BIOINFORMATION
Discovery at the interface of physical and biological sciences

open access www.bioinformation.net *Software*

Volume 6(10)

PDBToSDF: Create ligand structure files from PDB file

Naresh Babu Muppalaneni[1] & Allam Appa Rao[2]*

[1]Department of Computer Science & Engineering, Acharya Nagarjuna University, Guntur, India; [2]Jawaharlal Nehru Technological University Kakinada, Kakinada, India; Allam Appa Rao - Email: apparaoallam@gmail.com; *Corresponding author

Received June 21, 2011; Accepted June 28, 2011; Published August 02, 2011

Abstract:
Protein Data Bank (PDB) file contains atomic data for protein and ligand in protein-ligand complexes. Structure data file (SDF) contains data for atoms, bonds, connectivity and coordinates of molecule for ligands. We describe PDBToSDF as a tool to separate the ligand data from pdb file for the calculation of ligand properties like molecular weight, number of hydrogen bond acceptors, hydrogen bond receptors easily.

Keywords: Protein Data Bank file, Structure Data file, Molecular weight

Background:
The Protein Data Bank (PDB) format provides a standard representation for macromolecular structure data derived from X-ray diffraction and NMR studies. This representation was created in the 1970's and a number of software tools have been developed and used for this purpose. Structure-data file (SDF) is a family of chemical-data file formats developed by MDL; it is intended especially for structural information. SDF files wrap the molfile (MDL_Molfile) format. Multiple compounds are delimited by lines consisting of four dollar signs ($$$$). A feature of the SDF format is its ability to include associated data **[1]**. Creating SDF files from the given PDB file will help to find the ligand property.

Methodology:
Application Software:
The application software using java **[2]** has been developed to divide the Structure Data File (SDF), the calculation of the properties of ligands as molecular weight, the number of hydrogen bond acceptors and donors on the basis that you can search in the ZINC database.

Separating ligand data from PDB file:
(1) Identify the lines starting with HET, which represents ligands in PDB file; (2) Identify the HETATOM for individual ligand based on HET atom; (3) Similarly find the CONECT data which represents bonding information for individual ligand; (4) Separate and save the information in Step 2, Step 3 in .sdf file.

Calculation of molecular weight:
Sum up of atomic weights for each atom is generated in .sdf file. **(Table 1, see Supplementary material)**

Hydrogen bond acceptors, donors:
Acceptor atoms have a lone pair of electron; usually sum of N, O. A donor atom is connected with at least one H atom; usually sum of OH, NH. When PDBToSDF file is executed, SDF file is generated, showing the molecular weight for each ligand. In this example, 1AH3.pdb is the input, it has 3 ligands namely AYA, NAP, TOL. **Figure 1** shows the molecular weights for these 3 ligands.

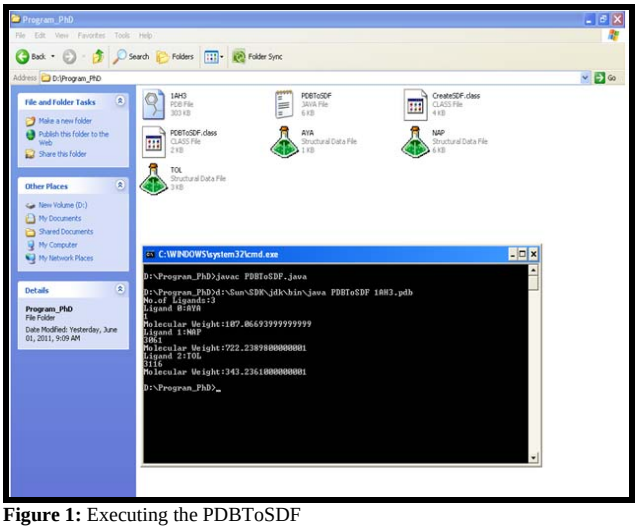

Figure 1: Executing the PDBToSDF

References:
[1] Dalby A *et al. J Chem Inf Comput Sci.* 1992 **32:** 244
[2] Java 2: The Complete Reference, Fifth Edition, Herbert Schildt, TMH

