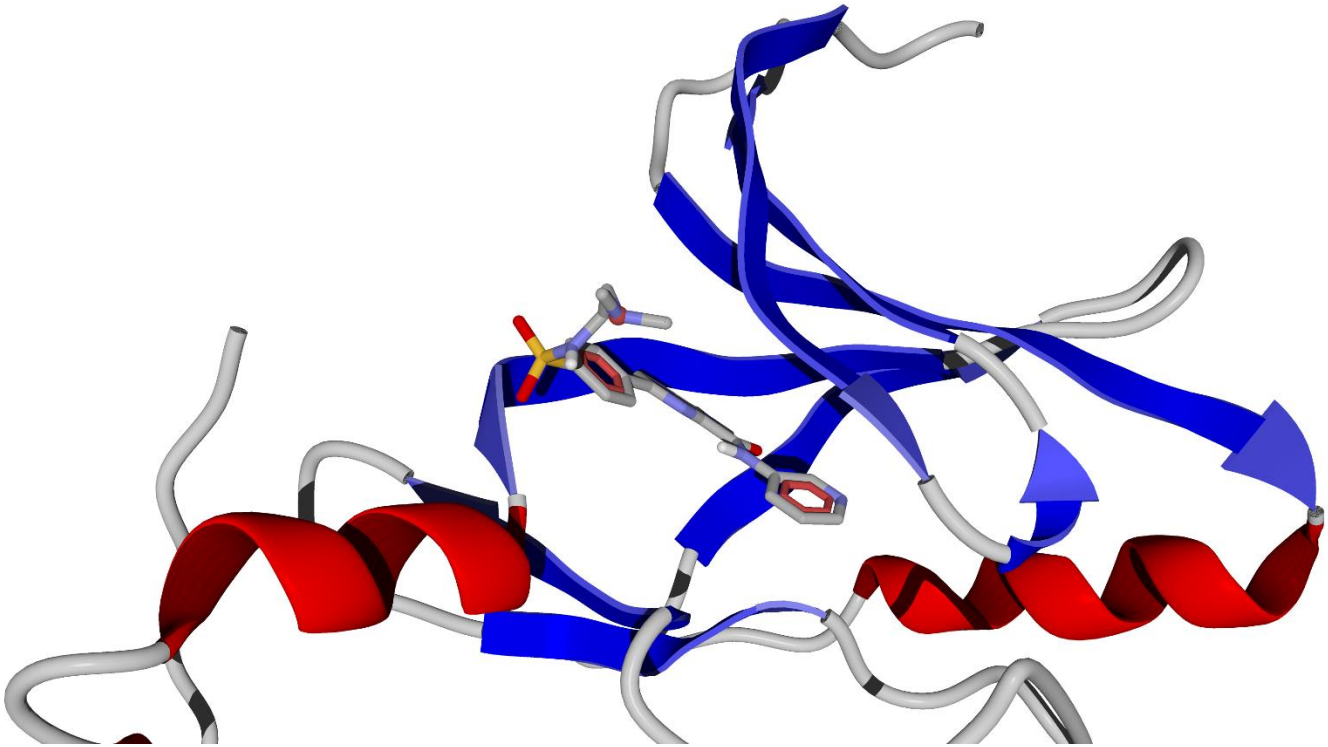
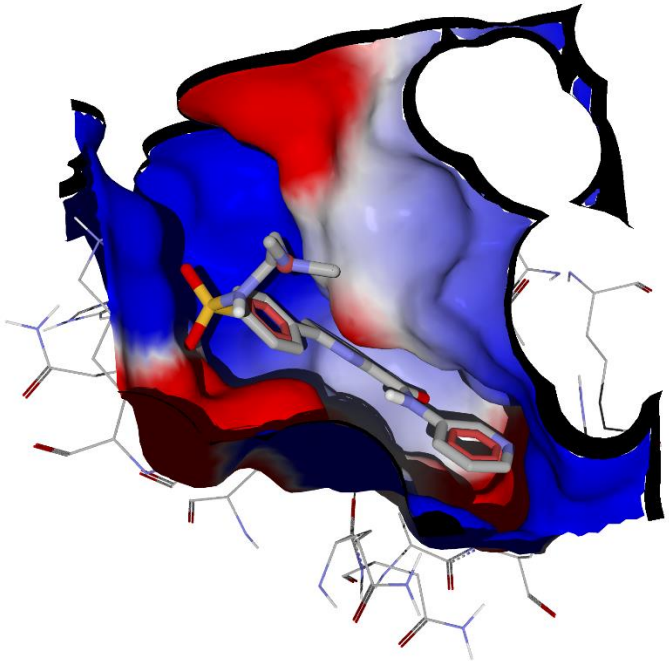
python™
www.python.org

