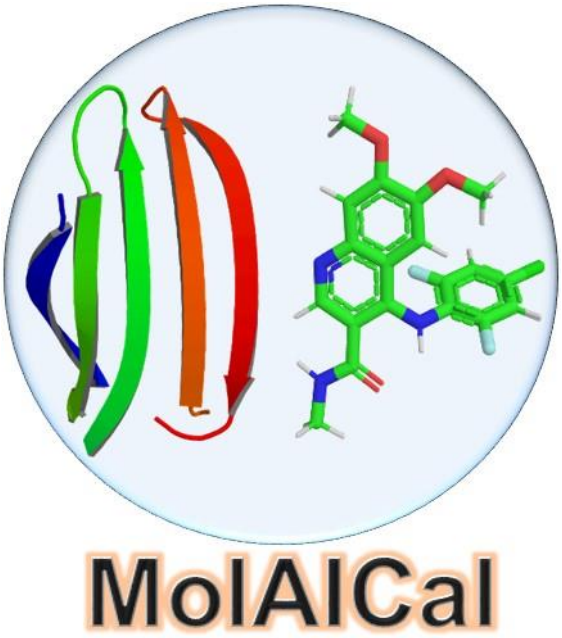
MolAICal

A drug design software combined by artificial intelligence and classical programming

Manual



V1.0

Official site: https://molaical.github.io

Qifeng Bai
Email: molaical@yeah.net
ORCID iD: https://orcid.org/0000-0001-7296-6187
School of Basic Medical Sciences
Lanzhou University
Lanzhou, Gansu 730000, P. R. China

Contents

1. General information of MolAICal

1.1. Introduction

Computer-aided drug design (CADD) employs the computational strategies to screen, develop, modify and analyze drugs and similar biologically active compounds[1]. With the development of technology, artificial intelligence methods such as deep learning, etc are permeated into the CADD research fields. The CADD and deep learning have been widely used in *de novo* drug design, bioactivity prediction, biological image analysis and synthesis prediction successfully[2]. The CADD combined with deep learning can save capital and improve the speed for drug discovery.

Here, the MolAICal soft package is introduced which can be used for drug design by deep learning model, classical algorithms such as genetic algorithm, etc. The detail functions of MolAICal contain as below:

1 The *de novo* drug design: MolAICal can perform fragment grow on the pocket of receptor by using the assigned molecular library or the molecular library generated from deep learning models.

2 The drug virtual screening: MolAICal combined with Autodock Vina can perform drug virtual screening based on the molecular database generated from deep learning models.

3 The useful tools for drug discovery

   1) Channel radii measure for channel receptor such as ion channel proteins.
   2) Potential of mean force for conformational change measurement of receptor induced by ligands, etc.
   3) Quantitative structure-activity relationship (QSAR) for bioactivity prediction of ligands
   4) Lipinski's rule of five[3] for the calculation of drug-like properties.
   5) Pan-assay interference compounds (PAINS)[4] filter

   etc

MolAICal is freely for education, academic and other non-profit purposes. If you want to use MolAICal for any commercial purpose, you should contact with licensors (Email: molaical@yeah.net).

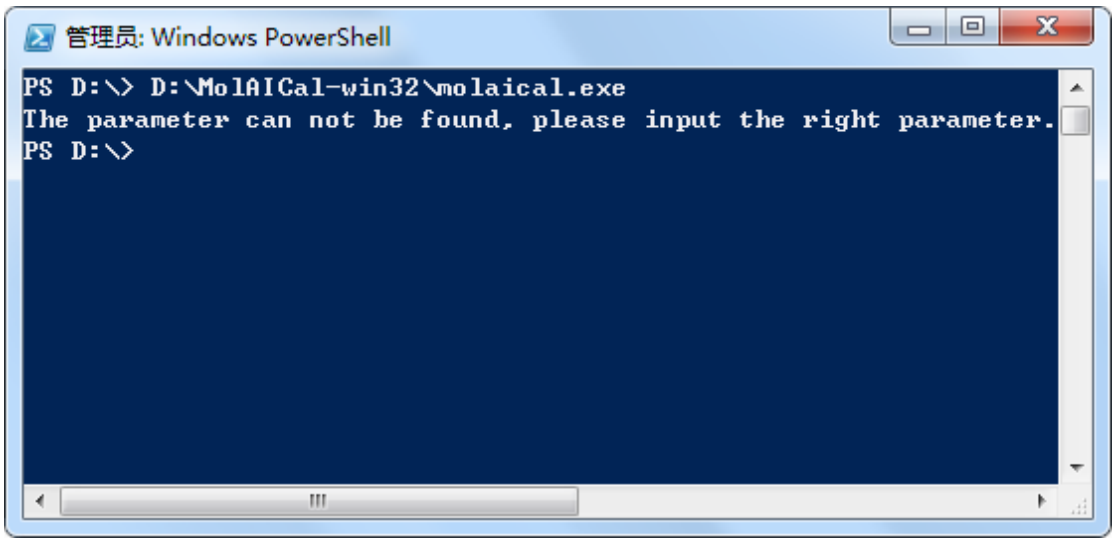
1.2. Installation

Download

MolAICal: https://molaical.github.io

For Window:

MolAICal need not be installed, just decompress the MolAICal file. MolAICal is a command-line tool. You can use the absolute path of MolAICal on DOS or Powershell of Windows. For example: if MolAICal is placed on the D: disk, then you can run as blow:



For Linux:

1. tar -xzvf MolAICal-linux64-xxx.tar.gz

Notice: Please replace the corrected characters in MolAICal-linux64-xxx.tar.gz according to your downloaded MolAICal.

2. cd MolAICal-linux64-xxx
3. chmod +x install.sh
4. ./install.sh

More detail tutorials, please go to MolAICal website: https://molaical.github.io

2. *De novo* drug design by MolAICal and DL

2.1. Theory

The rational drug design is the purpose to find new medications based on the knowledge of the biological targets such as protein, DNA, RNA, etc. Computer-aided drug design is an approach rational drug design to discover, design, optimize, and analyze ligands and similar biologically active drugs in recent years. The fragment-based drug discovery (FBDD) is a de novo drug design method to find the lead compounds. The fragments are small organic chips that are with small size and low molecular weight. The small chemical fragments may bind to the biological target weakly, it can produce a lead with a higher affinity via growing fragments based on the appointed previous anchor fragments in the receptor pocket.

The deep learning which employs the multiple layers to progressively get higher-level features from the raw input data belongs to machine learning and artificial intelligence (see Figure 2.1A). Deep learning has been successfully used in drug discovery[2] and medicine[5]. It can train the molecular generative model which can be used to generate 1D sequence or 2D structural molecules

by generative adversarial network (GAN)[6,7]. Traditionally, classical programming produces new drugs through inputting drug design rules and drug relative data such as molecular weight, logP, etc (see Figure 2.1B). The input drug data and drug design algorithm play an important role in the new drug discovery. For the deep learning method, the target drugs and drug relative data are given. The deep learning is responsible for finding the drug design rules via training the deep neural network (see Figure 2.1C). Although the deep learning can trained drug design rules for new similar properties drugs generation, it still need the effective way to generative 3D structural ligands in the 3D receptor pocket. The classical programing and deep learning have their own merit on drug discovery. Here, MolAICal combined the merits of de novo drug design and deep learning model for drug design. The molecular generative model which train from the fragments of FDA approved drugs by using GAN, can produce the fragments for ligands growth in the receptor pocket. Besides, MolAICal can perform classical de novo drugs which is based on the given small fragments.

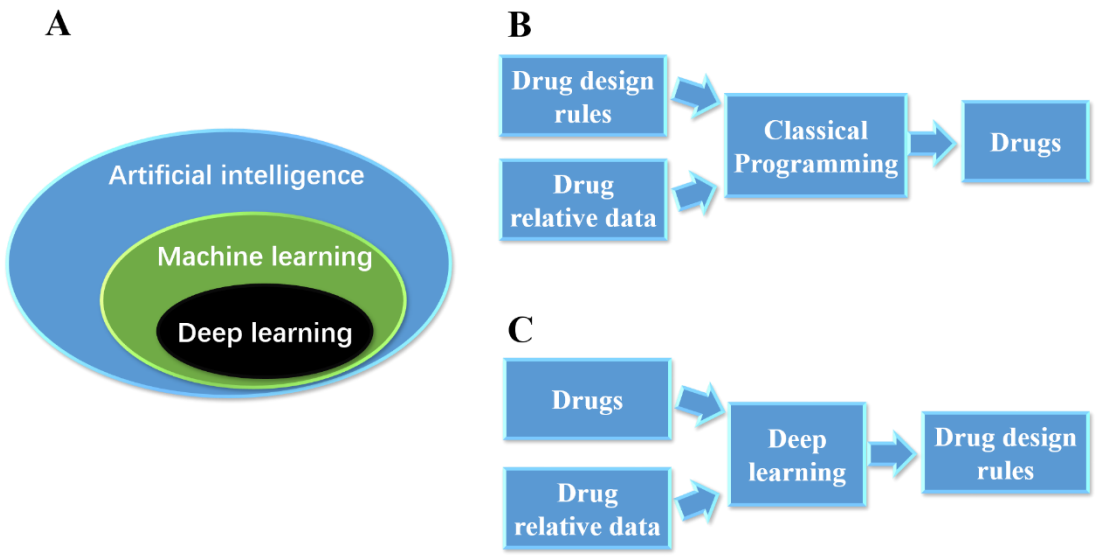


Figure 2.1 The deep learning and algorithm for drug design

2.2. Parameters in configure file

These parameters on this part are contained in the configure file which is the input file for *de novo* drug design. The examples of input configure files are shown in Appendix 1 and 2.

- **centerPoints** < center position of box >

Acceptable Values: decimal
Default Value: null
Description: The center coordinates of appointed box.

- **boxLengthXYZ** < box length >

Acceptable Values: decimal
Default Value: null

Description: It is box length which are sequentially lengths along X, Y, Z axis.

- **receptorPDB** < path of receptor >

Acceptable Values: String
Default Value: null
Description: The path of PDB format receptor in your operation system.

- **growMethod** < grow method >

Acceptable Values: fixFrag, randomFrag
Default Value: null
Description: The 'fixFrag' means the coordinates of initial molecular fragment is fixed, and this molecular fragment may be extracted from the ligand in the crystal receptor-ligand complex. The "randomFrag" means the positions of initial fragment are not fixed, and the initial fragment should be assigned as SMILES format. The initial position should also be appointed. If this part is on, it must set the "startAtomPosition" and "startSmiFrag".

- **startFragFile** < path of initial fragment >

Acceptable Values: String
Default Value: null
Description: The path of initial fragment in your operation system. The initial fragment is sybyl mol2 format.

- **startSmiFrag** < Initial SMILES fragment >

Acceptable Values: String
Default Value: null
Description: If the growMethod is "randomFrag", this parameter should be set. The value is SMILES such as C

- **startAtomPosition** < path of chosen atom file in receptor >

Acceptable Values: String
Default Value: null
Description: If the growMethod is "randomFrag", this parameter should be set. To set the position of initial SMILES fragment, it should appoint the atom of receptor as the coordinate reference. The appointed atom can be extracted from receptor structure by graphical tools such as pymol, vmd, UCSF Chimera, etc. The atom file must be standard PDB format.

- **numCycleGen** < number of grow cycle >

Acceptable Values: Integer
Default Value: null
Description: It is the number of grow cycle. Each cycle grow is based on the previous cycle.

- **selTopMols** < number of parent>

Acceptable Values: Integer
Default Value: null

Description: This part is designed by genetic algorithm. The "selTopMols" represents the number of parents.

- **selTopRootPercent** < percentage of parent selection >

Acceptable Values: decimal
Default Value: null
Description: It is the percentage of parent selection. Multiply "selTopMols" by "selTopRootPercent" is the selected number.

- **selElitePercent** < percentage of elitist selection >

Acceptable Values: decimal
Default Value: null
Description: It is the percentage of elitist selection.

- **maximumPOP** < maximum of populations >

Acceptable Values: Integer
Default Value: null
Description: It is the maximum of populations.

- **libStyle** < Selection way of fragments library >

Acceptable Values: mol2, SMILES, AIFrag
Default Value: null
Description: MolAICal supports three ways for ligands growth. The "mol2" means MolAICal uses its own mol2 format fragments library. The "SMILES" means MolAICal uses its own SMILES sequence library. The "AIFrag" means MolAICal call deep learning model named "AIGenMols" for generation new fragments.

- **genAIWay** < selected model for fragment generation >

Acceptable Values: Fdafrag
Default Value: null
Description: If you set "libStyle" to AIFrag. It should set the value of "genAIWay" which can be set to "Fdafrag".

- **genAINumber** < number of generated fragments >

Acceptable Values: Integer
Default Value: null
Description: If you set "libStyle" to AIFrag. It should set the value of "genAINumber". It can be an arbitrary integer value. For instance, if you set it to 81, it means deep learning model named "AIGenMols" will generate 81 different fragments.

- **perturbeSearch** < whether perform perturbation searching >

Acceptable Values: on or off
Default Value: null
Description: If "on", the procedure will grow fragment by perturbation searching around bonds. If

"off", the procedure will grow fragment along the hydrogen atom direction.

- **genAlgorithm** < searching algorithm for fragment generation anchoring on one point >

Acceptable Values: random, fibonacci
Default Value: null
Description: The "random" means the fragments chosen random points for perturbation growth. The "fibonacci" means the fragments choose the spherical Fibonacci points for perturbation growth.

- **ranGenOptNum** < the number times of fragment generation surrounded on anchoring point >

Acceptable Values: Integer
Default Value: null
Description: It represents the number of fragment generation surrounded on anchoring points. The big number will expend more computational resources, but the ligand grow will be more precise.

- **atomMutation** < mutation on one atom >

Acceptable Values: String
Default Value: null
Description: The value is "on" or "off". The "on" means atoms "C", "N" and "O" will be mutated during the process of ligand grow, and vice versa.

- **mutationPercent** < percentage of mutation ratio >

Acceptable Values: decimal
Default Value: null
Description: If the "atomMutation" is set, the "mutationPercent" should be appointed a value.

- **randomCrossMutation** < crossover and mutation >

Acceptable Values: String
Default Value: null
Description: This parameter is involved in crossover and mutation. It can be value of "on" or "off". "On" means the crossover and mutation are carried out during the process of ligand growth, and vice versa. MolAICal regards the ligand as the chromosome. The fragments connected by rotated bonds regards as "A", "T", "G", "C", etc.

- **randomCrossRatio** < ratio for cross operation >

Acceptable Values: decimal
Default Value: null
Description: If the "randomCrossMutation" is set, the "randomCrossRatio" should be appointed. It is the ratio of crossover operation in the generated population. This mutation unit is one fragment.

- **randomMutationRatio**< ratio of mutation >

Acceptable Values: decimal
Default Value: null
Description: If the "randomCrossMutation" is set, the "randomMutationRatio" should be appointed. It is the ratio of mutation. This mutation unit is one fragment.

- **randomCrossCompareNum** < number of selected molecules for crossover and mutation >

Acceptable Values: Integer
Default Value: null
Description: If the "randomCrossMutation" is set, the "randomCrossCompareNum" should be appointed. It represents the number of selected molecules from the left molecular population. For example, if the value is 2, it means the current ligand will crossover and mutate with 2 ligands of the left random remainder populations.

- **randomCrossRange** < range of crossover and mutation >

Acceptable Values: Discrete integer
Default Value: null
Description: If the "randomCrossMutation" is set, the "randomCrossRange" should be appointed. It corresponds to "numCycleGen". The discrete integer corresponds to the cycle of "numCycleGen" which represents this cycle performs crossover and mutation.

- **writeResultParallel** < parallel writing mode >

Acceptable Values: String
Default Value: null
Description: If "on", it means the temporal file is saved as the parallel mode, and vice versa.

- **unwantedFrag** < filtering unwanted fragments >

Acceptable Values: String
Default Value: null
Description: If "on", it will filter the generated molecules contained unwanted fragments, and vice versa.

- **PAINS** < filtering Pan-assay interference compounds >

Acceptable Values: String
Default Value: null
Description: If "on", it will filter the generated molecules contained Pan-assay interference compounds (PAINS) fragments, and vice versa.

- **growFilter** < filtering wrong position molecules >

Acceptable Values: String
Default Value: null
Description: If "on", it will filter the generated molecules which are overlap and out of box fragments, and vice versa.

- **growFilterRatio** < cutoff of overlap molecule >

Acceptable Values: decimal
Default Value: null
Description: This ratio is used to filter overlap molecule during the storing process of temporal file.

- **resultsFilterRatio** < cutoff of overlap molecule >

Acceptable Values: decimal
Default Value: null
Description: This ratio is used to filter overlap molecule during the storing process of final results ligands.

- **syntheticAccessibility** < synthetic accessibility >

Acceptable Values: String
Default Value: null
Description: If "on", it will filter the unreachable criterion ligands according to the set value of synthetic accessibility, and vice versa.

- **synAccessibilityValue** < cutoff of synthetic accessibility >

Acceptable Values: decimal
Default Value: null
Description: If the "syntheticAccessibility" is set, the "synAccessibilityValue" should be appointed. Synthetic accessibility (SA) is a cutoff ranging from 0 to 100 in which value 100 is maximal synthetic accessibility, it means this molecule is most easily synthesizable.

- **MolsimilarPercent** < filtering the similar ligands >

Acceptable Values: decimal
Default Value: null
Description: This parameter is used to filter similar ligands and add new novel ligands into the parents' populations according to the cutoff.

- **RulesOfFive** < Lipinski's rule of five >

Acceptable Values: String
Default Value: null
Description: If "on", it will filter the unreachable criterion ligands according to the Lipinski's rule of five, and vice versa. The Lipinski's rule of five is changed with the passage of time[8], please set suitable value according to the research reports or your customization.

- **xlogPvalue** < XLogP value >

Acceptable Values: decimal
Default Value: null
Description: It is the XLogP value. If the "RulesOfFive" is set, the "xlogPvalue" should be appointed.

- **acceptors** < number of hydrogen bond acceptors >

Acceptable Values: Integer
Default Value: null
Description: It is the number of hydrogen bond acceptors. If the "RulesOfFive" is set, the "acceptors" should be appointed.

