Supplementary material

Codes of Methods

1. Extraction of R groups

```
#Core Function

from rdkit import Chem

from rdkit.Chem import BRICS

...

fo = open('../Result/'+Name+'_RGroup.txt', 'w')

try:

        smiles = line.strip()

        mol = Chem.MolFromSmiles(smiles)

        frags = list(BRICS.BRICSDecompose(mol))

        frags = [m for m in frags if m.count('*')==1]

        f.write('\n'.join(frags)+'\n')

except:

        pass

...
```

2. Molecule enumeration

```
#Core Function

from indigo import *

indigo = Indigo()

...
```

```
def rGroupChange(rgroup, bondID, mol):
    mol = mol.clone()
    rgroup = rgroup.clone()
    mapping = mol.merge(rgroup)
    if rgroup.smiles() == '[H]':
        index = 0
    else:
        index = 1
    mapping.mapAtom(rgroup.getAtom(index)).addBond(mol.getAtom(bondID), 1)
    return mol
...
```

3. Calculation of pKa value

```
#Core Function
import jpype as jp
def getPkaOfMolfile(molfile, plugin, pKaAtom):
    try:
        mol = jp.JPackage('chemaxon').formats.MolImporter.importMol(molfile)
        plugin.setMolecule(mol)
        plugin.run()
        if plugin.getpKa(pKaAtom) != 'nan':
            return plugin.getpKa(pKaAtom)
        else:
            return - 1
        except:
                return - 1

...
jarpath = 'E:\jar\marvin\chemaxon\jchem.jar'
jp.startJVM(jp.getDefaultJVMPath(), '-ea', '-Djava.class.path=%s' %(jarpath))
PluginClass =jp.JClass('chemaxon.marvin.calculations.pKaPlugin')
plugin = PluginClass()
getPkaOfMolfile(molfile, plugin, pKaAtom)
jp.shutdownJVM()
...
```

4.1. Drug-likeness screening: Lipinski's Rule of Five

```
...
#Core Function
def calROF(smiles):
        #Lipinski's Rule of Five (Drug-like)
        from rdkit import Chem
        from rdkit.Chem import Descriptors
        m = Chem.MolFromSmiles(smiles)
        r1 =    Descriptors.NumHDonors(m) #<=5
        r2 = Descriptors.NumHAcceptors(m) #<=10
        r3 = Descriptors.ExactMolWt(m) #<500
```

```
    r4 = Descriptors.MolLogP(m) #<=5
    return [r1<=5, r2<=10, r3<500, r4<=5]
```

...

4.2. Drug-likeness screening: Rule of Three (Lead-like drugs)

...

```
#Core Function
def calROT(smiles):
    #Rule of Three (Lead-like drugs)
    from rdkit import Chem
    from rdkit.Chem import Descriptors
    m = Chem.MolFromSmiles(smiles)
    r1 = Descriptors.ExactMolWt(m) #<300
    r2 =    Descriptors.NumHDonors(m) #<=3
    r3 = Descriptors.NumHAcceptors(m) #<=3
    r4 = Descriptors.MolLogP(m) #<=3
    return [r1<300, r2<=3, r3<=3, r4<=3]
```

...

4.3. Drug-likeness screening: Rapid Elimination Of Swill (REOS)

...

```
#Core Function
def calREOS(smiles):
    #Rapid Elimination Of Swill (REOS)
    #Nature Reviews Drug Discovery 2, 259-266 (April 2003)
```

```
from rdkit import Chem

from rdkit.Chem import Descriptors

m = Chem.MolFromSmiles(smiles)

r1 = Descriptors.ExactMolWt(m) #200<r1<500

r2 = Descriptors.MolLogP(m) #-5<=r2<=5

r3 =    Descriptors.NumHDonors(m) #<=5

r4 = Descriptors.NumHAcceptors(m) #<=10

r5 = sum([atom.GetFormalCharge() for atom in m.GetAtoms()]) #-2<=r5<=2

r6 = Descriptors.NumRotatableBonds(m) #<=8

r7 = Descriptors.HeavyAtomCount(m) #15<=r7<=50

return [200<r1 and r1<500, -5<=r2 and r2<=5, r3<=5,

          r4<=10, -2<=r5 and r5<=2, r6<=8, 15<=r7 and r7<=50]
```
...