

One further potential improvement that we have investigated but so far have been less successful with, is the use of stick patterns, rather than semi-continuous patterns consisting of equidistant points. Not only will this use less memory (on average 40 – 100 peaks are present in one pattern), but the calculation of the similarity function is faster, too. Moreover, it allows for easy selection of the most prominent features only, again decreasing computational requirements. However, the crucial updating step during training so far has proved difficult to define.

Applications

The trained map may be used to visualize the contents of the database in a variety of ways. For individual compounds, one can show areas that contain similar crystal packings. Pair-wise comparisons need not to be performed for the whole database, which is computationally very expensive: one can concentrate on the codebook vectors of the map. All compounds that map to similar units are candidates for further investigation.

A particular advantage of using semi-continuous powder patterns is that experimental patterns can directly be mapped. No peak picking is required. One can even assign chemical meaning to some characteristics of the unit vectors. If many peaks are present in the low 2θ ranges, it means that the cell volume is quite large. For other forms of spectroscopy, some peaks may even be interpreted directly. All this may aid in the elucidation of structure parameters of unknown compounds.

One can also use the map as a means of stratified sampling: a small set of compounds (e.g. one for each unit) can be selected that covers the complete chemical space. In polymorph prediction, this feature can be used to reduce the number of structures before (expensive) energy minimisation.

As a final example, trained maps can be used for database quality control. Although many steps in the sequence from measuring data to storing the results in a database can be automated and in fact are, there is still more than enough opportunity for error. Compounds with either very unrealistic powder patterns

or properties very dissimilar to compounds mapped to the same units are candidates for further investigation.

In conclusion, the examples in this paper show how the versatility of the R environment can contribute to a better understanding of the connections in large databases of molecules. The necessary speed is obtained by using compiled code; the trained maps are stored as R objects which can easily be interrogated, even by inexperienced R users, and on which new objects can easily be mapped. One could even envisage a web-based application similar to examples mentioned in (Kohonen, 2001).

Acknowledgements

René de Gelder is acknowledged for his crystallographic input; Willem Melssen for providing expertise on self-organising maps.

Bibliography

- F. H. Allen. The Cambridge Structural Database: a quarter of a million crystal structures and rising. *Acta Crystallogr.*, B58:380–388, 2002. [24](#)
- R. de Gelder, R. Wehrens, and J. A. Hageman. A generalized expression for the similarity spectra: application to powder diffraction pattern classification. *J. Comput. Chem.*, 22(3):273–289, 2001. [24](#)
- T. Kohonen. *Self-Organizing Maps*. Number 30 in Springer Series in Information Sciences. Springer, Berlin, 3 edition, 2001. [24](#), [28](#)
- R. Wehrens, W.J. Melssen, L. Buydens, and R. de Gelder. Representing structural databases in a self-organizing map. *Acta Cryst.*, B61:548–557, 2005. [24](#)

*Institute for Molecules and Materials
Analytical Chemistry
The Netherlands*

R.Wehrens,e.willighagen@science.ru.nl

Generating, Using and Visualizing Molecular Information in R

by Rajarshi Guha

Introduction

R, as a statistical computing environment, has a wide variety of functionality that makes it suitable for

modeling purposes in a number of fields. In the case of cheminformatics we are often presented with a large amount of information, from which chemical meaning and insight must be extracted, using computational tools. In many cases, problems in chem-

informatics are essentially data mining problems in which the data is chemical information. This information can be experimentally (such as assay data) or computationally generated. In the latter case, one requires access to a toolkit or library that is able to generate chemical information such as fingerprints, similarity values or molecular descriptors. A number of such toolkits are available such as JOELib, OEChem and the CDK (Steinbeck et al., 2006, 2003). In most situations one must write a program using the toolkit to generate chemical information which can then be imported into an R session and subsequently analyzed.

In terms of a consistent workflow, it would be useful to be able to access cheminformatics functionality from within the R environment itself. One example of such an application would be the development of a virtual screening pipeline which involves accessing and manipulating structures, evaluation of molecule descriptors and fingerprints and then the development of predictive models. In the interests of code reuse, we would prefer not to implement cheminformatics functionality within an R package but instead reuse established toolkits. The rest of this article will describe how the CDK project can be integrated into the R environment to provide cheminformatics functionality resulting in a seamless workflow for chemical data mining.

