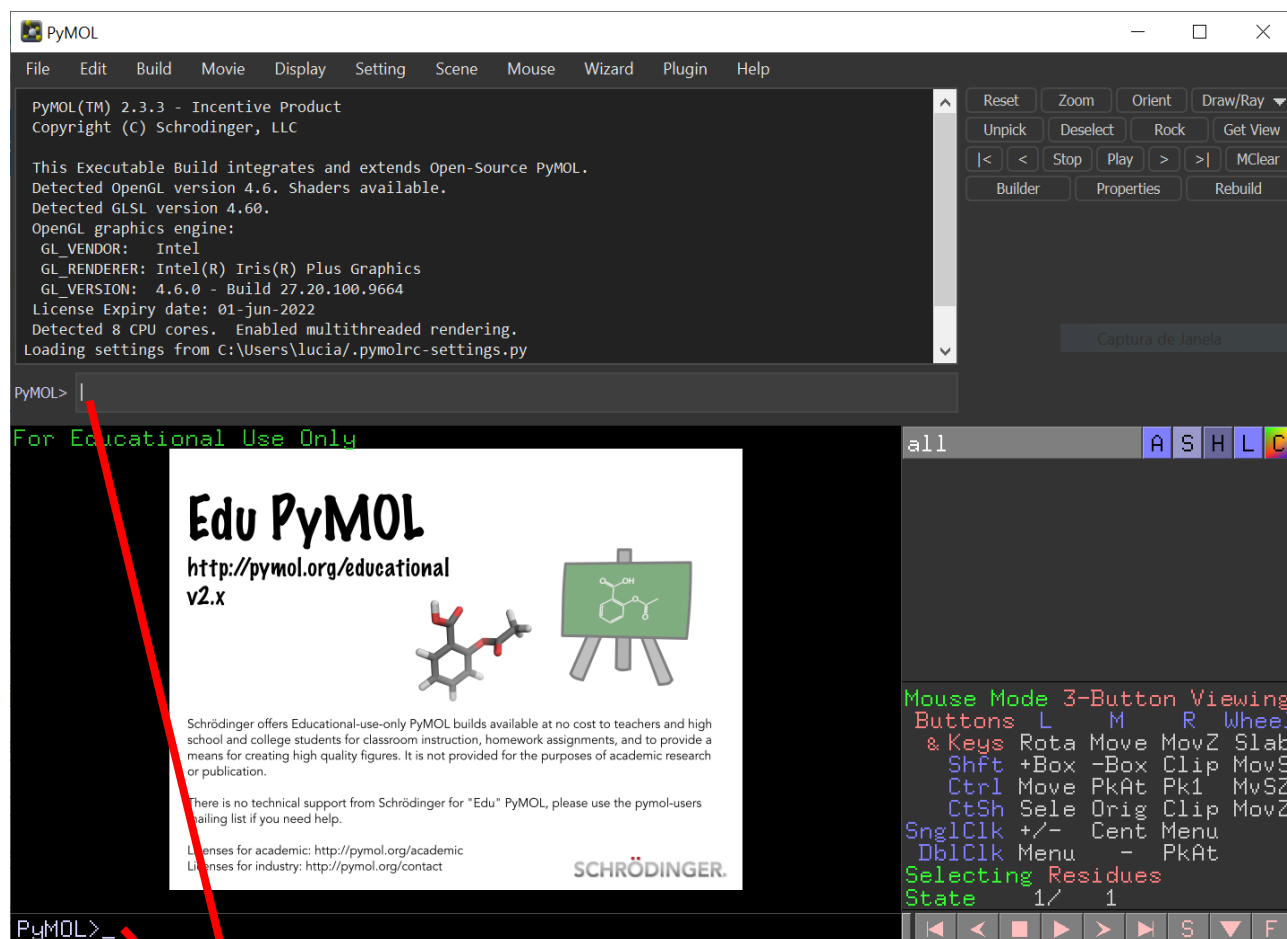


Tutorial – Visualização Gráfica com PyMOL e Docking com Vina

Criado por Lucianna Santos – 23/11/2021

Interface do PyMOL 2.x



Linha de comando

Menu de comandos:



A = Action, S = Show, H = Hide, L = Label, C = Color

Comandos básicos:

Para arrastar – clique com o botão do meio do mouse e arraste.

Para alterar o zoom – clique com o botão direito do mouse e arraste para cima (diminuir zoom) ou para baixo (aumentar zoom).

Para girar – clique com o botão esquerdo do mouse e arraste.

Para “cortar” a exibição de parte da imagem (em níveis de profundidade) – role o botão do meio do mouse.

Como executar comandos no PyMOL?

Em geral, os comandos podem ser executados a partir do painel de comandos, utilizando os menus descritos acima, ou a partir da linha de comando. Para utilizar a linha de comando, digite o comando e dê **enter**. Neste tutorial utilizaremos tanto a linha de comando quanto o painel de comandos.

Como carregar uma nova molécula no PyMOL

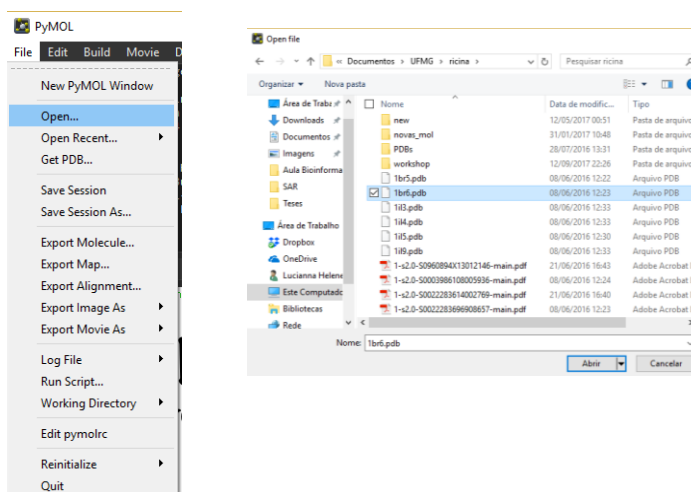
A) Comando fetch <PDB> . Exemplo: fetch 1br6.

