Open Access Publications of Noel O'Boyle

November 2, 2011

Contents

Part I

Cheminformatics toolkits

Chemistry Central Journal

ChemistryCentral

Software

Open Access

Pybel: a Python wrapper for the OpenBabel cheminformatics toolkit

Noel M O'Boyle*[1,2], Chris Morley[3] and Geoffrey R Hutchison[4]

Address: [1]Unilever Centre for Molecular Science Informatics, Department of Chemistry, University of Cambridge, Lensfield Road, Cambridge CB2 1EW, UK, [2]Cambridge Crystallographic Data Centre, 12 Union Road, Cambridge CB2 1EZ, UK, [3]OpenBabel Development Team and [4]Department of Chemistry, University of Pittsburgh, Chevron Science Center, 219 Parkman Avenue, Pittsburgh, PA 15260, USA

Email: Noel M O'Boyle* - baoilleach@gmail.com; Chris Morley - c.morley@gaseq.co.uk; Geoffrey R Hutchison - geoffh@pitt.edu

* Corresponding author

Published: 9 March 2008

Chemistry Central Journal 2008, **2**:5 doi:10.1186/1752-153X-2-5

Received: 23 January 2008
Accepted: 9 March 2008

This article is available from: http://journal.chemistrycentral.com/content/2/1/5

Abstract

Background: Scripting languages such as Python are ideally suited to common programming tasks in cheminformatics such as data analysis and parsing information from files. However, for reasons of efficiency, cheminformatics toolkits such as the OpenBabel toolkit are often implemented in compiled languages such as C++. We describe Pybel, a Python module that provides access to the OpenBabel toolkit.

Results: Pybel wraps the direct toolkit bindings to simplify common tasks such as reading and writing molecular files and calculating fingerprints. Extensive use is made of Python iterators to simplify loops such as that over all the molecules in a file. A Pybel Molecule can be easily interconverted to an OpenBabel OBMol to access those methods or attributes not wrapped by Pybel.

Conclusion: Pybel allows cheminformaticians to rapidly develop Python scripts that manipulate chemical information. It is open source, available cross-platform, and offers the power of the OpenBabel toolkit to Python programmers.

Background

Cheminformaticians often need to write once-off scripts to create extract data from text files, prepare data for analysis or carry out simple statistics. Scripting languages such as Perl, Python and Ruby are ideally suited to these day-to-day tasks [1]. Such languages are, however, an order of magnitude or more slower than compiled languages such as C++. Since cheminformaticians regularly deal with molecular files containing thousands of molecules and many cheminformatics algorithms are computationally expensive, cheminformatics toolkits are typically written in compiled languages for performance.

OpenBabel is a C++ toolkit with extensive capabilities for reading and writing molecular file formats (over 80 are supported) as well as for manipulating molecular data [2]. Many standard chemistry algorithms are included, for example, determination of the smallest set of smallest rings, bond order perception, addition of hydrogens, and assignment of Gasteiger charges. In relation to cheminformatics, OpenBabel supports SMARTS searching [3], molecular fingerprints [4] (both Daylight-type, and structural-key based), and includes group contribution descriptors for LogP [5], polar surface area (PSA) [6] and molar refractivity (MR) [5].

Of the current popular scripting languages, Python [7] is the *de-facto* standard language for scripting in cheminformatics. Several commercial cheminformatics toolkits have interfaces in Python: OpenEye's closed-source successor to OpenBabel, OEChem [8], is a C++ toolkit with interfaces in Python and Java; Rational Discovery's RDKit [9], which is now open source, is a C++ cheminformatics toolkit with a Python interface; the Daylight toolkit [10] from Daylight Chemical Information Systems, written in C, only has Java and C++ wrappers but PyDaylight [11], available separately from Dalke Scientific, provides a Python interface to the toolkit; the Cambios Molecular Toolkit [12] from Cambios Consulting is a commercial C++ toolkit with a Python interface. There are also toolkits entirely implemented in Python: Frowns [13], an open source cheminformatics toolkit by Brian Kelley, and PyBabel [14], an open source toolkit included in the MGLTools package from the Molecular Graphics Labs at the Scripps Research Institute. Note that the latter is not related to the OpenBabel project; rather its name derives from the fact that its aim was to implement in Python some of the functionality of Babel v1.6 [15], a command-line application for converting file formats which is a predecessor of OpenBabel.

Here we describe the implementation and application of Pybel, a Python module that provides access to the OpenBabel C++ library from the Python programming language. Pybel builds on the basic Python bindings to make it easier to carry out frequent tasks in cheminformatics. It also aims to be as 'Pythonic' as possible; that is, to adhere to Python language conventions and idioms, and where possible to make use of Python language features such as iterators. The result is a module that takes advantage of Python's expressive syntax to allow cheminformaticians to carry out tasks such as SMARTS matching, data field manipulation and calculation of molecular fingerprints in just a few lines of code.

Implementation

SWIG bindings

Python bindings to the OpenBabel toolkit were created using SWIG [16]. SWIG (Simplified Wrapper and Interface Generator) is a tool that automates the generation of bindings to libraries written in C or C++. One of the advantages of SWIG compared to other automated wrapping methods such as Boost.Python [17] or SIP [18] is that SWIG also supports the generation of bindings to several other languages. For example, OpenBabel also uses SWIG to generate bindings for Perl, Ruby and Java. An additional advantage is that SWIG will directly parse C or C++ header files while Boost.Python and SIP require each C++ class to be exposed manually. The input to SWIG is an interface file containing a list of OpenBabel header files for which to generate bindings. Using the signatures in the header files, SWIG generates a C file which, when compiled and linked with the Python development libraries and OpenBabel, creates a Python extension module, *openbabel*. This can then be imported into a Python script like any other Python module using the "*import openbabel*" statement.

For a small number of C++ objects and functions, it was necessary to add some convenience functions to facilitate access from Python. Certain types of molecule files have additional data present in addition to the connection table. OpenBabel stores these data in subclasses of OBGenericData such as OBPairData (for the data fields in molecule files such as MOL files and SDF files) and OBUnitCell (for the data fields in CIF files). To access the data it is necessary to 'downcast' an instance of OBGenericData to the specific subclass. For this reason, two convenience functions were added to the interface file, one to cast OBGenericData to OBPairData, and one to cast to OBUnitCell. Another convenience function was added to convert a Python list to a C array of doubles, as this type of input is required for a small number of OpenBabel functions.

Iterators are an important feature of the OpenBabel C++ library. For example, OBAtomAtomIter allows the user to easily iterate over the atoms attached to a particular atom, and OBResidueIter is an iterator over the residues in a molecule. The OpenBabel iterators use the dereference operator to access the data, the increment operator to iterate to the next element, and the boolean operator to test whether any elements remain. Iterators are also a core feature of the Python language. However, the iterators used by OpenBabel are not automatically converted into Python iterators. To deal with this, Python iterator classes that wrap the dereference, increment and boolean operators behind the scenes were added to the SWIG interface file, so that Python statements such as "*for attached_obatom in OBAtomAtomIter(obatom)*" work without problem.

Pybel module

The SWIG bindings provide direct access from Python to the C++ objects and functions in the OpenBabel API (application programming interface). The purpose of the Pybel module is to wrap these bindings to present a more Pythonic interface to OpenBabel (Figure 1). This extra level of abstraction is useful as Python programmers expect Python libraries to behave in certain ways that a C++ library does not. For example, in Python, attributes of an object are often directly accessed whereas in C++ it is typical to call Get/Set functions to access them. A C++ function returning a particular object might require a pointer to an empty object as a parameter, whereas the Python equivalent would not. Even something as simple

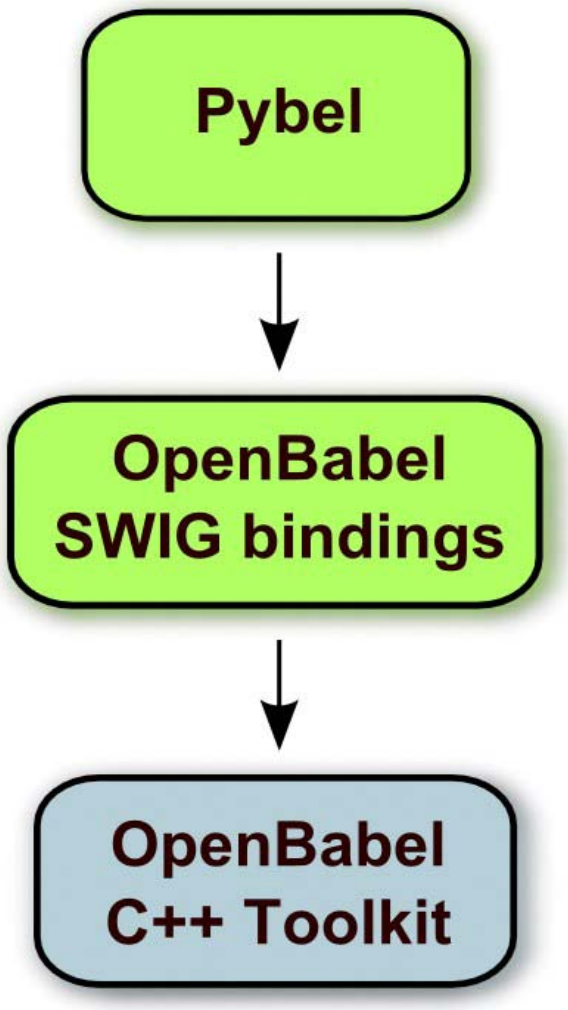

Figure 1
The relationship between Python modules described in the text and the OpenBabel C++ library. Python modules are shown in green; the C++ library is shown in blue.

as differences in the conventions for the case of letters used in variable and method names is a problem, as it makes it more likely for Python programmers to introduce bugs in their code.

One of the key aims of Pybel was to reduce the amount of code necessary to carry out common tasks. This is especially important for a scripting language where programming is often done interactively at a command prompt. In addition, as for any programming language, repeated entry of code for routine and common tasks (so-called 'boilerplate code') is a common cause of errors in code. Reading and writing molecule files is one of the most common tasks for users of OpenBabel but requires several lines of code if using the SWIG bindings. The following code shows how to store each molecule in a multimolecule SDF file in a list called *allmols*:

```
import openbabel

allmols = []

obconversion = openbabel.OBConversion()

obconversion.SetInFormat("sdf")

obmol = openbabel.OBMol()

notatend = obconversion.ReadFile(obmol,
"inputfile.sdf")

while notatend:

   allmols.append(obmol)

   obmol = openbabel.OBMol()

   notatend = obconversion.Read(obmol)
```

To replace this somewhat verbose code, Pybel provides a *readfile* method that takes a file format and filename and returns molecules using the 'yield' keyword. This changes the method into a 'generator', a Python language feature where a method behaves like an iterator. Iterators are a major feature of the Python language which are used for looping over collections of objects. In Pybel, we have used iterators where possible to simplify access to the toolkit. As a result, the equivalent to the preceding code is:

```
import pybel

allmols = [mol for mol in pybel.read
file("sdf", "inputfile.sdf")]
```

The benefits of iterator syntax are clear when dealing with multimolecule files. For single molecule files, however, the user needs to remember to explicitly request the iterator to return the first and only molecule using the *next* method:

```
mol = pybel.readfile("mol", "input
file.mol").next()
```

Pybel provides replacements for two of the main classes in the OpenBabel library, OBMol and OBAtom. The following discussion describes the Pybel Molecule class which wraps an instance of OBMol, but the same design principles apply to the Pybel Atom class. Table 1 summarises the attributes and methods of the Molecule object. By wrapping the base class, Pybel can enhance the Molecule

Table 1: Attributes and methods supported by the Pybel Molecule object

Attribute	Description*
OBMol	The underlying OBMol object
atoms	A list of Pybel Atoms
charge	The total charge (GetTotalCharge)
data	A MoleculeData object for access to data fields
dim	The dimensionality of the coordinates (GetDimension)
energy	The heat of formation (GetEnergy)
exactmass	The mass calculated using isotopic abundance (GetExactMass)
flags	The set of flags used internally by OpenBabel (GetFlags)
formula	The stoichiometric formula (GetFormula)
mod	The number of nested BeginModify() calls (Internal use) (GetMod)
molwt	The standard molar mass (GetMolWt)
spin	The total spin multiplicity (GetTotalSpinMultiplicity)
sssr	The smallest set of smallest rings (GetSSSR)
title	The title of the molecule (often the filename) (GetTitle)
unitcell	Unit cell data (if present)
Method	
write	Write the molecule to a file or return it as a string
calcfp	Return a molecular fingerprint as a Fingerprint object
calcdesc	Return the values of the group contribution descriptors
__iter__	Enable iteration over the Atoms in the Molecule

*Where a Molecule attribute is a direct replacement for a 'Get' method of the underlying OBMol, the name of the method is given in parentheses.

object by providing (1) direct access to attributes rather than through the use of Get methods, (2) additional attributes of the object, and (3) additional methods that act on the object.

(1) As mentioned earlier, it is typical in Python to access attribute values directly rather than using Get/Set methods. With this in mind, the Molecule class adds attributes such as *energy*, *formula* and *molwt* (among others) which give the values returned by calling *GetEnergy()*, *GetFormula()* and *GetMolWt()*, respectively on the underlying OBMol (see Table 1 for the full list).

(2) One of the aims of Pybel is to simplify access to some of the most common attributes. With this in mind, an *atoms* attribute has been added which returns a list of the atoms of the molecule as Pybel Atoms. Access to the data fields associated with a molecule has been simplified by creation of a MoleculeData object which is returned when the *data* attribute of a Molecule is accessed. MoleculeData presents a dictionary interface to the data fields of the molecule. Accessing and updating these field is more convoluted if using the SWIG bindings. Compare the following statements for accessing the "comment" field of the variable *mol*, an OBMol:

```
# Using the SWIG bindings

value = openbabel.toPairData(mol.GetData
["comment"]).GetValue()

# Using Pybel

value = pybel.Molecule(mol).data ["com
ment"]
```

It should be noted that all of these attributes are calculated on-the-fly rather than stored for future access as the underlying OBMol may have been modified.

(3) Four additional methods have been added to the Pybel Molecule (Table 1). The first is a *write* method which writes a representation of the Molecule to a file and takes care of error handling. As with reading molecules from files (see above), this method simplifies the procedure significantly compared to using the SWIG bindings directly. In addition, a *calcfp* method and a *calcdesc* method have been added which calculate a binary fingerprint for the molecule, and some descriptor values, respectively. In the OpenBabel library these are not methods of the OBMol, but rather are loaded as plugins (by OBFingerprint.FindFingerprint and OBDescriptor.FindType, respectively) to which an OBMol is passed as input. The *__iter__* method is a special Python method that enables iteration over an object; in the case of a Molecule, the defined iterator loops over the Atoms of the Molecule. This feature enables constructions such as "*for atom in mol*" where *mol* is a Pybel Molecule.

SMARTS is a query language developed by Daylight Chemical Information Systems for molecular substructure

searching [3]. As implemented in the OpenBabel toolkit, finding matches of a particular substructure in a particular molecule is a four step process that involves creating an instance of OBSmartsPattern, initialising it with a SMARTS pattern, searching for a match, and finally retrieving the result:

```
obsmarts = openbabel.OBSmartsPattern()

obsmarts.Init("[#6] [#6]")

obsmarts.Match(obmol)

results = obsmarts.GetUMapList()
```

Since a SMARTS query can be thought of as a regular expression for molecules, in Pybel we decided to wrap the SMARTS functionality in an analogous way to Python's regular expression module, *re*. With these changes, the same process takes only two steps, an initialisation step and a search step:

```
smarts = pybel.Smarts("[#6] [#6]")

results = smarts.findall(pybelmol)
```

Pybel was not written to replace the SWIG bindings but rather to make it simpler to perform common tasks. As a result, Pybel does not attempt to wrap every single method and class in the OpenBabel library. Because of this, a user may often want to interconvert between an OBMol and a Molecule, or an OBAtom and an Atom. This is quite a straightforward process. A Pybel Molecule can be created by passing an OBMol to the Molecule constructor. In the following example an OBMol is created using the SWIG bindings and then written to a file using Pybel:

```
obmol = openbabel.OBMol()

a = obmol.NewAtom()

a.SetAtomicNum(6)

a.SetVector(0.0, 1.0, 2.0) # Set coordi
nates

b = obmol.NewAtom()

obmol.AddBond(1, 2, 1) # Single bond from
Atom 1 to Atom 2

pybel.Molecule(obmol).write("mol", "out
putfile.mol")
```

The OBMol wrapped by a Pybel Molecule can be accessed through the *OBMol* attribute. This makes it easy to call a method not wrapped by Pybel, such as *OBMol.NumRotors*, which returns the number of rotatable bonds in a molecule:

```
mol = pybel.readfile("mol", "input
file.mol").next()

numrotors = mol.OBMol.NumRotors()
```

Documentation and Testing

To minimise programming errors, programs written dynamically-typed languages such as Python should be tested comprehensively. Pybel has 100% code coverage in terms of unit tests, as measured by Ned Batchelder's coverage.py [19]. It also has several doctests, short snippets of Python code included in documentation strings which serve as both examples of usage and as unit tests.

The Pybel API is fully documented with docstrings. These can be accessed in the usual way with the help() command at the interactive Python prompt after importing Pybel: for example, "*help(pybel.Molecule)*". In addition, the OpenBabel Python web page [20] contains a complete description of how to use the SWIG bindings and the Pybel API. The webpage also contains links to HTML versions of the OpenBabel API documentation and Pybel API documentation. The latter is included in Additional File 1.

Results and Discussion

The principle aim of Pybel is to make it simpler to use the OpenBabel toolkit to carry out common tasks in cheminformatics. These common tasks include reading and writing molecule files, accessing data fields of a molecule, computing and comparing molecular fingerprints and SMARTS matching. Here we present some examples that illustrate how Pybel may be used to carry out common cheminformatics tasks.

Removal of duplicate molecules

When merging different datasets or as a final step in preprocessing, it may be necessary to identify and remove duplicate molecules. In the following example, only the unique molecules in the multimolecule SDF file "inputfile.sdf" will be written to "uniquemols.sdf". Here we will assume that a unique InChI string (IUPAC International Chemical Identifier) indicates a unique molecule. A similar procedure could be performed using the OpenBabel canonical SMILES format, by replacing "inchi" with "can" in the following:

```
import pybel

inchis = []
```

```
output = pybel.Outputfile("sdf", "uniquemols.sdf")

for mol in pybel.readfile("sdf", "input file.sdf"):

   inchi = mol.write("inchi")

   if inchi not in inchis:

      output.write(mol)

      inchis.append(inchi)

output.close()
```

Selection of similar molecules

Another common task in cheminformatics is the selection of a set of molecules of similar structure to a target molecule. Here we will assume that structural similarity is indicated by a Tanimoto coefficient [21] of at least 0.7 with respect to Daylight-type (that is, based on hashed paths through the molecular graph) fingerprints. Note that Pybel redefines the | operator (bitwise OR) for Fingerprint objects as the Tanimoto coefficient:

```
import pybel

targetmol = pybel.readfile("sdf", "target mol.sdf").next()

targetfp = targetmol.calcfp()

output = pybel.Outputfile("sdf", "similar mols.sdf")

for mol in pybel.readfile("sdf", "input file.sdf"):

   fp = mol.calcfp()

   if fp | targetfp >= 0.7:

      output.write(mol)

output.close()
```

Applying a Rule of Fives filter

In an influential paper, Lipinski et al. [22] performed an analysis of drug compounds that reached Phase II clinical trials and found that they tended to occupy a certain range of values for molecular weight, LogP, and number of hydrogen bond donors and acceptors. Based on this, they proposed a rule with four criteria to identify molecules that might have poor absorption or permeation properties. This is the Lipinski Rule of Fives, so-called as the numbers involved are all multiples of five. The following example shows how to filter a database to identify only those molecules that pass all four of the Lipinski criteria. The values of the Lipinski descriptors are also added to the output file as data fields. Note that whereas molecular weight is directly available as an attribute of a Molecule, and LogP is available as one of the three group contribution descriptors calculated by OpenBabel, we need to use SMARTS pattern matching to identify the number of hydrogen bond donors and acceptors. The SMARTS patterns used here correspond to the definitions of hydrogen bond donor and acceptor used by Lipinski:

```
import pybel

HBD = pybel.Smarts("[#7,#8;!H0]")

HBA = pybel.Smarts("[#7,#8]")

def lipinski(mol):

   """Return the values of the Lipinski descriptors."""

   desc = {'molwt': mol.molwt,

      'HBD': len(HBD.findall(mol)),

      'HBA': len(HBA.findall(mol)),

      'LogP': mol.calcdesc(['LogP'])['LogP']}

   return desc

passes_all_rules = lambda desc: (desc['molwt'] <= 500 and

         desc ['HBD'] <= 5 and desc['HBA'] <= 10 and

         desc ['LogP'] <= 5)

if __name__=="__main__":

   output = pybel.Outputfile("sdf", "passLipinski.sdf")

   for mol in pybel.readfile("sdf", "inputfile.sdf"):

      descriptors = lipinski(mol)

      if passes_all_rules(descriptors):
```

```
        mol.data.update(descriptors)

        output.write(mol)

output.close()
```

Future work

The future development of Pybel is closely linked to any changes and improvements to OpenBabel. With each new release of the OpenBabel API, the SWIG bindings will be updated to include any additional functionality. However, additions to the Pybel API will only occur if they simplify access to new features of the OpenBabel toolkit of general use to cheminformaticians. In general, the Pybel API can be considered stable, and an effort will be made to ensure that future changes will be backwards compatible.

Conclusion

Pybel provides a high-level Python interface to the widely-used OpenBabel C++ toolkit. This combination of a high performance cheminformatics toolkit and an expressive scripting language makes it easy for cheminformaticians to rapidly and efficiently write scripts to manipulate molecular data.

Pybel is freely available from the OpenBabel web site[2] both as part of the OpenBabel source distribution and for Windows as an executable installer. Compiled versions are also available as packages in some Linux distributions (openbabel-python in Fedora, for example).

Availability and Requirements

Project name: Pybel

Project home page: http://openbabel.sf.net/wiki/Python

Operating system(s): Platform independent

Programming language: Python

Other requirements: OpenBabel

License: GNU GPL

Any restrictions to use by non-academics: None

Authors' contributions

GRH is the lead developer of OpenBabel and created the SWIG bindings. NMOB developed Pybel, and extended the SWIG interface file. CM compiled the SWIG bindings on Windows and added convenience functions to the OpenBabel API to facilitate access from scripting languages. All authors read and approved the final manuscript.

Additional material

Additional file 1

Pybel API. The HTML documentation of the Pybel API (application programming interface).
Click here for file
[http://www.biomedcentral.com/content/supplementary/1752-153X-2-5-S1.zip]

Acknowledgements

The idea for the Pybel module was inspired by Andrew Dalke's work on PyDaylight [11]. We thank the anonymous reviewers for their helpful comments.

References

1. Ousterhout JK: **Scripting: Higher Level Programming for the 21st Century.** [http://home.pacbell.net/ouster/scripting.html].
2. **OpenBabel v.2.1.1** [http://openbabel.sf.net]
3. **SMARTS – A Language for Describing Molecular Patterns** [http://www.daylight.com/dayhtml/doc/theory/theory.smarts.html]
4. Flower DR: **On the properties of bit string-based measures of chemical similarity.** *J Chem Inf Comput Sci* 1998, **38:**379-386.
5. Wildman SA, Crippen GM: **Prediction of physicochemical parameters by atomic contributions.** *J Chem Inf Comput Sci* 1999, **39:**868-873.
6. Ertl P, Rohde B, Selzer P: **Fast calculation of molecular polar surface area as a sum of fragment-based contributions and its application to the prediction of drug transport properties.** *J Med Chem* 2000, **43:**3714-3717.
7. **Python** [http://www.python.org]
8. **OEChem: OpenEye Scientific Software: Santa Fe, NM.** .
9. **RDKit** [http://www.rdkit.org]
10. **Daylight Toolkit: Daylight Chemical Information Systems, Inc.: Aliso Viejo, CA.** .
11. **PyDaylight: Dalke Scientific Software, LLC: Santa Fe, NM.** .
12. **Cambios Molecular Toolkit: Cambios Computing, LLC: Palo Alto, CA.** .
13. **Frowns** [http://frowns.sf.net]
14. **PyBabel in MGLTools** [http://mgltools.scripps.edu]
15. **Babel v.1.6** [http://smog.com/chem/babel/]
16. **SWIG v.1.3.31** [http://www.swig.org]
17. **Boost.Python** [http://www.boost.org/libs/python/doc/]
18. **SIP – A Tool for Generating Python Bindings for C and C++ Libraries** [http://www.riverbankcomputing.co.uk/sip/]
19. **coverage.py** [http://nedbatchelder.com/code/modules/coverage.html]
20. **OpenBabel Python** [http://openbabel.sourceforge.net/wiki/Python]
21. Jaccard P: **La distribution de la flore dans la zone alpine.** *Rev Gen Sci Pures Appl* 1907, **18:**961-967.
22. Lipinski CA, Lombardo F, Dominy BW, Feeney PJ: **Experimental and computational approaches to estimate solubility and permeability in drug discovery and development settings.** *Adv Drug Del Rev* 1997, **23:**3-25.

Chemistry Central Journal

ChemistryCentral

Software

Open Access

Cinfony – combining Open Source cheminformatics toolkits behind a common interface

Noel M O'Boyle*[1] and Geoffrey R Hutchison[2]

Address: [1]Cambridge Crystallographic Data Centre, 12 Union Road, Cambridge CB2 1EZ, UK and [2]Department of Chemistry, University of Pittsburgh, Chevron Science Center, 219 Parkman Avenue, Pittsburgh, PA 15260, USA

Email: Noel M O'Boyle* - oboyle@ccdc.cam.ac.uk; Geoffrey R Hutchison - geoffh@pitt.edu

* Corresponding author

Published: 3 December 2008

Chemistry Central Journal 2008, **2**:24 doi:10.1186/1752-153X-2-24

Received: 9 October 2008

Accepted: 3 December 2008

This article is available from: http://journal.chemistrycentral.com/content/2/1/24

Abstract

Background: Open Source cheminformatics toolkits such as OpenBabel, the CDK and the RDKit share the same core functionality but support different sets of file formats and forcefields, and calculate different fingerprints and descriptors. Despite their complementary features, using these toolkits in the same program is difficult as they are implemented in different languages (C++ versus Java), have different underlying chemical models and have different application programming interfaces (APIs).

Results: We describe Cinfony, a Python module that presents a common interface to all three of these toolkits, allowing the user to easily combine methods and results from any of the toolkits. In general, the run time of the Cinfony modules is almost as fast as accessing the underlying toolkits directly from C++ or Java, but Cinfony makes it much easier to carry out common tasks in cheminformatics such as reading file formats and calculating descriptors.

Conclusion: By providing a simplified interface and improving interoperability, Cinfony makes it easy to combine complementary features of OpenBabel, the CDK and the RDKit.

Background

Cheminformatics toolkits are essential to the day-to-day work of the practising cheminformatician. They enable the user to deal with such tasks as handling different chemistry file formats, substructure searching, calculation of molecular fingerprints, and structure diagram generation. The main Open Source cheminformatics libraries under active development are OpenBabel [1], the Chemistry Development Kit (CDK) [2], and the RDKit [3]. OpenBabel is a C++ toolkit with bindings in Perl, Python, Ruby and Java, the CDK is a Java toolkit, while the RDKit is another C++ toolkit with Python bindings. While the CDK has its origins in academia, both OpenBabel and the RDKit originated in companies (OpenEye and Rational Discovery, respectively) and have subsequently been developed by the community under Open Source licenses.

In general, all of these toolkits share the same core functionality although the implementation details and underlying chemical model may differ. However, as a result of their independent development and history, each has functionality specific to itself and each toolkit supports different sets of file formats and forcefields, and can calculate different molecular fingerprints and molecular descriptors (Table 1). Despite the diversity of these toolkits and the potential benefits in being able to access all of them at the same time, there has been little work on interoperability between them. This has resulted in a balkanization of this field such that users of one toolkit rarely use another toolkit.

One way to achieve interoperability of chemical toolkits is through the use of standard file formats for exchange of

Table 1: Some features of toolkits which are not shared by all three toolkits.

CDK
A large number of descriptors (some overlap with RDKit)
Pharmacophore searching (like RDKit*)
Calculation of maximum common substructure
2D structure layout (like RDKit) and depiction
MACCS keys (also RDKit) and E-State fingerprints
Integration with the R statistical programming environment
Support for mass-spectrometry analysis (representations for cleavage reactions, structure generation from formulae)
Fragmentation schemes (ring fragments, Murcko)
3D structure generation using a template and heuristics (like OpenBabel)
3D similarity using ultrafast shape descriptors
Gasteiger π charge calculation
OpenBabel
Not just focused on cheminformatics
Supports a very large number of chemical file formats including quantum mechanics file formats, molecular mechanics trajectories, 2D sketchers
3D structure generation using a template method (like CDK)
Included in all major Linux distributions
Bindings available from several scripting languages apart from Python, as well as the Java and .NET platforms
Conformation generation and searching
InChI (also CDK) and InChIKey generation
Support for crystallographic space groups
Several forcefield implementations: UFF (also RDKit), MMFF94, MMFF94s, Ghemical
Ability to add custom data types to atoms, bonds, residues, molecules
RDKit
A large number of descriptors (some overlap with CDK)
Fragmentation using RECAP rules
2D coordinate generation (like CDK) and depiction
3D coordinate generation using geometry embedding
Calculation of Cahn-Ingold-Prelog stereochemistry codes (R/S)
Pharmacophore searching (like CDK)
Calculation of shape similarity (based on volume overlap)
Chemical reaction handling and transforms
Atom pairs and topological torsions fingerprints
Feature maps and feature-map vectors
Machine-learning algorithms

* Where the term "like" is used, it indicates that the implementation details differ.

data. For example, the CML project has defined a standardised XML format for chemical data [4], with successive releases refining and extending the original standard. The OpenSMILES effort [5] has attempted to resolve ambiguities in the published SMILES definition [6] to create a standard. While these efforts deserve support, they face inevitable problems achieving consensus and they require changes to existing software to support the standard. The large number of chemical file formats supported by OpenBabel (currently over 80) illustrates both the potential of achieving a standard as well as the difficulties.

An alternative is interoperability at the API (application programming interface) level. This has the advantage that it does require any changes to existing software. However, there are at least three barriers to overcome: the need for a programming language that can access all the toolkits simultaneously, the difficulty of exchanging chemical models between different toolkits, and differences in the API for core cheminformatics tasks shared by the toolkits.

Here we describe Cinfony, a Python module that overcomes these barriers to provide interoperability at the API level. Cinfony allows access to OpenBabel, the CDK, and the RDKit through a common interface, and uses a simple yet robust method to pass chemical models between toolkits. Pybel, one of the components of Cinfony, has been described previously [7]. It provides access to OpenBabel from standard Python. In this work, we show that the API developed for Pybel may be considered a generic API for accessing any cheminformatics toolkit. We describe the design and implementation of the Cinfony API for OpenBabel, the RDKit and the CDK. Next, we show how Cinfony simplifies the process of accessing the toolkits and how it can be used in practice to combine the power of the three Open Source toolkits. Finally, we dis-

cuss performance and some results from comparisons of the toolkits.

Implementation

Common Application Programming Interface

Cinfony presents the same interface to three cheminformatics toolkits, OpenBabel, the CDK and the RDKit. These are available through three separate modules: *openbabel*, *cdk* and *rdkit*. The API is designed to make it easy to carry out many of the common tasks in cheminformatics, and covers the core functionality shared by all of the toolkits. Table 2 gives an overview of the API. The complete API is available here (see Additional file 1).

The main class containing chemical information is the Molecule class. Rather than create a new chemical model, the Molecule class is a light wrapper around the molecule object in the underlying library, for example, around OBMol in the case of OpenBabel. Attribute values such as the molecular weight are calculated dynamically by querying the underlying molecule. This ensures that if the underlying OBMol, for example, is altered, the attribute values returned will still be correct. The actual underlying object (an OpenBabel OBMol, a CDK Molecule, or an RDKit Mol) can be accessed directly at any point.

The Molecule class also contains several methods that act on molecules such as methods for calculating fingerprints, adding hydrogens, and calculating descriptor values. This makes it easy to access these methods, and also brings them to the attention of the user. In the underlying toolkit these methods may not be present as part of the molecule class, and in fact, they can be difficult to find in the toolkit's API. For example, the Cinfony method Molecule.addh() adds explicit hydrogens to the molecule. Although the OBMol of OpenBabel has a corresponding method, OBMol.AddHydrogens(), the RDKit uses a global method, AddHs(Mol), while the CDK requires the user to instantiate a HydrogenAdder object, which can then be used to add hydrogens.

The Molecule methods described in the original Pybel API [7] have been extended to handle hydrogen addition and removal, structure diagram generation, assignment of 3D geometry to 0D structures and geometry optimisation using forcefields. Both the CDK and the RDKit are capable of 2D coordinate generation and 2D depiction. However, since OpenBabel currently has neither of these capabilities, a fourth toolkit, OASA, is used by Pybel for this purpose. OASA is a lightweight cheminformatics toolkit implemented in Python [8].

A new development in the latest version of OpenBabel is 3D coordinate generation and geometry optimisation using one of a number of forcefields. Since these methods are also available in the RDKit, and are under development in the CDK, two additional methods have been added to the Cinfony Molecule: make3D(), for 3D coordinate generation, and localopt(), for geometry optimisation. Particularly in the case of OpenBabel, these new methods simplify the process of generating 3D coordinates. Compare a single call to make3D() in Cinfony with the following OpenBabel code:

```
structuregenerator = openbabel.OBOp.Find
Type('Gen3D')

structuregenerator.Do(mol)

mol.AddHydrogens()
```

Table 2: An overview of the Cinfony API.

Class name	Purpose
Molecule	Wraps a molecule instance of the underlying toolkit and provides access to methods that act on molecules
Atom	Wraps an atom instance of the underlying toolkit
MoleculeData	Provides dictionary-like access to the information contained in the tag fields in SDF and MOL2 files
Outputfile	Handles multimolecule output file formats
Smarts	Wraps the SMARTS functionality of the toolkit in an analogous way to the Python 're' module for regular expression matching
Fingerprint	Simplifies Tanimoto calculation of binary fingerprints
Function name	
readfile	Return an iterator over Molecules in a file
readstring	Return a Molecule
Variable name	
descs	A list of descriptor IDs
forcefields	A list of forcefield IDs
fps	A list of fingerprint IDs
informatsaa	A list of input format IDs
outformats	A list of output format IDs

```
ff = openbabel.OBForceField.FindType("MMFF94")

ff.Setup(mol)

ff.SteepestDescent(50)

ff.GetCoordinates(mol)
```

The Cinfony API is identical for all of the toolkits. However, the values returned by particular API calls are not necessarily standardised across toolkits. This Cinfony design decision is in agreement with the Principle of Least Surprise [9]; when the user accesses the underlying toolkit directly, they will get the same result as found when using Cinfony. This design decision places the responsibility on the user to become familiar with differences in how the toolkits behave. For example, all of the toolkits allow the calculation of path-based fingerprints. These encode all paths in the molecular graph up to a path length of *P* into a binary vector of length *V*, but the default values for *V* and *P* are different for each toolkit: 1024 and 7 for OpenBabel, 1024 and 8 for the CDK, and 2048 and 7 for RDKit. Although it is possible to alter these parameters for the CDK and the RDKit and so standardise *V* and *P* to 1024 and 7 for all of the toolkits, it is reasonable to assume that the developers of each package have chosen sensible defaults. In addition, the implementation details of each of the fingerprinters would still be different; for example, the RDKit sets four bits when hashing each molecular path, the others set one; OpenBabel does not set any bits for the one-atom fragments, N, C and O.

Interoperability

The ability to transfer chemical models between toolkits is essential to the goal of interoperability. However, the internal representation of a molecule is specific to a particular toolkit. For example, as well as the connection table and coordinates (if present), it may include derived data relating to aromaticity, the number of implicit hydrogens on an atom, or stereochemical configuration. Fortunately, the problem of transfer and storage of chemical information has already been solved by the development of molecular file formats, of which over 80 are now supported by OpenBabel. Specifically, the MDL MOL file format [10] and the SMILES format [5,6] are shared by all three toolkits, and are used by Cinfony to exchange information on molecules with 2D or 3D coordinates (MOL file format), and no coordinates (SMILES format), respectively.

By using existing file formats rather than trying to interconvert the internal models themselves, Cinfony takes advantage of the existing input/output code of each toolkit which is well-tested and mature. In addition, the translation process is transparent to the user. However, the user should be aware of known limitations of particular readers or writers. For example, the SMILES parser in CDK 1.0.3 ignores atom-based stereochemistry and thus that information is lost if a 0D *rdkit* or *obabel* Molecule with atom-based stereochemistry is converted to a *cdk* Molecule.

Cinfony Molecules are interconverted using the Molecule() constructor. For example, if obabelmol is an *obabel* Molecule, then the corresponding *rdkit* Molecule can be constructed using rdkit.Molecule(pybelmol). This mechanism can also be used to interface Cinfony to other cheminformatics toolkits. The only requirements are that the object passed to the Molecule() constructor needs to have a _cinfony attribute set to True, and an _exchange attribute containing a tuple (0, SMILES string) or (1, MOL file) depending on whether the molecule is 0D or not.

Implementation

The Python scripting language has two main implementations. The most widely used implementation is the original reference implementation of Python in C, referred to as CPython when necessary to distinguish it from other implementations. The next most widely used implementation is Jython, an implementation of Python in Java. Although most users of Python do so through CPython, Jython scripts have the advantage of being able to access Java libraries natively. They can also be compiled into Java classes to be used from Java programs. Jython scripts are also useful in contexts where Java is required but it is more convenient to work in Python; for example, to implement a Java web servlet or a node in a Java workflow environment such as KNIME [11].

As discussed earlier, one of the barriers to interoperability is the requirement for a programming language that can simultaneously access more than one of the toolkits. From CPython it is possible to use Cinfony modules to connect to OpenBabel (*pybel*), the CDK (*cdkjpype*) and the RDKit (*rdkit*). From Jython, there are modules for OpenBabel (*jybel*) and the CDK (*cdkjython*). Convenience modules *obabel* and *cdk* are provided that automatically import the appropriate OpenBabel or CDK module depending on the Python implementation. The relationship between these Cinfony modules and the underlying cheminformatics libraries is summarised in Figure 1.

pybel and jybel

OpenBabel provides SWIG [12] bindings for both CPython and Java (among other languages). *pybel* is a wrapper around the CPython bindings, and has previously been described in detail [7]. *jybel* is an implementation of the Cinfony API that allows the user to access OpenBabel from Jython using the Java bindings. Despite the fact that

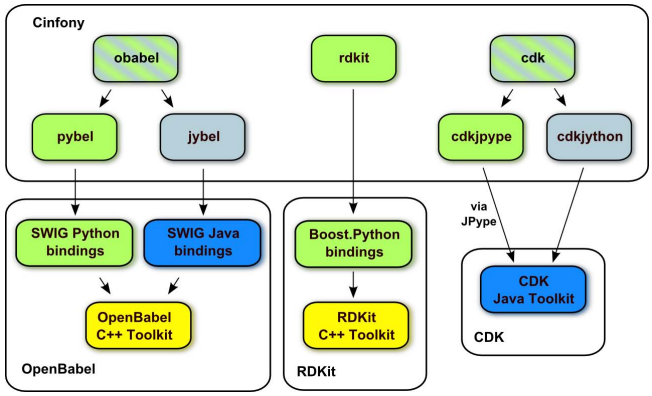

Figure 1
Relationship of Cinfony modules to Open Source toolkits. Python modules are accessible from CPython (green), Jython (pale blue), or both (striped green and pale blue). Java libraries are indicated by dark blue, while C++ libraries are yellow.

jybel is used from a Java implementation of Python, and accesses a C++ library through the Java Native Interface (JNI), the *jybel* code differs from *pybel* in very few respects. In Jython, it is not possible to iterate directly over the wrapped STL vectors used by OpenBabel as their Java SWIG bindings do not implement the Iterable interface. Also, the current Jython implementation is 2.2 and does not support generator expressions, which were introduced in Python 2.4. Although both C++ and Python have the concept of a global function or variable, this is not the case in Java. SWIG places such functions, and get/set methods for accessing the variables, in a special class named openbabel. Global constants are placed in another class called openbabelConstants. A convenience module, *obabel*, is provided which automatically imports the appropriate module depending on the Python implementation.

cdkjpype and cdkjython

Since Jython runs on top of the Java Virtual Machine (JVM), it can access Java libraries such as the CDK natively. To access Java libraries from CPython, the Python library JPype [13] is needed. This starts an instance of the JVM and uses the JNI to communicate back and forth. Overall, the differences between the two wrappers are minor. Jython and JPype differ in the syntax used to handle Java exceptions. Also, JPype returns unicode strings from the CDK and these need to be converted to regular strings (otherwise problems arise if they are passed to an OpenBabel method expecting a std::string). The appropriate CDK wrapper, *cdkjpype* or *cdkjython*, will be imported if the user imports the convenience module *cdk*.

rdkit

Support for Python scripting has been part of the design of the RDKit from the start. The Python bindings in RDKit were created using Boost.Python [14], a framework for interfacing Python and C++. The Cinfony module *rdkit* uses these bindings to implement its API. It is currently not possible to access RDKit from Jython. RDKit has only preliminary support for Java bindings; when these are complete, a corresponding module will be added to Cinfony.

Dependency handling

A fully-featured installation of Cinfony relies on a large number of open source libraries. In particular, the 2D depiction capabilities introduce dependencies on several graphics libraries which may be problematic to install on a particular platform (Cairo and its Python bindings, Python Imaging Library, AGG and the Python wrapper AggDraw). With this in mind, Cinfony treats all dependencies as optional and only raises an Exception if the user calls a method or imports a module that requires a missing dependency.

For example, the Python Imaging Library (PIL) is required for displaying a 2D depiction on the screen. If all of the components of cinfony are installed except for PIL, Cinfony works perfectly except that an Exception is raised if the Molecule.draw() method is called with show = True (the default). The image can however be written to a file without problems (show = False, filename = "image.png"). Similarly, if a user is only interested in using the CDK and the RDKit, it is not necessary to install OpenBabel.

Full installation instructions for Windows, MacOSX and Linux are available from the Cinfony website. It should be noted that for Windows users, there is no need to compile or search for missing libraries as the dependencies are included as binaries in the Cinfony distribution.

Results

Cinfony API

The original Pybel API was designed to make it easy to use OpenBabel to perform the most common tasks in cheminformatics and to do so using idiomatic Python. Subsequently, we realised that the resulting API could be considered a generic API for wrapping the core functionality of any cheminformatics toolkit. Cinfony implements an extended version of the original Pybel API for the CDK and the RDKit, as well as OpenBabel. While the original Pybel was restricted to CPython, Cinfony can also be used from Jython to access the CDK and OpenBabel.

Cinfony helps cheminformaticians avoid the steep learning curve associated with starting to use a new toolkit.

With Cinfony, all of the core functionality of the toolkits can be accessed with the same interface. For example, in Cinfony, a molecule can be created from a SMILES string with:

```
mol = toolkit.readstring("smi", SMILESstring)
```

RDKit

```
mol = Chem.MolFromSmiles(SMILESstring)
```

OpenBabel

```
mol = openbabel.OBMol()

obconversion = openbabel.OBConversion()

obconversion.SetInFormat("smi")

obconversion.ReadString(mol, SMILESstring)
```

CDK

```
builder = cdk.DefaultChemObjectBuilder.getInstance()

sp = cdk.smiles.SmilesParser(builder)

mol = sp.parseSmiles(SMILESstring)
```

The RDKit was designed with Python scripting in mind, and of the three toolkits is the most concise. On the other hand, OpenBabel uses a characteristically C++ approach. An empty molecule is created, and is passed to an OBConversion instance as a container for the molecule read from the SMILES string. The SmilesParser in the CDK requires an instance of an object implementing the IChemObjectBuilder interface.

Another advantage of a common API is that a script written for one toolkit can easily be modified to use another. As an example, here is a script that selects molecules that are similar to a particular target molecule. This script is taken from the original Pybel paper [7], but uses the CDK instead of OpenBabel and will run equally well from Jython and CPython. The only differences compared to the original script are that "pybel" has been replaced with "cdk", and the import statement has been changed from "import pybel":

```
from cinfony import cdk

targetmol = cdk.readfile("sdf", "targetmol.sdf").next()
targetfp = targetmol.calcfp()

output = cdk.Outputfile("sdf", "similarmols.sdf")

for mol in cdk.readfile("sdf", "inputfile.sdf"):

   fp = mol.calcfp()

   if fp | targetfp >= 0.7:

      output.write(mol)

output.close()
```

Alternatively, we could just have made a single change to the original script, by replacing the import statement from "import pybel" with "from cinfony import cdk as pybel".

Using Cinfony to combine toolkits

Another goal of Cinfony is to make it easy to combine toolkits in the same script. This allows the user to exploit the complementary capabilities of different toolkits (Table 1). For example, let's suppose the user wants to (1) convert a SMILES string to 3D coordinates with OpenBabel, then (2) create a 2D depiction of that molecule with the RDKit, next (3) calculate descriptors with the CDK, and finally (4) write out an SDF file containing the descriptor values and the 3D coordinates. The full Python script is only seven lines long:

```
from cinfony import rdkit, cdk, obabel

mol = obabel.readstring("smi", "CCC=O")

mol.make3D()

rdkit.Molecule(mol).draw(show = False, filename = "aldehyde.png")

descs = cdk.Molecule(mol).calcdesc()

mol.data.update(descs)

mol.write("sdf", filename = "aldehyde.sdf")
```

For cheminformaticians interested in developing QSAR or QSPR models, Cinfony can be used to simultaneously calculate descriptors from the RDKit, the CDK and OpenBabel. For example, the following script reads a multiline input file, with each line consisting of a SMILES string followed by a property value. For each molecule, it calculates all of the OpenBabel, RDKit and CDK descriptors (except for CDK's CPSA) and writes out the results as a tab-sepa-

rated file suitable for reading with the statistical package R [15]. Note that in this example script, if descriptors share the same name only one is retained. This is the case for the TPSA descriptor in OpenBabel, which is replaced by the RDKit's TPSA descriptor.

```
import string

from cinfony import obabel, cdk, rdkit

# Read in SMILES strings and observed prop
erty values

smiles, propvals = [], []

for line in open("data.txt"):

   broken = line.rstrip().split()

   smiles.append(broken [0])

   propvals.append(float(broken))

mols = [obabel.readstring("smi", smile)
for smile in smiles]

# Calculate descriptor values using
OpenBabel,

# the CDK (apart from 'CPSA') and the RDKit

cdkdescs = [x for x in cdk.descs if x !=
'CPSA']

descs = []

for mol in mols:

   d = mol.calcdesc()

   d.update(cdk.Molecule(mol).calcdesc(cd
kdescs))

   d.update(rdkit.Molecule(mol).calcdesc(
))

   descs.append(d)

# Write a file suitable for 'read.table'
in R

outputfile = open("inputforR.txt", "w")

descnames = sorted(descs [0].keys(), key =
string.lower)

print >> outputfile, "\t".join(["Prop
erty"] + descnames)

for smile, propval, desc in zip(smiles,
propvals, descs):

   descvals = [str(desc[descname]) for
descname in descnames]

   print >> outputfile, "\t".join([smile,
str(propval)] +

descvals)

outputfile.close()
```

Performance

Accessing cheminformatics libraries using Cinfony allows the user to rapidly develop scripts that manipulate chemical information. However, there is a small price to be paid. Firstly, there is the cost of moving objects across the interface between Python and the cheminformatics libraries. Secondly, the additional code required by Cinfony to implement a standard API may slow performance further.

To assess the performance penalty for accessing cheminformatics toolkits using Cinfony rather than directly in the native language, we looked at two simple test cases: (1) iterating over an SDF file containing 25419 molecules, (2) iterating and printing out the molecular weight of each of the molecules. The SDF file used was 3_p0.0.sdf, the first portion of the drug-like subset of the ZINC 7.00 dataset [16]. The Cinfony scripts, Java and C++ source code are available as Additional file 2. The results are shown in Table 3.

While accessing the CDK using Jython is almost as fast as a pure Java implementation, there is a considerable overhead associated with using JPype to access the CDK from CPython (89% slower for the second test case). This overhead is due to passing objects between the JVM and CPython. For OpenBabel, there is little performance cost associated with accessing OpenBabel from either implementation of Python, although the *jybel* scripts are somewhat slower than *pybel* scripts. A small portion of this speed difference can be attributed to a slower startup (about 1.6 seconds for *jybel*, compared to 0.8 seconds for *pybel*). Finally, from the RDKit results in Table 3, it is clear that using Boost.Python to wrap a C++ library is more efficient than using SWIG. The difference in run times between the C++ and Python implementations is negligible.

In practice, the performance of a particular Cinfony script will depend on the extent to which information is passed

Table 3: Performance of Cinfony modules compared to a native Java or C++ implementation.

	Iterate over SDF		Iterate and calculate molecular weight	
CDK	Time (s)	Normalised	Time (s)	Normalised
Native Java	21.2	1.00	36.8	1.00
cdkjython	23.1	1.09	41.6	1.13
cdkjpype	33.0	1.57	69.5	1.89
OpenBabel				
Native C++	31.9	1.00	43.0	1.00
pybel	34.1	1.07	45.1	1.05
jybel	38.0	1.19	49.6	1.15
RDKit				
Native C++	99.7	1.00	100.7	1.00
rdkit	99.9	1.00	101.0	1.00

The times reported are wallclock times from the best of three runs on a dual-core Intel Pentium 4 3.2 GHz machine with 1GB RAM.

back and forth between Python and the underlying Java or C++ library. Where most of the time is spent on computation in the underlying library, the speed difference between a native implementation and one using Cinfony is expected to be small.

Comparison of toolkits

Cinfony makes it easy to compare the results obtained by different toolkits for the same operations. This can be useful in identifying bugs, applying a test suite, or finding the strengths and weaknesses of particular implementations. For example, where different toolkits calculate the same descriptors, if the calculated values are not highly correlated it may indicate a bug in one or the other. Earlier, we mentioned that a difference in the treatment of implicit hydrogens causes different toolkits to give different values for molecular weight unless hydrogens are explicitly added. Ensuring that a particular result is in agreement with that obtained by another toolkit can act as a sanity check in such instances to avoid errors.

When carrying out the same operation with several toolkits, it is often convenient to iterate over the toolkits in an outer loop:

```
from cinfony import obabel, rdkit, cdk

for toolkit in [obabel, rdkit, cdk]:

   print toolkit.readstring("smi",
"CCC").molwt
```

As an example of how such comparisons can be used to identify bugs in toolkits, let us consider depiction. As a dataset, we randomly chose 100 molecules from PubChem [17], with subsequent filtering to remove multicomponent molecules. For each molecule, PubChem provides an SDF file containing coordinates for a 2D depiction, as well as the depiction itself as a PNG file. PubChem uses the CACTVS toolkit [18] to generate the 2D coordinates as well as the corresponding depiction. Using a script similar to the following, we used Cinfony to generate 2D depictions using OASA (the depiction library used by *pybel*), the CDK and a development version of RDKit that all use the same 2D coordinates taken from the SDF file:

```
from cinfony import pybel, rdkit

for toolkit in [rdkit, pybel]:

   name = toolkit.__name__

   for mol in toolkit.readfile("sdf",
"dataset.sdf"):

      mol.draw(filename = "%s_%s.png" %
(mol.title, name),

         show = False,

         usecoords = True)
```

When the resulting images were compared for the PubChem entry CID7250053, an error was found in the depiction of the stereochemistry of an isopropyl group (Figure 2). Since the error only occurred in certain cases, it had not been previously noticed and would have been difficult to identify without such a comparative study. Once reported, the problem was quickly solved and the subsequent RDKit release depicted the stereochemistry correctly. A comparison of depictions by commercial toolkits

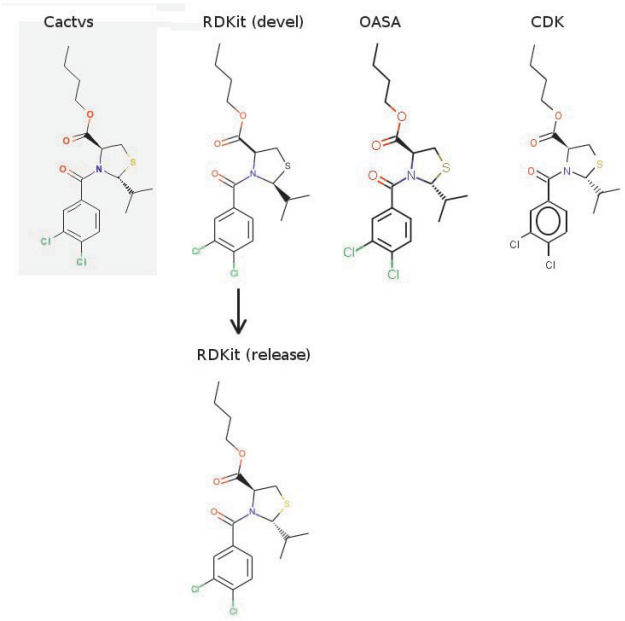

Figure 2
Comparison of depictions of PubChem CID7250053 using different toolkits. The depiction using the development version of RDKit showed incorrect stereochemistry for the isopropyl substituent of the thiazole ring.

and depictions generated by Cinfony is available here (see Additional file 3).