- **donors** < number of hydrogen bond donors >

Acceptable Values: Integer
Default Value: null
Description: It is the number of hydrogen bond donors. If the "RulesOfFive" is set, the "donors" should be appointed.

- **mwvalue** < molecular weight >

Acceptable Values: decimal
Default Value: null
Description: It is the molecular weight. If the "RulesOfFive" is set, the "mwvalue" should be appointed.

- **rotatablebonds** < number of rotatable bonds >

Acceptable Values: integer
Default Value: null
Description: It is the number of rotatable bonds. If the "RulesOfFive" is set, the "rotatablebonds" should be appointed.

- **pcClusterNumber** < number of clusters >

Acceptable Values: integer
Default Value: null
Description: Performing appointed number of clusters on the final generated ligands by K-means algorithm.

- **adjustOutSimPer** < fingerprint similarity >

Acceptable Values: decimal
Default Value: 0.999999
Description: This parameter filters similar ligands according to the values of fingerprint similarity when performing the cluster on the final generated ligands. The 0.999999 means it filters the same molecule approximatively.

- **selMolMaxWeight** < maximum molecular weight >

Acceptable Values: decimal
Default Value: null
Description: This parameter filters the final generated ligands which is greater than maximum molecular weight.

- **selMolMinWeight** < minimum molecular weight >

Acceptable Values: decimal
Default Value: null
Description: This parameter filters the final generated ligands which is smaller than minimum molecular weight.

- **selBindingScore** < cutoff of binding score >

Acceptable Values: decimal
Default Value: null
Description: This parameter filters the final generated ligands which have higher binding score than the set binding score.

- **obabelPath** < path of obabel >

Acceptable Values: String
Default Value: path of MolAICal
Description: It represents the path of obabel of Open Babel. The default value is the path of MolAICal, so you can omit it if you have the full package of MolAICal.

- **pcClusterdFile** < path of cluster file>

Acceptable Values: String
Default Value: current directory
Description: It represents the path of cluster file named "PC.dat". The default value is the current directory, so you can omit it.

- **moloutputdir** < storing directory of final ligands >

Acceptable Values: String
Default Value: current directory
Description: It represents the storing directory of the final ligands. The default value is the current directory named "results", so you can omit it.

- **coreNum** < number of cores for parallel computing >

Acceptable Values: Integer
Default Value: null
Description: It represents the number of cores for parallel computing. You can set it to an arbitrary value, but MolAICal can use the limited core. For example, MolAICal can use maximally ~15 cores for loading 67 molecular fragments at one job.

2.3. The commands for fragments grow

The command parameters of MolAICal for *de novo* drug design are as below:
-denovo: means perform job of *de novo* drug design. Value is "grow".
-i: input path of configure file.
-g: whether calculate grids, values is "on" or "off", default value is "on".
-l: whether perform fragment grow, values is "on" or "off", default value is "on".
-o: whether output results, values is "on" or "off", default value is "on".

Here, the example commands for fragment grow are listed:
1) If you want to run full process of fragment growth, you can use command:
molaical.exe -denovo grow -i InputParFile.dat

2) This command omits the grids generation, only contains fragments grow and output results.
molaical.exe -denovo grow -g off -i InputParFile.dat

3) This command is output results only.
molaical.exe -denovo grow -g off -l off -i InputParFile.dat

4) This command is no results output.
molaical.exe -denovo grow -g off -l off -o off -i InputParFile.dat

5) This command generates grids only
molaical.exe -denovo grow -g on -l off -o off -i InputParFile.dat

3. Virtual screening by deep learning model

3.1. Drug virtual screening by MolAICal, DL model and Autodock Vina

3.1.1. Theory

Virtual screening (VS) which is different from de novo drug design, is a computational method used in drug discovery to find potential small molecules which most likely bind to the drug target, typically protein receptor or DNA, etc. The VS contains ligand-based and structure-based virtual screening[9]. Especially, the structure-based virtual screening involves molecular docking which searches good binding poses of the ligands into the protein target followed by applying a scoring function to estimate the affinity ability between ligands and receptors. Molecular docking is likely the theory of the Lock and Key Model (see Figure 3.1.1). The Autodock Vina[10] is an excellent open source molecular docking software which has been studied and applied to the fields of drug discovery. The score function of Autodock Vina is also studied and optimized in used to estimate the binding ability between the ligands and receptors. The Vinardo score[11] is a variant of score function of Autodock Vina which can improve the prediction of binding ability between ligands and receptors.

The deep learning which employs the multiple layers to progressively get higher level features from the raw input data belongs to machine learning and artificial intelligence. Deep learning has been successfully used in drug discovery[2] and medicine[5]. It can train the molecular generative model which can be used to generate 1D sequence or 2D structural molecules by generative adversarial network (GAN)[6,7]. However, the rational drug design need study the 3D structural ligand in the receptor pocket. The de novo drug design provides a way to design 3D drugs in the receptor pocket

by deep learning model and classical programming algorithm such as genetic algorithm, etc. Besides, the molecular docking supplies another way to take advantage of the deep learning model and classical programming. The strategies for virtual screening is that MolAICal invokes the deep leaning model to generate the appointed number of drugs, and then call Autodock Vina to perform the molecular docking to evaluate the binding ability between ligand and receptor (see Figure 3.1.1). Besides, MolAICal also can utilize the appointed small molecular database for drug virtual screening.

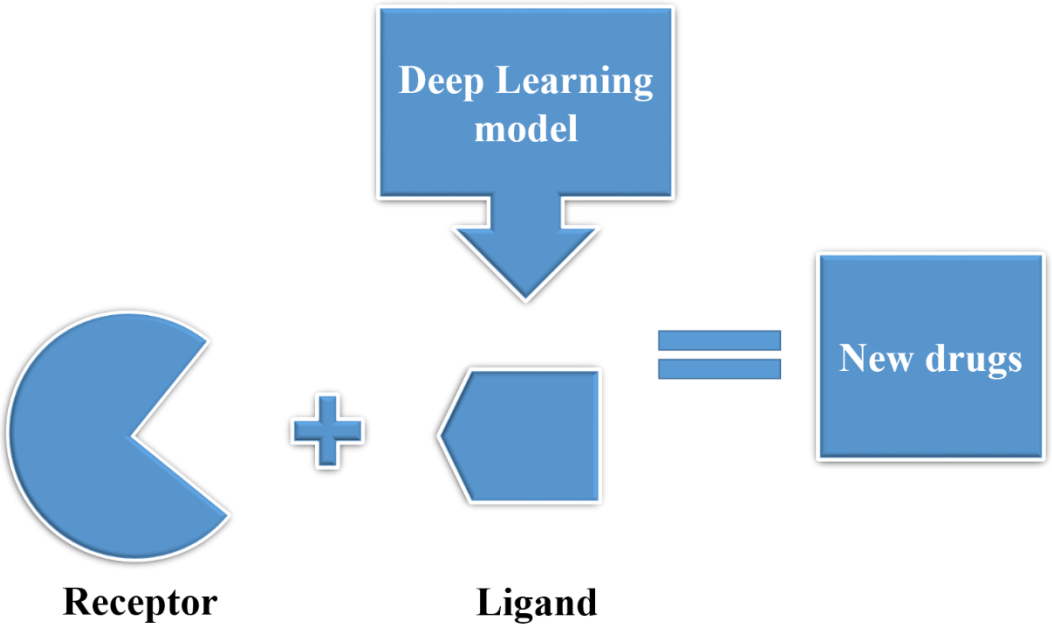


Figure 3.1.1 Virtual screening based on molecular docking and deep learning model

3.1.2. Introduction of parameters of Vina configure file

This part introduces the parameters of drug virtual screening by MolAICal, deep learning (DL) model (Note: only 64bit MolAICal can perform the AI and molecular docking completely. 32bit MolAICal can only perform molecular docking. Please run this process on 64 bit operating system.). Before you run this drug virtual screening, please prepare the necessary Vina configure files named "conf.txt". The pdbqt format files of protein must be together with "conf.txt". The content "conf.txt" is like as below:

receptor = protein.pdbqt

cpu = 1
center_x = -30.011
center_y = 1.665
center_z = -36.581

size_x = 30
size_y = 30
size_z = 30

num_modes = 3

Where the “receptor” represents protein file, “cpu” means number of core for running, the “center_x”, “center_y”, “center_z” are the x, y, z coordinates of box center you set, the “size_x”, “size_y”, “size_z” are lengths along x, y, z axis of box you set, the “num_modes” is the number of generated conformation by Autodock Vina.

3.1.3. Command introduction for virtual screening

The command parameters of MolAICal for drug design virtual screening are as below:

-dock: means performing drug virtual screening
-s: selection of deep learning model. Values are “ZINCMol”, “FDAMol”, “FDAFrag”
-n: number of generated molecules by deep learning model.
-nf: number of generated molecules in one folder. It avoids very huge of molecules in one folder.
-p: path of obabel.exe. The default value is in the same directory of MolAICal. So you can omit it if you have full MolAICal soft package.
-nc: number of core for running drug virtual screening
-g: switch AI molecular generations. “on” means generation. “off” means no generation.
-b: switch molecular format conversion. “on” means conversion. “off” means no conversion.
-v: switch virtual screening (VS) based on docking. “on” means VS. “off” means no VS.
-m: molecular format for VS, etc.

