

R and Python

- Very popular interop (due to similarity)

R and Python

- Very popular interop (due to similarity)
- R→: `pyRserve`, `rpy2`, ...

R and Python

- Very popular interop (due to similarity)
- R→: `pyRserve`, `rpy2`, ...
- →R: `reticulate`, ...

pyRserve

```
>>> import pyRserve
>>> conn = pyRserve.connect()
>>> conn.voidEval('doubleit <- function(x) { x*2 }')
>>> conn.eval('doubleit(2)')
4.0
>>> conn.voidEval('tripleit <- function(y) { y*3 }')
>>> conn.r.tripleit(5)
15.0
```

pyRserve

```
R> t.test(c(1,2,3,1), c(1,6,7,8))

      Welch Two Sample t-test

data:  c(1, 2, 3, 1) and c(1, 6, 7, 8)
t = -2.3054, df = 3.564, p-value = 0.09053
alternative hypothesis: true difference in means is not
   equal to 0
95 percent confidence interval:
 -8.4926941  0.9926941
sample estimates:
mean of x mean of y
     1.75      5.50
```

pyRserve

```
>>> conn.r.t.test(numpy.array([1,2,3,1]), numpy.array
    ([1,6,7,8]))

<TaggedList
(statistic=TaggedArray([-2.30541984], key=['t']),
parameter=TaggedArray([3.56389482], key=['df']),
p.value=0.09053264073333127,
conf.int=AttrArray([-8.49269413, 0.99269413],
attr={'conf.level': array([0.95])}),
estimate=TaggedArray([1.75, 5.5 ],
key=['mean of x', 'mean of y']),
null.value=TaggedArray([0.],
key=['difference in means']),
alternative='two.sided',
method='Welch Two Sample t-test',
data.name='arg_0_ and arg_1_')>
```

rpy2

- The usage is very similar to `pyRserve` via `rpy2.interface`

rpy2

- The usage is very similar to `pyRserve` via `rpy2.interface`
- `rpy2.robjects` provides more high-level facilities:
  - Bindings for functions, environments, arbitrary R objects

rpy2

- The usage is very similar to `pyRserve` via `rpy2.interface`
- `rpy2.robjects` provides more high-level facilities:
  - Bindings for functions, environments, arbitrary R objects
  - Extended bindings for vectors, matrices, data frames (incl. R-style subsetting and assigning)

rpy2

- The usage is very similar to `pyRserve` via `rpy2.interface`
- `rpy2.robjects` provides more high-level facilities:
  - Bindings for functions, environments, arbitrary R objects
  - Extended bindings for vectors, matrices, data frames (incl. R-style subsetting and assigning)
  - Bindings between `rpy2` and `numpy`, `pandas`

rpy2

```
from rpy2.robjects.packages import importr, data
datasets = importr('datasets')
mtcars_env = data(datasets).fetch('mtcars')
mtcars = mtcars_env['mtcars']
from rpy2.robjects.lib.dplyr import (DataFrame, filter,
    mutate, group_by, summarize)

dataf = (DataFrame(mtcars) >>
         filter('gear>3') >>
         mutate(powertoweight='hp*36/wt') >>
         group_by('gear') >>
         summarize(mean_ptw='mean(powertoweight)'))
```

reticulate

- Object/type conversions, incl. lists, data frames, functions

reticulate

- Object/type conversions, incl. lists, data frames, functions
- Import modules and refer to them from R

reticulate

- Object/type conversions, incl. lists, data frames, functions
- Import modules and refer to them from R
- RStudio allows mixing Python and R chunks in R Markdown, supports `matplotlib` output

reticulate

- Object/type conversions, incl. lists, data frames, functions
- Import modules and refer to them from R
- RStudio allows mixing Python and R chunks in R Markdown, supports `matplotlib` output

```
R> library(reticulate)
R> np <- import("numpy")
R> a <- np$array(c(1:4))
R> cumsum(a)
[1]  1  3  6 10
```

Holy ... Graal?

- GraalVM: extension of the JVM, supporting polyglot programming

Holy ... Graal?

