Molecular dynamics data analysis example

Peter Kekenes-Huskey, 2014

Here we'll go over some basic analysis of a molecular dynamics trajectory using molecular dynamics

Tutorial

File prep

This is here for provenance...

```
 2014  perl -e 'for(0..1630){print "$_ "}'
 2015  perl -e 'for(0..1630){print "$_ "}' > idx
 2019  catdcd -o small.pdb -otype pdb -s step2_solvator.pdb -stype pdb -i idx -first 1
-last 1 x.dcd
 2023  catdcd -o small.dcd -s step2_solvator.pdb -stype pdb -i idx  -stride 10 x.dcd
```

View protein in VMD

The data I have provided is that of a small protein (parvalbumin) and its dynamics over a 1-2 nanosecond interval, as computed from molecular dynamics. I've provided a few instructions to visualize the protein

Best to follow tutorial for using vmd, but here are some basic steps

- Open small.pdb
- Load data into molecule (add small.dcd)
- Graphics->Representations (choose Ribbon under Drawing Methods)
- Hit triangle in bottom right corner to see the trajectory

What you should see is a series of helical regions (alpha helices) that are linked by disordered 'loops'. Roughly speaking, the dynamics provide an indication of the stability, so the fairly static helices are generally more stable than the highly flexible loop regions. The command below will show in the image in a separate window

In [1]:
```
import Image
img = Image.open("parv.png")
img.show()
```

Analysis

Now we'll pick two atoms at either end of a helix and measure the distance over time. Within vmd it is possible to select those atoms, show a line linking them and plot/save the distances vs time.

- Graphics->Representation, click 'Create Rep', enter "name CA" into the 'Selected Atoms' box, and choose the VDW drawing method
- Return to the main visualization window and press '2' to enter the distance selection mode. Click two spheres (C-alpha [CA] atoms). If all goes well, you should see a line linking those two atoms.
- Once a line is shown linking the two atoms, select Graphics->Labels, change the 'Atoms' pull down menu to 'Bonds'. You should see an entry corresponding to the line you drew.
- Select/Highlight the entry, select the 'Graph' tag, and press the graph button. You'll see that the distance changes as a function of time, but stays fairly close (within 1-2 Ang) to the mean.
- Choose File->Save Coordinates and enter "name CA and (resid 25 or resid 39)" into the selected atoms box. This search string saves the 3D coordinates of the C-alpha atom for residue 25 and 39, which correspond to two residues on either end of an alpha helix. Hit save and call the file 'positions.txt'

In [2]:
```
import Image
img = Image.open("label.png")
img.show()
```

Python analysis
Now we will analyze these data with python. Note: VMD saves additional information in the txt file that we don't need, so it needs to be removed. In Linux/Mac, you could type 'grep ATOM position.txt' and copy the output to a new file. In windows, you might just have to select the entries manually! We'll presume now that you have a positions.txt file that contains only entries that begin with 'ATOM'.

In [3]:
```
# open file and extract columns 6-8, which correspond to the 3D coordinates
data = np.loadtxt('positions.txt', usecols=[6,7,8])
```

Note that positions for two atoms are reported in the file, so we need to separate them. You can do this by saving every other coordinate

In [4]:
```
# takes every 2nd entry, starting from index 0 (resid 25)
resid25 = data[0::2,]
# uncomment to show the xyz coordinates for resid 25
#print resid25
```

In [5]:
```
# takes every 2nd entry, starting from index 1 (resid 39)
resid39 = data[1::2,]
```

Get coordinates at a single time slice and compute distance

In [6]:
```
timeIndex = 0  # first time step
resid25xyz = resid25[timeIndex,:]
resid39xyz = resid39[timeIndex,:]

# get distance
difference = resid39xyz - resid25xyz
diffsquared = difference*difference
diffSqdSum = np.sum(diffsquared)
distance = np.sqrt(diffSqdSum)
print distance
```

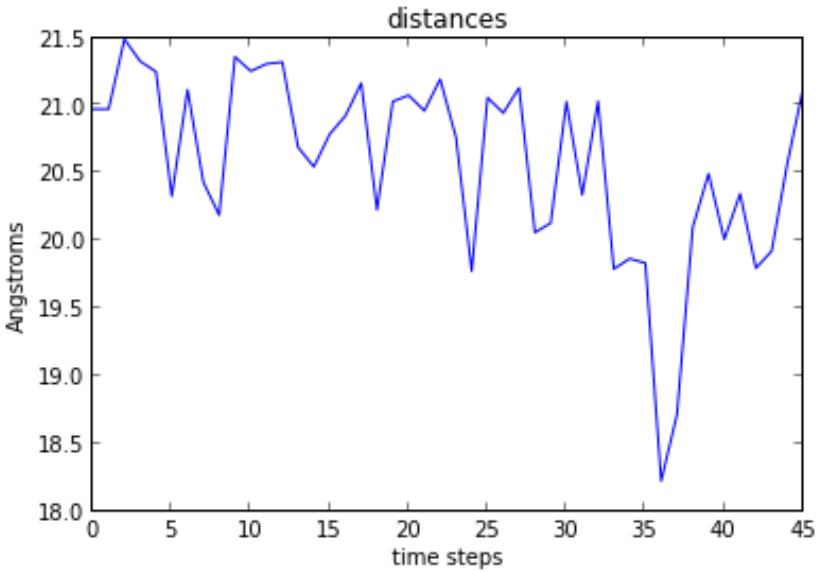
```
20.9738846664
```

Make this into a function instead that operates on ALL time slices

In [7]:
```
def dist(res1, res2):
    #return np.linalg.norm(res2-res1) # same thing,fewer steps
    diff = res1-res2
    sums = np.sum(diff*diff,axis=1) # sums columns corresponding to xyz coord
    return np.sqrt(sums)
#return np.sqrt(np.sum(diff*diff),axis=0,axis=0)
```

In [8]:
```
dists = dist(resid25, resid39)
# reports size of coordinate matrix [numTimeSteps,numCoordinates] and returns
times = np.shape(resid25)[0]
times = np.arange(times)
plot(times,dists)
xlabel("time steps")
ylabel("Angstroms")
title("distances")
```

Out[8]: <matplotlib.text.Text at 0x4536110>



Exercise 1

Compare the dynamics of a 'floppier' region of the protein (say an atom from the center of two loop regions) with the stable helix we just analyzed

- Repeat the process for reporting the distances between two atoms (this time with the floppy regions)
- Plot the distance wrt time for both the floppy and helical regions on the same graph.
- Compute the standard deviation for both regions and plot as a bargraph (hint: google for 'matplotlib and bar plot' for tips)

What does the standard deviation tell you about the structure or stability of the two regions?

Exercise 2

Repeat Ex. 1 for an atom at the center of two adjacent helices. How do those distances compare to the previous distances you computed?

Exercise 3

Compute another interesting geometric property as a function of time. You may need to explicitly loop over coordinates, e.g.:

In [9]:
```
numTimeSteps  = np.shape(resid25)[0]
```

```
xyCoords = np.zeros([numTimeSteps,2])
for i in np.arange(numTimeSteps):
    xyCoords[i,:] = resid25[i,0:2]

#print xyCoords
```

In time, I will introduce you to a number of approaches for analyzing protein dynamics and stability. These are especially useful tools for understanding how proteins carry out their function, binding other proteins, etc.

In [9]: