SANCHO SCIENCE

Journal of Bioinformatics & Research 1(2) 2012: 36-40

Research Article
Open Access

ISSN: 2277-3363

Python Program to Generate Atom Records from PDB Protein Files for Drug Design Studies

Srinivasu Nulaka[1*], Appa Rao Allam[2], Suresh Kopparthi[3]

[1] *Department of Computer Science (MCA), KGRL College, Bhimavaram, Andhra Pradesh, India*
[2] *Former Vice-chancellor, JNTUK, Kakinada, Andhra Pradesh, India*
[3] *Department of Computer Science, MP College of Engg & Tech , Bhimavaram, Andhra Pradesh, India*

Received: 05 February 2012 Accepted: 01 March 2012 Online: 12 March 2012

ABSTRACT

Protein data bank is the major repository source of macromolecule 3-dimensional files in varied formats such as .ent, .pdb, .xml, .mmcif etc. The most widely used format is .ent and .pdb by many drug design experts in various computer-aided studies. The major requirement for any drug design or docking study is 3D format of protein as input without any heteroatoms. Other soft wares perform the task of separating protein and heteroatoms from PDB file, however, they require manual intervention. Therefore, a python based program was reported here that takes 3-dimensional PDB file as input and returns a file with only ATOM records in it. This output file is more important as an input in docking or protein-ligand interaction studies. The lines of python code were given in the manuscript.

Key Words: protein data bank, python, protein, ligand

INTRODUCTION

The use of individual protein structure models in biology is increasingly widespread in molecular modeling and drug design studies. However, as the availability of many protein sequences and complete genomes made possible new and powerful methods of sequence analysis [1-5], a database of many three dimensional models that is complete at the level of a family, organism, or functional network encourages new kinds of applications. For example, a good drug target is a protein that is likely to have high ligand specificity [6-8]. For example, when a human pathogen needs to be inhibited, a good target may be a protein whose binding site shape is different from related binding sites in all of the human proteins. Similarly, when a human metabolic pathway needs to be regulated, the target identification could focus on the particular protein in the pathway that has the binding site most dissimilar from its human homologs [9].

Protein Data Bank

The Protein Data Bank file format is a textual file format describing the three dimensional structures of molecules held in the Protein Data Bank. Most of the information in that database pertains to proteins, and the pdb format accordingly provides for rich description and annotation of protein properties. However, proteins are often crystallized in association with other molecules or ions such as water, ions, nucleic acids, drug molecules and so on, which therefore can be described in the pdb format as heteroatoms.

Biologists strive to understand the function of a protein. Ultimately, a laboratory experiment is needed to confirm the function of a protein, but a computational prediction can be useful both in itself and in suggesting an appropriate experiment. A variety of computational techniques have been studied, ranging from sequence comparison, to machine learning, to structure analysis and simulation.

Protein-Ligand Interactions

Many ligands interact with proteins in many biological processes such as enzyme catalysis, cell signaling etc. and regulate these processes. Information on protein-ligand interactions are accumulating at a faster pace through different experimental techniques such as X-ray crystallography, NMR, calorimetry, and equilibrium binding studies. Also in recent years an increasing number of small molecular ligands are designed and tested as potential inhibitors of enzymes involved in disease processes. Knowledge about these ligands and their target proteins will be valuable in

*Corresponding Author: *nulaka@gmail.com*

understanding the fundamental process of molecular recognition and will aid in the design of novel ligands and potent drugs [10]. In order to bind to a protein, a ligand has to exhibit correct shape and interaction properties complementary to the residues exposed towards the binding pocket of a target protein. Since protein-ligand binding is a process of mutual molecular recognition, rational drug design is greatly concerned with understanding the principles of molecular recognition. The statistical analysis of geometries of protein-ligand complexes provides a powerful tool to retrieve and correlate information about recognition patterns with respect to protein binding [11-13].

