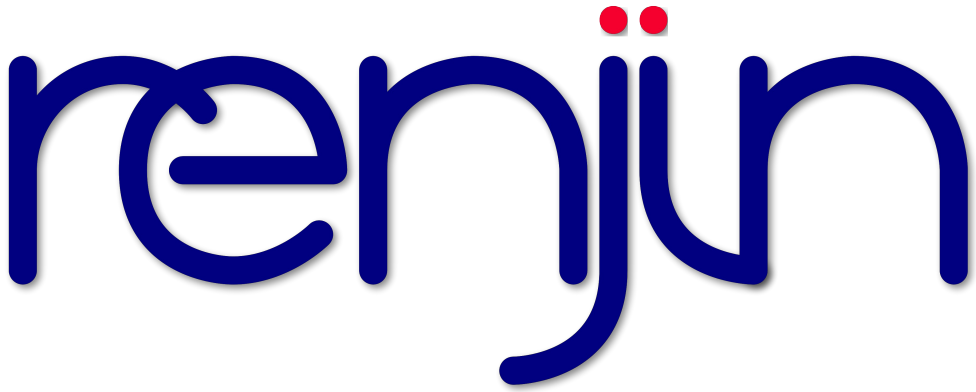
Introduction to Renjin

The R interpreter in the JVM



Maarten-Jan Kallen
mj@bedatadriven.com

Presentation outline

- What is Renjin?
- Renjin's design goals
- Other R ‘engines’
- Example Java application
- Example Google Appengine deployment
- Where to find information and help

To understand what Renjin is,
you should really understand what (GNU) R is.

What is R?

- A dynamically interpreted, functional programming language specifically designed with statistical applications in mind
- “R implements the grammar of the S language, with a few extensions” (Chambers, 2008)
- There is no language reference, only the reference implementation referred to as GNU R
- The core of GNU R (i.e. the set of primitive functions) is written in C

How does R work?

- You type an expression,
- the language interpreter (or 'evaluator') parses and evaluates that expression,
- the interpreter prints out the result of the expression.

  = Read-Eval-Print-Loop ('REPL')

  Obviously, the 'evaluator' is the core element!

What makes up R?

- In his book *Software for data analysis* (2008) John Chambers states three concepts that are important to R:
  - the language,
  - the evaluator, and
  - the management of memory for objects.
- For GNU R as a whole, I would add library management to this list

What is Renjin?

- Interpreter for the R language written in Java:
  - primitive functions are written in Java
  - JVM takes care of memory management (such as garbage collection)
  - third-party Java libraries for things such as LAPACK/BLASS and file system access
- Like GNU R, it's open source

Overview of other R ‘engines’

- GNU R = standard interpreter written in C
- Pretty quick R (pqR) = fork of GNU R
- TIBCO Enterprise Runtime for R (TERR) = new (closed-source) interpreter written in C++
- FastR = new (open-source) interpreter written in Java at Purdue University
- Few others which are mostly academic projects
- Revolution R Enterprise and Oracle R Distribution are not much more than repackaged versions of GNU R

Renjin's design goals

- Run R scripts and programs in the JVM: enable cloud computing to use scalability and reliability of Platform-as-a-Service providers like Google Appengine
- Improve performance and handle large(r) data sets
- Greater flexibility through abstraction of things such as data types and file system access
- High level of compatibility with GNU R 2.14.2 and CRAN packages

We keep the best of R, namely the language itself, and improve the plumbing...

...which is pretty much everything else.

Free benefits

- Seamless integration with the rest of the enterprise: use tools in the Java ecosystem for profiling/debugging, project/dependency management, release/repository management, continuous integration, component lifecycle management, etc.
- More productive development cycle: no need to port a prototype written in R to Java or C++ for production use

Abstraction: GNU R doesn't do it

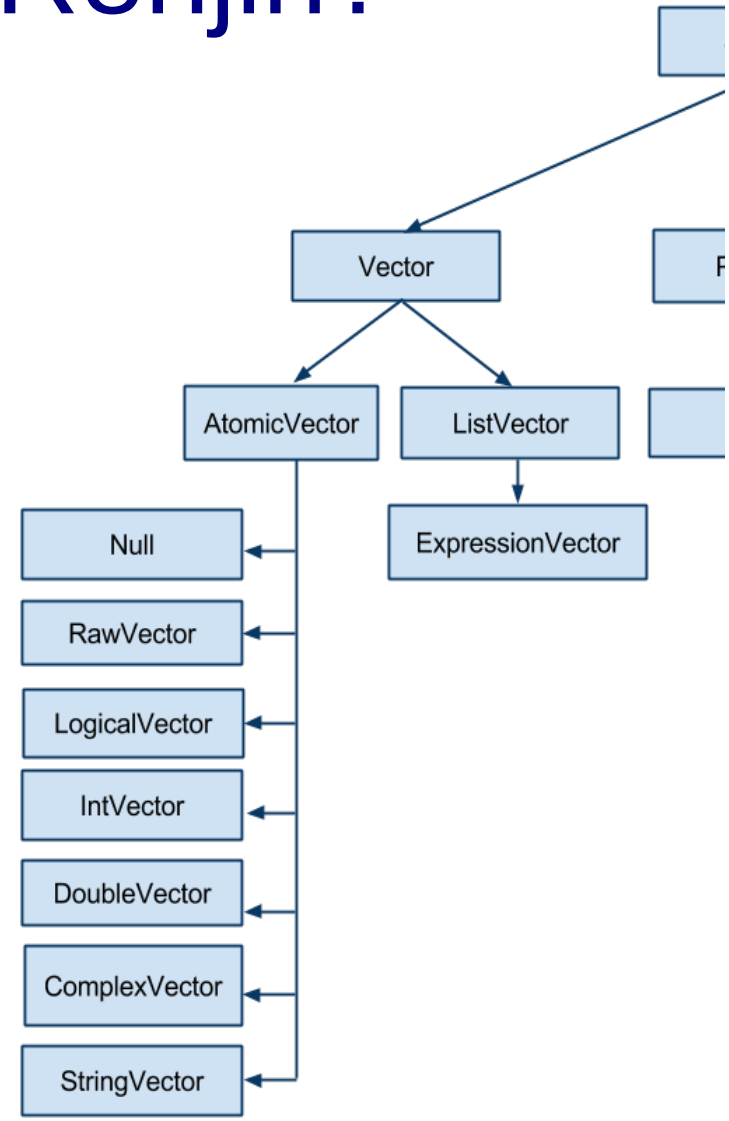
- GNU R version 3.1 introduces the `anyNA(x)` function that does `any(is.na(X))`
- `anyNA(x)` is faster and uses less memory
- but, like `paste0()`, it introduces yet another function
- a discussion about the name of the function happened **after** it was implemented: http://bit.ly/anyNA (GNU R bug 15239)

Abstraction: Renjin does it

- If you call `is.na(x)` on an atomic vector `x` of length > 100, it will return a view and then `any()` will operate on that view
- No copy of `x` is made
- No extra function is required
- All of the following statements also benefit from this abstraction:
  - `any(is.na(x) | is.na(y))`
  - `all(!is.na(x))`
  - `x<-is.na(x);any(x)`

How does it work in Renjin?

- *Vector* and *AtomicVector* are Java interfaces
- All vector types have abstract classes
- the view returned by `is.na(x)` is just a special implementation of the *LogicalVector* class called *IsNaVector*
- *IsNaVector* implements the *Vector* interface with an `isElementNA()` method that can be used in a loop over the vector



Using abstraction, the performance issue of `any(is.na(x))` is solved in the interpreter itself.

Example 1: Renjin in a Java application

- Step 1: open up your favorite IDE (e.g. Eclipse or IntelliJ IDEA)
- Step 2: drop the Renjin Script Engine with dependencies JAR into your project library (download from http://bit.ly/1qz2Vz2)
- Step 3: use the Java scripting API (i.e. `import javax.script.*;`) to evaluate R code within your Java program

Example 2: Renjin in Google Appengine

- git clone https://github.com/bedatadriven/renjin-examples.git
- cd appengine-simple
- mvn appengine:devserver (this will download all necessary dependencies: *mvn* requires Maven 3.0+ to be installed)
- point your browser at http://localhost:8888/ and rejoice!

Will my R code work in Renjin?

Probably not.

Current compatibility

- No graphics (not planned either)
- No S4
- No MASS package
- No Rcpp (not planned either)
- About 73% of all primitive functions implemented
- Users cannot install GNU R packages, but we provide a repository of CRAN packages built for Renjin at http://packages.renjin.org

Using CRAN packages

Just try `library("pkgname")`, no installation required, just an internet connection.

Conclusion

- Renjin is not yet for everyone, nor for exploratory analysis
- Best option for tight integration of R programs into a Java application
- No-brainer for deployment onto Google Appengine, Microsoft Azure, Red Hat OpenShift, etc.
- When we reach ~100% compatibility with GNU R, better performance will make Renjin a serious alternative
- We are looking for contributors and users (i.e. testers)

Where to find information

- Main website at http://www.renjin.org for downloads, documentation and blog
- GitHub at https://github.com/bedatadriven/renjin for the source code and the issue tracker
- Documentation for Java developers at http://docs.renjin.org

Where to get help

- Google group (i.e. public mailing list) for developers (and users) at http://groups.google.com/group/renjin-dev
- StackOverflow now has the [renjin] tag: http://stackoverflow.com/questions/tagged/renjin
- Commercial support by BeDataDriven ☺

How to stay up to date

- Follow @mj_kallen or @bedatadriven on Twitter
- Sign up for the newsletter or subscribe to the blog's RSS feed at http://www.renjin.org