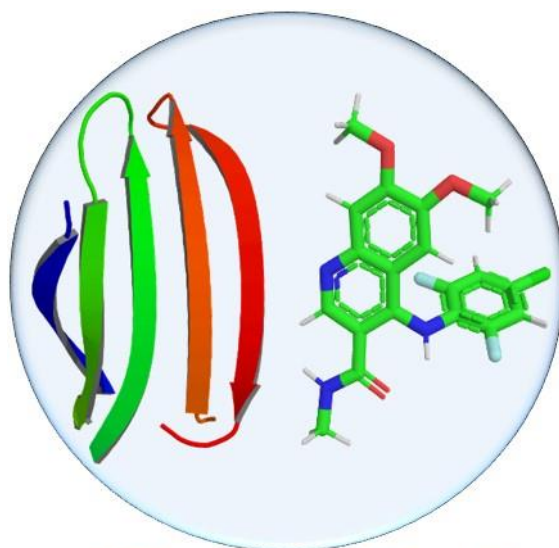


MolAICal

A drug design software combined by artificial intelligence and
classical programming

Manual



MolAICal

V1.1.2

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

Official site mirror in China: <https://molaical.gitee.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

Part I. Classical Algorithm	4
1. General information of MolAICal	4
1.1. Introduction	4
1.2. Installation	5
1.3. How to cite MolAICal	6
2. <i>De novo</i> drug design by MolAICal and DL	6
2.1. Theory	6
2.2. Parameters in configure file	7
2.3. The commands for fragments grow	15
3. Molecular docking, virtual screening and deep learning model	16
3.1. Molecular docking	16
3.1.1. MolAICaID introduction	16
3.1.2. Prepare files for MolAICal docking	16
3.1.3. Molecular docking parameters	17
3.2. Drug virtual screening by MolAICal, DL model and MolAICaID	18
3.2.1. Theory	18
3.2.2. Introduction of parameters of MolAICaID configure file	19
3.2.3. Command introduction for AI virtual screening	20
3.2.4. Command introduction for virtual screening	21
3.3. MM/GBSA calculation	22
3.3.1. MM/GBSA calculation based on NAMD	22
3.3.2. Entropy calculation based on NAMD	23
4. Useful tools for drug design	24
4.1. Channel radii measurement	24
4.2. Potential of mean force	26
4.3. Molecular properties calculation	27
4.3.1. Quantitative structure-activity relationship (QSAR)	27
4.3.2. Lipinski's rule of five calculation	29
4.3.3. PAINS calculation	31
4.3.4. Synthetic Accessibility calculation	31
4.4. Vinardo score calculation	32
4.4.1. Batch of Vinardo score calculation	32
4.4.2. Vinardo score Decomposition	33
4.4.3. Grid file generation for score calculation	34
4.4.4. Box generation based on receptor	34
4.4.5. Binding free energy from pKd or pKi	35
4.5. Molecular descriptor	36
5. Analysis for drug design	37
5.1. k-means clustering	37
5.2. Similarity search	38
5.2.1. Fingerprint for similarity search	38
5.2.2. 3D structures similarity comparison	38

5.3. Merge and split files	39
5.3.1. Extract data from one column of file	39
5.3.2. Merge column of two files	39
5.3.3. Merge two files	40
5.3.4. Split and number statistics of mol2 file	40
5.3.5. Molecular format conversion	41
5.3.5.1. Change Mol2 to SMILES format.....	41
5.3.5.2. Conversion of different molecular formats.....	41
5.3.6. Split and extract PDBQT file	42
5.4. Split molecules into fragments	43
Part II. Artificial intelligence	44
1. Protein fold.....	45
2. Deep learning models of Binding affinity	45
2.1. Binding affinity prediction by OnionNet model.....	45
2.2. Binding affinity prediction by Pafnucy model	46
2.3. Molecular generation by deep learning model.....	47
2.3.1 Molecular generation for drug-like molecules	47
2.3.2 Generating similar ligands based on a known ligand	48
Appendix.....	48
1. Configure file of fragments growth using DL model.....	48
2. Configure file of fragments growth without DL model.....	50
3. Configure file for simple radii measurement	52
4. Configure file for radii measurement with CHARMM ff.....	53
REFERENCES	53

Part I. Classical Algorithm

1. General information of MolAICal

1.1. Introduction

Computer-aided drug design (CADD) employs computational strategies to screen, develop, modify, analyze and design the biologically active compounds¹. With the development of technology, artificial intelligence methods such as deep learning emerge into the CADD research fields. The CADD and deep learning have been widely used in *de novo* drug design, bioactivity prediction, biological image analysis, synthesis prediction and so on². The CADD combined with deep learning can save capital and improve the speed of drug discovery.

Here, MolAICal soft package is developed to design drugs by deep learning models, webserver classical algorithms such as genetic algorithm, etc. Some functions of MolAICal are listed as below:

- 1 *De novo* drug design: MolAICal can perform fragment growth on the pocket of the receptor by using the assigned molecular library or the molecular library generated from deep learning models.
- 2 Molecular docking and virtual screening: MolAICal can perform molecular docking and drug virtual screening based on the known ligand database or molecular database generated from deep learning models.
- 3 Predicting binding affinity between receptor and ligand based on NAMD trajectories by MM/GBSA, among others.
- 4 Integrating deep learning models and webserver for drug design that contain molecular generation, drug modification, binding affinity prediction and so on.
- 5 The useful tools for drug discovery
 - 1) Pore radii measure for receptors such as ion channel proteins.
 - 2) Potential of mean force for free energy calculation based on principal components.
 - 3) Quantitative structure-activity relationship (QSAR) for bioactivity prediction of ligands
 - 4) Lipinski's rule of five³ for the calculation of drug-like properties.
 - 5) Pan-assay interference compounds (PAINS)⁴ filter

...among others.

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

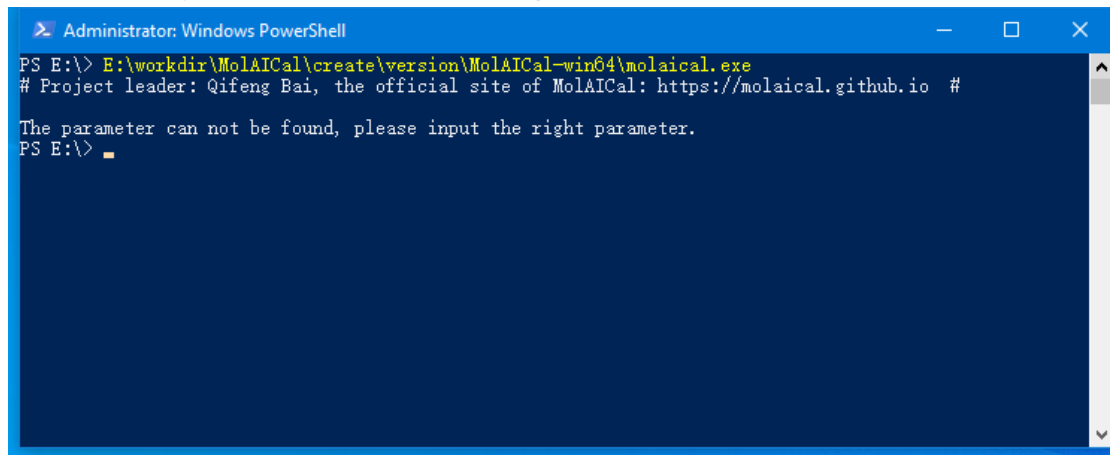
MolAICal: <https://molaical.github.io> or <https://molaical.gitee.io>