MATERIALS AND METHODS

Traditional synthesis of a series of new compounds utilizing combinatorial chemistry and high-throughput screening can be carried out at high cost and also are time consuming; whereas on the other hand, screening small molecule databases for novel compounds represents an alternative process. Docking various ligands to the protein of interest followed by scoring to determine the affinity of binding and to reveal the strength of interaction has become increasingly important in the context of drug discovery. Screening large databases of compounds can provide a feasible, alternative technique against high throughput screening, but depends on the fast and accuracy of the docking algorithm.

The development of a new drug is a lengthy, complex process, of which the identification of an appropriate lead molecule is a critical component [14]. Ever since, the pharmaceutical industry is under pressure to increase its success rate in bringing drugs to the market. It is estimated that on an average it should take 14 years to bring a compound from hit identification to an approved drug [15] and the costs associated with this process are enormous, with the majority of the expense being incurred in the development phase of the value chain.

As a consequence, there has been an increased investment in discovery in technologies aimed at achieving these goals. This includes investments in functional genomics, for improved identification and validation of therapeutic targets; in HTS [16] and combinatorial chemistry for hit identification. Drug design is most powerful when it is a part of an entire drug lead discovery process. Once a target has been identified, it is necessary to obtain accurate structural information. Crystal structures are the most common source of structural information for drug design, since structures determined to high resolution may be available. Computer-aided drug design begins with the identification of a potential ligand binding site on the target molecule. Ideally, the target site is a pocket or protuberance with a variety of potential hydrogen bond donors and acceptors, hydrophobic characteristics, and sizes of molecular surfaces. The ligand binding site can be the active site, as in an enzyme, an assembly site with another macromolecule, or a communication site necessary in the mechanism of the molecule.

Softwares and programs

Many softwares are available from scientific resources which has the ability to perform protein-ligand docking studies. In order to perform this task using these softwares the basic requirements is the submission of protein and ligand separately. It is this aspect which has become important in recent studies because separating a ligand from a protein requires another software other than the one used for docking studies. The problem becomes more stringent when the number of such proteins utilized in docking program increases. Mere less to say the softwares cannot really create protein and ligand files separately. In other words the softwares require manual intervention virtually to cut the ligand from the protein and paste it in a new window to save them as respective files. This files are used as input in any docking program (Auto Dock, DOCK, GOLD etc) instead of spending more time in separating protein and ligand a python based code was developed to perform the required task. Keeping this problem in view the program was written in such a way that it takes the protein file in protein Data Bank format (along with Hetero atoms such as chemical molecules, ligands, cofactors, water molecules) and returns protein file with only ATOM records devoid of hetero atoms.

RESULTS AND DISCUSSION

There has been no doubt that many soft wares would perform the similar task, however, all these are time consuming as the separation has to be done manually. While searching for the programs to write the code, many programming languages are tried but at the end python was selected. This is because python has various modules that can be used to define the protein structure in 3-dimensional format. In other words, python lines up with another language scientific python which has predefined modules that reads the 3D structural format of protein from protein data bank.

The three dimensional structure format of a protein contains number of records (single or multiple) as well as data releated to x,y,z coordinates. Solvation and charge terms, hetero atoms such as ligands, chemical molecules, cofactors and water molecules respectively, therefore to read all the lines in the input file it requires plenty of code to be written in any language hence to reduce this aspect Bio python based scientific program was implemented in python and the respective Scientific python function that reads the protein three dimensional file was called from python at the beginning the imported aspect to read and write the PDB files is given below.

```
import sys, string, re
from Scientific.IO.PDB import PDBFile
# read and write PDB-Files
```

The PDB file format specifies that the first two letters contain the right-justified chemical element name. A later modification allowed the initial space in hydrogen names to be replaced by a digit. Many programs ignore all this and treat the name as an

arbitrary left-justified four-character name. This makes it difficult to extract the chemical element accurately; most programs write the '"CA"' for C_alpha in such a way that it actually stands for a calcium atom. For this reason a special element field has been added later, but only few files use it. In the absence of an element field, the code attempts to guess the element using all information available.