Conclusion

Cinfony makes it easy to combine complementary features of the three main Open Source cheminformatics toolkits. By presenting a standard simplified API, the learning curve associated with starting to use a new toolkit is greatly reduced, thus encouraging users of one toolkit to investigate the potential of others.

Cinfony is freely available from the Cinfony website [19], both as Python source code and as a Windows distribution containing dependencies. Installation instructions are provided for MacOSX, Linux and Windows.

Availability and requirements

Project name: Cinfony

Project home page: http://cinfony.googlecode.com

Operating system(s): Platform independent

Programming language: Python, Jython

Other requirements: OpenBabel, CDK, RDKit, Java, OASA, JPype, Python Imaging Library

License: BSD

Any restrictions to use by non-academics: None

Competing interests

The authors declare that they have no competing interests.

Authors' contributions

NMOB conceived and developed Cinfony. GRH is the lead developer of OpenBabel and created the Python and Java SWIG bindings. All authors read and approved the final manuscript.

Additional material

Additional file 1

Miniwebsite API. *A mini-website of the Cinfony API documentation.*
Click here for file
[http://www.biomedcentral.com/content/supplementary/1752-153X-2-24-S1.zip]

Additional file 2

Timing Code. *A zip file containing Python, Java and C++ code used for run time comparisons for two test cases.*
Click here for file
[http://www.biomedcentral.com/content/supplementary/1752-153X-2-24-S2.zip]

Additional file 3

Miniwebsite Depictions. *A mini-website showing a comparison of the depictions generated by several cheminformatics toolkits.*
Click here for file
[http://www.biomedcentral.com/content/supplementary/1752-153X-2-24-S3.zip]

Acknowledgements

Cinfony would not be possible without the work of many Open Source projects. In particular, we thank several developers who responded quickly to bug reports or queries: Beda Kosata (OASA), Greg Landrum (RDKit), Tim Vandermeersch (OpenBabel), Steve Ménard (JPype). Thanks also to Gilbert Mueller and Chris Morley for feedback on installing Cinfony. NMOB thanks Google Code for providing free web hosting and development tools for Cinfony. We thank the anonymous reviewers for several useful suggestions.

References

1. **OpenBabel v.2.2.0** [http://openbabel.org]
2. Steinbeck C, Hoppe C, Kuhn S, Floris M, Guha R, Willighagen E: **Recent Developments of the Chemistry Development Kit (CDK) – An Open-Source Java Library for Chemo- and Bioinformatics.** *Curr Pharm Des* 2006, **12:**2110-2120.
3. Landrum G: **RDKit.** [http://www.rdkit.org].
4. Murray-Rust P, Rzepa HS: **Chemical Markup, XML, and the Worldwide Web. 1. Basic Principles.** *J Chem Inf Comput Sci* 1999, **39:**928-942.

5. Apodaca R, O'Boyle N, Dalke A, Van Drie J, Ertl P, Hutchison G, James CA, Landrum G, Morley C, Willighagen E, De Winter H: **OpenSMILES.** [http://www.opensmiles.org].
6. **Daylight Chemical Information Systems Manual** [http://www.daylight.com/dayhtml/doc/theory/theory.smiles.html]
7. O'Boyle NM, Morley C, Hutchison GR: **Pybel: a Python wrapper for the OpenBabel cheminformatics toolkit.** *Chem Cent J* 2008, **2:**5.
8. Kosata B: **OASA.** [http://bkchem.zirael.org/oasa_en.html].
9. Raymond ES: *The Art of UNIX Programming* 2003 [http://www.catb.org/~esr/writings/taoup/index.html]. Reading, MA: Addison-Wesley
10. **Symyx CTfile formats** [http://www.mdli.com/downloads/public/ctfile/ctfile.jsp]
11. **KNIME – Konstanz Information Miner** [http://knime.org]
12. **SWIG v.1.3.36** [http://www.swig.org]
13. Ménard S: **JPype.** [http://jpype.sf.net].
14. **Boost.Python** [http://www.boost.org/libs/python/doc/]
15. R development core team: **R: A language and environment for statistical computing.** [http://www.R-project.org].
16. Irwin JJ, Shoichet BK: **ZINC – A Free Database of Commercially Available Compounds for Virtual Screening.** *J Chem Inf Model* 2005, **45:**177-182.
17. **PubChem** [http://pubchem.ncbi.nlm.nih.gov/]
18. **CACTVS Chemoinformatics Toolkit: Xemistry GmbH: Lahntal, Germany.** .
19. O'Boyle NM: **Cinfony.** [http://cinfony.googlecode.com].

Journal of Cheminformatics

SOFTWARE Open Access

Open Babel: An open chemical toolbox

Noel M O'Boyle[1], Michael Banck[2], Craig A James[3], Chris Morley[4], Tim Vandermeersch[4] and Geoffrey R Hutchison[5*]

Abstract

Background: A frequent problem in computational modeling is the interconversion of chemical structures between different formats. While standard interchange formats exist (for example, Chemical Markup Language) and *de facto* standards have arisen (for example, SMILES format), the need to interconvert formats is a continuing problem due to the multitude of different application areas for chemistry data, differences in the data stored by different formats (0D versus 3D, for example), and competition between software along with a lack of vendor-neutral formats.

Results: We discuss, for the first time, Open Babel, an open-source chemical toolbox that speaks the many languages of chemical data. Open Babel version 2.3 interconverts over 110 formats. The need to represent such a wide variety of chemical and molecular data requires a library that implements a wide range of cheminformatics algorithms, from partial charge assignment and aromaticity detection, to bond order perception and canonicalization. We detail the implementation of Open Babel, describe key advances in the 2.3 release, and outline a variety of uses both in terms of software products and scientific research, including applications far beyond simple format interconversion.

Conclusions: Open Babel presents a solution to the proliferation of multiple chemical file formats. In addition, it provides a variety of useful utilities from conformer searching and 2D depiction, to filtering, batch conversion, and substructure and similarity searching. For developers, it can be used as a programming library to handle chemical data in areas such as organic chemistry, drug design, materials science, and computational chemistry. It is freely available under an open-source license from http://openbabel.org.

Introduction

The history of chemical informatics has included a huge variety of textual and computer representations of molecular data. Such representations focus on specific atomic or molecular information and may not attempt to store all possible chemical data. For example, line notations like Daylight SMILES [1] do not offer coordinate information, while crystallographic or quantum mechanical formats frequently do not store chemical bonding data. Hydrogen atoms are frequently omitted from x-ray crystallography due to the difficulty in establishing coordinates, and are often ignored by some file formats as the "implicit valence" of heavy atoms that indicates their presence. Other types of representations require specification of atom types on the basis of a specific valence bond model, inclusion of computed partial charges, indication of biomolecular residues, or multiple conformations.

While attempts have been made to provide a standard format for storing chemical data, including most notably the development of Chemical Markup Language (CML) [2-6], an XML dialect, such formats have not yet achieved widespread use. Consequently, a frequent problem in computational modeling is the interconversion of molecular structures between different formats, a process that involves extraction and interpretation of their chemical data and semantics.

We outline for the first time, the development and use of the Open Babel project, a full-featured open chemical toolbox, designed to "speak" the many different representations of chemical data. It allows anyone to search, convert, analyze, or store data from molecular modeling, chemistry, solid-state materials, biochemistry, or related areas. It provides both ready-to-use programs as well as a complete, extensible programmer's toolkit for developing cheminformatics software. It can handle reading,

* Correspondence: geoffh@pitt.edu
[5]University of Pittsburgh, Department of Chemistry, 219 Parkman Avenue, Pittsburgh, PA 15217, USA
Full list of author information is available at the end of the article

writing, and interconverting over 110 chemical file formats, supports filtering and searching molecule files using Daylight SMARTS pattern matching [7] and other methods, and provides extensible fingerprinting and molecular mechanics frameworks. We will discuss the frameworks for file format interconversion, fingerprinting, fast molecular searching, bond perception and atom typing, canonical numbering of molecular structures and fragments, molecular mechanics force fields, and the extensible interfaces provided by the software library to enable further chemistry software development.

Open Babel has its origin in a version of OELib released as open-source software by OpenEye Scientific under the GPL (GNU Public License). In 2001, OpenEye decided to rewrite OELib in-house as the proprietary OEChem library, so the existing code from OELib was spun out into the new Open Babel project. Since 2001, Open Babel has been developed and substantially extended as an international collaborative project using an open-source development model [8]. It has over 160,000 downloads, over 400 citations [9], is used by over 40 software projects [10], and is freely available from the Open Babel website [11].

Features

File Format Support

With the release of Open Babel 2.3, Open Babel supports 111 chemical file formats in total. It can read 82 formats and write 85 formats. These encompass common formats used in cheminformatics (SMILES, InChI, MOL, MOL2), input and output files from a variety of computational chemistry packages (GAMESS, Gaussian, MOPAC), crystallographic file formats (CIF, ShelX), reaction formats (MDL RXN), file formats used by molecular dynamics and docking packages (AutoDock, Amber), formats used by 2D drawing packages (ChemDraw), 3D viewers (Chem3D, Molden) and chemical kinetics and thermodynamics (ChemKin, Thermo). Formats are implemented as "plugins" in Open Babel, which makes it easy for users to contribute new file formats (see Extensible Interface below). Depending on the format, other data is extracted by Open Babel in addition to the molecular structure; for example, vibrational frequencies are extracted from computational chemistry log files, unit cell information is extracted from CIF files, and property fields are read from SDF files.

A number of "utility" file formats are also defined; these are not strictly speaking a way of storing the molecular structure, but rather present certain functionality through the same interface as the regular file formats. For example, the *report format* is a write-only utility format [12] that presents a summary of the molecular structure of a molecule; the *fingerprint format* [13] and *fastsearch format* [14] are used for similarity and substructure searching (see below); the *MolPrint2D* and *Multilevel Neighborhoods of Atoms* formats calculate circular fingerprints defined by Bender *et al.* [15,16] and Filimonov *et al.* [17,18] respectively.

Each format can have multiple options to control either reading or writing a particular format. For example, the InChI format has 12 options including an option "K" to generate an InChIKey, "T <param>" to truncate the InChI depending on a supplied parameter and "w" to ignore certain InChI warnings. The available options are listed in the documentation, are shown in the Graphical User Interface (GUI) as checkboxes or textboxes, and can be listed at the command-line. In fact, all three are generated from the same source; a documentation string in the C++ code.

Fingerprints and Fast Searching

Databases are widely used to store chemical information especially in the pharmaceutical industry. A key requirement of such a database is the ability to index chemical structures so that they can be quickly retrieved given a query substructure. Open Babel provides this functionality using a path-based fingerprint. This fingerprint, referred to as *FP2* in Open Babel, identifies all linear and ring substructures in the molecule of lengths 1 to 7 (excluding the 1-atom substructures C and N) and maps them onto a bit-string of length 1024 using a hash function. If a query molecule is a substructure of a target molecule, then all of the bits set in the query molecule will also be set in the target molecule. The fingerprints for two molecules can also be used to calculate structural similarity using the Tanimoto coefficient, the number of bits in common divided by the union of the bits set.

Clearly, repeated searching of the same set of molecules will involve repeated use of the same set of fingerprints. To avoid the need to recalculate the fingerprints for a particular multi-molecule file (such as an SDF file), Open Babel provides a *fastindex* format that solely stores a fingerprint along with an index into the original file. This index leads to a rapid increase in the speed of searching for matches to a query - datasets with several million molecules are easily searched interactively. In this way, a multi-molecule file may be used as a lightweight alternative to a chemical database system.

Bond Perception and Atom Typing

As mentioned above, many chemical file formats offer representations of molecular data solely as lists of atoms. For example, most quantum chemical software packages and most crystallographic file formats do not offer definitions of bonding. A similar situation occurs in the case of the Protein Data Bank (PDB) format; while standardized [19] files contain connectivity

information, non-standard files exist that often do not provide full connectivity information. Consequently, Open Babel features methods to determine bond connectivity, bond order perception, aromaticity determination, and atom typing.

Bond connectivity is determined by the frequently used algorithm of detecting atoms closer than the sum of their covalent radii, with a slight tolerance (0.45 Å) to allow for longer than typical bonds. To handle disorder in crystallographic data (e.g., PDB or CIF files), atoms closer than 0.63 Å are not bonded. A further filtering pass is made to ensure standard bond valency is maintained; each element has a maximum number of bonds, if this is exceeded then the longest bonds to an atom are successively removed until the valence rule is fulfilled.

After bond connectivity is determined, if needed or requested by the user, bond order perception is performed on the basis of bond angles and geometries. The method is similar to that proposed by Roger Sayle [20] and uses the average bond angle around an un-typed atom to determine sp and sp^2 hybridized centers. 5-membered and 6-membered rings are checked for planarity to estimate aromaticity. Finally, atoms marked as unsaturated are checked for an unsaturated neighbor to give a double or triple bond. After this initial atom typing, known functional groups are matched, followed by aromatic rings, followed by remaining unsatisfied bonds based on a set of heuristics for short bonds, atomic electronegativity, and ring membership.

Atom typing is performed by "lazy evaluation," matching atoms against SMARTS patterns to determine hybridization, implicit valence, and external atom types. Atom type perception may be triggered by adding hydrogens (which requires determination of implicit and explicit valence), exporting to a file format that requires atom types, or as requested by the user. To minimize the amount of typing required, when importing from a format with atom types specified, a lookup table is used to translate between equivalent types.

An important part of atom typing is aromaticity detection and assignment of Kekulé bond orders (kekulization). In Open Babel, a central aromaticity model is used, largely matching the commonly used Daylight SMILES representation [1], but with added support for aromatic phosphorous and selenium. Potential aromatic atoms and bonds are flagged on the basis of membership in a ring system possibly containing 4n+2 π electrons. Aromaticity is established only if a well-defined valence bond Kekulé pattern can be determined. To do this, atoms are added to a ring system and checked against the 4n+2 π electron configuration, gradually increasing the size to establish the largest possible connected aromatic ring system. Once this ring system is determined, an exhaustive search is performed to assign single and double bonds to satisfy all valences in a Kekulé form. Since this process is exponential in complexity, the algorithm will terminate if more than 30 levels of recursion or 15 seconds are exceeded (which may occur in the case of large fused ring systems such as carbon nanotubes).

Canonical Representation of Molecules

In general, for any particular molecular structure and file format, there are a large number of possible ways the structure could be stored; for example, there are N! ways of ordering the atoms in an MOL file. While each of the orderings encodes exactly the same information, it can be useful to define a canonical numbering of the atoms of a molecule and use this to derive a canonical representation of a molecule for a particular file format. For a zero-dimensional file format without coordinates, such as SMILES, the canonical representation could be used to index a database, remove duplicates or search for matches.

Open Babel implements a sophisticated canonicalization algorithm that can handle molecules or molecular fragments. The atom symmetry classes are the initial graph invariants and encode topological and chemical properties. A cooperative labeling procedure is used to investigate the automorphic permutations to find the canonical code. Although the algorithm is similar to the original Morgan canonical code [21], various improvements are implemented to improve performance. Most notably, the algorithm implements heuristics from the popular nauty package [22,23]. Another aspect handled by the canonical code is stereochemistry as different labelings can lead to different parities. This is further complicated by the possibility of symmetry-equivalent stereocenters and stereocenters whose configuration is interdependent. The full details will be the subject of a separate publication.

Coordinate Generation in 2D and 3D

Open Babel, version 2.3, has support for 2D coordinate generation (Figure 1) through the donation of code by Sergei Trepalin, based on the code used in the MCDL chemical structure editor [24-26]. The MCDL algorithm aims to layout the molecular structure in 2D such that all bond lengths are equal and all bond angles are close to 120°. The layout algorithm includes a small database of around 150 templates to help layout cages and large fragment cycles. To deal with the problem of overlapping fragments, the algorithm includes an exhaustive search procedure that rotates around acyclic bonds by 180°.

Coordinate generation in 3D was introduced in Open Babel version 2.2, and improved in version 2.3, to enable

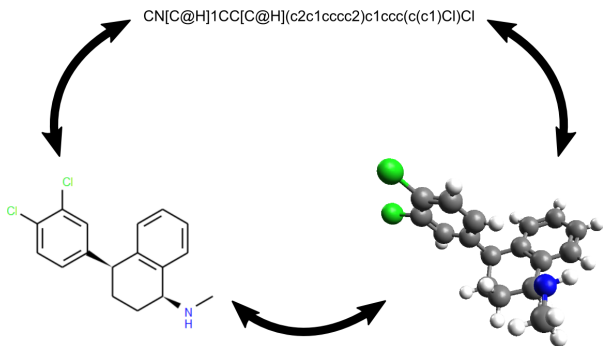

Figure 1 Interconversion of 0D, 2D and 3D structures. The structures shown are of sertraline, a selective serotonin reuptake inhibitor (SSRI) used in the treatment of depression. A SMILES string for sertraline is shown at the top; this can be considered a 0D structure (only connectivity and stereochemical information). From this, Open Babel can generate a 2D structure (bottom left, depicted by Open Babel) or a 3D structure (bottom right, depicted by Avogadro), and all of these can be interconverted.

conversion from 0D formats such as SMILES to 3D formats such as SDF (Figure 1). The 3D structure generator builds linear components from scratch following geometrical rules based on the hybridization of the atoms. Single-conformer ring templates are used for ring systems. The template matching algorithm iterates through the templates from largest to smallest searching for matches. If a match is found, the algorithm continues but will not match any ring atoms previously templated except in the case of a single overlap (the two ring systems of a spiro group) or an overlap involving exactly two adjacent atoms (two fused ring systems). After an initial structure is generated, the stereochemistry (cis/trans and tetrahedral) is corrected to match the input structure. Finally, the energy of the structure is minimized using the MMFF94 forcefield [27-31] and a low energy conformer found using a weighted rotor search.

While the 3D structure builder produces reasonable conformations for molecules without rings or with ring systems for which a template exists, the results may be poor for molecules with more complex ring systems or organometallic species. Future work will be performed to compare the results of Open Babel with other programs with respect to both speed and the quality of the generated structures [32].

Stereochemistry

A recent focus of Open Babel development has been to ensure robust translation of stereochemical information between file formats. This is particularly important when dealing with 0D formats as these explicitly encode the perceived stereochemistry. Open Babel 2.3 includes classes to handle cis/trans double bond stereochemistry, tetrahedral stereochemistry and square-planar stereochemistry (this last is still under development), as well as perception routines for 2D and 3D geometries, and routines to query and alter the stereochemistry.

The detection of stereogenic units starts with an analysis of the graph symmetry of the molecule to identify the symmetry class of each atom. However, given that a complete symmetry analysis also needs to take stereochemistry into account, this means that the overall stereochemistry can only be found iteratively. At each iteration, the current atom symmetry classes are used to identify stereogenic units. For example, a tetrahedral center is identified as chiral if it has four neighbors with different symmetry classes (or three, in the case where a lone pair gives rise to the tetrahedral shape).

Forcefields

Molecular mechanics functions are provided for use with small molecules. Typical applications include energy evaluation or minimization, alone or as part of a larger workflow. The selection of implemented force fields allows most molecular structures to be used and parameters to be assigned automatically. The MMFF94 (s) force field can be used for organic or drug-like molecules [27-31]. For molecules containing any element of the periodic table or complex geometry (i.e. not supported by MMFF94), the UFF force field can be used instead [33]. Recently, code implementing the GAFF force field [34,35] was also contributed and released as part of version 2.3. All of the forcefields allow the application of constraints on particular atom positions, or particular distances.

Several conformer searching methods have been implemented using the forcefields, all based on the "torsion-driving" approach. This approach involves setting torsion angles from a set of predefined allowed values for a particular rotatable bond. The most thorough search method implemented is a systematic search method, which iterates over all of the allowed torsion angles for each rotatable bond in the molecule and retains the conformer with the lowest energy. Since a systematic search may not be feasible for a molecule with multiple rotatable bonds, a number of stochastic search methods are also available: the random search method, which tries random settings for the torsion angles (from the predefined allowed values), and a weighted rotor search, a stochastic search method that converges on a low energy conformer by weighting particular torsion angles based on the relative energy of the generated conformer. With Open Babel 2.3, conformer search based on a genetic algorithm is also available which allows the application of filters (e.g. a diversity filter) and different scoring functions. This latter method can be used to generate a library of diverse conformers,

or like the other methods to seek a low energy conformer [36].

Implementation

Technical Details

Open Babel is implemented in standards-compliant C++. This ensures support for a wide variety of C++ compilers (MSVC, GCC, Intel Compiler, MinGW, Clang), operating systems (Windows, Mac OS X, Linux, BSD, Windows/Cygwin) and platforms (32-bit, 64-bit). Since version 2.3, it is compiled using the CMake build system [37,38]. This is an open-source cross-platform build system with advanced features for dependency analysis. The build system has an associated unit test framework CTest, which allows nightly builds to be compiled and tested automatically with the results collated and displayed on a centralized dashboard [39].

To simplify installation Open Babel has as few external dependencies as possible. Where such dependencies exist, they are optional. For example, if the XML development libraries are not available, Open Babel will still compile successfully but none of the XML formats (such as Chemical Markup Language, CML) will be available. Similarly, if the Eigen matrix and linear algebra library is not found, any classes that require fast matrix manipulation (such as OBAlign, which performs least squares alignment) will not be compiled.

While the majority of the Open Babel library is written in C++, bindings have been developed for a range of other programming languages, including Java and the .NET platform, as well as the so-called "dynamic" scripting languages Perl, Python, and Ruby. These are automatically generated from the C++ header files using the SWIG tool. As described previously [40], in the case of Python an additional module is provided named Pybel that simplifies access to the C++ bindings. These interfaces facilitate development of web-enabled chemistry applications, as well as rapid development and prototyping.

Code Architecture

The Open Babel codebase has a modular design as shown in Figure 2. The goal of this design is threefold:

1. To separate the chemistry, the conversion process and the user interfaces reducing, as far as possible, the dependency of one upon another.
2. To put all of the code for each chemical format in one place (usually a single file) and make the addition of new formats simple.
3. To allow the format conversion of not just molecules, but also any other chemical objects, such as reactions.

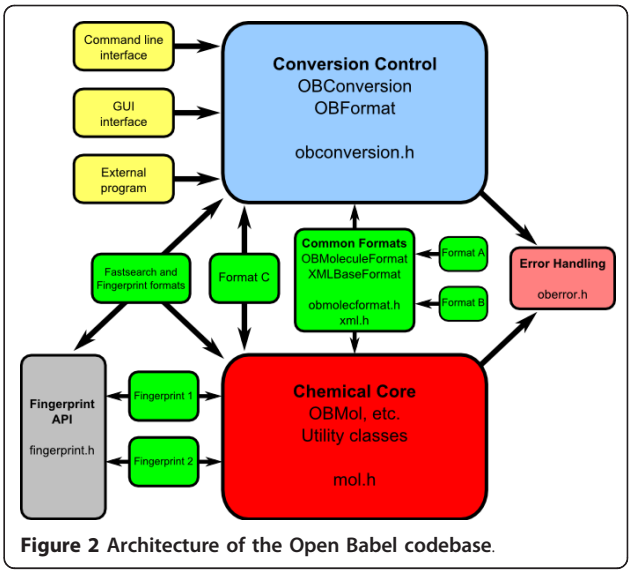

Figure 2 Architecture of the Open Babel codebase.

The code base can be considered as consisting of the following modules (Figure 2):

- The Chemical Core, which contains OBMol etc. and has all of the chemical structure description and manipulation. This is the heart of the application and its API can be used as a chemical toolbox. It has no input/output capabilities.
- The Formats, which read and write to files of different types. These classes are derived from a common base class, OBFormat, which is in the Conversion Control module. They also make use of the chemical routines in the Chemical Core module. Each format file contains a global object of the format class. When the format is loaded the class constructor registers the presence of the class with OBConversion. This means that the formats are plugins - new formats can be added without changing any framework code.
- Common Formats include OBMoleculeFormat and XMLBaseFormat from which most other formats (like Format A and Format B in the diagram) are derived. Independent formats like Format C are also possible.
- The Conversion Control, which also keeps track of the available formats, the conversion options and the input and output streams. It can be compiled without reference to any other parts of the program. In particular, it knows nothing of the Chemical Core: mol.h is not included.
- The User Interface, which may be a command line application, a Graphical User Interface (GUI), or may be part of another program that uses Open Babel's input and output facilities. This depends only

on the Conversion Control module (obconversion.h is included), but not on the Chemical Core or on any of the Formats.

- The Fingerprint API, as well as being usable in external programs, is employed by the fastsearch and fingerprint formats.
- The Fingerprints, which are bit arrays that describe an object and which facilitate fast searching. They are also built as plugins, registering themselves with their base class OBFingerprint which is in the Fingerprint API.
- Other features such as Forcefields, Partial Charge Models and Chemical Descriptors, although not shown in the diagram, are handled similarly to Fingerprints.
- The Error Handling can be used throughout the program to log and display errors and warnings.

Extensible Interface

The utility of software libraries such as Open Babel depends on the ability of the design to be extended over time to support new functionality. To facilitate this, Open Babel implements a *plugin interface* for file formats, fingerprints, charge models, descriptors, "operators" and molecular mechanics force fields. This ensures a clean separation of the implementation of a particular plugin from the core Open Babel library code, and makes it easy for a new plugin (e.g. a new file format) to be contributed; all that is needed is a single C++ file and a trivial change to one of the build files. The operator plugins provide a very general mechanism for operating on a molecule (e.g. energy minimization or 3D coordinate generation) or on a list of molecules (e.g. filtering or sorting) after reading but before writing.

Plugins are dynamically loaded at runtime. This decreases the overall disk and memory footprint of Open Babel, allowing external developers to choose particular functionality needed for their application and ignore other, less relevant features. It also allows the possibility of a third-party distributing plugins separately to the Open Babel distribution to provide additional functionality.

Open-Source License and Open Development

Open Babel is open-source software, which offers end users and third-party developers a range of additional rights not granted by proprietary chemistry software. Open-source software, at its most basic level, grants users the rights to study how their software works, to adapt it for any purpose or otherwise modify it, and to share the software and their modifications with others. In this sense, Open Source functions in similar ways to the processes of open peer review, publication, and citation in science. The rights granted by open source licenses largely coincide with the norms of scientific ethics to enable verifiability, repeatability, and building on previous results and theories.

Beyond these rights, Open Babel (like most other open-source projects) offers open development – that is, all development occurs in public forums and with public code repositories. This results in greater input from the community as any user can easily submit bug reports or feature suggestions, get involved in discussions on the future direction of Open Babel or even become a developer him/herself. In practice, the number of active contributors has increased over time through this level of open, public development (Figure 3). Moreover, it means that the development of the code is completely transparent and the quality of the software is available for public scrutiny. Indeed, since its inception, over 658 bugs have been submitted to the public tracker and fixed [41].

Validation and Testing

Open Babel includes an extensive test suite comprising 60 different test programs each with tens to hundreds of tests. In early 2010, a nightly build infrastructure and dashboard was put in place with support from Kitware, Inc. This has greatly improved code quality by catching regressions, and also ensures that the code compiles cleanly on all platforms and compilers supported by Open Babel. Some examples of tests that are run each night are:

(1) The MMFF94 forcefield code is tested against the MMFF94 validation suite.

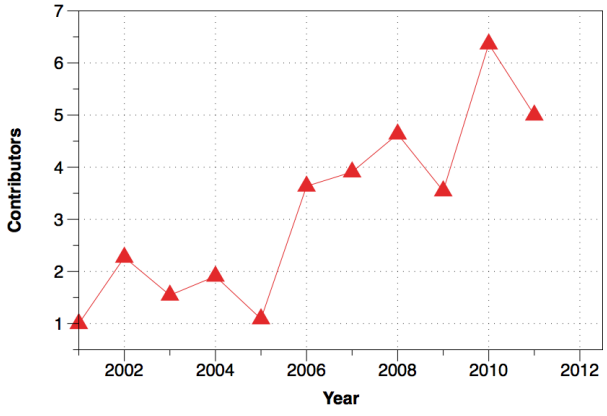

Figure 3 Number of contributors over time. Note that this graph only includes developers who directly commited code to the Open Babel source code repository, and does not include patches provided by users.

(2) The OBAlign class, which was developed using Test-Driven Development (TDD) methodology, is run against its test suite.
(3) Handling of symmetry is validated by converting several test cases between SMILES, 2D and 3D SDF, and InChI (there are also several test programs with unit tests for the individual stereo classes in the API).
(4) The SMARTS parser is tested using over 250 valid and invalid SMARTS patterns, and the SMARTS matcher is tested using 125 basic SMARTS patterns.
(5) The LSSR (Least Set of Smallest Rings) code is tested for invariance against changing the atom order for a series of polycyclic molecules.

Recently the development team has placed a major focus on increasing the robustness of file format translation particularly in relation to the commonly used SMILES and MDL Molfile formats. Translating between these formats requires accurate stereochemistry perception, inference of implicit hydrogens, and kekulization of delocalized systems. While it is difficult to ensure that any complex piece of code is free of bugs, and Open Babel is no exception, validation procedures can be carried out to assess the current level of performance and to find additional test cases that expose bugs. The following procedure was used to guide the rewriting of stereochemistry code in Open Babel, a project that began in early 2009. Starting with a dataset of 18,084 3D structures from PubChem3D as an SDF file, we compared the result of (a) conversion to SMILES, followed by conversion of that to Canonical SMILES to (b) conversion directly to Canonical SMILES. This procedure can be used to flush out errors in reading the original SDF file, reading/writing SMILES (either due to stereochemistry errors or kekulization problems), and is also a test (to some extent) of the canonicalization code. At the time of starting this work (March 2009), the error rate found was 1424 (8%); by Oct 2009, combined work on stereochemistry, kekulization and canonicalization had reduced this to 190 (~1%), and continued improvements have reduced the number of errors down to two (shown in Figure 4) for Open Babel 2.3.1 (~0.01%). The first failure is due to a kekulization error in a polycyclic aromatic molecule incorporating heteroatoms: (a) gave c1ccc2c(c1)c1[nH][nH]c3c4c1c(c2)ccc4cc1c3cccc1 while (b) gave c1ccc2c(c1)c1nnc3c4c1c(c2)ccc4cc1c3cccc1. This error led to confusion over whether or not the aromatic nitrogens have hydrogens attached (they do not). The second failure involves confusion over the canonical stereochemistry at a bridgehead carbon: (a) gave C1CN2[C@@H](C1)CCC2 while (b) gave C1CN2[C@H](C1)CCC2. This is actually a meso compound and so both SMILES strings are correct and represent the same molecule. However the canonicalization algorithm should have chosen one stereochemistry or the other for the canonical representation.

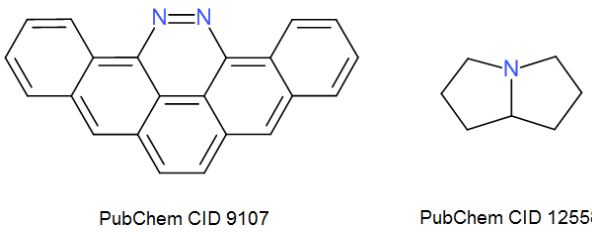

Figure 4 The two failures found in the validation test for reading/writing SMILES.

Another area of focus was the canonicalization algorithm, which can be used to generate canonical SMILES as well as other formats. The algorithm can be tested by ensuring that the same canonical SMILES string is obtained even when the order of atoms in a molecule is changed (while retaining the same connection table). The test stresses all areas of the library, including aromaticity perception, kekulization, stereochemistry, and canonicalization. The development of the canonicalization code in Open Babel was guided by applying this test to the 5,151,179 molecules in the eMolecules catalogue (dated 2011-01-02) with 10 random shuffles of the atom order. At the time of the Open Babel 2.2.3 release, there were 24,404 failures of the canonicalization algorithm; this has now been reduced to only four (shown in Figure 5, < 0.001%). The Open Babel nightly test suite ensures that this test passes for a number of problematic molecules. Although the canonicalization algorithm is still not perfect, we believe that the current level of performance (99.99992% success on the eMolecules catalogue) is acceptable for general use and with time we intend to improve performance further.

Given that the error rate for canonicalization and handling of stereochemistry is now quite low, the next area of focus for the Open Babel development team is to improve the handling of implicit valence for "unusual atoms." This is particularly important for organometallic species and inorganic complexes.

Using Open Babel

Applications

The Open Babel package is composed of a set of user applications as well as a programming library. The main command line application provided is *obabel* (a small upgrade on the earlier *babel*), which facilitates file format conversion, filtering (by SMARTS, title, descriptor value, or property field), 3D or 2D structure generation,

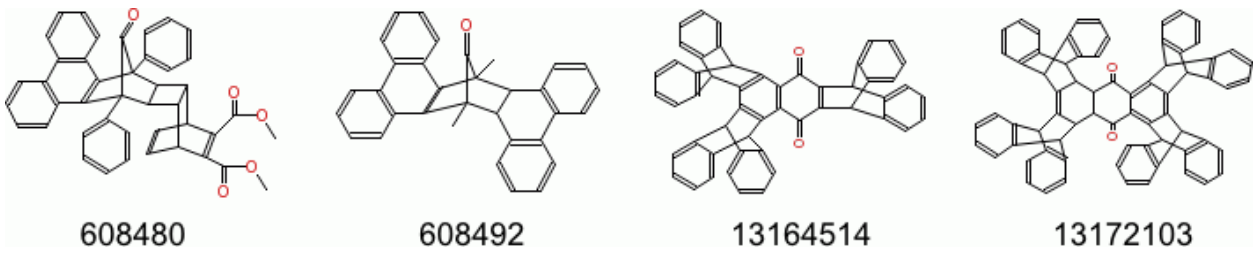

Figure 5 The four failures found in the validation test for canonicalization.

conversion of hydrogens from implicit to explicit (and vice versa), and removal of small fragments or of duplicate structures. A number of features are provided to handle multi-molecule file formats (such as SDF or MOL2) and to use or manipulate the information in property fields and molecule titles. Here is an example of using *obabel* to convert from SDF format to SMILES:

```
obabel inputmols.sdf -O outputmols.smi
```

A more complicated use would be to extract all molecules in an SDF file whose titles start with "active":

```
obabel inputmols.sdf -aT -o copy -O outputmols.sdf –filter "title='active*'"
```

The *copy* format specified by "-o copy" is a utility format that copies the exact contents of the input file (for the filtered molecules) directly to the output, without perception or interpretation. The "-aT" indicates that only the title of the input SDF file should be read; full chemical perception is not required.

The Open Babel graphical user interface (GUI) provides the same functionality. Figure 6 is a screenshot of the GUI carrying out the same filtering operation described in the *obabel* example above. The left panel deals with setting up the input file, the right panel handles the output and the central panel is for setting conversion options. Depending on whether a particular option requires a parameter, the available options are displayed either as check boxes or as text entry boxes. These interface elements are generated dynamically directly from the text description and help text provided by each format plugin.

Programming Library

The Open Babel library allows users to write chemistry applications without worrying about the low-level details of handling chemical information, such as how to read or write a particular file format, or how to use SMARTS for substructure searching. Instead, the user can focus on the scientific problem at hand, or on creating a more easy-to-use interface (e.g. a GUI) to some of Open Babel's functionality. The Open Babel API (Application Programming Interface) is the set of classes, methods and variables provided by Open Babel to the user for use in programs. Documentation on the complete API (generated using Doxygen [42]) is available from the Open Babel website [43], or can be generated from the source code.

The functionality provided by the Open Babel library is relied upon by many users and by several other software projects, with the result that introducing changes to the API would cause existing software to break. For this reason, Open Babel strives to maintain API stability over long periods of time, so that existing software will continue to work despite the release of new Open Babel versions with additional features, file formats and bug fixes. Open Babel uses a version numbering system that indicates how the API has changed with every release:

- Bug fix releases (e.g. 2.0.0 versus 2.0.1) do not change API at all
- Minor version releases (e.g. 2.0 versus 2.1) will add to the API, but will otherwise be backwards-compatible
- Major version releases (e.g. 2 versus 3) are not backwards-compatible, and have changes to the API (including removal of deprecated classes and functions)

Figure 7 shows an example C++ program that uses the two main classes OBConversion and OBMol to print out the molecular weight of all of the molecules in an SDF file. This could be used, for example, to investigate differences in the molecular weight distribution between two databases. The same program is shown in Figure 8 but implemented using the Python bindings.

Examples of Use

Open Babel has already been referenced over 400 times for various uses. The most common use of Open Babel is through the *obabel* command line application (or the corresponding graphical user interface) for the interconversion of chemical file formats. Such conversions may also involve the calculation or inference of additional molecular information or application of a filter. Some

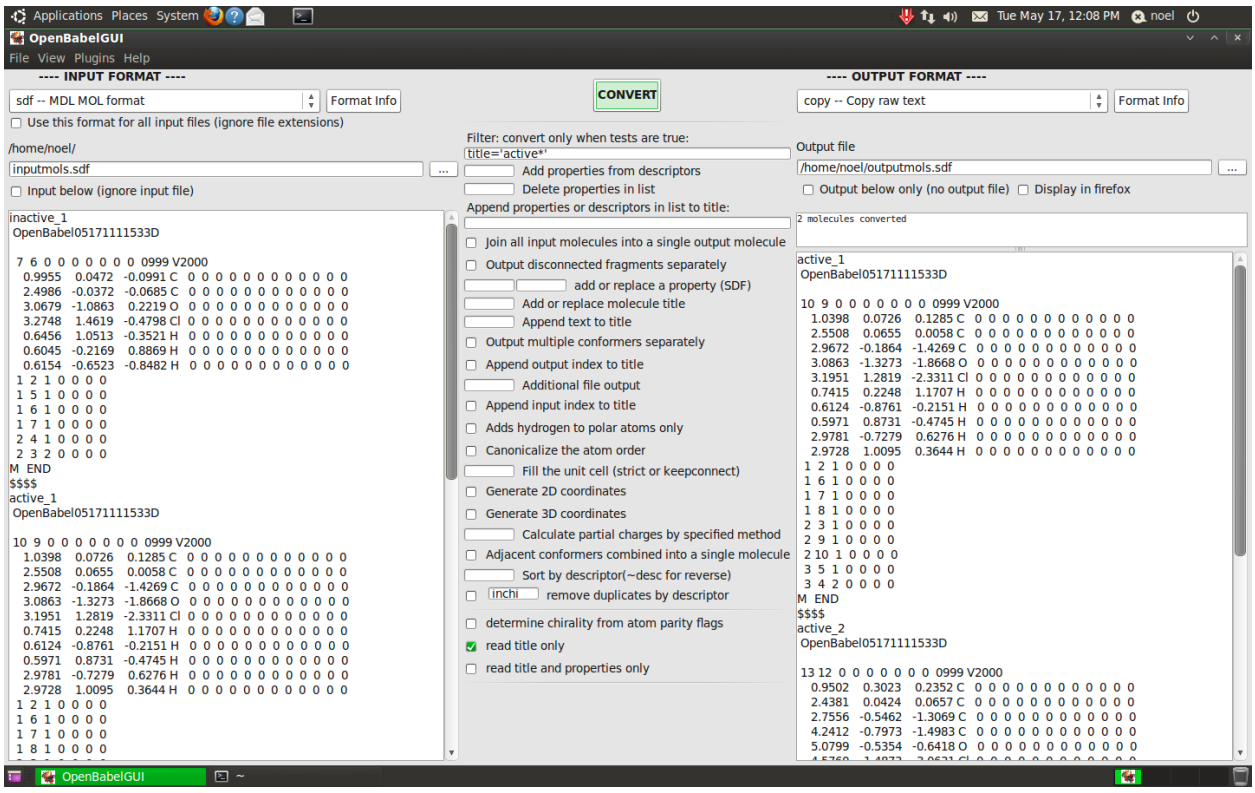

Figure 6 Screenshot of the Open Babel GUI. In the screenshot, the Open Babel GUI is running on Bio-Linux 6.0, an Ubuntu derivative.

published examples of these include the following:

- interconversion of chemical file formats or representations [44-47]
- addition of hydrogens [48-50]
- generation of 3D molecular structures [51-53]
- calculation of partial charges [54,55]
- generation of molecular fingerprints [56-59]
- removal of duplicate molecules from a dataset [60]
- calculation of MOL2 atom types [61]

An interesting example that shows how a particular chemical representation may be used to facilitate a scientific study is the crystallographic study of Fábián and Brock who used Open Babel to generate InChI strings for molecules in the Cambridge Structural Database [62]. Exploiting the fact that InChIs of enantiomers are identical expect at the enantiomer sublayer ("/m0"

```
#include <iostream>

#include <openbabel/obconversion.h>
#include <openbabel/mol.h>

int main(int argc,char **argv)
{
  OBConversion conv;
  conv.SetInFormat("sdf");
  OBMol mol;

  bool notatend = conv.ReadFile(&mol, "dataset.sdf");
  while (notatend)
  {
    std::cout << "Molecular Weight: "
              << mol.GetMolWt() << std::endl;

    mol.Clear();
    notatend = conv.Read(&mol);
  }

  return 0;
}
```

Figure 7 Example C++ program that uses the Open Babel library. The program prints out the molecular weight of each molecule in the SDF file "dataset.sdf".

```
import openbabel as ob

conv = ob.OBConversion()
conv.SetInFormat("sdf")
mol = ob.OBMol()

notatend = conv.ReadFile(mol, "dataset.sdf")
while notatend:
    print "Molecular Weight: %f" % mol.GetMolWt()

    mol.Clear()
    notatend = conv.Read(mol)
```

Figure 8 Example Python program that uses the Open Babel library. The program prints out the molecular weight of each molecule in the SDF file "dataset.sdf".

Table 1 Software applications and libraries that use Open Babel

Name	Description	Reference	Web page
Avogadro	GUI for molecular modelling and computational chemistry	G. Hutchison M. Hanwell	http://avogadro.openmolecules.net/
cclib	Parse computational chemistry output files	[72]	http://cclib.sf.net/
CCP1GUI	GUI for computational chemistry	Jens Thomas	http://www.cse.scitech.ac.uk/ccg/software/ccp1gui
ChemAzTech	Manage a chemical laboratory database	Rémy Dernat	http://chemaztech.sf.net/
ChemSpotlight	Chemistry file indexer for MacOSX	G. Hutchison	http://chemspotlight.openmolecules.net/
ChemT	GUI for generating combinatorial libraries	Rui Abreu	http://www.esa.ipb.pt/~ruiabreu/chemt
ChemTool	2D molecular drawing	[73]	http://ruby.chemie.uni-freiburg.de/~martin/chemtool
CMDF	Library for handling and preparing multi-scale multi-paradigm simulations	[74]	http://web.mit.edu/mbuehler/www/research/CMDF/CMDF.htm
Confab	Systematically generate conformers	[36]	http://confab.googlecode.com/
DockoMatic	Automate the preparation and analysis of AutoDock runs	[75]	http://sf.net/projects/dockomatic/
DOVIS 2.0	Automate the preparation and analysis of AutoDock runs	[76]	http://www.bhsai.org/dovis.html
FAF-Drugs2	ADMET filtering of molecular datasets	[77]	http://www.mti.univ-paris-diderot.fr/fr/downloads.html
FMiner2	Large-scale chemical graph mining based on backbone refinement classes	[78,79]	http://www.maunz.de/wordpress/bbrc
Ghemical	GUI for computational chemistry	Tommi Hassinen	http://www.uku.fi/~thassine/projects/ghemical
Gnome Chemistry Utils	2D chemical editor, 3D viewer, chemical calculator and periodic table for Linux	Jean Bréfort	http://gchemutils.nongnu.org/
iBabel	MacOSX interface to Open Babel and other Open chemistry tools	Chris Swain	http://homepage.mac.com/swain/Sites/Macinchem/page65/ibabel3.html
Kalzium	GUI showing information on the periodic table of the elements	Carsten Niehaus	http://edu.kde.org/kalzium/
Lazar	Lazy Structure-Activity Relationships for toxicity prediction	[80]	http://www.in-silico.de/software/
Molekel	GUI for computational chemistry	Ugo Varetto	http://molekel.cscs.ch/
molsKetch	2D chemical editor	Harm van Eersel	http://molsketch.sf.net/
MyChem	Chemistry extension to the MySQL database	J. Pansanel	http://mychem.sf.net/
NanoEngineer-1	Computer-aided design for the nanoscale	Nanorex, Inc.	http://nanoengineer-1.net/
NanoHive-1	Simulator for the study, experimentation, and development of nanotech entities	Brian Helfrich	http://www.nanohive-1.org/
OpenMD	Open Source molecular dynamics engine	[81]	http://openmd.net/
Open3DQSAR	High-throughput chemometric analysis of molecular interaction fields	[82,83]	http://www.open3dqsar.org/
OSRA	Extracts chemical structures from images	[84]	http://osra.sf.net/
PgChem	Chemistry extension to the PostgreSQL database	Ernst-Georg Schmidt	http://pgfoundry.org/projects/pgchem
Pharao	Pharmacophore discovery and searching	Silicos NV	http://www.silicos.be/
Pharmer	Pharmacophore searching	[85]	http://smoothdock.ccbb.pitt.edu/pharmer
Piramid	Shape-based alignment of molecules	Silicos NV	http://www.silicos.be/
PyADF	Library for handling and preparing quantum mechanical multi-scale simulations	[86]	http://www.ipc.kit.edu/cfn-ysg/158.php
PyRx	GUI for virtual screening with protein-ligand docking	Sargis Dallakyan	http://pyrx.scripps.edu/
QMForge	GUI for analysing results of quantum chemistry calculations	[72]	http://qmforge.sf.net/
RMG	Reaction Mechanism Generator	[87]	http://rmg.sf.net/
Sci3D	Interactive visualization of 3D models of scientific data, such as molecular structures and surfaces	T.J. O'Donnell	http://sci3d.sf.net/
Sieve	Filter molecules from datasets	Silicos NV	http://www.silicos.be/
SMIREP	Generation of fragment-based structure-activity relationships	[88]	http://www.karwath.org/systems/smirep.html
Stripper	Extract molecular scaffolds	Silicos NV	http://www.silicos.be/

Table 1 Software applications and libraries that use Open Babel *(Continued)*

Toxtree	Toxic hazard estimation using decision trees	Ideaconsult Ltd.	http://toxtree.sf.net/
V_Sim	Visualize atomic structures such as crystals and grain boundaries	Damien Caliste	http://inac.cea.fr/L_Sim/V_Sim/index.en.html
WebBabel	Web application for file format conversion	T.J. O'Donnell	http://webbabel.sf.net/
XDrawChem	2D molecular editor	Bryan Herger	http://xdrawchem.sf.net/
XtalOpt	Extension to Avogadro for crystal-structure prediction	[89]	http://xtalopt.openmolecules.net/
YASARA	GUI for molecular graphics, modeling and simulation	Elmar Krieger	http://www.yasara.org/
ZODIAC	GUI for molecular modelling and docking	[90]	http://www.zeden.org/

or "/m1"), they used the InChIs as part of a workflow to identify kryptoracemates (a class of racemic crystals where the enantiomers are not related by space-group symmetry) in the database.

To implement new methods, or access additional molecular information, it is necessary to use the Open Babel library directly either from C++ or using one of the supported language bindings. Some examples of published studies that have done this include the following:

• Dehmer *et al.* implemented molecular complexity measures based on information theory [63].

• Langham and Jain developed a model for chemical mutagenicity based on atom pair features [64].

• Fontaine *et al.* implemented a method, anchor-GRIND, that uses an anchor point of a molecular scaffold to compare molecular interaction fields when different substituents are present [65].

• Konyk *et al.* have developed a plugin for Open Babel that adds support for the Web Ontology Language (OWL) to allow automated reasoning about chemical structures [66].

• Kogej *et al.* (AstraZeneca) implemented a 3-point pharmacophore fingerprint called TRUST [67].

Table 2 Web applications and databases that use Open Babel

Name	Description	Reference	Web page
ChemDB	Database of small molecules	[91]	http://cdb.ics.uci.edu/
Cheméo	Chemical structure and property search engine	Céondo Ltd	http://www.chemeo.com/
ChemMine Tools	Web application for analysing and clustering small molecules	[92]	http://chemmine.ucr.edu/
eMolecules	Chemical vendor search engine	eMolecules.com	http://emolecules.com/
FragmentStore	Database for comparison of fragments found in metabolites, drugs and toxic compounds	[93]	http://bioinf-applied.charite.de/fragment_store/
Frog2	FRee Online druG 3D conformation generation	[94]	http://bioserv.rpbs.univ-paris-diderot.fr/cgi-bin/Frog2
hBar Lab	Web application providing on-demand access to computer-aided chemistry	hBar Solutions ApS	https://www.hbar-lab.com/
IUPHAR-DB	Database of human drug targets and their ligands	[95]	http://www.iuphar-db.org/
OpenCDLig	Web application for sharing resources about cyclodextrin/ligand complexes	[96]	https://kdd.di.unito.it/casmedchem/
PSMDB	Protein - Small-Molecule Database	[97]	http://compbio.cs.toronto.edu/psmdb/
SambVca	Web application for calculation of buried volume of organometallic ligands	[98]	https://www.molnac.unisa.it/OMtools/sambvca.php
ScafBank	Database of molecular scaffolds	[99]	http://202.127.30.184:8080/scafbank.html
SMARTCyp	Web application for prediction of sites of cytochrome P450 mediated metabolism	[100]	http://www.farma.ku.dk/smartcyp/
sMol Explorer	Web application for exploring small-molecule datasets	[101]	http://www3a.biotec.or.th/isl/index.php/smol-explorer
SuperImposé	Web application for structural similarity between ligands, binding sites or proteins	[102]	http://farnsworth.charite.de/superimpose-web/
SuperToxic	Database of toxic compounds	[103]	http://bioinformatics.charite.de/supertoxic/
SuperSite	Detailed information on, and comparisons of, protein-ligand binding sites	[104]	http://bioinf-tomcat.charite.de/supersite/
SuperSweet	Database of natural and artificial sweeteners	[105]	http://bioinf-applied.charite.de/sweet/
STITCH2	Chemical-protein interactions	[106]	http://stitch.embl.de/
VCCLAB	Virtual Computational Chemistry Laboratory	[107]	http://www.vcclab.org/
wwLigCSRre	Web application that performs ligand-based screening using 3D similarity	[108]	http://bioserv.rpbs.univ-paris-diderot.fr/Help/wwLigCSRre.html

• Many other examples exist [68-71].

The vital role that a cheminformatics toolkit plays in the development of scientific resources is shown by Tables 1 and 2. Table 1 lists examples of stand-alone applications or programming libraries that rely on Open Babel, either calling the library directly or via one of the command-line executables. Table 2 contains examples of web applications and databases that either use Open Babel on the server or where Open Babel was used in the preparation of the data.