1) For Window operating system:

MolAICal need not be installed, just **decompress** the MolAICal file. MolAICal is a command-line tool. Users can use the absolute path of MolAICal on DOS or Powershell of Windows. For example: if MolAICal is placed on the E: disk, then users can run the blow example command (Users should modify it according to the actual path of molaical.exe):

```
#> E:\workdir\MolAICal\create\version\MolAICal-win64\molaical.exe
```

Press **Enter** Key, it should show similar message as follows:



```
Administrator: Windows PowerShell
PS E:\> E:\workdir\MolAICal\create\version\MolAICal-win64\molaical.exe
# Project leader: Qifeng Bai, the official site of MolAICal: https://molaical.github.io #
The parameter can not be found, please input the right parameter.
PS E:\>
```

Until now, install is complete.

2) For Linux operating system: use the below command to install MolAICal:

1. tar -xzf 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

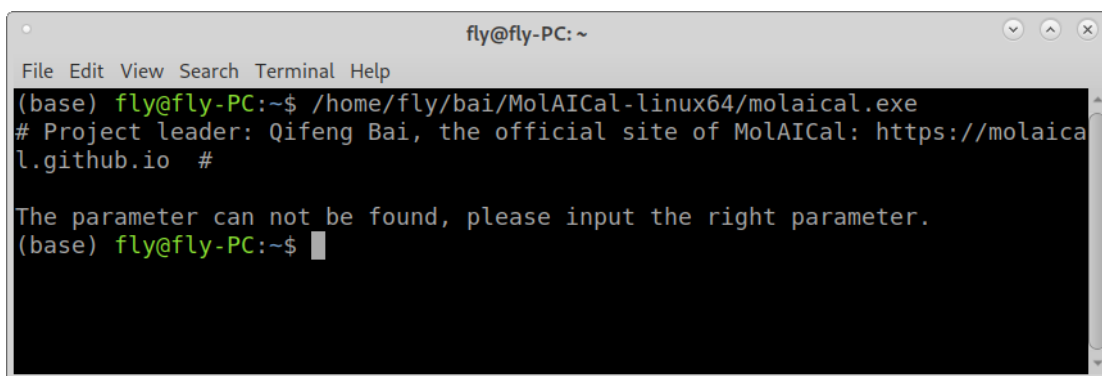
3. chmod +x install.sh

4. ./install.sh

At last, users can run the blow example command in Linux terminal console (Users should modify it according to the actual path of molaical.exe):

```
#> /home/fly/bai/MolAICal-linux64/molaical.exe
```

Press **Enter** Key, it should show similar message as follows:



```
fly@fly-PC: ~
File Edit View Search Terminal Help
(base) fly@fly-PC:~$ /home/fly/bai/MolAICal-linux64/molaical.exe
# Project leader: Qifeng Bai, the official site of MolAICal: https://molaical.github.io #
The parameter can not be found, please input the right parameter.
(base) fly@fly-PC:~$
```

Until now, install is complete.

Notice: MolAICal only provides Windows and Linux 64-bit versions. You should not change the name or path of MolAICal after install is complete in Linux System. If users change the name or path of MolAICal, please re-install MolAICal again. To use MolAICal expediently, users can set the system path of MolAICal. So the command molaical.exe can be used without a given path.

For more detailed tutorials, please go to MolAICal official website: <https://molaical.github.io> or official site mirror in China: <https://molaical.gitee.io>

1.3. How to cite MolAICal

If MolAICal is used in your work, please cite the below two papers:

1. Bai, Q., Research and development of MolAICal for drug design via deep learning and classical programming. *arXiv preprint* 2020, arXiv:2006.09747.
<https://arxiv.org/abs/2006.09747>
2. Bai, Q. et al. MolAICal: a soft tool for 3D drug design of protein targets by artificial intelligence and classical algorithm. *Briefings in bioinformatics* 22, bbaa161 (2021).
<https://doi.org/10.1093/bib/bbaa161>

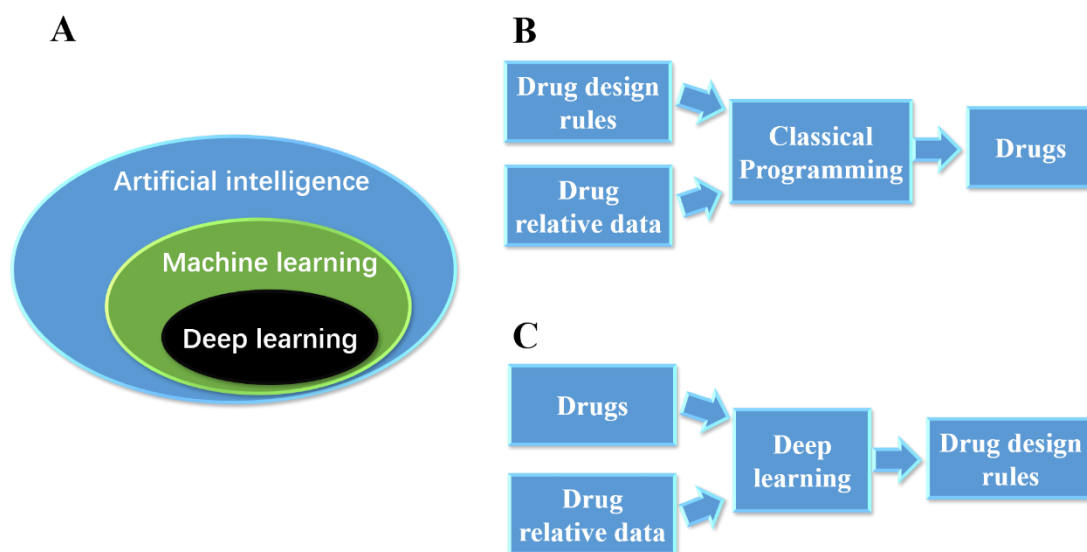
2. *De novo* drug design by MolAICal and DL

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, analyze and design the biologically active drugs. 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, but it can be regarded as the anchored fragment. Then, new compounds can be produced via growing fragments based on this anchored fragment 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 Part I Figure 2.1A). Deep learning has been successfully used in drug discovery² and medicine⁵. It can train the molecular generative model that 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 Part I 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 Part I 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 programming 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 that is based on the given small fragments.



Part I 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 the appointed box.

➤ **boxLengthXYZ** < box length >

Acceptable Values: decimal

Default Value: null

Description: It is box lengths 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 the initial molecular fragment are fixed, and this molecular fragment may be extracted from the ligand in the crystal receptor-ligand complex. The “randomFrag” means the positions of the 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 the initial fragment in your operation system. The initial fragment is Sybyl mol2 format.

➤ **startSmiFrag** < Initial SMILES fragment >

Acceptable Values: String

Default Value: None

Description: If the growMethod is “randomFrag”, this parameter should be set. If the value of **startSmiFrag** is the Canonical SMILES string such as C, the Canonical SMILES string will be converted to the 3D seed file which name is appointed by parameter **startFragFile**. If the value of **startSmiFrag** is equal to “null” or “off”, it means the user-defined fragment in mol2 format is employed as the initial seed. In this case, the users can set their own initial fragment file in the part of parameter **startFragFile**.

➤ **gSeedKValue** < control constant >

Acceptable Values: decimal

Default Value: 0.25

Description: If the growMethod is “randomFrag”, this parameter can be selected to set. This is control constant. A small value of gSeedKValue can reduce the searching range around the assigned

point, and vice versa.

➤ **gSeedSearchNum** < Searching number >

Acceptable Values: Integer

Default Value: 2000

Description: If the growMethod is “randomFrag”, this parameter can be selected to set. This parameter can set the searching number around the assigned point.

➤ **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 in 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 population.

➤ **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 molecules during the storing process of the temporal file.

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

Acceptable Values: decimal

Default Value: null

Description: This ratio is used to filter overlap molecules 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^{8,9}, 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 are 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: *Deprecated*

Default Value: null

Description: Deprecated. This function is currently useless in this version. It can be omitted when the parameters are assigned.

➤ **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, value is “on” or “off”, default value is “on”.

-l: whether perform fragment grow, value is “on” or “off”, default value is “on”.

-o: whether output results, value 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 the 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. Molecular docking, virtual screening and deep learning model

3.1. Molecular docking

3.1.1. MolAICaID introduction

MolAICaID which is derived from Autodock Vina¹⁰ is packaged into MolAICal soft package. MolAICaID use the same license of Autodock Vina (APACHE LICENSE, VERSION 2.0 <https://www.apache.org/licenses/LICENSE-2.0>). It is responsible for molecular docking function in MolAICal software package. MolAICaID is compiled based on boost_1_72_0. The 3130 complexes of proteins-ligands is chosen for assaying the ‘docking’ and ‘ranking’ power of MolAICaID from the refined-set of PDBbind database¹¹. 20 docking poses of ligand are generated on the receptor pocket in every extracted complex of PDBbind database. The generated ligand which has the lowest RMSD with original ligand is considered to have the best binding affinity among the 20 docking poses. In addition, the Pearson and Spearman correlation coefficients are compared between the experimental binding affinity and predicted binding affinity which is from the ligand with the lowest RMSD in every extracted complex. For Autodock Vina, the accuracy rate of docking is 54.665% on the 3130 complexes if the ligand with lowest RMSD has the lowest binding affinity value among the 20 generated ligands. And the Pearson and Spearman correlation coefficients are 0.5259 and 0.5421 based on the experimental binding affinity. (The default weight values in Autodock Vina: “weight_gauss1” is: -0.035579; “weight_gauss2” is: -0.005156; “weight_repulsion” is: 0.84024500000000002; “weight_hydrophobic” is: -0.035069000000000003; “weight_hydrogen” is: -0.587439000000000004). For MolAICaID in MolAICal, the accuracy rate of docking is 54.888% on the 3130 complexes if the ligand with the lowest RMSD has the lowest binding affinity value among the 20 generated ligands. And the Pearson and Spearman correlation coefficients are 0.5335 and 0.5489 based on the experimental binding affinity. It indicates that MolAICal has better ‘docking’ and ‘ranking’ power than Autodock Vina.

3.1.2. Prepare files for MolAICal docking

The PDBQT file format is necessary for molecular docking by MolAICal. MolAICal supplies the below commands for files preparation via referring to MGLTools¹².

1) Prepare the receptor file. It will generate PDBQT file with the same name of receptor file if the suffix is not counted. The command is shown as below:

```
#> molaical.exe -dock receptor -i protein.pdb
```

2) Prepare the ligand file. It will generate PDBQT file with the same name of ligand file if the suffix

is not counted. The command is shown as below:

```
#> molaical.exe -dock ligand -i ligand.pdb
```

3) Adding hydrogen on molecular file. The input file can be in PDB, PDBQ, PDBQT, SYBYL mol2 or PQR format. The output file is in PDBQT format. The command is shown as below:

```
#> molaical.exe -dock addh -i ligand.pdbqt
```

4) Transform PDBQT format into PDB format. The command is shown as below:

```
#> molaical.exe -dock pdbqt2pdb -i ligand.pdbqt
```

5) Calculate two PDBQT format files in the same atom order. The command is shown as below:

```
#> molaical.exe -dock rmsd -f 1.pdbqt -s 2.pdbqt
```

Note: -f corresponds the first file; -s corresponds the second file.

3.1.3. Molecular docking parameters

MolAICaID input parameters are as below:

--receptor arg:	rigid part of the receptor (PDBQT)
--flex arg:	flexible side chains, if any (PDBQT)
--ligand arg:	ligand (PDBQT)
--center_x arg:	X coordinate of the center
--center_y arg:	Y coordinate of the center
--center_z arg:	Z coordinate of the center
--size_x arg:	size in the X dimension (Angstroms)
--size_y arg:	size in the Y dimension (Angstroms)
--size_z arg:	size in the Z dimension (Angstroms)
--out arg:	output models (PDBQT), the default is chosen based on the ligand file name
--log arg:	optionally, write log file
--cpu arg:	the number of CPUs to use
--seed arg:	explicit random seed
--exhaustiveness arg (=8):	exhaustiveness of the global search (roughly proportional to time): 1+
--num_modes arg (=9):	maximum number of binding modes to generate
--energy_range arg (=3):	maximum energy difference between the best binding mode and the worst one displayed (kcal/mol)
--config arg:	the above options can be put here
--help:	display usage summary
--help_advanced:	display usage summary with advanced options
--version:	display program version

It can put above parameters into a file without prefix "--". For example, the content of the configure file "conf.txt" can be as below:

```
receptor = pro.pdbqt  
out = all.pdbqt
```

```
cpu = 1  
center_x = -10.733  
center_y = 12.416  
center_z = 68.829
```

```
size_x = 25  
size_y = 30  
size_z = 25
```

```
num_modes = 1
```

