

使用 MolAICal 的深度学习模型和分子对接程序进行药物的虚拟筛选

作者：Qifeng Bai (update 2021-10-16)

更多教程（含英文教程）请见如下：

MolAICal 官方主页：<https://molaical.github.io>

MolAICal 官方主页中国镜像：<https://molaical.gitee.io>

MolAICal 中文博客：<https://molaical.gitee.io/cntutorial.html>

1. 简介

一种新药的研发大概需要耗费 26 亿美元。即使有大量资金的投入，90%的新药仍会在临床试验和获批上市过程中夭折[1]。本教程介绍了 MolAICal 基于人工智能和分子对接进行药物虚拟筛选的流程，其中 model.pdb 是优化的蛋白质模型文件，你可以替换成自己的蛋白质模型。此方法将帮助药物学家、化学家及其它领域的科学家根据靶点的活性口袋合理设计药物。

2. 工具

2.1. 所需软件下载地址

1) MolAICal: <https://molaical.github.io> 或 <https://molaical.gitee.io>

国内镜像 MolAICal: <https://molaical.gitee.io>

2) UCSF Chimera: <https://www.cgl.ucsf.edu/chimera/>

2.2. 操作示例文件

所有用到的操作教程文件均可在下面的网站下载：

<https://gitee.com/molaical/tutorials/tree/master/002-AIVS>

3. 操作流程

这一步的前提是你熟悉了使用 MolAICal 进行分子对接，“model.pdb”是优化的“6lu7.pdb”。用户也可以选择直接使用“6lu7.pdb”。具体步骤如下：

3.1. 使用 UCSF Chimera 将蛋白质和配体结构分开

1) 首先，在 UCSF Chimera 中载入复合物结构。File→Open→model.pdb (如图 1 所示)

