Created by Yi Jie (Josh) Tseng --- 2/24/2022

Modified by Yi Jie (Josh) Tseng --- 11/13/2023

Software requirement: PyMOL, Python3, autodock\_vina openbabel, and mgtools.

Description:

This README is for **autodock.py** script, which is used to define search space, prepare receptor and ligands for input (for single or a library) and run autodock\_vina. Before running autodock\_vina, few things are required:

1. A search box and a center position

2. PDBQT file of receptor

3. PDBQT file of ligand(s)

For more information on how to run **autodock.py**, type:

**python3 autodock.py -h**

The instructions below will go over how to prepare for these requirements.

Instructions:

1. Set up working directory:

This script can create folders to organize inputs, outputs and logs. To do this, in a terminal, type python3 autodock.py -s <XXX> True -f <No.>. -s flag is to set up folders. The first argument is the folder name, the "True" in the second argument is to turn on or off the numbering mechanism. -f option is to specify a certain working number. Please see below examples:

* 1. Specify folder name + numbering mechanism on. If this command is executed multiple times, it will generate folders with specified name and ordered two-digit numbers. But the numbering mechanism can be overwritten by the -f flag (see 4th example).

Ex:

1st: python3 autodock.py -s test True -> test01/

2nd: python3 autodock.py -s test True -> test02/

3rd: python3 autodock.py -s test True -> test03/

4th: python3 autodock.py -s test True -f 8 -> test08/

* 1. Specify folder name + numbering mechanism off. -f flag is recommended. If the command is executed multiple times without -f, it could potentially overwrite a previous work.

Ex:

1st: python3 autodock.py -s test -> test/

2nd: python3 autodock.py -s test -> test/ (Overwritting the first run)

3rd: python3 autodock.py -s test -f 1 -> test01

* 1. If no name is specified, by default, the folder name is "Dock" and the numbering mechanism is automatically on. Again, the numbering mechanism can be overwritten by -f flag.

Ex:

1st: python3 autodock.py -s -> Dock01/

2nd: python3 autodock.py -s -> Dock02/

3rd: python3 autodock.py -s -f 5 -> Dock05/

1. Identifying seach box and center position:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* IMPORTANT \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

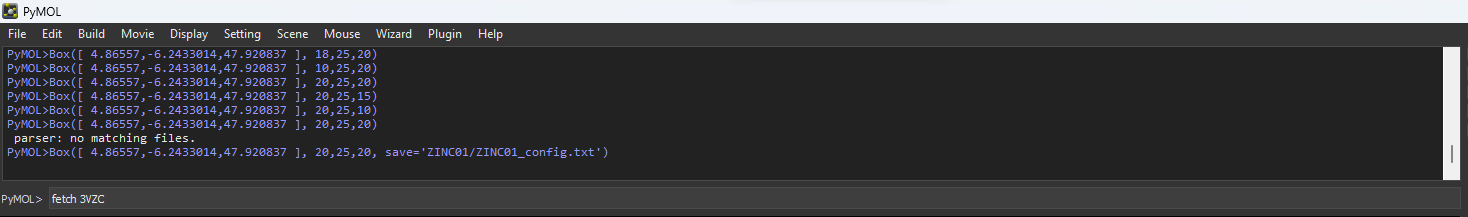
PyMOL is required to visualize the search box and center position in this script.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

* 1. Load structures, if a PDB ID is known, type:

**fetch XXXX**

XXXX = the 4-digit ID of a crystal structure.



* 1. To select residues, type:

**show sticks**

This will show the sidechains of the residues.



Then start selecting by clicking on the residues. If you want to select a lot of residues in an area, you can hold **Shift and drag** to draw a box to select residues. To unselect all residues, just simply click on the empty space. If you just want to unselect one residue, simply click on the residue to again.

At the bottom right corner, PyMOL is by default in Residue mode, meaning it will only select Residues. By clicking on the word “Residues”, you can change to a different selection mode. To select a single atom, click the word “Residues” till you see the word “Atoms”. Then click on the atom you wish to select on the structure.