Edited by P Kangueane
Citation: Muppalaneni & Rao. Bioinformation 6(10): 383-386 (2011)

BIOINFORMATION

open access

Supplementary material:

Table 1: Shows Molecular weight for atoms

Atom	Atomic Weight
C	12.0107
H	1.00794
O	15.9994
Cl	35.453
S	32.066
F	18.998
N	14.0067
Br	79.904
P	30.974

CODE:

```
import java.util.*;
import java.io.*;
class CreateSDF
{
public void create(String ligname,String filename)
{
FileInputStream fstream;
DataInputStream in;
FileWriter fstream1;
BufferedWriter out;
try{
                        // Open the file that is the first
                        // command line parameter
                        fstream = new FileInputStream(filename);
                        // Get the object of DataInputStream
in = new DataInputStream(fstream);
      BufferedReader br = new BufferedReader(new InputStreamReader(in));
String strLine;
fstream1= new FileWriter(ligname+"_tmp.sdf");
out = new BufferedWriter(fstream1);
int count=0;
int start=-1;
int no_of_atoms=0;
while ((strLine = br.readLine()) != null)
            {
            if(strLine.startsWith("HETATM "))
            if(strLine.indexOf(ligname)>=0)
            {no_of_atoms++;count++;
            if(start==-1)
            {
            StringTokenizer st= new StringTokenizer(strLine);
            st.nextToken();
            start=Integer.parseInt(st.nextToken());
            System.out.println(start);
            }
  // Create file
out.write(strLine+"\n");
  //Close the output stream
//out.write(lineTokens[6]+"\t"+lineTokens[7]+"\t"+lineTokens[8]+"\t"+lineTokens[11]+"\n");
}//if close
}//while close
                        fstream = new FileInputStream(filename);
                        // Get the object of DataInputStream
in = new DataInputStream(fstream);
  br = new BufferedReader(new InputStreamReader(in));
int atomid;
int no_of_bonds=0;
String[] lTokens=new String[100];
while ((strLine = br.readLine()) != null)
            {int x=0;
            if(strLine.startsWith("CONECT"))
            {
StringTokenizer st=new StringTokenizer(strLine);
            while(st.hasMoreTokens())
            lTokens[x++]=st.nextToken();

            int bond_start=Integer.parseInt(lTokens[1]);
            for(int y=2;y<x;y++)
            {
```

```
                int p=bond_start;
int q=Integer.parseInt(lTokens[y]);
if(p<q && bond_start>=start && bond_start<start+no_of_atoms)
no_of_bonds++;
}
st= new StringTokenizer(strLine);
                st.nextToken();
                atomid=Integer.parseInt(st.nextToken());
                if(atomid>=start &&atomid<start+no_of_atoms)
                {out.write(strLine+"\n");}
                }//if close
                }//while close
in.close();
out.close();
writeToFile(ligname,no_of_atoms,no_of_bonds,start);
}//try block
catch (Exception e){//Catch exception if any
                                        System.err.println("Error: " + e.getMessage());
}
}//method close
public void writeToFile(String ligname,int no_of_atoms,int no_of_bonds,int start)
{
FileInputStream fstream;
DataInputStream in;
BufferedReader br;
FileWriter fstream1;
BufferedWriter out;
String lineTokens[]=new String[1000];
String formula[]=new String[9];
int form_count[]=new int[9];
double[] mol_weight={12.0107,1.00794,15.9994,35.453,32.066,18.998,14.0067,79.904,30.974};
String strLine;
try
{
fstream = new FileInputStream(ligname+"_tmp.sdf");
                        // Get the object of DataInputStream
in = new DataInputStream(fstream);
  br = new BufferedReader(new InputStreamReader(in));
fstream1= new FileWriter(ligname+".sdf");
out = new BufferedWriter(fstream1);
// writing header
