AI in Drug Discovery and Development

Prof. Rajnish Kumar
Dept. of Pharmaceutical Engineering and Technology
IIT-(BHU), Varanasi

Week-06
Lecture-30

Welcome to the course "AI in Drug Discovery and Development." In today's session, we will have a hands-on analysis of the MD trajectory using MD analysis. So, for that, we have to follow this link: tinyurl.com/aidd-mda. MDA is for MD analysis. Once you open that link, you will land here; this is a Colab notebook for MD analysis tutorial.

ipynb. So, we have discussed that MD simulation is a computational method that simulates the physical movement of atoms and molecules over time by solving Newton's equations of motion. So, there are multiple applications of MD analysis where we can use it for studying protein folding, ligand binding, molecular interactions, and other biochemical processes at the atomic level. So, the fun part is running the MD simulation that we discussed when we talked about MD simulations.

So, once we do the MD simulation, what we obtain from it is a trajectory, actually an MD trajectory, and this MD trajectory contains, you know, frames, or you can say them as a kind of pictures of the conformation of proteins or any other molecule if you are doing MD simulations on, for example, RNA, DNA, or even small molecules. So, we will get those poses. So, these are poses per unit time, actually. For example, every femtosecond we get some poses. And then it is just like a video, actually a normal video, which is composed of, you know, different images that represent the confirmation of something at that specific point in time.

So, there are multiple ways to analyze, you know, the trajectory. So, some of those you know regularly used methods or those you know metrics which we use to evaluate the trajectory are the RMSD, root mean square deviation, which is used to assess the stability of molecular structure. Root mean square fluctuation, which is used to evaluate the flexibility of different regions of a molecule, Radial distribution function, which is used to study the spatial arrangement of atoms in the system. Principal component analysis, which is a dimensionality reduction technique used to identify the motion modes in a system, simplifies the interpretation of complex data. So, in this specific tutorial in this hands-on session, what we are going to do is use the MD analysis, which is a very powerful toolkit for analyzing the MD analysis data, the trajectory data.

And then we will also use, for example, a Python library called Py3Dmol, which is a

visualization library that we will use to visualize those molecules and the trajectory. So, the first thing I would suggest you do is to copy this Colab notebook to your own Google Drive; for that, you have to click on the file and then save a copy in Drive. And once it is copied, you can just say, for example, when I click on copy in Drive, it will say "you are copying in Drive," and then it will open in another tab, okay? So that you shall do, because then you can access the data, you can, you know, mount your own Google Drive. And then you can access the data from your own Google Drive and save the data in your Google Drive as well. Okay, so I'll just close this down and then go back to this one.

So, once we have done that, once we have copied it. So, the next step is that we will have to install the required tools and dependencies that we will be using to analyze the trajectory. So, for that we use this command called PIP install, MD analysis is one of them. So, for that, you can go to, you know, if you follow this link, you will get all the details, and you can also get a load of, you know, examples and tutorials where you can see what you can do with the MD analysis. So, we install MD analysis, NGL view to visualize the trajectory, and then Py3DMol again for visualization.

NumPy for the calculations, scikit-learn for the analysis, and Matplotlib for plotting the graphs and data, okay. So, for that, you just have to click on this. So, once you click on "Run this cell," it means that when you press the play button, this cell will run, and then it will install all the dependencies. So now you can see it has installed everything that is needed. Okay, so once we have installed everything, the next step is to import the required tools.

So we will be needing all these tools like the MD analysis from, you know, the MD analysis tool, and from MD analysis, we import the RMS for root mean square deviation and fluctuation, and for alignment and for doing PCA, so all these things we will do. To visualize a trajectory in the notebook, we can import the nglview, and we can use multiple tools like Py3Dmol as well. So, the next thing is we will import some other modules for analyzing the data, like pandas, NumPy, and, you know, scikit-learn, okay. And then we will import some floating and system tools, like matplotlib, and we will also import seaborn, which can be used for floating. And then we will import warnings to avoid, you know, ignoring the warnings, and then we will change the working directory to the current directory as well by importing the OS, okay.

So, now we have imported everything that every module will be needing. So, the next thing is getting the data. So, here in this specific analysis, we will be downloading the data from one of my GitHub repositories, where we ran a small MD simulation on this protein coded as 6SAF. So, if you wanted to know more details about this protein, you can just follow this link or you can also go to, you know, my GitHub repo as well, where you will

find the data associated with it. You will also find details on how you can run the MD simulation using GROMACS, as we have provided here in this GitHub repo.

For that, we need to run this cell again, so once we run this cell, you can see here I get the md.gro file. And then we get the md_center.xtc file, so this xtc file is generated in GROMACS, and that is the trajectory file. So then we will need these two files: md.gro and md_center.xtc, the trajectory file. So, once you have downloaded this, the next step is to see here that we have downloaded it. So, the next step is to make sure that the data is there. So, just issue the ls command.

So, you can see there is data since I have run it already. So, you can see all those plots are also there, okay, but anyway. So, you can see the data as md.gro and then md_center.

xtc. So, these files are there. The MD analysis works in a very beautiful way. It specifies all those known atoms and their coordinates for the trajectory as in a universe, actually. And then, from that universe, once you define the universe, you can call all those functions to analyze atoms and then molecules, and do all those sorts of analyses, okay So, then we will define the universe U as equal to mda.

universe md.gro and then md_center.xtc. And then if you print the print U, it means how many you know; it will print the details about the universe. So, it says that the universe has 27,465 atoms. So, we have 27,465 atoms.

So, that includes. So, this is just a protein MD simulation. So, we have that small protein. And then we have, you know, the water molecules, and then we have ions as well in it. And then you can print, you know, use the command print u.

atoms. So, what are those atoms associated with in the universe? So, you can see that atom group atom 1 is of type n of rest name glycine, and then you know you see all those details, actually. And then you can see that the last atoms are the chloros of type Cl of the rest named Cl. And this is coming from the, you know, the ions, actually, okay. And then you can also get some more details about this, you know, like the print atoms, total atoms, print residues, how many residues there are (8623), and then the segment is 1, okay. So, next what we will do is try to import this trajectory data.

And then try to visualize whether it looks fine or not because many times when you run the MD simulation. So, what you will see is that there is this periodic boundary condition which we discussed. So, in this periodic boundary condition, sometimes you see breaks in the trajectory, actually. So, to visualize whether the trajectory is intact and there are no breaks. No bond is, you know, broken, so we just have to, you know, look at it actually,

and to do that, we will use this function here; we will run this cell.

So, you run this cell. So, what this cell is doing is exporting all those, you know, trajectories or the confirmations because it is composed of, now this XTC is composed of coordinates for all these, like, 1000 frames. So, in this case, we have 1,000 frames in this data. So, all those 1000 frames are exported as 1000 PDBs, which we will then open in Py3DMol. And then when we align them on top of each other and render them in the form of an animation, okay. So, what you will see here is that once you run this cell, you will see the trajectory of the molecules you are visualizing.

So, you are visualizing the trajectory. So, you can see how these atoms in this protein are actually moving. Now, you can see a little bit of a jump; actually, those frames are not smooth because they are not aligned on top of each other, okay They were centered when we exported them from GROMACS. So, they are centered, but they are not aligned on top of each other. So, then there is a little bit, you know, you can see kind of a shaky video, right? So, to fix that, what we will do is align the trajectory using the tool or the function in the MD analysis, which is align.

alignTraj. So, u is the universe, and again we are aligning at the top of the universe itself. And then we will be using the C alpha carbon atoms, the carbon alpha atoms for each protein, okay. So, let us run this cell. When we click on this, it will align your trajectory, okay? So, align all those frames on top of one another. So, it will start from, you know, for example, the 0th frame or the first frame, and then it will align the second frame on top of the first frame, and then it will align the third frame on top of the second frame, and likewise, okay.

So what you will see here is that we can just import; we can run this cell as well to avoid, you know, errors in visualizing all those graphs and plots. So, once we have aligned this, what we will do is. We will all import it again, but before that, you know we can specify what protein is since we just have protein in this case in this trajectory. So, we can skip this as well, but if we have, for example, a ligand in this trajectory. So, we need to specify what protein is, what a ligand is, and if there are any others; for example, if multiple proteins are present.

So, we can specify chain A, chain B, chain C, or protein A, protein B, protein C, and we can, you know, analyze those individually as well. And this is, as I said, the beauty of, you know, the MD analysis that you can play with a lot, and then you can generate wonderful analysis using this tool. So, once we have specified that, the next step is to just have a look at the aligned trajectory. So, when you click, when you run this cell, okay. So, what you will see here is that after running this cell, you will see the video or the trajectory like this.

So now, if you compare this animation or this visualization with the earlier one. So, now you can see that there is, you know, no jumping in the frames. And there is very little motion, and that motion is specifically restricted to, you know, the outer or, you know, the terminal part of the protein. Where the C and N terminals are a little bit flexible, they shake a lot, and those loops are also a little bit, you know, shaking, okay? So, this is after the alignment; actually, when we align the trajectories on top of each other. So, once we have aligned, we are okay that our trajectory is fine.

So, there is no problem, no break, and it is properly aligned. So, the next step is to start analyzing, okay. So, the first thing we will do is the protein RMSD (root mean square deviation). So, as we discussed, it is a quantitative measure. So, it is a quantitative measure of the average distance between atoms, typically backbone or C alpha atoms, of protein structures over time.

And it reflects how much a protein conformation changes during a molecular dynamics simulation compared to a reference structure. So, usually we use the initial frame, or frame 1, actually. So, what does it tell us? Why is it important? So, we can monitor the structural stability because we can see if we have it when we start the MD. So, you know the atoms will move a little bit more, and then after some time they will try to settle down.

So then we will actually see a plateau. And this plateau in this RMSD indicates that the protein has reached a stable conformation. It is important to show that the protein is now stable; actually, the structure is stable. And then you can use it to detect the conformational changes; it helps in identifying whether a protein undergoes significant structural shifts, like folding, unfolding, or domain movement. As well as we can assess the equilibrium, it can be used to verify that the system is well equilibrated before analyzing further simulation data. And then we can compare different conditions as well as being useful for comparing structural dynamics across the mutants, like having one wild-type protein and then mutating this protein.

And then we run two MD simulations, wild type versus mutated, and then we can see how that mutation is affecting the stability of the protein, the conformational stability of the protein, or the conformational dynamics of the protein. So, for that, it can be used: liganded versus unliganded, or, you know, different force fields as well, and then it supports drug design. So, in ligand-bound simulations, RMSG reveals if the binding pocket or the ligand remains stable. Okay, so for that, what we are going to do is run this cell. When we run this cell, what it is doing is importing the RMS (root mean square) function from the MD analysis.

So, this RMS function is importing, and it is also importing matplotlib for plotting as well, and then we calculate the RMSD using rms.rmsd. And then here you can see the selected backbone. So, now we are calculating the backbone RMSD, right? And that typically contains nitrogen, C alpha, carbon, oxygen, and everything.

Okay. Except for the side chain of these peptides or the amino acids in a protein, everything is counted as a backbone. Alternatively, what we can do is calculate the RMSD using the C alpha as well by using the command name C alpha instead of backbone or protein, the whole protein including the side chains. Or we can also do the specific residues, like in this case; for example, we can do this little bit here. So, we can calculate the RMSD from residue 50 to 100 as well, okay. So, once we say that after we do the transpose, and since the time we run these calculations, the time recorded was in picoseconds.

So, to convert it into nanoseconds, we divide it by 1000. And then the time is now because you know nanoseconds, and then the values are the RMSD, okay. So, then we use the plotting library Matplotlib, and then we plot it. So, now we can see that we have a nice plot showing the time on the x-axis. In nanoseconds and then the RMSD in angstroms on the y-axis, right? So, now you can see here, for example, when it started at time 0, okay.

So, the RMSD of the whole protein is actually okay; it is frame-wise, frame-wise whole protein, right? So, now the RMSD jumped up a little bit, then it went to one long storm, and then it was fluctuating a little bit, and then slowly it was coming to this plateau, actually. Since this is just one nanosecond of data, if we had run it for maybe 100 nanoseconds, we could have seen a very stable RMSD plot. So, if it is showing a lot of fluctuation, it means that your protein, you know, the structure is fluctuating a lot. Means either there are multiple domains that are acquiring multiple confirmation orientations during the simulation. Or there is some kind of mechanism where the proteins are like opening and closing, or there are a lot of conformational changes.

So, then within the last command, you know, fig.savefig('rmsd_background.tif'). So, we have now saved this figure as a TIFF file as well. Okay, now we have done the RMSD. So, the next major analysis is the protein RMSF calculation, root mean square fluctuation, which quantifies the average positional deviation of each atom or residue from its mean position during a molecular dynamic simulation.

So, it provides a residue-level insight into how flexible different parts of a protein are over time. So, compared to the RMSD in RMSF, what we get is residue-level fluctuation, you know? Root mean square fluctuation of the individual residues, and in this case, we can typically see how those specific domains or parts of domains are flexible in nature, and then they are moving. So, the importance of RMSF is that it highlights the flexible and

rigid regions. So, it helps identify loop regions, terminal ends, or domains with high mobility versus the stable core region.

And then we can assess the binding effects. We can reveal how ligand binding or mutations affect protein flexibility at specific sites. And it also supports the functional insights where flexible regions may correspond to active sites, allosteric sites, or the interaction interfaces. And then we can validate the simulations. It helps compare the simulated flexibility pattern with the experimental B factors from X-ray crystallography. And then, of course, it enables drug design by detecting flexible pockets or hotspots for ligand binding.

So for that, again, what we will do is, in this case, we will use the C alphas, okay. So, you selected the protein atoms and named them C alpha, and then we used this to create the RMSF, and then we will plot this again using matplotlib. So, now you can see this. This is a C alpha RMSF for the protein, okay. And then you can see the RMSF here, and then you see the residue number.

And you can see the terminals, like the C and N terminals; they are fluctuating a lot. And that is, you know, usual as well because those are actually at the end, and they are free to move. So, they are not, you know, making any bonds or bonding with any of the surrounding, you know, atmospheric molecules. So, they are free to move, and they show a lot of, you know, RMSF. And then we can show this RMSF value as a custom B factor and display it on the 3D structure as well where we can show which part of the protein is more flexible and which part of the protein is not.

So, it is something like we just take this RMSF data of the individual amino acids or, you know, the individual atoms, right? And then we just put it on top of one of the structures and show a color map, actually. Let us run this cell when we run this cell. So, what we are doing here is I can just quickly explain. So, we are importing all those tools.

However, we have already imported it, but we can do it again as well. And then in this step, we are creating a data frame with the RMSF value. Okay, RMSF data is put into a data frame, r.rmsf, and then in step two, we are saving the protein structure with RMSF as the B factor. And then we are mapping the RMSF to B factor for all atoms using the residue IDs, and then we are saving the new PDB with custom B factors.

And then we are visualizing that using Py3Dmol. Okay, so now when we run this, you see here we have got the protein and then RMSF mapped on top of the protein, right? So, now we can see, for example, if you look at this scale as well. You can see the range from 0.4 to 1.8. Now, you can see that the red region, which is the core region of these beta sheets, is structurally very stable.

So, it is not moving much while the terminal part, for example, you can see this blue here, okay, and then a little bit of green in those loops as well, and then in this alpha helix, the terminal alpha helix as well. So, you can see some green and yellow, which shows that it is a bit flexible. So, this is how we can just, you know, overlay the RMSF as the B factor and then visualize it. So, you can generate a nice visualization, and then it is more informative as well. Okay, the next thing is the next analysis we will be doing is the radius of gyration.

So, the radius of gyration is a measure of the compactness and overall size of a protein structure. So, it quantifies the distribution of atomic positions around the center of mass of the protein. So, in simple terms, we can say that it is like how spread out or compact a protein structure is at any given time. So, it is often calculated using the atoms of the protein, usually C alpha atoms, and is used to assess changes in the protein's conformations during your MD simulation.

So, what kind of importance does ROG have? So, we can monitor protein folding. So, as a protein folds, its radius of gyration typically decreases as the structure becomes more compact. A larger radius of gyration usually indicates a more extended or unfolded state. And then we can identify structural transitions as well; sudden changes in RoG can indicate major conformational changes in the protein, such as folding, unfolding, or domain movement. Or we can check, you know, the protein stability as well, where a plateau in RoG over time signifies that the protein has reached a stable state and is no longer undergoing significant conformational changes. And then we can assess the solvent accessibility, as larger values of RoG can indicate more exposed solvent-accessible areas in the protein structure.

And then we can evaluate the different states of proteins, like RoG, which is useful in comparing different protein states, such as folded versus unfolded or bound versus unbound. So, when we run this cell, let us run this cell. So, what we do here is select the C alpha atoms, and then we calculate the radius of gyration per frame and get the time. So, now we have the time in, you know, picoseconds. So, we divide it by 1000 to convert it to nanoseconds, then we convert it into arrays, and then we plot this using matplotlib, and here we get the radius of gyration.

So now you can see. So, this is how we get upload versus time versus, you know, the radius of variation in angstroms, and you can see that because our protein is very small and compact. So, there is not much change; you can see there is not much change in the ROG. However, if we would like a large protein with multiple domains. So, we could have seen that you know how to fold, unfold, or bring together two different domains in a protein

during the MD simulation. So, all that kind of information we could have gotten if we had that kind of protein, okay.

So, based on what kind of structure you know we are simulating. So, we need to use all those methods to analyze the trajectory. Okay, and then we can calculate the pairwise RMSD as well. So, that is, you know, calculating the root mean square deviation of each frame with each other frame. So, we have 1000 frames, right? And then we can calculate the RMSD of those 1000 frames with these 1000 frames, okay? So, it is kind of a heat map, actually; I will just show you. So, if you click on this, what it will do is compute the distance matrix using the alpha carbon (CA) and then we will plot the distance matrix as well.

So, what you do is once you run it, it will compute the distance matrix using the alpha carbons, the CA matrix, and then we plot the distance matrix. So, now we get this kind of diffusion map distance matrix, okay. So, we get this diffusion map distance matrix where we have a heat map bar or the label which indicates, you know, the purple color as 0 and then the yellow color as 1.6. And then the representing the RMSD, and we have the frames from 1 to 1000, actually, and 1 to 2000, okay.

So, now you can see, like from frame zero, okay, and then frame zero, there will always be a value of zero, so you will have this diagonal, which represents the value zero, right? And then frame one with zero, frame two with zero, and so on, frame thousand with zero. Okay, and comparing that, you can see that the terminal part, okay, so it is always a little bit, you know, flexible, and then showing higher RMSD. So, if you have some patches here, like yellow patches in between, as well. So, that represents, you know, the domain motion, actually.

So, because this is a very small protein. So, we did not see that in this case, but if you have a larger protein with multiple domains. So, you can see all those kinds of phenomena in those structures. Okay, another important analysis is the principal component analysis, and we will talk a lot about this. Because you know that whenever we have, for example, a large number of variables, which of those variables are really important? So, that is a kind of question that we have discussed earlier, like how PCA is used for dimensionality reduction, if we want to select the best parameters out of those thousands or ten thousand parameters.

So, we can use principal component analysis. Okay, so in MD simulation data, we have, for example, those thousand frames. So, those thousand frames are representing those protein molecules in the energy landscape, actually. But out of those, which are the important components or important frames representing the variance in this data? So, for that, we will use the PCA. So, what we will do here is first import all those, you know,

those modules that will be needed, and then we will do the PCA analysis on the backbone carbon, okay? So first we will select the backbone carbon atoms. Okay, now we have selected the backbone, and then we will do the PC analysis on the backbone carbon.

And then we compute the variance and the cumulative variance. So, we will calculate the variance for the first principal component, and then we will print the cumulative variance for the first PC as well as the fifth PC. And then we can plot the cumulative variance, okay. And then we can transform this data using the first five principal components and create a DataFrame for the first five PCs. And add the time data, and we can also, you know, print the number of backbone atoms and principal components. So, here you can see that the first principal component is able to cover 45 percent of the variance in the data.

And the fifth one shows a variance of 0.19 percent of the data. So, the first PC can explain 45 percent of the variance in the data, okay. And then this is, you know, a cumulative variance that is explained by the principal component. You can see that we have the PC here from 0 to 50, and then you can see that we have the cumulative variance from 0 to, you know, 1. So, we can say, for example, that around 50 percent of the variance can be explained by the first 5 principal components, okay. And these principal components are usually orthogonal, which actually means that PC 1 and PC 2 do not share the same data.

So, they mean this will show the variance in this direction, and then PC 2 will show the variance in the orthogonal direction, okay. So, it means they do not have overlapping information. Okay, and then what we can do is extend this data, and then we can put a 2D map as well as a 3D map as well. So let us run this cell as well. Here we are, you know, plotting principal components 1 and 3 from this data, so you will get something like this.

where you can see how the data is spread among PC1 and then PC3, and you can see from this data that you can also get the 3D density plot map of PC data. So, ah, you can see that here we have PC 1 and then PC 3, and then we have the density as well. This is the end; actually, by doing the PCA analysis, we can analyze the variance and identify the variables or features that are responsible for explaining the maximum variance. The minimum number of features that can explain the maximum variance can be identified from the principal component analysis.

This was, you know, kind of a quick introduction to what we can do with MD analysis. So, what we have covered is that we have covered all those basic analyses, and now you can also do the remaining analysis. For example, you can do the hydrogen bond analysis, the radial distribution function, and if you go to the MD analysis website, you will find a lot of tutorials where you can use those tutorials to learn it in detail as well. MD analysis is not only one tool that is being used; there are other tools as well. For example, PyTraj is

also available, which is mainly developed by the Amber team.

Then you have the MDTraj, and in VMD, you can also do a lot of analysis. All these tools are extensively used for analyzing molecular dynamics trajectories. And with that, what you can do is just play with this Colab notebook, analyze your data, and write more analysis and code into it to perform specific analyses as well. And with that, thank you.