A screenshot of a computer program

Description automatically generated

* 1. Calculating the search box:

If the binding site of the target is known, select a set of binding site residues. This script can calculate the center coordinaces of the selected residues based on the average of the C-alpha (CA) atom coordinances of the selected residues. If only a single residue is selected, for example, a bound ligand. By selecting the ligand molecule, it can calculate the cooridinace of the center position of the molecule based on the average of the cooridinances of all atoms. If one single atom is selected, it will use the coordinance of the selected atom as the center position.

To do the above steps, using PyMOL to navigate to the working directory where the **autodock.py** is in, run this script in PyMOL by typing:

**run autodock.py**

and

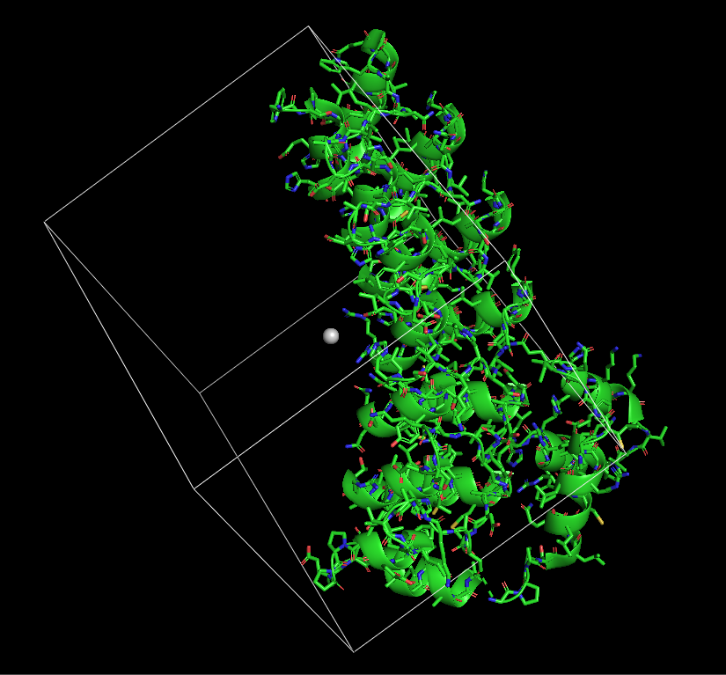
**Box('sele', X\_size, Y\_size, Z\_size)**

'sele' is the seleted residues, usually you don’t need to change it, when you select anything in PyMOL, the default name of your selection is ‘sele’. This 'sele' can be replaced by a list of known coordinance like [X, Y, Z]. X\_size, Y\_size and Z\_size is the dimention of the box. Once the center position and the dimention of the search box is optimal, save the configuration file running the same function by typing:

**Box('sele', X\_size, Y\_size, Z\_size, save='WORKING\_FOLDER/config.txt')**

The name of the configuration can be arbitrary. But it is recommended to include the name of your working folder. For example: ‘Dock01/config.txt’

The search box will look something like below figure, where the white sphere is the center of the search box.



You can also fine tune the location of the center by manually moving the white sphere. To do that click the bottom right panel shown in the figure below. Then hold **Shift and click on the white sphere using the wheel of the mouse and hold it to move**.

A screenshot of a computer program

Description automatically generated

Click here.

One the sphere is moved to your desire location, click on the same location of the bottom right panel, then click on the sphere, and in the command line type:

**Box('sele', X\_size, Y\_size, Z\_size)**

It will then draw the box accordingly.

Additionally info:

If you have a list of cooridiancesm you can also replace the ‘sele’ with the list and it will draw the box based on the cooridiances you provided:

**Box([4.6772785, -7.5425525, 48.58454],** **X\_size, Y\_size, Z\_size)**

Useful command in PyMOL:

1. show stick -> This will show all side chains in sticks.
2. print cmd.get\_coords('sele') -> To get the xyz cooridinances for the selected atom.
3. Preparation of receoptor and ligand PDBQT files:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* IMPORTANT \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Preparations of the receptors and ligands should be done on the supercomputer!!!!!!!!!!!!!!