Then use command for molecular docking:

```
#> molaicald --config conf.txt
```

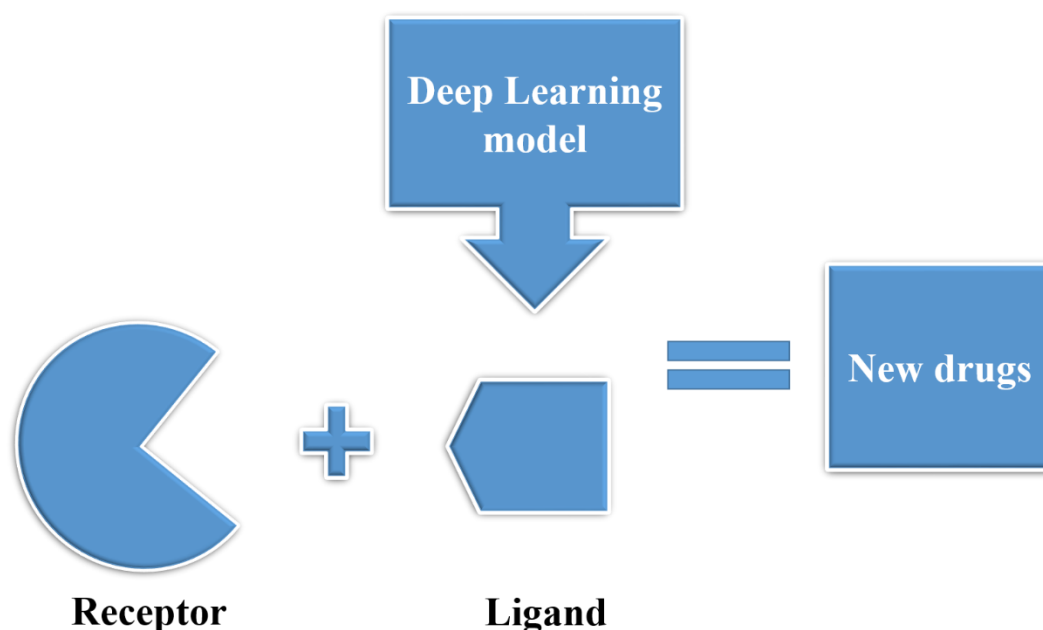
3.2. Drug virtual screening by MolAICal, DL model and MolAICalD

3.2.1. 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 that most likely bind to the drug target, typically protein receptor or DNA, etc. The VS contains ligand-based and structure-based virtual screening¹³. Especially, the structure-based virtual screening involves molecular docking that 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 Part I Figure 3.1.1). The Autodock Vina¹⁰ is an excellent open source molecular docking software that has been studied and applied to the field of drug discovery. MolAICalD is a branch of Autodock Vina that has improved the docking and ranking powers for drug design. 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 score¹⁴ is a variant of score function of Autodock Vina which can improve the prediction of binding ability between ligands and receptors.

Deep learning which employs 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 discovery² and medicine⁵. 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, rational drug design needs to 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, molecular docking supplies another way to take advantage of the deep learning model and classical programming. The strategies for virtual screening are that MolAICal invokes the deep learning model to generate the appointed number of drugs, and then call the molecular docking module MolAICalD of MolAICal to perform the molecular docking to evaluate the binding ability between ligand and receptor (see Part I Figure 3.1.1). Besides, MolAICal can also utilize the appointed molecular database for drug virtual screening through invoking molecular docking program.



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

3.2.2. Introduction of parameters of MolAICalD 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 is deprecated. Please run this process on 64 bit operating system.). Before you run this drug virtual screening, please prepare the necessary MolAICalD 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 CPU core for running, the “center_x”, “center_y” and “center_z” are the x, y, z coordinates of box center you set, the “size_x”, “size_y” and “size_z” are lengths along x, y, z axis of box you set, the “num_modes” is the number of generated conformations by MolAICalD.

3.2.3. Command introduction for AI virtual screening

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

- dock:** value is AI, which means performing drug virtual screening by AI.
- 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:** the generated file name by AI module.
- nc:** number of CPU cores 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:** the configure file for molecular docking VS in the appointed directory. The default value is: “conf.txt”.

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

1) The simple command, the omitted parameters use the default values

```
#> molaical.exe -dock AI -s FDAMol -n 30 -nf 10 -nc 3
```

2) Run full drug virtual screening process by MolAICal, deep learning model, and MolAICalD

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

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

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

6) Command for virtual screening with the assigned SMILES generated file “genvs.txt” by calling deep learning module

```
#> molaical.exe -dock AI -s FDAMol -n 30 -p genvs.txt -nf 10 -nc 3 -g on -b on -v on
```

7) Command contained full parameters with the appointed docking configure file “conf2.dat”

```
#> molaical.exe -dock AI -s ZINCMol -n 30 -p genvs.txt -nf 10 -nc 3 -g on -b on -v on -m conf2.dat
```

8) This command have no drug virtual screening task

```
#> molaical.exe -dock AI -s FDAMol -n 30 -p genvs.txt -nf 10 -nc 3 -g on -b on -v off -m conf2.dat
```

3.2.4. Command introduction for virtual screening

This command is for virtual screening (VS) based on the known database simply via MolAICaID. The command parameters of MolAICaI for drug virtual screening are as below:

-dock: value is vs, means performing drug virtual screening.

-i: input directory which contains the molecules files. The format of molecule in the directory can be in PDB, PDBQ, PDBQT, SYBYL PQR or mol2 format.

-k: the files contain the keyword for VS. The default value is “.mol2”.

-nc: number of CPU cores for running drug virtual screening.

-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: the configure file for molecular docking VS in the appointed directory. The default value is: “conf.txt”.

Before run VS process, you should split files into directories, for instance:

```
#> molaical.exe -tool mol2 -w split -n 1 -v true -m number -i "E:\all.mol2" -o "E:\split"
```

Note: You can use the second line string of mol2 file as the file name by using the below command:

```
#> molaical.exe -tool split -w split -n 1 -v true -i "D:\aa.mol2" -o "D:\split"
```

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

1) The simple command, the omitted parameters use the default values.

```
#> molaical.exe -dock vs -i D:\split
```

2) The simple command, the docked molecular format and running CPU cores are appointed.

```
#> molaical.exe -dock vs -i D:\split -k .mol2 -nc 3
```

3) Run full drug virtual screening process.

```
#> molaical.exe -dock vs -i D:\split -k .mol2 -b on -v on -m conf2.dat -nc 3
```

4) Run drug virtual screening process without molecular format conversion.

```
#> molaical.exe -dock vs -i D:\split -k .mol2 -b off -v on -m "conf2.dat" -nc 3
```

5) Run drug virtual screening process by using molecular PDBQT format directly.

```
#> molaical.exe -dock vs -i D:\split -k .pdbqt -b off -v on -nc 3
```

3.3. MM/GBSA calculation

3.3.1. MM/GBSA calculation based on NAMD

Molecular mechanics generalized Born surface area (MM/GBSA)^{15,16} 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:

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

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

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

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 $\Delta E_{\text{internal}}$ of bond, angle, and dihedral energies. The unit of ΔG_{bind} is kcal/mol. 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 Part I 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

```

Part I Figure 3.2.1 The detail information of NAMD running log file

The parameters for calculating binding free energy are 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 the ligand.

Example command line as below:

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

3.3.2. Entropy calculation based on NAMD

To calculate the entropy of protein-protein, protein-ligand, etc, the Carma¹⁷ software is introduced to calculate the entropy contribution between two molecules based on the NAMD trajectories. Carma computes the entropy using both Schlitter's and Andricioaei's formulas¹⁸ starting from DCD and PSF files. A more detailed introduction about Carma can be found on the official website of Carma: <https://utopia.duth.gr/~glykos/Carma.html>. You should download the new version of Carma (For example: "carma64") for calculating the entropy from <http://utopia.duth.gr/~glykos/progs>. Here, MolAICal supplies a tool for calculate the entropy between two molecules based on the results of Carma. The command is as follows:

```
#> molaical.exe -entropy -c 15522.260938 -r 15211.880284 -l 1454.409253 -t 310.0
```

- entropy**: means delta entropy calculation
- c**: the entropy of complex
- r**: the entropy of the receptor or first molecule
- l**: the entropy of the ligand or second molecule

-t: the Kelvin temperature

4. Useful tools for drug design

4.1. Channel radii measurement

The Metropolis Monte Carlo simulated annealing method¹⁹ 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 the channel. The calculation of channel radii is realized in MolAICal. The examples of configure files for radii calculation shows in Appendix 3 and 4. The parameters for channel radii calculation are listed as below:

➤ **pdopath** < 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 that 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: Value calculated from the minimum radius

Description: It is the minimum explored detector. The MolAICal program will calculate radii of protein, nanotube, etc. by the minimum explored detector.

➤ **conpar** < control constant >

Acceptable Values: decimal

Default Value: 0.15

Description: It is control constant. The detail parameter is shown in equation 4 in this paper²⁰. Usually, it uses the default value of 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 a large system, it needs to 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 the channel in the VMD software²¹. 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 the 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 the color values of VMD software.

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

Acceptable Values: integer

Default Value: null

Description: It represents the width of the curve. The “lineColor” is responsible for the 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 the 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 the 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: the 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).

$$\Delta G = -k_B T \ln \rho(x, y)$$

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 grids 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 the 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 running QSAR are 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: $y = a*x_1 + b*x_2 + c*x_3$
- 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
```

The default output file is named “QSAROutFile.dat” which content is interpreted as follows:

Q²-LOO: Leave-one-out cross validation. This algorithm is applied once for each instance, choosing all other instances as a training set and using the selected instance as a single-item test set. (see: https://doi.org/10.1007/978-0-387-30164-8_469)

R² fitting: Correlation coefficients between experimental and predicted values (see below equation PI-4.3.1.1)

$$R^2 = 1 - \frac{SS_{\text{res}}}{SS_{\text{tot}}} = 1 - \frac{\sum_i (y_i - f_i)^2}{\sum_i (y_i - \bar{y})^2} \quad (\text{PI} - 4.3.1.1)$$

Where y_i , f_i , and \bar{y} are experimental, predicted and average values, respectively. SS_{res} is sum of squares of residuals, also called the residual sum of squares. SS_{tot} is total sum of squares.

R² adjusted: is also called Adjusted R-Squared. The adjusted R² is defined as equation PI-4.3.1.2:

$$adjusted\ R^2 = 1 - \frac{PRESS/df_e}{SS_{tot}/df_t} \quad (\text{PI} - 4.3.1.2)$$

Where df_e is the degrees of freedom $n - p - 1$ of the estimate of the underlying population error variance, and df_t is the degrees of freedom $n - 1$ of the estimate of the population variance of the dependent variable.

RSS: residual sum of squares (see equation PI-4.3.1.3)

$$RSS = \sum_i (y_i - f_i)^2 \quad (\text{PI} - 4.3.1.3)$$

PRESS: predictive residual sum of squares that is a form of cross-validation used in regression analysis to provide a summary measure of the fit of a model to a sample of observations that were not themselves used to estimate the model (https://en.wikipedia.org/wiki/PRESS_statistic) (see equation PI-4.3.1.4).

$$PRESS = \sum_{i=1}^n (y_i - f_{i/i})^2 \quad (\text{PI} - 4.3.1.4)$$

SDEC: standard deviation error in calculation (see equation PI-4.3.1.5).

$$SDEC = \sqrt{\frac{RSS}{n}} \quad (\text{PI} - 4.3.1.5)$$

SDEP: standard deviation error in prediction (see equation PI-4.3.1.6).

$$SDEP = \sqrt{\frac{PRESS}{n}} \quad (\text{PI} - 4.3.1.6)$$

For more detailed knowledge and equations, please read the related content in Wikipedia: https://en.wikipedia.org/wiki/Coefficient_of_determination

external Q2: the explained variance in prediction (see equation PI-4.3.1.7).

$$Q_{ext}^2 = 1 - \frac{\sum_{i=1}^{n_{ext}} (y_i - f_i)^2}{\sum_{i=1}^{n_{ext}} (y_i - \bar{y})^2} \quad (\text{PI} - 4.3.1.7)$$

Where y_i , and f_i , are experimental and predicted values in the test set, respectively. \bar{y} is the average value of train set data. If there are outliers, external Q2 will become a negative value. So users can only refer to external Q2, ignoring some abnormal values, or select other suitable algorithms and models for specific purposes.

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 drug³. 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 statistical 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 ligands, the author can set suitable parameters according to the studied need. MolAICal supplies an easy way to set rules parameters.

Notice: please distinguish the molecular weight and molecular mass. Please see the detailed information about them: https://en.wikipedia.org/wiki/Molecular_mass. The MolAICal outputs the molecular mass of ligands.

-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 assay⁴. PAINS tend to react with numerous biological receptors nonspecifically rather than binding the desired target specifically²². Some disruptive functional groups are shared by many PAINS. The Common PAINS contains “toxoflavin, isothiazolones, curcumin, hydroxyphenyl hydrazones, phenol-sulfonamides, enones, rhodanines, catechols, and quianones”²². 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 Babel²³ to 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)²⁴ 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"
```

Note: the higher value of SA indicates the compound is easier to be synthesized.

4.4. Vinardo score calculation

Vinardo score which is based on Vina score function¹⁴, is trained on basis of PDBbind database^{11,25,26}. The score function of Vina is show as below. The Vinardo score can be trained by combined with the below equations.

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

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

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

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

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

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 Gauss₁ score, Gauss₂ 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 Gauss₁ 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 Gauss₂ 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 Gauss₁ score, Gauss₂ 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 Gauss₁ score

```
#> molaical.exe -tool grid -i "D:\1a1e\boxPar.dat" -p "D:\1a1e\protein2.pdb" -n gaussScore1
```

2) Grid file generation of Gauss₂ 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 file generation needs the box parameters which are extracted on the basis of receptor. MolAICal supplies a function to generate box parameters that can display in UCSF Chimera²⁷. The parameters are 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 the box parameter by using the 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 needs to calculate the binding free energy according to the Ki, Kd, pKd or pKi values from PDDBind database^{25,28,29}. Here, the MolAICal provides an easy way to calculate binding free energy if the value of Ki, Kd, 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/dm³, which is the same with mol/L (also named “M”). For example:

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

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	10 ⁻²¹ mol/m ³

		(6 particles per 10 L)	
--	--	------------------------	--

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 binding free energy is calculated as the below equation:

$$\text{Binding free energy} = RT * \log_e(10^{-\text{pK}_x})$$

where T and R are the temperature and gas constant, respectively.

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 calculating way. It can choose "pkx" (means pKd or pKi) or "molar" (means Ki or Kd).

-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. Molecular descriptor

DRAGON is used with commercial license. To let the researcher study freely and handily. The PaDEL-Descriptor³⁰ and Mordred³¹ are introduced to calculate descriptors. PaDEL-Descriptor which contains 12 types of fingerprints (total 16092 bits) and 1875 descriptors (1444 1D, 2D descriptors and 431 3D descriptors) is free for all (e.g. personal, academic, non-profit, non-

commercial, government, commercial, etc) to use. Mordred which contains 1826 descriptors (Copyright (c) 2015-2017, Hirotomo Moriwaki) uses the BSD 3-Clause "New" or "Revised" License (see: <https://github.com/mordred-descriptor/mordred/blob/develop/LICENSE>). Based on the functions of PaDEL-Descriptor and Mordred, MolAICal supplies molecules in SDF molecular format for descriptor calculation. The Mordred is used to calculate 3D molecular descriptors via converting SMILES string to 3D conformational structure based on MMFF94 force field³² in MolAICal. The command for descriptors calculation is below:

1) Calculating the molecular descriptors based on Mordred. It will out the 2D descriptor file named "without3D-descriptors.csv" and 3D descriptor file named "with3D-descriptors.csv".