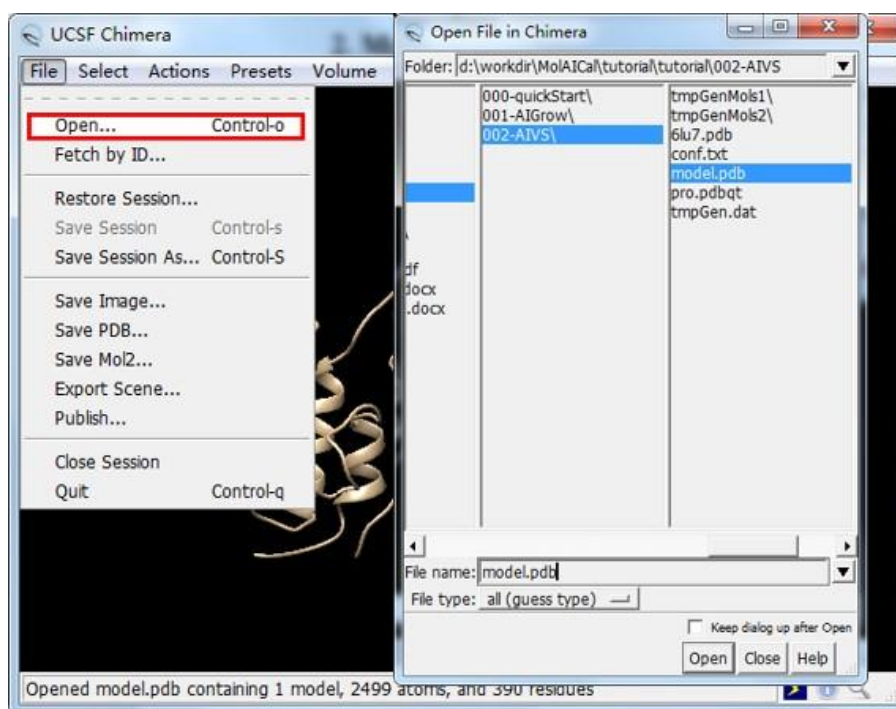


图 1. 载入蛋白结构文件

- 2) 选择配体 LIG 并将其删除 (如图 2 所示)。使用图 2 中相同的方法将水 HOH 选中并删除。

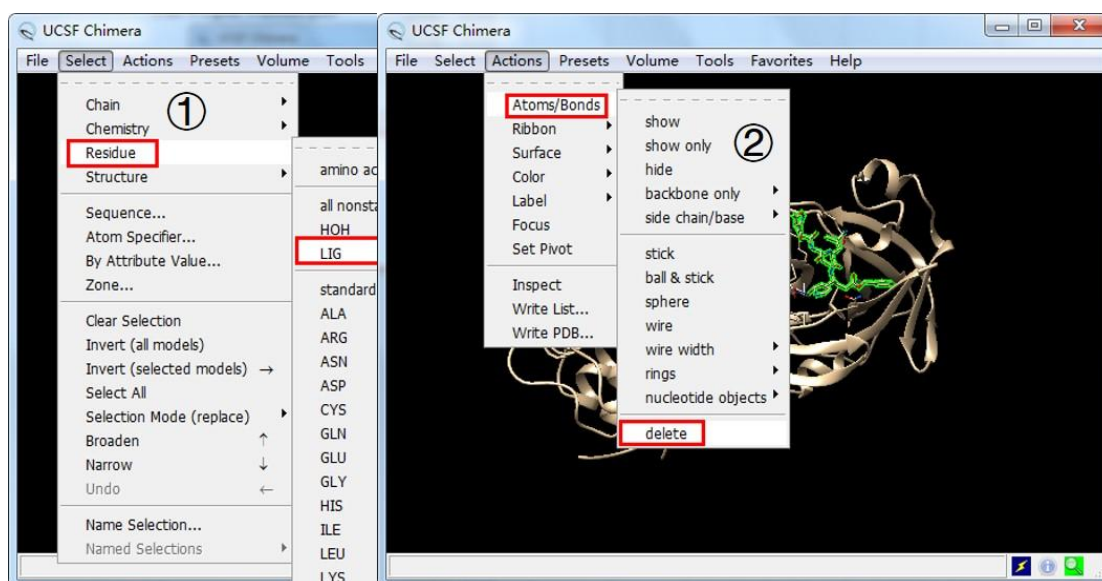


图 2. 选中并删除配体

- 3) 保存没有配体的蛋白质结构并且命名为“protein.pdb” (如图 3 所示)。

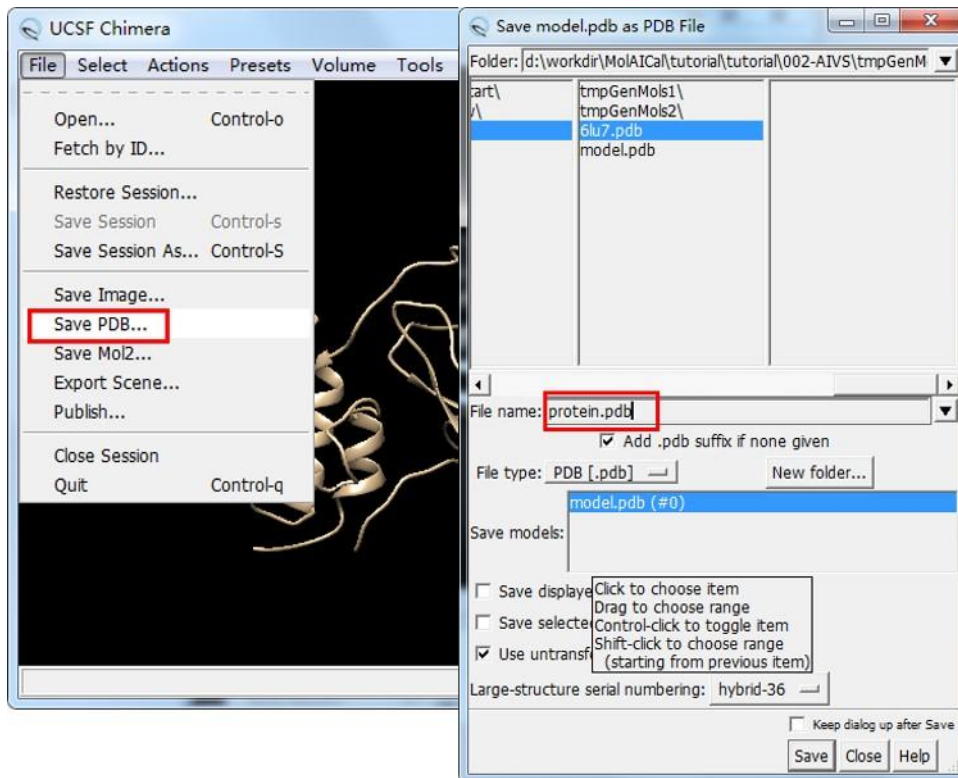


图 3. 保存蛋白质结构