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Before preparing the ligands make sure **mgtools** is installed. If not, download mgtools from <https://ccsb.scripps.edu/mgltools/downloads/> . Untart mgltools\_x86\_64Linux2\_1.5.7p1.tar.gz and refer to the README in the mgltools\_x86\_64Linux2\_1.5.7p1 folder for installation.

* 1. Preparation of single receptor or ligand:

To prepare the required PDBQT files for receptor and ligands (XXXX.pdbqt), through passing -p flag, this script runs scripts named **prepare\_ligand4.py and prepare\_receptor4.py** from **mgtools**, which is a software for molecular structure analysis and can convert PDB files to PDBQT files and assign partial charges to all atoms. To do this, run the script in the terminal by typing:

**python3 autodock.py -p -r path/to/receptor.pdb -l path/to/ligand.pdb**

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* **You can also run only single receptor or a single ligand** \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

NOTE: For instruction to instal mgtools please refer to the README in:

**~/fsl\_groups/grp\_MolecularDock/compute/autodock\_vina/mgltools**

* 1. Preparation of multiple ligands:

You can also prepare multiple ligands with PDB or MOL2 formats. To do this, place the ligands' PDB of MOL2 files into <working\_folder>/Ligands/ folder. And type:

**python3 autodock.py -p -l path/to/Ligands/\*.pdb**

or

**python3 autodock.py -p -l path/to/Ligands/\*.mol2**

* 1. Preparation for a library of ligand
     1. ZINC ligands:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* **CAUTION** \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

There are A LOT of ligands from ZINC15. Before you download, PLEASE BE MINDFUL of the following:

1. The quota of grp\_MolecularDock is 40GB and there are unlimited file limits. So DO NOT execute the command in 3.3.1. It will fill up the entire 40GB and you will.

not be able to do anything.

2. The quota of grp\_MolecularDock/compute is 20TB, but it only allows 1,000,000 files. So if you know you are going to have more than one million ligands, it will

also not going to work. So please make sure you don't use more than one million file ligands. If you absolutely need that many ligands, it will be the best to

break up the ligands into smaller segments before you run the command.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

To prepare a library of ligands, first go to **https://zinc15.docking.org/tranches/home/**, click on the 3D tab on the top, and select the tranches of ligands you want to download. Once you have selected the desired tranches, click on the download button, at the bottom of the popup window, select the "AutoDock (\*.pdbqt.zt)" for download format and select "WGET" for download method.This function is built to run under Linux environment and to process the ligand in PDBQT format. So when downloading the ZINC15 file, PDBQT and WGET option should be selected. Then run:

**python3 autodock.py -Z <Downloaded\_file.gz.wget> <Working\_folder> <Value of file count limit> <Boolen for updating submission script, default=False> <your\_email>**

It will automatically download the ligands using the commands within the downloaded file, it will unzip the downloaded ligands, compile them into a big master file (default as master\_lig.pdbqt) and number each ligand. The master file is for easier file management. If a file count limit is given, it will then segment the master file based on this value. The purposes of segmenting the master file are: 1. Autodock vina only deals with one molecule per PDBQT file 2. lowering the compuational wall time by multi-threading. This will split the molecules into total#\_of\_molecules/filelimit groups with filelimit per group. For example, if there are 27 total molecules and the filelimit = 9, it will result in 3 folders and each folder with 9 PDBQT files.

If a True is passed for the 4th argument, it will update the job submission script automatically. Additionally, it will prepare ligand input text files. Each ligand input contains **total\_#ligand / 5000** lines of ligand paths. They are just paths to the ligand PDBQT files. This is a work around to our slurm-array system, which only allows 5000 subjobs at the same time. So, each working unit will take and process **total\_#ligand / 5000** ligands one after another. This also significantly increases working efficiency and decreases the wall time from days to hours.