out.write(ligname+"\n\n"+"Structure written by Naresh\n");
out.write(" "+no_of_atoms+" "+no_of_bonds+"  0  0  1  0            999 V2000\n");
int i=0;
formula[0]="C";formula[1]="H";formula[2]="O";formula[3]="Cl";
formula[4]="S";formula[5]="F";formula[6]="N";formula[7]="Br";
formula[8]="P";
form_count[0]=0;form_count[1]=0;form_count[2]=0;form_count[3]=0;
form_count[4]=0;form_count[5]=0;form_count[6]=0;form_count[7]=0;
form_count[8]=0;
StringTokenizer st;
                while ((strLine = br.readLine()) != null)
                {
                i=0;
//out.write(strLine);
if(strLine.startsWith("HETATM "))
{
 st = new StringTokenizer(strLine);
while(st.hasMoreTokens())
  lineTokens[i++]=st.nextToken();
for(int n=0;n<formula.length;n++)
if(formula[n].equals(lineTokens[11]))
form_count[n]++;
out.write("   "+lineTokens[6]+"0   "+lineTokens[7]+"0   "+lineTokens[8]+"0 "+lineTokens[11]+"   0  0  0  0  0  0 \n");
}//if close
 }//while close
in.close();
//out.close();
fstream = new FileInputStream(ligname+"_tmp.sdf");
in = new DataInputStream(fstream);
  br = new BufferedReader(new InputStreamReader(in));
int a[][]=new int[no_of_atoms+1][no_of_atoms+1];
for(int j=0;j<=no_of_atoms;j++)
for(int k=0;k<=no_of_atoms;k++)
a[j][k]=0;

while ((strLine = br.readLine()) != null)
                {i=0;
```

```
if(strLine.startsWith("CONECT"))
		{
		st=new StringTokenizer(strLine);
		while(st.hasMoreTokens())
		lineTokens[i++]=st.nextToken();
		int bond_start=Integer.parseInt(lineTokens[1]);
		for(int j=2;j<i;j++)
		{
		int p=bond_start-start+1;
int q=Integer.parseInt(lineTokens[j])-start+1;
if(a[p][q]==0 && p<q)
		out.write((bond_start-start+1)+"  "+(Integer.parseInt(lineTokens[j])-start+1)+"  1  0  0  0\n");
}//for
		}// conect if close
		}//while

out.write("M  END"+"\n"+"$$$$ ");
in.close();
out.close();
}//try close
catch (Exception e){//Catch exception if any
                                        System.err.println("Error: " + e.getMessage());
}
double mw=0;
for(int n=0;n<formula.length;n++)
{
System.out.print(formula[n]+""+form_count[n]);
mw+=mol_weight[n]*form_count[n];
}
System.out.print("Molecular Weight:"+mw+"\n");
}
}//class close
class PDBToSDF
{
  public static void main(String args[])
		{
    try{
                        // Open the file that is the first
                        // command line parameter
                        FileInputStream fstream = new FileInputStream(args[0]);
                        // Get the object of DataInputStream
                        DataInputStream in = new DataInputStream(fstream);
      BufferedReader br = new BufferedReader(new InputStreamReader(in));
                        String strLine;
                        //Read File Line By Line
String hetLine[]=new String[100];
int count=0;
                        while ((strLine = br.readLine()) != null)       {
                                // Print the content on the console
String het=strLine.substring(0,4);
//finding the lines beginning with HET
if(het.equals("HET "))
{
hetLine[count]=strLine;
count++;
}
}//while close
                        //Close the input stream
                        in.close();
System.out.println("No.of Ligands:"+count);
//Findinf the Ligand names
String ligname[]=new String[100];
CreateSDF c=new CreateSDF();
for(int i=0;i<count;i++)
{
StringTokenizer lig=new StringTokenizer(hetLine[i]);
lig.nextToken();
ligname[i]=lig.nextToken();
System.out.println("Ligand "+i+":"+ligname[i]);
c.create(ligname[i],args[0]);
}
                        }catch (Exception e){//Catch exception if any
                                System.err.println("Error: " + e.getMessage());
                        }
		}
}
```