Here, the example commands for drug virtual screenings are listed:

1) The simple command, the omitted parameters use the default values.
molaical.exe -dock -s FDAMol -n 30 -nf 10 -nc 3

2) Run full drug virtual screening process by MolAICal, deep learning model, and Autodock Vina.
molaical.exe -dock -s FDAMol -n 30 -nf 10 -nc 3 -g on -b on -v on

3) Only run molecular artificial intelligence (AI) generation.
molaical.exe -dock -s FDAMol -n 30 -nf 10 -nc 3 -g on -b off -v off

4) Only run conversion of molecular formats only
molaical.exe -dock -s FDAMol -n 30 -nf 10 -nc 3 -g off -b on -v off

5) Only run Vina docking only. So you can customize your own AI SMILES file, but this file must be named “tmpGen.dat”.

molaical.exe -dock -s FDAMol -n 30 -nf 10 -nc 3 -g off -b off -v on

6) Command for virtual screening by calling deep learning mode
molaical.exe -dock -s FDAMol -n 30 -p "C:/Program Files (x86)/OpenBabel-2.4.1/obabel.exe" -nf 10 -nc 3 -g on -b on -v on

7) Command contained full parameters.
molaical.exe -dock -s ZINCMol -n 30 -p "C:/Program Files (x86)/OpenBabel-2.4.1/obabel.exe" -nf 10 -nc 3 -g on -b on -v on -m pdbqt

8) This command have no drug virtual screening task.
molaical.exe -dock -s FDAMol -n 30 -p "C:/Program Files (x86)/OpenBabel-2.4.1/obabel.exe" -nf 10 -nc 3 -g on -b on -v off -m pdbqt

9) Run AI generation and conversion of molecular formats, so you can convert the results to the formats you want.
molaical.exe -dock -s FDAMol -n 30 -nf 10 -nc 3 -g on -b on -v off -m mol2

3.2. Commands for MM/GBSA using NAMD

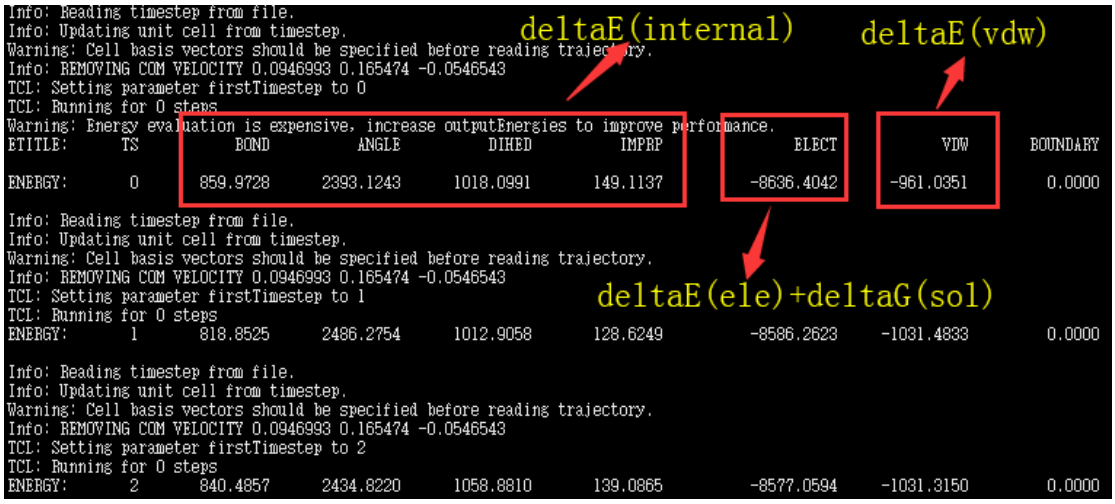
Molecular mechanics generalized Born surface area (MM/GBSA)[12,13] is a popular way to estimate the binding free energy between two molecules such as ligand-protein, protein and protein, etc. Here, MM/GBSA is calculated by MolAICal on basis of MD log file of NAMD software. The MM/GBSA is estimated by the equation as below:

$$\Delta G_{bind} = \Delta H - T\Delta S \approx \Delta E_{MM} + \Delta G_{sol} - T\Delta S$$

$$\Delta E_{MM} = \Delta E_{internal} + \Delta E_{ele} + \Delta E_{vdw}$$

$$\Delta G_{sol} = \Delta G_{GB} + \Delta G_{SA}$$

Where ΔE_{MM}, ΔG_{sol}, and $T\Delta S$ represent the gas phase MM energy, solvation free energy (sum of polar contribution ΔG_{GB} and nonpolar contribution ΔG_{SA}) and conformational entropy, respectively. ΔE_{MM} contains van der Waals energy ΔE_{vdw}, electrostatic ΔE_{ele} and $\Delta E_{internal}$ of bond, angle, and dihedral energies. If there are no binding-induced structural changes in the process of molecular dynamics (MD) simulations, the entropy calculation can be omitted. Currently, the MolAICal can evaluate the binding free energy without entropy calculation based on the MD results log of NAMD. The variables in the above equations correspond to Figure 3.2.1.




```
Info: Reading timestep from file.
Info: Updating unit cell from timestep.
Warning: Cell basis vectors should be specified before reading trajectory.
Info: REMOVING COM VELOCITY 0.0946993 0.165474 -0.0546543
TCL: Setting parameter firstTimestep to 0
TCL: Running for 0 steps
Warning: Energy evaluation is expensive, increase outputEnergies to improve performance.
ETITLE:      TS           BOND          ANGLE          DIHED          IMPRP               ELECT            VDW       BOUNDARY

ENERGY:       0       859.9728      2393.1243      1018.0991       149.1137         -8636.4042      -961.0351         0.0000

Info: Reading timestep from file.
Info: Updating unit cell from timestep.
Warning: Cell basis vectors should be specified before reading trajectory.
Info: REMOVING COM VELOCITY 0.0946993 0.165474 -0.0546543
TCL: Setting parameter firstTimestep to 1
TCL: Running for 0 steps
ENERGY:       1       818.8525      2486.2754      1012.9058       128.6249         -8586.2623      -1031.4833         0.0000

Info: Reading timestep from file.
Info: Updating unit cell from timestep.
Warning: Cell basis vectors should be specified before reading trajectory.
Info: REMOVING COM VELOCITY 0.0946993 0.165474 -0.0546543
TCL: Setting parameter firstTimestep to 2
TCL: Running for 0 steps
ENERGY:       2       840.4857      2434.8220      1058.8810       139.0865         -8577.0594      -1031.3150         0.0000
```

Figure 3.2.1 The detail information of NAMD running log file

The parameters for calculating binding free energy is as below:

-mmgbsa: means MM/GBSA calculation
-c: the MD log of receptor and ligand
-r: the MD log of receptor
-l: the MD log of ligand.

Example command line as below:

molaical.exe -mmgbsa -c "D:\workdir\ mmgbsa\complex.log" -r "D:\workdir\ mmgbsa\protein.log" -l "D:\workdir\mmgbsa\ligand.log"

4. Useful tools for drugs design

4.1. Channel radii measurement

The Metropolis Monte Carlo simulated annealing method[14] is used to measure the channel radii of objects such as nanotube, ion channel protein, etc. The initial can be defined anywhere in the channel, and the explore vector is approximately along the direction of channel. The calculation of channel radii is realized in MolAICal. The examples of configure files for radii calculation shows in the Appendix 3 and 4. The parameters for channel radii calculation is listed as below:

- **pdbpath** < The path of object with pdb format>

Acceptable Values: String
Default Value: null
Description: It should point out the path of object for radii measurement.

- **psfpath** < The path of object with psf format>

Acceptable Values: String
Default Value: null
Description: The path of object with psf format is not necessary if you only perform the simple radii measurement in the channel of object. If you want to measure the radii of object by using CHARMM force field, you should give the path of psf file which corresponds to pdb file.

- **cpoint** < The initial point for radii measurement>

Acceptable Values: decimal
Default Value: null
Description: This is the initial point for radii measurement. The values following cpoint are x, y, z coordinates.

- **Vector** < The measurement direction>

Acceptable Values: decimal
Default Value: null
Description: This parameter has three values which represent x, y and z directions. For example, 1.00 0.00 0.00 represent measure channel radii along x axis. 0.00 1.00 0.00 represent measure channel radii along y axis. 0.00 0.00 1.00 represent measure channel radii along z axis.

- **sample** < The interval of channel radii measurement along appointed direction>

Acceptable Values: decimal
Default Value: null
Description: It is the interval of channel radii measurement along the appointed direction. You can set it according to your measure requirement.

- **minprobe** < The minimum explored probe>

Acceptable Values: decimal
Default Value: 0.259
Description: It is the minimum explored detector. Usually, it uses the default value, the program will calculate the minimum explored detector.

- **conpar** < control constant>

Acceptable Values: decimal
Default Value: 0.15
Description: It is control constant. The detail parameter is shown on equation 4 in this paper[15]. Usually, it use the default value 0.15.

- **endrad** < cutoff for radii measurement termination>

Acceptable Values: decimal
Default Value: null
Description: It is the cutoff value for radii measurement termination. If the measured radii are larger than the setting cutoff, the program will end.

- **selatoms** < select atoms within the range of object>

Acceptable Values: decimal
Default Value: null
Description: For large system, it need calculate a lot of unnecessary atoms. In this case, the selected atoms can save computing time. This value should be larger than the cutoff value of endrad parameter.