Conclusions

In November 2011, Open Babel will mark 10 years of existence as an independent project, and for the first time, we have discussed its development and features. As shown by more than 400 citations, it has become an essential tool for handling the myriad of molecular file formats encountered in diverse branches of chemistry. While more work remains to be done, through validation processes such as those described above and the recent introduction of a nightly build and testing framework, we aim to improve the quality and robustness of the toolkit with each new release.

Looking forward to the future, one of the goals of the project is to extend support to molecules that currently are not handled very well by existing cheminformatics toolkits. Typically toolkits focus on the types of molecules of principal importance to the pharmaceutical industry, namely stable organic molecules comprising wholly of 2-center 2-electron covalent bonds. Molecules outside this set - such as radicals, organometallic and inorganic molecules, molecules with coordinate bonds or 3-center 2-electron bonds - are poorly supported in general. Future releases of Open Babel will provide substantially improved handling of such species. We also seek to improve speed and coverage of important methods such as structure generation, kekulization and canonicalization.

Open Babel is freely available from http://openbabel.org, and new community members are very welcome (users, developers, bug reporters, feature requesters). For information on how to use Open Babel, please see the documentation at http://openbabel.org/docs and the API documentation at http://openbabel.org/api.

Availability and Requirements

Project Name: Open Babel

Project home page: http://openbabel.org

Operating system(s): Cross-platform

Programming language: C++, bindings to Python, Perl, Ruby, Java, C#

Other requirements (if compiling): CMake 2.4+

License: GNU GPL v2

Any restrictions to use by non-academics: None

Acknowledgements and Funding

We would like to thank all users and contributors to the Open Babel project over its history, including OpenEye Scientific Software Inc. for their initial OELib code. We also thank the Blue Obelisk Movement for ideas, comments on this manuscript, and support. We thank SourceForge for providing resources for issue tracking and managing releases, and Kitware for additional dashboard resources. NMOB is supported by a Health Research Board Career Development Fellowship (PD/2009/13).

Author details

[1]Analytical and Biological Chemistry Research Facility, Cavanagh Pharmacy Building, University College Cork, Co. Cork, Ireland. [2]Department of Chemistry, Technische Universität München, Garching D-85747, Germany. [3]eMolecules, Inc., 420 Stevens Ave #120, Solana Beach, CA 92075, USA. [4]Open Babel development team. [5]University of Pittsburgh, Department of Chemistry, 219 Parkman Avenue, Pittsburgh, PA 15217, USA.

Authors' contributions

GRH is the lead developer of the Open Babel project. CAJ, CM, MB, NMOB, and TV are developers of Open Babel. All authors read and approved the final manuscript.

Competing interests

The authors declare that they have no competing interests.

Received: 27 June 2011 Accepted: 7 October 2011
Published: 7 October 2011

References

1. Weininger D: **SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules.** *J Chem Inf Comput Sci* 1988, **28**:31-36.
2. Murray-Rust P, Rzepa H: **Chemical markup, XML, and the Worldwide Web. 1. Basic principles.** *J Chem Inf Comput Sci* 1999, **39**:928-942.
3. Murray-Rust P, Rzepa HS: **Chemical Markup, XML and the World-Wide Web. 2. Information Objects and the CMLDOM.** *J Chem Inf Model* 2001, **41**:1113-1123.
4. Murray-Rust P, Rzepa H, Wright M: **Development of chemical markup language (CML) as a system for handling complex chemical content.** *New J Chem* 2001, **25**:618-634.
5. Murray-Rust P, Rzepa H: **Chemical Markup, XML, and the World Wide Web. 4. CML Schema.** *J Chem Inf Comput Sci* 2003, **43**:757-772.
6. Holliday GL, Murray-Rust P, Rzepa HS: **Chemical Markup, XML, and the World Wide Web. 6. CMLReact, an XML Vocabulary for Chemical Reactions.** *J Chem Inf Model* 2006, **46**:145-157.
7. Daylight Theory: :, SMARTS http://www.daylight.com/dayhtml/doc/theory/theory.smarts.html.
8. Fogel K: *Producing Open Source Software: How to Run a Successful Free Software Project* O'Reilly Media, Inc. Sebastopol, CA; 2005.
9. Citations were generated by Google Scholar:[http://scholar.google.com/scholar?as_q=openbabel&num=10&as_occt=any&as_publication=&as_ylo=2001].
10. A selection of such projects is included below. :, The full list is available at: http://openbabel.org/wiki/Related_Projects.
11. Open Babel: :[http://openbabel.org/].
12. Open Babel Report Format: :[http://openbabel.org/docs/2.3.0/FileFormats/Open_Babel_report_format.html].
13. Open Babel Fingerprint Format: :[http://openbabel.org/docs/2.3.0/FileFormats/Fingerprint_format.html].
14. Open Babel Fastsearch Format: :[http://openbabel.org/docs/2.3.0/FileFormats/Fastsearch_format.html].
15. MolPrint2D Format: :[http://openbabel.org/docs/2.3.0/FileFormats/MolPrint2D_format.html].
16. Bender A, Mussa HY, Glen RC, Reiling S: **Molecular Similarity Searching Using Atom Environments, Information-Based Feature Selection, and a Naïve Bayesian Classifier.** *J Chem Inf Model* 2004, **44**:170-178.
17. MNA Format: :[http://openbabel.org/docs/2.3.0/FileFormats/Multilevel_Neighborhoods_of_Atoms_(MNA).html].

18. Filimonov D, Poroikov V, Borodina Y, Gloriozova T: **Chemical Similarity Assessment through Multilevel Neighborhoods of Atoms: Definition and Comparison with the Other Descriptors.** *J Chem Inf Model* 1999, **39**:666-670.
19. PDB Format v3.2: :[http://www.wwpdb.org/documentation/format32/v3.2.html].
20. PDB: Cruft to Content: :[http://www.daylight.com/meetings/mug01/Sayle/m4xbondage.html].
21. Morgan HL: **The Generation of a Unique Machine Description for Chemical Structures-A Technique Developed at Chemical Abstracts Service.** *J Chem Docum* 1965, **5**:107-113.
22. Nauty: :[http://cs.anu.edu.au/~bdm/nauty/].
23. McKay BD: **Practical graph isomorphism.** *Congressus Numerantium* 1981, **30**:45-87.
24. Gakh A, Burnett M: **Modular Chemical Descriptor Language (MCDL): Composition, connectivity, and supplementary modules.** *J Chem Inf Comput Sci* 2001, **41**:1494-1499.
25. Trepalin SV, Yarkov AV, Pletnev IV, Gakh AA: **A Java Chemical Structure Editor Supporting the Modular Chemical Descriptor Language (MCDL).** *Molecules* 2006, **11**:219-231.
26. Gakh AA, Burnett MN, Trepalin SV, Yarkov AV: **Modular Chemical Descriptor Language (MCDL): Stereochemical modules.** *J Cheminf* 2011, **3**:5.
27. Halgren T: **Merck molecular force field .1. Basis, form, scope, parameterization, and performance of MMFF94.** *J Comput Chem* 1996, **17**:490-519.
28. Halgren T: **Merck molecular force field .2. MMFF94 van der Waals and electrostatic parameters for intermolecular interactions.** *J Comput Chem* 1996, **17**:520-552.
29. Halgren T: **Merck molecular force field .3. Molecular geometries and vibrational frequencies for MMFF94.** *J Comput Chem* 1996, **17**:553-586.
30. Halgren T, Nachbar R: **Merck molecular force field .4. Conformational energies and geometries for MMFF94.** *J Comput Chem* 1996, **17**:587-615.
31. Halgren T: **Merck molecular force field .5. Extension of MMFF94 using experimental data, additional computational data, and empirical rules.** *J Comput Chem* 1996, **17**:616-641.
32. Andronico A, Randall A, Benz RW, Baldi P: **Data-driven high-throughput prediction of the 3-D structure of small molecules: review and progress.** *J Chem Inf Model* 2011, **51**:760-776.
33. Rappe A, Casewit C, Colwell K, Goddard W III, Skiff WM: **UFF, a full periodic table force field for molecular mechanics and molecular dynamics simulations.** *J Am Chem Soc* 1992, **114**:10024-10035.
34. Wang J, Wolf RM, Caldwell JW, Kollman PA, Case DA: **Development and testing of a general amber force field.** *J Comput Chem* 2004, **25**:1157-1174.
35. Wang J, Wang W, Kollman PA, Case DA: **Automatic atom type and bond type perception in molecular mechanical calculations.** *J Molec Graph Model* 2006, **25**:247-260.
36. O'Boyle NM, Vandermeersch T, Flynn CJ, Maguire AR, Hutchison GR: **Confab - Systematic generation of diverse low-energy conformers.** *J Cheminf* 2011, **3**:8.
37. CMake: :[http://www.cmake.org/].
38. Martin K, Hoffman B: **Mastering CMake: A Cross-Platform Build System.** Kitware, Inc., Clifton Park, NY;, 5 2010.
39. CDash Dashboard for Open Babel: :[http://my.cdash.org/index.php?project=Open+Babel].
40. O'Boyle N, Morley C, Hutchison GR: **Pybel: a Python wrapper for the OpenBabel cheminformatics toolkit.** *Chem Cent J* 2008, **2**:5.
41. Open Babel Bug Tracker: :[https://sourceforge.net/tracker/?limit=25&func=&group_id=40728&atid=428740&status=2].
42. Doxygen: :[http://www.doxygen.org/].
43. Open Babel API: :[http://openbabel.org/api].
44. Myers J, Allison T, Bittner S, Didier B, Frenklach M, Green W, Ho Y, Hewson J, Koegler W, Lansing C, *et al*: **A collaborative informatics infrastructure for multi-scale science.** *Cluster Computing* 2005, **8**:243-253.
45. Lind P, Alm M: **A Database-Centric Virtual Chemistry System.** *J Chem Inf Model* 2006, **46**:1034-1039.
46. Amini A, Shrimpton PJ, Muggleton SH, Sternberg MJE: **A general approach for developing system-specific functions to score protein-ligand docked complexes using support vector inductive logic programming.** *Proteins: Struct, Funct, Bioinf* 2007, **69**:823-831.
47. Arbor S, Marshall GR: **A virtual library of constrained cyclic tetrapeptides that mimics all four side-chain orientations for over half the reverse turns in the protein data bank.** *J Comput-Aided Mol Des* 2008, **23**:87-95.
48. Huang Z, Wong CF: **A Mining Minima Approach to Exploring the Docking Pathways of p-Nitrocatechol Sulfate to YopH.** *Biophys J* 2007, **93**:4141-4150.
49. Hill AD, Reilly PJ: **A Gibbs free energy correlation for automated docking of carbohydrates.** *J Comput Chem* 2008, **29**:1131-1141.
50. Armen RS, Chen J, Brooks CL III: **An Evaluation of Explicit Receptor Flexibility in Molecular Docking Using Molecular Dynamics and Torsion Angle Molecular Dynamics.** *J Chem Theory Comp* 2009, **5**:2909-2923.
51. Liu L, Ma H, Yang N, Tang Y, Guo J, Tao W, Jaa Duan: **A Series of Natural Flavonoids as Thrombin Inhibitors: Structure-activity relationships.** *Thromb Res* 2010, **126**:e365-e378.
52. Wallach I, Jaitly N, Lilien R: **A Structure-Based Approach for Mapping Adverse Drug Reactions to the Perturbation of Underlying Biological Pathways.** *PLoS One* 2010, **5**:e12063.
53. Paila YD, Tiwari S, Sengupta D, Chattopadhyay A: **Molecular modeling of the human serotonin1A receptor: role of membrane cholesterol in ligand binding of the receptor.** *Molecular BioSystems* 2011, **7**:224-234.
54. Melville JL, Hirst JD: **TMACC: Interpretable Correlation Descriptors for Quantitative Structure–Activity Relationships.** *J Chem Inf Model* 2007, **47**:626-634.
55. Pencheva T, Lagorce D, Pajeva I, Villoutreix BO, Miteva MA: **AMMOS: Automated Molecular Mechanics Optimization tool for in silico Screening.** *BMC Bioinformatics* 2008, **9**:438.
56. Schietgat L, Ramon J, Bruynooghe M: **An Efficiently Computable Graph-Based Metric for the Classification of Small Molecules.** *Proceedings of the 11th International Conference on Discovery Science* Springer-Verlag Berlin, Heidelberg; 2008, 197-209.
57. Krier M, Hutter MC: **Bioisosteric Similarity of Molecules Based on Structural Alignment and Observed Chemical Replacements in Drugs.** *J Chem Inf Model* 2009, **49**:1280-1297.
58. Wang X, Huan J, Smalter A, Lushington GH: **Application of kernel functions for accurate similarity search in large chemical databases.** *BMC Bioinformatics* 2010, **11**:S8.
59. Cheng T, Li Q, Wang Y, Bryant SH: **Binary Classification of Aqueous Solubility Using Support Vector Machines with Reduction and Recombination Feature Selection.** *J Chem Inf Model* 2011, **51**:229-236.
60. Mihaleva VV, Verhoeven HA, de Vos RCH, Hall RD, van Ham RCHJ: **Automated procedure for candidate compound selection in GC-MS metabolomics based on prediction of Kovats retention index.** *Bioinformatics* 2009, **25**:787-794.
61. Bas DC, Rogers DM, Jensen JH: **Very fast prediction and rationalization of pKa values for protein-ligand complexes.** *Proteins: Struct, Funct, Bioinf* 2008, **73**:765-783.
62. Fabian L, Brock CP: **A list of organic kryptoracemates.** *Acta Cryst* 2010, **B66**:94-103.
63. Dehmer M, Barbarini N, Varmuza K, Graber A: **A Large Scale Analysis of Information-Theoretic Network Complexity Measures Using Chemical Structures.** *PLoS One* 2009, **4**:e8057.
64. Langham JJ, Jain AN: **Accurate and Interpretable Computational Modeling of Chemical Mutagenicity.** *J Chem Inf Model* 2008, **48**:1833-1839.
65. Fontaine F, Pastor M, Zamora I: **Anchor-GRIND: Filling the gap between standard 3D QSAR and the GRid-INdependent Descriptors.** *J Med Chem* 2005, **48(7)**:2687-94.
66. Konyk M, De Leon A, Dumontier M: **Chemical knowledge for the semantic web.** *Data Integration in the Life Sciences* Springer-Verlag Berlin, Heidelberg; 2008, 169-176.
67. Kogej T, Engkvist O, Blomberg N, Muresan S: **Multifingerprint Based Similarity Searches for Targeted Class Compound Selection.** *J Chem Inf Model* 2006, **46**:1201-1213.
68. Reynès C, Host H, Camproux A-C, Laconde G, Leroux F, Mazars A, Deprez B, Fahraeus R, Villoutreix BO, Sperandio O: **Designing Focused Chemical Libraries Enriched in Protein-Protein Interaction Inhibitors using Machine-Learning Methods.** *PLoS Computational Biology* 2010, **6**:e1000695.
69. Lagorce D, Pencheva T, Villoutreix BO, Miteva MA: **DG-AMMOS: A New tool to generate 3D conformation of small molecules using Distance Geometry and Automated Molecular Mechanics Optimization for in silico Screening.** *BMC Chemical Biology* 2009, **9**:6.

70. Gómez MJ, Pazos F, Guijarro FJ, de Lorenzo V, Valencia A: **The environmental fate of organic pollutants through the global microbial metabolism.** *Molecular Systems Biology* 2007, **3**:114.
71. Kazius J, Nijssen S, Kok J, Bäck T, IJzerman AP: **Substructure Mining Using Elaborate Chemical Representation.** *J Chem Inf Model* 2006, **46**:597-605.
72. O'Boyle NM, Tenderholt AL, Langner KM: **cclib: A library for package-independent computational chemistry algorithms.** *J Comput Chem* 2008, **29**:839-845.
73. Brüstle M: **Chemtool - Moleküle zeichnen mit dem Pinguin.** *Nachrichten aus der Chemie* 2001, **49**:1310-1313.
74. Buehler M, Dodson J, van Duin A: **The Computational Materials Design Facility (CMDF): A powerful framework for multi-paradigm multi-scale simulations.** *Materials Research Society symposium proceedings* 2006, **894**: LL3.8.
75. Bullock CW, Jacob RB, McDougal OM, Hampikian G, Andersen T: **Dockomatic - automated ligand creation and docking.** *BMC Research Notes* 2010, **3**:289.
76. Jiang X, Kumar K, Hu X, Wallqvist A, Reifman J: **DOVIS 2.0: an efficient and easy to use parallel virtual screening tool based on AutoDock 4.0.** *Chem Cent J* 2008, **2**:18.
77. Lagorce D, Sperandio O, Galons H, Miteva MA, Villoutreix BO: **FAF-Drugs2: Free ADME/tox filtering tool to assist drug discovery and chemical biology projects.** *BMC Bioinformatics* 2008, **9**:396.
78. Maunz A, Helma C, Kramer S: **Efficient mining for structurally diverse subgraph patterns in large molecular databases.** *Machine Learning* 2010, **83**:193-218.
79. Maunz A, Helma C, Kramer S: **Large-scale graph mining using backbone refinement classes.** *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2009)* ACM Paris; 2009, 617-626.
80. Helma C: **Lazy structure-activity relationships (lazar) for the prediction of rodent carcinogenicity and Salmonella mutagenicity.** *Mol Diversity* 2006, **10**:147-158.
81. Meineke MA, Vardeman CF, Lin T, Fennell CJ, Gezelter JD: **OOPSE: an object-oriented parallel simulation engine for molecular dynamics.** *J Comput Chem* 2005, **26**:252-271.
82. Tosco P, Balle T: **Brute-force pharmacophore assessment and scoring with Open3DQSAR.** *J Cheminf* 2011, **3(Suppl 1)**:P39.
83. Tosco P, Balle T: **Open3DQSAR: a new open-source software aimed at high-throughput chemometric analysis of molecular interaction fields.** *J Mol Model* 2011, **17**:201-208.
84. Filippov IV, Nicklaus MC: **Optical Structure Recognition Software To Recover Chemical Information: OSRA, An Open Source Solution.** *J Chem Inf Model* 2009, **49**:740-743.
85. Koes DR, Camacho CJ: **Pharmer: Efficient and Exact Pharmacophore Search.** *J Chem Inf Model* 2011, **51(6)**:1307-14.
86. Jacob CR, Beyhan SM, Bulo RE, Gomes ASP, Götz AW, Kiewisch K, Sikkema J, Visscher L: **PyADF - A scripting framework for multiscale quantum chemistry.** *J Comput Chem* 2011, **32**:2328-2338.
87. Green HWilliam, Allen WJoshua, Ashcraft WRobert, Beran JGregory, Class ACaleb, Gao Connie, Franklin Goldsmith C, Harper RMichael, Jalan Amrit, Magoon RGregory, Matheu MDavid, Merchant SShamel, Mo DJeffrey, Petway Sarah, Raman Sumathy, Sharma Sandeep, Song Jing, Van Geem MKevin, Wen John, West HRichard, Wong Andrew, Wong Hsi-Wu, Yelvington EPaul, Yu Joanna: **RMG - Reaction Mechanism Generator v3.3.** 2011 [http://rmg.sourceforge.net/].
88. Karwath A, De Raedt L: **SMIREP: Predicting Chemical Activity from SMILES.** *J Chem Inf Model* 2006, **46**:2432-2444.
89. Lonie DC, Zurek E: **XTALOPT: An open-source evolutionary algorithm for crystal structure prediction.** *Comput Phys Commun* 2011, **182**:372-387.
90. Zonta N, Grimstead IJ, Avis NJ, Brancale A: **Accessible haptic technology for drug design applications.** *J Mol Model* 2008, **15**:193-196.
91. Chen JH, Linstead E, Swamidass SJ, Wang D, Baldi P: **ChemDB update full-text search and virtual chemical space.** *Bioinformatics* 2007, **23**:2348-2351.
92. Backman TWH, Cao Y, Girke T: **ChemMine tools: an online service for analyzing and clustering small molecules.** *Nucleic Acids Res* 2011, **39(Web Server issue)**:W486-91.
93. Ahmed J, Worth CL, Thaben P, Matzig C, Blasse C, Dunkel M, Preissner R: **FragmentStore–a comprehensive database of fragments linking metabolites, toxic molecules and drugs.** *Nucleic Acids Res* 2010, **39**: D1049-D1054.
94. Miteva MA, Guyon F, Tuffery P: **Frog2: Efficient 3D conformation ensemble generator for small compounds.** *Nucleic Acids Res* 2010, **38**: W622-W627.
95. Sharman JL, Mpamhanga CP, Spedding M, Germain P, Staels B, Dacquet C, Laudet V, Harmar AJ, NC-IUPHAR: **IUPHAR-DB: new receptors and tools for easy searching and visualization of pharmacological data.** *Nucleic Acids Res* 2010, **39**:D534-D538.
96. Esposito R, Ermondi G, Caron G: **OpenCDLig: a free web application for sharing resources about cyclodextrin/ligand complexes.** *J Comput-Aided Mol Des* 2009, **23**:669-675.
97. Wallach I, Lilien R: **The protein-small-molecule database, a non-redundant structural resource for the analysis of protein-ligand binding.** *Bioinformatics* 2009, **25**:615-620.
98. Poater A, Cosenza B, Correa A, Giudice S, Ragone F, Scarano V, Cavallo L: **Samb Vca: A Web Application for the Calculation of the Buried Volume of N-Heterocyclic Carbene Ligands.** *Eur J Inorg Chem* 2009, **2009**:1759-1766.
99. Yan B-b, Xue M-z, Xiong B, Liu K, Hu D-y, Shen J-k: **ScafBank: a public comprehensive Scaffold database to support molecular hopping.** *Acta Pharmacologica Sinica* 2009, **30**:251-258.
100. Rydberg P, Gloriam DE, Olsen L: **The SMARTCyp cytochrome P450 metabolism prediction server.** *Bioinformatics* 2010, **26**:2988-2989.
101. Ingsriswang S, Pacharawongsakda E: **sMOL Explorer: an open source, web-enabled database and exploration tool for Small MOLecules datasets.** *Bioinformatics* 2007, **23**:2498-2500.
102. Bauer RA, Bourne PE, Formella A, Frommel C, Gille C, Goede A, Guerler A, Hoppe A, Knapp EW, Poschel T, *et al*: **Superimpose: a 3D structural superposition server.** *Nucleic Acids Res* 2008, **36**:W47-W54.
103. Schmidt U, Struck S, Gruening B, Hossbach J, Jaeger IS, Parol R, Lindequist U, Teuscher E, Preissner R: **SuperToxic: a comprehensive database of toxic compounds.** *Nucleic Acids Res* 2009, **37**:D295-D299.
104. Bauer RA, Gunther S, Jansen D, Heeger C, Thaben PF, Preissner R: **SuperSite: dictionary of metabolite and drug binding sites in proteins.** *Nucleic Acids Res* 2009, **37**:D195-D200.
105. Ahmed J, Preissner S, Dunkel M, Worth CL, Eckert A, Preissner R: **SuperSweet–a resource on natural and artificial sweetening agents.** *Nucleic Acids Res* 2010, **39**:D377-D382.
106. Kuhn M, Szklarczyk D, Franceschini A, Campillos M, von Mering C, Jensen LJ, Beyer A, Bork P: **STITCH 2: an interaction network database for small molecules and proteins.** *Nucleic Acids Res* 2009, **38**:D552-D556.
107. Tetko IV, Gasteiger J, Todeschini R, Mauri A, Livingstone D, Ertl P, Palyulin VA, Radchenko EV, Zefirov NS, Makarenko AS, *et al*: **Virtual Computational Chemistry Laboratory - Design and Description.** *J Comput-Aided Mol Des* 2005, **19**:453-463.
108. Sperandio O, Petitjean M, Tuffery P: **wwLigCSRre: a 3D ligand-based server for hit identification and optimization.** *Nucleic Acids Res* 2009, **37**: W504-W509.

doi:10.1186/1758-2946-3-33
Cite this article as: O'Boyle *et al.*: Open Babel: An open chemical toolbox. *Journal of Cheminformatics* 2011 **3**:33.

Part II

Enzyme reaction mechanisms

BIOINFORMATICS APPLICATIONS NOTE Vol. 21 no. 23 2005, pages 4315–4316
doi:10.1093/bioinformatics/bti693

Databases and ontologies

MACiE: a database of enzyme reaction mechanisms

Gemma L. Holliday[1], Gail J. Bartlett[2,†], Daniel E. Almonacid[1], Noel M. O'Boyle[1], Peter Murray-Rust[1], Janet M. Thornton[2] and John B. O. Mitchell[1,*]

[1]Unilever Centre for Molecular Science Informatics, Department of Chemistry, University of Cambridge, Lensfield Road, Cambridge, CB2 1EW, UK and [2]EMBL-EBI, Wellcome Trust Genome Campus, Hinxton, Cambridge, CB10 1SD, UK

Received on July 21, 2005; revised on September 22, 2005; accepted on September 23, 2005
Advance Access publication September 27, 2005

ABSTRACT

Summary: MACiE (mechanism, annotation and classification in enzymes) is a publicly available web-based database, held in CMLReact (an XML application), that aims to help our understanding of the evolution of enzyme catalytic mechanisms and also to create a classification system which reflects the actual chemical mechanism (catalytic steps) of an enzyme reaction, not only the overall reaction.
Availability: http://www-mitchell.ch.cam.ac.uk/macie/
Contact: jbom1@cam.ac.uk

A great deal of knowledge about enzymes, including structures, gene sequences, mechanisms, metabolic pathways and kinetic data, now exists. However, it is spread between many different databases and throughout the literature. Here we announce the completion of the initial version of MACiE, a unique database of the chemical mechanisms of enzymatic reactions.

Web resources such as BRENDA (Schomburg *et al*., 2004), KEGG (Kanehisa *et al*., 2004) and the International Union of Biochemistry and Molecular Biology (IUBMB) Enzyme Nomenclature website (IUBMB, 2005, http://www.chem.qmul.ac.uk/iubmb/enzyme/) contain descriptions of the overall reactions performed by enzymes, accompanied in some cases by a textual or graphical description of the mechansim. MACiE is unique in combining detailed stepwise mechanistic information (including 2D animations), a wide coverage of both chemical space and the protein structure universe, and the chemical intelligence of CMLReact (Holliday,C.L., Murray-Rust,P., and Rzepa,H.S., 2005, manuscript submitted to *J. Chem. Inf. Modeling*). MACiE usefully complements both the mechanistic detail of the Structure–Function Linkage Database (SFLD) for a small number of enzyme superfamilies (Pegg *et al*., 2005) and the wider coverage with less chemical detail provided by EzCatDB (Nagano, 2005) which also contains a limited number of 3D animations.

*To whom correspondence should be addressed.
†Present Address: Bioinformatics Support Service (Biochemistry Building), Centre for Bioinformatics, Division of Molecular Biosciences, Faculty of Life Sciences, Imperial College London, London, SW7 2AZ, UK

DESIGN

The MACiE dataset evolved from that published in the Catalytic Site Atlas (CSA) (Bartlett *et al*., 2002; Porter *et al*., 2004), and each entry is selected so that it fulfils the following criteria:

(1) There is a 3D crystal structure of the enzyme deposited in the Protein Databank (PDB) (Berman *et al*., 2000).
(2) There is a relatively well-understood mechanism available. Taken from the literature, these cover a variety of methodologies, including chemical and biochemical studies, quantum mechanical calculations and structual biology reports.
(3) The enzyme is unique at the H level of the CATH classification—a hierarchical classification system of protein domain structures (Orengo *et al*., 1997)—unless there is a homologue with a significantly different chemical mechanism.
(4) Where there are a number of possible PDB codes available the entry should be, if possible, a wild-type enzyme.

All MACiE enzymes are also contained in the Enzyme Commission (EC) classification system (IUBMB, 2005, http://www.chem.qmul.ac.uk/iubmb/enzyme/), that is, they all have four number codes describing their overall reaction. The first level (Class) describes the basic reaction type. The second and third levels (subclass and sub-subclass, respectively) describe the reaction in further detail and the final level (serial number) describes substrate specificity. For example, the β-lactamases (Fig. 1) are assigned the EC number 3.5.2.6, i.e. a hydrolase (3) acting on a C–N bond (5) in a cyclic amide (2) with a β-lactam as the substrate (6).

In MACiE, the data centre on the catalytic steps involved in the chemical mechanism as well as the overall reaction. Each entry includes the following steps:

- Enzyme name and EC number
- PDB code and CATH codes of all domains in the enzyme
- Diagram and annotation of the overall reaction
- Primary literature references

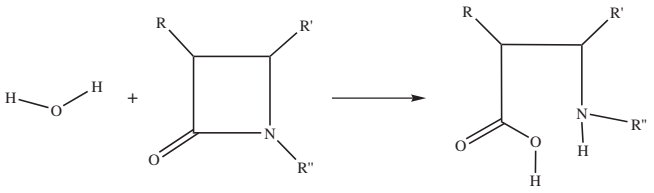

Fig. 1. The overall reaction for a β-lactamase.

- Diagram and annotation of all reaction steps, including:
  —The Ingold mechanism (Ingold, 1969)
  —Diagram and function of catalytic amino acid residues
  —Information on the reactive centres and bond changes
- Comments on the reaction (where applicable).

CONTENT

The criteria defined in the Design section initially produced a dataset of 100 entries. A single EC number may cover a plurality of MACiE entries when different mechanisms bring about the same overall chemical transformation, as with the two types of 3-dehydroquinate dehydratase, and thus 100 MACiE entries span only 96 EC numbers.

The 100 enzymes in Version 1 of MACiE incorporate domains from 140 CATH homologous superfamilies. MACiE currently covers 56 of the 174 EC sub-subclasses present in the PDB, thus, we feel that we have a representative coverage of EC reaction space (comparative EC wheels are available at URL http://www-mitchell.ch.cam.ac.uk/macie/ECCoverage/). We anticipate that all 158 sub-subclasses for which both structures and reliable mechanisms are available will be represented in the forthcoming MACiE Version 2.

SOFTWARE

The data are initially entered in MDL's ISIS/Base, a database package for chemical reactions, validated by at least two people, and then converted into CMLReact using the Jumbo Toolkit (Wakelin *et al*., 2005) to create an information and semantically rich database. At this stage we add extra fields of information to the CMLReact version of MACiE that are unavailable in the ISIS version, including the CATH code. Jumbo is a set of Java-based software which converts the MDL file format produced from ISIS/Base into CMLReact. The MacieConverter section of Jumbo performs the following functions:

- Integration of the files in the ISIS/Base version of MACiE
- Identification of reactant, product and spectator molecules
- Splitting of groups of molecules
- Automatic mapping of atoms within the reaction
- Checking for mass and charge conservation throughout the reaction (stoichiometry)
- Integration and checking of MACiE Dictionary entries.

Once the conversion process has been completed, a further tool in the Jumbo Toolkit, called CMLSnap (Holliday *et al*., 2004), can be used to create an animation of the reaction. This animation includes all of the atoms and bonds involved as well as the electron movements, which are calculated automatically. It is expected that CML will become our primary method of data entry and storage.

CURATION

The annotation process involves input and validation steps. Terms have been rigorously defined either from the IUPAC Gold Book (McNaught *et al*., 1997), such as chemical terms like hydrolysis, or from primary literature, such as mechanism, which is defined using Ingold's terminology (Ingold, 1969), originally put forward in the 1930s. All of the technical and scientific terms used in MACiE are contained in the MACiE dictionary, which is available at the URL http://www-mitchell.ch.cam.ac.uk/macie/glossary.html and is also available as a raw XML file.

The entries online are accessed via an HTML look-up table and include all of the information available in the database. The original ISIS/Base format file and the raw CML files can be supplied.

FUTURE WORK

Future work includes expanding the dataset to include a representative set of EC numbers (at the sub-subclass level), creating a search interface for MACiE and developing authoring tools for MACiE in CML. Ongoing research focuses on the evolution of enzyme catalysis and the classification of enzyme reaction mechanisms.

ACKNOWLEDGEMENTS

G.J.B. would like to thank Dr Jonathan Goodman for his invaluable help with organic chemistry queries. We would also like to thank the EPSRC (G.L.H. and J.B.O.M.), the BBSRC (G.J.B. and J.M.T.—CASE studentship in association with Roche Products Ltd; N.M.O.B. and J.B.O.M.—grant BB/C51320X/1), the Chilean Government's Ministerio de Planificación y Cooperación and Cambridge Overseas Trust (D.E.A.) for funding and Unilever for supporting the Centre for Molecular Science Informatics.

Conflict of Interest: none declared.

REFERENCES

Bartlett,G.J. *et al*. (2002) Analysis of catalytic residues in enzyme active sites. *J. Mol. Biol*., **324**, 105–121.

Berman,H.M. *et al*. (2000) The Protein Data Bank. *Nucleic Acids Res*., **28**, 235–242.

Holliday,G.L. *et al*. (2004) CMLSnap: animated reaction mechanisms. *Internet J. Chem*., **7**, Article 4.

Ingold,C.K. (1969) *Structure and Mechanism in Organic Chemistry*. 2nd edn, Cornell University Press, Ithaca, NY, Chapters 5–15.

Kanehisa,M. *et al*. (2004) The KEGG resource for deciphering the genome. *Nucleic Acids Res*., **32**, D277–D280.

McNaught,A.D. and Wilkinson,A. (1997) International Union of Pure and Applied Chemistry Compendium of Chemical Terminology (''The Gold Book''). 2nd edn, ISBN 0-8-654-26848.

Nagano,N. (2005) EzCatDB: the Enzyme Catalytic-mechanism Database. *Nucleic Acids Res*., **33**, D407–D412.

Orengo,C.A. *et al*. (1997) CATH—a hierarchic classification of protein domain structures. *Structure*, **5**, 1093–1108.

Pegg,S.C-H. *et al*. (2005) Representing structure-function relationships in mechanistically diverse enzyme superfamilies. *Pac. Symp. Biocomput*., 358–369.

Porter,C.T. *et al*. (2004) The Catalytic Site Atlas: a resource of catalytic sites and residues identified in enzymes using structural data. *Nucleic Acids Res*., **32**, D129–D133.

Schomburg,I. *et al*. (2004) BRENDA, the enzyme database: updates and major new developments. *Nucleic Acids Res*., **32**, D431–D433.

Wakelin,J. *et al*. (2005) CML tools and information flow in atomic scale simulations. *Mol. Simul*., **31**, 315–322.

Published online 1 November 2006 *Nucleic Acids Research, 2007, Vol. 35, Database issue* ***D515–D520***
doi:10.1093/nar/gkl774

MACiE (Mechanism, Annotation and Classification in Enzymes): novel tools for searching catalytic mechanisms

Gemma L. Holliday*, Daniel E. Almonacid[1], Gail J. Bartlett, Noel M. O'Boyle[1], James W. Torrance, Peter Murray-Rust[1], John B. O. Mitchell[1] and Janet M. Thornton

EMBL-EBI, Wellcome Trust Genome Campus, Hinxton, Cambridge CB10 1SD, UK and [1]Unilever Centre for Molecular Science Informatics, Department of Chemistry, University of Cambridge, Lensfield Road, Cambridge CB2 1EW, UK

Received August 4, 2006; Revised September 18, 2006; Accepted October 1, 2006

ABSTRACT

MACiE (Mechanism, Annotation and Classification in Enzymes) is a database of enzyme reaction mechanisms, and is publicly available as a web-based data resource. This paper presents the first release of a web-based search tool to explore enzyme reaction mechanisms in MACiE. We also present Version 2 of MACiE, which doubles the dataset available (from Version 1). MACiE can be accessed from http://www.ebi.ac.uk/thornton-srv/databases/MACiE/

INTRODUCTION

Enzymes are proteins that catalyse the repertoire of chemical reactions found in nature, and as such are vitally important molecules. What is so fascinating about these proteins is that they have a wonderful diversity and can carry out highly complex chemical conversions under physiological conditions and retain their stereospecificity and regiospecificity, unlike many organic chemical reactions. They range in size and can have molecular weights of several thousand to several million Daltons, and still they can catalyse reactions on molecules as small as carbon dioxide or nitrogen, or as large as a complete chromosome.

Although enzymes are large molecules, the actual catalysis only takes place in a small cavity, the active site. It is here that a small number of amino acid residues contribute to catalytic function, and where the substrates bind. With the advent of structure determination methods for proteins and by using clever chemical/biochemical experimental design, scientists have been able to propose catalytic mechanisms for many enzymes. Although a great deal of knowledge exists for enzymes, including their structures, gene sequences, mechanisms, metabolic pathways and kinetic data, it tends to be spread between many different databases and throughout the literature. Most web resources relating to enzymes [such as BRENDA (1), KEGG (2), the IUBMB Enzyme Nomenclature website (http://www.chem.qmul.ac.uk/iubmb/enzyme/) (3) and IntEnz (4)] focus on the overall reaction, accompanied in some cases by a textual or graphical description of the mechanism. However, this does not allow for detailed *in silico* searching of the chemical steps which take place in the reaction. MACiE (5) combines detailed stepwise mechanistic information [including 2-D animations (6)], a wide coverage of both chemical space and the protein structure universe, and the chemical intelligence of the Chemical Markup Language for Reactions (CMLReact) (7). This usefully complements both the mechanistic detail of the Structure–Function Linkage Database (SFLD) for a small number of rather 'promiscuous' enzyme superfamilies (8) and the wider coverage with less chemical detail provided by EzCatDB (9), which also contains a limited number of 3D animations. Entries in MACiE are linked, where appropriate, to all of these related data resources.

DATASET AND CONTENT

The dataset for MACiE version 2 was devised to increase the enzyme reaction space coverage of MACiE while trying to keep structural homology to a minimum. Each entry added in the new version was selected so that it fulfils the following criteria:

(i) The EC sub-subclass was not previously in MACiE.
(ii) There is a three-dimensional crystal structure of the enzyme deposited in the Protein Data Bank (wwPDB) (10).
(iii) There is a mechanism available from the primary literature which explains most of the observed experimental results.

*To whom correspondence should be addressed. Tel: +44 1223 492535; Fax: +44 1223 494486; Email: gemma@ebi.ac.uk
Present address:
Gail J. Bartlett, Division of Mathematical Biology, National Institute of Medical Research, The Ridgeway, Mill Hill, London NW7 1AA, UK

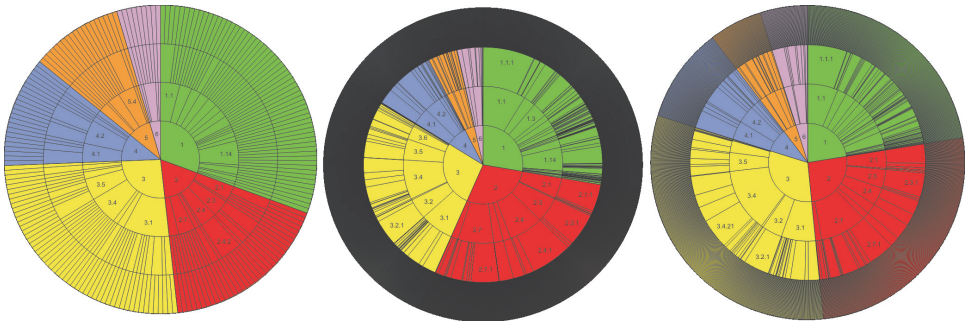

Figure 1. EC wheels showing the EC coverage of MACiE Version 2 (left), the complete EC space (centre) and the coverage of EC space in the PDB by unique EC serial numbers (right).

(iv) The enzyme is unique at the H level of the CATH code (11), unless the homologue already in MACiE has a significantly different chemical mechanism.

Using the above criteria MACiE was expanded from 100 entries in version 1 to a total of 202 entries, which span 199 EC numbers (version 1 spanned 96 EC numbers) and covers a total of 862 reaction steps. There are almost 4000 EC numbers defined, but the number of different reaction mechanisms needed to bring about all these overall transformations is not clear. For example, the serine protease family of proteins has many different substrates, but the mechanisms are broadly similar. In contrast the β-lactamase enzymes, which have the same EC number, have four completely different mechanisms. Within the EC code, the fourth digit usually defines the substrate specificity, which can be very variable in large enzyme families—but the reaction mechanisms for enzymes with the same first three digits are usually essentially the same. In total there are 224 EC sub-subclasses, with only 181 having known structures (12). Of these MACiE covers 158, i.e. 87%. However, there are probably many more mechanisms that are yet to be defined or discovered.

As can be seen from Figure 1, MACiE covers a good proportion of the EC reaction space, with an average relative difference between the size of corresponding EC classes of 4%, with the transferases having the largest difference. When the coverage with respect to EC code present in the PDB is examined, it can be seen that MACiE again represents the coverage of enzymes with known structures very well, with an average relative difference between the corresponding EC classes in MACiE of 5%.

All entries in MACiE contain overall reaction annotation including the information detailed in Table 1. Each elementary reaction or step within an entry is fully annotated as is detailed in Figure 2, this includes comments that have been added by the annotators. An extension of the content from MACiE Version 1 is the addition of inferred return steps. These are explicitly labelled as being inferred in the comment field and are necessary to return the enzyme to a state where it is ready to undergo another round of catalysis.

There is sometimes more than one proposed mechanism that is consistent with the available experimental data. In MACiE, we have attempted not only to choose the best supported mechanism, but also where possible to annotate enzymes with reasonable alternative mechanisms. Unfortunately, in the current release such annotations are only available as comments on the stage or overall reaction, although future releases of MACiE will include full entries for these alternatives.

Table 1. Overall reaction annotation content

Catalysis and reaction specific information	Non-catalysis specific information
Enzyme name (common IUPAB/JCBN name)	PDB code
EC code	Non-catalytic domain CATH code
Catalytic residues involved	Non-catalytic UniProt code
Cofactors involved	Species name (common and scientific)
Reactants and products	Other database identifiers, e.g. EzCatDB, SFLD, etc.
Catalytic domain CATH code	Literature references
Catalytic UniProt code	
Bonds involved, formed, cleaved, changed in order	
Reactive centres	
Overall reaction comments	

Further details of the annotation process and a glossary of terms used can be found on the MACiE website (http://www.ebi.ac.uk/thornton-srv/databases/MACiE/documentation/ and http://www.ebi.ac.uk/thornton-srv/databases/MACiE/glossary.html, respectively).

DATABASE STRUCTURE

The challenge with MACiE has been to capture and usefully represent all the different catalytic steps that occur during the course of an enzymatic reaction. These reactions may consist of any number of steps, and in MACiE we have reactions ranging from 1 step to 16 steps. The representation of these reactions has evolved from a flat file entered in a commercially available chemical database program (ISIS/Base) to the highly structured and powerful CMLReact (7), which is an application of XML (the eXtensible Markup Language). The final step in this evolution has been the conversion of the CMLReact into the relational database format of MySQL.

CMLReact has a heirarchical structure, facilitating its conversion into the relational database format of MySQL. The conversion relies on the CML Schema and requires the MACiE entries to be consistent with the Schema, which adds an internal consistency check into our authoring process.

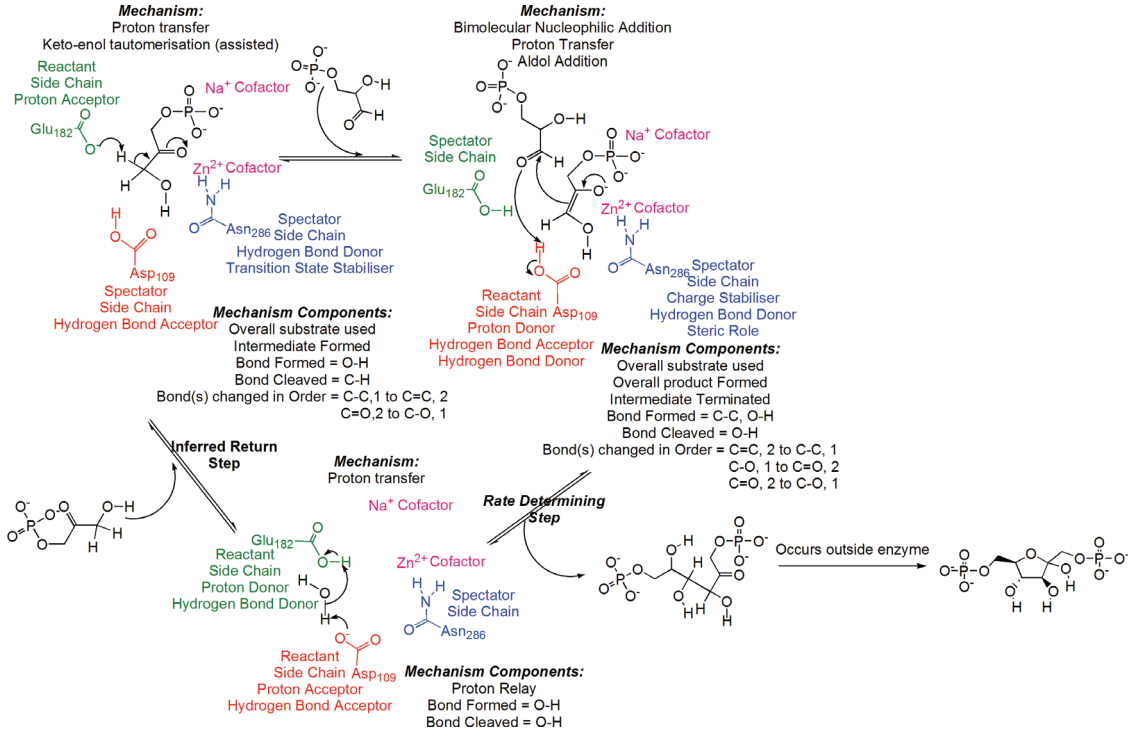

Figure 2. An example of the annotation found in a MACiE entry. Reaction shown corresponds to fructose-bisphosphate aldolase (entry 52).

Table 2. Searches available in MACiE

Basic	Complex
MACiE entry identifier	Species name (overall annotation)
Current EC codes	Overall reactants and products
Obsolete EC codes	Reaction comments (overall reactions and steps)
Catalytic Domain CATH codes	Amino acid residues (up to six residues)
All CATH codes	Step mechanisms and/or mechanism components (single and combinations of)
PDB code	Chemical changes
Enzyme name	Chemical changes with mechanism or mechanism components
Catalytic Domain UniProt Codes	Chemical changes with amino acid residues
All UniProt Codes	Amino acid residues with mechanism or mechanism components
	Chemical changes with amino acid residues and mechanisms or mechanism components
	Alternative mechanisms

Each CML tag-type becomes an MySQL table; each tag becomes a row in that MySQL table; each attribute of that tag corresponds to a column in the MySQL table. The tree structure of the CML is preserved in the MySQL version; for each row of each table, there are columns specifying which row of which other table corresponds to the row's parent tag in the CML version.

The CML version of MACiE, which is the official archive version, is available from the website as individual entries, and the new website uses the relational version of MACiE to perform the online analysis and searching.

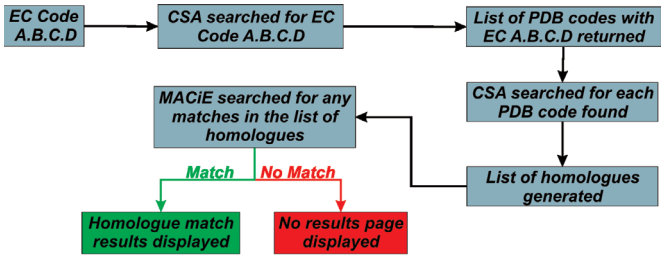

Figure 3. EC code search heuristics.

DATABASE FEATURES

The original release of MACiE contained static images and annotation for the overall reaction and each step associated with the mechanism; it also included an animated reaction mechanism for approximately half the reactions then in MACiE. Links to various related resources, such as the RCSB PDB (13), IUBMB nomenclature database, CATH, EzCatDB, PDBSum (14), BRENDA, the Catalytic Site Atlas (15), KEGG and the Enzyme Structures Database, were also included. This new release extends these links to include the Macromolecular Structures Database (MSD) (16), SFLD, UniProt (17), and replaces the IUBMB nomenclature database links with links to IntEnz. The new features in MACiE are detailed in the following sections.

Searching MACiE

There are two levels of search implemented in MACiE. The basic level searches are implemented from the main page (http://www.ebi.ac.uk/thornton-srv/databases/MACiE) and are

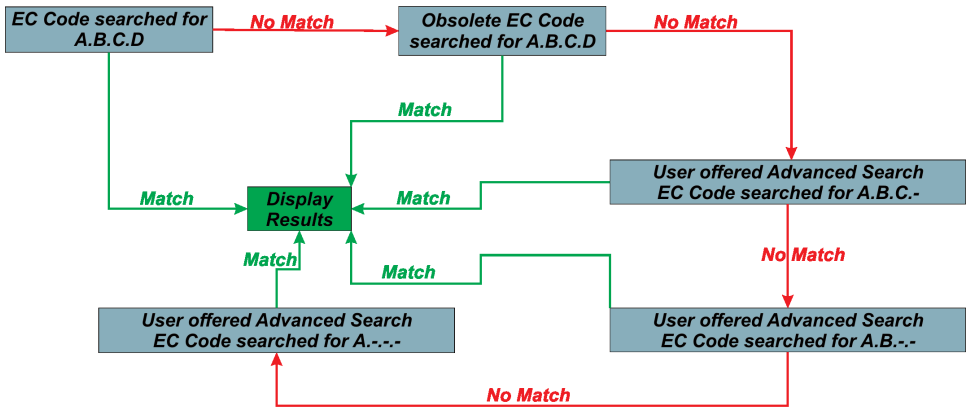

Figure 4. Advanced EC search heuristics.

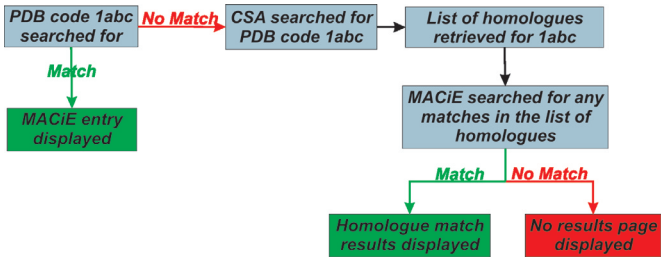

Figure 5. PDB search heuristics.

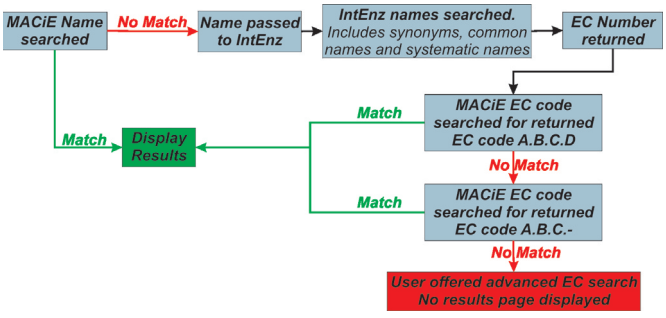

Figure 6. Enzyme name search heuristics.

mainly for accessing the entries from the top level, i.e. for searching entries in MACiE by EC code, enzyme name, etc. The complex searches are all available from the query pages of MACiE (http://www.ebi.ac.uk/thornton-srv/databases/MACiE/queryMACiE.html) and are mainly for searching for specific mechanisms, mechanism components or residues and their functions in the reaction steps, although there are some overall reaction searches implemented as well. Table 2 lists the searches available in MACiE and the Supplementary Data contain a detailed listing of the searches available.

The following sections describe searching by EC code, PDB code or enzyme name, all of which use heuristics to extend the coverage of MACiE.

EC code. The EC code search implemented in MACiE is detailed in Figure 3 and can be accessed at any point in the scheme shown. The search for current EC numbers will always walk up the EC code tree until it finds a match, no matter at what level the search is entered. Thus the search will always return a result. As the EC code of enzymes may change over time, a search for obsolete EC codes has also been implemented, although this search will not always return a result. However, it should be noted that the higher up the EC hierarchy search has gone, the less likely it is that the returned mechanism will be a match to the query. The obsolete EC code search works in the same way as the current EC code.

If no matches are found at the serial number level of the EC code, an advanced search option will allow the user to search for a structural homologue of an enzyme with a given EC code, which is shown in Figure 4 and described below. This advanced search option takes the entered EC code and finds the PDB codes of all of the matches to that EC code in the Catalytic Site Atlas (CSA). A homology search is then performed on those PDB codes for a match in MACiE. This homology search is described in more detail in the following section.

The CSA is a database of catalytic residues in proteins of known structure. It contains much less mechanistic information than MACiE, but has a considerably wider coverage of protein structures than MACiE does. This wider coverage is partly because the CSA contains not only manually annotated entries, but also contains entries that are automatically annotated based on sequence alignment to the manual entries.

PDB code. There are over 19 000 crystal structures relating to enzymes deposited in the PDB. As MACiE entries require extensive literature searching and analysis, only a small fraction of these PDB entries are covered explicitly, 202 in total. However, we have used the CSA to identify homologues of these enzymes, extending this coverage to 7528 PDB codes.

Figure 5 details the search performed in MACiE, when a protein structure described by a PDB code is entered. Although the entries returned by this search will be homologues, this does not guarantee that the mechanism and the catalytic residue assignments are the same. This is because the homology method (see below) can retrieve very distant relatives. Owing to this limitation, all homologous entries are compared by EC code, and when there is a divergence between the MACiE entry and the homologue at the serial number level, this is clearly indicated to the user. We also

list the amino acid residues that are annotated as catalytic in both MACiE and the CSA. Thus it is clear if there is any difference between EC numbers and catalytic residues. If the EC number differs but the catalytic residues between query and homologue are of identical types, it can be inferred that the mechanisms are likely to be the same, but where both differ, the mechanisms are unlikely to be transferable. From the results page we link both to the MACiE entry and the CSA entry.

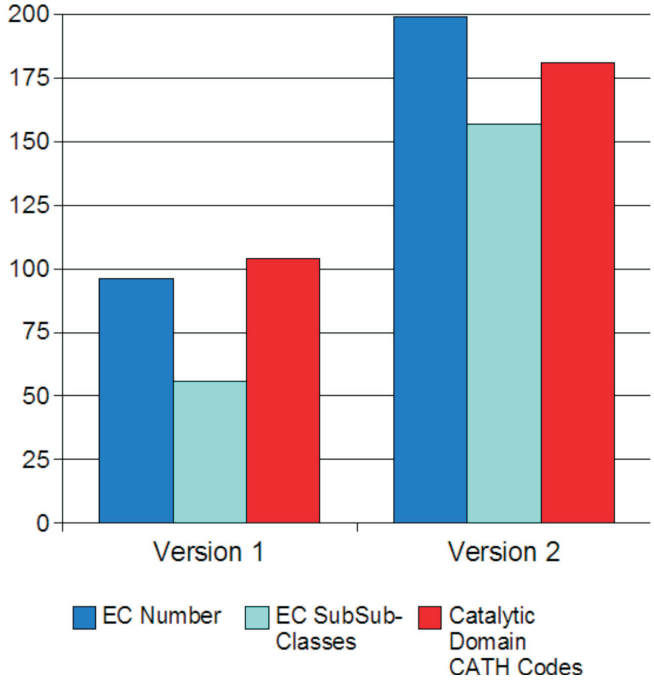

Figure 7. Growth of MACiE. This shows the growth in the number of EC codes (blue), EC sub-sub classes (cyan) and catalytic domain CATH codes (red) in MACiE.

Homology in MACiE. We have been working to bring MACiE and the CSA closer together. This includes using the CSA to determine homologues (those enzymes which are evolutionarily related) of entries in MACiE. The CSA finds homologues using a PSI-BLAST search (with an *E*-value cut-off of 0.0005 and five iterations) against all sequences currently in the PDB, plus all sequences in a non-redundant subset of UniProt. The UniProt sequences are included purely in order to increase the range of the PSI-BLAST search by bridging gaps between distantly related sequences in the PDB; only sequences occurring in the PDB are retrieved for entry into the CSA. In the CSA, and thus MACiE, homologous entries are only included if the residues which align with the catalytic residues in the parent literature entry are identical in residue type. In other words, there must be no mutations at the catalytic residue positions. There are, however, a few exceptions to this rule:

(i) In order to allow for the many active site mutants in the PDB, one (and only one) catalytic residue per site can be different in type from the equivalent in the parent literature entry. This is only permissible if all residue spacing is identical to that in the parent literature entry, and there are at least two catalytic residues.
(ii) Sites with only one catalytic residue are permitted to be mutant provided that the residue number is identical to that in the parent entry.
(iii) Fuzzy matching of residues is permitted within the following groups: [V,L,I], [F,W,Y], [S,T], [D,E], [K,R], [D,N], [E,Q], [N,Q]. This fuzzy matching cannot be used in combination with rules (i) or (ii) above.

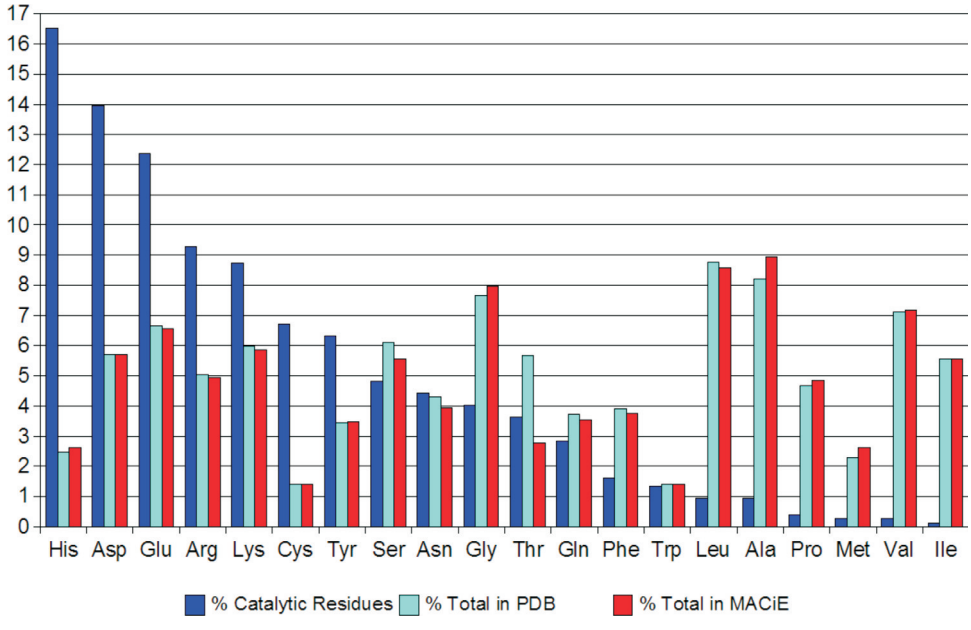

Figure 8. Frequency distribution of amino acid residues. This shows the frequency of catalytic amino acid residues in MACiE (blue), versus the frequency of residues in MACiE (cyan), versus the frequency of residues in the wwPDB (red). The frequency of catalytic amino acid residues in MACiE is calculated by taking the number of residues (of a given type) annotated in MACiE divided by the total number of annotated residues in MACiE, multiplied by 100.

Enzyme name. This is currently implemented as a partial string match, thus entering 'beta' will return all the β-lactamases and betaine-aldehyde dehydrogenase. If no results are returned from the partial name search, then the name search heuristics (shown in Figure 6) are implemented.

This search utilizes the IntEnz database (4). MACiE searches for a name in IntEnz, either a synonym, alternative name or common name, and returns the EC code of that name. The EC code is then used to search MACiE. If no matches are found to the sub-subclass level of the EC code, the user is offered an advanced EC code search (see Figure 4).

Statistics

The other major development in MACiE has been the inclusion of database statistics that are all generated on the fly from the SQL tables. A full listing of the statistics available can be found in the Supplementary Data. The growth of MACiE is shown in Figure 7 in terms of EC coverage and CATH coverage.

The statistics in MACiE can also be used to examine the function and distribution of amino acid residues (G.L. Holliday, D.E. Almonacid, J.M. Thornton and J.B.O. Mitchell, manuscript in preparation) (see Figure 8), the distribution of mechanism and mechanism components and the bond order changes occurring in each step of the reaction.

FUTURE DEVELOPMENTS

MACiE is a continually developing resource, and in the future we hope to include 3D data, which will incorporate various statistics and searches related to the analysis of these data. We will also continue to extend the coverage of MACiE to include alternative reaction mechanisms that have been suggested for various enzymes, as well as new mechanisms. Finally, we intend to build a user interface which will allow for chemical diagrams to be drawn and used to search MACiE, an entry process which is more usable and also to implement the classification of enzyme mechanisms that we are developing.

SUPPLEMENTARY DATA

Supplementary Data are available at NAR Online.

ACKNOWLEDGEMENTS

We would like to thank the EPSRC (G.L.H. and J.B.O.M.), BBSRC (G.J.B. and J.M.T.—CASE studentship in association with Roche Products Ltd; N.M.O.B. and J.B.O.M.—grant BB/C51320X/1), the Wellcome Trust, EMBL, IBM (G.L.H. and J.M.T.), the Chilean Government's Ministerio de Planificación y Cooperación and the Cambridge Overseas Trust (D.E.A.) for funding and Unilever for supporting the Centre for Molecular Science Informatics. J.W.T. is funded by a European Molecular Biology Laboratory studentship, and is also affiliated with Cambridge University Department of Chemistry. Funding to pay the Open Access publication charges for this article was provided by the Wellcome Trust.

Conflict of interest statement. None declared.

REFERENCES

1. Schomburg,I., Chang,A., Ebeling,C., Gremse,M., Heldt,C., Huhn,G. and Schomburg,D. (2004) BRENDA, the enzyme database: updates and major new developments. *Nucleic Acids Res*., **32**, D431–D433.
2. Kanehisa,M., Goto,S., Kawashima,S., Okuno,Y. and Hattori,M. (2004) The KEGG resource for deciphering the genome. *Nucleic Acids Res*., **32**, D277–D280.
3. IUBMB (2005) Recommendations of the Nomenclature Committee of the International Union of Biochemistry and Molecular Biology on the nomenclature and classification of enzyme-catalysed reactions.
4. Fleischmann,A., Darsow,M., Degtyarenko,K., Fleischmann,W., Boyce,S., Axelsen,K., Bairoch,A., Schomburg,D., Tipton,K.F. and Apweiler,R. (2004) IntEnz, the integrated relational enzyme database. *Nucleic Acids Res*., **32**, D434–D437.
5. Holliday,G.L., Bartlett,G.J., Almonacid,D.E., O'Boyle,N.M., Murray-Rust,P., Thornton,J.M. and Mitchell,J.B.O. (2005) MACiE: a database of enzyme reaction mechanisms. *Bioinformatics*, **21**, 4315–4316.
6. Holliday,G.L., Mitchell,J.B.O. and Murray-Rust,P. (2004) CMLSnap: animated reaction mechanisms. *Internet J. Chem*., **7,** Article 4.
7. Holliday,G.L., Murray-Rust,P. and Rzepa,H.S. (2006) Chemical Markup, XML, and the World Wide Web. 6. CMLReact, an XML vocabulary for chemical reactions. *J. Chem. Inf. Model*., **46**, 145–157.
8. Pegg,S.C.-H., Brown,S.D., Ojha,S., Seffernick,J., Meng,E.C., Morris,J.H., Chang,P.J., Huang,C.C., Ferrin,T.E. and Babbitt,P.C. (2006) Leveraging enzyme structure–function relationships for functional inference and experimental design: the Structure–Function Linkage Database. *Biochemistry*, **45**, 2545–2555.
9. Nagano,N. (2005) EzCatDB: the Enzyme Catalytic-mechanism DataBase. *Nucleic Acids Res*., **33**, D407–D412.
10. Berman,H.M., Henrick,K. and Nakamura,H. (2003) Announcing the worldwide Protein Data Bank. *Nature Struct. Biol*., **10**, 980.
11. Orengo,C.A., Michie,A.D., Jones,S., Jones,D.T., Swindells,M.B. and Thornton,J.M. (1997) CATH—a hierarchic classification of protein domain structures. *Structure*, **5**, 1093–1108.
12. Martin,A.C. (2004) PDBSprotEC: a Web-accessible database linking PDB chains to EC numbers via SwissProt. *Bioinformatics*, **20**, 986–988.
13. Berman,H.M., Westbrook,J., Feng,Z., Gilliland,G., Bhat,T.N., Weissig,H., Shindyalov,I.N. and Bourne,P.E. (2000) The Protein Data Bank. *Nucleic Acids Res*., **28**, 235–242.
14. Laskowski,R.A., Chistyakov,V.V. and Thornton,J.M. (2005) PDBsum more: new summaries and analyses of the known 3D structures of proteins and nucleic acids. *Nucleic Acids Res*., **33**, D266–D268.
15. Porter,C.T., Bartlett,G.J. and Thornton,J.M. (2004) The Catalytic Site Atlas: a resource of catalytic sites and residues identified in enzymes using structural data. *Nucleic Acids Res*., **32**, D129–D133.
16. Golovin,A., Oldfield,T.J., Tate,J.G., Velankar,S., Barton,G.J., Boutselakis,H., Dimitropoulos,D., Fillon,J., Hussain,A., Ionides,J.M. *et al*. (2004) E-MSD: an integrated data resource for bioinformatics. *Nucleic Acids Res*., **32**, D211–D216.
17. Bairoch,A., Apweiler,R., Wu,C.H., Barker,W.C., Boeckmann,B., Ferro,S., Gasteiger,E., Huang,H., Lopez,R., Magrane,M. *et al*. (2005) The Universal Protein Resource (UniProt). *Nucleic Acids Res*., **33**, D154–D159.

Part III

QSAR

BIOINFORMATICS APPLICATIONS NOTE Vol. 22 no. 20 2006, pages 2565–2566
doi:10.1093/bioinformatics/btl416

Data and text mining

PYCHEM: a multivariate analysis package for python

Roger M. Jarvis[1,4,*], David Broadhurst[1,4], Helen Johnson[2], Noel M. O'Boyle[3] and Royston Goodacre[1,4]

[1]School of Chemistry, The University of Manchester, PO Box 88, Sackville Street, Manchester M60 1QD, UK, [2]Faculty of Life Sciences, University of Manchester, Stopford Building, Oxford Road, Manchester M13 9PT, UK, [3]Unilever Centre for Molecular Science Informatics, Department of Chemistry, University of Cambridge, Lensfield Road, CB2 1EW, UK and [4]Manchester Interdisciplinary Biocentre, 131 Princess Street, Manchester M1 7DN, UK

Received on April 4, 2006; revised on July 5, 2006; accepted on July 26, 2006
Advance Access publication July 31, 2006
Associate Editor: Martin Bishop

ABSTRACT

Summary: We have implemented a multivariate statistical analysis toolbox, with an optional standalone graphical user interface (GUI), using the Python scripting language. This is a free and open source project that addresses the need for a multivariate analysis toolbox in Python. Although the functionality provided does not cover the full range of multivariate tools that are available, it has a broad complement of methods that are widely used in the biological sciences. In contrast to tools like MATLAB, PyChem 2.0.0 is easily accessible and free, allows for rapid extension using a range of Python modules and is part of the growing amount of complementary and interoperable scientific software in Python based upon SciPy. One of the attractions of PyChem is that it is an open source project and so there is an opportunity, through collaboration, to increase the scope of the software and to continually evolve a user-friendly platform that has applicability across a wide range of analytical and post-genomic disciplines.

Availability: http://sourceforge.net/projects/pychem

Contact: Roger.Jarvis@manchester.ac.uk or admin@pychem.org.uk

Supplementary information: Further information is available from the project home page at http://pychem.sf.net/ whilst details of data generation are available at http://biospec.net/

1 INTRODUCTION

Increasingly in the life sciences many experiments generate data which are of a multivariate nature, where many observations are recorded for each sample under analysis. Interpretation of such complex data cannot generally be performed by taking a univariate approach, since no single measurement is necessarily adequate enough to describe the problem being addressed. In fact, the application of univariate methodology is in many cases totally inappropriate as the complexity of information contained within large biological datasets reflects the complexity of the system(s) being studied. Typical multivariate analysis problems involve unsupervised learning such as factor analysis, for reducing the dimensionality of data and modeling of variance; linear regression, for formulating input to output transformation models based on supervised learning which are predictive generally for quantitative trait(s); and discriminant analysis, for distinguishing between different sample groups and for subsequent predictions on new samples. In fact, multivariate analysis encompasses many more methods than these examples of linear modeling imply (Brereton, 2003); but these tools are perhaps those most commonly used for the modeling of biological data.

Many programs currently exist for multivariate analysis. Flexible environments for mathematical computing are available in the form of MATLAB (The Mathworks, Natick, MA, USA), GNU Octave (http://www.octave.org/) which aims to be a free equivalent of MATLAB, and R (http://www.r-project.org/); which has many other bio-analysis modules, such as Vegan (for environmetrics) and Bioconductor (for genomic analysis). These products provide powerful tools for multivariate analysis through command line interpreters, which allow the user to perform their analysis with a great degree of flexibility. However, they require some investment in time to become familiar with the interpreters syntax, and are not necessarily straightforward for people with little computational experience. In addition, a number of graphical multivariate software tools are also available; Evince (UmBio, Umeå, Sweden), The Unscrambler (CAMO, Woodbridge, NJ, USA), Pirouette (Infometrix, Bothell, WA, USA), S-Plus (Insightful, Seattle, WA. USA) and SIMCA (Umetrics, Umeå, Sweden) are all good tools for basic multivariate analysis although, with the exception of S-Plus, they lack the flexibility of the interpreter style interfaces.

Thus there is currently a requirement for a flexible, extensible, free and open source graphical environment for performing multivariate analysis, which can be used by both experts and casual users. The increasing popularity of scripting languages such as Python (http://www.python.org/) within the life sciences community offers the technology and critical mass for such a project. A platform of this type addresses the requirements outlined above, with the additional benefit that it allows for the rapid development of new cross-platform software approaches, and the integration of currently available software libraries through application programming interfaces (APIs).

2 THE MULTIVARIATE ANALYSIS TOOLBOX FOR PYTHON

The PyChem project aims to provide a simple multivariate analysis toolbox with a powerful and intuitive GUI front-end.

*To whom correspondence should be addressed.

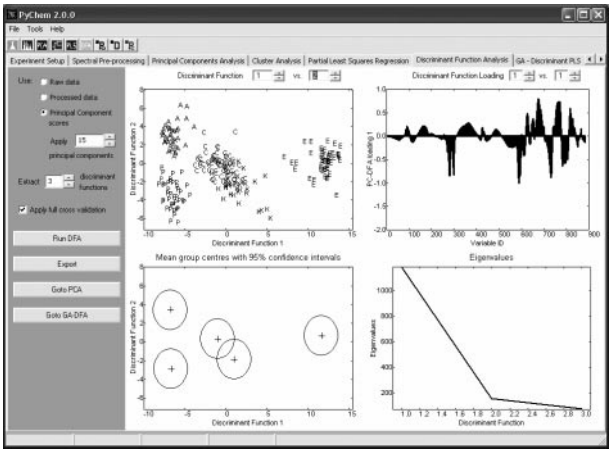

Fig. 1. A screenshot demonstrating the feature selection functionality available in PyChem, in this example microarray data (Golub *et al*., 1999) have been analysed consisting of 72 samples represented by 7070 genes. The GA directed search can be used to highlight genes that are particularly important for discrimination.

The project is implemented in Python and utilizes the wxPython (http://www.wxpython.org/), Boa Constructor (http://boa-constructor.sourceforge.net/) and SciPy (http://scipy.org/) packages (see Fig. 1 for an example screenshot) amongst others. The software was designed to provide a range of algorithms that address three fundamental questions commonly asked by the researcher.

(1) What is the shape of the data—including sources of variance and outlier identification?

(2) How similar are different samples?

(3) Which measurements from the original data can be attributed to observed differences and/or similarities?

To help answer these questions, the initial release includes algorithms for the pre-processing of multivariate data (such as scaling, baseline correction, filtering and derivatization), principal components analysis (PCA) (Jolliffe, 1986), partial least squares regression (PLS1) (Martens and Naes, 1989), discriminant function analysis (DFA) (Manly, 1994), cluster analysis [using the C clustering library for Python (http://bonsai.ims.u-tokyo.ac.jp/~mdehoon/software/cluster/) (Eisen *et al*., 1998; de Hoon *et al*., 2004)], and a number of genetic algorithm (GA) based tools for performing feature selection (Jarvis and Goodacre, 2005), see Fig. 1.

The software is able to handle any 2D dataset where each sample is defined by a series of discrete or continuous measurements. Data can be imported from flat ASCII files that use the standard delimiters. Typical data of this type include those generated from microarrays, proteomics, spectroscopic methods (UV-Vis, infrared and Raman), mass spectrometry, NMR, or indeed any data arrays representing samples for which multiple discrete measurements have been acquired. Once data have been imported into PyChem they can be saved in an XML format [implemented using cElementTree (http://effbot.org/)] as a PyChem experiment, which allows for the subsequent storage of multiple experimental results within a single file. This allows for the capture of the state of the system at a point in time, so that results of multivariate analyses can be stored and the progression of the analysis recorded, which is particularly useful for tracking data analyses as part of GLP. The additional benefit of using the XML data structure for storage is that it introduces the potential for engineering simple bespoke interfaces to database storage systems.

PyChem provides simple grid-style user interfaces for the input of experimental and sample metadata, so producing a series of vectors describing the origin and identity of each sample and measured variable. For unsupervised analyses, such as PCA, the software simply requires a vector, or multiple vectors of sample labels for plotting; in addition for supervised analyses, vectors are required to (1) represent putative class structures or some quantitative trait (e.g., level of abiotic or biotic interference) and (2) identify groups in to which the data should be split for the purpose of cross-validation. In supervised analyses the issue of model validation is crucial; when a model is formulated there is a possibility that it will overfit the data and find a relationship between the data and the target class structure or dependent variables, which does not hold for subsequent predictions; i.e. the model has learnt the training data perfectly and is not able to generalize. This situation can be avoided by performing some form of model validation. In the current version of PyChem (2.0.0) we use the preferred approach of data splitting (Brereton, 2003), which works by dividing the measured X-variables in to three groups; a model training set, model cross-validation data and finally an independent test set. The model is trained on the first set, optimized on the second set and then tested for accuracy on the third set of 'hold-out' data.

A major emphasis of this work has been in providing clear and useful graphical reports for the interpretation of results. The GUI uses wxPyPlot (http://www.cyberus.ca/~g_will/wxPython/wxpyplot.html), with a small modification to include text plotting. In the future even more focus will be given to the structure of graphical reporting in PyChem, as well as the functionality associated with the plotting canvases. Finally, all results, both graphical and numerical, can easily be exported from PyChem, with numerical results in ASCII file format to allow for use in other software applications.

ACKNOWLEDGEMENTS

R.M.J., D.B., H.J., N.M.O.B. and R.G. would like to thank the BBSRC for funding (NMOB; grant BB/C51320X/1). Funding to pay the Open Access publication charges for this article was provided by the BBSRC.

Conflict of Interest: none declared.

REFERENCES

Brereton,R. (2003) Chemometrics: data analysis for the laboratory and chemical plant, 1st edn. Chichester: John Wiley & Sons Ltd.

Eisen,M. *et al*. (1998) Cluster analysis and display of genome-wide expression patterns. *Proc. Natl Acad. Sci.USA*, **95**, 14863–14868.

de Hoon,M. *et al*. (2004) Open source clustering software. *Bioinformatics*, **20**, 1453–1454.

Golub,T. *et al*. (1999) Molecular classification of cancer: class discovery and class prediction by gene expression monitoring. *Science*, **286**, 531–537.

Jarvis,R. and Goodacre,R. (2005) Genetic algorithm optimization for pre-processing and variable selection of spectroscopic data. *Bioinformatics*, **21**, 860–868.

Jolliffe,I.T. (1986) *Principal Component Analysis*. Springer-Verlag, New York.

Manly,B.F.J. (1994) *Multivariate Statistical Methods: A Primer*. Chapman & Hall/CRC, New York.

Martens,H. and Naes,T. (1989) *Multivariate Calibration*. John Wiley & Sons, Chichester.

Chemistry Central Journal

Methodology

Simultaneous feature selection and parameter optimisation using an artificial ant colony: case study of melting point prediction

Noel M O'Boyle*[1,2], David S Palmer[1,3], Florian Nigsch[1] and John BO Mitchell[1]

Address: [1]Unilever Centre for Molecular Science Informatics, Dept. of Chemistry, University of Cambridge, Lensfield Rd, Cambridge, CB2 1EW, UK, [2]Cambridge Crystallographic Data Centre, 12 Union Rd, Cambridge, CB2 1EZ, UK and [3]Department of Chemistry, Aarhus University, 8000 Aarhus C, Denmark

Email: Noel M O'Boyle* - baoilleach@gmail.com; David S Palmer - dsp@chem.au.dk; Florian Nigsch - fn211@cam.ac.uk; John BO Mitchell - jbom1@cam.ac.uk

* Corresponding author

Published: 29 October 2008

Chemistry Central Journal 2008, **2**:21 doi:10.1186/1752-153X-2-21

Received: 1 August 2008
Accepted: 29 October 2008

This article is available from: http://journal.chemistrycentral.com/content/2/1/21

Abstract

Background: We present a novel feature selection algorithm, Winnowing Artificial Ant Colony (WAAC), that performs simultaneous feature selection and model parameter optimisation for the development of predictive quantitative structure-property relationship (QSPR) models. The WAAC algorithm is an extension of the modified ant colony algorithm of Shen et al. (*J Chem Inf Model* 2005, **45**: 1024–1029). We test the ability of the algorithm to develop a predictive partial least squares model for the Karthikeyan dataset (*J Chem Inf Model* 2005, **45**: 581–590) of melting point values. We also test its ability to perform feature selection on a support vector machine model for the same dataset.

Results: Starting from an initial set of 203 descriptors, the WAAC algorithm selected a PLS model with 68 descriptors which has an RMSE on an external test set of 46.6°C and R^2 of 0.51. The number of components chosen for the model was 49, which was close to optimal for this feature selection. The selected SVM model has 28 descriptors (cost of 5, ε of 0.21) and an RMSE of 45.1°C and R^2 of 0.54. This model outperforms a *k*NN model (RMSE of 48.3°C, R^2 of 0.47) for the same data and has similar performance to a Random Forest model (RMSE of 44.5°C, R^2 of 0.55). However it is much less prone to bias at the extremes of the range of melting points as shown by the slope of the line through the residuals: -0.43 for WAAC/SVM, -0.53 for Random Forest.

Conclusion: With a careful choice of objective function, the WAAC algorithm can be used to optimise machine learning and regression models that suffer from overfitting. Where model parameters also need to be tuned, as is the case with support vector machine and partial least squares models, it can optimise these simultaneously. The moving probabilities used by the algorithm are easily interpreted in terms of the best and current models of the ants, and the winnowing procedure promotes the removal of irrelevant descriptors.

Background

Quantitative Structure-Activity and Structure-Property Relationship (QSAR and QSPR) models are based upon the idea, first proposed by Hansch [1], that a molecular property can be related to physicochemical descriptors of the molecule. A QSAR model for prediction must be able to generalise well to give accurate predictions on unseen test data. Although it is true in general that the more descriptors used to build a model, the better the model predicts the training set data, such a model typically has very poor predictive ability when presented with unseen test data, a phenomenon known as overfitting [2]. Feature selection refers to the problem of selecting a subset of the descriptors which can be used to build a model with optimal predictive ability [3]. In addition to better prediction, the identification of relevant descriptors can give insight into the factors affecting the property of interest.

The number of subsets of a set of n descriptors is 2^n-1. Unless n is small (<20) it is not feasible to test every possible subset, and the number of descriptors calculated by cheminformatics software is usually much larger (CDK [4], MOE [5] and Sybyl [6] can respectively calculate a total of 95, 146 and 248 1D and 2D descriptors). Feature selection methods can be divided into two main classes: the filter approach and the wrapper approach [3,7,8]. The filter approach does not take into account the particular model being used for prediction, but rather attempts to determine *a priori* which descriptors are likely to contain useful information. Examples of this approach include ranking descriptors by their correlation with the target value or by estimates of the mutual information (based on information theory) between each descriptor and the response. Another commonly used filter in QSAR is the removal of highly correlated (or anti-correlated) descriptors [9]. Liu [10] presents a comparison of five different filters in the context of prediction of binding affinities to thrombin. The filter approach has the advantages of speed and simplicity, but the disadvantage that it does not explicitly consider the performance of the model containing different features. Correlation criteria can only detect linear dependencies between descriptor values and the response, but the best performing QSAR models are often non-linear (support vector machines (SVM), neural networks (NN) and random forests (RF), for example). In addition, Guyon and Elisseeff show that very high correlation (or anti-correlation) does not necessarily imply an absence of feature complementarity, and also that two variables that are useless by themselves can be useful together [3].

The wrapper approach conducts a search for a good feature selection using the induction algorithm as a black box to evaluate subsets and calculate the value of an objective function. The objective function should provide an estimate of how well the model will generalise to unseen data drawn from the same distribution. The purpose of the search is to find the feature selection that optimises this value. The most well-known deterministic wrapper is sequential forward selection [11] (SFS) which involves successive additions of the feature that most improves the objective function to the subset of descriptors already chosen. A related algorithm, sequential backwards elimination [12] (SBE), successively eliminates descriptors starting from the complete set of descriptors. Both of these algorithms suffer from the problem of 'nesting'. In the case of SFS, nesting refers to the fact that once a particular feature is added it cannot be removed at a later stage, even if this would increase the value of the objective function. More sophisticated methods, such as the sequential forward floating selection (SFFS) algorithm of Pudil et al. [13], include a backtracking phase after each addition where variables are successively eliminated if this improves the objective function. Wrapper methods specific to certain models have also been developed. For example, the Recursive Feature Elimination algorithm of Guyon et al. [14] and the Incremental Regularised Risk Minimisation of Fröhlich et al. [15] are specific to models built using support vector machines.

Stochastic wrappers attempt to deal with the size of the search space by incorporating some degree of randomness into the search strategy. The most well known of these algorithms is the genetic algorithm [16] (GA), whose search procedure mimics the biological process of evolution. A number of models are created randomly in the first generation, the best of which (as measured by the objective function) are selected and interbred in some way to create the next generation. A mutation operator is applied to the new models so that random sampling of the local space occurs. Over the course of many generations, the objective function is optimised. Genetic algorithms were first used for feature selection in QSAR by Rogers and Hopfinger [17] and are now used widely [9,18,19]. Other stochastic methods which have been used for feature selection in QSAR are particle swarm optimisation [20,21] and simulated annealing [22].

An additional difficulty in the development of QSAR models is the fact that some regression methods have parameters that need to be optimised to obtain the best performance for a particular problem. The Support Vector Machine (SVM) is an example of such a method. A SVM is a kernel-based machine learning method used for both classification and regression [23-25] which has shown very good performance in QSAR studies [9]. In ε-SVM regression, the algorithm finds a hyperplane in a transformed space of the inputs that has at most ε deviation from the output y values. Deviations greater than ε are penalised by multiplying by a cost value C. The transfor-

mation of the inputs is carried out by means of kernel functions, which allows nonlinear relationships between the inputs and the outputs to be handled by this essentially linear method. For a particular problem and kernel, the values of C and ε must be tuned.

Here we describe WAAC, Winnowing Artificial Ant Colony, a stochastic wrapper for feature selection and parameter optimisation that combines simultaneous optimisation of the selected descriptors and the model parameters to create a model with good predictive accuracy. This method does not require any pre-processing of the data apart from removal of zero-variance and duplicate descriptors. The only requirement is that allowed values of parameters of the models must be specified. As a result, this method is suitable for use as an automatic generator of predictive models.

The WAAC algorithm is a novel stochastic wrapper derived from the modified Ant Colony Optimisation (ACO) algorithm of Shen *et al.* [26]. Ant colony algorithms take their inspiration from the foraging of ants whose cooperative behaviour enables the shortest path between nest and food to be found [27]. Ants deposit a substance called pheromone as they walk, thus forming a pheromone trail. At a branching point, an ant is more likely to choose the trail with the greater amount of pheromone. Over time as pheromones evaporate, only those trails that have been reinforced by the passage of many ants will retain appreciable amounts of pheromone, with the shortest trail having the greatest amount of pheromone. In the end, all of the ants will travel by the shortest trail. Artificial ant colony systems may be used to solve combinatorial optimisation problems by making use of the ideas of cooperation between autonomous agents through global knowledge and positive feedback that are observed in real ant colonies [28].

The first use of artificial ant systems for variable selection in QSAR was the ANTSELECT algorithm of Izrailev and Agrafiotis [29]. The ANTSELECT algorithm involves the movement of a single ant through feature space. Initially equal weights are assigned to each descriptor. The probability of the ant choosing a particular descriptor in the next iteration is the weight for that descriptor divided by the sum of all weights. After the fitness of the model is assessed, all of the weights are reduced by multiplying by (1-ρ), where ρ is the evaporation coefficient. The weights of those descriptors selected in the current iteration are then increased by a constant multiple of the fitness score. Gunturi *et al.* [30] used a modification of the ANTSELECT algorithm in a recent study of human serum albumin binding affinity in which the number of features selected was fixed *a priori* and, in addition, could not include descriptors that had a correlation coefficient greater than 0.75.

Since the ANTSELECT algorithm uses only a single ant, it cannot make use of one of the most important features of ant colony algorithms, collective intelligence. Instead, premature convergence will occur due to positive reinforcement of models that have performed well earlier in the local search. In addition, the search space will be poorly covered. Although the authors recommend that the algorithm should be repeated several times to minimise the likelihood of convergence to a poor local minimum, the use of an ant colony is a much more robust solution.

Shen *et al.* [26] presented an ACO algorithm that differed from ANTSELECT in several ways. Their algorithm, which they called a modified ACO, is similar to our WAAC algorithm in that it involves a colony of ants, each of which remembers its best model and score, as well as its current model and score. In Shen *et al.*'s algorithm, for every descriptor there are both positive and negative weights. The probability that an ant will choose a particular descriptor is given by the positive weight for that descriptor divided by the sum of the positive and negative weights. After every iteration, the weights are reduced by multiplying by (1-ρ) as for ANTSELECT. The positive weight for a particular descriptor is increased by the sum of the fitness scores of all ants in the current iteration that have selected it, as well as the fitness scores of the best models of all ants that have selected it in that model. Similarly, the negative weight for a particular descriptor is decreased by an amount based on the fitness scores of models that have not selected it.

In the following section, we describe the WAAC algorithm in detail, as well as the dataset and model used to test the algorithm. In the Results and Discussion sections, we describe the performance of the WAAC algorithm, compare it to other models on the same dataset, and discuss some practical considerations in usage.

Methods

WAAC algorithm

The WAAC algorithm uses a population of candidate models termed an 'ant colony'. Each ant represents a model; that is, it is associated with a particular feature selection as well as particular values for the model (for example, SVM) parameters. The set of descriptors is stored as a binary fingerprint of length F (the number of descriptors), where a value of 1 for the n^{th} bit indicates that the n^{th} descriptor is selected, and 0 indicates that it is not. For each parameter of the model, a range of discrete values is required. The parameter values used by a particular ant are stored in a list of length P, where P is the number of adjustable parameters of the model. The fitness of each model is measured using an objective function specified by the user.

The initial population of ants is randomly placed in feature and parameter space. The bits of the binary fingerprints representing the feature selections are initialised to either 0 or 1 with equal probability, so that on average each ant corresponds to a model based on approximately 50% of the descriptors. Conversely, each descriptor is initially selected by approximately 50% of the ants. The initial parameter values for each ant are chosen at random from the available values for each parameter.

Figure 1 shows a schematic of the WAAC algorithm. After initialisation, the algorithm enters the optimisation phase. For each descriptor, a moving probability is calculated by taking the average of the fraction of ants which have currently selected that descriptor and the fraction that have selected that descriptor in their best model. This moving probability is used to determine the chance that a particular ant will select a particular descriptor in the next iteration. At the start of the optimisation phase, the moving probabilities for all of the descriptors will be approximately equal to 0.5 (since the best model will be the current model and each descriptor is selected by approximately 50% of the ants).

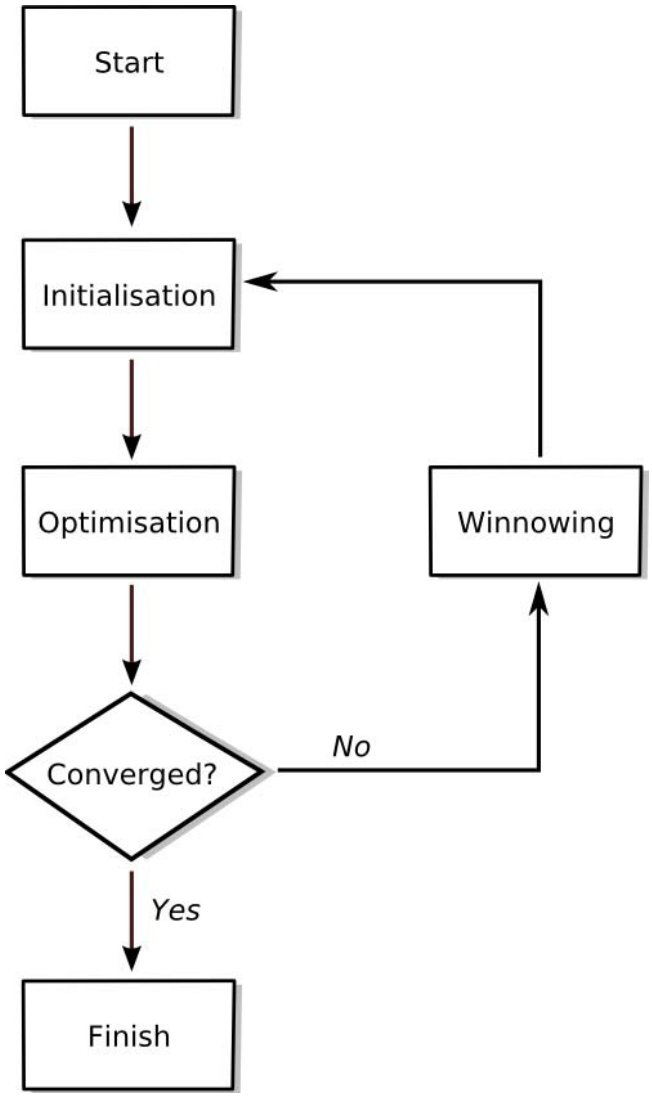

Figure 1
Outline of the WAAC algorithm.

Similarly, for each parameter there is a moving probability associated with every allowed value. These moving probabilities sum to unity (since each ant needs to select exactly one allowed value for each parameter), and are calculated by taking the average of the fraction of ants which have currently selected a particular allowed value and the fraction of ants that have selected that value in their best model. At the start of the optimisation phase, each allowed value of a parameter will be selected by approximately N/P ants where N is the number of ants, and P the number of allowed values.

At the start of the optimisation phase, the ants move more or less randomly, as the moving probabilities are essentially equal for all features and parameter values. However, over the course of the optimisation phase as particular descriptors are found to occur frequently in the best models associated with the ants, due to positive feedback these descriptors will be more likely to be chosen in subsequent iterations. This global optimisation procedure is combined with local optimisation due to the influence of the current positions of the ants on the moving probabilities. Note that the ants do not move about relative to their position in a previous iteration; rather, their subsequent location in feature space is determined by the best and current feature selections of all of the ants. Note that nesting is not a problem, as in each step of the optimisation the ants are free to explore descriptor combinations which did not exist in the previous step.

After multiple iterations of the optimisation algorithm, a winnowing procedure is applied. This reduces the search space by retaining only those descriptors that have been chosen by at least 20% of the ants in their best models, and removing the rest. Parameter values are reinitialised randomly. Some descriptors may be retained that do not improve the models, but the subsequent reinitialisation of the ants on the smaller search space will allow the subsequent optimisation phase to identify better models which exclude that descriptor. Note that no information is carried from one optimisation procedure to the next. In particular, memory of previous best models does not guide future searching. This means that the randomly initialised models in the new optimisation phase are always poorer than the best models of the previous phase, but the reduction in the size of the feature space means that the

performance of the model quickly recovers and matches or improves on earlier performance.

As shown in Figure 1, the optimisation phase and winnowing procedure are repeated until convergence is achieved or a specific number of iterations have occurred. The best model found at any point in the entire optimisation procedure should be chosen as the final best model. An implementation of WAAC in R [31] is available from the authors on request.

Dataset

We use the Karthikeyan dataset [32] of melting point values as described in Nigsch et al. [33]. This is a dataset of melting points of 4119 diverse organic molecules which cover a range of melting points from 14 to 392.5°C, with a mean of 167.3°C and a standard deviation of 66.4°C. Each molecule is described by 203 2D and 3D descriptors, which is the full range of descriptors available in the software MOE 2004.03 [5].

The dataset was randomly divided 2:1 into training data and an external test set (1373 molecules, see additional file 1: externaltest.csv for the original data). The training data was further randomly divided 2:1 into a training set used for model building (1831 molecules, see additional file 2: internaltraining.csv) and an internal test set (915 molecules, see additional file 3: internaltest.csv).

Objective function

The goal of the WAAC algorithm is to find the feature subset and parameter values that will give the best predictive accuracy for a model based on given training data. During the course of the optimisation, the algorithm needs to be guided by an objective function that will give an estimate of the predictive accuracy of a particular model.

Here we examine the performance of the WAAC algorithm on the Karthikeyan dataset using as our objective function the root mean squared error of the predictions on the internal test set, RMSE(int). Each model is built on the training set using whatever features and parameter values have been selected, and then used to predict the melting point values for the internal test set.

Statistical testing

To assess the quality of a model, we report three statistics: the squared correlation coefficient, R^2, the Root-Mean-Square-Error, RMSE, and the bias. These are defined in Equations 1 to 3. A parenthesis nomenclature is used to indicate whether the statistic refers to a model tested on the entire training data (tr) (this includes the internal test set), the internal test set only (int), or the external test set (ext).

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(y_i^{obs} - y_i^{pred}\right)^2} \quad (1)$$

$$R^2 = 1 - \sum_{i=1}^{n}\left(y_i^{obs} - y_i^{pred}\right)^2 / \sum_{i=1}^{n}\left(y_i^{obs} - \overline{y}^{obs}\right)^2 \quad (2)$$

$$bias = \frac{1}{n}\sum_{i=1}^{n}\left(y_i^{obs} - y_i^{pred}\right) \quad (3)$$

In the prediction of the external test set, an outlier is defined as any point with a residual greater than 4 standard deviations from the mean.

Models

We used the WAAC algorithm to simultaneously optimise the chosen features and number of components in a Partial Least Squares (PLS) model. The *plsr* method in the *pls* package in R [31] was used to build the PLS model. Scaling was set to true. A range of 20 allowed parameter values for the number of components in the model was initially set to cover from 1 to 191 inclusive in steps of 10. After each winnowing, the step size was reset so that the maximum value for the number of components was less than the number of remaining descriptors. For the WAAC algorithm itself, a colony of 50 ants was used, and the algorithm was run for 800 iterations with winnowing every 100 iterations. For comparison, the algorithm was run for the same length without any winnowing.

In addition, we used the WAAC algorithm to optimise a Support Vector Machine (SVM) model. The *svm* method in the *e1071* package in R [31] was used to perform ε-regression with a radial basis function. A range of allowed parameter values for the SVM were chosen based on a preliminary run: values for C from 1 to 31 inclusive in steps of 2, and values of ε from 0.01 to 1.61 inclusive in steps of 0.1. Since two parameters needed to be optimised for this model, the length of each optimisation phase in the WAAC algorithm was extended to 150 iterations and the algorithm was run for 1500 iterations in total.

To compare to other feature selection methods, we used the training data to build a Random Forest model [34] using the *randomForest* package in R (using the default settings of *mtry* = *N*/3, *ntree* = 500, *nodesize* = 5). We also compared to the best of thirteen *k* Nearest Neighbours (*k*NN) models trained on the training set, where *k* was 1, 5, 10 or 15. For the models based on multiple neighbours, separate models were created where the predictions were combined using exponential, geometric, arithmetic, or inverse distance weighting (for more details, see Nigsch et

al. [33]). The best performing model, as measured by leave-one-out cross validation on the training data, was the 15 NN model with exponential weighting. Hereafter, this model is referred to as the *k*NN model.

Genetic algorithm

For comparison with the WAAC algorithm, a genetic algorithm for feature selection was implemented in the R statistical programming environment [31]. 50 chromosomes were randomly initialised so that each chromosome on average corresponded to a model based on half of the descriptors. A selection operator chose 10 chromosomes using tournament selection with tournaments of size 3. Once selected, that chromosome was removed from the pool for further selection. A crossover operator was applied to the selected chromosomes, as a single-point crossover between randomly selected (with replacement) chromosomes yielding a pair of children in each case. Each child was subject to a mutation operator which, for a given bit on a chromosome, had a probability of 0.04 of flipping it. The process of crossover and mutation was repeated until 50 offspring were created. The next generation was then formed by the 25 best chromosomes in the original population along with the best 25 of the offspring.

Results

The WAAC algorithm was used to search parameter and feature space for a predictive SVM model for the Karthikeyan dataset for both a PLS model and an SVM

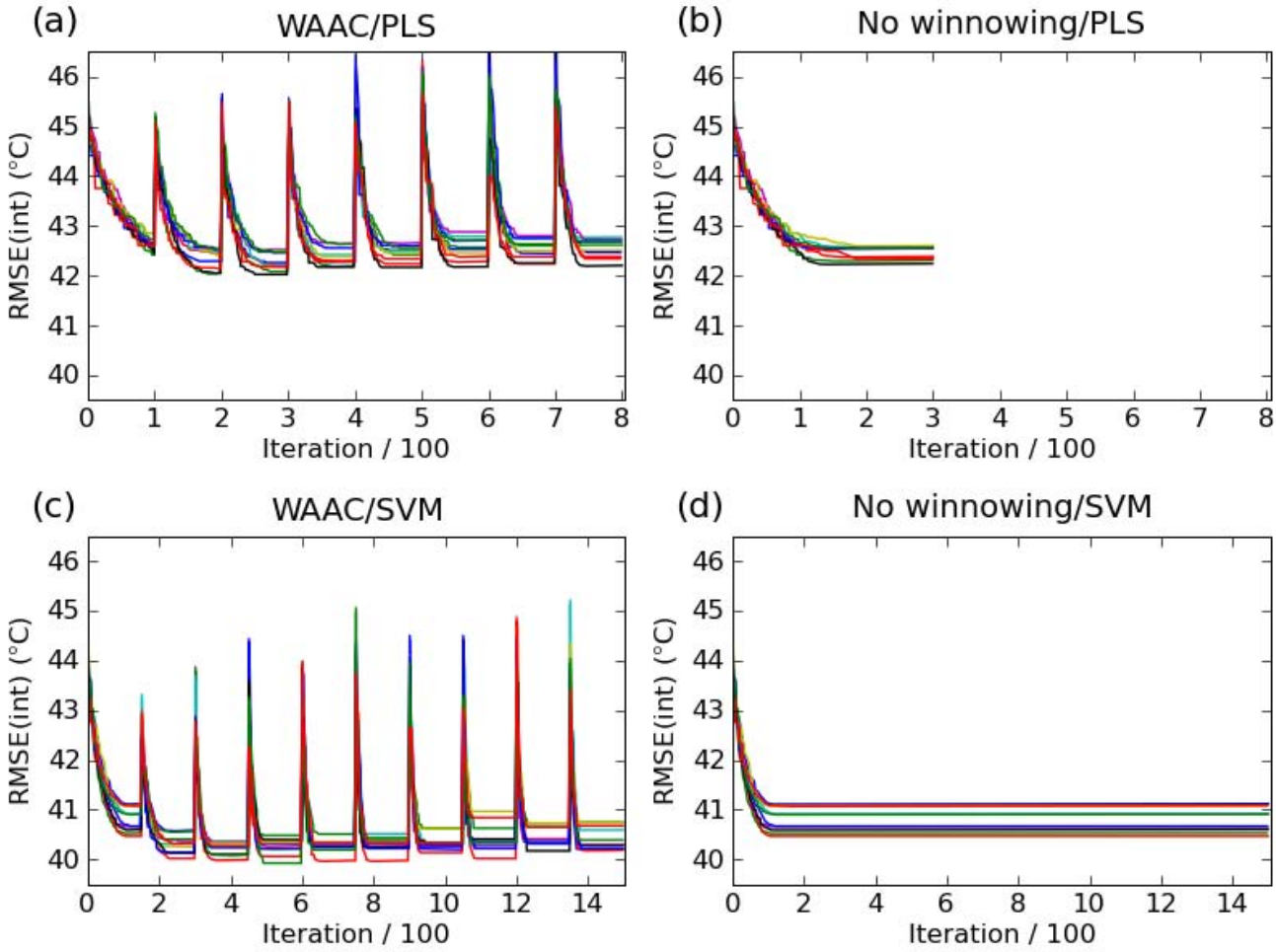

Figure 2

Value of the objective function for the best model at each iteration of the WAAC algorithm for the PLS model (top) and the SVM model (bottom). The figures on the right, (b) and (d), show the effect of having a single optimisation phase without any winnowing. Ten repetitions of the algorithm are shown, with corresponding repetitions starting from the same initial random seed.

model. Figures 2(a) and 2(c) show the progress of the algorithm for the PLS and SVM models respectively, as measured by the value of the objective function for the best model found so far in a particular optimisation phase. Each experiment was performed 10 times with different random seeds. For each repetition, the model with the lowest value of the objective function was chosen from among the best models found in each optimisation phase. Of these ten models, the one with the fewest descriptors was chosen as the single final model. This reduces the possibility of finding by chance a model which had an optimal value of the objective function but poor predictive ability.

The selected models for WAAC/PLS and WAAC/SVM are shown in Table 1. Of the 203 original descriptors, only 68 were selected for the PLS model, and 28 for the SVM model. The final models were evaluated by training on the entire training data of 2746 molecules, and predicting the melting point value of the external test set. The results are shown in Figure 3 and summarised in Table 2. The summary statistics for the PLS model are: for the training set, RMSE(tr) = 44.4°C, R^2(tr) = 0.52, bias = -0.0°C; for the test set, RMSE(ext) = 46.6°C, R^2(ext) = 0.51, bias = -0.74°C. For comparison, the value of the objective function RMSE(int) was 42.8°C. There was a single outlier, *mol4161* (Figure 4). The summary statistics for the SVM model are: for the training set, RMSE(tr) = 30.7°C, R^2(tr) = 0.77, bias = -1.6°C; for the test set, RMSE(ext) = 45.1°C, R^2(ext) = 0.54, bias = -2.1°C. The value of the objective function RMSE(int) was 40.2°C. Three molecules were identified as outliers to the model: *mol41*, *mol4161* and *mol4195*. These are drawn as filled circles in Figure 3, and their structures are shown in Figure 4.

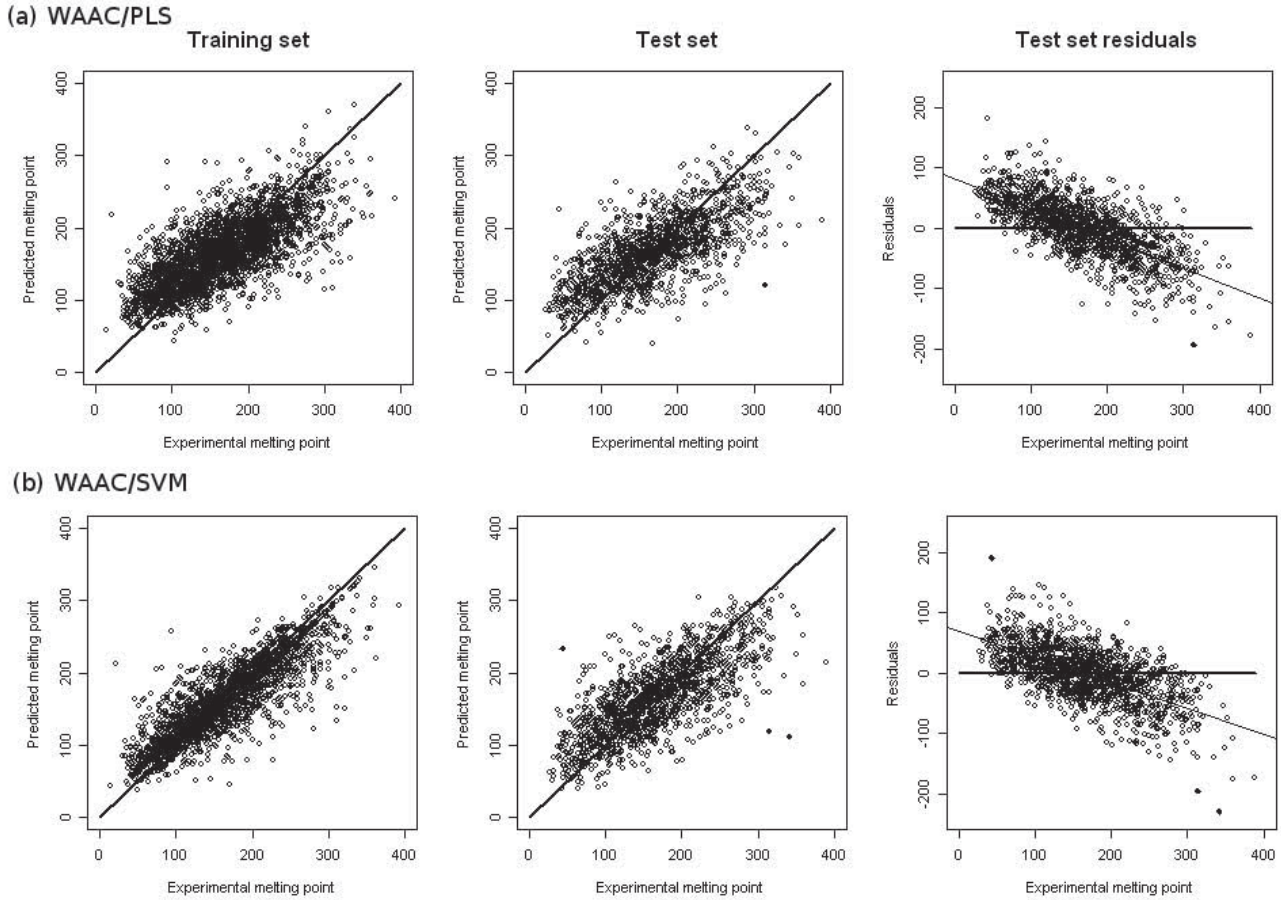

Figure 3
Performance of models developed with WAAC: (a) a PLS model and (b) an SVM model. The first two columns contain predictions for the training set and test set, respectively. The line $x = y$ is shown for comparison. The column on the right shows the residuals from the test set prediction along with a line of best fit (light line); for comparison, the line $x = 0$ is shown (heavy line). Outliers are shown as filled circles in the test set prediction and residuals plots. All values in °C.

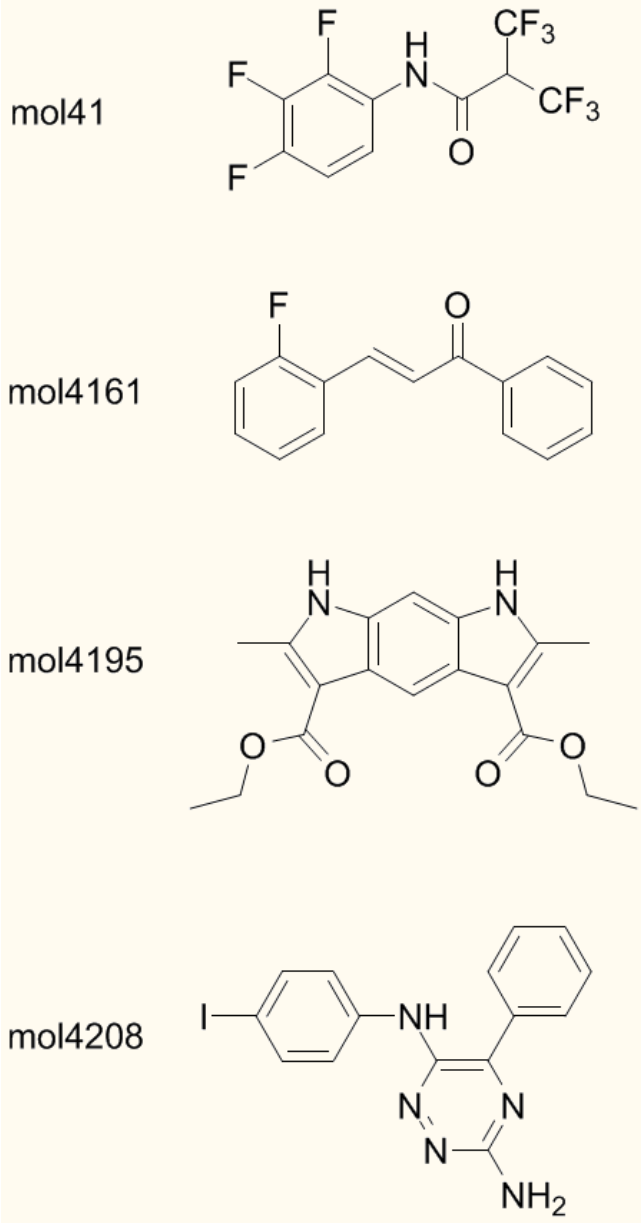

Figure 4
Structures of outliers for the models discussed in the text. An outlier is defined as any molecule with a residual greater than four standard deviations from the mean. Molecules 41, 4161 and 4195 are outliers for the WAAC/SVM model; molecules 4161 and 4208 are outliers for both the RF and *k*NN models; molecule 4161 is the single outlier to the WAAC/PLS model.

For the PLS model the optimised number of components was 49. In order to assess whether the WAAC algorithm sufficiently explored parameter space, we carried out a parameter scan across all allowed values for the parameter with the feature selection found in the best model, and calculated the value of the objective function, RMSE(int). As shown in Figure 5 (solid line), the value of the objective function obtained with 49 components is almost at the minimum, although three larger values for the number of components give slightly better models (42.78°C RMSE(int) versus 42.73°C). For the SVM model, the optimised parameter values associated with the selected model were a cost value of 5, and a value for ε of 0.21. When we carried out a parameter scan across all allowed values of the cost and ε (272 models in total), only one scored higher than the best model, and even then, only marginally: 40.22°C RMSE(int) for cost = 5 and ε = 0.11, versus 40.23°C for the best model.

Figure 2(a) shows the value of the objective function for the best PLS model at each iteration for the WAAC algorithm compared to a single optimisation phase without any winnowing, Figure 2(b). The same random seeds are used for corresponding repetitions of the experiments, to ensure that the effect observed is not due to different initial models. In the absence of winnowing, premature convergence occurs and poorer solutions are found. This is also the case for the best SVM model shown in Figure 2(c) and 2(d).

The Random Forest (RF) and *k*NN models for the same data are shown in Figure 6 and Table 1. Although performance on the training set does not give any indication of predictive ability, it is interesting to note how the different models have completely different RMSE(tr) and R^2(tr). Performance on the external test set, which was not used to derive any of the models, allows us to assess predictive ability. On the basis of RMSE(ext), the RF model (44.5°C) is as good as, or slightly better than, the WAAC/SVM model (45.1°C), followed by the WAAC/PLS model (46.6°C) and then the *k*NN model (48.3°C). A similar order of predictive ability is shown by R^2(ext), (RF: 0.55, WAAC/SVM: 0.54, WAAC/PLS: 0.51, *k*NN: 0.47). The bias shows a slightly different order for the two WAAC-derived models (RF: -0.4°C, WAAC/PLS: -0.7°C, WAAC/SVM: -2.1°C, *k*NN: -4.1°C).

However, looking at the test set predictions in the second column of Figures 3 and 6 it is clear, particularly for the RF model, that a systematic error occurs at the extremes of the melting point values in the dataset: low values are systematically overpredicted, while high values are underpredicted. In order to quantify the extent of this problem, we plotted the test set residuals versus the experimental melting point, and used linear regression to find the line of best fit (shown in the third column in Figures 3 and 6). For a model without this type of predictive bias, the expected slope is 0. The WAAC/SVM model performs best with a slope of -0.43, followed by the *k*NN and WAAC/PLS models which both have slopes of -0.49, while the RF

Table 1: Description of the best models found by the WAAC algorithm

	WAAC/PLS	WAAC/SVM
Number of descriptors	68	28
2D descriptors	petitjean, weinerPath, weinerPol, a_ICM, b_1rotR, chi0_C, chi1, reactive, a_heavy, a_nH, a_nF, a_nO, a_nS, VadjEq, VadjMa, balabanJ, PEOE_RPC+, PEOE_VSA+3, PEOE_VSA+4, PEOE_VSA+5, PEOE_VSA+6, PEOE_VSA-1, PEOE_VSA-4, PEOE_VSA_FPNEG, PEOE_VSA_PPOS, PC+, PC-, Q_PC+, Q_RPC+, Q_VSA_FHYD, Q_VSA_FNEG, Q_VSA_FPNEG, Q_VSA_FPOL, Q_VSA_FPOS, Q_VSA_FPPOS, Q_VSA_PNEG, Q_VSA_PPOS, Kier1, Kier3, KierA1, KierA2, apol, vsa_acc, SlogP_VSA3, SlogP_VSA5, SMR_VSA3, SMR_VSA5, TPSA	radius, weinerPol, b_1rotR, b_rotR, chi1v_c, a_nO, a_nP, balabanJ, PEOE_VSA+2, PEOE_VSA+3, PEOE_VSA-1, PEOE_VSA-5, PEOE_VSA-6, Q_RPC+, SlogP_VSA1, SlogP_VSA4, SlogP_VSA9, SMR_VSA2, SMR_VSA4, SMR_VSA6, TPSA
3D descriptors	AM1_dipole, AM1_Eele, E_sol, E_strain, E_tor, MNDO_HF, MNDO_dipole, MNDO_E, dipole, PM3_HF, ASA-, ASA_H, CASA-, FASA_H, FASA_P, VSA, glob, std_dim1, std_dim3, vol	E_oop, E_strain, E_vdw, PM3_LUMO, FASA_P, FCASA+, rgyr
Parameters	components = 49	Cost = 5, ε = 0.21

model has a slope of -0.53. The standard errors of all of these values are 0.01.

Another effect of this systematic error is that the predicted values are bunched closer around the mean than the experimental values. The mean and standard deviation of the experimental values in the test set are 167.3°C and 66.4°C, respectively. All of the model predictions have a similar mean: 166.5, 165.2, 167.0 and 163.2°C for the WAAC/PLS, WAAC/SVM, RF and *k*NN models respectively. However, for the RF model the standard deviation of the predicted values is much smaller than that of the other models: 47.1, 51.6, 41.0 and 49.5°C for the WAAC/PLS, WAAC/SVM, RF and *k*NN models respectively.

Another widely used stochastic method for feature selection is a genetic algorithm (GA). Hasegawa et al. [35] were one of the first to use a GA in combination with a PLS

Table 2: Summary statistics for the models discussed in the text

	WAAC/PLS	WAAC/SVM	SVM	*k*NN	Random Forest
Training set					
RMSE (°C)	44.4	30.7	36.2	47.6	17.8 (44.7)*
R^2	0.52	0.77	0.68	0.44	0.92 (0.51)*
bias (°C)	0.0	-1.6	-2.3	-3.4	0.0
Test set					
RMSE (°C)	46.6	45.1	43.9	48.3	44.5
R^2	0.51	0.54	0.56	0.47	0.55
bias (°C)	-0.7	-2.1	-2.3	-4.1	-0.4
mean (°C)	166.5	165.2	165.0	163.2	167.0
standard deviation (°C)	47.1	51.6	49.3	49.5	41.0
Line of best fit through test set residuals					
Slope	-0.49	-0.43	-0.44	-0.49	-0.53

* Out-of-bag estimates for RMSE and R^2 are shown in parenthesis.

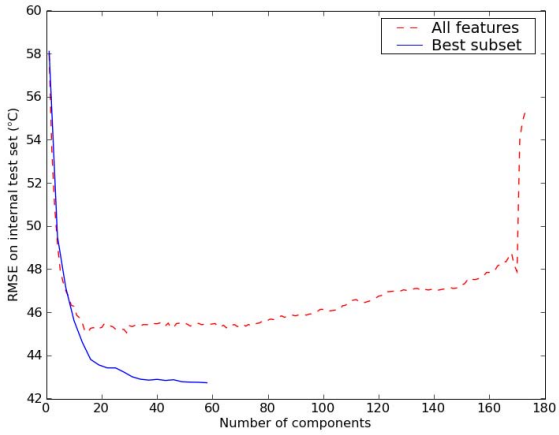

Figure 5
The effect of the number of components on the predictive ability of a PLS model. The red dashed line is a model based on all of the features, whereas the model represented by the blue solid line is based only on the subset selected by the WAAC algorithm. The best subset line ends at 59 components, as there are only 59 features in this subset. The line for all features is truncated at 174 components as the RMSE rapidly increases after this point.

model to perform feature selection. The performance of the GA for feature selection is shown in Figure 7 compared to the WAAC algorithm. For both algorithms, the number of PLS components was fixed at 49. Convergence is much slower for the GA algorithm. In addition, the model with the fewest number of descriptors from 10 repetitions of each algorithm had 95 descriptors in the case of GA/PLS (objective function of 42.6 °C) but only 57 for WAAC/PLS (objective function value of 42.3 °C).

Discussion

The development of the WAAC algorithm arose from an attempt to overcome the limitations of the modified ACO and ANTSELECT algorithms. Both of these algorithms determine probabilities by summing weights based on fitness scores. However, we observed that as convergence is achieved the fitness scores of the ant models in a particular iteration differ very little from each other. Thus, WAAC uses the fraction of the number of ants that have chosen a particular descriptor rather than a function of the fitness of the ants that have chosen that feature. Another problem with the use of weights is that they increase monotonically over the course of the algorithm whereas the sum of the number of ants has a clear bound. In addition, WAAC uses a value for ρ of 1, that is, complete evaporation. Values less than 1 were found to delay convergence without any corresponding improvement in the result. This makes sense when we consider that the evaporation parameter is supposed to help strike a balance between exploitation of information on previous models (global search) and exploration of local feature space (local search). However, this aspect is already included in Shen et al.'s algorithm and WAAC by the influence of the best models (global search) and current models (local search) on the moving probabilities. As a result of this simpler approach, the moving probabilities now have a meaningful interpretation: the probability of choosing a particular descriptor in the next iteration is equal to the average of the fraction of ants that have chosen that descriptor in their current model and the fraction of ants that have chosen it in their best model.

Since the WAAC algorithm requires a range of allowed parameter values for the model, it is generally worthwhile to do an exploratory run of the algorithm to determine reasonable values. In addition, it is important that the number of allowed values for each parameter is less than the number of ants (preferably much less) to ensure that the parameter space is adequately sampled. An appropriate size for the ant population depends on the number of descriptors and the extent of the interaction between them. Model space will be better sampled if more ants are used, but the calculation time will also increase. However, since the feature-selection space is of size 2^n-1, where n is the number of descriptors, the exact number of ants is not expected to affect the ability of the algorithm to find solutions. An ant population of between 50 and 100 ants is recommended. For the WAAC/PLS study, the relationship between the population size and the best value of the objective function is shown in Figure 8; there is little improvement beyond 50 ants. The length of the optimisation phase should be sufficient to allow the objective function to start to converge to an optimum value. It is not necessary to allow the optimisation phase to proceed much further, as after this point the descriptors chosen in the best models reinforce themselves and broad sampling of the search space no longer occurs. The winnowing procedure and subsequent reinitialisation on a smaller search space is a more effective way of finding the optimum model.

In the past, the development and comparison of feature selection methods for QSAR have involved the use of a standard dataset first reported in 1990, the Selwood dataset [36] of the activity of 31 antifilarial antimycin analogues, whose structures are represented by 53 calculated physicochemical descriptors. However, comparisons between different algorithms have been hampered by the fact that many of the descriptors are highly-correlated, and in addition, a true test using an external test set is not feasible due to the small number of samples. Advances in computing power mean that it is no longer appropriate to use such a small dataset for the purposes of testing feature

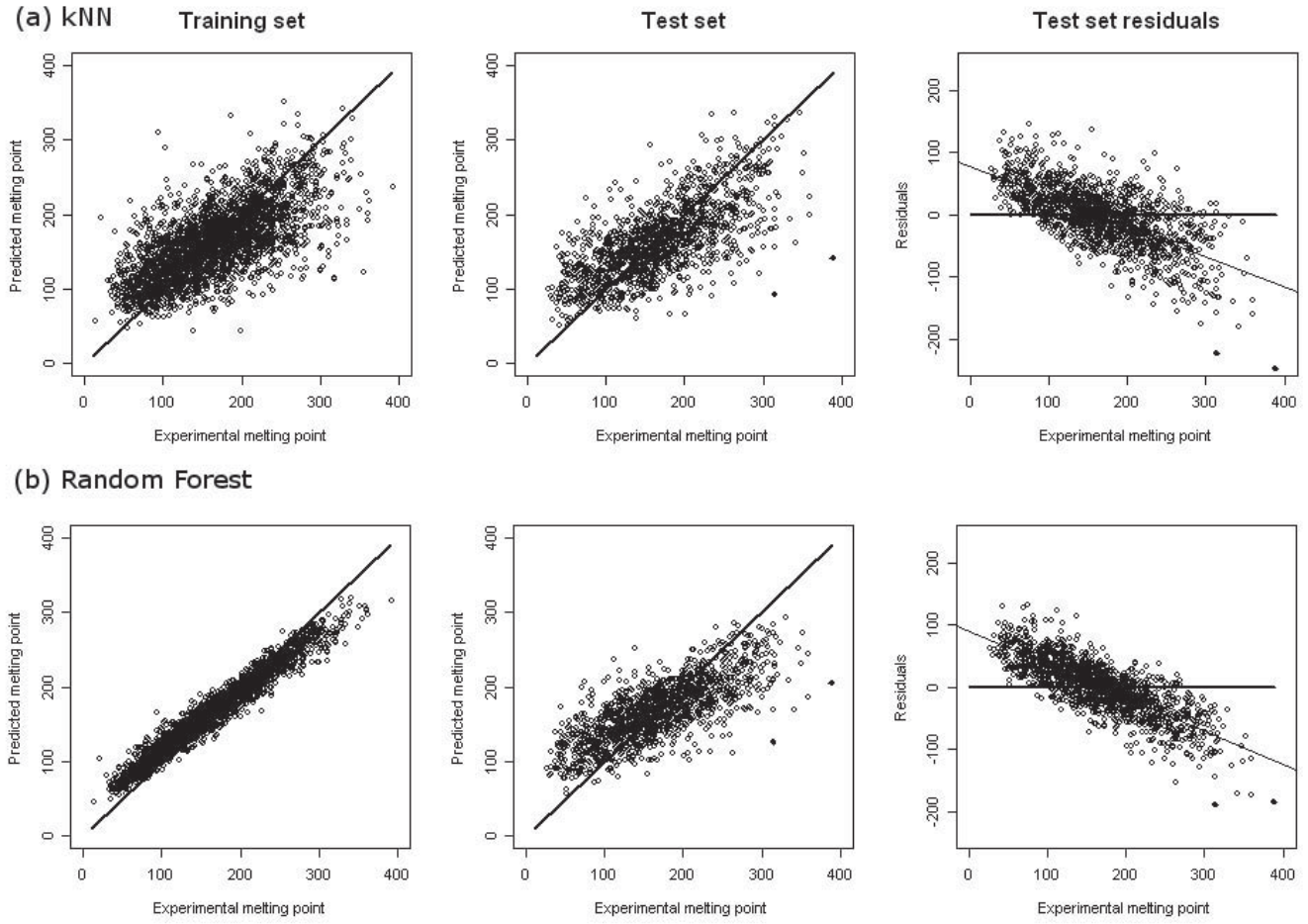

Figure 6

Performance of (a) a *k*NN model, and (b) a Random Forest model. The first two columns contain predictions for the training set and test set, respectively. The line $x = y$ is shown for comparison. The column on the right shows the residuals from the test set prediction along with a line of best fit (light line); for comparison, the line $x = 0$ is shown (heavy line). Outliers are shown as filled circles in the test set prediction and residuals columns. All values in °C.

selection algorithms. The Karthikeyan dataset used here is much more representative of the feature selection problems that occur in modern QSAR and QSPR studies.

PLS models are prone to overfitting. Figure 5 shows a comparison between a PLS model that uses the best subset (as selected by WAAC) and one using all of the descriptors. It is clear that the development of a predictive PLS model requires a variable selection step. Even if the number of components is optimised, performance is significantly poorer if all features are used instead of just the subset selected by the WAAC algorithm. It is also worth noting that PLS is a linear method, whereas SVM is a non-linear method. If the underlying link between descriptor values and the melting point cannot be adequately described by a linear combination of descriptor values, then the performance of the PLS method is likely to suffer. This may explain why, despite containing fewer than half the number of descriptors, the SVM model performed better than the PLS model.

Although the WAAC algorithm is capable of simultaneously optimising the feature selection as well as the parameter values, in some instances it may be preferable to use the WAAC algorithm simply for feature selection and optimise the parameter values separately for each model. This will only be computationally feasible where the model has a small number of parameters which need to be optimised and where the parameter optimisation can be efficiently carried out. For example, the optimal number of components for a PLS model could be determined by internal cross validation. When compared to

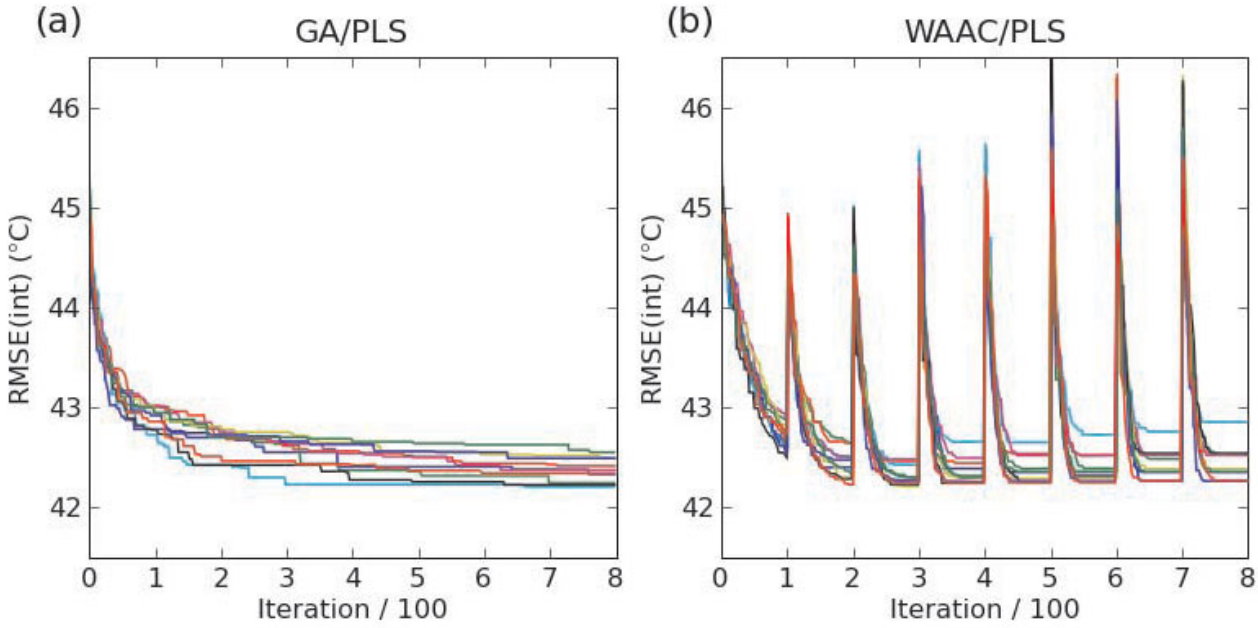

Figure 7
Value of the objective function for the best PLS model at each iteration of (a) a genetic algorithm and (b) the WAAC algorithm. Ten repetitions of each algorithm are shown. The number of PLS components was set to 49.

the use of a genetic algorithm for optimising the feature selection of a PLS model, the WAAC algorithm performs well, both in terms of faster convergence and in its ability to produce models with fewer descriptors. It should be noted, however, that genetic algorithms have many different implementations as well as several parameters. This result, on a single dataset, cannot therefore be seen as conclusive.

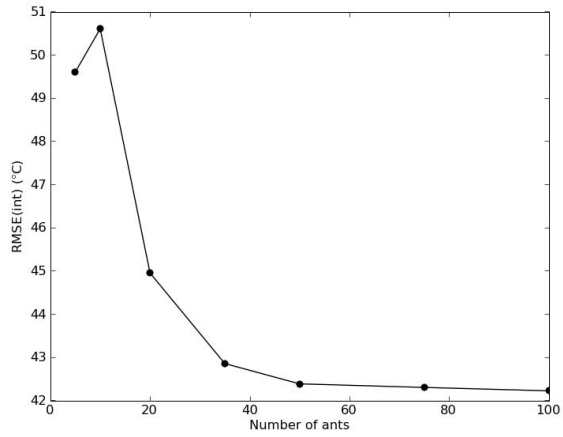

Figure 8
Relationship between the population size and the minimum value of the objective function for the WAAC/PLS model. The value of the objective function is the minimum found from ten repetitions of the algorithm.

In comparison to PLS models, the inclusion of a large number of descriptors does not necessarily lead to overfitting for SVM models. Although both Guyon et al. [14] and Fröhlich et al. [15], for example, have developed descriptor selection methods for SVM, an SVM model built on the entire set of descriptors and using the optimized parameters from the WAAC algorithm actually performs slightly better on the external test set. Here, the main effect of the WAAC algorithm is the identification of a minimum subset of descriptors which are the most important for the development of a predictive model. Such a procedure is especially useful when the descriptor values are derived from experimental measurement or require expensive calculation (for example, those derived from QM calculations). It also aids interpretability of the results.

Of the 28 descriptors selected by the WAAC/SVM model, three-quarters are 2D descriptors. Of these, many involve the area of the van der Waals surface associated with particular property values. For example the PEOE_VSA+2 descriptor is the van der Waals surface area (VSA) associated with PEOE (Partial Equalisation of Orbital Electronegativity) charges in the range 0.10 to 0.15. Also selected

were descriptors relating to hydrophobic patches on the VSA (SlogP_VSA1, for example), the contribution to molar refractivity (SMR_VSA2, for example) which is related to polarisability, and the polar surface area (TPSA). Since the intermolecular interactions in a crystal lattice are dependent on complementarity between the properties of the VSA of adjacent molecules, the selection of these descriptors seems reasonable. Two descriptors were selected relating to the number of rotatable bonds (b_1rotR and b_rotR). These properties are related to the melting point through their effect on the change in entropy (ΔS_{fus}) associated with the transformation to the solid state. Hydrogen bonds make an important energetic contribution to the formation of the crystal structure. This probably explains the selection of the descriptor for the number of oxygen atoms (a_nO), although strangely the number of nitrogen atoms is not included (it was however included in five out of the ten ant models). Four descriptors were selected by all ten ant models: b_1rotR, SlogP_VSA1, PEOE_VSA-6 and balabanJ. Balaban's *J* index is a topological index that increases in value as a molecule becomes more branched [37]. It seems possible that increased branching makes packing more difficult, and leads to lower melting points.

The WAAC algorithm appears to be robust to the presence of highly correlated descriptors. Despite the fact that such descriptors were not filtered from the dataset, the selected WAAV/SVM model contains only two pairs of descriptors with an absolute Pearson correlation coefficient greater than 0.8: b_rotR/b_1rotR (0.97) and SMR_VSA2/PEOE_VSA-5 (0.81). If the WAAC algorithm were unable to filter highly correlated descriptors, we would expect to see many more correlations as 16 of the chosen descriptors were highly correlated (absolute value greater than 0.8) with at least one descriptor not included in the final model. For example, radius has a correlation of 0.86 with respect to diameter (not unexpectedly). weinerPol is highly correlated with 35 other descriptors, none of which were chosen in the final model. PM3_LUMO is correlated with both AM1_LUMO (0.97) and MNDO_LUMO (0.96), but neither of other two appear.

For a small number of molecules, our models make very poor predictions. This may either be due to a lack of sufficient training molecules with particular characteristics, or it may be due to a fundamental deficiency in the information used to build the models. For example, for the WAAC/SVM models, three outliers can be detected whose residuals are more than four standard deviations from the mean (Figures 3 and 4). A polyfluorinated amide, *mol41*, is predicted to have a melting point of 233°C although its experimental melting point is 44°C. The melting points of the other two outliers were both underestimated: *mol4161*, m.p. 314.5°C but predicted 119°C, and *mol4195*, m.p. 342°C, but predicted 111°C. Both of these molecules have extended conjugated structures, causing the molecule to be planar over a wide area, and which are likely to give rise to extensive π-π stacking in the solid state. As a result, they are conformationally less flexible than might be expected from the number of rotatable bonds. *mol4161* is also an outlier to the other three models; for WAAC/PLS it is the only outlier, whereas the RF and *k*NN predictions have a second outlier, *mol4208* (Figure 4).

The WAAC algorithm described here is particularly useful when a machine learning method is prone to overfitting if presented with a large number of descriptors, such as is the case with PLS. However, not all machine learning methods require a prior feature selection procedure. The Random Forest (RF) method of Breiman uses consensus prediction of multiple decision trees built with subsets of the data and descriptors to avoid overfitting. For comparison with the WAAC results, we predicted the melting point values for the external test data using an RF model built on the training data. We also compared to a 15 Nearest Neighbour model (*k*NN) where the predictions of the set of neighbours were combined using an exponential weighting. In our comparison, the RMSE(ext) and R^2(ext) show that the RF and WAAC/SVM models are very similar, and are better than the WAAC/PLS and *k*NN models. However, analysis of the residuals shows that the RF is more prone to bias at high and low values of the melting point compared to the other models.

A predictive bias was observed for all models at the extremes of the range of melting points. A similar effect was observed by Nigsch et al. for a kNN model of melting point prediction [33]. The effect was attributed to the fact that the density of points in the training set is less at the extremes of the range of melting point values. This means that the nearest neighbours to a point near the extreme are more likely to have melting points closer to the mean. This effect is most pronounced for the RF model, and the explanation may be similar.

In this study the WAAC algorithm was guided using the RMSE of prediction for an internal test set, RMSE(int). The choice of which objective function to use should be considered carefully. If an objective function is chosen which does not explicitly penalise the number of descriptors but only does so implicitly (for example, RMSE(int)), irrelevant descriptors may accumulate in the converged model. When using such an objective function, the winnowing procedure implemented in WAAC plays an important role in removing these descriptors after the optimisation phase by initiating a new search of a reduced feature space which makes it less likely that irrelevant descriptors will be selected. This effect is shown in Figure 2(b) and 2(d),

where poorer models were found when the WAAC feature selection and parameter optimisation procedure was applied without winnowing.

An alternative type of objective function is one that explicitly penalises the number of descriptors. Such functions typically contain a cost term which is adjusted based on some *a priori* knowledge of the number of descriptors desired in the model. For example, the modified ACO algorithm of Shen *et al.* [26] was guided by a fitness function with two terms, one relating to the number of descriptors and the other to the fit of the model to the training set. Objective functions such as this quickly force models into a reduced feature space by favouring models with fewer descriptors. However, the moving probabilities used to choose descriptors will be misleading as they will largely be based on those descriptors present in models with fewer descriptors rather than those with the best predictive ability. As a result, descriptors with good predictive ability may be removed by chance. It should be noted that an objective function that simply optimises a measure of fit to the training data is not a suitable choice for the development of a model with predictive ability. Optimising the RMSE on the entire training data, RMSE(tr), or optimising the R^2(tr) value, will produce an overfitted model that fits the training data exceptionally well but performs poorly on unseen data.

Near the end of each optimisation phase, the majority of ants converge to the same feature selection and parameter values, causing the same model to be repeatedly evaluated. It should be possible to gain a significant speedup if instead of re-evaluating a model, a cached value were used. Caching could be simply done by storing the objective function and models for all of the ants from the last few iterations. This is especially important if an objective function is used whose value varies on re-evaluation as is the case, for example, with the RMSE from n-fold cross-validation, RMSE(cv). Since for each ant the best score is retained, the value of the objective function will tend towards the optimistic tail of the distribution of values of the RMSE(cv). However, it should not have a major effect on the results of the feature selection and parameter optimisation, as model re-evaluation generally occurs only once the majority of the ants' models have already converged.

Conclusion

The key elements to developing an effective QSPR model for prediction are accurate data, relevant descriptors and an appropriate model. Where there is no *a priori* information available on relevant descriptors, some form of feature selection needs to be performed.

We have presented WAAC, an extension of the modified ACO algorithm of Shen et al. [26], which can perform simultaneous optimisation of feature selection and model parameters. In addition, the moving probabilities used by the algorithm are easily interpreted in terms of the best and current models of the ants, and our winnowing procedure promotes the removal of irrelevant descriptors.

We have shown that the WAAC algorithm can be used to simultaneously optimise parameter values and the selected features for PLS and SVM models for melting point prediction. In particular, the resulting SVM model based on 28 descriptors performed as well as a Random Forest model that used the entire set of 203 descriptors.

Authors' contributions

NMOB conceived and developed the WAAC algorithm, applied it to the melting point dataset, analysed the results and drafted the manuscript. DSP was involved in the interpretation of the results, revising the manuscript and carried out the Random Forest calculations. FN implemented the kNN model. JBOM contributed to the analysis of data and revising the manuscript. All authors read and approved the final manuscript.

Additional material

Additional file 1

***The external test set**. The models were evaluated by testing on this external test set.*
Click here for file
[http://www.biomedcentral.com/content/supplementary/1752-153X-2-21-S1.csv]

Additional file 2

***The internal training set**. The WAAC feature selection algorithm was trained on this.*
Click here for file
[http://www.biomedcentral.com/content/supplementary/1752-153X-2-21-S2.csv]

Additional file 3

***The internal test set**. The objective function used to guide the WAAC feature selection algorithm was calculated using this internal test set.*
Click here for file
[http://www.biomedcentral.com/content/supplementary/1752-153X-2-21-S3.csv]

Acknowledgements

We thank the BBSRC (NMOB and JBOM – grant BB/C51320X/1), Pfizer (DSP and JBOM – through the Pfizer Institute for Pharmaceutical Materials Science), and Unilever for funding FN and JBOM and for supporting the Centre for Molecular Science Informatics. NMOB thanks Dr. Jen Ryder, Dr. Daniel Almonacid and Dr. Avril Coghlan for helpful comments on the manuscript.

References

1. Hansch C, Maloney PP, Fujita T, Muir RM: **Correlation of biological activity of phenoxyacetic acids with Hammett substituent constants and partition coefficients.** *Nature* 1962, **194:**178-180.
2. Hawkins DM: **The problem of overfitting.** *J Chem Inf Comput Sci* 2004, **44:**1-12.
3. Guyon I, Elisseeff A: **An introduction to variable and feature selection.** *J Mach Learn Res* 2003, **3:**1157-1182.
4. Steinbeck C, Han Y, Kuhn S, Horlacher O, Luttmann E, Willighagen E: **The Chemistry Development Kit (CDK): An Open-Source Java Library for Chemo- and Bioinformatics.** *J Chem Inf Comput Sci* 2003, **43:**493-500.
5. *MOE (Molecular Operating Environment), v2004.03* [http://www.chemcomp.com]. Chemical Computing Group Inc., Montreal, Quebec, Canada
6. 2006 [http://www.tripos.com]. SYBYL 7.1. Tripos Inc., 1699 Hanley Road, St. Louis, MO 63144
7. John GH, Kohavi R, Pfleger K: **Irrelevant features and the subset selection problem.** In *Machine learning, Proceedings of the Eleventh International Conference: 10–13 July 1994; Amherst* Edited by: Cohen WW, Hirsh H. Morgan Kaufmann; 1994:121-129.
8. Kohavi R, John GH: **Wrappers for feature subset selection.** *Artif Intell* 1997, **97:**273-324.
9. Dudek AZ, Arodz T, Gálvez J: **Computational methods in developing quantitative structure-activity relationships (QSAR): a review.** *Comb Chem High Through Screen* 2006, **9:**213-228.
10. Liu Y: **A comparative study on feature selection methods for drug discovery.** *J Chem Inf Comput Sci* 2004, **44:**1823-1828.
11. Whitney AW: **A direct method of nonparametric measurement selection.** *IEEE Trans Comput* 1971, **20:**1100-1103.
12. Marill T, Green DM: **On the effectiveness of receptors in recognition systems.** *IEEE Trans Inform Theory* 1963, **9:**11-17.
13. Pudil P, Novovièová J, Kittler J: **Floating search methods in feature selection.** *Patt Recog Lett* 1994, **15:**1119-1125.
14. Guyon I, Weston J, Barnhill S, Vapnik V: **Gene selection for cancer classification using support vector machines.** *Mach Learn* 2002, **46:**389-422.
15. Fröhlich H, Wegner JK, Zell A: **Towards optimal descriptor subset selection with support vector machines in classification and regression.** *QSAR Comb Sci* 2004, **23:**311-318.
16. Goldberg DE: *Genetic Algorithms in Search, Optimization and Machine Learning* Boston: Kluwer Academic Publishers; 1989.
17. Rogers D, Hopfinger AJ: **Application of genetic function approximation to quantitative structure-activity relationships and quantitative structure-property relationships.** *J Chem Inf Comput Sci* 1994, **34:**854-866.
18. Wegner JK, Zell A: **Prediction of aqueous solubility and partition coefficient optimized by a genetic algorithm based descriptor selection method.** *J Chem Inf Comput Sci* 2003, **43:**1077-1084.
19. von Homeyer A: **Evolutionary Algorithms and their Applications in Chemistry.** In *Handbook of Chemoinformatics Volume 3*. Edited by: Gasteiger J. Weinheim: Wiley-VCH; 2003:1239-1280.
20. Agrafiotis DK, Cedeno W: **Feature selection for structure-activity correlation using binary particle swarms.** *J Med Chem* 2002, **45:**1098-1107.
21. Lin WQ, Jiang JH, Shen Q, Shen GL, Yu RQ: **Optimized block-wise variable combination by particle swarm optimization for partial least squares modeling in quantitative structure-activity relationship studies.** *J Chem Inf Model* 2005, **45:**486-493.
22. Guha R, Jurs PC: **Development of linear, ensemble and nonlinear models for the prediction and interpretation of the biological activity of a set of PDGFR inhibitors.** *J Chem Inf Comput Sci* 2004, **44:**2179-2189.
23. Vapnik VN: *The nature of statistical learning theory* New York: Springer Verlag; 1995.
24. Hastie T, Tibshirani R, Friedman J: *The elements of statistical learning: data mining, inference, and prediction* New York: Springer; 2001.
25. Smola AJ, Schölkopf B: **A tutorial on support vector regression.** *Stat Comput* 2004, **14:**199-222.
26. Shen Q, Jiang JH, Tao JC, Shen GL, Yu RQ: **Modified Ant Colony Optimization Algorithm for Variable Selection in QSAR Modeling: QSAR Studies of Cyclooxygenase Inhibitors.** *J Chem Inf Model* 2005, **45:**1024-1029.
27. Goss S, Aron S, Deneubourg JL, Pasteels JM: **Self-organized shortcuts in the Argentine ant.** *Naturwissenschaften* 1989, **76:**579-581.
28. Dorigo M, Di Caro G, Gambardella LM: **Ant algorithms for discrete optimization.** *Artif Life* 1999, **5:**137-172.
29. Izrailev S, Agrafiotis DK: **Variable selection for QSAR by artificial ant colony systems.** *SAR QSAR Environ Res* 2002, **13:**417-423.
30. Gunturi SB, Narayanan R, Khandelwal A: **In silico ADME modelling 2: Computational models to predict human serum albumin binding affinity using ant colony systems.** *Bioinorg Med Chem* 2006, **14:**4118-4129.
31. **R: A Language and Environment for Statistical Computing** 2006 [http://www.R-project.org]. R Foundation for Statistical Computing, Vienna, Austria
32. Karthikeyan M, Glen RC, Bender A: **General melting point prediction based on a diverse compound data set and artificial neural networks.** *J Chem Inf Model* 2005, **45:**581-590.
33. Nigsch F, Bender A, van Buuren B, Tissen J, Nigsch E, Mitchell JBO: **Melting point prediction employing k-nearest neighbour algorithms and genetic parameter optimization.** *J Chem Inf Model* 2006, **46:**2412-2422.
34. Breiman L: **Random Forests.** *Mach Learn* 2001, **45:**5-32.
35. Hasegawa K, Miyashita Y, Funatsu K: **GA Strategy for Variable Selection in QSAR Studies: GA-Based PLS Analysis of Calcium Channel Antagonists.** *J Chem Inf Comput Sci* 1997, **37:**306-310.
36. Selwood DL, Livingstone DJ, Comley JCW, O'Dowd AB, Hudson AT, Jackson P, Jandu KS, Rose VS, Stables JN: **Structure-activity relationships of antifilarial antimycin analogues: a multivariate pattern recognition study.** *J Med Chem* 1990, **33:**136-142.
37. Balaban AT: **Highly discriminating distance-based topological index.** *Chem Phys Lett* 1982, **89:**399-404.

Part IV

The Rest

BMC Bioinformatics

Software

Userscripts for the Life Sciences

Egon L Willighagen*[1], Noel M O'Boyle[2], Harini Gopalakrishnan[3], Dazhi Jiao[3], Rajarshi Guha[3], Christoph Steinbeck[4] and David J Wild[3]

Address: [1]Cologne University Bioinformatics Center, Cologne University, Cologne, Germany, [2]Cambridge Crystallographic Data Centre, Cambridge, UK, [3]School of Informatics, Indiana University, Bloomington, USA and [4]Wilhelm-Schickard-Institut, Center for Bioinformatics, University of Tübingen, Tübingen, Germany

Email: Egon L Willighagen* - egonw@users.sf.net; Noel M O'Boyle - baoilleach@gmail.com; Harini Gopalakrishnan - hgopalak@indiana.edu; Dazhi Jiao - djiao@indiana.edu; Rajarshi Guha - rguha@indiana.edu; Christoph Steinbeck - c.steinbeck@steinbeck-molecular.de; David J Wild - djwild@indiana.edu

* Corresponding author

Published: 21 December 2007

BMC Bioinformatics 2007, **8**:487 doi:10.1186/1471-2105-8-487

Received: 31 August 2007
Accepted: 21 December 2007

This article is available from: http://www.biomedcentral.com/1471-2105/8/487

Abstract

Background: The web has seen an explosion of chemistry and biology related resources in the last 15 years: thousands of scientific journals, databases, wikis, blogs and resources are available with a wide variety of types of information. There is a huge need to aggregate and organise this information. However, the sheer number of resources makes it unrealistic to link them all in a centralised manner. Instead, search engines to find information in those resources flourish, and formal languages like Resource Description Framework and Web Ontology Language are increasingly used to allow linking of resources. A recent development is the use of userscripts to change the appearance of web pages, by on-the-fly modification of the web content. This opens possibilities to aggregate information and computational results from different web resources into the web page of one of those resources.

Results: Several userscripts are presented that enrich biology and chemistry related web resources by incorporating or linking to other computational or data sources on the web. The scripts make use of Greasemonkey-like plugins for web browsers and are written in JavaScript. Information from third-party resources are extracted using open Application Programming Interfaces, while common Universal Resource Locator schemes are used to make deep links to related information in that external resource. The userscripts presented here use a variety of techniques and resources, and show the potential of such scripts.

Conclusion: This paper discusses a number of userscripts that aggregate information from two or more web resources. Examples are shown that enrich web pages with information from other resources, and show how information from web pages can be used to link to, search, and process information in other resources. Due to the nature of userscripts, scientists are able to select those scripts they find useful on a daily basis, as the scripts run directly in their own web browser rather than on the web server. This flexibility allows the scientists to tune the features of web resources to optimise their productivity.

Background

The web has seen an explosion of chemistry and biology related resources in the last 15 years: thousands of scientific journals, databases, wikis, blogs, and regular HTML pages are available containing information relevant to chemists and biologists [1-4]. While each of those resources is valuable in itself, integrating information from these resources increases the value even more: for example, PubChem provides a wealth of data but could be complemented with 3D models to create an even richer information source.

The original goal of the world wide web was to hyperlink individual web pages allowing humans to explore a web of knowledge. For individual web pages these links can be created manually, as is still done in blogs, wikis, and static HTML pages; for large databases this is, however, not feasible. Userscripts are small programs that can alter the HTML content rendered by web browsers. For example, a userscript may add book prices from competitors to the Amazon.com website, or may remove unwanted advertisements from a site. Using the same approach, userscripts can also solve the problem of interlinking web resources, by adding to web pages of one resource dynamically generated hyperlinks into another. By selecting a specific set of userscripts, the user can tune a website to provide all kinds of facilities not anticipated by the original author of the site. For example, userscripts have been used in bioinformatics to enhance the iHOP web page [5]: the script extracts user assigned tags from a third party resource, and shows them as a tag cloud on iHOP pages for particular genes.

Automatic hyperlinking is only possible though the use of unique identifiers such as the PDB ID, the CAS registration number and, more recently, the IUPAC International Chemical Identifier (InChI). While identifiers are easily used to connect databases, such as done in the SRS system [6] or in meta database software like BioWarehouse [7], the sheer number of web resources makes it impossible integrate all resources. Consequently, (bio)chemical search engines, such as ChemSpider [8] and tools to harvest information from web resources, such as ChemXtreme [9] and BioSpider [10,11], as well as systems that standardize algorithmic access to resources and services, such as BioMOBY [12], have emerged.

Another reason why identifiers do not always allow linking resources is that many of them are database specific, such as the PDB ID and the Digital Object Identifier (DOI), and sometimes even restricted in being used, as with the CAS registration number. Open standard identifiers address this problem. Such identifiers can be derived from ontologies, dictionaries, encyclopedia, or computed by an algorithm. The Gene Ontology terms are often used as identifiers [13], indicating that a specific database entry is related to the cited term in the ontology, and therefore related to entries from other databases annotated with that term.

Identifiers that can be calculated algorithmically are even better, because they do not need to be looked up in a list of identifiers. Instead, anyone can calculate them from the object itself. For example, for molecular structures the InChI [14] is the ideal replacement for database specific identifiers such as the CAS registration number, the PubChem compound identifier and the ChEBI identifier. These all require a look up or conversion table to convert one identifier into another. Using the InChI, one can look up information in all databases without having to know the database specific identifier.

In addition to the unique identifier, one additional functionality is needed to create a link to a particular database: the database must provide either an API (Application Programming Interface) which can be queried using the identifier or else provide a uniform scheme for deep linking to a web page containing information about the entry behind the identifier. For example, looking up structures in PubChem is done with a scheme in which the InChI is embedded verbatim. To look up the structure of methane (InChI=1/CH4/h1H4), the URL http:www.ncbi.nlm.nih.gov/entrequery.fcgi?CMD=search&DB=pccompound&term=%22InChI=1/CH4/h1H4%22[InChI] is used.

The plethora of resources is overwhelming, and both users and database developers may have preferred subsets, e.g. more trusted, resources. It is therefore worthwhile to have a system that allows users to choose which resources they want to have linked with which other resources. Userscripts provide the necessary technology to allow this within web browsers. Here we describe several userscripts we have developed to create links between web resources of interest to researchers in the life sciences.

Implementation

We use the following techniques to link various web resources in this paper: userscripts, unique identifiers, microformats, and web resource interfaces. The following sections describe how these are used in this work.

Userscripts

A userscript is a small program written in JavaScript that is automatically run within a web browser (often by a plugin or add-on) when the user accesses pages that match a particular URL. Userscripts allow the user to modify the HTML content of a web page on-the-fly, by adding or removing elements or by moving them around. For example, userscripts exist that remove pop-up advertisements

from web pages, and that alter the Amazon.com web page to provide book prices from alternative suppliers. A repository of userscripts exists at userscripts.org/citeuserscripts-dotorg. Chemists and biologists can find relevant userscripts by searching with the terms "chemistry" or "biology".

Of the popular web browsers, only Opera provides built-in support for userscripts (referred to as 'User JavaScript'). To enable userscript support in other browsers, a third-party extension needs to be installed: Greasemonkey [15] for Firefox, Creammonkey [16] for Safari, IE7pro [17] or Turnabout [18] for Internet Explorer. The userscripts presented in the Results section are targeted at Greasemonkey, although it should be possible to run them in any browser with only minor changes.

The web browser user has full control over which userscripts she wants to have installed, allowing her to customise web pages exactly the way she wishes. Once installed, it is possible to individually enable or disable installed scripts. For example, for Greasemonkey see the "Manage User Scripts" option in the Tools menu under "Greasemonkey", or to disable the extension completely, click on the Greasemonkey icon in the status bar. Further control is provided by specifying to which web pages the script applies. Userscripts define default rules (e.g. http://www.biomedcentral.com/), but the user is normally able to override these.

The userscript has two main methods to find the HTML content to which to add or remove elements. The most accurate one is to analyse the document object model (DOM). This approach is used by the Sechemtic userscript to find uses of chemical microformats (see example below under Microformats). The other method is to use regular expressions to find certain strings in the text of the web page. This works particularly well for identifiers with a unique and well described syntax. For example, a regular expression for InChIs will have fewer false positives than one for PDB identifiers.

As with any program that you run on your computer, it is important to consider security when installing userscripts. Although the security model used by Greasemonkey prevents attacks by malicious websites, it is unable to detect or prevent the user himself installing a malicious userscript. Such scripts do exist; recently, malicious userscripts were uploaded to Userscripts.org that attempted to steal information from users' cookies. In that case, once the problem was discovered the malicious userscripts were easily detected and removed by the administrator. We recommend that unless you are familiar with JavaScript and carefully inspect the source code, you should only install userscripts from a trusted source.

Unique identifiers

Recognition of biological and chemistry relevant information on web pages is simplified by using identifiers [19]. Such identifiers may or may not be marked up with semantic markup such as microformats (see below). Identifiers are widely used to make connections between databases, and often identify a specific entry in a database. Some examples of this are the PDB identifier, Digital Object Identifiers, PubChem compound identifier, and the CAS registry number for, respectively the PDB, DOI, PubChem, and the Chemistry Abstract Service databases. In this study we use DOIs, InChIs, and PDB identifiers as our unique identifiers (Table 1).

Table 1: Userscripts for the life sciences. A summary of the resources and identifiers used by userscripts for the life sciences. The Identification method indicates how the userscript recognises relevant information on a web page. The Identifiers column describes the unique identifier searched for. The Resources column indicates the web resource to which a link is created, or from which data is extracted.

		Technologies and Resources used	
Name	Identification method	Identifiers	Resources
Jmol4PubChem	HTML tags on PubChem	PubChem ID	Pub3D [36]
OSCAR3 on HTML	natural language processing	chemical structure name	-
PDB-Jmol	regular expression	PDB ID	First Glance in Jmol [41]
Sechemtic	microformats	InChI, SMILES, CAS number	PubChem [32] eMolecules [43] Google [54]
Add quotes to DOIs	regular expression	DOI	Postgenomic [3] Chemical blogspace [4]
Add quotes to molecules	microformats	InChI	Chemical blogspace [4]
Add to Connotea	regular expression	DOI	Connotea [30]

Microformats

Microformats [20] are a lightweight specification that extends HTML to add semantic markup to web pages. For example, hCard is a microformat that allows semantic mark up of address information [21], and hCalendar is a microformat specification for the representation of calendar information about events [22].

A microformat specification has also been suggested for chemistry that would make it much easier to recognise compound names, InChIs, SMILES and CAS registry numbers. Userscripts, or indeed any other programs, would then no longer need to depend on regular expressions to find names and identifiers, but could use this markup to accurately extract the identifier.

For example, a web page implementing the InChI microformat would wrap any InChIs in a HTML element with a @class attribute as follows: InChI=1/. This information can easily be extracted using the document.evaluate method which takes an XPath [23] expression (//span[@class="in chi"] in this case):

```
allInChIs = document.evaluate(
  '//span[@class="inchi"]', document, null,
  XpathResult.UNORDERED_NODE_SNAPSHOT_TYPE,
  null
);
```

This code returns all HTML nodes that mark up InChI strings using the InChI microformat. By iterating over these nodes, the userscript can insert new HTML elements, such as links to external resources as shown here in code taken from the Sechemtic userscript:

```
for (var i=0; i<allInChIs.snapshotLength; i++){
    spanElement = allInChIs.snapshotItem(i);
    inchi = spanElement.innerHTML;
    // create a link to PubChem
    newElement = document.createElement('a');
    newElement.href = "http://www.ncbi.nlm." +
      "nih.gov/entrez/query.fcgi?CMD=search" +
      "&DB=pccompound&term=%22" + inchi +
      "%22[InChI]";
    newElement.innerHTML =
      "<sup>PubChem</sup>";
    spanElement.parentNode.insertBefore(
      newElement, spanElement.nextSibling
    );
}
```

Web resource interfaces

Web databases are the primary source of information used by the discussed userscripts. While it is easy to have scripts create links to external web resources, it is also possible for them to retrieve information from those resources and include it in the HTML content of the web page the user is browsing. The latter is, for example, performed by the userscript that adds comments from Postgenomic.com and Chemical blogspace to journal web pages.

The general approach userscripts use to retrieve information from external web resources uses HTTP just like any web browser itself. To simplify the process, userscripts tend to use a combination of *XMLHttpRequest*, possibly via the Greasemonkey *GM_xmlhttpRequest* wrapper method, and the JavaScript Object Notation (JSON) format [24] for data representation. The XMLHttpRequest method retrieves the information using a URL that normally points to a data interface, or API. The Postgenomic.com software has such an API that returns the blog posts that discuss a particular article, as identified by its DOI. Chemical blogspace uses the same API, and adds another one to return blog posts that discuss a particular molecule, as identified by its InChI. Both database APIs can return the information as JSON objects, which is how they are used in the discussed userscripts.

Since our userscripts rely on a particular API or specially-constructed URL to access an external resource, they will fail if the external resource changes its API or the URL it provides to access it. This will not affect the browsing experience of the user, but the additional functionality provided by the userscript will no longer be available. To deal with this, each of the userscripts described in this article checks once a day for a new version and prompts the user to install it if one is available. This means that when a userscript is updated to deal with a new API or URL, every user will quickly have access to the latest version.

Results

This paper introduces userscripts that have been written in our research groups as exemplars of how web resources can be integrated and to outline how they can be used in research. Our userscripts can be classified into two broads areas: those that link chemical and biological data to websites, and those that affect how we interact with the scientific literature.

In the following sections, we describe in detail how functionality is added to the web page being browsed. Table 1 summarises the resources linked to, or accessed, by each script, as well as the unique identifier used.

Interacting with the scientific literature

OSCAR3 running on HTML

Published journal articles and other web documents with chemistry content are not normally marked up by the publishers or authors to provide machine readable representations of chemical structures and related information. As a result, there has been active interest in methods which can mark documents up automatically. In particular, OSCAR3 [25,26], developed at the Unilever Centre for Molecular Informatics at the University of Cambridge, and used by the Royal Society of Chemistry in their Project Prospect [27], searches documents for chemical names, spectra, and other chemical information, and automatically marks up the content using XML tags (to the extent of where possible generating machine readable SMILES and InChI structures for chemicals referenced in the document).

We have created a userscript, ChemGM.user.js that will automatically run OSCAR on a web page and provide inline hypertext links to PubChem for chemical structure names that are found in the page (including 2D structure depictions generated by another web service and PubChem searches). The userscript can be run on any web page, but it is particularly applicable to online journal articles and chemistry blogs. An example highlighting the effect of this userscript is shown in Figure 1. Note that though the images use an article from *Chemistry Central*

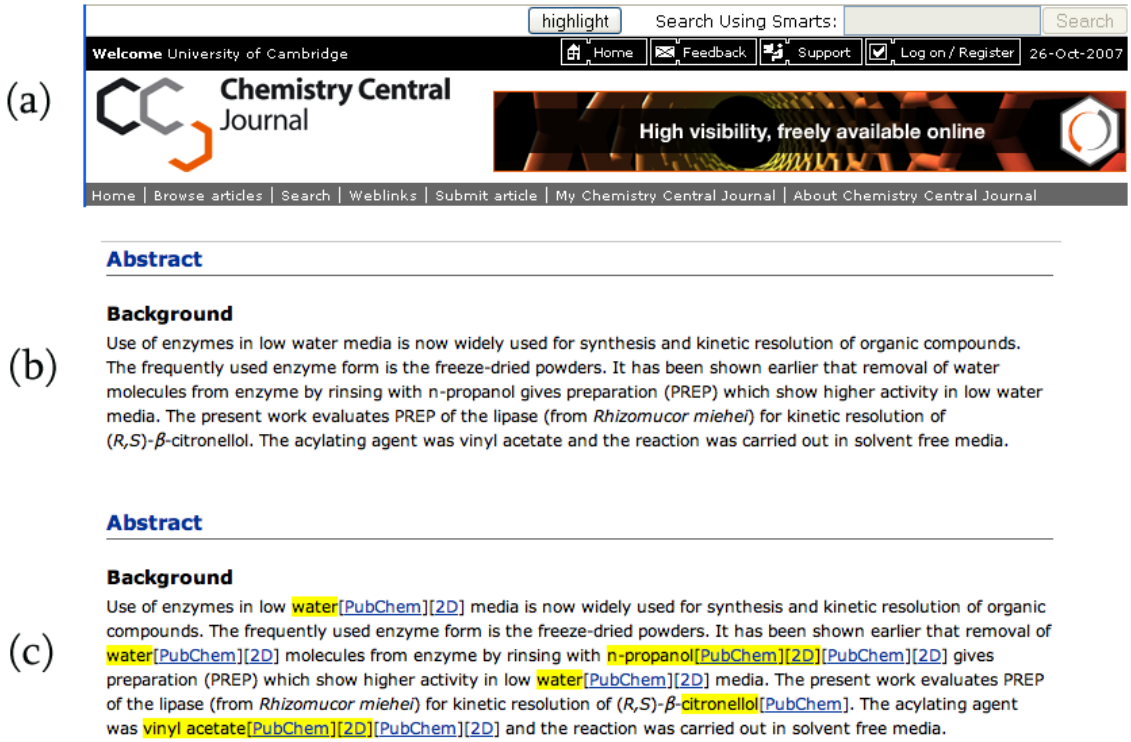

Figure 1

Highlighting and annotating chemical terms in an online journal article. Screenshots showing the effect of the ChemGM.user.js userscript on the *Chemistry Central Journal* web page (full URL: [47]) for Majumder et al. [48]. (a) When the userscript is running a toolbar is added to the top of every webpage. Clicking the highlight button in the toolbar causes the contents of the webpage to be analysed for chemical terms. (b) shows the original text of the abstract. (c) After a minute or so, any chemical terms recognised are highlighted in yellow, and are annotated with hypertext links to their entries in PubChem (if available) and a 2D depiction of the image.

Journal, the script can be applied to any web page, irrespective of its source or content.

Add quotes from Chemical blogspace and Postgenomic to DOIs

It can be a challenge to keep up with the primary literature in a field. At the same time, there are a large number of scientific blogs, many of which have reviews of the recent literature or highlight interesting papers. The Postgenomic web site was developed by Euan Adie and later hosted by Nature Publishing Group and currently aggregates information from over 750 scientific blogs [3]. The source code is open and has been used by one of the authors (ELW) to establish a similar site, Chemical blogspace, for over 140 blogs with chemical content [4]. Both of these sites identify references to journal articles in blogs, and make this information available through an API. Compared to the Postgenomic website, the Chemical blogspace site also identifies molecules referenced in blogs either by microformat markup of InChI and SMILES, or by analysing links to Wikipedia [28]. If the latter link points to a wiki page that contains a PubChem compound identifier or an InChI, then the molecular structure is linked to the blog post.

This userscript uses the aggregated information collected by Postgenomic and Chemical blogspace. It runs whenever the user accesses the website of a journal publisher. It identifies any DOIs on the page, and uses the Chemical blogspace and Postgenomic APIs to find out whether those DOIs have been referenced in a blog post. If so, an icon is added to the web page next to the DOI which, if hovered over with the mouse, causes a popup to appear containing the name of the citing blog post, the blog name, and the first few lines of text of the blog. The full content can be accessed by clicking on the title of the blog post. In this way, content from blog articles widely dispersed in terms of the web is brought directly to where it is likely to be of most interest – the journal web site. Figure 2 shows the effect of this userscript when running on the HTML version of Spjuth et al. [29].

Providing reviews of journal articles is only one of the uses of such a userscript. It is also a general way to create a link between the content of a blog post and a particular paper. In this way, bloggers can use blog posts to enhance the original journal website without any intervention required by the publisher. For example, the author of a paper may write a blog post which provides additional supporting information for a journal article or includes the article preprint for those who do not have a subscription. Alternatively, the author of a paper may write a blog post and include the DOIs of all of the references. This

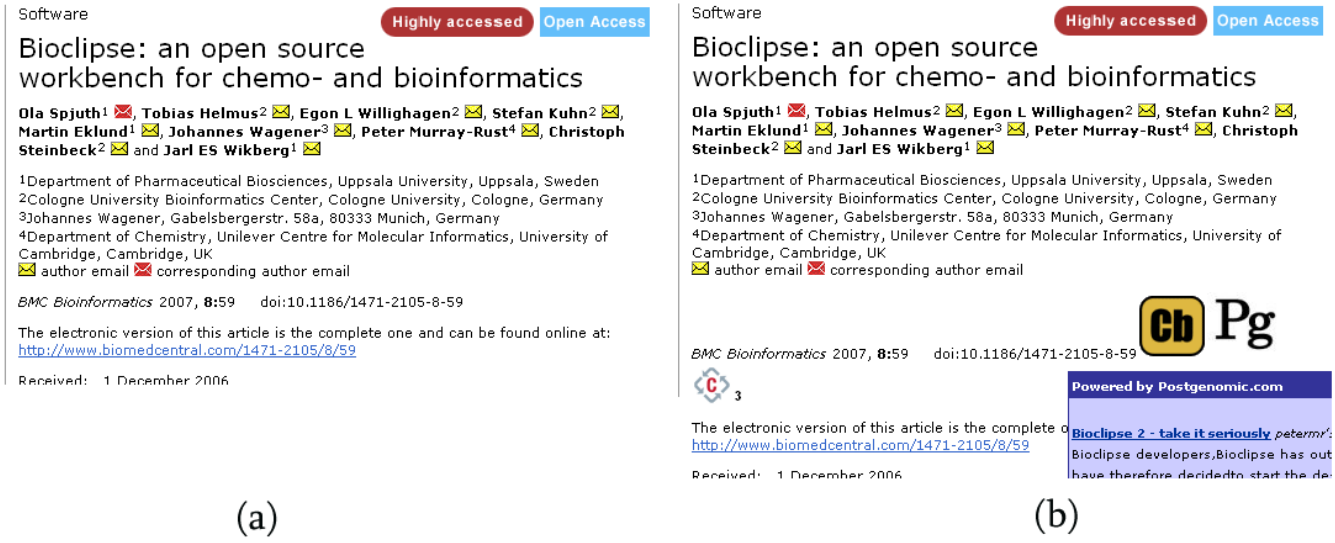

Figure 2

Adding information to DOIs on journal web pages. Screenshots from the *BMC Bioinformatics* web page (full URL: [49]) for Spjuth et al. [29] (a) without any userscript enabled, and (b) showing the effect of the two userscripts "Add quotes from Chemical blogspace and Postgenomic to DOIs" and "Add to Connotea". The latter added a Connotea logo (a 'c' surrounded by linking arrows), which links to the Connotea dialog box for adding this paper to your library, and a number indicating how many people have already bookmarked this paper, which links to the existing entry for this paper on Connotea. The "Add quotes" userscript added the Cb logo, which links to the Chemical blogspace page for this paper, and a Pg logo, linking to the Postgenomic page. The popup titled "Powered by Postgenomic.com" (only partially shown) appears when the mouse is placed on the Pg logo, and contains quotes from and links to the citing blog articles.

would not only promote his/her own paper (all of the cited papers would show a blog comment pointing to the citing paper), but would result in an eventual network of citations which could be used to measure the impact of a paper.

Add to Connotea

Connotea is a social bookmarking site developed by Nature Publishing Group for scientists [30]. It allows a user to bookmark websites using either the DOI or a URL, and to tag those bookmarks. Crucially, it also provides an API for retrieving information.

The "Add to Connotea" userscript has two aspects. Firstly, it makes it easy to add papers to Connotea from journal webpages, by adding a hyperlink in the form of the Connotea logo next to every DOI identified on a journal web page. Clicking on the logo brings the user to the Connotea page for adding new papers. This aspect of the userscript is not entirely novel. A userscript has previously been developed which allows the user to add papers to Connotea from NCBI PubMed [31]. In addition, a small number of publishers (which includes BioMed Central and Nature Publishing Group), provide a facility to add papers to Connotea directly from their website. Our userscript differs in that it will work on the website of *any* journal publisher where the text contains DOIs.

The second aspect of this userscript is more interesting in the context of this paper. The userscript queries the Connotea API to find out how many people have previously added this paper to their Connotea account. It then adds this number next to the Connotea icon. Clicking on the number brings you to the Connotea page for that paper. From here it is possible to access comments on the paper. More useful perhaps, is the ability to find related papers by looking at the other papers a particular Connotea user has tagged with the same tag. Figure 2 shows the effect of this userscript when running on the HTML version of Spjuth *et al.*[29].

This aspect of the userscript has the potential to affect the way we read the literature. The number of times a particular paper has been bookmarked on Connotea can be considered a measure of its importance or its interest. In the past, measures such as the number of citations have served this purpose, but this information is generally not shown on journal web pages as it is not freely available. Another effect of this userscript is to link the paper the user is viewing to related papers through the Connotea website. If a researcher finds that a particular paper has been bookmarked on Connotea and is of interest to him or her, he or she can is likely to find other relevant papers by browsing through the other papers bookmarked by the same Connotea user with the same tag.

On a technical note, this userscript illustrates some techniques necessary for accessing an API that requires a user name and password and that, in addition, only permits one API request every two seconds or so. Note that this userscript requires the user to have a Connotea account (which is freely available at Ref. [30]).

Linking to chemical and biological data sources

Enhancement of PubChem with 3D structures

The PubChem repository is a public collection of over 10 million compounds [32]. The database contains 2D structures as well as a number of precomputed properties (such as number of heavy atoms and topological polar surface area [33]). The web interface to this database allows a wide variety of queries. The results are usually represented in the form of a summary web page containing images of the 2D structures of all the compounds satisfying the query with links to pages for individual compounds which provide a summary of the properties of the compound. In many cases it would be useful to be able to view an image of the 3D structure of a molecule. However, PubChem currently does not contain 3D structures for the compounds stored in the database.

To address this problem, we developed a database of 3D structures of PubChem compounds as part of our web service infrastructure for chemoinformatics [34]. The structures were generated using a two-step process in which the SMILES were converted to a set of rough 3D coordinates using stochastic proximity embedding [35] and subsequently geometry optimised using the MMFF94 force field, using in-house code. A number of compounds were excluded from the final 3D database since the force field did not contain parameters for certain atom types. However the 3D database, known as Pub3D [36], contains approximately 99% of the compounds in PubChem. Pub3D is wrapped by a set of web services which encapsulate common queries including finding a structure by compound ID (CID) or finding structures matching a SMARTS pattern.

Using this web service interface we created a userscript called 3DStructureView.user.js that allows 3D structures from our database to be shown when users visit the PubChem website (see Figure 3). The script is designed to work only on the summary and detail pages that a user views after a PubChem search. It parses the page and identifies the compound ID which is then used to construct a call to the Pub3D database. The return value is a string containing the 3D structure of the compound, in SD format, which is used to construct an appropriate URL. The result of this process is that the user can now click on a link titled *3DView(Jmol)*, which will cause a Jmol applet [37-39] window to appear showing the 3D structure of the compound in question.

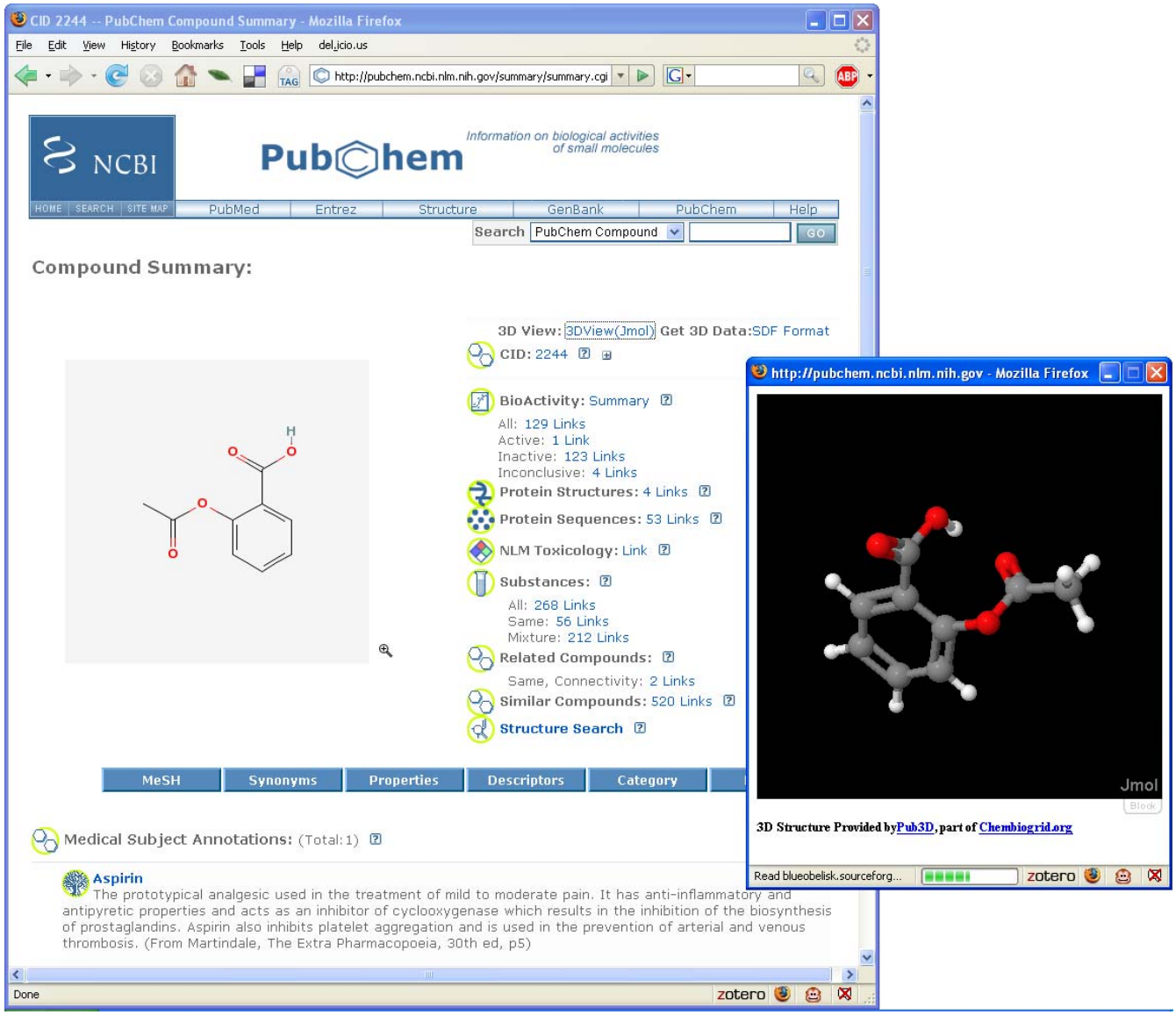

Figure 3
Adding 3D models to PubChem. Screenshot of the PubChem web page for aspirin (full URL: [50]) with the "3DStructureView" userscript enabled. The userscript added the first line of text in the compound summary information. Clicking on the "3DView(Jmol)" link causes a window to popup showing a 3D model of the structure. Clicking on the "SDF Format" link allows the user to download the calculated 3D structure of the molecule in SDF file format.

As an example, after installing the script, one can navigate to the PubChem website [32] and search for entries related to aspirin. This should return slightly more than thirty hits. If one then clicks on the compound ID for the first hit, one is taken to a summary page which provides various details regarding the molecular structure and biological activity of aspirin. In addition to the data provided by PubChem, the userscript has enhanced the page to add two links: *3DView(Jmol)* and *SDF Format*. The former link will bring up an instance of the Jmol applet showing a 3D structure of aspirin, while the second link allows one to download the 3D structure in the SD file format (see Figure 3).

PDB-Jmol Greasemonkey Script

The Protein Data Bank [40] is a repository of experimentally-determined 3D coordinates of proteins. Each entry has a PDB ID, which is a unique four letter identification code consisting of a number followed by three characters which can be either letters or numbers; for example, 1abe, 114L and 6NN9. The PDB-Jmol Greasemonkey userscript identifies all PDB IDs on web pages and adds hyperlinks

to the FirstGlance in Jmol web page [41] for that protein. This website uses the Open Source molecular viewer Jmol to show the protein as a 3D model which can be manipulated by the user. In this way, the user can instantly view the 3D structure of any PDB ID mentioned on a website, and in particular, if the user is reading the HTML version of a journal article on-line, all PDB IDs in the paper will similarly be enhanced. Figure 4 shows an example of the latter case where PDB identifiers in the the online version of Mardia et al. [42] have been identified and links added.

As this userscript runs on all web pages accessed by the user, and since the search term is simply 4 characters long, additional constraints are necessary to prevent excessive false positive identification. The userscript only looks for PDB IDs if it finds one of the following terms in the web page: "protein", "PDB", or "enzyme".

Sechemtic

Sechemtic is a small userscript that detects use of microformats (see Implementation) to markup molecular identifiers, as well as regular molecular names. It recognises markup for the IUPAC InChI and SMILES, and creates links for those molecules to web resources like eMolecules [43], PubChem [32] and a link to Google to search for more information (see Figure 5). It should be noted that, in particular with Google searches, the links based on InChIs are more useful as the same molecule may be represented by several different SMILES strings but only a single InChI.

From a technological point of view, these scripts are very simple in nature; the semantic nature of the (chemical) microformats is what makes this simple script possible. The semantic markup in HTML for InChIs that is picked up by the userscript looks like InChI =1/CH4/h1H4 while the markup for a SMILES string looks like CCO.

Add quotes from Chemical blogspace to molecules

This userscript, quite similar to the one that adds comments to DOIs, runs on all web pages accessed by the user. Using the same method as the Sechemtic userscript (see above), it identifies any molecules referenced on a page which have been marked up with the appropriate tags. It also supports the (non-marked up) InChI tags on PubChem. It then uses the Chemical blogspace API to find out whether this molecule has been referenced in a blog post. The remainder is as for the previous userscript; an icon is added which contains a popup to the citing blog post. Figure 6 shows the effect of the userscript on the PubChem page for methane (InChI=1/CH4/h1H4). A full

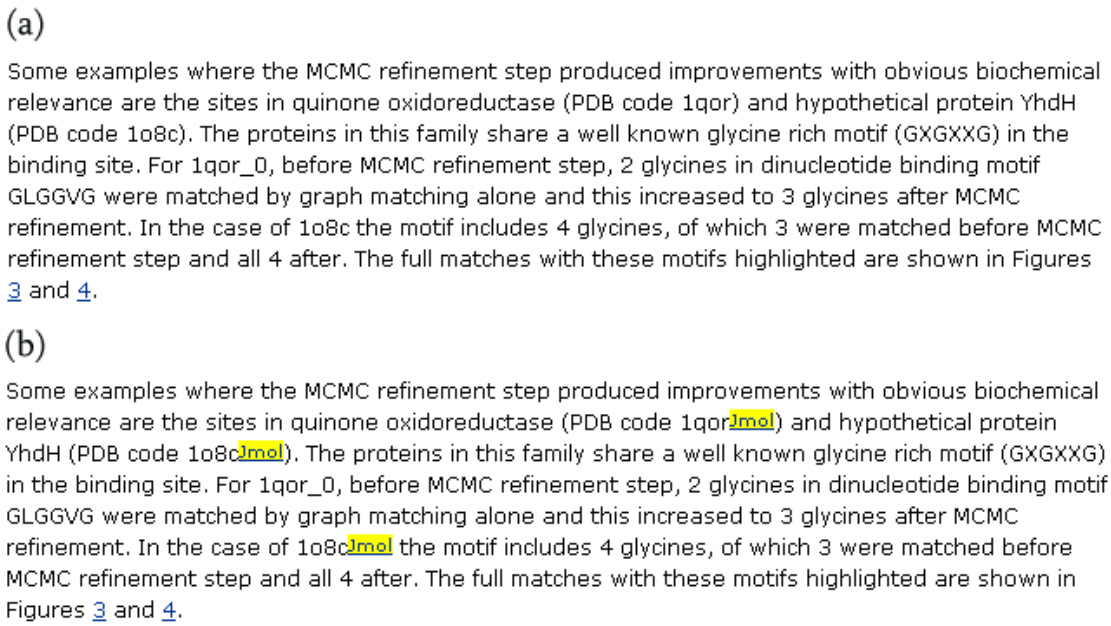

Figure 4

The effect of the PDB-Jmol userscript. Screenshots from the *BMC Bioinformatics* web page (full URL: [51]) for Mardia et al. [42] showing a paragraph containing PDB identifiers (a) without the PDB-Jmol userscript installed, and (b) with the PDB-Jmol userscript installed. The "Jmol" text in yellow is a hyperlink to the FirstGlance in Jmol page [41] for a particular protein structure.

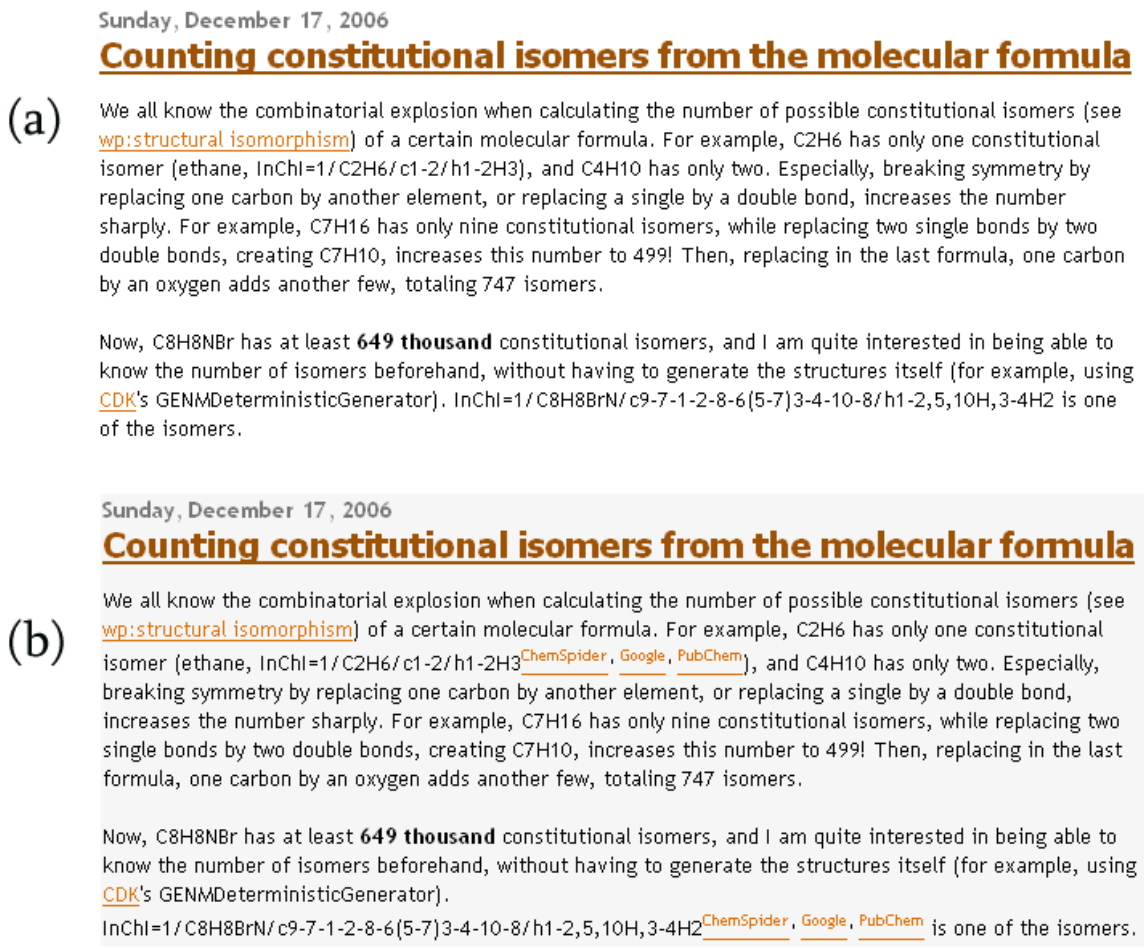

Figure 5
Annotating chemical terms marked up with microformats. Screenshots showing a blog post (full URL: [52]) containing chemical terms marked up with chemical microformats, (a) without and (b) with the "Sechemtic" userscript enabled. The added hyperlinks allow the user to look up the structure in Google, ChemSpider and PubChem.

list of molecules with comments in Chemical blogspace is available from Ref. [44].

A possible use of this script is to link all discussions of a particular drug in the blogosphere to a static page containing information on the drug. Another use is to link discussions on syntheses of molecules to pages containing references to the molecule.

Discussion

Here we have focused on the development of userscripts that enhance web pages for biologists and chemists. If all of these userscripts are installed, any web page with a PDB code will now contain a link to view the structure in 3D, journal webpages will show chemical structure markup and blog comments on articles, 3D structures and links to appropriate blog posts will be available from PubChem, and any use of chemical microformats will be picked up adding links to Google, eMolecules, ChemSpider and PubChem.

These examples show that userscripts offer a powerful technology to improve the way we read the scientific literature and access (bio)chemical databases. This is done by dynamically combining web resources, and enriching the information content of the primary resources. Theoretically, such links can be made on the web server itself, and this is commonly done, but it does not give the user the flexibility to choose what features to install. The crucial point about userscripts is that they do not require the involvement of the web site provider. All of the enhancements are done on-the-fly by the user's browser.

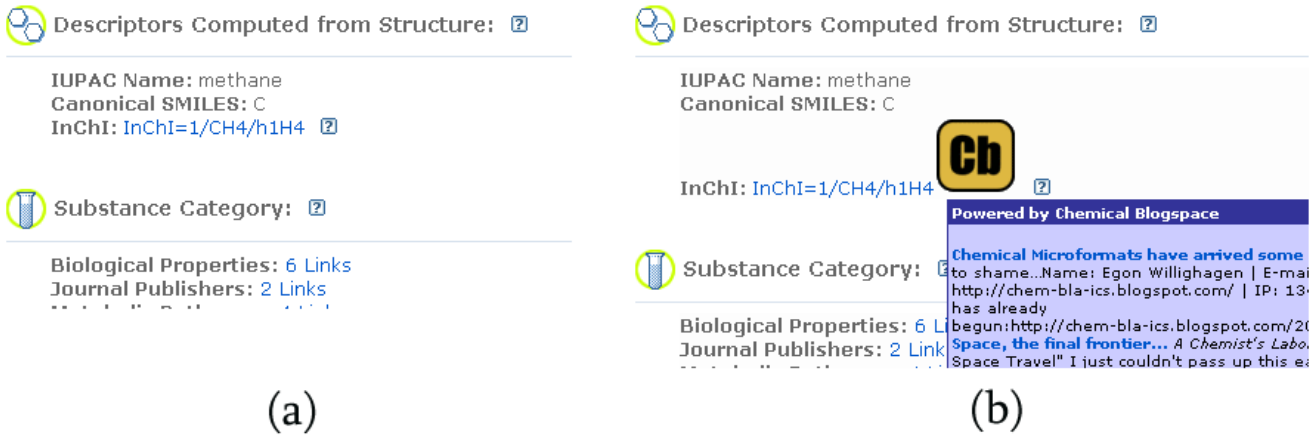

Figure 6
Adding comments from the blogosphere to molecules. Screenshots from the PubChem web page for methane (full URL: [53]), (a) without and (b) with the "Add quotes from Chemical blogspace to molecules" userscript enabled. The InChI *InChI=1/CH4/h1H4* is identified by the userscript, which then adds the Cb logo. The logo is a link to the Chemical blogspace page for this molecule. The popup titled "Powered by Chemical blogspace" (only partially shown) appears when the mouse is placed on the Cb logo, and contains quotes from and links to blog posts that discuss this molecule.

The userscripts combine a number of technologies for data retrieval and communication. Information from HTML pages is extracted using identifiers, regular expressions, XPath queries and microformats. It is noted that the syntax of (bio)chemical and other identifiers is generally not distinct enough to detect them with perfect recall and optimal precision. It is easiest to write regular expressions for the DOI and the InChI with a high precision, compared to, for example, the PDB ID which has a syntax which can clash with other web page content.

Microformats offer a solution for such less well-defined identifiers. This technology is used to wrap identifiers with some semantic markup so that the userscript can easily extract the identifiers using XPath queries. However, microformats do not incorporate a mechanism to provide details on what a microformat means. That is, microformats are not backed up by a specified ontology. As a result the chemical 'smiles' microformat, to markup SMILES, may collide with a microformat specification to markup moods.

Once the identifier is extracted by whatever means, the userscripts can either create links to other web resources, or query those resources and embed results into the HTML of the web page on which the userscript is run. While any HTTP-based approach can be used for this, the example userscripts show that combining XMLHttpRequest with JSON [24] is a rather straightforward approach.

Conclusion

We have shown that userscripts are a simple and useful way of integrating bio- and chemoinformatics web resources. In particular, they permit (a) the augmentation of existing websites with functionality not envisioned or indeed wanted by the original author, (b) the integration of information from different domains, and (c) a connection point between the social web (wikis, blogs etc.) and traditional web tools and sites. We continue to find interesting uses for userscripts, and we hope this manuscript will spur others to do likewise.

Availability and requirements

- **Project name:** Userscripts for Chemistry and Biology
- **Project home page:** Blue Obelisk [45] website [46]. Download link: http://blueobelisk.sf.net/wiki/Userscripts
- **Operating system(s):** Platform independent
- **Programming language:** JavaScript
- **Other requirements:** Firefox with Greasemonkey add-on (or equivalent) for userscript support; Java is required to view the Jmol applet; a Connotea account is required for the Add to Connotea userscript
- **License:** GNU GPL, BSD
- **Any restrictions to use by non-academics:** none

Authors' contributions

NMOB, ELW, HG and DJ have written userscripts mentioned in this text. RG developed and maintains the 3D structure database and contributed to the development of the Pub3D userscript. DW and CS devised and tested some of the userscripts. All authors have read and approved the final manuscript.

Acknowledgements

Pedro Beltrão is acknowledged for laying the foundations of the userscript that adds blog comments to journal web pages. We thank the anonymous reviewers for their constructive comments.

References

1. Galperin MY: **The Molecular Biology Database Collection: 2007 update.** *Nucleic Acids Res* 2007, **35:**D3-D4.
2. Fox JA, McMillan S, Ouellette BFF: **A compilation of molecular biology web servers: 2006 update on the Bioinformatics Links Directory.** *Nucleic Acids Res* 2006, **34:**W3-W5.
3. **Postgenomic** [http://postgenomic.com/]
4. **Chemical blogspace** [http://cb.openmolecules.net/]
5. Good BM, Kawas EA, Kuo BY, Wilkinson MD: **iHOPerator: User-scripting a personalized bioinformatics Web, starting with the iHOP website.** *BMC Bioinformatics* 2006, **7:**534.
6. Etzold T, Ulyanov A, Argos P: **SRS: information retrieval system for molecular biology data banks.** *Method Enzymol* 1996, **266:**114-128.
7. Lee T, Pouliot Y, Wagner V, Gupta P, Calvert DS, Tenenbaum J, Karp P: **BioWarehouse: a bioinformatics database warehouse toolkit.** *BMC Bioinformatics* 2006, **7:**170.
8. **ChemSpider** [http://chemspider.com/]
9. Karthikeyan M, Krishnan S, Pandey AK, Bender A: **Harvesting Chemical Information from the Internet Using a Distributed Approach: ChemXtreme.** *J Chem Inf Model* 2006, **46:**452-461.
10. **BioSpider** [http://biospider.ca/]
11. Knox C, Shrivastava S, Stothard P, Eisner R, Wishart D: **BioSpider: A Web Server for Automating Metabolome Annotations.** *Pacific Symp Biocomp* 2007, **12:**145-156.
12. Wilkinson MD, Links M: **BioMOBY: an open source biological web services proposal.** *Brief Bioinform* 2002, **3(4):**331-341.
13. Ashburner M, Ball CA, Blake JA, Botstein D, Butler H, Cherry JM, Davis AP, Dolinski K, Dwight SS, Eppig JT, Harris MA, Hill DP, Issel-Tarver L, Kasarskis A, Lewis S, Matese JC, Richardson JE, Ringwald M, Rubin GM, Sherlock G: **Gene ontology: tool for the unification of biology. The Gene Ontology Consortium.** *Nat Genet* 2000, **25:**25-29.
14. **IUPAC International Chemical Identifier (InChI)** [http://www.iupac.org/inchi/]
15. **Greasemonkey** [http://www.greasespot.net/]
16. **Creammonkey** [http://creammonkey.sourceforge.net/]
17. **IE7pro** [http://www.ie7pro.com/]
18. **Turnabout** [http://www.reifysoft.com/turnabout.php]
19. Coles SJ, Day NE, Murray-Rust P, Rzepa HS, Zhang Y: **Enhancement of the chemical semantic web through the use of InChI identifiers.** *Org Biomol Chem* 2005, **3(10):**1832-1834.
20. **Microformats** [http://microformats.org/]
21. **hCard Microformat** [http://microformats.org/wiki/hcard]
22. **hCalendar Microformat** [http://microformats.org/wiki/hcalendar]
23. **XML Path Language (XPath) 2.0 – W3C Recommendation** [http://www.w3.org/TR/2007/REC-xpath20-20070123/]
24. **JSON** [http://json.org/]
25. Townsend JA, Adams SE, Waudby CA, de Souza VK, Goodman JM, Murray-Rust P: **Chemical documents: machine understanding and automated information extraction.** *Org Biomol Chem* 2004, **2:**3294-3300.
26. Corbett P, Murray-Rust P: **High-thoughput identification of chemistry in life science texts.** In *Computational Life Sciences II Volume 4216*. Edited by: Berthold MR, Glen R, Fischer I. Berlin/Heidelberg: Springer-Verlag; 2006:107-118.
27. **RSC Project Prospect** [http://www.rsc.org/Publishing/Journals/ProjectProspect/]
28. **Wikipedia** [http://www.wikipedia.org/]
29. Spjuth O, Helmus T, Willighagen EL, Kuhn S, Eklund M, Wagener J, Murray-Rust P, Steinbeck C, Wikberg JES: **Bioclipse: An open source workbench for chemo- and bioinformatics.** *BMC Bioinformatics* 2007, **8:**59.
30. **Connotea** [http://www.connotea.org/]
31. **pubmed2connotea** 2006 [http://lindenb.integragen.org/pubmed2connotea/].
32. **PubChem** [http://pubchem.ncbi.nlm.nih.gov/]
33. Ertl P, Rohde B, Selzer P: **Fast Calculation of Molecular Polar Surface Area as a Sum of Fragment Based Contributions and Its Application to the Prediction of Drug Transport Properties.** *J Med Chem* 2000, **43:**3714-3717.
34. Dong X, Gilbert KE, Guha R, Heiland R, Kim J, Pierce ME, Fox GC, Wild DJ: **Web Service Infrastructure for Chemoinformatics.** *J Chem Inf Model* 2007, **47:**1303-1307.
35. Agrafiotis DK: **Stochastic Proximity Embedding.** *J Comp Chem* 2003, **24:**1215-1221.
36. **Pub3D** [http://rguha.ath.cx/~rguha/cicc/p3d/]
37. **Jmol: an open-source Java viewer for chemical structures in 3D** [http://www.jmol.org]
38. Willighagen E, Howard M: **Fast and Scriptable Molecular Graphics in Web Browsers without Java3D.** *Available from Nature Precedings* 2007 [http://dx.doi.org/10.1038/npre.2007.50.1].
39. Herráez A: **Biomolecules in the computer: Jmol to the rescue.** *Biochem Mol Biol Edu* 2006, **34:**255-261.
40. Berman HM, Westbrook J, Feng Z, Gilliland G, Bhat TN, Weissig H, Shindyalov IN, Bourne PE: **The Protein Data Bank.** *Nucleic Acids Res* 2000, **28(1):**235-242.
41. **FirstGlance in Jmol** [http://firstglance.jmol.org/]
42. Mardia KV, Nyirongo VB, Green PJ, Gold ND, Westhead DR: **Bayesian refinement of protein functional site matching.** *BMC Bioinformatics* 2007, **8:**257.
43. **eMolecules** [http://emolecules.com/]
44. **Chemical blogspace – Chemical Compounds** [http://cb.openmolecules.net/inchis.php]
45. Guha R, Howard MT, Hutchison GR, Murray-Rust P, Rzepa H, Steinbeck C, Wegner J, Willighagen EL: **The Blue Obelisk-interoperability in chemical informatics.** *J Chem Inf Model* 2006, **46:**991-998.
46. **Blue Obelisk Userscripts** [http://blueobelisk.sourceforge.net/wiki/Userscripts]
47. **Chemistry Central Journal web page for Majumder et al** [http://journal.chemistrycentral.com/content/1/1/10]
48. Majumder AB, Shah S, Gupta MN: **Enantioselective transacetylation of (R,S)-β-citronellol by propanol rinsed immobilized *Rhizomucor miehei* lipase.** *Chem Cent J* 2007, **1:**10.
49. **BMC Bioinformatics web page for Spjuth et al** [http://www.biomedcentral.com/1471-2105/8/59]
50. **PubChem page for aspirin** [http://pubchem.ncbi.nlm.nih.gov/summary/summary.cgi?cid=2244]
51. **BMC Bioinformatics web page for Mardia et al** [http://www.biomedcentral.com/1471-2105/8/257]
52. **Counting consitutional isomers from the molecular formula** [http://chem-bla-ics.blogspot.com/2006/12/counting-stereoisomers-from-molecular_17.html]
53. **PubChem page for methane** [http://pubchem.ncbi.nlm.nih.gov/summary/summary.cgi?cid=297]
54. **Google** [http://google.com/]

Journal of Cheminformatics

SOFTWARE Open Access

Confab - Systematic generation of diverse low-energy conformers

Noel M O'Boyle[1,2*], Tim Vandermeersch[2], Christopher J Flynn[1], Anita R Maguire[1] and Geoffrey R Hutchison[2,3]

Abstract

Background: Many computational chemistry analyses require the generation of conformers, either on-the-fly, or in advance. We present Confab, an open source command-line application for the systematic generation of low-energy conformers according to a diversity criterion.

Results: Confab generates conformations using the 'torsion driving approach' which involves iterating systematically through a set of allowed torsion angles for each rotatable bond. Energy is assessed using the MMFF94 forcefield. Diversity is measured using the heavy-atom root-mean-square deviation (RMSD) relative to conformers already stored. We investigated the recovery of crystal structures for a dataset of 1000 ligands from the Protein Data Bank with fewer than 1 million conformations. Confab can recover 97% of the molecules to within 1.5 Å at a diversity level of 1.5 Å and an energy cutoff of 50 kcal/mol.

Conclusions: Confab is available from http://confab.googlecode.com.

Introduction

The generation of molecular conformations is an essential part of many computational analyses in chemistry, particularly in the field of computational drug design. Methods such as 3D QSAR, protein-ligand docking and pharmacophore generation and searching [1] all require the generation of conformers, whether on-the-fly (as part of the method) or pre-generated by a stand-alone conformer generator. In contrast to 3D structure generators (such as CORINA [2], DG-AMMOS [3] and smi23d [4]), which focus on the generation of a single low-energy conformation, conformation generators create an ensemble of conformers that cover the entire space of low-energy conformations or that part of conformational space occupied by biologically-relevant conformers.

Several proprietary conformation generators are currently available (including OMEGA [5], ROTATE [6], Catalyst [7], Confort [8], ConfGen [9], Balloon [10] and MED-3DMC [11] among others) but only recently have open source conformation generators appeared: Frog2 [12] generates conformers using a Monte Carlo approach, while Multiconf-DOCK [13] adapts the systematic search code from DOCK5 [14] to generate diverse conformers via a torsion-driving approach.

Confab 1.0 is the first release of Confab, an open source conformation generator whose goal is the systematic coverage of conformational space. Accuracy has been favoured over the introduction of approximations to improve performance. The algorithm starts with an input 3D structure which, after some initialisation steps, is used to generate multiple conformers which are filtered on-the-fly to identify diverse low energy conformers. Conformations are generated using the torsion-driving approach from a set of predefined allowed torsion angles. Ring conformations are not currently sampled.

The first section of the paper describes the algorithm used by the software and some implementation details. After this, two applications of the software are described: an analysis of the conformational space of a dataset of 1000 molecules (which includes a comparison to Multiconf-DOCK), and an investigation of the conformational preferences of a particular phenyl sulfone.

Methods

Algorithm

The Confab algorithm is outlined in Figure 1. The input required is a 3D structure with reasonable bond lengths and angles. Since the algorithm does not currently

* Correspondence: baoilleach@gmail.com
[1]Analytical and Biological Chemistry Research Facility, University College Cork, Western Road, Cork, Co. Cork, Ireland
Full list of author information is available at the end of the article

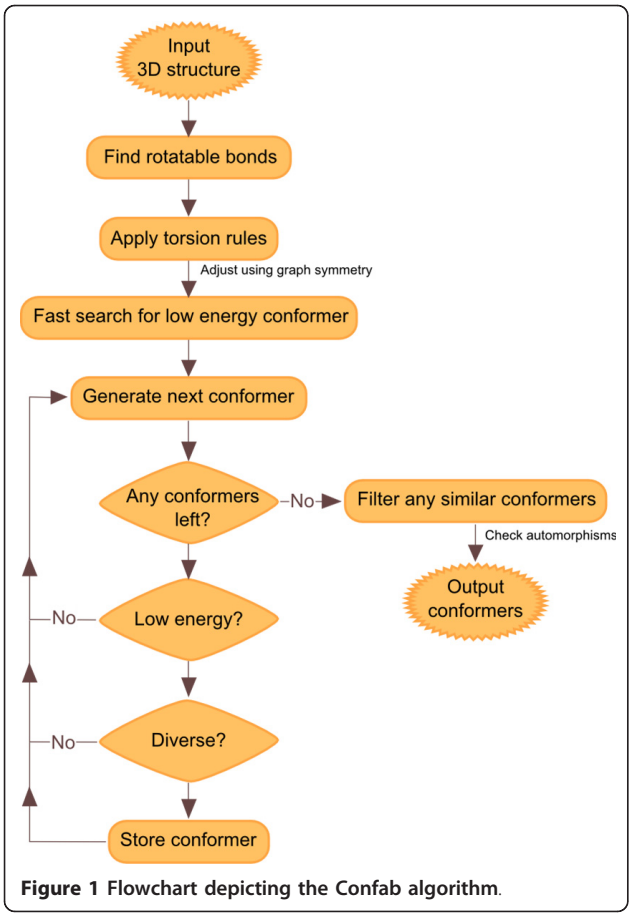

Figure 1 Flowchart depicting the Confab algorithm.

explore ring conformations, any rings present should be in reasonable conformations.

The first step of the algorithm is the identification of rotatable bonds. These are defined as all acyclic single bonds where both atoms of the bond are connected to at least two non-hydrogen atoms, but neither atom of the bond is sp-hybridised. Note that this definition excludes rotation around bonds that interchange hydrogens (for example, the rotation of the hydrogens of a methyl group), but this does not imply any loss of accuracy as it is usual practice to exclude hydrogens when calculating the RMSD (see below).

The method used by Confab to generate conformations is known as the torsion-driving approach. A set of allowed torsion angles for each rotatable bond is assigned to each bond by searching for a match to predefined SMARTS strings in a user-configurable file (torlib.txt) included in the Confab distribution. This file is part of the Open Babel project and it assigns values to particular rotatable bonds using data from Huang et al. [15].

Once the allowed torsion angles are assigned, they are corrected for topological (that is, graph) symmetry. The presence of such symmetry allows performance to be improved by eliminating redundant evaluations, thus reducing the number of conformations that will be tested. 2-fold symmetry is identified when a rotatable bond involves an sp^2 hybridised carbon atom where the neighbouring two atoms affected by the rotation are both of the same symmetry class. When this occurs the allowed values of that torsion are halved by restricting them to those less than 180°. The same is done for the case of 3-fold symmetry at an sp^3 hybridised carbon where the three neighbours are of the same symmetry class; in this case the torsion angles are restricted to those less than 120°. If graph symmetry is identified at both ends of a rotatable bond, the result is multiplicative; a 2-fold and a 3-fold symmetry combine to restrict allowed values of the torsion angles to 360/6 = 60°.

The next step is to obtain an estimate of the energy of the most stable conformer. Throughout Confab, energies are calculated using the MMFF94 forcefield [16]. The values of the bond stretching, angle bend, stretch bend and out-of-plane bending terms are constant for all conformers of the same molecule; only the torsion, Van der Waals and electrostatic terms were repeatedly evaluated. A low energy conformer is found using a simple greedy algorithm. Each torsion angle is optimised starting with the most central torsion and proceeding outwards. As this procedure is relatively fast (compared to the combinatorial problem of searching for the global optimum) it is repeated up to 16 times by testing the four most central torsions in different orders. The lowest energy conformer found is used as a reference point for applying an energy cutoff during the conformer search. If, during the actual conformer generation a lower energy conformer is found, this lower energy is used instead for the reference from that point on.

The main part of the algorithm is the systematic generation and assessment of all conformers described by the allowed torsion angles. Confab generates each of these in turn up to a user-specified cutoff (the default is 10^6) and determines its energy relative to the lowest energy conformer found so far. If this is within a user-specified energy cutoff (50 kcal/mol by default), it is assessed for diversity to the conformers already stored (see below). If it is found to be diverse, it is itself stored otherwise it is discarded. The algorithm then moves onto the next conformer.

Rather than iterate in a 'depth-first' manner over the torsions and their allowed angles, Confab uses a Linear Feedback Shift Register (LFSR) to iterate in a random order over all of the conformers. A LFSR allows the generation of all integers from 1 to N pseudorandomly without repetition and without any memory overhead (which is important for large values of N). By iterating randomly, Confab avoids biasing generated conformers towards a particular region of conformational space, for example towards the input conformation. It also helps

increase diversity if the number of possible conformations is greater than the cutoff for the number tested.

Diversity is ensured by calculating the heavy-atom RMSD (after least-squares alignment) of the newly generated conformation to those previously stored. The alignment is carried out using the QCP algorithm of Theobald [17] (which we found to be about twice as fast as the popular Kabsch alignment method [18]). Despite this, when a molecule has many conformers and a large number of conformers have been stored, full pairwise RMSD calculations take an excessive amount of time. To minimise the number of RMSD evaluations required to discard a conformer, chosen conformers are stored in a tree structure that effectively clusters conformers on-the-fly by RMSD. Figure 2(a) shows a typical 'diversity tree' where each level of the tree is associated with a smaller RMSD diversity from 3.0 Å down to the cutoff specified by the user (1.6 Å in the figure). Each node of the tree represents a stored conformation. Sibling nodes (that is, nodes at the same level that share the same parent node) differ by at least the RMSD diversity associated with that level. Note that sibling nodes are ordered and that the first child node of each parent is the same as the parent itself.