4) 关闭本次会话，重新载入“model.pdb”，选择配体，反选并删除反选的蛋白（如图 4 所示）。

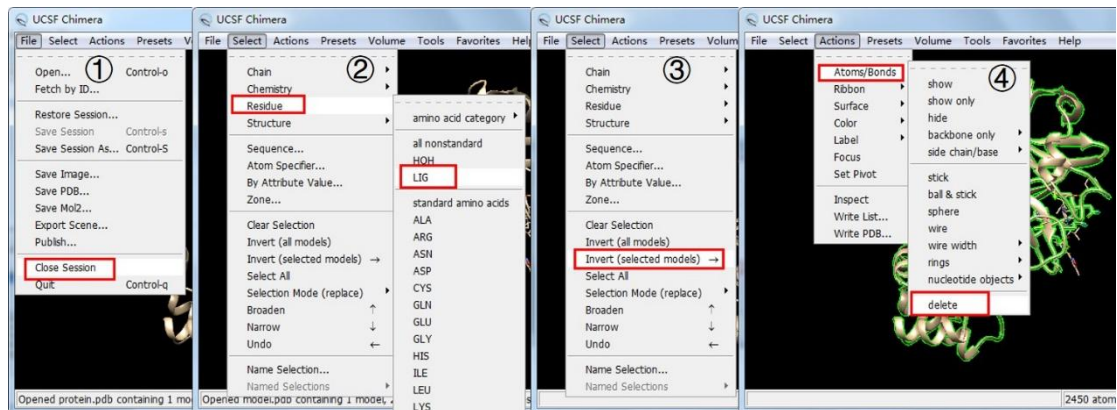


图 4. 选中并删除受体蛋白

5) 将配体文件保存为“ligand.pdb”（如图 5 所示）。

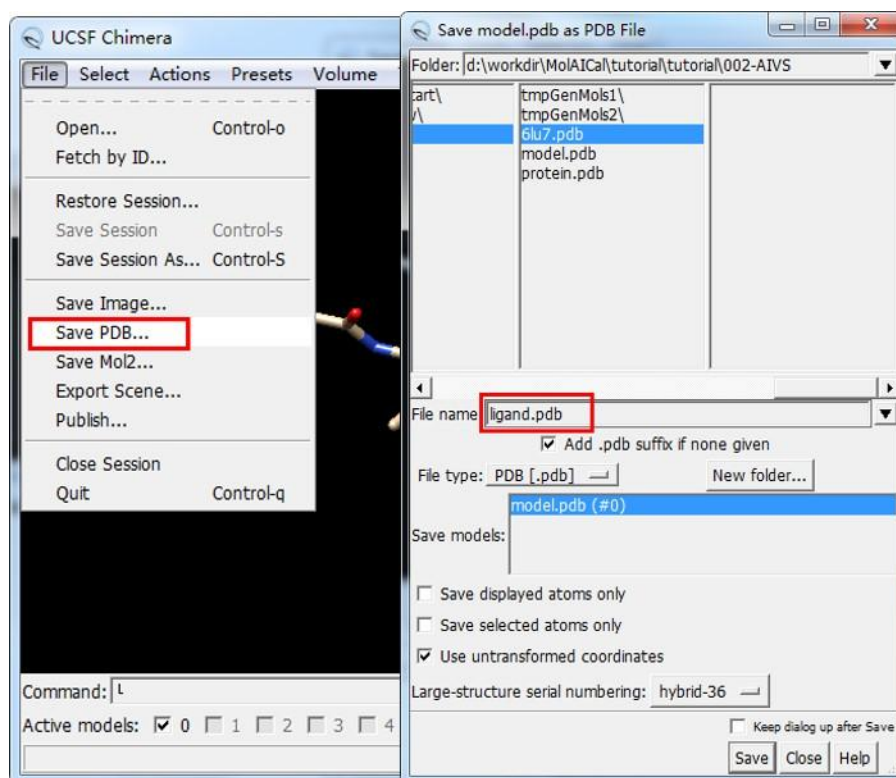


图 5. 保存配体文件

3.2. 计算盒子质心和长度

1. 参照上述步骤选择配体或者重新载入“ligand.pdb”并选择配体。然后选择距离工具：
Tools→Structure Analysis→Distance (如图 6 所示):