- **numpt** < number of points in the surface>

Acceptable Values: integer
Default Value: null
Description: The final output contains grid file which can show the grid surface of channel in the VMD software[16]. This parameter can determine the number of surface points showed in the VMD software.

- **surColor** < surface color>

Acceptable Values: string
Default Value: null
Description: It is the surface color. Because the color is shown in the VMD software, the value of surColor can be chosen from color values of VMD software.

- **lineColor** < color of radii measurement routing>

Acceptable Values: string
Default Value: null
Description: The channel radii are measured along the appointed direction. The whole routing shows a curve in the channel. This parameter sets the color of curve. The value of lineColor can be chosen from color values of VMD software.

- **lineWidth** < width of radii measurement routing>

Acceptable Values: integer
Default Value: null
Description: It represents the width of curve. The "lineColor" is responsible for color of measured routing, while the "lineWidth" is responsible for the width of measured routing.

- **searchNum** < measured number of radii in one plane>

Acceptable Values: integer
Default Value: null
Description: It is a trial number for radii measurement of the channel in one plane.

- **material** < material property of sphere surface >

Acceptable Values: integer
Default Value: null
Description: It represents the material property of sphere surface which is shown in VMD software. It can be chosen from the material values of VMD software.

The command parameters of MolAICal for channel radii measurement are as below:

-channel: means radii measurement function.
-cpp: path of input configure file which contains the parameters for radii calculations
-fc: means the calculated way. It contains two ways. One way is simple radii calculation, the other is radii measurement with CHARMM force field which needs PSF file format.

Here, the example commands for channel radii measurement are shown as below:

1) Command for simple radii calculation.
molaical.exe -channel radii -cpp D:/GramicidinA/parameter.dat -fc simple

2) Command for radii calculation with CHARMM force field.
molaical.exe -channel radii -cpp D:/GramicidinA/parameter.dat -fc charmm

3) Command with default parameters. The default calculated way is simple radii calculation.
molaical.exe -channel radii -cpp D:/GramicidinA/parameter.dat

4.2. Potential of mean force

The potential of mean force (PMF) can be used to calculate the free energy changes as the function of a coordinate of system. The PMF along the coordinate is computed from the average distribution function (see below equation).
$\Delta \mathrm{G} = -k_B \mathrm{T} \ln \rho(\mathrm{x}, y)$
Where T and k_B is the temperature and Boltzmann constant, respectively. The x and y represent two principal components.

The command parameters of MolAICal for PMF calculation are as below:

-pmf: means PMF calculation function
-i: path of input data file
-g: number of grid for density calculation in a square
-l: number of color levels.
-m: choose contours from DISLIN. Currently, it contains “conshd” and “contur”
-b: is LABELS. It determines which label types will be plotted on an axis. The values can be referred in the DISLIN help
-x: is x-axis label
-y: is y-axis label
-t: is temperature.
-f: is output figure file.

The example commands for PMF are shown as below:

1) Do PMF calculation by using the default parameter
molaical.exe -pmf -i D:/pmf/diher.dat

2) Do PMF calculation with “conshd” contour
molaical.exe -pmf -i D:/pmf/diher.dat -g 20 -l 10 -m conshd -x "x data" -y "y data"

3) Do PMF calculation with “contur” contour without numerical label
molaical.exe -pmf -i D:/pmf/diher.dat -g 20 -l 10 -m contur -b none -x "x data" -y "y data"

4) Do PMF calculation with “contur” contour without numerical label.
molaical.exe -pmf -i D:/pmf/diher.dat -g 20 -l 10 -m contur -b float -x "x data" -y "y data" -t 310 -f Figure.tif

4.3. Molecular properties calculation

4.3.1. Quantitative structure-activity relationship (QSAR)

Quantitative structure-activity relationship (QSAR) is a method involved in regression or classification used in the chemical and biological sciences and engineering. Here, the QSAR is embedded into MolAICal software with genetic algorithm. The detail parameters for run QSAR is listed as below:

-qsar: means QSAR calculation function
-i: is the path of input data file for QSAR calculation
-im: is mutation ratio
-ic: number selection for fittest calculation
-ie: whether choose elitism way
-ip: population size
-iw: choose the validation. Currently, it contains least squares R2 and cross validation Q2.
-in: number of set variables. If it set 3, it will be like: y = a*x1+b*x2+c*x3
-iv: number of evolution
-is: interval steps for repeating population generation.
-io: number core for parallel computing.

The example commands for QSAR calculations are as below:

1) QSAR calculation with default parameters
molaical.exe -qsar GA -i D:\\QSAR\\all_QSAR_no.txt

2) QSAR calculation with full parameters
molaical.exe -qsar GA -i D:\\QSAR\\test1.txt -im 0.5 -ic 5 -ie true -ip 200 -iw Q2 -in 3 -iv 150000 -is 100 -io 3

3) QSAR calculation use parallel running way acquiescently
molaical.exe -qsar GA -i D:\\QSAR\\test1.txt -im 0.5 -ic 5 -ie true -ip 200 -iw Q2 -in 3 -iv 150000 -is 100

4.3.2. Lipinski's rule of five calculation

Lipinski's rule of five also known as the rule of five (RO5) is a rule to estimate drug-like or determine if a chemical compound has pharmacological or biological activity that would be likely orally active drug[3]. The RO5 cutoff values are multiples of five, which is the origin of the rule's name. It contains three or four rules. The MolAICal evaluate drug-like with five rules:

1) No more than 5 hydrogen bond donors
2) No more than 10 hydrogen bond acceptors
3) A molecular mass less than 500 daltons
4) An octanol-water partition coefficient (log P) that does not exceed 5
5) No more than 10 rotatable bonds

In the past years, some rules of Lipinski's rule of five have changed according to the statistic research on the properties of FDA approved drugs (see *J Med Chem. 2019 Feb 28;62(4):1701-1714*)[8]. To estimate the drug-like of ligands, the author can set suitable parameters according to the studied need. MolAICal supplies an easy way to set rules parameters.

-tool: appoint tool function. Here, it is “ruleoffive”
-f: molecular format
-n: molecule file
-i: the file which contains molecules list
-o: output results file
-x: logP
-a: number of hydrogen bond acceptors
-d: number of hydrogen bond donors
-m: molecular mass
-r: number of rotatable bonds

The example commands for RO5 calculations are as below:

1) Calculate RO5 with default original rules (see above five rules)
molaical.exe -tool ruleoffive -f mol2 -n "D:\zinc_1879871.mol2"

2) Calculate RO5 of ligands set. Note: “mol2List.dat” must be in the directory of molecular set.
molaical.exe -tool ruleoffive -f mol2list -i "D:\mol2List.dat" -o "D:\result.dat"

3) Calculate RO5 with your given values of rules
molaical.exe -tool ruleoffive -f mol2list -i "D:\mol2List.dat" -o "D:\result.dat" -x 5.0 -a 10 -d 5 -m 500.0 -r 10

4.3.3. PAINS calculation

Pan-assay interference compounds (PAINS) are the compounds which often show the false positive results in the biological assay[4]. PAINS tend to react with numerous biological receptors nonspecifically rather than binding the desired target specifically[17]. A number of disruptive functional groups are shared by many PAINS. The Common PAINS contains “toxoflavin, isothiazolones, curcumin, hydroxyphenyl hydrazones, phenol-sulfonamides, enones, rhodanines, catechols, and quianones”[17]. The MolAICal software provides an easy way to calculate PAINS:

-tool: appoint tool function. Here, it is “pains”
-f: molecular format. It contains “SMILES” and “mol2” formats. We recommend “SMILES” format. You can use Open Babel[18] change it to “SMILES” format.
-n: input molecular file or SMILES sequence
-i: input file which contains molecules list or sequences
-o: output results

The example commands for PAINS calculations are as below:
1) Calculate PAINS with SMILES format
molaical.exe -tool pains -f smiles -n "c1ccccc1N=Nc1ccccc1"

2) Calculate PAINS with abbreviation SMILES command parameter
molaical.exe -tool pains -f smi -n "c1ccccc1N=Nc1ccccc1"

3) Calculate PAINS with mol2 format
molaical.exe -tool pains -f mol2 -n "D:\ZINC00154323.mol2"

4) Calculate PAINS from the file with contains the list of molecular SMILES sequences
molaical.exe -tool pains -f smilist -i "D:\painsTest.txt" -o "D:\smiResult.txt"

5) Calculate PAINS from the file with contains mol2 list
molaical.exe -tool pains -f mol2list -i "D:\list.dat" -o "D:\mol2Result.txt"

4.3.4. Synthetic Accessibility calculation

The synthetic accessibility (SA)[19] is an important aspect of drug design. The prediction of SA can guide the de novo drug design and give chemists useful information for compound synthesis. The calculated parameters of SA by MolAICal is shown as below:

-tool: appoint tool function. Here, it is "sa"
-i: input molecular SMILES sequence or file contained the list of molecular SMILES sequences.