3.3.2 FDA approached ligands:

To prepare FDA approved ligands, first go to **https://zinc.docking.org/substances/subsets/fda/?page=1** , click on the download icon in the middle top and select SMI format. You could technically download the SMI format but for some reason the downloading SMI format always goes wrong for me. Once the SMI file is downloaded, do:

**python3 autodock.py -F <downloadedligand>.smi <working\_folder> <filelimit> True <email@example.com>**

If SMI format is not available, which happens sometimes on the ZINC website, you can download TXT format, and replace the .smi file in the above command with the downloaded TXT file. This script will automatically convert the TXT file into SMI format. The above command will compile all molecules into a master file in PDBQT format, segment the ligands based on the filelimit, and update the slurm submission script.

4. Run autodock:

4.1 Before running autodock, **receptorPDBQT**, **ligandPDBQT** and **configuration** files are required. Below is how to execute single docking:

Ex:

**python3 autodock.py -d -r Dock01/Receptor/receptor.pdbqt -l Dock01/Ligand/ligand.pdbqt -c Dock01/config.txt**

4.2 To run more than one docking, using the slurm array system on the supercomputer is preferred. If not yet got a slurm submission scrpit, do:

**python3 autodock.py -u <mode:str> <email:str> <max\_array\_No:int> <working\_folder:str> <exhaustiveness:int> <name:str> <prep\_lig\_txt:boolen, default False>**

For all applications, node = 1, cpu = 1 and memory = 1G should be enough.

node = No. of node requesting, default = 1

cpu = No. of cpu requesting, default = 1

mode = Identifier normal or virtual screen mode. If doing normal docking, input norm. If for virtual screen, input vs.

email = Your email so the supercomputer can send you an email when the run is completed.

max\_array\_No = The number of tasks you need to run, usually the total number of ligands

If you don’t know the number of the tasks you need to run, you can do the following:

**python3 autodock.py -m <path\_to\_your\_working\_folder>**

It will print out the number of ligands.

working\_folder = The working folder of all ligands and receptors. Ex: Dock01/

exhaustiveness = Specifying exhaustiveness.

name = File name of the submission script

prep\_lig\_txt = Prepare the ligand input text file mentioned in 3.3.1, this should be set True when the total ligand count is >= 5000

Ex:

**python3 autodock.py -u vs example@byu.edu 9 Dock01 32 submit\_test.sh**

This will output a slurm script named submit\_test.sh that looks like this:

#!/bin/bash

#SBATCH --time=1-00:00:00 # walltime

#SBATCH --nodes=1 # number of nodes

#SBATCH --ntasks=1 # number of processor cores (i.e. tasks)

#SBATCH --mem-per-cpu=100M # memory per CPU core

#SBATCH -J "AD\_norm" # job name

#SBATCH --mail-user=example@byu.edu # email address

#SBATCH --mail-type=END

#SBATCH --array=1-9

# LOAD MODULES, INSERT CODE, AND RUN YOUR PROGRAMS HERE

source $HOME/.bashrc

# Get everything in the folder into an array

Lig\_array=($(ls Dock01/Ligands/\*.pdbqt))

# Make sure to input a receptor file (first argument) and a config file (second argument)

recep=$1

config=$2

for i in ${SLURM\_ARRAY\_TASK\_ID};

do

python3 autodock.py -d -r $recep -l ${Lig\_array[$((SLURM\_ARRAY\_TASK\_ID - 1))]} -c $config -e 32

done

This script will request 1 nodes, each node with 1 cpus and will take everything that has extension of .pdbqt in the Dock01/Ligands/ and multi-thread them to 9 different tasks.

4.2.1 To run docking on a library n > 1 ligands, you should have already updated the slurm script in step 3. Run autodock\_vina by typing:

**python 3 autodock.py -dv -r <path\_to\_receptorPDBQT> -c <path\_to\_config\_file>**

5. Data analysis:

For single or multiple ligand docking, the structural results will be output to the **working\_folder/output** and the energy output will be output to the **working\_folder/logs**.

For docking of a library, it will not output any structures, but it will output a file with the best binding energy of each ligand into **working\_folder/logs/Virtural\_screen\_result.out.** By default, autodock.py will sort out the top 10 ligands into a folder **<working\_folder>/top\_Ligands.**

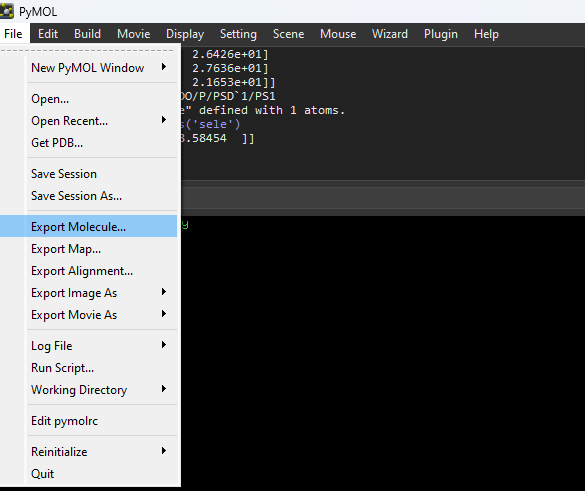
**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* CAUTION \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**The ligands in the top\_ligands/ are the original ligand files, meaning the files that have not gone through the docking processes. To obtain the docked ligands, a more explicit docking using higher exhaustiveness is required.**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

Then you can select the specific ligand to do more explicit docking by increasing the exhaustiveness to, for example, 256.

To visually evaluate the docking, download the result from the supercomputer and load the receptorPDBQT and the resultPDBQT into PyMOL and export and save the combined molecule. This is to combine the receptor and the ligand into one file so you can evaluate the polar interactions such as hydrogen bonds. If two molecules are not combined, it will not show interactions between the ligand and the receptor.

:A screenshot of a computer

Description automatically generated

Load the combined molecule again to PyMOL and type:

**zoom organic**

**set transparency, 0.6**

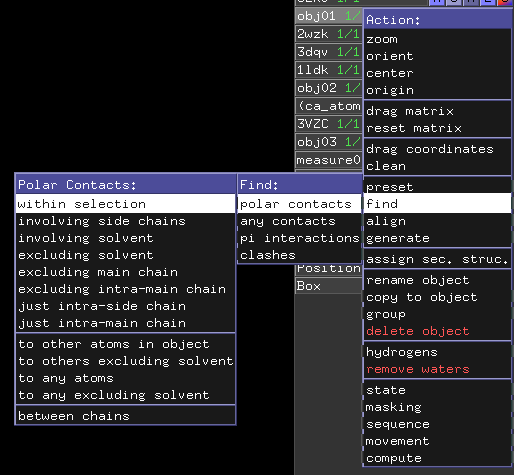
**show surface**

The above commands will 1. Zoom into the ligand, 2. Show the protein surface and set transparency to 0.6.

To find the polar interactions, click on the “A” on the right panel next to the object name that corresponds to your combined molecules. And click on the “find” -> “polar contacts” -> “within selection”.

A screenshot of a computer game

Description automatically generated



Useful PyMOL commands

1. zoom organic -> Zoom into any organic molecules. To zoom in to specific molecules, you can click on it and type: **zoom sele**
2. show surface -> To show the surface of the protein.
3. set transparency, 0.6 -> To set the transparency of the shown surface.

6. Data Packaging:

As mentioned, during virtual screening, there are A LOT of ligand files. To maintain the system quota, it is recommended to go through this process. It will take the **master\_ligand.pdbqt, top\_Ligands and the Virtural\_screen\_result.out** and compress them into a tar.gz file and save this tar.gz file into **~/fsl\_groups/grp\_MolecularDock/autodock\_result** To run this process do:

**python3 autodock.py --package <working\_folder> <tar\_out> <master> <result\_file>**

working\_foler = the fold that the docking process ran in.

tar\_out = the compressed tar.gz file name

master = An optional argument for master file

result\_file = An optional argument for result file

Useful tar commands:

tar -tzf <your\_archive.tar.gz> -> To print out the content of a tar.gz file without extracting.

tar -xzf <your\_archive.tar.gz> <to location> -> To extract tar.gz file to a specific location.

tar -czf <archive\_name.tar.gz> <file1> <file2> ... -> To compress files/folders in to a tar.gz file.