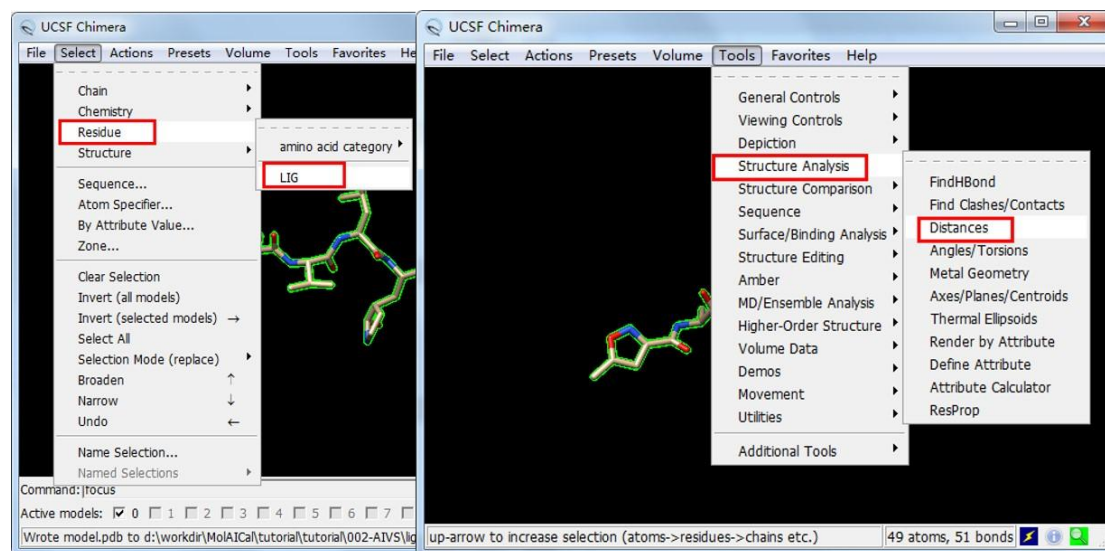


图 6. 选择距离工具

2. 根据配体计算蛋白质活性口袋的质心坐标 (如图 7 所示)。

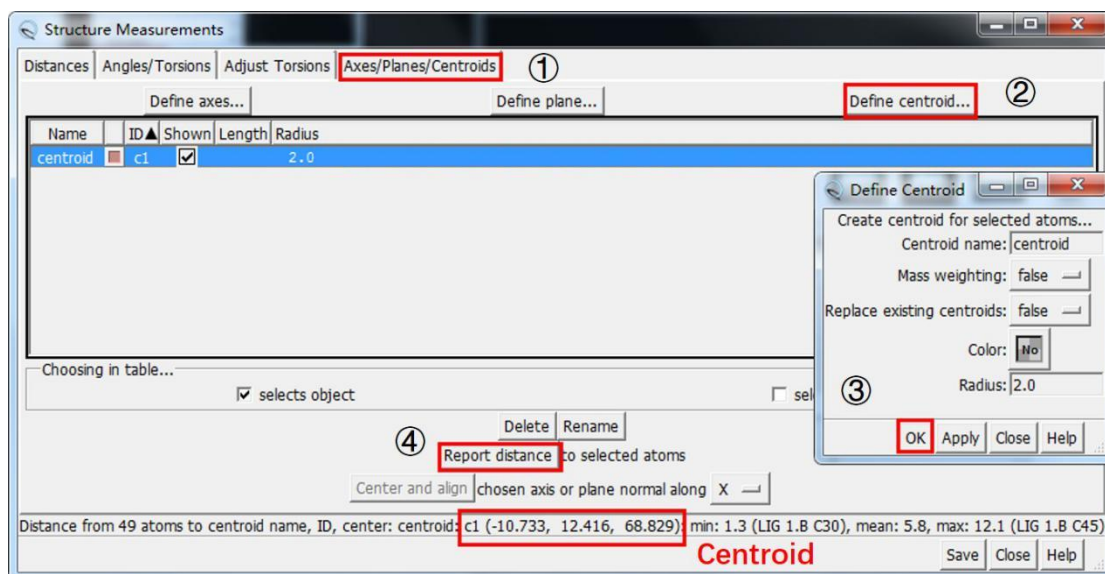


图 7. 获得质心坐标

创建 “**conf.txt**” 并将质心坐标写入该文件：

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

Notice: “**conf.txt**” 是 MolAICal 的默认名称。假如你想创建成其它名字(例如, xxx.txt), 在虚拟筛选的时候, MolAICal 的命令行要添加参数 “-m xxx.txt”; 更多的运行细节, 可以参考 MolAICal 的手册。除此之外, 根据具体的研究, 用户可以直接修改 “conf.txt” 中的参数。