Docking Root Mean Square Deviation (RMSD) in Python

By Prof. Walter F. de Azevedo Jr.



Any program to calculate root mean square deviation (RMSD) should read the atomic coordinates for the ligand (crystallographic position) and for the pose (computer-generated position). Here we describe a program to calculate RMSD, named *rmsd_pdb.py*. This program reads two files in Protein Data Bank (PDB)(Berman, Westbrook, Feng *et al*. 2000; Berman, Battistuz, Bhat *et al*. 2002; Westbrook *et al*., 2003) format and calculates the RMSD between them. This program was written in Python 3 programming language and uses the NumPy lbrary (numpy.org).



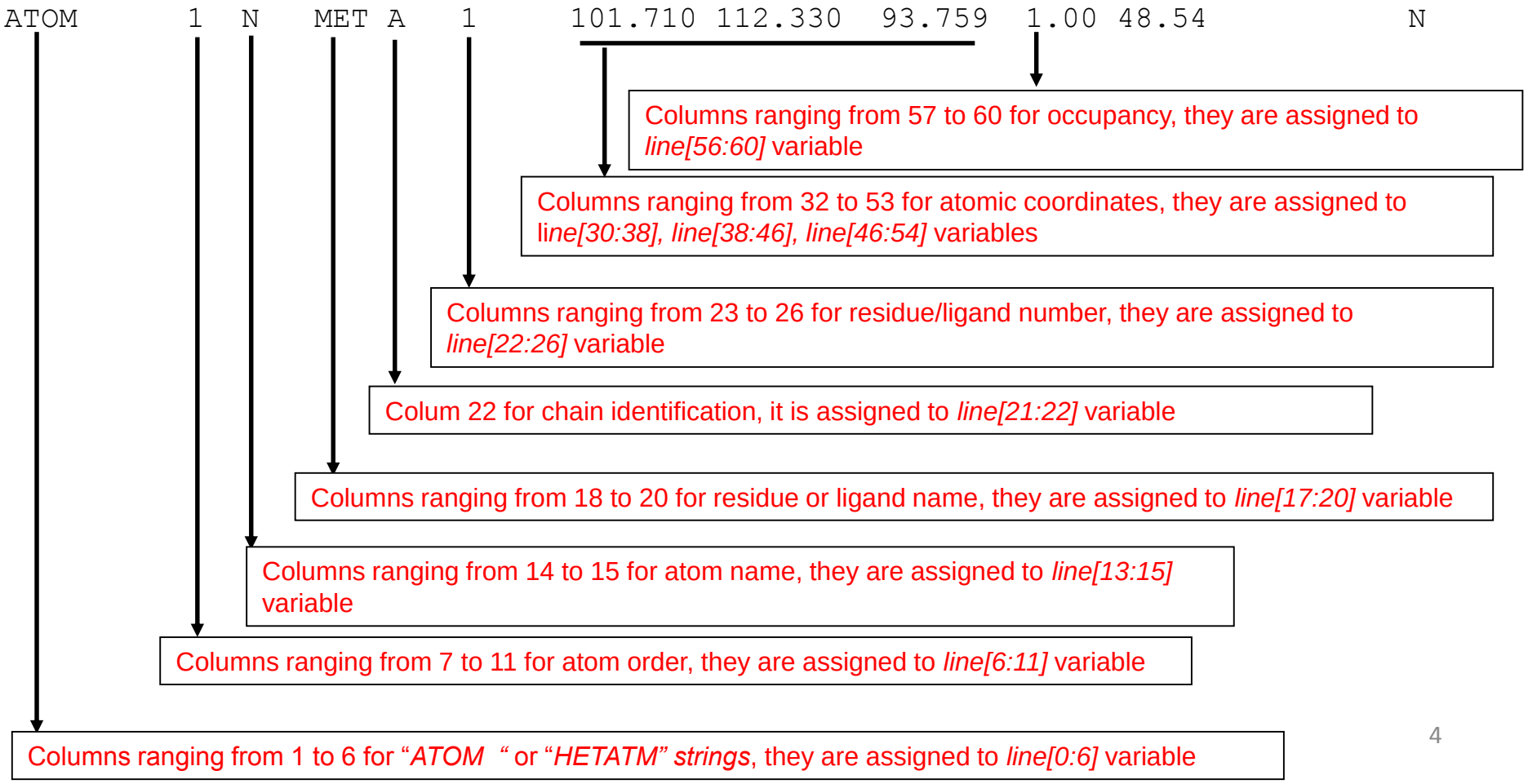
Electrostatic interaction for cyclin-dependent kinase and an inhibitor (PDB access code: 4ACM), as shown by the program *Molegro Virtual Docker* (MVD) (Thomsen & Christensen, 2006).