It should be noted that some commercial products do use R as a backend. However the combination of R and the CDK is attractive since both are Open-Source products and given that one is a full fledged programming language and the other is an extensive programming library, their combination provides the user with a large amount of flexibility in terms of data access, generation and modeling.

Requirements

As noted, we focus on the integration of the CDK project with R. The CDK project is a Java framework for cheminformatics development. Consequently, to access the classes and methods of the CDK libraries from R, we require an R-Java bridge. This requirement is satisfied by using the SJava package (Temple-Lang, 2005). The examples in this article have been tested with SJava 0.68 and R 2.1.0 running on Fedora Core 3 and 4. In addition, a recent build of the CDK project is required and either all the individual jar files or else the single comprehensive jar file must be placed in the user's CLASSPATH variable. Finally in order to visualize molecular structures a recent build of the Jmol (Howard, 2005) jar file as well as a copy of the JChemPaint (Krause et al., 2000) jar file should also be placed in the users CLASSPATH.

If one looks at the documentation for the CDK project it is clear that it provides a very wide variety of cheminformatics functionality. Performing

certain tasks using the CDK can be quite involved (such as loading arbitrary file formats) whereas other tasks can consist of a single call to a static method (such as getting a fingerprint). Though SJava provides the means to access all the functionality of the CDK, many tasks can become tedious to perform in the R environment. Furthermore, the CDK methods will generally return non-primitive Java objects which do not have a corresponding R type. Thus the user must keep track of R and Java objects. To alleviate these problems we have provided a Java library and associated R wrapper functions, known as the `rcdk` library available in the form of an R package from <http://cheminfo.informatics.indiana.edu/~rguha/code/R/index.html#rcdk>. It provides functions to perform a number of common tasks without having to directly access the CDK API via SJava. The aim of the package is to remove some of the tedious conversions between Java types and R types as well as simplify a number of tasks which would otherwise require the user to be more than casually familiar with the CDK API. Table 1 summarizes the R functions currently available in the `rcdk` package.

Generating & using molecular information

As described above, R is well suited to manipulating and analyzing data. The first step, however, is to obtain the data. For cheminformatics problems an important source of data are chemical structures, from which we can generate fingerprints, molecular descriptors and so on. Easy access to the CDK API using the SJava package provides the user with a streamlined workflow, whereby statistical and chemical information can be generated and manipulated within a single environment. In this section we present two examples of how we can access the CDK classes to obtain information regarding chemical structure and then use the data for modeling purposes.

A common task in a cheminformatics setting is clustering. When clustering molecules, we usually consider a variety of structural features of the molecules in question. One approach is to calculate a set of molecular descriptors. Another common approach is to evaluate binary fingerprints. Thus for example, we may have a collection of structure files. To perform a fingerprint based clustering, we would load the files into the R environment and then call the fingerprint method present in the CDK API. The return value of this method is a `java.util.BitSet` object which must be converted to a form usable by R. This is easily performed by parsing the String representation of the `BitSet` object. With the help of the

Function	Description
<code>edit.molecule</code>	Brings up the JChemPaint 2D structure editor and returns the final structure(s) as a CDK Molecule object
<code>get.desc.values</code>	Extracts the numerical values from a CDK DescriptorValue object returned by a descriptors calculate method
<code>get.fingerprint</code>	Returns a list of numeric objects or a single numeric object containing the positions of the bits that are set to 1 for the specified molecules fingerprint. Uses the default settings for the CDK Fingerprinter class
<code>load.molecules</code>	Loads one or more molecular structure files from disk and returns a list of CDK Molecule objects
<code>view.molecule</code>	Brings up a Jmol window displaying the molecule contained in the specified file. A Jmol script string can also be supplied
<code>view.molecule.table</code>	Displays multiple molecular structure files and associated numerical data in tabular format

Table 1: A summary of the functions available in the **rdck** package.

rdck package functions this can be easily achieved in a few lines of code:

```
> fnames <- list.files(pattern='*.sdf')
> molecules <- load.molecules(fnames)
> fprinter <- JNew('Fingerprinter')
> fplist <- lapply(molecules,
+ function(mol,fpr) {
+   .Java(fpr, 'getFingerprint', mol)
+ }, fpr=fprinter)
>
```

To convert the fingerprint returned by the CDK to a form usable by R we can use the following code:

```
> s <- gsub('{ }', '',
+         fplist[[1]]$toString())
> s <- strsplit(s, split=',')[[1]]
> s <- as.numeric(s)
```

The `load.molecule` function encapsulates a number of calls to the CDK API via SJava to load a molecular structure file. Since the CDK API provide a single static method to obtain fingerprints, the call using SJava is simple enough that a wrapper function is not required. The **rdck** package provides a convenience function, `get.fingerprint`, that encapsulates the above call to the CDK library and parsing of the return value to a numeric.