3. 设置对接盒子的体积

计算最终盒子尺寸。你可以将 X, Y, Z 长度分别设置为 25, 30, 25。在 MolAICal 中使用下文提到的命令生成“**box.bild**”(注意: X, Y, Z 坐标的双引号是必须添加的, X, Y, Z 坐标之间的间隔为一个空格。):

1) 执行以下命令, 获得“**box.bild**”:

```
#> molaical.exe -tool box -i "-10.733 12.416 68.829" -l "25.0 30.0 25.0" -o "box.bild"
```

2) File→open, 然后打开“**box.bild**”, 检查生成的盒子大小是否合适 (如图 8 所示)。

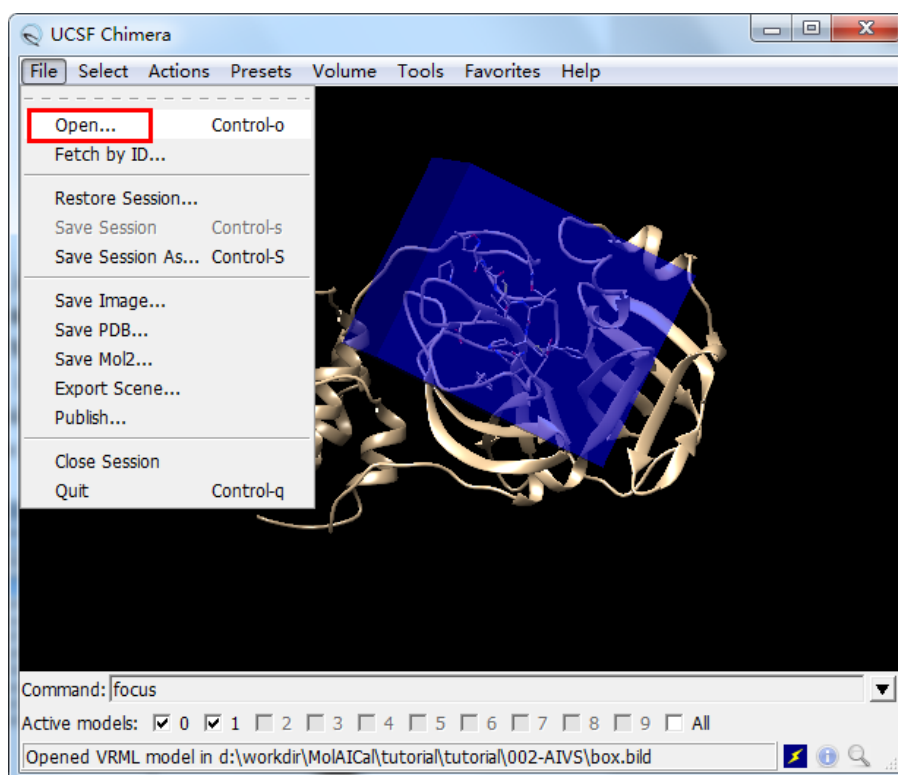


图 8. 使用 UCSF Chimera 打开 box.bild

如上图所示盒子大小 25, 30, 25 是合适的, 因此确定最终质心参数为-10.733, 12.416, 68.829, 最终盒子沿 X, Y, Z 轴的长度为 25.0, 30.0, 25.0。

注意: 如果你用 VMD 软件计算了几何中心, 最终的中心点参数将是-10.86, 12.57, 68.82。这两种方法得到的结果都可以用于本教程, 本教程暂使用 UCSF Chimera 算出来的质心坐标。

3.3. 虚拟筛选前将蛋白质结构转换为 PDBQT 格式

使用下面的命令可以准备受体蛋白的 PDBQT 文件:

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

说明: MolAICal-xxx 是你实际下载版本的目录。

到处为止所有文件准备就绪。

3.4. 用深度学习模型和分子对接进行虚拟筛选

```
#> cd 002-AIVS
```

最后在后台运行以下命令:

Linux 系统:

```
#> molaical.exe -dock AI -s ZINCMol -n 6 -nf 3 -nc 3 >& vs.log &
```

-n: 代表对接产生的总分子数目

-nf: 单个文件夹中包含的分子数量

-nc: 执行命令使用的 CPU 数

Windows 系统 (使用 PowerShell):

```
#> molaical.exe -dock AI -s ZINCMol -n 6 -nf 3 -nc 3
```