The example commands for SA calculations are as below:
1) Calculate SA with molecular SMILES sequence
molaical.exe -tool sa -i "FC(F)(F)c1cc(ccc1)N5CCN(CCc2nnc3[C@H]4CCC[C@H]4Cn23)CC5"

2) Calculate SA from the file contained the list of molecular SMILES sequences
molaical.exe -tool sa -i "D:\\SA\\SmilTest.smi"

4.4. Vinardo score calculation

Vinardo score which is based on Vina score function[11], is trained on basis of PDBbind database[20-22]. The score function of Vina is show as below. The Vinardo score can be trained by combined with the below equations.

$$Gauss_1(d) = e^{-((d-o_1)/s_1)^2} \quad (1)$$

$$Gauss_2(d) = e^{-((d-o_2)/s_2)^2} \quad (2)$$

$$Repulsion(d) = \begin{cases} d^2, & \text{if } d < 0\text{Å} \\ 0, & \text{if } d \geq 0\text{Å}, \end{cases} \quad (3)$$

$$Hydrophobic(d) = \begin{cases} 1, & \text{if } d < p_1 \\ p_2 - d, & \text{if } p_1 \leq d \leq p_2 \\ 0, & \text{if } d > p_2 \end{cases} \quad (4)$$

$$Hbond(d) = \begin{cases} 1, & \text{if } d < h_1 \\ 1 - \dfrac{d - h_1}{-h_1}, & \text{if } h_1 \leq d \leq 0\text{Å} \\ 0, & \text{if } d > 0\text{Å} \end{cases} \quad (5)$$

4.4.1. Batch of Vinardo score calculation

MolAICal retrains the Vinardo score according to the above five equations. And the command parameters are as below:

-score: score function for evaluation of binding affinity. Here, it is "vinardo"
-i: input molecular file or file contained molecules' list
-o: output result
-g: the referred grid file for binding affinity calculation
-p: pick up the computing way for Vinardo score calculations
-n: number of cores for parallel computing

The example commands for Vinardo score calculations are as below:

1) Calculate Vinardo score without loading all molecules
molaical.exe -score vinardo -i D:\workdir\ligandList.dat -o D:\workdir\output.dat -g D:\workdir\MAICGrid.dat

2) Calculate Vinardo score by loading all molecules.
molaical.exe -score vinardo -i D:\workdir\ligandList.dat -o D:\workdir\output.dat -g D:\workdir\MAICGrid.dat -p memoryWay -n 2

4.4.2. Vinardo score Decomposition

The MolAICal can calculate the $Gauss_1$ score, $Gauss_2$ score, Repulsion score, Hydrophobic score, Hbond score and Vinardo score based on one ligand, respectively (see equations in 3.7). The calculated parameters are below:

-tool: appoint tool function. Here, it is "vinardo"
-i: input molecular file
-g: the referred grid file
-t: type of score which is gaussScore1, gaussScore2, repulsionScore, hydrophobicScore, hbondScore or vinardoScore.
-n: number of cores for parallel computing

1) Calculate Vinardo score from one molecule.
molaical.exe -tool vinardo -i D:\workdir\1a1e_ligand.mol2 -g D:\workdir\vinardoScore.dat -t vinardoScore

2) Calculate $Gauss_1$ score with 1 core of computer
molaical.exe -tool vinardo -i D:\workdir\1a1e_ligand.mol2 -g D:\workdir\gaussScore1.dat -t gaussScore1 -n 1

3) Calculate $Gauss_2$ score
molaical.exe -tool vinardo -i D:\workdir\1a1e_ligand.mol2 -g D:\workdir\gaussScore2.dat -t gaussScore2

4) Calculate Repulsion score
molaical.exe -tool vinardo -i D:\workdir\1a1e_ligand.mol2 -g D:\workdir\repulsionScore.dat -t repulsionScore

5) Calculate Hydrophobic score
molaical.exe -tool vinardo -i D:\workdir\1a1e_ligand.mol2 -g D:\workdir\hydrophobicScore.dat -t hydrophobicScore

6) Calculate Hbond score
molaical.exe -tool vinardo -i D:\workdir\1a1e_ligand.mol2 -g D:\workdir\hbondScore.dat -t hbondScore

4.4.3. Grid file generation for score calculation

The $Gauss_1$ score, $Gauss_2$ score, Repulsion score, Hydrophobic score, Hbond score and Vinardo score need the grid files for binding score calculation. MolAICal supplies a module to calculate the relative grid files for binding score calculation. The parameters are below:

-tool: appoint tool function. Here, it is "grid"
-i: the path of box file which is measured based on receptor
-p: the path of receptor file
-n: term of Vinardo score which can be gaussScore1, gaussScore2, repulsionScore, hydrophobicScore, hbondScore or vinardoScore
The example commands are as below:

1) Grid file generation of $Gauss_1$ score
molaical.exe -tool grid -i "D:\1a1e\boxPar.dat" -p "D:\1a1e\protein2.pdb" -n gaussScore1

2) Grid file generation of $Gauss_2$ score
molaical.exe -tool grid -i "D:\1a1e\boxPar.dat" -p "D:\1a1e\protein2.pdb" -n gaussScore2

3) Grid file generation of Repulsion score
molaical.exe -tool grid -i "D:\1a1e\boxPar.dat" -p "D:\1a1e\protein2.pdb" -n repulsionscore

4) Grid file generation of Hydrophobic score
molaical.exe -tool grid -i "D:\1a1e\boxPar.dat" -p "D:\1a1e\protein2.pdb" -n hydrophobicscore

5) Grid file generation of Hbond score
molaical.exe -tool grid -i "D:\1a1e\boxPar.dat" -p "D:\1a1e\protein2.pdb" -n hbondscore

6) Grid file generation of Vinardo score
molaical.exe -tool grid -i "D:\1a1e\boxPar.dat" -p "D:\1a1e\protein2.pdb" -n vinardoScore

4.4.4. Box generation based on receptor

Grid files generation need the box parameters which are extracted on basis of receptor. MolAICal supply a function to generate box parameters which can display in UCSF Chimera[23]. The parameters is below:

-tool: appoint tool function. Here, it is “box”
-i: box center
-l: box length
-t: transparency shown in UCSF Chimera
-c: color shown in UCSF Chimera
-g: grid space. Default value is 1.0 Å

The example commands are as below:
1) Calculate box parameter by using default value. (Note: **the double quotes are necessary for X, Y, Z coordinates. The interval distance between X, Y, Z coordinates should be one space**.)
molaical.exe -tool box -i “52.646 7.634 53.411” -l “30.0 30.0 30.0” -o "D:\chimerabox\box.bild"

2) Calculate box parameter with appointed value
molaical.exe -tool box -i “52.646 7.634 53.411” -l “40.0 30.0 30.0” -t 0.5 -c red -g 1.0 -o "D:\box.bild"

3) This command emphasizes the importance of double quotes. The minus sign must be enclosed by double quote.
molaical.exe -tool box -i "-30.011 1.665 -36.581" -l "30.0 30.0 30.0" -o "D:\box.bild"

4.4.5. Binding free energy from pKd or pKi

To train the Vinardo score, it need calculate the binding free energy according to the pKd or pKi values from PDBbind database. Here, the MolAICal provides an easy way to calculate binding free energy if the value of pKd or pKi is given.

Here, the International System of Units (SI) is introduced (see Table 4.4.5.1). Most laboratory and literatures uses mol/dm^3, which is the same with mol/L (also named “M”). For example:

$mol/m^3 = 10^{-3}$ $mol/dm^3 = 10^{-3}$ mol/L = 10^{-3} M = 1 mmol/L = 1 mM.

Table 4.4.5.1 Molar concentration units

Name	Abbreviation	Concentration	Concentration (SI unit)
millimolar	mM	10^{-3} mol/L	10^{0} mol/m^3
micromolar	μM	10^{-6} mol/L	10^{-3} mol/m^3
nanomolar	nM	10^{-9} mol/L	10^{-6} mol/m^3
picomolar	pM	10^{-12} mol/L	10^{-9} mol/m^3
femtomolar	fM	10^{-15} mol/L	10^{-12} mol/m^3
attomolar	aM	10^{-18} mol/L	10^{-15} mol/m^3
zeptomolar	zM	10^{-21} mol/L	10^{-18} mol/m^3
yoctomolar	yM	10^{-24} mol/L (6 particles per 10 L)	10^{-21} mol/m^3

The "millimolar" and "micromolar" refer to mM and μM (10^{-3} mol/L and 10^{-6} mol/L), respectively. About the detail relative information of molar concentration units, please see the website: https://en.wikipedia.org/wiki/Molar_concentration

The parameters for binding free energy calculation in MolAICal are shown as below:

-tool: appoint tool function. Here, it is “pkdpki”
-i: input value
-t: type of calculate way. It can choose “pkx” (means pKd or pKi) or “molar”
-u: unit of molar. It can choose values from the second column values of Table 4.4.5.1.
-k: temperature. The default temperature is 298.15 K

1) Calculate binding free energy from pkd (pkd = -logKd or pki = -logKi), use default temperature: 298.15 K
molaical.exe -tool pkdpki -i 11.92 -t pkx

2) Calculate binding free energy from pkd (pkd = -logKd or pki = -logKi), use appointed temperature.
molaical.exe -tool pkdpki -i 11.92 -t pkx -k 300

3) Calculate from Kd or Ki with concentration. Default is M (mol/L)
molaical.exe -tool pkdpki -i 1.2 -t molar

4) Calculate from Kd or Ki with pm unit
molaical.exe -tool pkdpki -i 1.2 -t molar -u pm