In molecular docking simulations, the best pose is the one closer to the structure determined by x-ray crystallography. Therefore, we must establish a methodology that assesses the quality of the computer-generated solution (pose). This quality can be calculated using the root mean-square deviation (RMSD), which is a measure of the differences between values predicted by a model and the values actually observed from the object being modeled or estimated (protein-igand complex). The docking RMSD is calculated between two sets of atomic coordinates, in this case, one for the crystallographic structure (x_1, y_1, z_1; the object being modeled) and another for the atomic coordinates obtained from the docking simulations (x_2, y_2, z_2; predicted model) (Heberlé & De Azevedo, 2011). A summation is then taken over all N non-hydrogen atoms being compared, using the following equation:

$$RMSD = \sqrt{\frac{\sum_{j=1}^{N}\left[\left(x_{1,j}-x_{2,j}\right)^2+\left(y_{1,j}-y_{2,j}\right)^2+\left(z_{1,j}-z_{2,j}\right)^2\right]}{N}}$$

(equation 1)

Below we have a typical line for atomic coordinates in a Protein Data Bank (PDB) file. Each field brings a specific information related to the atoms in the structure. These lines starts either with “ATOM ” or “HETATM” strings. Additional information for each field is given red.



Last fields are indicated below.

```
ATOM      1  N   MET A   1      101.710 112.330  93.759  1.00 48.54           N
```

Columns ranging from 62 to 65 for B-values, they are assigned to *line[61:65]* variable

Colunan 77-77 for chemical element, it is assigned to *line[76:77]* vairable

Besides the fields indicated beforehand, we have a field for segment identification, columns ranging from 73 to 76, which are assigned to *line[72:76]* variable. In addition, we have a field for atomic charge, columns ranging from 79 to 80), which are assigned to *line[78:80]* variable.

python™
www.python.org

Program to calculate docking RMSD
Program: *rmsd_pdb.py*

Abstract
Program to calculate docking RMSD based on the atomic coordinates present in two files in the Protein Data Bank (PDB) (Berman, Westbrook, Feng *et al*. 2000; Berman, Battistuz, Bhat *et al*. 2002; Westbrook *et al*., 2003) format. Both files should have exactly the same atom order. The program reads both file names and calculates RMSD in Å, showing the results on the screen.