- GraalVM: extension of the JVM, supporting polyglot programming
- Catch: it uses a different R implementation, FastR

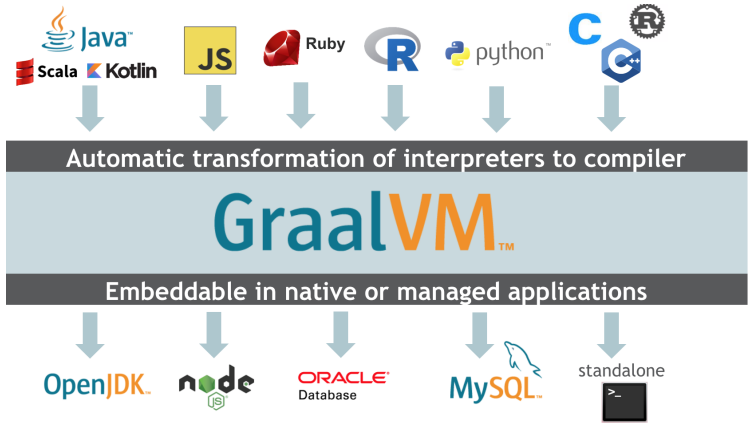

Figure 4: Pure magic in one image

Java/JavaScript/R example

```
# Run container
docker run -p 3000:3000 -it -v C:/dxfeed/graalvm-demos:/
    home oracle/graalvm-ce:1.0.0-rc15 bash

# Execute in the container
gu install r
cd /home/polyglot-javascript-java-r/
node --jvm --polyglot server.js
```

Java/JavaScript/R example

```
# In server.js
const express = require('express')
const app = express()
const BigInteger = Java.type('java.math.BigInteger')
app.get('/', function (req, res) {
  // Using Java standard library classes
  text += BigInteger.valueOf(10).pow(100)
          .add(BigInteger.valueOf(43)).toString() + '<br>'
  // Using R interoperability to create graphs
  text += Polyglot.eval('R',
    `svg(); require(lattice)
     x <- 5 * 1:1000; y <- sin(x); z <- cos(x)
     print(cloud(x~y*z, main="cloud plot"))
     grDevices:::svg.off()`);
  res.send(text)
})
app.listen(3000)
```

Java/JavaScript/R example

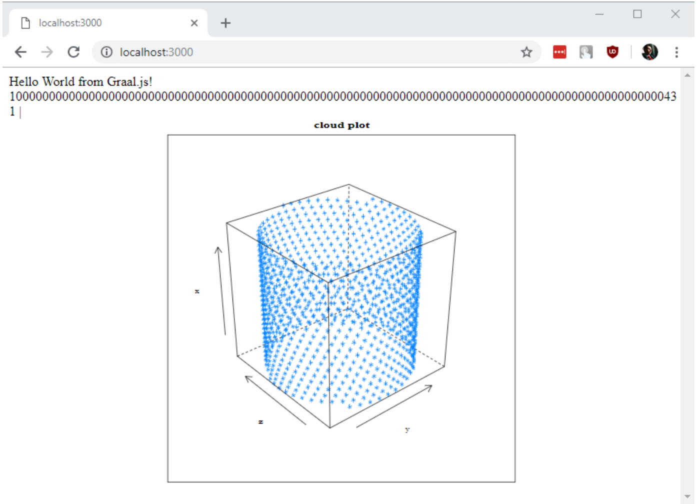

Figure 5: Drawing with R in your browser

Takeaway

Interop is hard, but we're standing on the shoulders of giants.

Takeaway

Interop is hard, but we're standing on the shoulders of giants.

Thanks!
https://github.com/tonytonov/talks

References

Malone, Todd (2014). *Interoperability in Programming Languages.*

Matloff, Norman (2011). *The Art of R Programming: A Tour of Statistical Software Design.* No Starch Press, San Francisco, CA, USA.

Urbanek, Simon (2003). *Rserve A fast way to provide R functionality to applications.*

Eddelbuettel, Dirk and Francois, Romain (2011). *Rcpp: Seamless R and C++ Integration.* Journal of Statistical Software, 40(8), 1-18.

R Development Core Team (2011). *Writing R Extensions.* R Foundation for Statistical Computing, Vienna, Austria.