5) Calculate from Kd or Ki with pm unit under 300 K
molaical.exe -tool pkdpki -i 1.2 -t molar -u pm -k 300

4.5. Analysis

4.5.1. k-means clustering

The k-means clustering is a popular way to cluster analysis in data mining. The results of drug virtual screening show very similar structures and affinity score. Even though the screened drugs have good binding scores, they maybe have very similar structures and functions. It is difficult to pick up the representative molecule for further drug assay. To solve this problem, the k-means algorithm is used to cluster the results of drug design for further activity assay. The parameters for k-means in MolAICal is:

-tool: appoint tool function. Here, it is “kmeans”
-n: clustering number
-i: the file of principal component
-o: output file of clustering results

The example commands are as below:
1) k-means command for clustering
molaical.exe -tool kmeans -n 3 -i "D:\PC.dat" -o "D:\result.dat"

4.5.2. Similarity search

4.5.2.1 Fingerprint for similarity search

Molecular fingerprints are a way to encode the molecular structure with a series of binary digits (bits) which can be used to compare the similarity between two molecules. The parameters in MolAICal are shown as below:

-tool: appoint tool function. Here, it is “finger”
-i: input file of molecules
-o: calculated output results

The example commands are as below:
1) Fingerprint calculation command
molaical.exe -tool finger -i "D:\workdir\listfiles.dat" -o "D:\workdir\result.dat"

4.5.2.2 3D structures similarity comparison

It can compare 3D molecular similarity between two ligands on basis of four specific points which contain the centroid of the ligands, two atoms are closest to and farthest from the centroid, and one atom is farthest from the previous atom. The detailed algorithm is introduced by Ballester PJ and Richards WG[24]. The parameters in MolAICal are shown as below:

-tool: appoint tool function. Here, it is "3Dcompare"
-i: the first input file, it can be mol2 file or list file contained molecular mol2 name
-s: the second file, it can be mol2 file or output file.
-f: file format, it can be "mol2" or "mol2list". The default value is "mol2".

The example commands are as below:
1) Calculate two molecular similarity with default value.
molaical.exe -tool 3Dcompare -i D:\workdir\MolAICal\example\3Dcompare\ligand.mol2 -s D:\workdir\MolAICal\example\3Dcompare\1_10023_out.mol2

2) Calculate two molecular similarity with mol2 format.
molaical.exe -tool 3Dcompare -i D:\workdir\MolAICal\example\3Dcompare\ligand.mol2 -s D:\workdir\MolAICal\example\3Dcompare\1_10023_out.mol2 -f mol2

3) Calculate molecular similarity from list file. Note: the first file is compared by the remainder following mol2 file. So please put your compared molecular name in the first place of list file.
molaical.exe -tool 3Dcompare -i D:\workdir\MolAICal\example\3Dcompare\list.txt -s D:\workdir\MolAICal\example\3Dcompare\result.dat -f mol2list

4.5.3. Merge and split files

4.5.3.1. Extract data from one column of file

This function services the drug clusters from the file which contains three independent columns. This function can merge two files by column. For example, one file contains affinity score, the other is contains fingerprints values. This function can merge these two files to one file which can be named as PC.dat for k-means clustering. The parameters in MolAICal are shown as below:

-tool: appoint tool function. Here, it is "colth"
-i: path of input file
-o: path of output file
-n: column order. It begins with 0. The first column is 0, and so on.

The example commands are as below:

1) Extract data from the second column of file
molaical.exe -tool colth -i "D:\mergecol\5ee7-final_result.txt" -o "D:\mergecol\ouputfile.dat" -n 1

4.5.3.2. Merge two files

MolAICal supply the function of merging two files. You can merge more files one by one based on the previous merged file. The parameters in MolAICal are shown as below:

-tool: appoint tool function. Here, it is "col"
-f: the path of first input file
-l: the path of second input file
-s: separator for merging two files
-o: path of output merged file

The example commands are as below:
1) command example for merging two files
molaical.exe -tool col -f "D:\name.dat" -l "D:\ Fingerprint.dat" -s " " -o "D:\all.dat"

4.5.3.3. Merge two mol2 files

ZINC database[25] supplies many small molecular sets with mol2 format. Merge and split molecular files with mol2 format are necessary handling for drug design. Here, MolAICal supplies the merge function for dealing with molecular files with mol2 format. The parameters in MolAICal are shown as below:
-tool: appoint tool function. Here, it is "merge"
-m: the format for merging files. Here, it includes mol2 format
-i: path of input directory
-o: path of output file

The example commands are as below:
1) Mol2 format files merging command example
molaical.exe -tool merge -m mol2 -i "D:\merge" -o "D:\merge\all.mol2"

4.5.3.4. Split and number statistics of mol2 file

In some case, it need split the molecular file for drug design. For instance, Autodock vina performs drug virtual screening one by one. It need divide the molecular set into one molecule. The MolAICal software supplies a function for splitting molecular set with mol2 format. The parameters in MolAICal are shown as below:

-tool: appoint tool function. Here, it is “split”
-w: choose functions. It can be “split” or “num”. “split” means file split, while “num” means to count the molecular number in the entire mol2 file.
-n: interval number for mol2 file splitting
-v: whether overlap the existed molecular file.
-i: path of input molecular mol2 file
-o: path of output directory

The example commands are as below:
1) Split molecular set to single molecule
molaical.exe -tool split -w split -n 1 -v true -i "D:\split\all.mol2" -o "D:\split\1"

2) Census the number of molecular set
molaical.exe -tool split -w num -i "D:\split\all.mol2"

4.5.3.5. Change Mol2 to SMILES format

This function just service for changing Mol2 to SMILES format of molecules in the De novo drug design. This function will continue to be developed. The parameters in MolAICal are shown as below:

-tool: appoint tool function. Here, it is “mol2tosmi”
-i: path of molecular input file with mol2 format

The example commands are as below:
1) command example of mol2 to SMILES
molaical.exe -tool mol2tosmi -i "D:/mol2tosmi/zinc_98180786.mol2"

4.5.4. Split molecules into fragments

It will get useful information on drug design through splitting appointed molecules into fragments. For example, the fragments which are split from FDA approved drugs, can give effective small chip for De novo drug design based on fragments growth. In addition, it can find the important rules for drug discovery via analysis of fragment features of appointed molecule set such as natural compounds, marine drugs, and so on. Here, MolAICal supplied one way to split one molecule into fragments by rotation bond. The parameters in MolAICal are shown as below:

-tool: appoint tool function. Here, it is “fragSplit”

-i: path of molecular list file
-o: path of output fragments
-oa: path of output fragments with adding hydrogen
-w: method for dealing with hydrogen. It contains “add”, “del” and “off”. The “add” means the split fragments will add hydrogen and copy it into the “-oa” appointed directory. The “del” means the split fragments will delete hydrogen and copy it into the “-oa” appointed directory. The “off” means the split fragments will not perform any action and copy it into the “-oa” appointed directory.

The example commands are as below:

1) Split molecules into fragments and do not perform any hydrogen operation
molaical.exe -tool fragSplit -i "D:\\fragment\\list.dat" -o "D:\\fragment\\1" -oa "D:\\fragment\\3"

2) Split molecules into fragments and do not perform any hydrogen operation
molaical.exe -tool fragSplit -i "D:\\fragment\\list.dat" -o "D:\\fragment\\1" -oa "D:\\fragment\\3" -w off

3) Split molecules into fragments with adding hydrogen
molaical.exe -tool fragSplit -i "D:\\fragment\\list.dat" -o "D:\\fragment\\1" -oa "D:\\fragment\\3" -w add

4) Split molecules into fragments with deleting hydrogen
molaical.exe -tool fragSplit -i "D:\\fragment\\list.dat" -o "D:\\fragment\\1" -oa "D:\\fragment\\3" -w del

Appendix

1. Configure file of fragments grow using DL model

```
# This is example of configure file with deep learning (DL) model.
# "#" is annotation

centerPoints                -30.011    1.665    -36.581
boxLengthXYZ                30.0 30.0 30.0
receptorPDB                 /home/bqf/create/example/GCGRNoLigand.pdb


# Two ways for defining the frag grow way.
# 1. First way for grow. The growMethod contains fixFrag and randomFrag
growMethod                  fixFrag
startFragFile               /home/bqf/create/example/startFrag.mol2

# 2. Second way for grow. It is suitable for no crystal ligand in the receptor pocket.
# growMethod                randomFrag
# startFragFile             D:/workdir/MolAICal/example/grow/genstartFrag.mol2
# startAtomPosition         D:/workdir/MolAICal/example/grow/resPosition.pdb
# startSmiFrag              C

# define the population
numCycleGen                 24
selTopMols                  200
selTopRootPercent           0.70
```

```
selElitePercent                 0.10
maximumPOP                      3000

# define read library way: mol2, SMILES, AIFrag
libStyle                        AIFrag
genAIWay                        Fdafrag
genAINumber                     81
# obabelPath                    C:\Program Files\OpenBabel-2.4.1\obabel.exe
# Search the conformation algorithm
# Two methods: "random" and "fibonacci"
perturbeSearch                  on
genAlgorithm                    fibonacci
ranGenOptNum                    361

# define mutation way
atomMutation                    off
mutationPercent                 0.50

# grow alogrithm
randomCrossMutation             on
randomCrossRatio                0.50
randomMutationRatio             0.50
randomCrossCompareNum           2
randomCrossRange                3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18
# value on or off
writeResultParallel             on

# filter molecule. Value is on or off.
unwantedFrag                    on
PAINS                           on
growFilter                      on
growFilterRatio                 0.50

# default is 0.50
resultsFilterRatio              0.50

# rules of five (see: J Med Chem. 2019 Feb 28;62(4):1701-1714.)
RulesOfFive                     on
xlogPvalue                      6.0
acceptors                       12
donors                          6
mwvalue                         1000.0
rotatablebonds                  14
```

```
# SA
syntheticAccessibility          off
synAccessibilityValue           50

MolsimilarPercent               0.90

# define output results
# pcClusterdFile                D:/workdir/MolAICal/example/grow/PC.dat
pcClusterNumber                 10
adjustOutSimPer                 0.999999
# moloutputdir                  D:/workdir/MolAICal/example/grow/OX2Orexin/results
selMolMaxWeight                 1000
selMolMinWeight                 100
selBindingScore                 -3

# define cpu and memory
coreNum                         20
```

