Cheminformatics for Ruby - Getting Started with Rubidium

Richard Apodaca

Blog post published November 6, 2007 in *Depth-First*
https://doi.org/10.59350/f4hnf-34f61

Abstract

Cheminformatics has seen the introduction of a diverse array of new open source software over the last few years. Using it all to its fullest potential is not always easy; differing languages, dependencies, interfaces, and varying levels of documentation make the job especially difficult. Rubidium is a new open source project aimed at changing that. Rubidium is a full-featured cheminformatics scripting environment for Ruby.

Keywords

Physical and Theoretical Chemistry (Subfield), Mineralogy and Gemology Studies (Topic), Geochemistry and Petrology (Subfield)

Recommended citation

Apodaca, R. (2007, November 6). Cheminformatics for Ruby - Getting Started with Rubidium. *Depth-First*. https://doi.org/10.59350/f4hnf-34f61

Copyright

Copyright © Richard Apodaca 2007. Distributed under the terms of the Creative Commons Attribution 4.0 International License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

Cheminformatics has seen the introduction of a diverse array of new open source software over the last few years. Using it all to its fullest potential is not always easy; differing languages, dependencies, interfaces, and varying levels of documentation make the job especially difficult. Rubidium is a new open source project aimed at changing that.

Rubidium is a full-featured cheminformatics scripting environment for Ruby. When complete, Rubidium will offer a single well-tested and well-documented Ruby interface to the best open source cheminformatics software. Rubidium-0.1.0 is now available for download.

https://doi.org/10.59350/f4hnf-34f61

Downloading and Installing Rubidium

Rubidium runs on JRuby, a pure Java implementation of the Ruby language. After installing JRuby on your system, you should be ready to install Rubidium.

Installation is most conveniently done with the Ruby package manager RubyGems.

The Rubidium RubyGem can be downloaded from RubyForge (large file). The `gem` command is all we need:

```
ll rbtk-0.1.0-jruby.gem
-rw-r--r-- 1 rich rich 12955136 Nov  6 07:56 rbtk-0.1.0-jruby.gem
$ jruby -S gem install rbtk-0.1.0.gem
```

Note: at the time of this writing, my installation of JRuby 1.0.1 was reporting an out of memory error when attempting to use the RubyForge RubyGems repository directly. Downloading Gems separately and then installing the local copy is a workaround.

Testing the Installation

Rubidium can be tested with the following code run in interactive JRuby (jirb):

```
jirb
irb(main):001:0> require 'rubygems'
=> true
irb(main):002:0> gem 'rbtk'
=> true
irb(main):003:0> require 'rubidium/lang'
=> true
irb(main):004:0> c=Rubidium::Converter.new
=> #&lt;Rubidium::Converter:0xbd4e3c ... &gt;
irb(main):005:0> c.set_formats 'smi', 'mol'
=> true
irb(main):006:0> c.convert 'c1ccccc1'
=> "\n  CDK    11/6/07,8:41\n\n  6  6  0  0  0  0  0  0  0  0999 V2000\n    0.0000    0.00
```

Low-Level Interface

There's not much yet to Rubidium itself beyond molecular language interconversions offered by the Chemistry Development Kit (CDK). But the CDK offers a wide range of cheminformatics functionality that is immediately accessible in raw form via JRuby itself. For example, we can calculate the TPSA of oxazepam:

```
jirb
irb(main):001:0> require 'rubygems'
=> true
irb(main):002:0> gem 'rbtk'
=> true
irb(main):003:0> require 'cdk/lang'
=> true
irb(main):004:0> import 'org.openscience.cdk.qsar.descriptors.molecular.TPSADescriptor'
=> ["org.openscience.cdk.qsar.descriptors.molecular.TPSADescriptor"]
irb(main):005:0> reader=CDK::SmilesReader.new
=> #&lt;CDK::SmilesReader:0x1088a1b ... &gt;
irb(main):006:0> mol=reader.read 'O=C3Nc1ccc(Cl)cc1C(c2ccccc2)=NC3O'
=> #&lt;Java::OrgOpenscienceCdk::Molecule:0x174f02c ... &gt;
irb(main):007:0> tpsa = TPSADescriptor.new
=> #&lt;Java::OrgOpenscienceCdkQsarDescriptorsMolecular::TPSADescriptor:0x14596d5 ...&gt;
irb(main):008:0> result = tpsa.calculate mol
=> #&lt;Java::OrgOpenscienceCdkQsar::DescriptorValue:0x171120a ..&gt;
irb(main):009:0> result.value.double_value
=> 61.69
```

Conclusions

There's much more to be done with Rubidium. As more software packages and their Ruby interfaces are added, a major challenge will be to maintain a simple yet powerful interface to the underlying capabilities.