mdanalysis

November 29, 2022

[ ]:
```
import MDAnalysis as mda
from MDAnalysis.analysis.dihedrals import Dihedral
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.ticker as tck
```

[ ]:
```
#axis
fontsize=28
figureParameters = {'figure.figsize' : (12,8),
                    'legend.fontsize': fontsize*0.7,
                    'axes.labelsize' : fontsize,
                    'axes.titlesize' : fontsize,
                    'xtick.labelsize': fontsize*0.8,
                    'ytick.labelsize': fontsize*0.8,
                    'xtick.direction': "in", # tick marks inside the frame
                    'ytick.direction': "in", # tick marks inside the frame
                    'axes.linewidth' : 3,
                    'axes.titlepad'  : 25}


def prettyTicks(ax):
    # Add tick marks on all sides of the figure
    ax.xaxis.set_ticks_position('both')
    ax.yaxis.set_ticks_position('both')

    #ax.xaxis.set_major_locator(tck.MultipleLocator(2))
    #ax.yaxis.set_major_locator(tck.MultipleLocator(0.01))

    ax.yaxis.set_minor_locator(tck.AutoMinorLocator())
    ax.xaxis.set_minor_locator(tck.AutoMinorLocator())

    ax.tick_params(which='minor', length=6, width=2, color='black')
    ax.tick_params(which='major', length=12, width=2, color='black')
```

[ ]:
```
# import the topology and trajectory files
u = mda.Universe('DBE_298_EQ_BOX.pdb','trajectory.0.dcd')
```

```
[ ]: # get objects for the atoms in each residue
     res = u.select_atoms('resname M1').groupby('resids')

     # get atoms in residues for calculating the dihedral
     # 1,0,4,2 correspond to Br-Cb-Cb-Br
     ags = [value.atoms[[1,0,4,2]] for key, value in res.items()]

     def getDihedrals(ags,start,nbins):
         # calc dihedrals over whole trajectory
         R = Dihedral(ags).run(start=start)

         # convert dihedral angle from degrees to radians for 0 - 2pi
         dhds = R.results.angles.flatten()
         dhds = (dhds * np.pi/180)
         dhds[dhds < 0] += 2*np.pi

         # create histogram
         #nbins = 400
         hist, bins = np.histogram(dhds,bins=nbins)
         bw = bins[1] - bins[0]
         area = bw * np.sum(hist)
         hist = hist/area
         bin_centers = 0.5*(bins[1:]+bins[:-1])
         return hist,bins,bin_centers, bw
```

```
[ ]: nbins = 400
     hist, bins, bin_centers, bw = getDihedrals(ags,0,nbins)
```

```
[ ]: # PLOTTING
     plt.rcParams.update(figureParameters)
     fig = plt.figure()
     ax = fig.gca()

     # plot histogram
     #ax.bar(bins[:-1],hist,width=0.02)

     # plot line graph
     ax.plot(bin_centers,hist,color='black',linewidth=2)
     ax.set(xlabel='Dihedral angle',ylabel='Probability')

     prettyTicks(ax)

     plt.savefig("probability.jpg")
     plt.show()
```

```
[ ]: # get probabilities of trans and gauche conformations
     peak1 = 1.3
```

```
peak2 = 3.2
peak3 = 5.0
upp = 2*np.pi

p1 = int(peak1/upp * nbins)
p2 = int(peak2/upp * nbins)
p3 = int(peak3/upp * nbins)

min1 = p1+np.argmin(hist[p1:p2])
min2 = p2+np.argmin(hist[p2:p3])

#integrate from min1 - min2
ptrans = bw * np.sum(hist[min1:min2])
pgauche = 1 - ptrans
print("P(trans)  = {ptrans}%\nP(gauche) = {pgauche}%".
 ↪format(ptrans=round(ptrans*100,2),pgauche=round(pgauche*100,2)))
```

[ ]:
```
temp = 298
dG = -1*(8.314/1000)*temp*np.log(ptrans/pgauche)
print('Free energy: {} kJ/mol'.format(round(dG,2)))
```

[ ]:
```
```