```
#> molaical.exe -tool mordred -i E:\006-QSAR\mordred\example.smi
```

2) Calculating the molecular descriptors based on SDF format via PaDEL-Descriptor. Please input the directory that only contains files in SDF format. It will out the 2D descriptor file named "2DDescriptor_mdl.csv" and 3D descriptor file named "3DDescriptor_mdl.csv".

```
#> molaical.exe -tool padel -f sdf -i E:\006-QSAR\PaDEL\sdf
```

3) Calculating the molecular descriptors based on MDL format via PaDEL-Descriptor. Please input the directory that only contains files in MDL format. It will out the 2D descriptor file named "2DDescriptor_mdl.csv" and 3D descriptor file named "3DDescriptor_mdl.csv".

```
#> molaical.exe -tool padel -f mdl -i E:\006-QSAR\PaDEL\mdl
```

4) Calculating the molecular descriptors based on SMILES format via PaDEL-Descriptor. Please input the directory that only contains file in SMILES format. It will out the 2D descriptor file named "2DDescriptor_2Dstructure.csv" based on SMILES strings and 2D descriptor file named "2DDescriptor_3Dstructure.csv" based on the converted 3D structures from SMILES strings.

```
#> molaical.exe -tool padel -f smi -i E:\006-QSAR\PaDEL\smi
```

5. Analysis for drug design

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"
```

5.2. Similarity search

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"
```

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 WG³³. 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 a list file. **Note:** the first line in file is compared by the remainder lines in file. So please put your compared molecular name in the first place of the file.

```
#> molaical.exe -tool 3Dcompare -i D:\workdir\MolAICal\example\3Dcompare\list.txt -s  
D:\workdir\MolAICal\example\3Dcompare\result.dat -f mol2list
```

5.3. Merge and split files

5.3.1. Extract data from one column of file

This function can extract one column data from a file. If the users want to extract the some appointed columns data from a file, the users can use this function to extract the column data one by one, and then, employ the function of MolAICal to merge these extracted files by column. 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:\mergocol\5ee7-final_result.txt" -o "D:\mergocol\ouputfile.dat" -n  
1
```

5.3.2. Merge column of two files

MolAICal supplies the function to merge two files by column. For example, one file contains affinity scores, the other is contains fingerprints values. This function can merge these two files to one file that can be named as “PC.dat” for k-means clustering. 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 column of 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"
```

5.3.3. Merge two files

ZINC database³⁴ 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: appointed keyword. MolAICal will merge the file whose name contains this keyword.

-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"
```

2) PDBQT format files merging command example

```
#> molaical.exe -tool merge -m PDBQT -i "D:\merge" -o "D:\merge\all.pdbqt"
```

5.3.4. Split and number statistics of mol2 file

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

-tool: appoint tool function. Here, it is “mol2”

-w: choose functions. It can be “split”, “chunk” or “num”. “split” means file split, “chunk” means to split the mol2 file to small chunk, “num” means to count the molecular number in the entire mol2 file.

-n: when choosing parameter “chunk”, it means the appointed number of molecules in one split chunk file. When choosing parameter “num”, it means the interval number for mol2 file splitting. The default value is 1.

-m: It can be “number” or “name”. If the value is “number”, it will split the molecular set into multiple molecular files named with ordered numbers, e.g. 1.mol2, 2.mol2 and so on. If the value is “name”, it will split the molecular set into multiple molecular files named with the string that under the line “@<TRIPOS>MOLECULE” in mol2 format file. The default value is “name”.

-v: whether overlap the existed molecular file. The default value is true for overlap.
-i: path of input molecular mol2 file
-o: path of output directory

The example commands are as below:

1) Split molecular set to multiple single molecules named by the string that under the line “@<TRIPOS>MOLECULE” in mol2 file:

```
#> molaical.exe -tool mol2 -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 mol2 -w num -i "D:\split\all.mol2"
```

3) Splits mol2 file to the small chunk which contains 20 molecules:

```
#> molaical.exe -tool mol2 -w chunk -n 20 -i "D:\split\all.mol2" -o "D:\split\2"
```

4) Split molecular set to multiple single molecules named by the ordered numbers:

```
#> molaical.exe -tool mol2 -w split -n 1 -v true -m number -i "D:\split\all.mol2" -o "D:\split\1"
```

5.3.5. Molecular format conversion

5.3.5.1. Change Mol2 to SMILES format

This function just services for changing Mol2 to SMILES format of molecules in *de novo* drug design. The generated SMILES can be compatible read to filtering function of MolAICal. For instance, it can generate SMILES strings from mol2 format for filtering unwanted molecules (see folder: MolAICal-xxx/filterMols). 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"
```

5.3.5.2. Conversion of different molecular formats

This function which is derived from Open Babel²³ can be used to convert different molecular formats such as pdb-mol2, sdf-pdb, sdf-mol2, etc. If you want to convert pdb to mol2 format, you must have the correct suffix. For example, if input file is PDB format file, it must have suffix .pdb.

If the output file is SDF format file, it must have suffix .sdf. The examples of molecular format conversion are as follows:

-i: input file with correct suffix

-o: output file with correct suffix

-c: select parameters for dealing with molecules. Currently, it can be “none”, “gen3d” and “addh”. The “none” is the default value that means molecular format conversion without 3D coordinates generation. The “gen3d” means molecular format conversion with 3D coordinates generation. The “addh” means molecular format conversion with adding hydrogen.

The example commands are as below:

1) PDB to Mol2. It must have suffix .pdb for input and suffix .mol2 for output.

```
#> molaical.exe -tool format -i E:/ligand.pdb -o E:/ligand.mol2
```

2) SDF to pdb. It must have suffix .sdf for input and suffix .pdb for output.

```
#> molaical.exe -tool format -i "E:/ligand.sdf" -o "E:/ligand.pdb"
```

Note: this is just an example, users can try more molecular format conversion.

3) Convert 2D molecule in 3D coordinates

```
#> molaical.exe -tool format -i E:/2D.pdb -o E:/3D.mol2 -c gen3d
```

3) Add hydrogen on molecule

```
#> molaical.exe -tool format -i E:/4.mol2 -o E:/4h.mol2 -c addh
```

5.3.6. Split and extract PDBQT file

The docking result file of MolAICaID may contain multiple docking poses. The MolAICaI software supplies a function for calculating total number of poses in the docking result file, splitting the molecular set into multiple molecular files with PDBQT format, and extracting the appointed docking pose in the docking result file. The parameters in MolAICaI are shown as below:

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

-t: appointed string for splitting the docking result file. The default value is “MODEL”

-n: it has three integer values: 0, -1, >0. “0” means 0 means split docking result file one by one; “-1” means calculate the total number of poses in the docking result file; “>0” means extracting an appointed molecule from docking result file in order.

-v: whether overlap the existed molecular file. The default value is true for overlap.

-i: path of input molecular PDBQT file.

-o: path of output directory.

The example commands are as below:

1) Split molecular set in PDBQT format one by one:

```
#> molaical.exe -tool pdbqt -i "E:\workdir\all.pdbqt" -o "E:\workdir\1"
```

2) Full command for splitting molecular set in PDBQT format one by one:

```
#> molaical.exe -tool pdbqt -t "MODEL" -v true -n 0 -i "E:\workdir\all.pdbqt" -o "E:\workdir\1"
```

3) Census the number of molecular set

```
#> molaical.exe -tool pdbqt -n -1 -i "E:\workdir\all.pdbqt"
```

4) Extract an appointed molecule from docking result file in order, e.g. obtain the second molecule:

```
#> molaical.exe -tool pdbqt -n 2 -i "E:\workdir\all.pdbqt" -o "E:\workdir\1"
```

5.4. Split molecules into fragments

It will get useful information on drug design by 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 default value is “off”.

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
```

Part II. Artificial intelligence

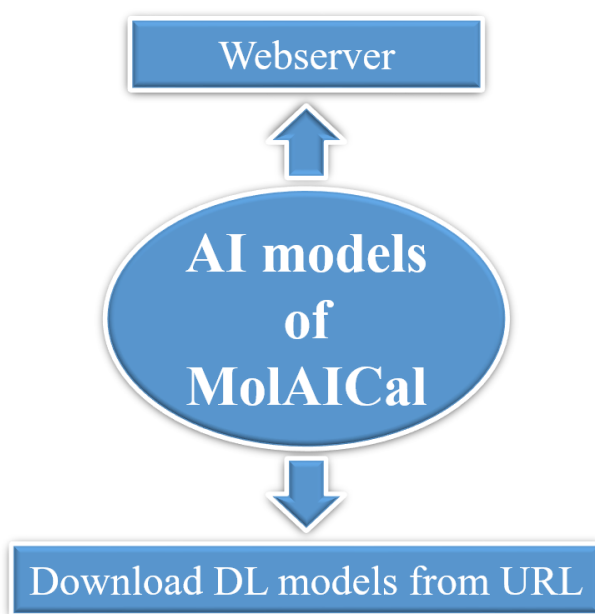
In this part, the excellent models **under the permitted license** or trained models by ourselves are collected into MolAICal to make the best of artificial intelligence (AI) such as deep learning for drug design. The deep learning model will continue to be updated if there are better models than the current models. I believe the above classical algorithm and the below AI models can promote the drug design progress.

Due to the big size of deep learning models, two ways are chosen for MolAICal (see Part II Figure 2): one is based on the webserver, the users can utilize functions of deep learning online, directly. The other is to download the deep learning models from the cloud server, and then, put them into the folder “MolAICal-xxx/mtools/AIModels” from the below URL:

<https://lzueducn->

my.sharepoint.com/:f:/g/personal/baiqf11_lzu_edu_cn/EpypiZ9_QDxIjoEjt0tl7osBVrssSaQ5x9Ulxue57ksUkg?e=2FYRVu

For more detailed procedures, please check the tutorial part: <https://molaical.github.io/tutorial.html>



Part II Figure 2. Two ways for AI models in MolAICal.

1. Protein fold

AlphaFold v2.0 is a very excellent deep learning frame for predicting the protein structure from a one-dimensional sequence³⁵. AlphaFold code is licensed under the Apache 2.0 License, the AlphaFold parameters are made available for non-commercial use only under the terms of the CC BY-NC 4.0 license. (More detailed license: <https://github.com/deepmind/alphafold#license-and-disclaimer>). AlphaFold is a very excellent tool for protein fold. MolAICal provides a webserver URL for protein fold based on a reduced AlphaFold version of the BFD database and with no ensembling (non-commercial use license under the terms of the CC BY-NC 4.0 license). If users want to use “caspl4” option for predicting protein structure with high-quality, users can send protein sequence to us for further academic cooperation or visit the official site for protein fold prediction: <https://colab.research.google.com/github/sokrypton/ColabFold/blob/main/AlphaFold2.ipynb>

MolAICal supplies this function via the webserver which URL can be obtained by the below command:

-model: specified AI model function

1) Getting the latest URL via MolAICal.

```
#> molaical.exe -model alphafold
```

2. Deep learning models of Binding affinity

Deep learning methods can be used to train the deep learning models for binding affinity prediction. To let researchers use deep learning models for binding affinity prediction, two deep learning models named OnionNet³⁶ and Pafnucy³⁷ are introduced under the permitted licenses. OnionNet uses GNU General Public License v3.0 (<https://choosealicense.com/licenses/gpl-3.0>). Pafnucy uses BSD 3-Clause License (<https://choosealicense.com/licenses/bsd-3-clause>). The source codes of OnionNet and Pafnucy are modified to calculate the binding affinity between receptor and ligands handily.

2.1. Binding affinity prediction by OnionNet model

To calculate the binding affinity between proteins and ligands by OnionNet, it needs to merge the files of proteins and ligands. MolAICal provides commands to calculate the binding affinities and to merge the files of proteins and ligands based on the trained OnionNet model as follows.

-model: specified AI model function

-r: path of receptor file

-l: path of ligand file

-c: merged complex file that containing protein and ligand files

-f: list file containing ligands' name

The example commands are as below:

1) Merge one protein and one ligand

```
#> molaical.exe -model mergeon -r E:/protein.pdb -l E:/ligand.mol2.pdb -c E:/complex.pdb
```

2) Merge one protein and many ligands

```
#> molaical.exe -model mergeon -r E:/GCGRNoLigand.pdb -f E:/list.txt
```

Finally, onionnet models can be run for binding affinity prediction when files are merged.

Notice: Currently, the below command are only supported to run on Linux system.

1) Calculate pK_x (pK_a or pK_i) for binding affinity prediction

```
molaical.exe -model onionnet -i /home/feng/onionnet/input_PDB_testing.dat -o result.csv
```

2.2. Binding affinity prediction by Pafnucy model

Pafnucy is another excellent deep learning model for binding affinity prediction. It can calculate the pK_x (pK_a or pK_i) between proteins and ligands. MolAICal provides commands to calculate the binding affinities between proteins and ligands based on the trained Pafnucy model in **Linux system** as follows.