2. Configure file of fragments grow without DL model

```
# This is example of configure file without deep learning (DL) model.
# "#" is annotation

centerPoints                    -30.011    1.665    -36.581
boxLengthXYZ                    30.0 30.0 30.0
receptorPDB                     /home/bqf/create/example/GCGRNoLigand.pdb

# Two ways for defining the frag grow way.
# 1. First way for grow. The growMethod contains fixFrag and randomFrag
growMethod                      fixFrag
startFragFile                   /home/bqf/create/example/startFrag.mol2

# 2. Second way for grow. It is suitable for no crystal ligand in the receptor pocket.
# growMethod                    randomFrag
# startFragFile                 D:/workdir/MolAICal/example/grow/genstartFrag.mol2
# startAtomPosition             D:/workdir/MolAICal/example/grow/resPosition.pdb
# startSmiFrag                  C

# define the population
numCycleGen                     24
selTopMols                      200
selTopRootPercent               0.70
```

```
selElitePercent                 0.10
maximumPOP                      3000

# define read library way: mol2, SMILES, AIFrag
libStyle                         mol2
genAIWay                         Fdafrag
genAINumber                      81

# Search the conformation algorithm
# Two methods: "random" and "fibonacci"
perturbeSearch                  on
genAlgorithm                    fibonacci
ranGenOptNum                    361

# define mutation way
atomMutation                    off
mutationPercent                 0.50

# grow alogrithm
randomCrossMutation               off
randomCrossRatio                  0.50
randomMutationRatio               0.50
randomCrossCompareNum             2
randomCrossRange                  3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18
# value on or off
writeResultParallel              on

# filter molecule. Value is on or off.
unwantedFrag                    on
PAINS                           on
growFilter                      on
growFilterRatio                 0.50
##default is 0.56
resultsFilterRatio              0.50

# rules of five (see: J Med Chem. 2019 Feb 28;62(4):1701-1714.)
RulesOfFive                     on
xlogPvalue                      6.0
acceptors                       12
donors                          6
mwvalue                         1000.0
rotatablebonds                  14

# SA
```

```
syntheticAccessibility      off
synAccessibilityValue       50

MolsimilarPercent           0.90

# define output results
pcClusterNumber             10
adjustOutSimPer             0.999999

# filters
selMolMaxWeight             1000
selMolMinWeight             100
selBindingScore             -3

# define cpu and memory
coreNum                     20
```

3. Configure file for simple radii measurement

```
pdbpath     D:/GramicidinA/1JNO.pdb
cpoint      0.1625 -0.629 -1.838
vector      0.00 0.00 1.00
sample      0.25
conpar      0.15
endrad      5.0
selatoms    5.1
numpt       300
surColor    blue
lineColor   red
lineWidth   2
searchNum   2000
material    Glass2
```

4. Configure file for radii measurement with CHARMM ff

```
pdbpath     D:/GramicidinA/1JNO.pdb
psfpath     D:/GramicidinA/1JNO.psf
cpoint      0.1625 -0.629 -1.838
vector      0.00 0.00 1.00
sample      0.25
```

conpar	0.15
endrad	5.0
selatoms	5.1
numpt	300
surColor	blue
lineColor	red
lineWidth	2
searchNum	2000
material	Glass2

REFERENCES

1 Kumar, A. & Jha, A. in *Anticandidal Agents* (eds Awanish Kumar & Anubhuti Jha) 63-71 (Academic Press, 2017).

2 Chen, H., Engkvist, O., Wang, Y., Olivecrona, M. & Blaschke, T. The rise of deep learning in drug discovery. *Drug Discov Today* **23**, 1241-1250 (2018).

3 Lipinski, C. A., Lombardo, F., Dominy, B. W. & Feeney, P. J. Experimental and computational approaches to estimate solubility and permeability in drug discovery and development settings. *Adv Drug Deliv Rev* **46**, 3-26 (2001).

4 Dahlin, J. L. *et al.* PAINS in the assay: chemical mechanisms of assay interference and promiscuous enzymatic inhibition observed during a sulfhydryl-scavenging HTS. *J Med Chem* **58**, 2091-2113 (2015).

5 Dana, D. *et al.* Deep Learning in Drug Discovery and Medicine; Scratching the Surface. *Molecules* **23** (2018).

6 De Cao, N. & Kipf, T. MolGAN: An implicit generative model for small molecular graphs. *arXiv preprint arXiv:1805.11973* (2018).

7 Guimaraes, G. L., Sanchez-Lengeling, B., Outeiral, C., Farias, P. L. C. & Aspuru-Guzik, A. Objective-reinforced generative adversarial networks (organ) for sequence generation models. *arXiv preprint arXiv:1705.10843* (2017).

8 Shultz, M. D. Two Decades under the Influence of the Rule of Five and the Changing Properties of Approved Oral Drugs. *J Med Chem* **62**, 1701-1714 (2019).

9 Rester, U. From virtuality to reality - Virtual screening in lead discovery and lead optimization: a medicinal chemistry perspective. *Curr Opin Drug Discov Devel* **11**, 559-568 (2008).

10 Trott, O. & Olson, A. J. AutoDock Vina: improving the speed and accuracy of docking with a new scoring function, efficient optimization, and multithreading. *J Comput Chem* **31**, 455-461 (2010).

11 Quiroga, R. & Villarreal, M. A. Vinardo: A Scoring Function Based on Autodock Vina Improves Scoring, Docking, and Virtual Screening. *PLoS One* **11**, e0155183 (2016).

12 Hou, T., Wang, J., Li, Y. & Wang, W. Assessing the performance of the MM/PBSA and MM/GBSA methods. 1. The accuracy of binding free energy calculations based on molecular dynamics simulations. *J Chem Inf Model* **51**, 69-82 (2011).

13 Hou, T., Wang, J., Li, Y. & Wang, W. Assessing the performance of the molecular mechanics/Poisson Boltzmann surface area and molecular mechanics/generalized Born surface area methods. II. The accuracy of ranking poses generated from docking. *J Comput Chem* **32**, 866-877 (2011).

14 Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H. & Teller, E. Equation of state calculations by fast computing machines. *The journal of chemical physics* **21**, 1087-1092 (1953).

15 Bai, Q. *et al.* Ligand induced change of beta2 adrenergic receptor from active to inactive conformation and its implication for the closed/open state of the water channel: insight from molecular dynamics simulation, free energy calculation and Markov state model analysis. *Phys Chem Chem Phys* **16**, 15874-15885 (2014).

16 Humphrey, W., Dalke, A. & Schulten, K. VMD: visual molecular dynamics. *J Mol Graph* **14**, 33-38, 27-38 (1996).

17 Baell, J. & Walters, M. A. Chemistry: Chemical con artists foil drug discovery. *Nature* **513**, 481-483 (2014).

18 O'Boyle, N. M. *et al.* Open Babel: An open chemical toolbox. *J Cheminform* **3**, 33 (2011).

19 Ertl, P. & Schuffenhauer, A. Estimation of synthetic accessibility score of drug-like molecules based on molecular complexity and fragment contributions. *J Cheminform* **1**, 8 (2009).

20 Wang, R., Fang, X., Lu, Y., Yang, C. Y. & Wang, S. The PDBbind database: methodologies and updates. *J Med Chem* **48**, 4111-4119 (2005).

21 Wang, R., Fang, X., Lu, Y. & Wang, S. The PDBbind database: collection of binding affinities for protein-ligand complexes with known three-dimensional structures. *J Med Chem* **47**, 2977-2980 (2004).

22 Li, Y. *et al.* Assessing protein-ligand interaction scoring functions with the CASF-2013 benchmark. *Nat Protoc* **13**, 666-680 (2018).

23 Pettersen, E. F. *et al.* UCSF Chimera--a visualization system for exploratory research and analysis. *J Comput Chem* **25**, 1605-1612 (2004).

24 Ballester, P. J. & Richards, W. G. Ultrafast shape recognition to search compound databases for similar molecular shapes. *J Comput Chem* **28**, 1711-1723 (2007).

25 Irwin, J. J. & Shoichet, B. K. ZINC--a free database of commercially available compounds for virtual screening. *J Chem Inf Model* **45**, 177-182 (2005).