The code snippet above converts the fingerprint to a numeric object and multiple fingerprints can be aggregated into a list object and then manipulated using the binary fingerprint package (Guha, 2005a). This package allows one to obtain a distance matrix for the set of fingerprints using the Tanimoto metric which can then be used as input to the numerous clustering routines available in R. For a more detailed discussion of this application the reader is referred to Guha (2005c).

Another important task in the field of cheminformatics is QSAR modeling. In statistical terms this is simply the development of predictive models using feature vectors obtained from molecular struc-

ture information. Clearly, the R environment is well suited for the development of QSAR models. To build QSAR models one needs a set of feature vectors characterizing various aspects of molecular structure. These features are generally termed molecular descriptors and one can find references to a huge variety of descriptors in the cheminformatics literature (Todeschini and Consonni, 2002) and a number of packages are available to perform descriptor calculation (Jurs et al., 1979; Chemical Computing Group Inc., 2004; Todeschini et al., 2005). The CDK contains a number of classes implementing a variety of descriptor routines, including constitutional (atom and bond counts), topological (Zagreb index, χ indices), geometrical (moments of inertia, gravitational index) and whole molecule (BCUT) descriptors. The design of the CDK descriptor package allows the user to automatically evaluate all the available descriptors, though this is still a work in progress. Let us consider an example in which a specific descriptor is to be evaluated, such as the BCUT (Pearlman and Smith, 1999) descriptor. By default the descriptor routine will return the highest and lowest eigenvalues of the property weighted Burden matrices. Thus the simplest way to use this descriptor is:

```
> desc <- .JavaConstructor('BCUTDescriptor')
> dval <- desc$calculate(molecule)
```

Here molecule is an object of class Molecule that has been previously obtained from a disk file or from the structure editor (see below). The return value of the `calculate()` method is an object of class DescriptorValue which contains both the numerical values of the descriptor as well as extra information regarding implementation and literature references. Though it is simple enough to extract the numerical values from this object, the design of the CDK descriptor package can make such extraction a tedious process in the R environment. The **rdck** package provides a convenience function to extract the descriptor value from the object returned by `calculate()`

as follows:

```
> dnum <- get.desc.values(dval)
```

The return value of this function is a numeric object which can contain one or more elements depending on how many values were calculated by the descriptor routine. A more detailed description of this application, including parameter specification, can be found in [Guha \(2005c\)](#). Once a set of descriptors have been calculated, they can be converted to a `data.frame` and used as input to feature selection and model development functionality provided by R.

Molecular editing

In many cheminformatics problems we start out with a set of molecular structures. In general we use a structure editor to make modifications to the structures. Models based on structural information must now be updated. Since, in general, structure editors are external programs it would be useful to be able to access a structure editor from within the R environment. Another situation is when one would like to predict a property of a new molecule using a QSAR model built in R. In both cases, being able to access a structure editor from within R leads to a more consistent and efficient workflow. This can be achieved by using the JChemPaint ([Krause et al., 2000](#)) module of the CDK project. JChemPaint can be used as a standalone 2D structure diagram editor. However, being part of the CDK project, it can also be embedded in other programs and can be queried to obtain the structures being drawn. These features allow it to be used from within R. The `rcdk` package provides a convenience function to bring up the structure editor and return the structure that was drawn. Its usage is simply

```
> molecule <- edit.molecule()
```

In addition, it is also possible to supply a structure to the editor and modify it. The return value is a Java object of class `Molecule`. This is not meant to be used directly in R but can be passed to other methods in the CDK libraries. It is important to note that the structure returned will only have 2D coordinates. For many cases, in which only connectivity information is required, such as QSAR models built from topological descriptors, this is sufficient. However this approach will not be useful for cases where 3D geometries are required. The CDK has recently been enhanced with the development of a 3D coordinate generation package. We are currently extending the `rcdk` package to make use of this functionality and thus allow the user to generate reasonable 3D structures from within the R environment.

Molecular visualization

The previous section has discussed how one can draw and edit molecular structures using the JChemPaint application from within the R environment. However another important task in a cheminformatics workflow is the visualization of 3D molecular structures. This can be achieved by using Jmol which is an open source program for visualizing a wide variety of molecular structures. This project also utilizes a number of CDK classes. As a result it can read a wide variety of molecular structure file formats as well as handle data structures returned by CDK classes.

As in the case of structure editing, the `rcdk` project provides Java classes that handle the details of instantiating a Jmol window and loading structure files. The associated R source file provides R functions that wrap these Java classes.

Currently the `rcdk` package provides two methods to view 3D structures. The first method takes in a single character variable containing the path to the file to view. It can also optionally take a command string which is passed onto Jmol which evaluates it as a script. Thus, to view a structure one could simply write

```
> view.molecule('data/dan001.xyz')
```

A screenshot of the resultant viewer is shown in [Fig. 7](#).

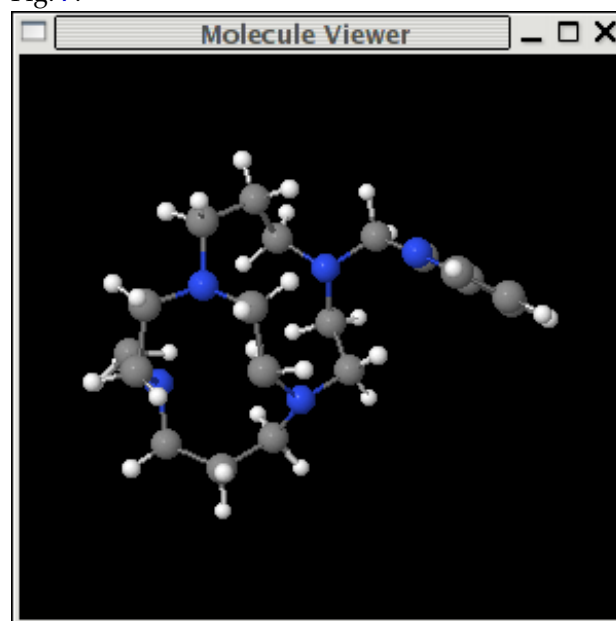


Figure 1: A screenshot of the single molecule viewer.

For tasks such as QSAR modeling one is usually working with a collection of molecules. In this case it would be useful to be able to view molecular structures and associated information (such as molecular descriptor values) in a table. This can be easily achieved by creating a table containing Jmol instances. The `rcdk` package provides a Java class

and associated R function wrapper that performs this task. Thus to view the structures and associated information for a set of molecules one could write:

```
> fnames <- c('data/dan001.xyz',
+ 'data/dan002.xyz', 'data/dan003.xyz',
+ 'data/dan004.xyz')
>
> cnames <- c('Structure', 'Label1',
+ 'Label2', 'Label3', 'Label4')
>
> moldata <- data.frame(matrix(
+ runif(4*4), nrow=4))
>
> view.molecule.table(fnames, cnames, moldata)
```

The resultant table is shown in Fig. 7. However, since each Jmol instance is a full fledged molecular viewer this can be a strain on resources. A future extension to the `rcdk` package will allow the use of the 2D structure diagram generator class present in the CDK, which would result in a much more light weight table.

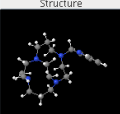
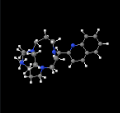
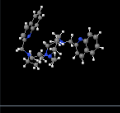
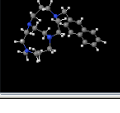
Structure	Label1	Label2	Label3	Label4
	0.657	0.376	0.078	0.693
	0.819	0.477	0.338	0.111
	0.534	0.74	0.726	0.14
	0.892	0.89	0.83	0.941

Figure 2: A screenshot of the table view.

