

Programming for Chemical and Life Science Informatics

I573 - Week 5
(Programming with OEChem)

Rajarshi Guha

10th February, 2009

Where to Get It?

- OpenEye [downloads](#)
- You'll need an academic license
- *cheminfo* has the license installed
- We have access to the C++ and Python interfaces

Setup

- Working OEChem in Python requires some setup
- Add `/usr/local/openeye/wrappers/python` to your `PYTHONPATH` variable
- Add `/usr/local/openeye/wrappers/lib` to your `LD_LIBRARY_PATH` variable
- When writing Python code with OEChem, always import the library

```
from openeye.oechem import *
```

- When coding in C/C++ the gcc invocation will require

```
-I/usr/local/openeye/include -L/usr/local/openeye/lib \  
-loechem -loesystem -loeplatform -loebio \  
-lz -lm -lpthread
```

Resources

- OEChem [API](#)
- [C++](#) manuals
- [Python](#) manuals

Where is it Used?

- A polished commercial toolkit
- Widely used in the pharma industry
- Provide free academic licenses, so common in academia as well
- Well known public sites using OE tools include
 - ▶ [PubChem](#)
 - ▶ [PDB](#)
 - ▶ [ZINC](#)

What Does it Provide?

- Basic chemical objects
 - ▶ Atoms, bonds and molecules
- Complex objects
 - ▶ Protein sequences, conformers
- Support for many chemical formats
- SMILES parsing/generation
- Good stereochemistry support
- Multiple models of aromaticity

Atoms & Bonds

- They only exist in the context of a molecule

```
mol = OEMol()  
a1 = mol.NewAtom(6)  
a2 = mol.NewAtom(6)  
a3 = mol.NewAtom(6)  
b1 = mol.NewBond(a1,a2, 1)  
b2 = mol.NewBond(a2,a3, 1)
```

- Atoms have a variety of properties that can be get or set
 - ▶ atomic number, formal charge
 - ▶ symmetry
 - ▶ type
- Get the serial number of an atom or bond

```
a1.GetIdx()  
b1.GetIdx()
```

Molecules

- There are two types of molecule objects
 - ▶ OEMol
 - ▶ OEGraphMol
- OEMol handles single and multi-conformer molecules
- OEGraphMol handles single conformer molecules
- When programming in Python, doesn't matter unless we need to specifically consider conformers
- Create an empty molecule

```
mol = OEMol()
```

- Given the molecule, we can loop over atoms or bonds

```
mol = OEMol()  
OEParseSmiles(mol, 'C1CC(CC=N)CC1')  
for atom in mol.GetAtoms():  
    print atom.GetAtomicNum()
```

- When a molecule is read in, it is already *configured*
 - ▶ aromaticity detected
 - ▶ ring perception is performed

- We consider SMILES parsing
- First create a molecule object and then parse a SMILES *into* it

```
mol = OEmol()  
status = OEParseSmiles(mol, 'c1ccccc1')
```

- OEParseSmiles function returns a boolean
 - ▶ TRUE indicates that the SMILES was parsed OK
 - ▶ If not, it will return FALSE as well as printing out a warning message

- We can easily write a molecule as a SMILES string

```
>>> mol = OEMol()  
>>> status = OEParseSmiles(mol, 'C1CCC(CC=CC(C)(C))CCC1')  
>>> smi = OECreateCanSmiString(mol)  
>>> smi  
'CC(C)C=CCC1CCCCC1'
```

- Be careful when reusing molecule objects
- If you want to use it for a new molecule, you must *explicitly* clear it

- For example

```
>>> mol = OEMol()  
>>> OEParseSmiles(mol, 'C1CCC1')  
>>> OEParseSmiles(mol, 'c1ccccc1')
```

- If we now generate the canonical SMILES for mol we **won't** get benzene

```
>>> smi = OECreateCanSmiString(mol)  
>>> smi  
'c1ccccc1.C1CCC1'
```

- Instead we get *both* molecules, since we didn't clear the molecule object

- The correct way is to clear the molecule object

```
>>> mol = OEMol()  
>>> OEParseSmiles(mol, 'C1CCC1')  
>>> mol.Clear()  
>>> OEParseSmiles(mol, 'c1cccc1')  
>>> smi = OECreateCanSmiString(mol)  
>>> smi  
'c1cccc1'
```