An example of code from PDBfile

```
# Amino acid and nucleic acid residues
# amino_acids = ['ALA', 'ARG', 'ASN', 'ASP', 'CYS',
'CYX', 'GLN', 'GLU', 'GLY',
          'HIS', 'HID', 'HIE', 'HIP', 'HSD', 'HSE', 'HSP',
'ILE', 'LEU',
          'LYS', 'MET', 'PHE', 'PRO', 'SER', 'THR',
'TRP', 'TYR', 'VAL',
          'ACE', 'NME', 'NHE']

nucleic_acids = [ 'A', 'C', 'G', 'I', 'T', 'U',
            '+A', '+C', '+G', '+I', '+T', '+U',
             'RA', 'RC', 'RG', 'RU',
             'DA', 'DC', 'DG', 'DT',
             'RA5', 'RC5', 'RG5', 'RU5',
             'DA5', 'DC5', 'DG5', 'DT5',
             'RA3', 'RC3', 'RG3', 'RU3',
             'DA3', 'DC3', 'DG3', 'DT3',
             'RAN', 'RCN', 'RGN', 'RUN',
             'DAN', 'DCN', 'DGN', 'DTN',
             ]
def defineAminoAcidResidue(symbol):

      symbol = symbol.upper()
   if symbol not in amino_acids:
      amino_acids.append(symbol)

def defineNucleicAcidResidue(symbol):

   symbol = symbol.upper()
   if symbol not in nucleic_acids:
      nucleic_acids.append(symbol)
```

Initially the input file has to be submitted in .ent formats then the lines were read. According to the records levels and the atom numbers with respect to each amino acid residue are retained as such to be produced in the output file. The input file can be submitted in compressed formats or the URL can be submitted to extract files from PDB file the major and important records are ATOM and HETATM records. When the file was split into their respective record types, if atom records are present in the file devoid of other records, still the file can be read by soft wares. Therefore keeping this information in view white spaces are strict, non blank lines and other supported record types such as HEADER, CRYST 1, SCALE n, MODEL etc, and unsupported record types were removed and the corresponding file is saved as output.

Protein from PDB comes in different chains and the end of the chain is represented by TER record, while the end of the file is represented by END record. Here, in this work the program was written in such a way that first it reads the PDB file for consistency supported AND unsupported record types and then the file shall be split into an output file contains only ATOM RECORDS. All other records are removed with in a short period of time the program can be handle number of inputs and outputs. As passing the input file is easier and the generated output file can be used further in drug design studies. Further the program can be extended to create different files with respect to records in the PDB file depending on their usages in further studies the program shown here in python language is advantageous than the other soft wares because the program is fast, easy to use etc, example of complete code is given.

```
    #! /usr/bin/env python

    import sys, string, re
    from Scientific.IO.PDB import PDBFile
    # read and write PDB-Files
    inputfile = 'C:\pdb1acb.ent'
    outputfile = 'outputfile'+'_AtomCoorOnly'
    title = 'post-processed PDB file for +inputfile+
containing only the ATOM entries'
    try:
      structure_input = PDBFile(inputfile, 'r')
    except:
      sys.stderr.write ("can't open %s: %s %s\n" %
                  (inputfile,          sys.exc_type,
sys.exc_value))
    # open the PDB file for the input structure
    try:
      structure_output = PDBFile(outputfile, 'w')
    except:
      sys.stderr.write ("can't open %s: %s %s\n" %
                  (outputfile,          sys.exc_type,
sys.exc_value))
    # open the PDB file for the output structure
    structure_output.writeComment(title)
    # write comments to the first line(s) of the new
PDBfile

    while 1:
      pdbline = structure_input.readLine()
      type = pdbline[0]
      data = pdbline[1]
      if type == 'END':
        break
        # exit loop when end of PDBfile is reached
      elif not type == 'ATOM':
        # if line doesn't describe an ATOM entry:
        continue
      else:
        position = data['position']
        # if ATOM position is described:
        # write the line to the output file
        structure_output.writeLine(type, data)
        #    write    transformed    coordinates    to
structure_output
    structure_output.close()
```