如果要在后台运行,请执行下面的命令:

```
#> powershell -windowstyle hidden -command "molaical.exe -dock AI -s ZINCMol -n 6 -nf 3 -nc 3"
```

如果你想依据已知的药物数据库进行经典的虚拟筛选,可以参考 MolAICal 教程中药物设计部分的第三部分(<https://molaical.github.io/tutorial.html>)。

4. 结果

4.1 查看结果

用户可以使用 Pymol (<https://github.com/cgohlke/pymol-open-source-wheels> 或 <https://www.cgohlke.com>) 直接载入 PDBQT 格式的分子. 这里使用 UCSF Chimera 来查看虚拟筛选的结果。

打开 002-AIVS\tmpGenMols1

1) 加氢 (可选)

```
#> molaical.exe -dock addh -i 1_out.pdbqt
```

2) 更改格式“pdbqt”成“pdb”

```
#> molaical.exe -dock pdbqt2pdb -i 1_out.pdbqt
```

上述命令将产生一个名为“1_out.pdb”的文件。然后使用 UCSF Chimera 打开配体分子“1_out.pdb”和受体分子“protein.pdb”。图 9 显示了对接筛选的结果,表明了 MolAICal 可以通过深度学习模型虚拟筛选到合适的配体分子。

注意: 图 9 的显示方面有些步骤省略了,假如用户想得到图 9 这样 surface 的图像,可以使用 UCSF Chimera 工具栏的选项: “Actions→Surface→show”

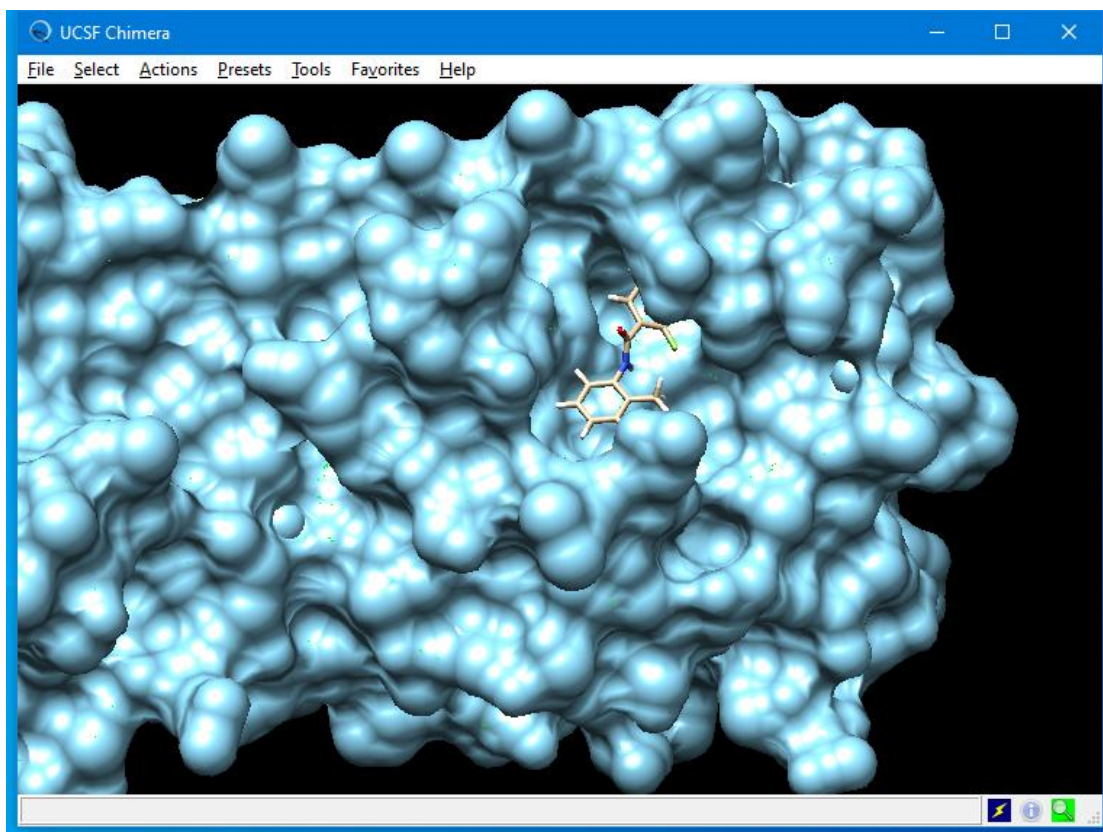


图 9. 基于 SARS-CoV-2 Mpro 活性口袋筛选出的分子

4.2 对虚拟筛选的结果进行排序

切换到 002-AIVS，然后运行如下命令：

```
#> molaical.exe -call run -c sfile -i printscore.py "tmpGenMols1/*out.pdbqt"
```

