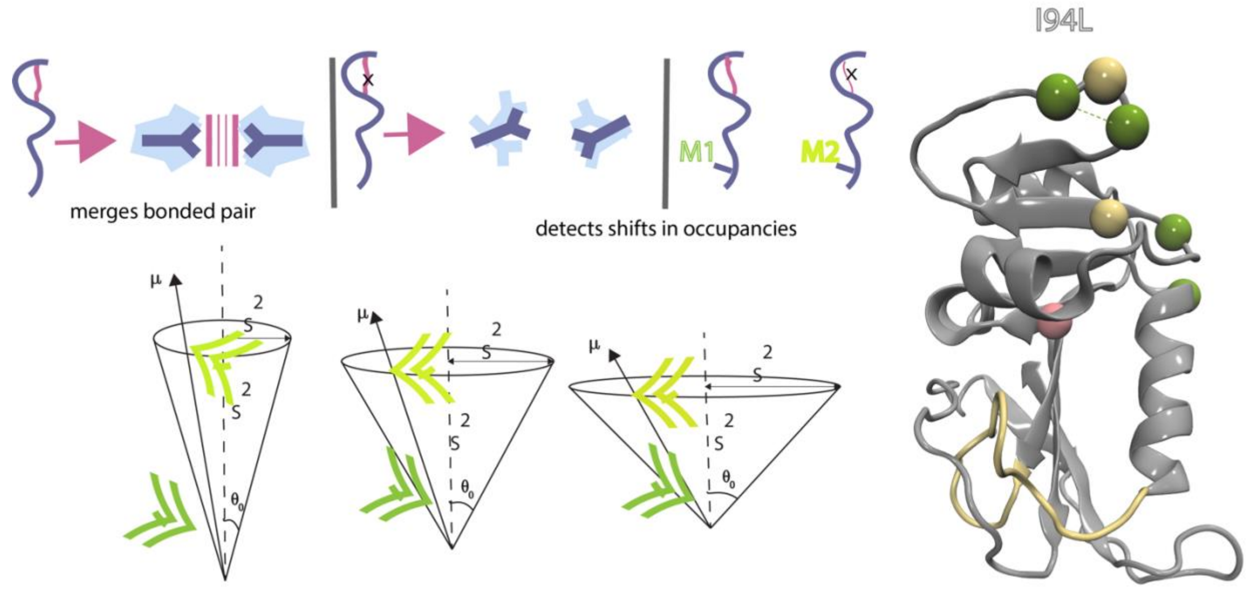
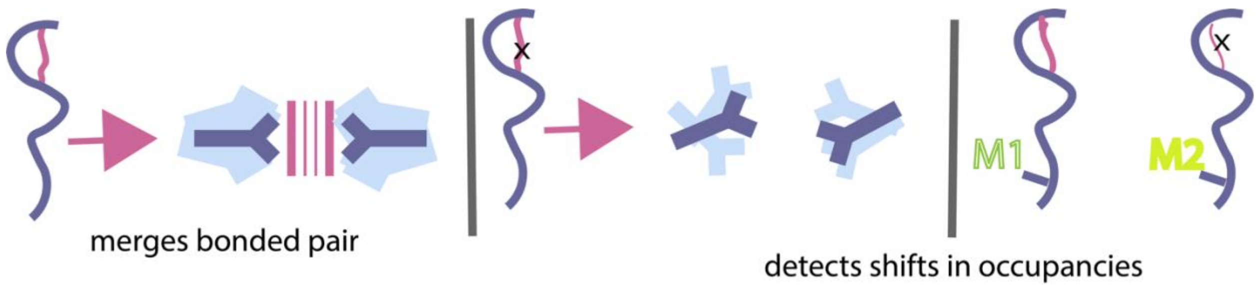
Hydrogen Bond Analysis Guide



MIDSTLAB, 2022

Ebru Cetin (ebrucetin@sabanciuniv.edu)

In this guide we will discuss in detail the calculation of hydrogen bond occupancy shift, given a set of hydrogen bonds acquired from VMD for a selected number of systems (eg. wild type protein and several of its mutants). This input can be created by Timeline Toolbox of VMD. The acquired set of hydrogen bonds are compared with each other to find out if there are any significant shifts in the hydrogen bond occupancies between systems.



To reach our aim, we must first select each index from VMD. Timeline Hydrogen Bond information will be frame-wise and assigned with indices. In order to play with them in the python script, we first should learn the address of every contributor**.**

1. index_extraction.tcl

After the system is properly wrapped and aligned (needed for correctly depicting ligand-protein complex hydrogen bonds) this script should be modified as follows,

```
set pro [atomselect top "protein"] # protein selection

set dhf [atomselect top "resname DHF"] # ligand selection

set outfile [open i94l-index.dat w] % !! This line needs to be changed before sourcing
```

Then the script should be run on Tk Console by the command:

```
source index_extraction.tcl
```

2. merging_bonds.py

This script does two things; first, merges all side-chain hydrogen bonds for a residue for the given time trajectory, second, depicts the bonds which shows the selected amount of occupancy shift from the wild type value (this is the reference system in our case). We have found focusing on hydrogen bond occupancies that shift by a cutoff value of 30% to work well for the DHFR systems we have worked with. For details on how we arrived at this conclusion, please see the JCIM 2022 manuscript. Nevertheless, users might find it useful to optimize the cutoff percentage for their system of interest.

In merging, all hydrogen bonds belonging to the same residue are taken under a key and occupancies are merged time frame-wise. Thus, if there are two hydrogen bonds on the same pair of residues in the same time frame, these count as a single value in the final occupancy data.

Before running this script 2 .dat files are required;

1. name-index.dat : which can be retrieved by index_extraction.tcl

2. name-hbonds.dat: which can be retrieved by VMD > Extensions > Timeline > Hbonds > writetoafile

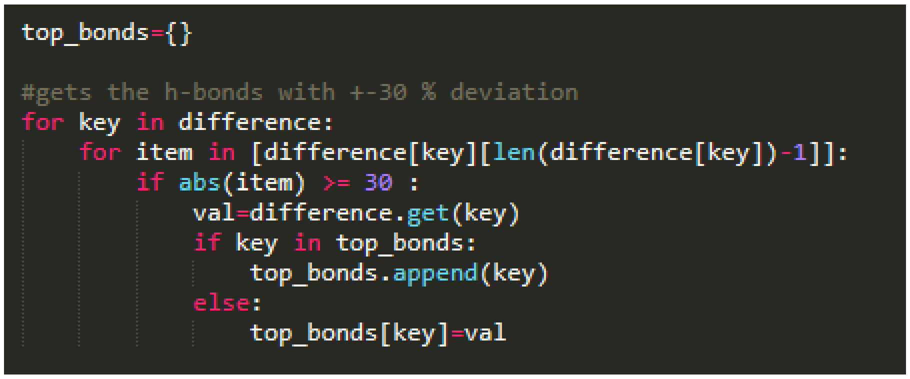
The script produces three output files;

1. ligand hydrogen bond information (if any, please change the ligand name to the one on your system)

2. merged hydrogen bonds information

3. information of bonds differing by +- 30 % from WT

30 % limit can be changed from the lines,

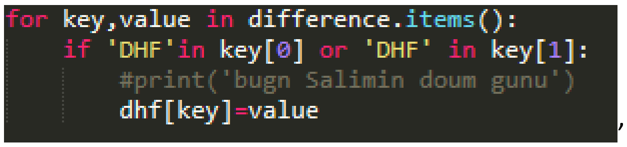


```
top_bonds={}

#gets the h-bonds with +-30 % deviation
for key in difference:
    for item in [difference[key][len(difference[key])-1]]:
        if abs(item) >= 30 :
            val=difference.get(key)
            if key in top_bonds:
                top_bonds.append(key)
            else:
                top_bonds[key]=val
```

You can change abs(item) >= 30 to any limit you may wish.

Ligand name on this script should match the VMD naming. So, one needs to change ‘DHF’s in the following lines to the ligand name depicted in the PDB file of interest to the user.



```
for key,value in difference.items():
    if 'DHF'in key[0] or 'DHF' in key[1]:
        #print('bugn Salimin doum gunu')
        dhf[key]=value
```

,

3. If_unique.py & plot_unique.py

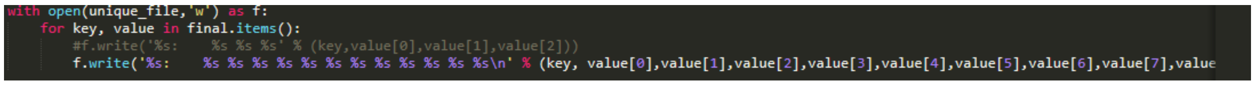
Here the comparison has been made within the set studied. Every mutant that has been processed by merging_bonds.py is taken into the mutant matrix and compared with the others one by one. Here actually we are not bound by the 30% difference limit.

The input is;

1. merged-bonds.txt

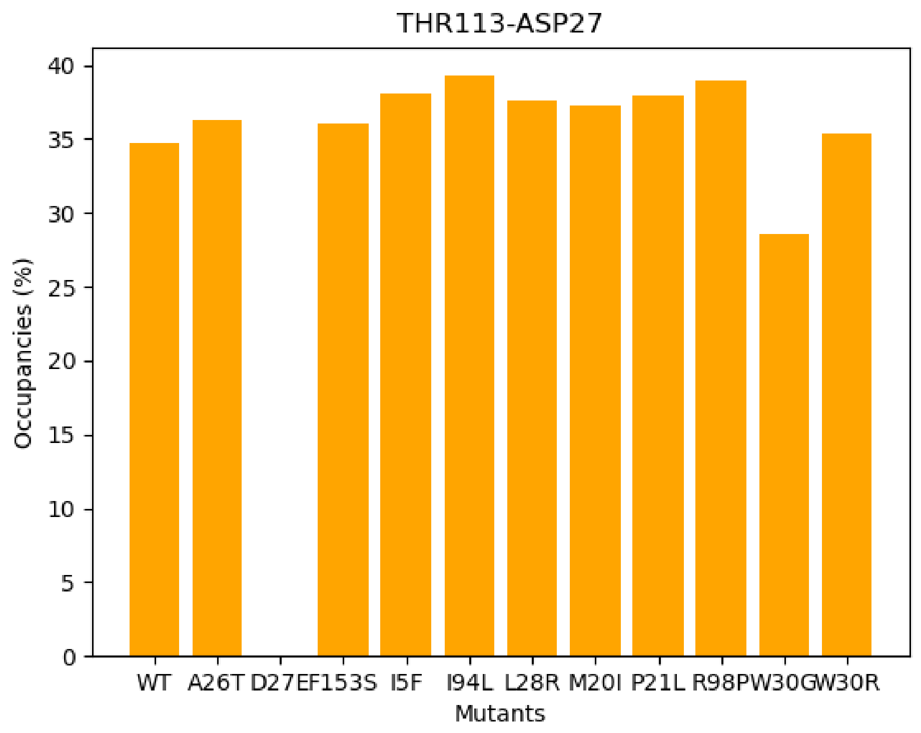
The name called target_name='mutant' should be changed to mutant name. Mutant matrix we have used is made for the single mutations. You can change mutant names to 'None' to convert them into one step mutations.

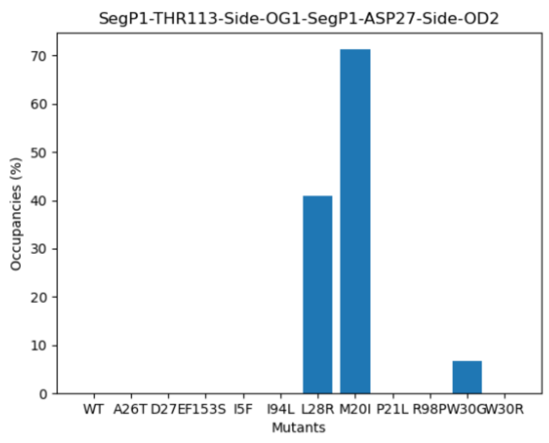
Your mutant matrix size and your output line to write the files, the following characters %s and corresponding value should be equal.



```
with open(unique_file,'w') as f:
    for key, value in final.items():
        #f.write('%s:    %s %s %s' % (key,value[0],value[1],value[2]))
        f.write('%s:    %s %s %s %s %s %s %s %s %s %s %s %s\n' % (key, value[0],value[1],value[2],value[3],value[4],value[5],value[6],value[7],value
```

For example, for one of the mutants, the output of the script has the following information (here it is L28R),





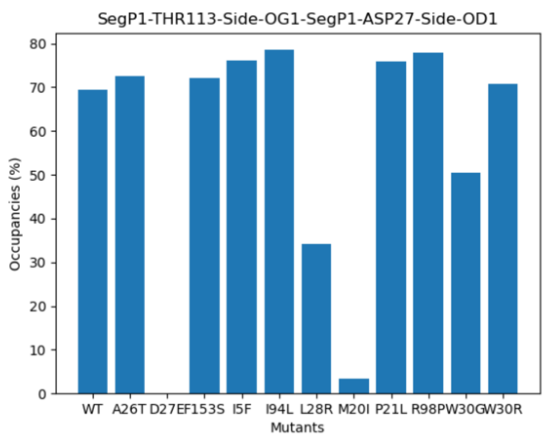


Figure 2. Orange is a common bond seen among different mutants. In blue graphs, L28R shows different behavior than that of the majority of the mutants.

L28R shows here between Threonine 113 side chain OG1 and Aspartate 27 OD2 almost all the time covering the way beneath the active site. (Because OD1 and OD2 belong to the same residue.) We kept the individual bond names here to be able to track down the bonds afterwards.

All the bonds given by the script should be analyzed manually. Then, the necessary curation of bonds should be made according to their relative behavior.

We have separated the bonds into two categories; first set has the common bonds that we see in nearly all the mutations, and the second has the unique bonds.

For seeing these results, plot_unique.py should be used.