To illustrate the algorithm, let us imagine adding a new conformation H to the tree depicted in Figure 2(a). The algorithm starts at the top of the tree and determines which of the two branches (A or B) to take at the 3.0 Å diversity level. To do so it checks whether H is within 3.0 Å RMSD of A. If so, it follows the tree down to the next level, and checks to see whether it is within 2.0 Å RMSD of A (note that it does not need to recalculate the RMSD to do this). If this is not true, then it checks for 2.0 Å similarity to C. If so, it follows C down to the next level; otherwise it checks against D. If it is not similar to D, H is stored in the tree as the next sibling at that level of the tree (this is depicted in Figure 2 (b)). When adding a new node for a conformation at a particular level, if the level is not at the bottom then child nodes containing that conformation are added at successively lower levels until the bottom level is reached. Overall, there are two possibilities; either the algorithm reaches the bottom level and finds that the new conformation is within the RMSD cutoff of an existing conformer, in which case it is discarded, or else it is of sufficient diversity to be stored at some level of the tree.

This algorithm greatly reduces the number of RMSD evaluations during the conformer generation loop. However it does not eliminate all conformations that are similar to those already stored; conformations may be retained that differ by less than the RMSD cutoff if they end up in different branches. To prune the set of retained conformations, while still avoiding a computationally expensive pairwise RMSD calculation, all of the retained conformations are added one-by-one to a new tree in order of increasing energy. This time the algorithm used for adding conformations to the diversity tree is more robust: all sibling conformations are tested for similarity, even after finding one that is similar. The result is that the same conformation may be added at several different points in the tree. This makes the tree more effective at eliminating similar conformations at the expense of a greater number of RMSD calculations.

Calculation of an RMSD can be overestimated when a molecule's structure has automorphisms (a permutation of the atoms of a molecule that preserves the bond connections). For example, if you consider a para-substituted phenyl ring where two conformations differ by a rotation of 180° around the substituted carbons, it is clear that the calculated RMSD between the conformations should be 0. However, if the symmetry of the phenyl ring is not taken into account this will not be the case and the RMSD will be overestimated as the corresponding atoms of the two structures have moved. The symmetry-corrected RMSD is obtained by iterating over the automorphisms of the molecule and taking the minimum value of the resulting RMSDs. For performance reasons,

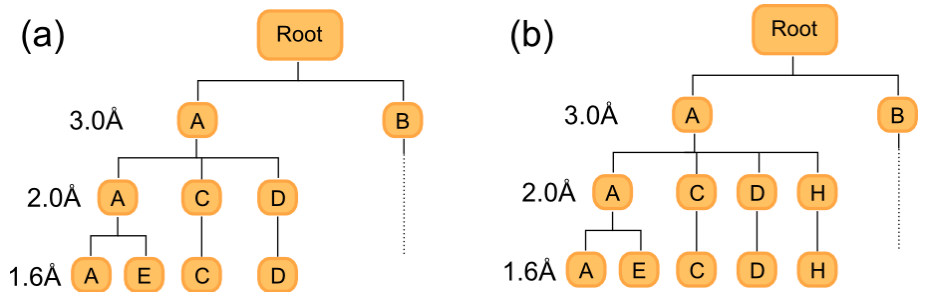

Figure 2 An example diversity tree used to filter conformations on-the-fly. (a) A diversity tree containing five conformations (A to E) used to filter conformations with an RMSD of less than 1.6 Å to one of the stored conformations. (b) The same diversity tree after addition of conformer H, where H is within 3.0 Å of A but not within 2.0 Å of A, C or D.

the calculation of the RMSD is not symmetry-corrected during the main conformation generation loop. However it is used afterwards when building the final diversity tree, thereby eliminating any conformations that were retained in error.

Implementation

Confab is essentially a modified version of Open Babel [19], a widely-used cheminformatics toolkit written in C++ and available under the open source GPL v2 licence [20]. In fact, some of the code written for Confab has been merged into the main Open Babel distribution (such as the original Kabsch alignment code) but due to an additional dependency (on tree.hh, see below) the core code has not been included in Open Babel v2.3.

The MMFF94 forcefield, the conformer generation framework and the automorphism detection are all provided by Open Babel. QCP alignment was implemented using Theobald's public domain code [21] in combination with the Eigen2 high performance linear algebra library [22]. The diversity analysis code relies on a tree data structure provided by the Open Source tree.hh library [23]. The code used to implement the Linear Feedback Shift Register (LFSR) was adapted from its corresponding Wikipedia article [24]. Tap values for the register were taken from Alfke's Xilinx application note [25].

The Confab distribution contains two command-line applications: *confab* and *calcrmsd*. The former implements the Confab algorithm to generate conformers given an input 3D structure, while the latter may be used to assess the performance of *confab* by comparing the generated conformers to a file containing crystal structures. Full details of these applications are available on the Confab website.

Coverage of Conformational Space

Dataset

To illustrate the performance of Confab, we used a dataset of 1000 small molecule crystal structures derived from that of Borodina et al. [26]. The original source is the PDB; thus this dataset represents bioactive conformations of molecules. The 3D structures of the 14504 ligands in the Borodina dataset were obtained using the PubChem Download Service (using the PubChem Substance IDs from Borodina et al.). Of these, 16 could not be handled by the MMFF94 forcefield, 5202 had no rotatable bonds (this fraction included a large number of trivial salts) and 2348 had more than 1 million conformers (according to Confab's torsion rules). 1000 structures were randomly chosen from the 6938 remaining. See Additional file 1 for the structures of these 1000 molecules.

To avoid bias towards the crystal structures, the input conformations for Confab were generated by building the 3D structure using Open Babel. After the initial structure generation, the structures were optimised using the MMFF94 forcefield (200 steps steepest descent). Since Confab does not explore ring conformations, ring conformations were taken from the crystal structure for the initial structure generation. See Additional file 2 for the generated structures.

Results

Figure 3(b) shows an overview of the dataset of 1000 structures in terms of the number of rotatable bonds in each molecule. Although the dataset contains molecules with up to 12 rotatable bonds, it is clear by comparison with the full dataset of Borodina et al. in Figure 3(a) that the reduced dataset is only a representative sample for molecules having up to 7 rotatable bonds. Beyond this, the restriction that the molecule must have fewer than 1 million conformers leads to the elimination of most of the molecules. For this reason, to avoid

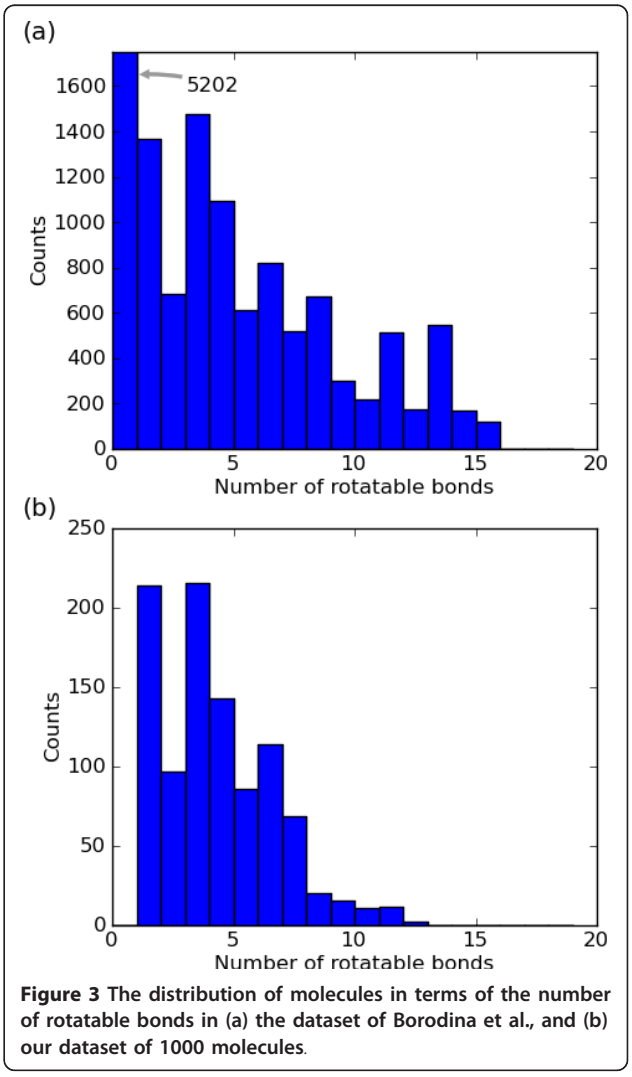

Figure 3 The distribution of molecules in terms of the number of rotatable bonds in (a) the dataset of Borodina et al., and (b) our dataset of 1000 molecules.

erroneous conclusions some of the following analyses (where stated) will not consider molecules having 8 or more rotatable bonds.

Confab was used to exhaustively generate all low energy conformers for each molecule in the dataset for diversity values ranging from 0.4 Å to 3.0 Å RMSD. The default setting of 50 kcal/mol was used as an energy cutoff. The default value of 1 million conformers was used as the conformer cutoff; this ensured exhaustive coverage of conformational space (as defined by Confab's torsion rules) as structures with more conformers were not included in the dataset (see above). Figure 4 shows the mean time for conformer generation per molecule. This is largely independent of the diversity level for diversity levels greater than or equal to 1.0 Å. For values less than this, an increasing amount of time is spent performing the pairwise RMSD calculations against stored conformations.

Performance of conformer generators is typically measured by the percent recovery of crystal structures with respect to a particular RMSD cutoff (see for example Ref [9]). This is simply the percentage of molecules which have a generated conformer within a particular RMSD of the crystal structure. Commonly used values for this RMSD cutoff are 2.0, 1.5 and 1.0 Å.

Figure 5(a) shows the percent recovery at these cutoffs for different values of the RMSD diversity. At 2.0 Å RMSD diversity, 99% are within 2.0 Å RMSD of the crystal (83% within 1.5, 41% within 1.0); at 1.5 Å RMSD diversity, 99% are within 2.0 Å (97% within 1.5, 50% within 1.0); at 1.0 Å RMSD diversity, 99% are within 2.0 Å RMSD (98% within 1.5, 89% within 1.0). As expected, the percentage of crystal structures that are found decreases as the RMSD diversity increases. In particular, the curves fall off steeply once the RMSD diversity is greater than the required cutoff.

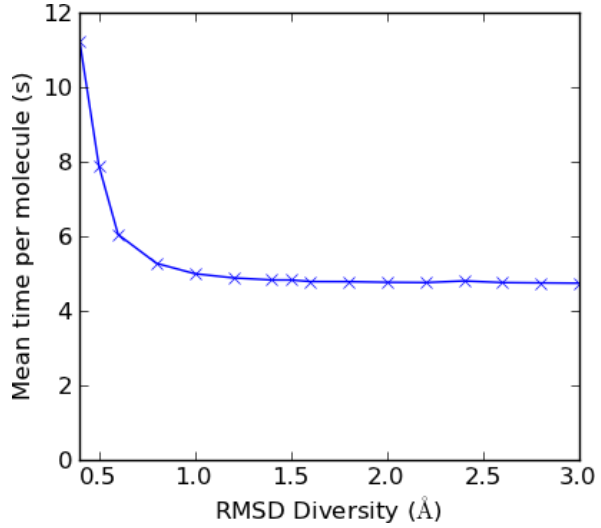

Figure 4 Effect of diversity level on speed of conformer generation. Times were measured on an Intel Xeon E5620 Processor (2.4GHz, 4C) with 32GB RAM.

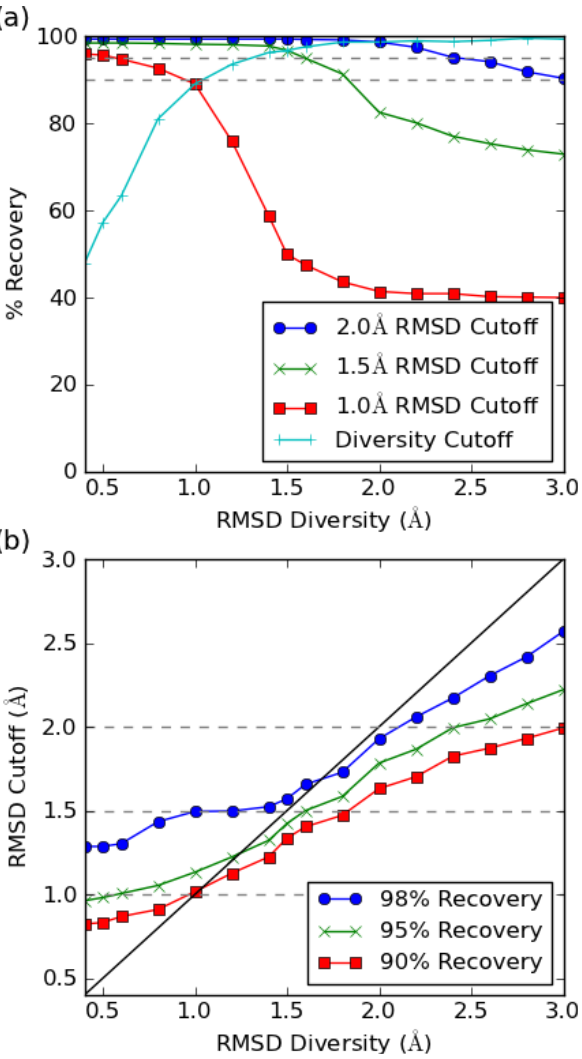

Figure 5 Performance measured as % recovery of crystal structures. (a) Performance for different RMSD cutoffs. The diversity cutoff is where the value of the RMSD diversity is used as the RMSD cutoff. (b) The RMSD cutoff required to achieve a particular level of % recovery. The diagonal line indicates the maximum RMSD cutoff expected when there is complete coverage.

An interesting question to ask is what RMSD diversity is required to recover X% of crystal structures with respect to a particular RMSD cutoff? Figure 5(b) shows the answer to this where X is 90%, 95% or 98%. For example to find 95% of the crystal structures within a 2.0 Å cutoff an RMSD diversity of 2.4 Å (or smaller) is required, but to find the same percentage to within 1.5 Å an RMSD diversity of 1.6 Å is needed. However, even an RMSD diversity of 0.4 Å will not recover 98%

of the structures to within 1.0 Å (it only recovers 96%), an indication of the inherent diversity of the generated conformers as discussed further below.

As pointed out by Borodina et al. [26], if the conformational space is perfectly covered and lacks any 'holes' then the RMSD diversity is an upper bound of the minimum RMSD to the crystal structure. In other words, at an RMSD diversity of 1.5 Å for example, all crystal structures should be found to within 1.5 Å. The diagonal line in Figure 5(b) indicates the maximum RMSD cutoff expected if this ideal behaviour is observed. It is clear from the figure that at low RMSD diversity the actual performance is poorer than this.

There are two main problems that give rise to gaps in conformational coverage. The first is that the allowed torsion values may not encompass the specific torsion angle observed in the crystal structure. For this dataset, there are 7 molecules for which the crystal structure could not be found within 2.0 Å even at 0.4 Å RMSD diversity. These molecules (PubChem substance IDs of 584680, 823881, 825747, 826196, 828032, 830919 and 834618), of which two represent different conformations of the same molecule, all involve sugar moieties and it may be that the allowed torsion angles of the glycosidic bond are too conservative.

The second is that the granularity of the allowed torsion settings may not be sufficiently fine to allow solutions to be found to within a low RMSD cutoff. For example, a carbon-carbon single bond has 12 allowed torsion values from 0 to 360° in increments of 30°. If such a bond is centrally located in a large molecule, even if the crystal structure has similar torsion angles to one of these conformers the RMSD may differ significantly.

Based on this dataset, the inherent granularity of the Confab generated conformers is around 1.4 Å, as indicated by the "Diversity Cutoff" line in Figure 5(a) which falls off sharply as the RMSD diversity decreases below 1.4 Å. This line indicates the percent recovery at different levels of RMSD diversity when the RMSD cutoff used is the same as the diversity level. The sharp fall off below 1.4 Å is a deviation from the ideal behaviour described by Borodina et al.

Table 1 shows the median number of generated conformers tested for molecules with different numbers of rotatable bonds. Broadly speaking, about one third of the conformers pass the energy cutoff applied. Although the size of each individual subset is not very large, and the values for 6 rotatable bonds seem to be biased towards a larger number of conformers, some general points can still be made.

The number of diverse conformers is much reduced by a higher diversity level. For example, for those molecules with 7 rotatable bonds there are approximately 11000 low energy conformers of which about 13% are diverse at 0.5 Å RMSD, only 1.3% are diverse at 1.0 Å RMSD, and only 0.16% are diverse at 1.5 Å RMSD.

The values in Table 1 are in broad agreement with those reported by Smellie et al. [27] for a representative subset of their dataset (see table three therein). They make the point that the number of conformers required to cover conformational space is really surprisingly low. For a molecule with 7 rotatable bonds in our dataset, conformational space can be covered to within 1.0 Å with merely hundreds of conformations while just tens of conformations will achieve a coverage of 1.5 Å. Of course, these figures are expected to increase with each additional rotatable bond.

For completeness, Table 1 also reports median values for the minimum RMSD to the crystal structure. However, as a metric for coverage these values give a misleadingly positive picture compared to the percent recovery values discussed above.

Comparison with Multiconf-DOCK

Multiconf-DOCK [13] is another open source conformer generator that uses a torsion driving approach to implement a systematic search to identify diverse low energy

Table 1 Relationship between the number of rotatable bonds, the number of conformers generated and the minimum RMSD to the crystal structure

Rotatable bonds†	Number of molecules	Total Conformers (median)	Low Energy Conformers (median)	Diverse Conformers (median)					Minimum RMSD to crystal (median)				
				0.5 Å	1.0 Å	1.5 Å	2.0 Å	3.0 Å	0.5 Å	1.0 Å	1.5 Å	2.0 Å	3.0 Å
1	214	3	3	3	1	1	1	1	0.18	0.40	0.45	0.45	0.45
2	97	36	25	8	2	1	1	1	0.34	0.54	0.74	0.80	0.80
3	216	72	44	19	4	1	1	1	0.39	0.70	1.02	1.06	1.06
4	143	1296	582	96	9	2	1	1	0.52	0.80	1.07	1.14	1.24
5	86	3024	1065	189	24	4	1	1	0.60	0.82	1.14	1.31	1.34
6	114	186624	24317	2953	192	24	5	1	0.71	0.90	1.21	1.49	1.78
7	69	34992	10679	1402	139	17	4	1	0.66	0.83	1.14	1.44	1.73

†The 61 molecules with 8 or more rotatable bonds are omitted.

conformers. This software differs in that it uses the AMBER force field [28,29] (as implemented in DOCK5) instead of MMFF94. In addition, it implements performance improvements such as search tree pruning by partial energy estimation [14]. Like Confab, the software requires a 3D structure as input.

Multiconf-DOCK was used to generate conformations for the 1000 structures in the dataset using the same input as for Confab but converted to MOL2 using Open Babel v2.3.0. It should be noted that the specified Sybyl atom types in the input MOL2 file have an effect on the conformations generated by Multiconf-DOCK. The parameters used were taken from the example provided with the Multiconf-DOCK distribution, except that no restriction was placed on the number of generated conformations and the energy cutoff was set to 50 kcal/mol (as used for Confab). Three different RMSD diversity levels were investigated: 2.0 Å, 1.5 Å and 1.0 Å. For all three diversity levels, the mean time spent per molecule was 6.3 s (measured on the same machine used for Figure 4).

The performance in terms of percent recovery is as follows: at 2.0 Å RMSD diversity, 99% are within 2.0 Å RMSD of the crystal structure (89% within 1.5, 55% within 1.0); at 1.5 Å RMSD diversity, 99% are within 2.0 Å (97% within 1.5, 64% within 1.0); at 1.0 Å RMSD diversity, 99% are within 2.0 Å (98% within 1.5, 80% within 1.0). These values are broadly similar to those for Confab (see above). The most noticeable differences occur for the percentage of structures found to within 1.0 Å RMSD; assuming that both programs successfully remove conformations that are within the diversity cutoff, Multiconf-DOCK outperforms Confab at the 2.0 Å and 1.5 Å RMSD diversity levels but Confab performs better at 1.0 Å RMSD diversity.

Table 2 shows the median number of conformers generated by Multiconf-DOCK, along with the minimum RMSD to the crystal structure, broken down by the number of rotatable bonds. Compared to Confab the number of conformers generated is far fewer. It is difficult to say whether this represents a less comprehensive coverage of conformational space or whether this is due to the use of different forcefields. In terms of the minimum RMSD to the crystal structure, once again we see that Multiconf-DOCK performs better than Confab at the 2.0 Å and 1.5 Å RMSD diversity levels but Confab is better at 1.0 Å RMSD diversity.

Distance Distribution in Conformations of a Phenyl Sulfone

Many conformer generators are focused on reproducing bioactive conformations. However it is worth remembering that the generation of conformers may also be useful in other contexts. Here we use Confab to as an aid to interpret the NMR spectra for the phenyl sulfone shown in Figure 6. The peak for the methylene carbon of the ethyl ester was split unexpectedly (compared to an analogous sulfone where the phenyl group was replaced by tert-butyl), and our hypothesis was that this was due to the close approach of the methylene carbon to one of the sulfonyl oxygens in solution. Confab was used to investigate whether low energy conformations existed where the methylene group was in close proximity to a sulfonyl oxygen.

Confab was used to generate a set of conformations of the molecule with a diversity of 0.2 Å and no energy cutoff. The resulting 2014 conformations were optimised using a MMFF94 forcefield (200 steps steepest descent; implemented using Pybel [30]) and the final energy recorded. For each of the conformations the minimum distance between a sulfonyl oxygen and the methylene carbon was measured.

Figure 7 shows a plot of these distances versus the relative energies of the conformers with marginal histograms showing the distribution of values. The methylene carbon does not approach the sulfonyl group very closely. For low energy conformers, the distances are clustered around 4.0 Å and 5.4 Å with the former more frequent. Taking 5 kcal/mol as a cutoff, the distance can be as low as 3.7 Å but shorter distances (down to 3.0 Å)

Table 2 Results for Multiconf-DOCK showing the relationship between the number of rotatable bonds, the number of conformers generated and the minimum RMSD to the crystal structure

Rotatable bonds	Diverse Conformers (median)			Minimum RMSD to crystal (median)		
	1.0 Å	1.5 Å	2.0 Å	1.0 Å	1.5 Å	2.0 Å
1	1	1	1	0.34	0.40	0.40
2	3	1	1	0.50	0.67	0.71
3	2	1	1	0.68	0.78	0.81
4	9	3	1	0.76	0.97	1.05
5	14	4	2	0.85	1.03	1.28
6	43	15	5	1.08	1.23	1.37
7	21	8	3	1.04	1.24	1.40

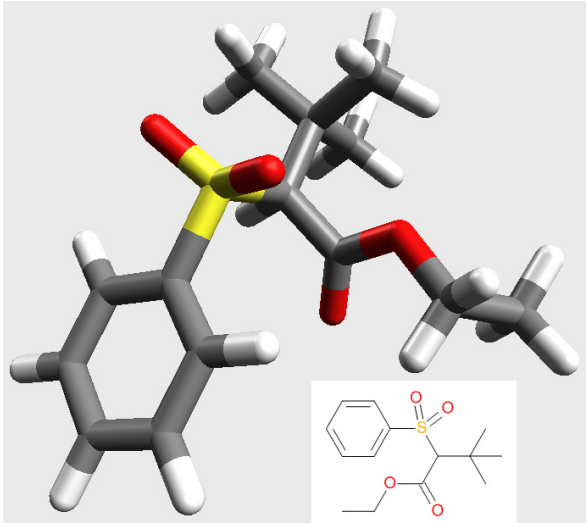

Figure 6 Structure of the phenyl sulfone studied.

are only possible with an associated energy penalty. Figure 6 shows one of the low energy conformations (relative energy of 4.6 kcal/mol) which has a distance of 3.7 Å between the groups of interest.

Conclusion

The goal of this first release of Confab is to ensure complete coverage of all of the low energy conformers of a molecule. While every effort is made to maximise performance, accuracy has been the main goal. Approximations that reduce the search space on the basis of heuristics have been avoided for this reason.

Using the results from Confab 1.0 as a comparison, future work will investigate strategies to to overcome the combinatorial explosion associated with large numbers of rotatable bonds [31] including the trade-off between speed and accuracy.

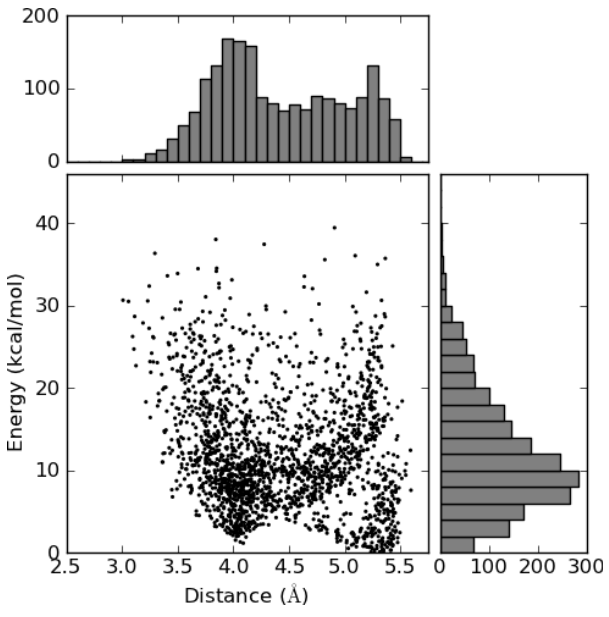

Figure 7 Scatterplot with marginal histograms of distance versus energy for the set of conformations of the phenyl sulfone in Figure 6.

Availability and Requirements

Project name: Confab

Project home page: http://confab.googlecode.com

Operating system(s): Cross-platform

Programming language: C++

Other requirements (if compiling): CMake 2.4+, Eigen2

Licence: GPL v2

Any restrictions to use by non-academics: None

Additional material

Additional file 1: Crystal structures used to test conformational coverage. This is a text file in SDF format containing biological conformations (as downloaded from PubChem) of 1000 molecules. This is a subset of the data used in the study by Borodina et al.

Additional file 2: Generated 3D structures used to test conformational coverage. This is a text file in SDF format containing 3D structures of the 1000 molecules in the dataset generated using Open Babel. These were used as the input to Confab.

Acknowledgements and Funding

NMOB is supported by a Health Research Board Career Development Fellowship, PD/2009/13. We thank several beta testers for their valuable feedback, and the anonymous reviewers for their constructive comments.

Author details

[1]Analytical and Biological Chemistry Research Facility, University College Cork, Western Road, Cork, Co. Cork, Ireland. [2]Open Babel development team. [3]Department of Chemistry, University of Pittsburgh, Chevron Science Center, 219 Parkman Avenue, Pittsburgh, PA 15260, USA.

Authors' contributions

NMOB devised and implemented Confab, and carried out the coverage analysis. GRH implemented the conformer generation framework in Open Babel and contributed to the forcefield code. TV implemented the automorphism code in Open Babel and contributed to the forcefield code. NMOB collaborated with CJF and ARM on the sulfone investigation. All authors read and approved the final manuscript.

Received: 9 February 2011 Accepted: 16 March 2011
Published: 16 March 2011

References

1. Schwab CH: **Conformations and 3D pharmacophore searching.** *Drug Discov Today Tech* 2010, **7**:e245-e253.
2. Sadowski J, Gasteiger J, Klebe G: **Comparison of Automatic Three-Dimensional Model Builders Using 639 X-Ray Structures.** *J Chem Inf Comput Sci* 1994, **34(4)**:1000-1008.
3. Lagorce D, Pencheva T, Villoutreix BO, Miteva MA: **DG-AMMOS: A New tool to generate 3D conformation of small molecules using Distance Geometry and Automated Molecular Mechanics Optimization for in silico Screening.** *BMC Chem Biol* 2009, **9**:6.

4. Gilbert K, Guha R: **smi23d.** [http://www.chembiogrid.org/cheminfo/smi23d/].
5. Hawkins PCD, Skillman AG, Warren GL, Ellingson BA, Stahl MT: **Conformer Generation with OMEGA: Algorithm and Validation using High Quality Structures from the Protein Databank and Cambridge Structural Database.** *J Chem Inf Model* 2010, **50**:572-584.
6. Renner S, Schwab CH, Gasteiger J, Schneider G: **Impact of Conformational Flexibility on Three-Dimensional Similarity Searching Using Correlation Vectors.** *J Chem Inf Model* 2006, **46**:2324-2332.
7. **Catalyst.** Accelrys Inc: San Diego, CA; [http://accelrys.com/].
8. **Confort.** Tripos Inc: St Louis, MO; [http://www.tripos.com/].
9. Watts KS, Dalal P, Murphy RB, Sherman W, Friesner RA, Shelley JC: **ConfGen: A Conformational Search Method for Efficient Generation of Bioactive Conformers.** *J Chem Inf Model* 2010, **50**:534-546.
10. Vainio MJ, Johnson MS: **Generating Conformer Ensembles Using a Multiobjective Genetic Algorithm.** *J Chem Inf Model* 2007, **47**:2462-2474.
11. Sperandio O, Souaille M, Delfaud F, Miteva MA, Villoutreix BO: **MED-3DMC: A new tool to generate 3D conformation ensembles of small molecules with a Monte Carlo sampling of the conformational space.** *Eur J Med Chem* 2009, **44**:1405-1409.
12. Miteva MA, Guyon F, Tufféry P: **Frog2: Efficient 3D conformation ensemble generator for small compounds.** *Nucleic Acids Res* 2010, **38**: W622-W627.
13. Sauton N, Lagorce D, Villoutreix BO, Miteva MA: **MS-DOCK: Accurate multiple conformation generator and rigid docking protocol for multi-step virtual ligand screening.** *BMC Bioinformatics* 2008, **9**:184.
14. Makino S, Kuntz ID: **Automated flexible ligand docking method and its application for database search.** *J Comput Chem* 1997, **18**:1812-1825.
15. Huang N, Shoichet BK, Irwin JJ: **Benchmarking Sets for Molecular Docking.** *J Med Chem* 2006, **49**:6789-6801.
16. Halgren TA: **Merck molecular force field. I. Basis, form, scope, parameterization, and performance of MMFF94.** *J Comp Chem* 1996, **17**:490-519.
17. Theobald DL: **Rapid calculation of RMSDs using a quaternion-based characteristic polynomial.** *Acta Cryst A* 2005, **61**:478-480.
18. Kabsch W: **A solution for the best rotation to relate two sets of vectors.** *Acta Cryst A* 1976, **32**:922-923.
19. Hutchison GR, Morley C, Vandermeersch T, O'Boyle NM, James C, *et al*: **Open Babel, v2.3.** [http://openbabel.org].
20. Free Software Foundation: **GNU General Public License, v2.** [http://www.gnu.org/licenses/old-licenses/gpl-2.0.html].
21. Theobald DL: **QCProt, v1.1.** [http://theobald.brandeis.edu/qcp/].
22. Guennebaud G, Jacob B, *et al*: **Eigen, v2.0.15.** [http://eigen.tuxfamily.org].
23. Peeters K: **tree.hh, v2.65.** [http://tree.phi-sci.com/].
24. **Linear feedback shift register.** *Wikipedia* [http://en.wikipedia.org/wiki/Linear_feedback_shift_register], Retrieved Aug 11, 2010.
25. Alfke P: **Efficient Shift Registers, LFSR Counters, and Long Pseudo-Random Sequence Generators.** *Xilinx application note* 1996 [http://www.xilinx.com/support/documentation/application_notes/xapp052.pdf], XAPP052.
26. Borodina YV, Bolton E, Fontaine F, Bryant SH: **Assessment of Conformational Ensemble Sizes Necessary for Specific Resolutions of Coverage of Conformational Space.** *J Chem Inf Model* 2007, **47**:1428-1437.
27. Smellie A, Kahn SD, Teig SL: **Analysis of Conformational Coverage. 1. Validation and Estimation of Coverage.** *J Chem Inf Comput Sci* 1995, **35(2)**:285-294.
28. Weiner SJ, Kollman PA, Case DA, Singh UC, Ghio C, Alagona G, Profeta S Jr, Weiner P: **A new force field for molecular mechanical simulation of nucleic acids and proteins.** *J Am Chem Soc* 1984, **106**:765-784.
29. Weiner SJ, Kollman PA, Nguyen DT, Case DA: **An all atom force field for simulations of proteins and nucleic acids.** *J Comput Chem* 1986, **7**:230-252.
30. O'Boyle NM, Morley C, Hutchison GR: **Pybel: a Python wrapper for the OpenBabel cheminformatics toolkit.** *Chem Cent J* 2008, **2**:5.
31. Beusen DD, Shands EFB, Karasek SF, Marshall GR, Dammkoehler RA: **Systematic search in conformational analysis.** *J Mol Struct THEOCHEM* 1996, **370**:157-171.

doi:10.1186/1758-2946-3-8
Cite this article as: O'Boyle *et al.*: Confab - Systematic generation of diverse low-energy conformers. *Journal of Cheminformatics* 2011 **3**:8.

Journal of Cheminformatics

BOOK REPORT Open Access

Review of "Data Analysis with Open Source Tools" by Philipp K Janert

Noel M O'Boyle

Book details Janert PK: Data Analysis with Open Source Tools Sebastopol, CA: O'Reilly Media 2010

Cheminformatics has been defined as the application of informatics methods to solve chemical problems [1]. Such chemical problems are often represented in terms of data, be it activity data for a series of compounds or descriptor values for a compound library. While this new book from the O'Reilly stable is not aimed specifically at cheminformaticians, the subtitle of "A Hands-On Guide for Programmers and Data Scientists" makes it clear that the target audience includes any scientists whose day-to-day work involves analysing and interpreting data.

The book is broadly divided into four parts on Graphics: Looking at Data, Analytics: Modeling Data, Computation: Mining Data and Applications: Using Data. First of all, it should be noted that this is not a book about statistics (as Chapter 1 states explicitly). Neither is it a manual for numpy, Sage, matplotlib, Gnuplot, R and so forth, as might be implied by the title. Instead, Janert focuses on discussing data analysis methods and techniques in depth, rather than skimming topics by following a cookbook or tutorial approach linked to particular software. This is as it should be - there are already documentation and manuals available for all of these programs, and the reader is simply alerted to the availability of the software, its capabilities are described and some examples of use shown.

This is a real practitioner's book. Janert, a former physicist and software engineer, is a consultant in data analysis and mathematical modelling. He has taken his hard-won knowledge and tried to get it all down on paper for the reader's benefit. For example, in a chapter with the provocative title of "What you really need to know about classical statistics" he explains why introductory statistics textbooks seem to cover methods and topics at odds with the problems data analysts deal with day-to-day; essentially classical methods were developed at a time of small and expensive datasets and no computational power, and hypothesis testing focused on determining whether an effect existed. Today we have ample computing power and may be dealing with very large datasets; also, we are usually more interested in the size of an effect (practical significance) rather than just whether it exists (statistical significance).

Topics that could not be squeezed into a chapter proper have been placed in shorter "Intermezzos" at the end of each section. For example, a short section on "What about map/reduce?" at the end of "Mining Data" reminds the reader that the map/reduce methodology (much hyped recently) is not a clever algorithm to speed things up, but rather a piece of infrastructure that makes it convenient to implement algorithms that are trivially parallelisable.

On the negative side, any cheminformatician who has been involved with QSAR studies will already be familiar with the multivariate analysis methods discussed here (Chapters 13 and 14), although I liked the observation that "you will actually spend *more* time on data sets that are totally worthless" in relation to clustering algorithms. Also there are two chapters (out of 19) which will be of little interest as they focus on business intelligence and financial calculations, although even there the reader will find an introduction to the use of Berkeley DB and SQLite from Python, tools which I highly recommend. There are also cases where the author perhaps gives too much detail, but this is hardly a criticism - in a book of some 500 pages there is plenty of room.

Overall though, I heartily recommend this book to anyone working in cheminformatics whether they develop methods or apply them. Too often we rely on summary statistics such as mean and standard deviation and forget to actually look at the data. Graphical analysis gives you a feel for the data, and can often highlight problems, interesting features, or mistaken assumptions. After reading this book, you should be very aware of both the advantages and pitfalls of a wide variety of

Correspondence: baoilleach@gmail.com
Analytical and Biological Chemistry Research Facility, University College Cork, Western Road, Cork, Co. Cork, Ireland

analysis methods but you will also be reminded that the goal of data analysis is not a picture or a number but insight.

Competing interests
The author declares that they have no competing interests.

Received: 8 March 2011 Accepted: 24 March 2011
Published: 24 March 2011

Reference

1. Gasteiger J: **Introduction.** In *Chemoinformatics - A Textbook.* Edited by: Gasteiger J, Engel T. Weinheim: Wiley-VCH; 2003:1-13.

doi:10.1186/1758-2946-3-10
Cite this article as: O'Boyle: Review of "Data Analysis with Open Source Tools" by Philipp K Janert. *Journal of Cheminformatics* 2011 **3**:10.

Journal of Cheminformatics

RESEARCH ARTICLE Open Access

Open Data, Open Source and Open Standards in chemistry: The Blue Obelisk five years on

Noel M O'Boyle[1*], Rajarshi Guha[2], Egon L Willighagen[3], Samuel E Adams[4], Jonathan Alvarsson[5], Jean-Claude Bradley[6], Igor V Filippov[7], Robert M Hanson[8], Marcus D Hanwell[9], Geoffrey R Hutchison[10], Craig A James[11], Nina Jeliazkova[12], Andrew SID Lang[13], Karol M Langner[14], David C Lonie[15], Daniel M Lowe[4], Jérôme Pansanel[16], Dmitry Pavlov[17], Ola Spjuth[5], Christoph Steinbeck[18], Adam L Tenderholt[19], Kevin J Theisen[20] and Peter Murray-Rust[4]

Abstract

Background: The Blue Obelisk movement was established in 2005 as a response to the lack of Open Data, Open Standards and Open Source (ODOSOS) in chemistry. It aims to make it easier to carry out chemistry research by promoting interoperability between chemistry software, encouraging cooperation between Open Source developers, and developing community resources and Open Standards.

Results: This contribution looks back on the work carried out by the Blue Obelisk in the past 5 years and surveys progress and remaining challenges in the areas of Open Data, Open Standards, and Open Source in chemistry.

Conclusions: We show that the Blue Obelisk has been very successful in bringing together researchers and developers with common interests in ODOSOS, leading to development of many useful resources freely available to the chemistry community.

Background

The Blue Obelisk movement was established in 2005 at the 229th National Meeting of the American Chemistry Society as a response to the lack of Open Data, Open Standards and Open Source (ODOSOS) in chemistry. While other scientific disciplines such as physics, biology and astronomy (to name a few) were embracing new ways of doing science and reaping the benefits of community efforts, there was little if any innovation in the field of chemistry and scientific progress was actively hampered by the lack of access to data and tools. Since 2005 it has become evident that a good amount of development in open chemical information is driven by the demands of neighbouring scientific fields. In many areas in biology, for example, the importance of small molecules and their interactions and reactions in biological systems has been realised. In fact, one of the first free and open databases and ontologies of small molecules was created as a resource about chemical structure and nomenclature by biologists [1].

The formation of the Blue Obelisk group is somewhat unusual in that it is not a funded network, nor does it follow the industry consortium model. Rather it is a grassroots organisation, catalysed by an initial core of interested scientists, but with membership open to all who share one or more of the goals of the group:

- **Open Data in Chemistry**. One can obtain all scientific data in the public domain when wanted and reuse it for whatever purpose.
- **Open Standards in Chemistry**. One can find visible community mechanisms for protocols and communicating information. The mechanisms for creating and maintaining these standards cover a wide spectrum of human organisations, including various degrees of consent.
- **Open Source in Chemistry**. One can use other people's code without further permission, including changing it for one's own use and distributing it again.

* Correspondence: baoilleach@gmail.com
[1]Analytical and Biological Chemistry Research Facility, Cavanagh Pharmacy Building, University College Cork, College Road, Cork, Co. Cork, Ireland
Full list of author information is available at the end of the article

Note that while some may advocate also for Open Access to publications, the Blue Obelisk goals (ODOSOS) focus more on the availability of the underlying scientific data, standards (to exchange data), and code (to reproduce results). All three of these goals stem from the fundamental tenants of the scientific method for data sharing and reproducibility.

The Blue Obelisk was first described in the CDK News [2] and later as a formal paper by Guha et al. [3] in 2006. Its home on the web is at http://blueobelisk.org. This contribution looks back on the work carried out by the Blue Obelisk over the past 5 years in the areas of Open Data, Open Source, and Open Standards in chemistry.

Scope

The Blue Obelisk covers many areas of chemistry and chemical resources used by neighbouring disciplines (*e.g.* biochemistry, materials science). Many of the efforts relate to cheminformatics (the scope of this journal) and we believe that many of the publications in Journal of Cheminformatics could be completely carried out using Blue Obelisk resources and other Open Source chemical tools. The importance of this is that for the first time it would allow reviewers, editors and readers to validate assertions in the journal and also to re-run and re-analyse parts of the calculation.

However, Blue Obelisk software and data is also used outside cheminformatics and certainly in the five main areas that, for example, Chemical Markup Language (CML) [4] supports:

1. **Molecules:** This is probably the largest area for Blue Obelisk software and data, and is reflected by many programs that visualise, transform, convert formats and calculate properties. It is almost certain that any file format currently in use can be processed by Blue Obelisk software and that properties can be calculated for most (organic compounds).
2. **Reactions:** Blue Obelisk software can describe the semantics of reactions and provide atom-atom matching and analyse stoichiometric balance in reactions.
3. **Computational chemistry:** Blue Obelisk software can interpret many of the current output files from calculations and create input for jobs. The Quixote project (see below and elsewhere in this issue) shows that Open Source approaches based on Blue Obelisk resources and principles are increasing the availability and re-usability of computational chemistry.
4. **Spectra:** 1-D spectra (NMR, IR, UV etc.) are fully supported in Blue Obelisk offerings for conversion and display. There is a limited amount of spectral analysis but the software gives a platform on which it should be straightforward to develop spectral annotation and manipulation. However, currently the Blue Obelisk lacks support for multi-dimensional NMR and multi-equipment spectra (*e.g.* GC-MS).
5. **Crystallography:** The Blue Obelisk software supports the bi-directional processing of crystal structure files (CIF) and also solid-state calculations such as plane-waves with periodic boundary conditions. There is considerable support for the visualisation of both periodic and aperiodic condensed objects.

Many of the current operations in installing and running chemical computations and using the data are integration and customisation rather than fundamental algorithms. It is very difficult to create universal platforms that can be distributed and run by a wide range of different users, and in general, the Blue Obelisk deliberately does not address these. Our approach is to produce components that can be embedded in many environments, from stand-alone applications to web applications, databases and workflows. We believe that a chemical laboratory with reasonable access to common software engineering techniques should be able to build customised applications using Blue Obelisk components and standard infrastructure such as workflows and databases. Where the Blue Obelisk itself produces data resources they are normally done with Open components so that the community can, if necessary, replicate them.

Much of the impetus behind Blue Obelisk software is to create an environment for chemical computation (including cheminformatics) where all of the components, data, specifications, semantics, ontology and software are Openly visible and discussable. The largest current uses by the general chemical community are in authoring, visualisation and cheminformatics calculations but we anticipate that this will shortly extend into mainstream computational chemistry and solid-state. Although many of the authors are employed as research scientists, there are also several people who contribute in their spare time and we anticipate an increasing value and use of the Blue Obelisk in education at all levels.

Open Source

The development of Open Source software has been one of the most successful of the Blue Obelisk's activities. The following sections describe recent work in this area, and Table 1 provides an overview of the projects discussed and where to find them online.

Cheminformatics toolkits

Open Source toolkits for cheminformatics have now existed for nearly ten years. During this period, some toolkits were developed from scratch in academia,

Table 1 Blue Obelisk Open Source Software projects discussed in the text

Name	Website
CML Tools	
CMLXOM	https://bitbucket.org/wwmm/cmlxom/
JUMBO	http://sourceforge.net/projects/cml/
Cheminformatics Toolkits	
Chemistry Development Kit (CDK)	http://cdk.sf.net
Cinfony	http://cinfony.googlecode.com
Indigo	http://ggasoftware.com/opensource/indigo
JOELib	http://sf.net/projects/joelib
Open Babel	http://openbabel.org
RDKit	http://rdkit.org
Web Applications	
ChemDoodle Web Components	http://web.chemdoodle.com
Jmol	http://jmol.org
Integration	
Bioclipse	http://www.bioclipse.net
CDK-Taverna	http://cdktaverna.wordpress.com
Lensfield2	https://bitbucket.org/sea36/lensfield2/
Interconversion	
CIFXOM [95]	https://bitbucket.org/wwmm/cifxom/
JUMBO-Converters	https://bitbucket.org/wwmm/jumbo-converters/
OPSIN	http://opsin.ch.cam.ac.uk
OSRA	http://osra.sf.net
Structure Databases	
Bingo	http://ggasoftware.com/opensource/bingo
Chempound (Chem#)	https://bitbucket.org/chempound
Mychem	http://mychem.sf.net
OrChem	http://orchem.sf.net
pgchem	http://pgfoundry.org/projects/pgchem/
Text mining	
ChemicalTagger [96]	http://chemicaltagger.ch.cam.ac.uk/
OSCAR4	https://bitbucket.org/wwmm/oscar4/
Computational Chemistry	
Avogadro	http://avogadro.openmolecules.net
cclib	http://cclib.sf.net
GaussSum	http://gausssum.sf.net
QMForge	http://qmforge.sf.net
Computational Drug Design	
Confab [97]	http://confab.googlecode.com
Pharao	http://silicos.be/download
Piramid	http://silicos.be/download
Sieve	http://silicos.be/download
Stripper	http://silicos.be/download
Other Applications	
AMBIT2	http://ambit.sf.net
Brunn	http://brunn.sf.net
Toxtree	http://toxtree.sf.net
XtalOpt	http://xtalopt.openmolecules.net

whereas others were made Open Source by releasing in-house codebases under liberal licenses. When the Blue Obelisk was established five years ago, the primary toolkits under active development were the Chemistry Development Kit (CDK) [5,6], Open Babel [7], and JOELib [8]. Of these, both the CDK and Open Babel continue to be actively developed.

The CDK project has been under regular development over the last five years. Several features have been implemented ranging from core components such as an extensible SMARTS matching system and a new graph (and subgraph) isomorphism method [9], to more application oriented components such as 3D pharmacophore searching and matching, and a variety of structural-key and hashed fingerprints. In addition, there have been a number of second generation tools developed on top of the CDK (see below). As well as the use of the CDK in various tools, it has been deployed in the form of web services [10] and has formed the basis of a variety of web applications.

Since 2006, major new features of Open Babel include 3D structure generation and 2D structure-diagram generation, UFF and MMFF94 forcefields, and significantly expanded support for computational chemistry calculations. In addition, a major focus of Open Babel development has been to provide for accurate conversion and representation in areas of stereochemistry, kekulisation, and canonicalisation. The project has also grown, in terms of new contributors, new support from commercial companies, and second-generation tools applying Open Babel to a variety of end-user applications, from molecular editors to chemical database systems.

Two new Open Source cheminformatics toolkits have appeared since the original paper. In 2006 Rational Discovery, a cheminformatics service company (since closed down), released RDKit [11] under the BSD License. This is a C++ library with Python and (more recently) Java bindings. RDKit is actively developed and includes code donated by Novartis. Recent developments include the Java bindings, as well as performance improvements for its database cartridge.

More recently, GGA Software Services (a contract programming company) released the Indigo toolkit [12] and associated software in 2009 under the GPL. Indigo is a C++ library with high-level wrappers in C, Java, Python, and the .NET environment. Like RDKit and other toolkits, Indigo provides support for tetrahedral and cis-trans stereochemistry, 2D coordinate generation, exact/substructure/SMARTS matching, fingerprint generation, and canonical SMILES computation. It also provides some less common functionality, like matching tautomers and resonance substructures, enumeration of subgraphs, finding maximum common substructure of *N* input structures, and enumerating reaction products.

Second-generation tools

Although feature-rich and robust cheminformatics toolkits are useful in and of themselves, they can also be seen as providing a base layer on which additional tools and applications can be built. This is one of the reasons that cheminformatics toolkits are so important to the open source 'ecosystem'; their availability lowers the barrier for the development of a 'second generation' of chemistry software that no longer needs to concern itself with the low-level details of manipulating chemical structures, and can focus on providing additional functionality and ease-of-use. Although a wide range of chemistry software has been built using Blue Obelisk components (see for example, the "Related Software" link on the Open Babel website, [13] listing over 40 projects as of this writing, or "Software using CDK" at the CDK website), in this section we focus on second-generation tools which themselves have been developed by members of the Blue Obelisk.

Bioclipse [14] (v2.4 released in Aug 2010) and Avogadro [15] (v1.0 in Oct 2009) are two examples of such software, based on the CDK and Open Babel, respectively. Bioclipse (Figure 1) is an award-winning molecular workbench for life sciences that wraps cheminformatics functionality behind user-friendly interfaces and graphical editors while Avogadro (Figure 2) is a 3D molecular editor and viewer aimed at preparing and analysing computational chemistry calculations. Both projects are designed to be extended or scripted by users through the provision of a plugin architecture and scripting support (using Bioclipse Scripting Language [16], or Python in the case of Avogadro). An interesting aspect of both Avogadro and Bioclipse is that they share some developers with the underlying toolkits and this has driven the development of new features in the CDK and Open Babel.

Both products in turn act as extensible platforms for other software. Bioclipse, for example is used by software such as Brunn [17], a laboratory information system for microplate based high-throughput screening. Brunn provides a graphical interface for handling different plate layouts and dilution series and can automatically generate dose response curves and calculate IC_{50}-values. Avogadro is used by Kalzium [18], a periodic table and chemical editor in KDE, and XtalOpt [19,20], an evolutionary algorithm for crystal structure prediction. XtalOpt provides a graphical interface using Avogadro and submits calculations using a range of solid-state simulation software to predict stable polymorphs.