这个命令将显示分子名称和对应的打分（按分数升序排列），假如用户想保存输出的结果，可以使用如下命令：

```
#> molaical.exe -call run -c sfile -i printscore.py "tmpGenMols1/*out.pdbqt" > results.log
```

说明：“molaical.exe”是实际 MolAICal 的安装目录。所有的脚本文件都保存在 MolAICal 子文件夹“scripts”中。

4.3 提取排名靠前的分子到新建的文件夹中

假如用户想将排名靠前的分子移动到新建的文件夹中，以便于分析结果，本教程提供了一种方法，下面的命令是将 2 个排名靠前的分子移到名为“results”的文件夹中：

```
#> molaical.exe -call run -c sfile -i get_top_results.py "tmpGenMols1/*out.pdbqt" 2 results
```

运行上述命令，2 个排名靠前的分子将被移动到“results”文件夹中

5. 续跑虚拟筛选任务 (选项)

如果虚拟筛选任务因不可预见的情况而提前终止，恢复过程只需两个步骤：

1. 通过以下命令将所有现有的 VS 结果文件（包含默认"_out.pdbqt"后缀）移动到另一个目录：

```
#> molaical.exe -call run -c sfile -i molaical_batch.py -sd tmpGenMols1 -ds -ik "_out.pdbqt" > tmplist
```

-ds: 是否搜索具有指定字符的文件，如果仅输入"-ds"，表示 True，搜索具有指定字符的文件。

-sd: 搜索的目录。

-ik: 用于搜索的关键字。

```
#> molaical.exe -call run -c sfile -i molaical_batch.py -cvs -il tmplist -ol l_complete_folder
```

-cvs: 是否准备继续运行虚拟筛选。如果仅输入"-cvs"，表示 True，准备继续运行虚拟筛选作业。

-il: 配体列表的文件。

-ol: 在使用"-cvs"继续虚拟筛选的部分中，是存储的文件夹。

注意：完整分子将被移动到目录"l_complete_folder"。

选项：有时，一个文件夹中可能有大量分子。为了减少一个文件夹中的分子数量，可以按如下方式指定一个文件夹中的分子数量：

```
#> molaical.exe -call run -c sfile -i molaical_batch.py -cvs -dp 3 -cvs -il tmplist -ol t1_folder
```

-dp: 在使用"-cvs"继续虚拟筛选的部分中，是每个文件夹中存储的筛选分子数量。默认值为"None"，表示将所有筛选分子存储在一个文件夹中。可以是 1、2 或 3...，表示每个文件夹中存储的分子数量。

注意：将生成文件夹"t1_folder1"、"t1_folder2"....，每个文件夹包含'-dp'后的存储数量。

2. 重复上述虚拟筛选步骤以处理未筛选的分子。

6. Linux 版 MolAIcal: 架构与加速方案

如图 10 所示, MolAIcal 容器不仅能调用容器系统的文件和程序, 还能通过挂载与映射机制访问宿主机库文件, 有效解决了库冲突或依赖缺失等问题。然而, 容器内程序的执行速度确实可能低于本地环境。