In main() function, the program reads the PDB file names and calls *read_PDB_return_coord(my_file)* function twice. This function returns three arrays with the atomic coordinates for each input PDB file. These arrays are passed to the function *calc_rmsd(x1,y1,z1,x2,y2,z2)*, that carries out RMSD calculation. The code for main function is shown below.

```
def main():
    # Read file names
    my_file1 = input("\nType first PDB file name  => ")
    my_file2 = input("\nType second PDB file name => ")
    # Call read_PDB_return_coord() to get atomic coordinates as arrays
    x1,y1,z1 = read_PDB_return_coord(my_file1)
    # Call read_PDB_return_coord() to get atomic coordinates as arrays
    x2,y2,z2 = read_PDB_return_coord(my_file2)
    # Call calc_rmsd()
    rmsd = calc_rmsd(x1,y1,z1,x2,y2,z2)
    # Get the size of the array
    n = len(x1)
    # Show results
    print("\nDocking RMSD = %6.3f"%rmsd," A for %5d"%n," atoms.")
main()
```

```
def read_PDB_return_coord(pdb_in):
    """Function to read atomic coordinates from a PDB file"""
    # Import libraries
    import sys
    import numpy as np
    # Setup empty lists
    x = []
    y = []
    z = []
    # Try to open PDB file
    try:
        my_PDB_fo = open(pdb_in,"r")
    except IOError:
        sys.exit("\nI can't file "+pdb_in+" file!")
    # Looping through PDB file
    for line in my_PDB_fo:
        if line[0:6] == "HETATM" or line[0:6] == "ATOM  ":
            if line[13:14] != "H":
                x.append(float(line[30:38]))
                y.append(float(line[38:46]))
                z.append(float(line[46:54]))
    # Convert list to array
    x_coord = np.array(x)
    y_coord = np.array(y)
    z_coord = np.array(z)
    # Return arrays
    return x_coord, y_coord, z_coord
```

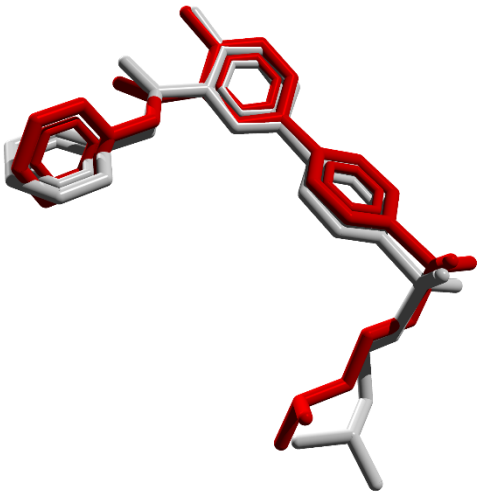
python™
www.python.org

The *read_PDB_return_coord()* function reads the atomic coordinates from a PDB file and returns the non-hydrogen coordinates as three NumPy arrays. In the for loop shown in red here, the function looping through each line of a PDB file and assigns the atomic coordinates to three lists, named *x*,*y*, and *z*. This assignment occurs only if the *line[0:6]* is equal to “ATOM “ or “HETATM”. In addition, the function tests whether the atom label is hydrogen “H”. This function only assigns the coordinates for non-hydrogen atoms to the *x*, *y*, and *z* lists. This function converts these lists to NumPy arrays, named *x_coord*, *y_coord*, and *z_coord.*

```
def calc_rmsd(x1,y1,z1,x2,y2,z2):
    """Function to calculate RMSD"""
    # Import library
    import math

    # Set count to zero
    sum_d2 = 0

    # Assign len(x1) to n (size of array)
    n =  len(x1)

    # Looping through all atomic coordinates to calculate sum of d2
    for i in range(n):
        # Calculate the square of Euclidian distance between two points
        d2 =  ( x1[i] - x2[i] )**2 + (y1[i] - y2[i])**2 + (z1[i] - z2[i])**2
        sum_d2 += d2

    # If n> 0 calculate rmsd
    if n > 0 :
        rmsd = math.sqrt(sum_d2/n)
    else:
        rmsd =  None
    # Return rmsd
    return rmsd
```

The *calc_rmsd()* function calculates RMSD based on the equation 1 and returns this value as a float.

We are going to use the atomic coordinates generated by a protein-docking simulation program called *Molegro Virtual Docker* (MVD) (Thomsen & Christensen, 2006). Default docking parameters for MVD program were used to generate the atomic coordinates for the pose. We saved the atomic coordinates for the crystallographic position for the active ligand in a file named *ligand.pdb*, and the lowest RMSD pose in a file named *pose.pdb*. The figure below shows the docking simulation results. The pose is in red and the crystallographic position in light grey. The structure used in the simulation was the cyclin-dependent kinase 2 in complex with 3-AMINO-6-(4-{[2-(DIMETHYLAMINO)ETHYL]SULFAMOYL}PHENYL)- N-PYRIDIN-3-YLPYRAZINE-2-CARBOXAMIDE (PDB access code: 4ACM) (Berg *et al.* 2012)



Crystallographic and pose positions.