A final example of second-generation Blue Obelisk software is the AMBIT2 [21,22] software, which was developed to facilitate registration of chemicals for the REACH EU directive, and is based on the CDK. It was distributed initially as a standalone Java Swing GUI, and

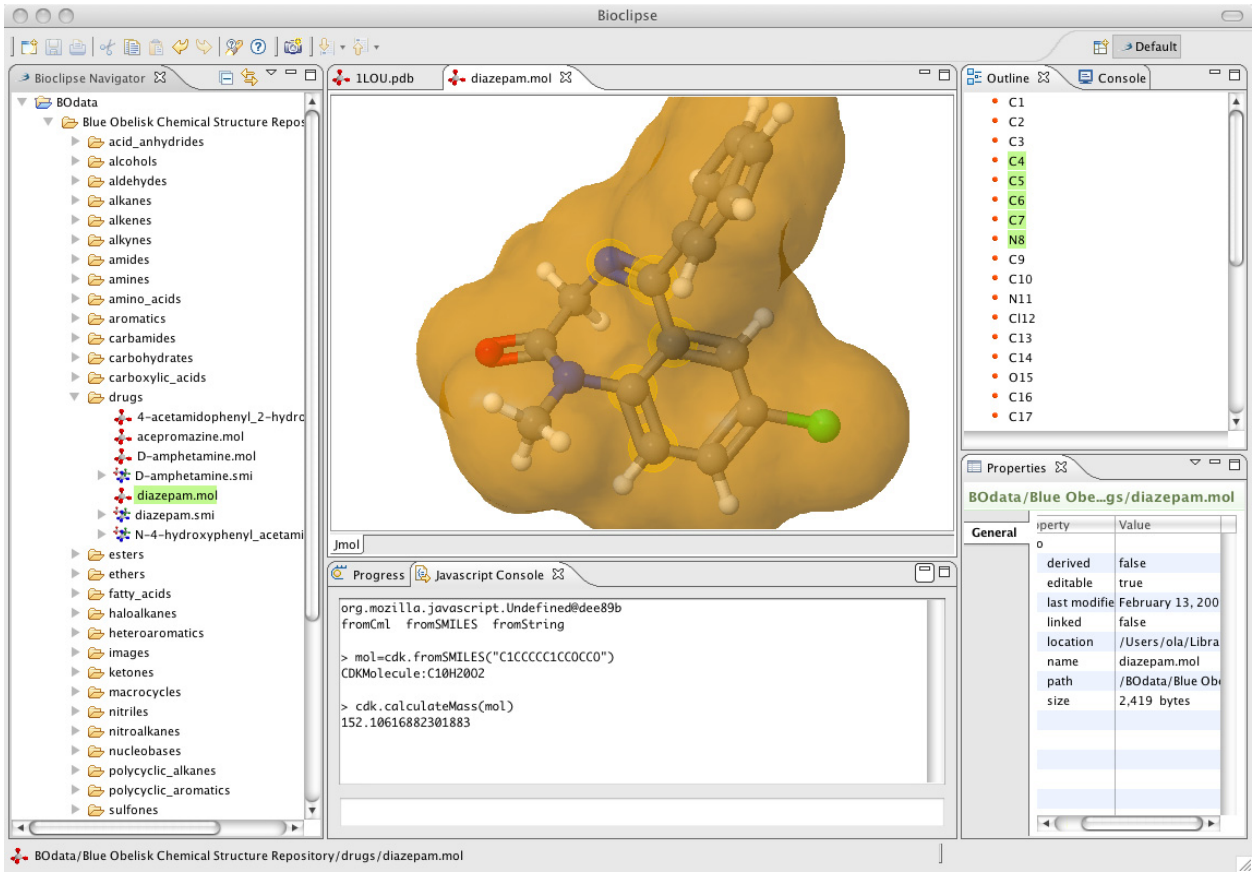

Figure 1 Screenshot of Bioclipse using Jmol to visualise a molecular surface.

more recently as downloadable web application archive, offering a web services interface to a searchable chemical structures database. Also integrated are descriptor calculations, as well as the ability to run and build predictive models, including modules of the open source Toxtree [22-24] software for toxicity prediction.

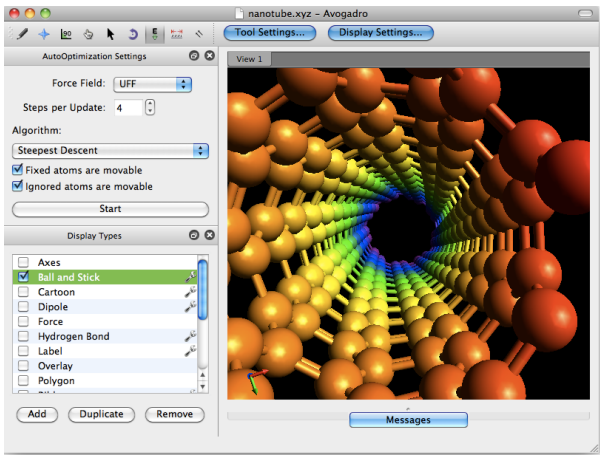

Figure 2 Screenshot of Avogadro showing a depiction of a carbon nanotube.

Computational chemistry analysis

Another area where the Blue Obelisk has had a significant impact in the past five years is in supporting quantum chemistry calculations and in interpreting their results. Electronic structure calculations have a long tradition in the chemistry community and a variety of programs exist, mostly proprietary software but with an increasing number of open source codes. However, since each program uses different input formats, and the the output formats vary widely (sometimes even varying between different versions of the same software), preparing calculations and automatically extracting the results is problematic.

Avogadro has already been mentioned as a GUI for preparing calculations. It uses Open Babel to read the output of several electronic structure packages. Avogadro generates input files on the fly in response to user input on forms, as well as allowing inline editing of the files before they are saved to disk. It also features intuitive syntax highlighting for GAMESS input files,

allowing expert users to easily spot mistakes before saving an input file to disk.

In addition to this, significant development of new parsing routines took place in an Avogadro plugin to read in basis sets and electronic structure output in order to calculate molecular orbital and electron density grids. This code was written to be parallel, using desktop shared memory parallelism and high level APIs in order to significantly speed up analysis. Most of this code was recently separated from the plugin, and released as a BSD licensed library, OpenQube, which is now used by the latest version of Avogadro. Jmol (see below) can also depict computational chemistry results including molecular orbitals.

In 2006, the Blue Obelisk project cclib [25] was established with the goal of parsing the output from computational chemistry programs and presenting it in a standard way so that further analyses could be carried out independently of the quantum package used. cclib is a Python library, and the current version (version 1.0.1) supports 8 different computational chemistry codes and extracts over 30 different calculated attributes. Two related Blue Obelisk projects build upon cclib. GaussSum [26], is a GUI that can monitors the progress of SCF and geometry convergences, and can plot predicted UV/Vis absorption and infrared spectra from appropriate logfiles containing energies and oscillator strengths for easy comparison to experimental data. QMForge [27] provides a GUI for various electronic structure analyses such as Frenking's charge decomposition analysis [28] and Mulliken or C-squared analyses on user-defined molecular fragments. QMForge also provides a rudimentary Cartesian coordinate editor allowing molecular structures to be saved via Open Babel.

The Quixote project epitomises the full use of the Blue Obelisk software and is described in detail in another article in this issue. Here we observe that it is possible to convert legacy chemistry file formats of all sorts into semantic chemistry and extract those parts which are suitable for input to computational chemistry programs. This chemistry is then combined with generic concepts of computational chemistry (*e.g.* strategy, machine resources, timing, accuracy etc.) into the legacy inputs for a wide range of programs. Quixote itself follows Blue Obelisk principles in that it does not manage the submission and monitoring of jobs but resumes action when the jobs have been completed, and then applies a range of parsing and transformation tools to create standardised semantic chemical content. A major feature of Quixote is that it requires all concepts to validate against dictionaries and the process of parsing files necessarily generates communally-agreed dictionaries, which represent an important step forward in the Open specifications for Blue Obelisk. When widely-deployed, Quixote will advertise the value of Open community standards for semantics to the world.

The Quixote project is not dependent on any particular technology, other than the representation of computational chemistry in CML and the management of semantics through CML dictionaries. At present, we use JUMBO-Converters [29] for most of the semantic conversion, Lensfield2 [30] for the workflow and Chempound (chem#) [31] to store and disseminate the results.

Web applications

While desktop software has composed the majority of scientific tools since the computer was introduced, the internet continues to change how applications and content are distributed and presented. The web presents new opportunities for scientists as it is an open and free medium to distribute scientific knowledge, ideas and education. Web applications are software that runs within the browser, typically implemented in Java or JavaScript. Recently, a new version of the HTML specification, HTML5, defined a well-developed framework for creating native web applications in JavaScript and this opens up new possibilities for visualising chemical data.

Jmol, the interactive 3D molecular viewer, is one of the most widely used chemistry applets, and indeed has seen widespread use in other fields such as biology and even mathematics (it is used for 3D depiction of mathematical functions in the Sage Mathematics Projects [32]). It is implemented in Java, and has gone from being a "Rasmol/Chime" replacement to a fully fledged molecular visualisation package, including full support for crystallography [33], display of molecular orbitals from standard basis set/coefficient data, the inclusion of dynamic minimisation using the UFF force field, and a full implementation of Daylight SMILES and SMARTS, with extensions to conformational and biomolecular substructure searching (Jmol BioSMARTS).

In 2009, iChemLabs released the ChemDoodle Web Components library [34] under the GPL v3 license (with a liberal HTML exception). This library is completely implemented in JavaScript and uses HTML5 to allow the scientist to present publication quality 2D and 3D graphics (see Figure 3) and animations for chemical structures, reactions and spectra. Beyond graphics, this tool provides a framework for user interaction to create dynamic applications through web browsers, desktop platforms and mobile devices such as the iPhone, iPad and Android devices.

The business end

Open Source provides a unique opportunity for commercial organisations to work with the cheminformatics community. Traditional business models rely on monetisation of source code, causing companies to repeat work

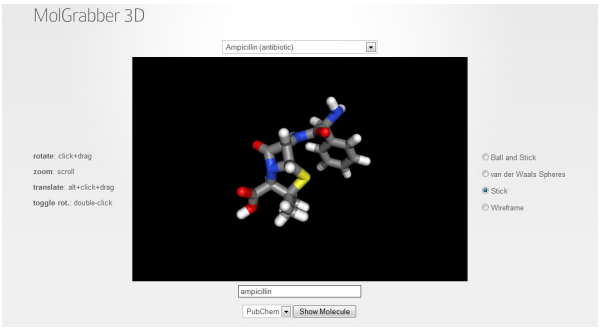

Figure 3 Screenshot of the MolGrabber 3D demo from ChemDoodle Web Components.

done by other companies. This model is sometimes combined with a free (gratis) model for people working at academic institutes, to increase adoption and encourage contributions from academics. This solution defines the return on investment as the IP on the software, but has the downside of investment losses due to duplication of software and method development, which become visible when proprietary companies merge. Some authors have argued that in the chemistry field few contributors are available to volunteer time to improve codes and IP considerations may prevent contributions from industry [35]. If true, this would hamper adoption of Open Source and Open Data in chemistry, and greatly slow the growth of projects such as those in the Blue Obelisk.

The Blue Obelisk community, however, takes advantage of the fact that much of the investment needed for development is either paid for by academic institutes and funding schemes, or by volunteers investing time and effort. In return, contributors get full access to the source code, and the Open Source licensing ensures that they will have access any time in the future. In this way, the license functions as a social contract between everyone to arrange an immediate return on investment. Effiectively, this approach shares the burden of the high investment in having to develop cheminformatics software from scratch, allowing researchers and commercial partners alike to focus on their core business, rather than the development of prerequisites. In the case of the Blue Obelisk, the rich collection of Open Source cheminformatics tools provided greatly reduces investment up front for new companies in the cheminformatics market. Such advantages have also been noted in the drug discovery field [36-38].

The use of Open Standards allows everyone to select those Blue Obelisk components they find most useful, as they can easily replace one component with another providing the same functionality, taking advantage that they use the same standards for, for example, data exchange. This way, licensing issues are becoming a marginal problem, allowing companies to select a license appropriate for their business model. This too, allows a company to create a successful product with significantly reduced cost and effort.

At the time of writing there are many commercial companies developing chemistry solutions around Open Source cheminformatics components provided by the Blue Obelisk community. Examples of such companies include iChemLabs, IdeaConsult, Wingu, Silicos, GenettaSoft, eMolecules, hBar, Metamolecular, and Inkspot Science. Some of these merely use components, but several actively contribute back to the Blue Obelisk project they use, or donate new Open Source cheminformatics projects to the community.

For example, iChemLabs released the ChemDoodle Web Components library under the GPL v3 license, based on the upcoming HTML5 Open Standard. It allows making web and mobile interfaces for chemical content. The project is already being adopted by others, including iBabel [39], ChemSpotlight [40] and the RSC ChemSpider [41,42].

Silicos has released several Open Source utilities [43] based on Open Babel, such as Pharao, a tool for pharmacophore searching, Sieve for filtering molecular structure by molecular property, Stripper for removing core scaffold structures from a molecule set, and Piramid for molecular alignment using shape determined by the Gaussian volumes as a descriptor. Additionally, contributions have been made to the Open Babel project itself.

Other companies use Blue Obelisk components and contribute patches, smaller and larger. For example, IXELIS donated the isomorphism code in the CDK, eMolecules donated canonicalisation code to Open Babel, Metamolecular improved the extensibility and unit testing suite of OPSIN, and AstraZeneca contributed code to the CDK for signatures. This is just a very minor selection, and the reader is encouraged to contact the individual Blue Obelisk projects for a detailed list.

In May 2011, a Wellcome Trust Workshop on Molecular Informatics Open Source Software (MIOSS) explored the role of Open Source in industrial laboratories and companies as well as academia (several of the presenters are among the authors of this paper). The meeting identified that Open Source software was extremely valuable to industry not just because it is available for free, but because it allows the validation of source code, data and computational procedures. Some of the discussion was on business models or other ways to maintain development of Open Source software on which a business relied. Companies are concerned about training and support and, in some cases, product liability. There are difficulties for software for which there is

no formal transaction other than downloading and agreeing to license terms. One anecdote concerned a company that wished to donate money to an Open Source project but could not find a mechanism to do so.

Industry participants also pointed out that there is a considerable amount of contribution-in-kind from industry, both from enhancements to software and also the development of completely new software and toolkits. Companies are now finding it easier to create mechanisms for releasing Open Source software without violating confidentiality or incurring liability. A phrase from the meeting summed it up: "The ice is beginning to melt", signifying that we can expect a rapid increase in industry's interest in Open Source.

Converting chemical names and images to structures

The majority of chemical information is not stored in machine-readable formats, but rather as chemical names or depictions. The OSRA and OPSIN projects focus on extracting chemical information from these sources. Such software plays a particularly important role for data mining the chemical literature, including patents and theses.

Optical Structure Recognition Application (OSRA) [44] was started in early 2007 with the goal to create the first free and open source tool for extraction and conversion of molecular images into SMILES and SD files. From the very beginning the underlying philosophy was to integrate existing open source libraries and to avoid "reinventing the wheel" wherever possible. OSRA relies on a variety of open source components: Open Babel for chemical format conversion and molecular property calculations, GraphicsMagick for image manipulation, Potrace for vectorisation, GOCR and OCRAD for optical character recognition. The growing importance of image recognition technology can be seen in the fact that only a few years ago there was only one widely available software package for chemical structure recognition - CLiDE (commercially developed at Keymodule, Ltd), but today there are as many as seven available programs.

OPSIN (Open Parser for Systematic IUPAC Nomenclature) [45] focuses instead on interpreting chemical names. The chemical name is the oldest form of communication used to describe chemicals, predating even the knowledge of the atomic structure of compounds. Chemical names are abundant in the scientific literature and encode valuable structural information. Through successive books of recommendations [46,47], IUPAC has tried to codify and to an extent standardise naming practices. OPSIN aims to make this abundance of chemical names machine readable by translating them to SMILES, CML or InChI. The program is based around the use of a regular grammar to guide tokenisation and parsing of chemical names, followed by step-wise application of nomenclature rules. It is able to offer fast and precise conversions for the majority of names using IUPAC organic nomenclature, and is available as a web service, Java library and standalone application for maximum interoperability.

Chemical database software

Registration, indexing and searching of chemical structures in relational databases is one of the core areas of cheminformatics. A number of structure registration systems have been published in the last five years, exploiting the fact that Open Source cheminformatics toolkits such as Open Babel and the CDK are available. OrChem [48], for example, is an open source extension for the Oracle 11G database that adds registration and indexing of chemical structures to support fast substructure and similarity searching. The cheminformatics functionality is provided by the CDK. OrChem provides similarity searching with response times in the order of seconds for databases with millions of compounds, depending on a given similarity cut-off. For substructure searching, it can make use of multiple processor cores on today's powerful database servers to provide fast response times in equally large data sets.

Besides the traditional and proven relational database approach with added chemical features ('cartridges'), there is growing interest in tools and approaches based on the web philosophy and practice. Several groups [49,50] are experimenting with the Resource Description Framework (RDF) language on the assumption that generic high-performance solutions will appear. RDF allows everything to be described by URIs (data, molecules, dictionaries, relations). The Chempound system [31], as deployed in Quixote and elsewhere, is an RDF-based approach to chemical structures and compounds and their properties. For small to medium-sized collections (such as an individual's calculations or literature retrieval), there are many RDF tools (e.g. SIMILE, Apache Jena) which can operate in machine memory and provide the flexibility that RDF offers. For larger systems, it is unclear whether complete RDF solutions (e.g. Virtuoso) will be satisfactory or whether a hybrid system based on name-value pairs (e.g. CouchDB, MongoDB) will be sufficient.

Collaboration and interoperability

One of the successes of the Blue Obelisk has been to bring developers together from different Open Source chemistry projects so that they look for opportunities to collaborate rather than compete, and to leverage work done by other projects to avoid duplication of effort. As an example of this, when in March 2008 the Jmol development team were looking to add support for energy

minimisation, rather than implement a forcefield from scratch they ported the UFF forcefield [51] implementation from Open Babel to Jmol. This code enables Jmol to support 2D to 3D conversion of structures (through energy minimisation). In a similar manner, efficient Jmol code for atom-atom rebonding has been ported to the CDK. Figure 4 shows the collaborative nature of software developed in the Blue Obelisk, as one project builds on functionality provided by another project.

Another collaborative initiative between Blue Obelisk projects was the establishment in May 2008 of the ChemiSQL project. This brought together the developers of several open source chemistry database cartridges (PgChem [52], Mychem [53], OrChem [48] and more recently Bingo [54]) with a view to making their database APIs more similar and collaborating on benchmark datasets for assessing performance. For two of these projects, PgChem and Mychem, which are both based on Open Babel, there is the additional possibility of working together on a shared codebase.

In the area of cheminformatics toolkits, two of the existing toolkits Open Babel and RDKit are planning to work together on a common underlying framework called MolCore [55]. This project is still in the planning stage, but if it is a success it will mean that the the two libraries will be interoperable (while retaining their existing focus) but also that the cost of maintaining the code will be shared among more developers, freeing time for the development of new features.

One of the goals of the Blue Obelisk is to promote interoperability in chemical informatics. When barriers exist to moving chemical data between different software, the community becomes fragmented and there is the danger of vendor lock-in (where users are constrained to using a particular software, a situation which puts them at a disadvantage). This applies as much to Open Source software as to proprietary software. Cinfony is a project (first release in May 2008) whose goal is to tackle this problem in the area of cheminformatics toolkits [56]. It is a Python library that enables Open Babel, the CDK, and RDKit (and shortly, Indigo and OPSIN) to be used using the same API; this makes it easy, for example, to read a molecule using Open Babel, calculate descriptors using the CDK and create a depiction using RDKit.

Another way through which interoperability of Blue Obelisk projects has been promoted and developed is through integration into workflow software such as Taverna [57] and KNIME [58] (both open source). Such software makes it easy to automate recurring tasks, and to combine analyses or data from a variety of different software and web services. A combination of the Chemistry Development Kit and Taverna, for instance, was reported in 2010 [59]. In the case of KNIME, it comes with built-in basic collection of CDK-based and Open Babel-based nodes, while other nodes for the RDKit and Indigo are available from KNIME's "Community Updates" site.

Open Standards

Chemical Markup Language, CML

Chemical Markup Language (CML) is discussed in several articles in this issue, and a brief summary here reiterates that it is designed primarily to create a validatable semantic representation for chemical objects. The five main areas (molecules, reactions, computational

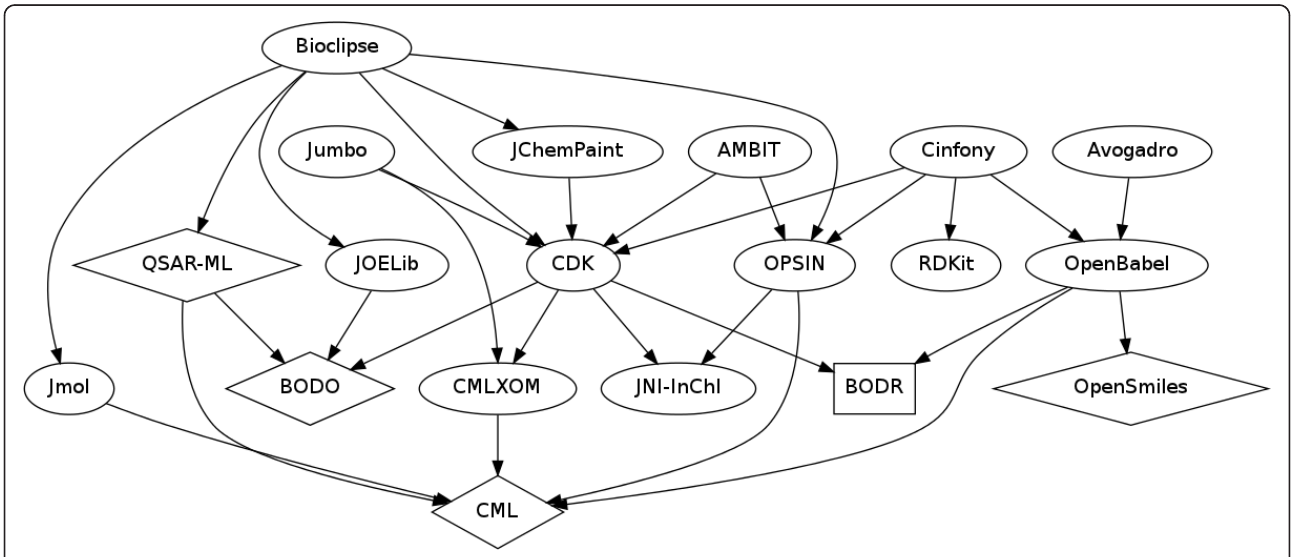

Figure 4 Dependency diagram of some Blue Obelisk projects. Each block represents a project. Square blocks show Open Data, ovals are Open Source, and diamonds are Open Standards.

chemistry, spectra and solid-state (see above)) have now all been extensively deployed and tested. CML can therefore be used as a reference for input and output for Blue Obelisk software and a means of representing data in Blue Obelisk resources.

CML, being an XML application, can inter-operate with other markup languages and in particular XHTML, SVG, MathML, docx and more specialised applications such as UnitsML and GML (geosciences). We believe that it would be possible using these languages to encode large parts of, say, first year chemistry text books in XML. Similarly, it is possible to create compound documents with word processing or spreadsheet software that have inter-operating text, graphics and chemistry (as in Chem4Word). Being a markup language, CML is designed for re-purposing, including styling, and therefore a mixture of these languages can be used for chemical catalogues, general publications, logbooks and many other types of document in the scientific process.

CML describes much of its semantics through conventions and dictionaries, and the emerging ecosystem (especially in computational chemistry) is available as a semantic resource for many of the applications and specifications in this article.

InChI

The IUPAC InChI identifier is a non-proprietary and unique identifier for chemical substances designed to enable linking of diverse data compilations. Prior to the development of the InChI identifier chemical information systems and databases used a wide variety of (generally proprietary) identifiers, greatly limiting their interoperability. Although its development predates the Blue Obelisk, software such as Open Babel has included InChI support since 2005, and support for InChI in Indigo is due in 2011.

Since the official InChI implementation is in C, it is difficult to access from the other widely used language for cheminformatics toolkits, Java. Early attempts to generate InChI identifiers from within Java involved programatically launching the InChI executable and capturing the output, an approach that was found to be fairly unreliable and broke the 'write once, run anywhere' philosophy of Java. The Blue Obelisk project JNI-InChI [60] was established in 2006 to solve this problem by using the Java Native Interface framework to provide transparent access to the InChI library from within Java and other Java Virtual Machine (JVM) based languages, supporting the wider adoption of this standard identifier by the chemistry community.

The Java Native Interface framework provides a mechanism for code running inside the JVM, to place calls to libraries written in languages such as C, C++ and Fortran, and compiled into native, machine specific, code. JNI-InChI provides a thin C wrapper, with corresponding Java code, around the IUPAC InChI library, exposing the InChI library's functionality to the JVM. To overcome the need to have the correct InChI library pre-installed on a system, JNI-InChI comes with a variety of precompiled native binaries and automatically extracts and deploys the correct one for the detected operating system and architecture. The JNI-InChI library comes with native binaries supporting a range of operating systems and architectures; the current version has binaries for 32- and 64-bit Windows, Linux and Solaris, 64-bit FreeBSD and 64-bit Intel-based Mac OS X - a number of which are not supported by the original IUPAC distribution of InChI. The JNI-InChI project has matured to support the full range of functionality of the InChI C library: structure-to-InChI, InChI-to-structure, AuxInfo-to-structure, InChIKey generation, and InChI and InChIKey validation. JNI-InChI provides the InChI functionality for a number of Open Source projects, including the Chemistry Development Kit, Bioclipse and CMLXOM/JUMBO, and is also used by commercial applications and internally in a number of companies. Through its widespread use and Open Source development model, a number of issues in earlier versions of the software have been identified and resolved, and JNI-InChI now offers a robust tool for working with InChIs in the JVM.

OpenSMILES

One of the most widely used ways to store chemical structures is the SMILES format (or SMILES string). This is a linear notation developed by Daylight Information Systems that describes the connection table of a molecule and may optionally encode chirality. Its popularity stems from the fact that it is a compact representation of the chemical structure that is human readable and writable, and is convenient to manipulate (e.g. to include in spreadsheets, or copy from a web page).

Despite its widespread use, a formal definition of the language did not exist beyond Daylight's SMILES Theory Manual and tutorials. This caused some confusion in the implementation and interpretation of corner cases, for example the handling of cis/trans bond symbols at ring closures. In 2007, Craig James (eMolecules) initiated work on the OpenSMILES specification [61], a complete specification of the SMILES language as an Open Standard developed through a community process. The specification is largely complete and contains guidelines on reading SMILES, a formal grammar, recommendations on standard forms when writing SMILES, as well as proposed extensions.

QSAR-ML

The field of QSAR has long been hampered by the lack of open standards, which makes it difficult to share and reproduce descriptor calculations and analyses. QSAR-ML was recently proposed as an open standard for exchanging QSAR datasets [62]. A dataset in QSAR-ML includes the chemical structures (preferably described in CML) with InChI to protect integrity, chemical descriptors linked to the Blue Obelisk Descriptor Ontology [63], response values, units, and versioned descriptor implementations to allow descriptors from different software to be integrated into the same calculation. Hence, a dataset described in QSAR-ML is completely reproducible. To allow for easy setup of QSAR-ML compliant datasets, a plugin for Bioclipse was created with a graphical interface for setting up QSAR datasets and performing calculations. Descriptor implementations are available from the CDK and JOELib, as well as via remote web services such as XMPP [64].

Remaining challenges

A core requirement for chemical structure databases and chemical registration systems in general is the notion of structure standardisation. That is, for a given input structure, multiple representations should be converted to one canonical form. Structure canonicalisation routines partially address this aspect, converting multiple alternative topologies to a single canonical form. However, the problem of standardisation is broader than just topological canonicalisation. Features that must be considered include

- topological canonicalisation
- handling of charges
- tautomer enumeration and canonicalisation
- normalisation of functional groups

Currently, most of the individual components of a 'standardisation pipeline' can be implemented using Blue Obelisk tools. The larger problem is that there is no agreed upon list of steps for a standardisation process. While some specifications have been published (e.g., PubChem) and some standardisation services and tools are available (for example, PubChem provides an online service to standardise molecules [65]) each group has their own set of rules. A common reference specification for standardisation would be of immense value in interoperability between structure repositories as well as between toolkits (though the latter is still confounded by differences in lower level cheminformatic features such as aromaticity models).

We have already discussed the development of an Open SMILES standard. While much progress has been made towards a complete specification, more remains to be done before this can be considered finished. After that point, the next logical step would be to start work on a standard for the SMARTS language, the extension to SMILES that specifies patterns that match chemical substructures.

Open Data

A considerable stumbling block in advocating the release of scientific data as Open Data has been how exactly to define "Open." A major step forward was the launch in 2010 of the Panton Principles for Open Data in Science [66]. This formalises the idea that Open Data maximises the possibility of reuse and repurposing, the fundamental basis of how science works. These principles recommend that published data be licensed explicitly, and preferably under CC0 (Creative Commons 'No Rights Reserved', also known as CCZero) [67]. This license allows others to use the data for any purpose whatsoever without any barriers. Other licenses compatible with the Panton Principles include the Open Data Commons Public Domain Dedication and Licence (PDDL), the Open Data Commons Attribution License, and the Open Data Commons Open Database License (ODbL) [68].

Despite this positive news, little chemical data compatible with these principles has become available from the traditional chemistry fields of organic, inorganic, and solid state chemistry. Table 2 lists a few notable exceptions, some of which are discussed further below. There is also data available using licenses not compatible with the Panton Principles, but where the user is allowed to modify and redistribute the data. A new data set in this category is the data from the ChEMBL database [69], which is available under the Creative Commons Share-Alike Attribution license. The RSC ChemSpider database [41], although not fully Open, also hosts Open

Table 2 Open Data in chemistry.

Name	License/Waiver	Description
Chempedia [98]	CC0	Crowd-sourced chemical names (project discontinued but data still available)
CrystalEye	PPDL	Crystal structures from primary literature
ONS Solubility	CC0	Solubility data for various solvents
Reaction Attempts	CC0	Data on successful and unsuccessful reactions

Overview of major open chemical data available under a license or waiver compatible with the Panton Principles.

Data; for example, spectral data when deposited can be marked as Open.

Importantly, publishing data as CC0 is becoming easier now that websites are becoming available to simplify publishing data. Two projects that can be mentioned in this context are FigShare [70], where the data behind unpublished figures can be hosted, and Dryad [71] where data behind publications can be hosted. Initiatives like this make it possible to host small amounts of data, and those combined are expected to become soon a substantial knowledge base.

Reaction Attempts

Although there are existing databases that allow for searching reactions, those using Open Data are harder to find. The Reaction Attempts database [72], to which anyone can submit reaction attempts data, consists mainly of reaction information abstracted from Open Notebooks in organic chemistry, such as the UsefulChem project from the Bradley group [73] and the notebooks from the Todd group [74]. Key information from each experiment is abstracted manually, with the only required information consisting of the ChemSpider IDs of the reactants and the product targeted in the experiment; and a link to the laboratory notebook page. Information in the database can be searched and accessed using the web-based Reaction Attempts Explorer [75].

Since the database reflects all data from the notebooks, it includes experiments in progress, ambiguous results and failed runs. Unlike most reaction databases that only identify experiments successfully reported in the literature, the Reaction Attempts Explorer allows researchers to easily find patterns in reactions that have already been performed, and since the data are open and results are reported across all research groups, intersections are easily discovered and possible Open Collaboration opportunities are easily found [76,77].

Non-Aqueous Solubility

Although the aqueous solubility of many common organic compounds is generally available, quantitative reports of non-aqueous solubility are more difficult to find. Such information can be valuable for selecting solvents for reactions, re-crystallization and related processes. In 2008, the Open Notebook Science Solubility Challenge was launched for the purpose of measuring non-aqueous solubility of organic compounds, reporting all the details of the experiments in an Open Notebook and recording the results as Open Data in a centralized database [78,79]. This crowdsourcing project was also supported by Submeta, Sigma-Aldrich, Nature Publishing Group and the Royal Society of Chemistry. The database currently holds 1932 total measurements and 1428 averaged solute/solvent measurements all of which are available under a CC0 license. Several web services and feeds are available to filter and re-use the dataset [80]. In particular, models have been developed for the prediction of non-aqueous solubility in 72 different solvents [81] using the method of Abraham et al [82] with descriptors calculated by the Chemistry Development Kit. These models are available online and will be refined as more solubility data is collected.

The Blue Obelisk Data Repository (BODR)

The Blue Obelisk has created a repository of key chemical data in a machine-readable format [83]. The BODR focuses on data that is commonly required for chemistry software, and where there is a need to ensure that values are standard between codes. Examples are atomic masses and conversions between physical constants. These data can be used by others for any purpose (for example, for entry into Wikipedia or use in in-house software), and should lead to an enhancement in the quality of community reference data. The Blue Obelisk provides also a complementary project, the Chemical Structure Repository [83]. It aims to provide 3D coordinates, InChIs and several physico-chemical descriptors for a set of 570 organic compounds.

NMRShiftDB

NMRShiftDB [84,85] represents one of the earliest resources for Open community-contributed data (first released in 2003). Research groups that measure NMR spectra or extract it from the literature can contribute that information to NMRShiftDB which provides an Open resource where entries can be searched by chemical structure or properties (especially peaks). Although it is difficult to encourage large amounts of altruistic contribution (as happens with Wikipedia), an alternative possible source of data could come from linking data capture with data publication. For example, the Blue Obelisk has enough software that it is possible to create a seamless chain for converting NMR structures in-house into NMRShiftDB entries. If and when the chemistry community encourages or requires semantic publication of spectra rather than PDFs, it would be possible to populate NMRShiftDB rapidly along the the lines of CrystalEye (see below). A similar approach has been demonstrated earlier using the Blue Obelisk components Oscar and Bioclipse using text mining approaches [86].

CrystalEye

CrystalEye [87] is an example of cost-effiective extraction of data from the literature where this is published both Openly and semantically. Software extracts Openly-published crystal structures from a variety of scholarly journals, processes them and then makes them available through a web interface. It currently contains

about 250,000 structures. CrystalEye serves as a model for a high-value, high-quality Open data resource, including the licensing of each component as Panton-compatible Open data.

Other areas of activity

While each Blue Obelisk project has its own website and point of contact (typically a mailing list), because of the breadth of Blue Obelisk projects it can be difficult for a newcomer to understand which of them, if any, can best address a particular problem. To address this issue, members of the Blue Obelisk established a Question & Answer website [88] (see Figure 5). This is a website in the style of Stack Overflow [89] that encourages high quality answers (and questions) through the use of a voting system. In the year since it was established, over 200 users have registered, many of whom had no previous involvement with the Blue Obelisk, showing that the Q&A website complements earlier existing channels of communication.

The rise of self-publishing and print-on-demand services has meant that publishing a book is now as straightforward as uploading to an appropriate website. Unlike the traditional publishing route where books with projected low sales volume would be expensive, websites such as Lulu [90] allow the sale of low-priced books on chemistry software, and books are now available for purchase on Jmol [91], the Chemistry Development Kit [92] and Open Babel [93].

Conclusions

We have shown that the Blue Obelisk has been very successful in bringing together researchers and developers with common interests in ODOSOS, leading to development of many useful resources freely available to the chemistry community. However, how best to engage with the wider chemistry community outside of the Blue Obelisk remains an open question. If the Blue Obelisk is truly to make an impact, then an attempt must be made to reach beyond the subscribers to the Blue Obelisk mailing list and blogs of members.

We hope to see this involvement between the Blue Obelisk and the wider community grow in the future. To this end, we encourage the reader to visit the Blue Obelisk website [94], send a message to our mailing list, investigate related projects or read our blogs.

Figure 5 Screenshot of the Blue Obelisk eXchange Question and Answer website.

Acknowledgements

NMOB is supported by a Health Research Board Career Development Fellowship (PD/2009/13). The OSRA project has been funded in whole or in part with federal funds from the National Cancer Institute, National Institutes of Health, under contract HHSN261200800001E. The content of this publication does not necessarily reflect the views of the policies of the Department of Health and Human Services, nor does mention of trade names, commercial products, or organisations imply endorsement by the US Government.

Author details

[1]Analytical and Biological Chemistry Research Facility, Cavanagh Pharmacy Building, University College Cork, College Road, Cork, Co. Cork, Ireland. [2]NIH Center for Translational Therapeutics, 9800 Medical Center Drive, Rockville, MD 20878, USA. [3]Division of Molecular Toxicology, Institute of Environmental Medicine, Nobels väg 13, Karolinska Institutet, 171 77 Stockholm, Sweden. [4]Unilever Centre for Molecular Sciences Informatics, Department of Chemistry, University of Cambridge, Lensfield Road, CB2 1EW, UK. [5]Department of Pharmaceutical Biosciences, Uppsala University, Box 591, 751 24 Uppsala, Sweden. [6]Department of Chemistry, Drexel University, 32nd and Chestnut streets, Philadelphia, PA 19104, USA. [7]Chemical Biology Laboratory, Basic Research Program, SAIC-Frederick, Inc., NCI-Frederick, Frederick, MD 21702, USA. [8]St. Olaf College, 1520 St. Olaf Ave., Northfield, MN 55057, USA. [9]Kitware, Inc., 28 Corporate Drive, Clifton Park, NY 12065, USA. [10]Department of Chemistry, University of Pittsburgh, 219 Parkman Avenue, Pittsburgh, PA 15260, USA. [11]eMolecules Inc., 380 Stevens Ave., Solana Beach, California 92075, USA. [12]Ideaconsult Ltd., 4.A.Kanchev str., Sofia 1000, Bulgaria. [13]Department of Engineering, Computer Science, Physics, and Mathematics, Oral Roberts University, 7777 S. Lewis Ave. Tulsa, OK 74171, USA. [14]Leiden Institute of Chemistry, Leiden University, Einsteinweg 55, 2333 CC Leiden, The Netherlands. [15]Department of Chemistry, State University of New York at Buffalo, Buffalo, NY 14260-3000, USA. [16]Université de Strasbourg, IPHC, CNRS, UMR7178, 23 rue du Loess 67037, Strasbourg, France. [17]GGA Software Services LLC, 41 Nab. Chernoi rechki 194342, Saint Petersburg, Russia. [18]Cheminformatics and Metabolism Team, European Bioinformatics Institute (EBI), Wellcome Trust Genome Campus, Hinxton, Cambridge, UK. [19]Department of Chemistry, University of Washington, Seattle, WA 98195, USA. [20]iChemLabs, 200 Centennial Ave., Suite 200, Piscataway, NJ 08854, USA.

Authors' contributions

The overall layout of the manuscript grew from discussions between NMOB, RG and ELW. The authorship of the paper is drawn from those people connected with fully Open Data/Standards/Source (OSI-compliant or OKF-compliant) projects associated with the Blue Obelisk. There are a large number of people contributing to these projects and because those projects are published in their own right it is not appropriate to include all their developers by default. We invited a number of 'project gurus' who have been active in promoting the Blue Obelisk, to be authors on this paper and most have accepted and contributed.

Competing interests

The authors declare that they have no competing interests.

Received: 1 July 2011 Accepted: 14 October 2011
Published: 14 October 2011

References

1. Matos PD, Alcantara R, Dekker A, Ennis M, Hastings J, Haug K, Spiteri I, Turner S, Steinbeck C: **Chemical Entities of Biological Interest: an update.** *Nucleic Acids Res* 2009, **38**:D249-D254.
2. Murray-Rust P: **The Blue Obelisk.** *CDK News* 2005, **2**:43-46.
3. Guha R, Howard MT, Hutchison GR, Murray-Rust P, Rzepa H, Steinbeck C, Wegner J, Willighagen EL: **The Blue Obelisk - Interoperability in Chemical Informatics.** *J Chem Inf Model* 2006, **46**:991-998.
4. Murray-Rust P, Rzepa HS: **Chemical Markup, XML, and the Worldwide Web. 1. Basic Principles.** *J Chem Inf Comput Sci* 1999, **39**:928-942.
5. Steinbeck C, Han Y, Kuhn S, Horlacher O, Luttmann E, Willighagen E: **The Chemistry Development Kit (CDK): An Open-Source Java Library for Chemo- and Bioinformatics.** *J Chem Inf Comput Sci* 2003, **43**:493-500.
6. Steinbeck C, Hoppe C, Kuhn S, Floris M, Guha R, Willighagen EL: **Recent developments of the chemistry development kit (CDK) - an open-source java library for chemo- and bioinformatics.** *Curr Pharm Design* 2006, **12**:2111-2120.
7. **Open Babel.** [http://openbabel.org].
8. **JOELib.** [http://sf.net/projects/joelib].
9. Rahman SA, Bashton M, Holliday GL, Schrader R, Thornton JM: **Small Molecule Subgraph Detector (SMSD) Toolkit.** *J Cheminf* 2009, **1**:12.
10. Dong X, Gilbert K, Guha R, Heiland R, Kim J, Pierce M, Fox G, Wild D: **A Web Service Infrastructure for Chemoinformatics.** *J Chem Inf Model* 2007, **47**:1303-1307.
11. **RDKit.** [http://rdkit.org].
12. **Indigo.** [http://ggasoftware.com/opensource/indigo].
13. **Open Babel - Related Software.** [http://openbabel.org/wiki/Related_Projects].
14. Spjuth O, Helmus T, Willighagen EL, Kuhn S, Eklund M, Wagener J, Murray-Rust P, Steinbeck C, Wikberg JES: **Bioclipse: an open source workbench for chemo- and bioinformatics.** *BMC Bioinformatics* 2007, **8**:59.
15. **Avogadro: an open-source molecular builder and visualization tool.** [http://avogadro.openmolecules.net].
16. Spjuth O, Alvarsson J, Berg A, Eklund M, Kuhn S, Masak C, Torrance G, Wagener J, Willighagen E, Steinbeck C, Wikberg J: **Bioclipse 2: A scriptable integration platform for the life sciences.** *BMC Bioinformatics* 2009, **10**:397.
17. Alvarsson J, Andersson C, Spjuth O, Larsson R, Wikberg J: **Brunn: An open source laboratory information system for microplates with a graphical plate layout design process.** *BMC Bioinformatics* 2011, **12**:179.
18. **Kalzium - Periodic Table and Chemistry in KDE.** [http://edu.kde.org/applications/science/kalzium/].
19. **XtalOpt - Evolutionary Crystal Structure Prediction.** [http://xtalopt.openmolecules.net].
20. Lonie DC, Zurek E: **XtalOpt: An open-source evolutionary algorithm for crystal structure prediction.** *Comput Phys Commun* 2011, **182**:372-387.
21. Jeliazkova N, Jeliazkov V: **AMBIT RESTful web services: an implementation of the OpenTox application programming interface.** *J Cheminf* 2011, **3**:18.
22. Jeliazkova N, Jaworska J, Worth A: **Open Source Tools for Read-Across and Category Formation.** In *In Silico Toxicology : Principles and Applications.* Edited by: Cronin M, Madden J. Cambridge UK: RSC Publishing; 2010:408-445.
23. **ToxTree.** [http://toxtree.sf.net].
24. Patlewicz G, Jeliazkova N, Safford RJ, Worth AP, Aleksiev B: **An evaluation of the implementation of the Cramer classification scheme in the Toxtree software.** *SAR QSAR Environ Res* 2008, **19**:495-524.
25. O'Boyle NM, Tenderholt AL, Langner KM: **cclib: A library for package-independent computational chemistry algorithms.** *J Comp Chem* 2008, **29**:839-845.
26. **GaussSum.** [http://gausssum.sf.net].
27. **QMForge.** [http://qmforge.sf.net].
28. Dapprich S, Frenking G: **Investigation of Donor-Acceptor Interactions: A Charge Decomposition Analysis Using Fragment Molecular Orbitals.** *J Phys Chem* 1995, **99**:9352-9362.
29. **JUMBO-Converters.** [https://bitbucket.org/wwmm/jumbo-converters].
30. **Lensfield 2.** [https://bitbucket.org/sea36/lensfield2].
31. **Chempound.** [https://bitbucket.org/chempound/chempound].
32. Stein W, *et al*: *Sage Mathematics Software* The Sage Development Team; 2011 [http://www.sagemath.org].
33. Hanson RM: **Jmol - a paradigm shift in crystallographic visualization.** *J Appl Cryst* 2010, **43**:1250-1260.
34. **ChemDoodle Web Components: HTML5 Chemistry.** [http://web.chemdoodle.com].
35. Stahl MT: **Open-source software: not quite endsville.** *Drug Discov Today* 2005, **10**:219-22.
36. DeLano WL: **The case for open-source software in drug discovery.** *Drug Discov Today* 2005, **10**:213-7.
37. Munos B: **Can open-source R&D reinvigorate drug research?** *Nat Rev Drug Discov* 2006, **5**:723-9.
38. Geldenhuys WJ, Gaasch KE, Watson M, Allen DD, Van der Schyf CJ: **Optimizing the use of open-source software applications in drug discovery.** *Drug Discov Today* 2006, **11**:127-32.
39. **iBabel.** [http://homepage.mac.com/swain/Sites/Macinchem/page65/ibabel3.html].
40. **ChemSpotlight.** [http://chemspotlight.openmolecules.net].
41. **ChemSpider - the free chemical database.** [http://www.chemspider.com].
42. **iChemLabs and RSC ChemSpider announce partnership.** [http://www.chemspider.com/blog/ichemlabs-and-rsc-chemspider-announce-partnership.html].
43. **Silicos Open Source Software.** [http://silicos.silicos-it.com/download.html].
44. **OSRA.** [http://cactus.nci.nih.gov/osra/].
45. Lowe DM, Corbett PT, Murray-Rust P, Glen RC: **Chemical Name to Structure: OPSIN, an Open Source Solution.** *J Chem Inf Model* 2011, **51**:739-753.
46. IUPAC: *Nomenclature of Organic Chemistry* Pergamon Press, Oxford; 1979.
47. IUPAC: *A Guide to IUPAC Nomenclature of Organic Compounds (Recommendations 1993)* Blackwell Scientific publications, Oxford; 1993.
48. Rijnbeek M, Steinbeck C: **OrChem - An open source chemistry search engine for Oracle(R).** *J Cheminf* 2009, **1**:17.
49. Willighagen EL, Brändle MP: **Resource description framework technologies in chemistry.** *J Cheminf* 2011, **3**:15.
50. Chen B, Dong X, Jiao D, Wang H, Zhu Q, Ding Y, Wild DJ: **Chem2Bio2RDF: a semantic framework for linking and data mining chemogenomic and systems chemical biology data.** *BMC Bioinformatics* 2010, **11**:255.
51. Rappé AK, Casewit CJ, Colwell KS, Goddard WA III, Skiff WM: **UFF, a full periodic table force field for molecular mechanics and molecular dynamics simulations.** *J Am Chem Soc* 1992, **114**:10024-10035.
52. **PgChem.** [http://pgfoundry.org/projects/pgchem/].
53. **Mychem.** [http://mychem.sf.net].
54. **Bingo.** [http://ggasoftware.com/opensource/bingo].
55. **MolCore.** [http://molcore.sf.net].
56. O'Boyle NM, Hutchison GR: **Cinfony-combining Open Source cheminformatics toolkits behind a common interface.** *Chem Cent J* 2008, **2**:24.
57. Hull D, Wolstencroft K, Stevens R, Goble C, Pocock MR, Li P, Oinn T: **Taverna: a tool for building and running workflows of services.** *Nucleic Acids Res* 2006, **34**:W729-W732.
58. **KNIME.** [http://www.knime.org].
59. Kuhn T, Willighagen E, Zielesny A, Steinbeck C: **CDK-Taverna: an open workflow environment for cheminformatics.** *BMC Bioinformatics* 2010, **11**:159.
60. **JNI-InChI.** [http://jni-inchi.sf.net/index.html].
61. **The OpenSMILES specification.** [http://opensmiles.org].
62. Spjuth O, Willighagen EL, Guha R, Eklund M, Wikberg JE: **Towards interoperable and reproducible QSAR analyses: Exchange of datasets.** *J Cheminf* 2010, **2**:5.
63. **The Blue Obelisk Descriptor Ontology.** [http://qsar.sourceforge.net/dicts/qsar-descriptors/index.xhtml].
64. Wagener J, Spjuth O, Willighagen EL, Wikberg JES: **XMPP for cloud computing in bioinformatics supporting discovery and invocation of asynchronous web services.** *BMC Bioinformatics* 2009, **10**:279.
65. **PubChem Standardization Service.** [http://pubchem.ncbi.nlm.nih.gov//standardize/standardize.cgi].
66. **Panton Principles - Principles for Open Data in Science.** [http://pantonprinciples.org].
67. **About CC0 - "No Rights Reserved".** [http://creativecommons.org/about/cc0].
68. **Open Licenses - Data.** [http://www.opendefinition.org/licenses/#Data].
69. Overington J: **ChEMBL. An interview with John Overington, team leader, chemogenomics at the European Bioinformatics Institute Outstation of the European Molecular Biology Laboratory (EMBL-EBI). Interview by Wendy A. Warr.** *J Comp Aided Mol Des* 2009, **23**:195-198.

70. **FigShare.** [http://figshare.com].
71. **Dryad.** [http://datadryad.org].
72. **Reaction Attempts Database.** [http://onswebservices.wikispaces.com/reactions].
73. Bradley JC: **Useful Chemistry: Reaction Attempts Book Edition 1 and UsefulChem Archive.**[http://usefulchem.blogspot.com/2010/04/reaction-attempts-book-edition-1-and.html].
74. Bradley JC: **Useful Chemistry: The Synaptic Leap Experiments on Reaction Attempts.**[http://usefulchem.blogspot.com/2010/05/synaptic-leap-experiments-on-reaction.html].
75. Bradley JC: **Useful Chemistry: Reaction Attempts Explorer.**[http://usefulchem.blogspot.com/2010/06/reaction-attempts-explorer.html].
76. Bradley JC: **Useful Chemistry: Visualizing Social Networks in Open Notebooks.**[http://usefulchem.blogspot.com/2010/12/visualizing-social-networks-in-open.html].
77. Bradley JC, Lang AS, Koch S, Neylon C: **Collaboration using Open Notebook Science in Academia.** In *Collaborative computational technologies for biomedical research*. Edited by: Ekins S, Hupcey MA, Williams AJ. Hoboken N.J.: John Wiley 2011:425-452.
78. Bradley JC, Neylon C, Guha R, Williams AJ, Hooker B, Lang ASID, Friesen B, Bohinski T, Bulger D, Federici M, Hale J, Mancinelli J, Mirza KB, Moritz MJ, Rein D, Tchakounte C, Truong HT: **Open Notebook Science Challenge: Solubilities of Organic Compounds in Organic Solvents.** *Nature Precedings* 2010 [http://dx.doi.org/10.1038/npre.2010.4243.3].
79. Bradley J, Guha R, Lang A, Lindenbaum P, Neylon C, Williams A, Willighagen E: **Beautifying Data in the Real World.** In *Beautiful Data*.. 1 edition. Edited by: Segaran T, Hammerbacher J. Sebastopol CA: O'Reilly; 2009:259-278.
80. **Open Notebook Solubility Web Services.** [http://onswebservices.wikispaces.com/solubility].
81. Bradley J: **Useful Chemistry: General Transparent Solubility Prediction using Abraham Descriptors.**[http://usefulchem.blogspot.com/2010/07/general-transparent-solubility.html].
82. Abraham MH, Smith RE, Luchtefeld R, Boorem AJ, Luo R, Acree WE Jr: **Prediction of solubility of drugs and other compounds in organic solvents.** *J Pharm Sci* 2010, **99**:1500-1515.
83. **Blue Obelisk Data Repository.** [http://bodr.sf.net].
84. **NMRShiftDB.** [http://www.nmrshiftdb.org].
85. Steinbeck C, Kuhn S: **NMRShiftDB - compound identification and structure elucidation support through a free community-build web database.** *Phytochemistry* 2004, **65**:2711-2717.
86. Willighagen EL: **Chemical Archeology: OSCAR3 to NMRShiftDB.org.**[http://chem-bla-ics.blogspot.com/2006/09/chemical-archeology-oscar3-to.html].
87. **CrystalEye.** [http://wwmm.ch.cam.ac.uk/crystaleye/].
88. **Blue Obelisk Q&A.** [http://blueobelisk.shapado.com].
89. **Stack Overflow.** [http://stackoverflow.com].
90. **Lulu.** [http://lulu.com].
91. Herráez A: In *How to use Jmol to study and present molecular structures. Volume 1*. Lulu Enterprises, Morrisville, NC, US; 2007.
92. Willighagen E: *Groovy Cheminformatics with the Chemistry Development Kit* Lulu Enterprises, Morrisville, NC, US; 2011.
93. Hutchison GR, Morley C, O'Boyle NM, James C, Swain C, De Winter H, Vandermeersch T: *Open Babel - Official User Guide* Lulu Enterprises, Morrisville, NC, US; 2011.
94. **Blue Obelisk web site.** [http://blueobelisk.org].
95. Day N, Murray-Rust P, Tyrrell S: **CIFXML: a schema and toolkit for managing CIFs in XML.** *J Appl Cryst* 2011, **44**:628-634.
96. Hawizy L, Jessop D, Adams N, Murray-Rust P: **ChemicalTagger: A tool for Semantic Text-mining in Chemistry.** *J Cheminf* 2011, **3**:17.
97. O'Boyle NM, Vandermeersch T, Flynn CJ, Maguire AR, Hutchison GR: **Confab - Systematic generation of diverse low-energy conformers.** *J Cheminf* 2011, **3**:8.
98. **Chempedia.** [http://chempedia.com].

doi:10.1186/1758-2946-3-37
Cite this article as: O'Boyle *et al.*: Open Data, Open Source and Open Standards in chemistry: The Blue Obelisk five years on. *Journal of Cheminformatics* 2011 **3**:37.