The example commands are as below:

1) Calculating binding affinity between a ligand and a protein pocket using default HDF filename

```
#> molaical.exe -model pafnucy -l lig_1.mol2 -p pocket.mol2 -o res.csv
```

2) Calculate binding affinity between a ligand and a protein pocket with appointed HDF filename

```
#> molaical.exe -model pafnucy -l lig_1.mol2 -p pocket.mol2 -t tmp.hdf -o res.csv
```

3) Calculate binding affinity between many ligands and a protein pocket

```
#> molaical.exe -model pafnucy -l "lig_1.mol2 lig_2.mol2" -p pocket.mol2 -t tmp.hdf -o res.csv
```

4) If there is different pocket for each ligand, list them all in the same order as ligands. And then, MolAICal can calculate binding affinities between “lig_1.mol2- pocket1.mol2” and “lig_2.mol2- pocket2.mol2”

```
#> molaical.exe -model pafnucy -l "lig_1.mol2 lig_2.mol2" -p "pocket1.mol2 pocket2.mol2" -t tmp.hdf -o res.csv
```

2.3. Molecular generation by deep learning model

2.3.1 Molecular generation for drug-like molecules

This function can generate drug-like and FDA-like molecules that can be used for drug virtual screening. Molecular generation can be carried out by the basic version of MolaICal. The example commands are as below:

- dock:** value is AI, which means performing drug virtual screening by AI.
- 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.
- nc:** number of CPU cores
- 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.

1) Generating FDA-like molecules.

```
#> molaical.exe -dock AI -s FDAMol -n 30 -nf 10 -nc 4 -g on -b on -v off
```

Note: It will generate a file named tmpGenVS.dat that contains 30 molecular SMILES strings, and 3 folders named tmpGenMols1, tmpGenMols2 and tmpGenMols3. Users can generate much more molecules according to their needs. The generated 3D molecules are in PDBQT format by default. If users need to change the molecular format, they can refer to “Molecular format conversion” part in the MolaICal manual.

2) Generating FDA fragment-like molecules.

```
#> molaical.exe -dock AI -s FDAFrag -n 30 -nf 10 -nc 4 -g on -b off -v off
```

Note: It will generate a file named tmpGenVS.dat that contains 30 molecular SMILES strings. Users can generate much more molecules according to their needs.

3) Generating drug-like molecules.

```
#> molaical.exe -dock AI -s ZINCMol -n 30 -nf 10 -nc 4 -g on -b on -v off
```

Note: It will generate a file named tmpGenVS.dat that contains 30 molecular SMILES strings, and 3 folders named tmpGenMols1, tmpGenMols2 and tmpGenMols3. Users can use generated molecules to construct their own database. The generated 3D molecules are in PDBQT format by default. If users need to change the molecular format, they can refer to “Molecular format conversion” part in the MolaICal manual.

2.3.2 Generating similar ligands based on a known ligand

This function can generate novel scaffolds and groups based on a known ligand. It can be used to modify or design new drugs. The deep learning generative model trained by ligdream³⁸ is used to produce the novel ligands starting from a supplied ligand based on the variational autoencoder (VAE), convolutional and recurrent network. The ligdream is under GNU Affero General Public License v3.0 (<https://www.gnu.org/licenses/agpl-3.0.en.html>). MolAICal supplies this function via webserver which URL can be obtained by the below command:

-model: specified AI model function

1) Getting the latest URL via MolAICal.

```
#> molaical.exe -model ligdream
```

Appendix

1. Configure file of fragments growth using DL model

This is an example of configure file with deep learning (DL) model.

“#” is the 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 (deprecated)      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 growth without DL model

This is an example of configure file without deep learning (DL) model.
 # “#” is the 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

pdspath 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 Smith, B. R., Eastman, C. M. & Njardarson, J. T. Beyond C, H, O, and N! Analysis of the elemental composition of U.S. FDA approved drug architectures. *J Med Chem* **57**, 9764-9773 (2014).
- 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 Wang, R., Fang, X., Lu, Y., Yang, C. Y. & Wang, S. The PDBbind database: methodologies and updates. *J Med Chem* **48**, 4111-4119 (2005).
- 12 Morris, G. M. *et al.* AutoDock4 and AutoDockTools4: Automated docking with selective receptor flexibility. *J Comput Chem* **30**, 2785-2791 (2009).
- 13 Rester, U. From virtuality to reality - Virtual screening in lead discovery and lead optimization: a medicinal chemistry perspective. *Current opinion in drug discovery & development* **11**, 559-568 (2008).
- 14 Quiroga, R. & Villarreal, M. A. Vinardo: A Scoring Function Based on Autodock Vina Improves Scoring, Docking, and Virtual Screening. *PLoS One* **11**, e0155183 (2016).
- 15 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).
- 16 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).
- 17 Glykos, N. M. Software news and updates. Carma: a molecular dynamics analysis program. *J. Comput. Chem.* **27**, 1765-1768 (2006).
- 18 Andricioaei, I. & Karplus, M. On the calculation of entropy from covariance matrices of the atomic fluctuations. **115**, 6289-6292 (2001).
- 19 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).
- 20 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).
- 21 Humphrey, W., Dalke, A. & Schulten, K. VMD: visual molecular dynamics. *J. Mol. Graph.* **14**, 33-38, 27-38 (1996).
- 22 Baell, J. & Walters, M. A. Chemistry: Chemical con artists foil drug discovery. *Nature* **513**, 481-483 (2014).
- 23 O'Boyle, N. M. *et al.* Open Babel: An open chemical toolbox. *J. Cheminform.* **3**, 33 (2011).
- 24 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).
- 25 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).
- 26 Li, Y. *et al.* Assessing protein-ligand interaction scoring functions with the CASF-2013 benchmark. *Nat. Protoc.* **13**, 666-680 (2018).
- 27 Pettersen, E. F. *et al.* UCSF Chimera--a visualization system for exploratory research and analysis. *J. Comput. Chem.* **25**, 1605-1612 (2004).
- 28 Kim, R. & Skolnick, J. Assessment of programs for ligand binding affinity prediction. *J Comput Chem* **29**, 1316-1331 (2008).
- 29 Karney, C. F., Ferrara, J. E. & Brunner, S. Method for computing protein binding affinity. *J Comput Chem* **26**, 243-251 (2005).
- 30 Yap, C. W. PaDEL-descriptor: an open source software to calculate molecular descriptors and fingerprints. *J Comput Chem* **32**, 1466-1474 (2011).
- 31 Moriwaki, H., Tian, Y. S., Kawashita, N. & Takagi, T. Mordred: a molecular descriptor calculator. *J Cheminform* **10**, 4 (2018).
- 32 Halgren, T. A. Merck molecular force field. II. MMFF94 van der Waals and electrostatic parameters for intermolecular interactions. **17**, 520-552 (1996).
- 33 Ballester, P. J. & Richards, W. G. Ultrafast shape recognition to search compound databases for similar molecular shapes. *J. Comput. Chem.* **28**, 1711-1723 (2007).
- 34 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).
- 35 Jumper, J. *et al.* Highly accurate protein structure prediction with AlphaFold. *Nature* (2021).
- 36 Zheng, L., Fan, J. & Mu, Y. OnionNet: a Multiple-Layer Intermolecular-Contact-Based Convolutional Neural Network for Protein-Ligand Binding Affinity Prediction. *ACS Omega* **4**, 15956-15965 (2019).
- 37 Stepniewska-Dziubinska, M. M., Zielenkiewicz, P. & Siedlecki, P. Development and evaluation of a deep learning model for protein-ligand binding affinity prediction. *Bioinformatics* **34**, 3666-3674 (2018).
- 38 Skalic, M., Jimenez, J., Sabbadin, D. & De Fabritiis, G. Shape-Based Generative Modeling for de Novo Drug Design. *J. Chem. Inf. Model.* **59**, 1205-1214 (2019).