- Could create a new molecule, but that would be inefficient

- Reading from files is relatively simple
- File formats are automatically detected

```
ifs = oemolstream()  
ifs.open('pdgfr.sdf')  
for mol in ifs.GetOEMols():  
    print mol.GetTitle()
```

- Writing is done similarly
- Need to specify a format

```
ifs = oemolistream()
ofs = oemolostream()
ifs.open('pdgfr.sdf')
ofs.open('mypdgfr.smi')
ofs.SetFormat(OEFormat_ISM)
for mol in ifs.GetOEMols():
    print mol.GetTitle()
    OEWriteMolecule(ofs, mol)
```

- When looping over a file, don't save the molecules (say in a list)
- The molecule object gets reused in the loop
- If you need to save molecules, create a new one and save it

```
ifs = oemolistream()
ifs.open('pdgfr.sdf')

mlist = []
for mol in ifs.GetOEMols():
    aMol = OEMol(mol)
    mlist.append(aMol)
```

Input/Output - SD Tags

- As before we can easily extract SD tag values

```
ifs = oemolistream()
ifs.open('comp.sdf')
mol = OEGraphMol()
OEReadMolecule(ifs, mol)

OEGetSDData(mol, 'PUBCHEM_COMPOUND_CID')
```

- We can loop over all tag-value pairs

```
for tagpair in OEGetSDDataPairs(mol):
    print tagpair.GetTag(), ' --> ', tagpair.GetValue()
```

- Set tag data using

```
OESetSDData(mol, 'myOwnTag', '1.2.3.4')
```

Tasks - File Conversion

```
ifs = oemolistream()
ifs.open('file1.sdf')

ofs = oemolostream()
ofs.open('file2.smi')
ofs.SetFormat(OEFormat_CAN)

for mol in ifs.GetOEMols():
    OEWriteMolecule(ofs, mol)
```

Task - Explicit Hydrogens

- Certain applications require explicit hydrogens
- Loop over each atom and add H's where required

```
mol = OEMol()
OEParseSmiles(mol, 'c1ccc(CC=CC)cc1')
for atom in mol.GetAtoms():
    OEAddExplicitHydrogens(mol, atom)
```

- We can add H's to all the atoms at one go

```
mol = OEMol()
OEParseSmiles(mol, 'c1ccc(CC=CC)cc1')
OEAddExplicitHydrogens(mol)
```

A SMARTS Primer

- SMARTS are regular expressions for chemical structures
- See the [Daylight manual](#) for a detailed description of the language
- Note that a valid SMILES string is a valid SMARTS string
- The reverse is not true
- Simple examples
 - ▶ c1ccccc1 - matches a benzene ring
 - ▶ C(=O)N - matches an amide bond
- Both patterns are valid molecules (benzene and formamide)
- Daylights [depictmatch](#) is a handy online tool to test out SMARTS patterns

Atom primitives

- * - any atom
- a - an aromatic atom (replace a with the atom symbol)
- A - an aliphatic atom
- D<n> - an atom of degree n
- X<n> - an atom with n total connections

Bond Primitives

- ~ - any bond
- -, =, #, : - single, double, triple or aromatic bond

Logical operators

- - not
- & - and, high precedence
- , - or
- ; - and, low precedence

Examples

- `C` would match any carbon atom
- `CCc` would match an ethyl group attached to an aromatic carbon atom
- `c:c` matches 2 aromatic carbons connected by an aromatic bond
- `c-c` matches 2 aromatic carbons connected by a single bond
- The last 2 will work for benzene and biphenyl individually
- What if you want to match both at the same time?
 - ▶ Use `c~c` or `cc`

Examples

- C(=O)[S,N] - would match a molecule with an amide or sulfamide
- [N,R] - an aliphatic nitrogen or a ring atom
- [NR] - an aliphatic nitrogen in a ring
- [nD2r5] - a nitrogen in a 5 membered aromatic ring

Task - Substructure Searching

- Check whether a substructure is present in a target molecule
- Identify atoms (or bonds) from the target that match the query

```
mol = OEGraphMol()
OEParseSmiles(mol, "CCC2CCC2Cc1ccccc1")

matcher = OESubSearch()
pattern = "c1ccccc1"

matcher.Init(pattern)

print matcher.SingleMatch(mol)

for matchbase in matcher.Match(mol, 1):
    for pair in matchbase.GetAtoms():
        print pair.target.GetIdx()
```