example_readProtein.py - C:\example_readProtein.py

File Edit Format Run Options Windows Help

```
#! /usr/bin/env python


import sys, string, re
from Scientific.IO.PDB import PDBFile
# read and write PDB-Files

inputfile = 'C:\pdb1acb.ent'


outputfile = 'outputfile'+'_AtomCoorOnly'

title = 'post-processed PDB file for +inputfile+ containing only the ATOM entrie

try:
  structure_input = PDBFile(inputfile, 'r')
except:
  sys.stderr.write ("can't open %s: %s %s\n" %
                    (inputfile, sys.exc_type, sys.exc_value))
# open the PDB file for the input structure

try:
  structure_output = PDBFile(outputfile, 'w')
except:
  sys.stderr.write ("can't open %s: %s %s\n" %
                    (outputfile, sys.exc_type, sys.exc_value))
# open the PDB file for the output structure


structure_output.writeComment(title)
# write comments to the first line(s) of the new PDBfile

while 1:
  pdbline = structure_input.readLine()
  type = pdbline[0]
  data = pdbline[1]
  if type == 'END':
    break
    # exit loop when end of PDBfile is reached
  elif not type == 'ATOM':
```

Ln: 19 Col: 0

Figure 1**:** Screen-shot image of code.

CONCLUSION

It has been emphasized in literature that protein and ligands have to be submitted separately before proceeding for docking or protein-ligand interaction studies. A program written in python uses scientific python sub modules to answer the major problem faced by drug design scientists. The problem has been successfully answered using python lines of code where the PDB input file is split into its respective protein only file with all its related data intact. The program reads one file at a time, thus producing the required file within few seconds, suggests the fastness of such program and this program finds utility in drug design studies. Further extensions can also be made by calling multiple files at a time and creating files with respect to records present in PDB file.

REFERENCES

1. Wolf, Y., Grishin, N., & Koonin, E. *J. Mol. Biol.* **299**, 897–905 (2000).
2. Yona, G., Linial, N. & Linial, M. *http://protomap.stanford.edu/*
3. Benson, D. *et al. Nucleic Acids Res* **28**, 15–18 (2000).
4. Fischer, D. and Eisenberg, D. *Proc. Natl. Acad. Sci. USA* **94**, 11929–11934 (1997).
5. Rychlewski, L., Zhang, B., & Godzik, A. *Fold. Des.* **3**, 229–238 (1998).

6. Huynen, M., Doerks, T., Eisenhaber, F. & Orengo, C. *J. Mol. Biol.* **280**, 323–326 (1998).
7. Teichmann, S. A., Park, J., & Chothia, C. *Proc. Natl. Acad. Sci. USA* **22**, 14658–14663 (1998).
8. Jones, D. T. *J. Mol. Biol.* **287**, 797–815 (1999).
9. Guex, N., Diemand, A., & Peitsch, M.C. *Trends Biochem. Sci.* **24**, 364–367 (1999).
10. Hendlich M., Bergner A., Günther J., Klebe G., *J. Mol. Biol.,* 2003, **326**, 607-620
11. Günther J., Bergner A., Hendlich M., Klebe G., *J. Mol. Biol.*, 2003, **326**, 621-636
12. Schmitt S., Kuhn D., Klebe G., *J. Mol. Biol.*, 2002, **323**, 387-406
13. Verdonk M. L., Cole J. C., Taylor R.,. *J.Mol. Biol.*, 1999, **289**, 1093-1108
14. Gohlke H., Hendlich M., Klebe G., *J. Mol. Biol.,* 2000, **295**, 337-356
15. Alvarez JC. High-throughput docking as a source of novel drug leads. *Curr Opin Chem Biol*. 8, 2004, 365–370
16. Myers S, Baker A. Drug discovery – an operating model for a new era. *Nat. Biotechnol.* 19, 2001, 727–730

****