To run *rmsd_pdb.py,* type *python rmsd_pdb.py*, and then type the PDB file names, as shown below.

```
C:\Users\Walter\>python rmsd_pdb.py

Type first PDB file name  => ligand.pdb

Type second PDB file name => pose.pdb

Docking RMSD =  1.116  A for    31  atoms.

C:\Users\Walter\>.....
```

The program *rmsd_pdb.py* obtained the same result shown by the program MVD.
To run this tutorial, you need the following files: *rmsd_pdb.py, ligand.pdb and pose.pdb*.
All files should be in the same folder.

Berg S, Bergh M, Hellberg S, Högdin K, Lo-Alfredsson Y, Söderman P, von Berg S, Weigelt T, Ormö M, Xue Y, Tucker J, Neelissen J, Jerning E, Nilsson Y, Bhat R Discovery of novel potent and highly selective glycogen synthase kinase-3ß (GSK3ß) inhibitors for Alzheimer's disease: design, synthesis, and characterization of pyrazines. J Med Chem. 2012; 55(21):9107-19.

Berman HM, Westbrook J, Feng Z, *et al.* The Protein Data Bank. Nucleic Acids Res 2000; 28: 235-42.

Berman HM, Battistuz T, Bhat TN, *et al.* The Protein Data Bank. Acta Crystallogr D Biol Crystallogr 2002; 58(Pt 6 No 1): 899-907.

Heberlé G., De Azevedo Jr. WF. Bio-inspired Algorithms Applied to Molecular Docking Simulations. Curr Med Chem 2011; 18: 1339-52.

Thomsen R, Christensen MH. MolDock: a new technique for high-accuracy molecular docking. J Med Chem. 2006;49:3315–21.

Westbrook J, Feng Z, Chen L, Yang H, Berman HM. The Protein Data Bank and structural genomics. Nucleic Acids Res 2003; 31(1): 489-491.

Last update on July 6th 2016.

This text was produced in a DELL Inspiron notebook with 6GB of memory, a 750 GB hard disk, and an Intel® Core® i5-3337U CPU @ 1.80 GHz running Windows 8.1. Text and layout were generated using PowerPoint 2013 and graphical figures were generated by *Molegro Virtual Docker* (Thomsen & Christensen, 2006). This tutorial uses Arial font.

Author

I graduated in Physics (BSc in Physics) at University of Sao Paulo (USP) in 1990. I completed a Master Degree in Applied Physics also at USP (1992), working under supervision of Prof. Yvonne P. Mascarenhas, the founder of crystallography in Brazil. My dissertation was about X-ray crystallography applied to organometallics compounds (De Azevedo Jr. *et al.*,1995).

During my PhD I worked under supervision of Prof. Sung-Hou Kim (University of California, Berkeley. Department of Chemistry), on a split PhD program with a fellowship from Brazilian Research Council (CNPq)(1993-1996). My PhD was about the crystallographic structure of CDK2 (Cyclin-Dependent Kinase 2) (De Azevedo Jr. *et al.*,1996). In 1996, I returned to Brazil. In April 1997, I finished my PhD and moved to Sao Jose do Rio Preto (SP, Brazil) (UNESP) and worked there from 1997 to 2005. In 1997, I started the Laboratory of Biomolecular Systems-Department of Physics-UNESP - São Paulo State University. In 2005, I moved to Porto Alegre/RS (Brazil), where I am now. My current position is coordinator of the Laboratory of Computational Systems Biology at Pontifical Catholic University of Rio Grande do Sul (PUCRS). My research interests are focused on application of computer simulations to analyze protein-ligand interactions. I'm also interested in the development of biological inspired computing and machine learning algorithms. We apply these algorithms to molecular docking simulations, protein-ligand interactions and other scientific and technological problems. I published over 160 scientific papers about protein structures and computer simulation methods applied to the study of biological systems (H-index: 36). These publications have over 4000 citations. I am editor for the following journals: