**Announcement**

SciCom version 0.2.2 has been released. SciCom (Scientific Computing) for Ruby brings the power of R to the Ruby community. SciCom is based on Renjin, a JVM-based interpreter for the R language for statistical computing.

**R on the JVM**

Over the past two decades, the R language for statistical computing has emerged as the de facto standard for analysts, statisticians, and scientists. Today, a wide range of enterprises – from pharmaceuticals to insurance – depend on R for key business uses. Renjin is a new implementation of the R language and environment for the Java Virtual Machine (JVM), whose goal is to enable transparent analysis of big data sets and seamless integration with other enterprise systems such as databases and application servers.

Renjin is still under development, but it is already being used in production for a number of client projects, and supports most CRAN packages, including some with C/Fortran dependencies.

**SciCom and Renjin**

SciCom integrates with Renjin and allows the use of R inside a Ruby script. In a sense, SciCom is similar to other solutions such as RinRuby, Rpy2, PipeR, etc. However, since SciCom and Renjin both target the JVM there is no need to integrate both solutions and there is no need to send data between Ruby and R, as it all resides in the same JVM. Further, installation of SciCom does not require the installation of GNU R; Renjin is the interpreter and comes with SciCom. Finally, although SciCom provides a basic interface to Renjin similar to RinRuby, a much tighter integration is also possible (see examples below).

**SciCom with Standard R Interface**

SciCom allows R programmers to use R commands inside a Ruby script in a way similar to RinRuby by calling method eval and passing to it an R script:



Programmers can also use here docs to integrate an R script inside a Ruby script. The next example show a model for predicting baseball wins based on runs allowed and runs scored. The data comes from Baseball-Reference.com.



The output of the program above is:



**The SciCom “language”**

SciCom also allows for implementing R scripts in a “language” that is just like Ruby, so that the developer does not need to know that she is actually writing an R script. All R methods are accessible through an R namespace.

The next script is the same baseball model done in R above using SciCom ‘language’:



We show bellow an example of calculating the correlation matrix without using the build-in functions. First this is done in an R script and then using SciCom:

Now the same code using SciCom



As another example, here is a SciCom script to print the number of days for every month is 2005:



As can be seen from these examples, R methods can be accessed through the R namespace in SciCom, so, R method ‘seq’ is called in SciCom as ‘R.seq’. R methods that are applied on objects can be called in two ways, either using the R namespace as in ‘R.factor’ or directly on the object, as in this case we did ‘cmonth.factor’.

This last example shows how SciCom allows method chaining, which is not possible in an R script.

**SciCom main properties are:**

* Allows access to R scripts from inside Ruby scripts;
* Allows for R scripts written in R by accessing method ‘R.eval’;
* Allows R scripts to be embedded inside here docs in Ruby;
* Creates a new ‘language’ that allows regular Ruby scripts to call R methods in such a way that programmers can be unaware of the fact that they are using R (although, of course, knowing R is of great benefit).

**SciCom installation and download:**

* Install Jruby
* jruby –S gem install scicom

**SciCom Homepages:**

* <http://rubygems.org/gems/scicom>
* <https://github.com/rbotafogo/scicom/wiki>

**Contributors:**

Contributors are welcome.

**SciCom History:**

* 19/Nov/2014: Another small bug fix
* 16/Nov/2014: Version 0.2.1 – Small bug fix
* 16/Nov/2014: Version 0.2.0 – Initial release