Accessing R from the CDK

This article has focused on accessing the cheminformatics functionality of the CDK from within the R environment. In many cases it is advantageous to be able to access the statistical functionality of R from within a CDK based program. To facilitate this, the CDK contains a set of modeling classes which allow the user to access a number of R routines. The classes are based on SJava and a detailed description of the design of these classes can be found in Guha (2005b). An example of an application using these classes can be found in Guha and Jurs (2005). Currently, the CDK modeling package provides access to linear regression (`lm`), neural networks (`mnet`) and PLS (`p1s.pcr`). Future work on these classes will include other modeling routines (`randomForest`, `lda`,

etc.). A downside of the use of the SJava package is that the modeling functionality is effectively restricted to the Linux platform. Though SJava can be used on Windows, installation can be problematic. In addition, the SJava package is not multi-threaded and as a result all instances of the CDK modeling classes share the same R session. Currently, the design of the classes takes this into account, but this adds an extra layer of complexity. Finally, extending the CDK modeling classes to include other R routines is quite tedious since it essentially involves the design of Java classes which are one-to-one mappings of R objects. One approach that is being considered is the use of the Rserve package (Urbanek, 2005). This package would allow for a much simpler design of the CDK modeling classes and avoids a number of problems associated with the SJava package. The downside is that it requires that the user manually run the Rserve daemon, either remotely or locally. As a result, unless the Rserve daemon is running, the CDK modeling classes will not be of any use.

Conclusions

This article has attempted to highlight the use of the CDK within the R environment for the purposes of cheminformatics tasks. Using the SJava package, the user has access to the wide array of cheminformatics functionality present in the CDK as well as allied projects such as JChemPaint and Jmol. The integration of the CDK project with R results in a very useful workflow for chemical data mining problems. As shown above, one can calculate fingerprints or molecular descriptors and then perform a variety of modeling tasks using the statistical functionality of R. Coupled with Jmol and JChemPaint, one can visualize both statistical results as well as molecular structures. Furthermore, the use of the SJava package allows the CDK project to utilize the statistical functionality of R from within CDK based programs.

Though the user can directly access CDK classes and methods, this can become cumbersome. As a result the `rcdk` package was designed to alleviate the tedium of some common tasks such as loading molecular structure files and visualizing 3D structures. The package consists of a set of Java classes in the form of a jar file and R wrapper functions. Future work involves the addition of a number of helper functions for common cheminformatics tasks as well making the various functions more robust in terms of error handling as well as what objects can be accepted.

Bibliography

Chemical Computing Group Inc. Molecular Operating Environment (MOE 2004.03), 2004. 30

- R. Guha. <http://cheminfo.informatics.indiana.edu/~rguha/code/R/index.html#bfp>, September 2005a. 30
- R. Guha. Using R to Provide Statistical Functionality for QSAR Modeling in CDK. *CDK News*, 2:7–13, 2005b. 32
- R. Guha. Using the CDK as a Backend to R. *CDK News*, 2:2–6, 2005c. 30, 31
- R. Guha and P. Jurs. Integrating R with the CDK for QSAR modeling. In *230th American Chemical Society Meeting & Conference*, Washington D.C., 2005. 32
- M. Howard. <http://www.jmol.org>, September 2005. 29
- P. Jurs, J. Chou, and M. Yuan. *Computer Assisted Drug Design*, chapter Studies of Chemical Structure Biological Activity Relations Using Pattern Recognition. American Chemical Society, Washington D.C., 1979. 30
- S. Krause, E. Willighagen, and C. Steinbeck. JChemPaint - Using the Collaborative Forces of the Internet to Develop a Free Editor for 2D Chemical Structures. *Molecules*, 5:93–98, 2000. 29, 31
- R. Pearlman and K. Smith. Metric Validation and the Receptor-Relevant Subspace Concept. *J. Chem. Inf. Comput. Sci.*, 39:28–35, 1999. 30
- C. Steinbeck, Y. Han, S. Kuhn, O. Horlacher, E. Luttmann, , and E. Willighagen. The Chemistry Development Kit (CDK): An Open-Source Java Library for Chemo- and Bioinformatics. *J. Chem. Inf. Comput. Sci.*, 43:493–500, 2003. 29
- C. Steinbeck, C. Hoppe, S. Kuhn, M. Floris, R. Guha, and E. Willighagen. Recent Developments of the Chemistry Development Kit (CDK) - An Open-Source Java Library for Chemo- and Bioinformatics. *Curr. Pharm. Des.*, in press, 2006. 29
- D. Temple-Lang. <http://www.omegahat.org/RSJava>, September 2005. 29
- R. Todeschini and V. Consonni. *Handbook of Molecular Descriptors*. Wiley-VCH, Berlin, 2002. 30
- R. Todeschini, V. Consonni, and M. Pavan. *Dragon*, 2005. 30
- S. Urbanek. <http://stats.math.uni-augsburg.de/Rserve>, September 2005. 32

Rajarshi Guha
Pennsylvania State University
rxg218@psu.edu