AI in Drug Discovery and Development
Prof. Rajnish Kumar
Dept. of Pharmaceutical Engineering and Technology
IIT-(BHU), Varanasi
Week-12
Lecture-56

Welcome to the course "AI in Drug Discovery and Development." So, in this session, we will have a hands-on demonstration of molecular structure representation. So, you know molecular structure representation is one of the very important aspects of drug discovery and development. Because most of the time those small molecules, or even those biologic molecules, are what we wanted to use for any sort of research project or analysis. So, we need to represent them in some structures that can be understood by the computers, right? So, in this session, we will see how we can use RDKit to represent those molecular structures and just play with them and analyze their structures. So, all sort of things we will do.

So, for that, we have to follow this link. We will copy this link: tinyurl.com/AIDD-rdkit. And then we will open it in any browser, whether it is, you know, Chrome, Internet Explorer, or Safari.

So, once you open it in a browser, you will see something like this. So, we will end up with the Colab notebook named "molecular structure representation.ipynb." Okay, in this typical Colab notebook, what we will do is see how we can use RDKit in Py3Dmol to visualize the structures and calculate some properties. And to do some analysis that will be useful for your research or for your learning, the best way is to just make a copy of it.

For that, you go to File. Ok, and then click on "Save a copy in Drive," and once you click on "Save a copy in Drive," it will show you creating a copy. And then once the notebook copies, it will show you a pop-up saying "Notebook copy complete," and then you can open it in a new tab. So you just click on it, and then we open it in a new tab. So what this action will do is make a local copy of this Colab notebook into your Google Drive because Whatever you will get when you enter that link, aidd-rdkit, is the Colab notebook that you will see hosted in my Google Drive.

So it's better to keep a copy in your files so that you can edit it, and then you can use it later as well. So, once you have copied it, you will see a copy of the 12.1 molecular structure representation. So, this notebook provides a short overview of different molecular structure representations, and it is inspired by several, you know, resources; the main one is the RDKIT cookbook. Where you will see a lot of code on how to do various sorts of

analysis using RDKit.

And then this Iwatobipen is again a wonderful blog where you see a lot of applications of cheminformatics. This guy has shown different sorts of, you know, tools and techniques, mostly using RDKit to do the analysis. And then GitHub Gist, so this is also one of the sources where you can find many details, tutorials, and more. And then the Pat Walters Practical Cheminformatics blog on Blogspot, which is also one of the sources where you can find all these tools and techniques. So the first thing we need to do is install RDKit, which is an open-source cheminformatics tool, and PY3DMol, which is a molecular visualizer.

So, we will run this cell. Once we run this cell, what it will do is install the RDKit, and then if we run the next cell as well, it will install the PY3DMol. Let us first install the RDKit, okay? Yeah, so it will probably not take much time, maybe like a couple of minutes. You can see here now that it has successfully installed the RDKit. And then similarly, you install the PY3DMol by running this cell as well, so now it has installed the PY3DMol.

Okay, and then we need to import all those, you know, modules for running all these, you know, analyses, like we will import PY3DMol and we will import RDKit. And from rdkit, we import the Chem function; from rdkit.Chem, we import the AllChem; we import the draw; import the IPython console in data structs as well; we import the, you know, the mol drawing and mol drawing options as well, okay. So, once we have imported all these, the next step is that we can work with the SMILES. So, the first step I will show you is how to work with the SMILES.

So, when we have a SMILES string. So, we can use more from Smile to render the molecule and keep in mind that these files do not contain the atomic coordinates and thus RDKit generates them while drawing the molecules. We talked about the SMILES earlier. So, the SMILES are the simplest molecular representation you know. However, it is very powerful.

So, it is a 1D molecular representation; it does not contain the 2D structure information or even the 3D structure information. However, when we enter those SMILES into the RDKIT Chem.MolFromSmiles function. So, what it does is it shows you the 2D structure, and then we will see how we can calculate the 3D structure as well. So, it calculates the 2D structure, you know.

So, here we have defined Donz, D O N Z, which is, of course, a short name for Donepezil. So, we have defined a variable called 'ok', where we call the function Chem.MolFromSmiles, and then we add the SMILES of this, you know, molecule. And

this SMILES you can get from, you know, different databases; like you can get it from PubChem as well. Or you can, you know, get it by drawing the structure; you can also draw the structure and convert it into SMILES.

And then you can use that as well, so let us run this one. When you run it, you can see that it is converting the SMILES into the structure; it visualizes the structure. And SMILES is a very simple, you know, representation. So, maybe just for a quick view, if I replace it with, you know, just CCCCC. Let us see what it will be; we will not change the name of the variable.

So we will just see what it does. So, now you know I have drawn the structure of pentane. It is an aliphatic hydrocarbon with just 5 carbon atoms. And then, we know that the aliphatic carbons are represented by a capital C and the aromatic carbons are represented by a small c. So, if you wanted to draw, you know, an aromatic ring, what we can do is, for example, c1 c c c c and then c1. So, what the 1 is representing here is that it represents the connection; like 1 is connected to this carbon, which is connected to this carbon. So, C1 means connect 1, which is a connection, and then between this C and this C, okay? And if we run this now, you will see that we have it. We have this one, actually. We can see that we get the benzene ring here represented correctly. So, going back, we just replace it with the donepezil, you know, SMILES.

So, we get the donepezil structure. So, now we can see we have got the donepezil structure. So, the next thing is that we can actually add the hydrogens, and we can explicitly add the hydrogens, and we can visualize them. So, the donzH is equal to Chem.

AddHs and then donz. So, don't we have defined another variable donzH which contains the added hydrogens? So, now, you can see that we have added the hydrogens, and those are visualized as well. Right? And then we can calculate and display the Gastiger charges as well. So, these Gastiger charges are very important because whenever we are preparing those molecules, we have seen in our theory sessions as well that we are preparing the molecules. So, accurate charge calculation is very important because that charge is essential whenever we are using those structures, for example, in molecular docking or for any other kind of interaction studies. So, those charge they play a very important role in calculation of those docking scores.

And if those charges are not correctly calculated, then those docking scores and further analysis will not be reliable. So, we calculate those charges. So, we click on this cell, and then what it is doing is calling the function AllChem.ComputeGasteigerCharges, and we are applying it on DonzH where we had added the hydrogens. And then Donz H charges are equal to Chem.

Mole(DonzH) and for each atom in DonzH_charges.GetAtoms(). So, we just apply this and then it you know it display the charges here as well. So, you can see here that it calculates the charges and displays those charges. Okay, and then what we can do is render it into 3D as well.

So, if we run this cell, what it will do is embed the Donz H into a 3D representation using PY3DMol. So, here we are using PY3DMol to visualize this molecule in a 3D representation. So, you can see here. This is our now 3D structure representation of the, you know, donepezil molecule. Okay, and then one more thing: here it will calculate, you know, the 3D conformation; then only it will represent the structure.

Once we have shown this in 3D, okay. So, the next thing we wanted to do was optimize the geometry because usually, whenever we convert a molecule into a 3D conformation. So, it might have multiple possibilities, and out of those, we wanted to see which one is the least energy conformation, and for that, we use this geometry optimization. So, in this cell, what we are doing is optimizing the geometry of DonzH, okay. So, we call the function called AllChem.MMFF (molecular mechanics force field) OptimizedMolecule(DonzH), and then we draw that molecule in 3D using PY3DMol. So, when we run it So, now you can see that it has optimized the geometry and of the molecules. So, now this is the optimized 3D geometry molecule that we can use further for doing a lot of analysis. We can use it for QSAR studies or for molecular docking studies, etc.

So, we can use it. So, that was about the structure preparation, how we can just call, you know, for example, the SMILES, and then we can prepare it and visualize it. So, now one more thing you can do with RDKIT is search for the substructures, actually. So, in this case, for example, we have this donepezil molecule, and I want to know if this molecule has a benzene ring. So, in that case I will use this function called as donz.

GetSubstructureMatches(Chem.MolFromSmarts('c1cccc1'). So, here we use a SMARTS pattern. This is a SMARTS pattern for, you know, a benzene ring. When we enter this SMARTS pattern here, we use this function. So, what it will do is highlight the part of the molecule that contains this substructure.

In this case, we were using the benzene ring. You can search for any other pattern as well, like whether you want to know if a molecule has a keto group. You know, a methoxy group or molecule has a cyclopentane structure, whatever kind you wanted to know. So, you can just use that SMART pattern here, and then you can search for it, and your molecules will get highlighted. So, this is another important use of, you know, RDKit, by using this function.

So, after that, what we will see now is how we can optimize the geometry and search for the substructures of a single molecule. So, now let us import some more molecules instead of working on one molecule; let us import an SDF file and then work on that. So, for that, you will download an SDF file from my GitHub repo, MLBase QSAR example underscore compounds dot SDF. So, for that, we use the wget command when you run this command.

So, it will download the files. So, the example compound dot SDF and 100 percent are downloaded, and then we have that, you know, here in our directory, and we can confirm this by using the ls command as well. So, if we issue the ls command here, it will show you that we have examples_underscore_compounds.sdf. So, now, once we have this file, what we will do with it is that we have to load it using the function called SDMolSupplier. So, we import the module called chem, and then from chem, we import the SDMolSupplier, and then we load the SDF file into this variable: supplier = SDMolSupplier('example_compounds.sdf.1').

Okay, here this is not dot one, but this is just, you know, SDF underscore SDF, okay. So, load the SDF file, supply the SDF SDmolSupplier example compound dot SDF, and then we extract the valid molecules in mols per mole in the supplier if mole is not none, and then we print how many valid molecules we have got, okay. So, let us run this cell. Once we run this cell, you will see that we have loaded 50 valid molecules. So, it means it contains the 50 molecules that we have now loaded into the mols, okay.

And then we can draw them by using these moles to grid, actually. So, there is a function in RDKit where you can visualize the structures of the molecules we have just imported. So, you run this cell, and here, just to keep it simple, we just wanted to see only 12 molecules. So, n_show = 12, and then subset_mols = mols[:n_show] And then we create an image where we use Draw.MolsToGridImage, with 4 mols per row and an image size of 200 by 200, and then we can add the legend, like the mol number as well, okay.

So, when we run this cell, what you will see is that it will draw the molecules mol 1, 2, 3, and 4. Up to 12, because we asked it to draw the 12, and then 4 molecules per row, okay? Per row, 4 molecules it is drawing. So, likewise we can we can just have a look at the structures by using this function ok. And then we can also you know create an interactive visualizer.

So, if we run this cell, okay. So, here we have got a function to display a selected molecule. And then we are looping this selective function to visualize all those molecules, okay. Here we can see that this is an interactive visualizer that shows the name of the molecule and the activity as well because this file contains the name of the molecule and the activity data. So, if it is here, it is N A. Then, with this scroll bar, you can scroll to different molecules.

And now you can see that we have got all of them in. So, the purpose of this visualization is to ensure that all your molecules are correctly represented. There is no problem with those molecules, either with the bond odors, hydrogens, salts, or other things, because that is very important, and if you remember, we talked about it whenever preparing the molecules. So, we talked about that it is very important to prepare the molecules carefully, especially whenever you are using them for the QSAR analysis, okay. After visualizing them, what you can do is generate the fingerprints as well, and then there are multiple kinds of fingerprints. And we talked about them whenever we were talking about these, you know, features.

So, molecules can be converted into fingerprints, and those fingerprints are like human fingerprints. So, they are unique to each molecule, okay. So, each molecule has some unique, you know, fingerprints that represent its structure, right? So, in this case, we will calculate the Morgan fingerprint. So, when we run this cell, what it will do is import the required modules, and then it will create a function called fpg, where the rdkitFingerprintGenerator function will create a variable fpg. and which will use the RDKit fingerprint rdFingerprintGenerator.GetMorganGenerator function with a radius of 2 and an fp size of 1024 bits.

We will generate the fingerprints for each molecule and then convert them into a binary matrix. And then we can convert them into a list of numpy arrays as well because that will be easy for the further tools to analyze. So, once we have calculated the fingerprints. So, these are just, you know, bits like 010101 representing the structure of a molecule and the length of 1024 bits, okay.

So, now that we have calculated the fingerprint, we can do a lot of analysis on those fingerprints; for example, we can identify which of those molecules are similar to each other. If you wanted to see if, out of those 50 molecules, some of them are similar to each other. So, for that, we can use a technique called the TSNE technique. This TSNE can be used to reduce the dimensionality of, you know, molecules.

So, of the features as well. Here, we will use the TSNE. It's okay when we run this cell. So, what it will do is show you the scatter plot where it is like PCA, actually principal component analysis, where you just plot those components; like in PCA, we plot the principal components 1 and 2. And then in TSNA, we plot TSNA 1 and TSNA 2 here. So, now what you see here is that these molecules, for example, the bunch of molecules that are closer to each other, are very similar in structure, okay. These molecules are very similar in structure, but the molecules that are spreading around are not similar to each other.

So, that is why they are, you know, spread around the area. So, by quickly looking at this analysis, we can see that our set of molecules contains 50 different molecules. It has at least two, you know, groups that contain similar scaffolds and structures with similar scaffolds, okay. And this is one of the techniques by which you can analyze the diversity of the molecule or the structural diversity. So, we can analyze the structure; we can analyze how diverse the structures are from each other. And not only that, this was how we could load the SDF file and then import any sort of data into RDKit.

So, you can even play with the CSV files as well, like if you have a CSV file. So, how to use that? Let us see that as well. So, in this case, we will again use wget to download a CSV file named new_one.csv. Let us run this cell. So, once you run these cells, it will download the new one dot csv, okay, and then here we will use pandas to, you know, load the csv file into a data frame. So, here we will use pandas to load the CSV file into a data frame named df = pd.read_csv("new_1.csv") and then we will display the first few rows of the data frame.

So, let us just run this cell. And then you can see that we have a CSV file which contains the index, the AMBIT InChI key, another kind of structural representation, and then score, activity, and SMILES. So, this is just a random file that contains some molecules with their SMILES represented as the SMILES and their activity data indicating whether those molecules are active or inactive. And the score where active shows 1 and inactive shows 0, and then the Ambit_InChi keys as well, okay. So, now we can convert these SMILES again into, you know, we can use the mols to grid image to visualize them in the 2D structure. So, we run this one, convert the SMILES column into the RDKit molecule object; first we convert them into RDKit molecule objects and then we display them as a grid.

So, you can see here, for example, that these are the molecular components of this SMILES file. We can play with the CSV data as well, and then we can always use all these, you know, in Colab. We can always use all these Unix commands, like the ls command, which will tell you what kind of files you have in the directory and in which directory you are right now. OK, so that was how we can import these files.

And now let us calculate some of the descriptors. In the earlier cells, we calculated the fingerprints to determine their similarity. Now let us calculate some additional descriptors. And for that, we usually use a module called descriptors in Rdkit. OK, when we run this cell, what it will do is import the module called descriptors and then load the CSV file into a data frame. And then we, so this we have already done, but if we are doing it independently, we can do it again.

So, we convert them into the RDKit molecule object, and then we calculate the descriptors,

and for that, we are calculating the molecular weight. Okay, by using the function called Descriptor: molecular weight, log P, TPSA, number of hydrogen bond donors, and number of hydrogen bond acceptors. So, we are going to calculate these five descriptors. So, we have created this list of descriptors, and then we calculate for descriptor underscore name descriptor function in descriptors dot item.

The mole that is the molecular object in our data frame is okay. And then we just display the updated data frame after calculating it with the df.head(), okay? So, here you can see that "head" means it will just show you the top 5 rows of the dataset of your file, okay. So, you can see here, for example, we have now calculated our molecular weight, log P, TPSA, number of hydrogen bond donors, and number of hydrogen bond acceptors. And then instead of using the DF head, we can also, you know, use just the DF.

So, that will display, you know, all the molecules. So, here you can see if it is not a lot; it will display all the molecules. So, you can see that the activities, smiles, mol, rdkit, mole object, and then molecular weight, log P, TPSA, and then we have how many molecules here, yeah, the number of hydrogen bond donors and the number of hydrogen bond acceptors. So, we have these five descriptors added to this data frame, where we have the molecular structure and we have added the descriptors.

Okay. So, this we will use when we run a colab for doing QSAR analysis as well. And not only that, we can always visualize those, you know, descriptors into some graphs as well. So, here, for example, I have shown what happens if we run this cell. So, what you will see here is that it is using those columns from your data frame file. Then, it is plotting them into these nice bar plots. So, you can see the molecular weight frequency, log P frequency, TPSA frequency, number of hydrogen bond donors frequency, and number of hydrogen bond acceptors frequency.

So, you can see how many times, or for example, here at least 24 compounds have no hydrogen bonds, actually. So, then you have all those sorts of analyses you can do by using, you know, these pandas, matplotlib, and RDKit, okay. And then we can do the clustering as well based on these descriptors; we can also cluster them by running this cell. So, what you will see here is that we are clustering, for example, based on molecular weight and log P. If we just look at molecular weight and log P, we can see some relevance in whether there are molecules with low molecular weight and low log P or high molecular weight and high log P.

So, all those kinds of, you know, analyses we can also do. Okay, and then we will quickly calculate the similarity matrix. Similarity is another important technique or method that is used to evaluate whether you have molecules similar to each other or not. So in this case,

for example, when we run this cell, what it will do is create a heat map of Tanimoto similarity, which is based on fingerprints, so first we calculate the fingerprints. Then we apply this Tanimoto similarity coefficient formula to calculate the similarity between those two molecules based on the fingerprint. If this code is 1, it means the compounds are completely similar, and if it is 0, then they are completely dissimilar, okay.

So, here you can see that by using Morgan fingerprints, we could plot the Tanimoto similarity coefficient of these 48 compounds, actually. So, from 0 to 48, how is compound 0 similar to compound 0, compound 1, compound 2, compound 3, compound 4, and so on? So, here the yellow color shows a very high similarity, meaning 1, and this diagonal line that you are seeing is comparing the same molecules. So, you always see here that the yellow colour means a value of 1. So, that was about, you know, those small molecules, and then you can also read the mole files as well; like this is a mole file for phenol, okay. So, these mole files are in a table format where you have the coordinates for the atoms and the connecting table as well, indicating which atom is connected to which atom.

And then it ends with them, and actually, okay. So, this is what you can do: you can also read files from the mole files as well. So, here you can see, for example, this is phenol; we have just read it from the mole file, and then we can render it in 3D like we did earlier for donepezil. So, you can see here the phenol molecule, but now it does not contain the explicit hydrogens. We can call the add hydrogen function, and we can add the hydrogens.

So, now you can see it has added the hydrogens. However, you can see that the added hydrogens are not actually placed correctly. So, what we have to do is energy minimize it and run the MMFF optimization. Only then will those hydrogen atoms be placed correctly, okay? So, in the next cell, we optimize the geometry using the MMFF optimization, and then you can see we get a correct structure for the phenol. And then, yes, not only that, we can also use PY3DMol for visualizing the protein structures. Like we did for, you know, molecular docking when we were talking about gnina in a tutorial, we can also use it for, you know, visualizing the proteins and all those biomolecules like RNA, DNA, etc. as well.

So, for that, we just import all those modules, and then after that, we can, you know, select the residues for visualization. We can define whether we want them in the stick rendering, cartoon, or ribbon styles, and then we can visualize them. So, here, for example, this is the structure of a protein. Where the donepezil is co-crystallized with the enzyme called acetylcholinesterase, and in the green, you can see the donepezil, and then the rainbow ribbon, you know, is for the protein acetylcholinesterase. And then you have the interacting amino acid residues, which are also rendered in stick form in this orange color, or the bronze color, okay.

So, you can always play with all these commands to, you know, make them as you want.

That is what we discussed today. So, in this session, we talked about all those structural representation methods by using RDKit, Py3DMol, and plotting, analyzing them, plotting the graphs, and all those things. So, all of these are very useful whenever you are working with cheminformatics or drug discovery. And with that, I would like to say thank you.