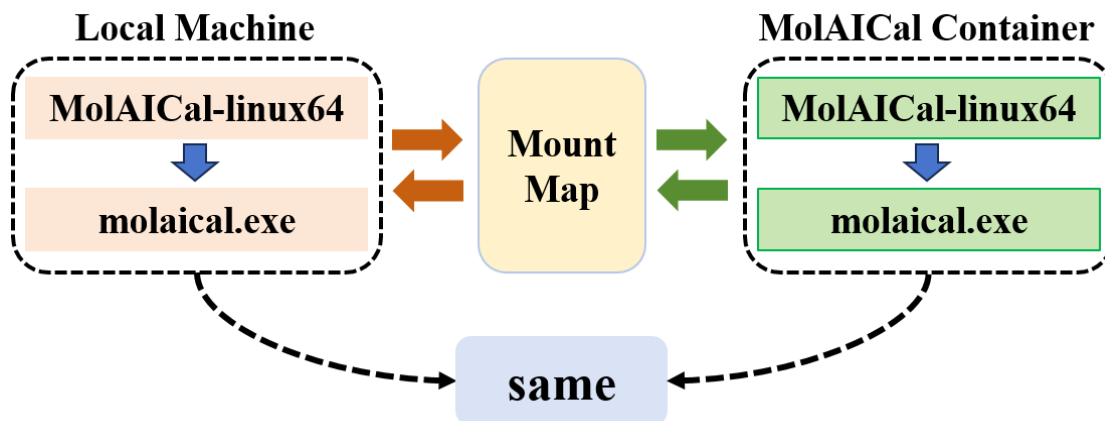


图 10. MolAIcal Linux 版本架构图

图 10 表明: 由于映射关系的存在, 容器中"molaical.exe"执行的计算命令与本地"molaical.exe"运行命令完全一致。

- ◆ 因此对于库依赖较少的任务(如使用 MolAIcal 进行虚拟筛选), 建议将 MolAIcal 从容器复制到本地:

1. 进入容器文件系统 (默认进入"/root"目录)

```
#> molaical.exe -eset shell in
```

2. 'cp'命令前半部分指向容器路径, 后半部分指向宿主机路径

```
#> cp -r soft/MolAIcal-linux64 <local machine directory>/
```

3. 退出容器文件系统

```
#> exit
```

- ◆ 最终通过相同方法在本地执行虚拟筛选, 可显著加速筛选过程。

参考文献

1. Fleming N. How artificial intelligence is changing drug discovery, Nature 2018;557:S55-S55.

附录

参考其它排序方法：

假如你在 Windows 环境下：

使用 Excel 打开“results.log”，其中分隔符“Separator”选择空格。或者直接 will 将“results.log”的内容复制到 Excel 中。在第二列，选择所有数据，并且根据需要选择工具栏中的“Sort Largest to Smallest”（降序）或“Sort Smallest to Largest”（升序）（见图 11）。

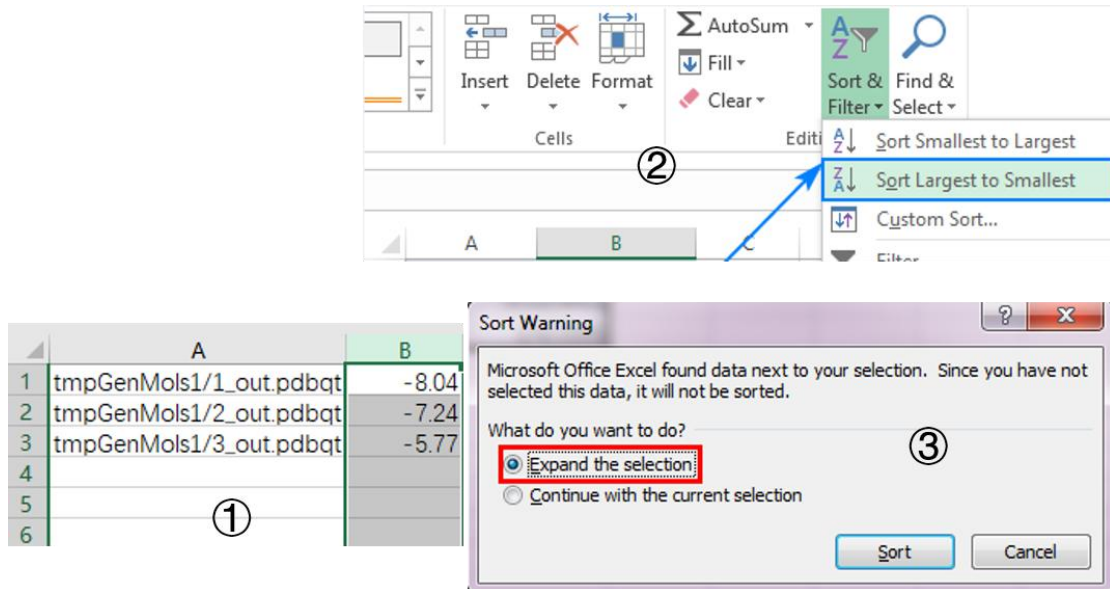


图 11. 排序结果

假如你在 Linux 环境下，用户可以使用如下命令：

```
#> sort -n -t ' ' -k 2r results.log > rank.dat
```

说明：参数“2r”是升序排列，而“1r”是降序排列。