Explicação: Faz o download de uma estrutura diretamente do site do PDB.

B) Comando load <PDB> . Exemplo: load 1br6.pdb.

Explicação: Carrega uma estrutura de um arquivo local.

C) File > Open.



Explicação: Abre uma estrutura PDB navegando as pastas no computador.

Parte 1 – Explorando opções de representação e preparando uma figura no Pymol

1) Abra o PyMOL pela a linha de comando:

```
$ pymol &
```

2) Abra a estrutura **PDB 7L10**, um complexo da protease principal do SARS-CoV-2 (*Main protease*) com um inibidor otimizado de uma campanha de triagem virtual de

2000 compostos aprovados. Como essa protease é ativa apenas em forma de dímero, buscamos sua “*biological assembly*”, dessa forma utilizaremos a sua forma que se acredita ser funcional:

```
PyMOL> set assembly, 1
PyMOL> fetch 7L10
```

- 3) Inicialmente estarão representados dois monômeros, conforme observado na unidade biológica. Ambos monômeros estão na mesma representação. Podemos separar, formando dois objetos distintos.

```
PyMOL> split_states 7L10
```

Agora temos os monômeros separados:

all	A	S	H	L	C
7L10 $\star/2$	A	S	H	L	C
7L10_0001 $1/1$	A	S	H	L	C
7L10_0002 $1/1$	A	S	H	L	C

Podemos remover o original:

```
PyMOL> delete 7L10
```

E renomear os outros objetos:

```
PyMOL> set_name 7L10_0001, 7L10_A
PyMOL> set_name 7L10_0002, 7L10_B
```

Apesar de renomear os objetos ambas as cadeias estão como A, ficando difícil de distinguir as cadeias:

```
/7L10_A/A/A/1 6 11 16 21 26 31 36 41 46 49 56 61 66 71 76
SGFRKMAFSPGKVEGCMVQVTCGTTTLNGLWLDVVCPRHVICTS--MLNPNYEDLLIRKSNHNFLVQAGNVQLRVIGH
/7L10_B/A/A/1 6 11 16 21 26 31 36 41 46 49 56 61 66 71 76
SGFRKMAFSPGKVEGCMVQVTCGTTTLNGLWLDVVCPRHVICTS--MLNPNYEDLLIRKSNHNFLVQAGNVQLRVIGH
```

Podemos alterar a cadeia do objeto 7L10_B para cadeia B:

```
PyMOL> select 7L10_B
PyMOL> alter (sele), chain="B"
```

```
/7L10_A/A/A/1 6 11 16 21 26 31 36 41 46 49 56 61 66 71 76
SGFRKMAFSPGKVEGCMVQVTCGTTTLNGLWLDVVCPRHVICTS--MLNPNYEDLLIRKSNHNFLVQAGNVQLRVIGH
/7L10_B/A/B/1 6 11 16 21 26 31 36 41 46 49 56 61 66 71 76
SGFRKMAFSPGKVEGCMVQVTCGTTTLNGLWLDVVCPRHVICTS--MLNPNYEDLLIRKSNHNFLVQAGNVQLRVIGH
```

- 4) Vamos representar a proteína como cartoon:

```
PyMOL> hide lines
PyMOL> show cartoon
```

Para distinguir melhor as duas cadeias, altere a cor da cadeia B para ciano:

PyMOL> **color cyan, chain B** OU PyMOL> **util.cbac chain B**

- 5) Vamos explorar diferentes representações da estrutura secundária, alterando a forma de representação de hélices e folhas-beta. A cada comando abaixo, observe as modificações:

PyMOL> **set cartoon_flat_sheets, 0**
PyMOL> **set cartoon_smooth_loops, 0**
PyMOL> **set cartoon_fancy_helices, 1**
PyMOL> **set cartoon_discrete_colors, 1**
PyMOL> **set cartoon_highlight_color, 1**

- 6) Vamos explorar uma outra exibição da estrutura da proteína na forma de bastões e as águas como esferas, e esconda a representação em cartoon:

PyMOL> **show sticks**
PyMOL> **show nb_spheres**
PyMOL> **hide cartoon**

- 7) Vamos voltar para a forma de cartoon:

PyMOL> **hide sticks, chain A or chain B**
PyMOL> **hide nb_spheres**
PyMOL> **hide wire**
PyMOL> **show cartoon**

- 8) Crie um objeto contendo apenas o inibidor da cadeia A e um contendo esse mesmo inibidor e as moléculas de água ou resíduos de proteína a no máximo 6 Å de distância do composto.

- para extrair um objeto nomeado **ligante**, contendo a molécula nomeada **STC** no arquivo PDB:

PyMOL> **extract ligante, resn XEY and chain A**
PyMOL> **show sticks, ligante**
PyMOL> **util.cbam ligante**

- a partir do objeto anteriormente criado (**ligante**), para criar um objeto nomeado **sitio_ligante**, que contenha o objeto anterior e todos os átomos a até 6 Å de distância dos átomos contidos neste objeto:

PyMOL> **create sitio_ligante, ligante around 6**
PyMOL> **hide cartoon, sitio_ligante**
PyMOL> **show sticks, sitio_ligante**
PyMOL> **util.cbam sitio_ligante**
PyMOL> **show nb_spheres, sitio_ligante and resn HOH**

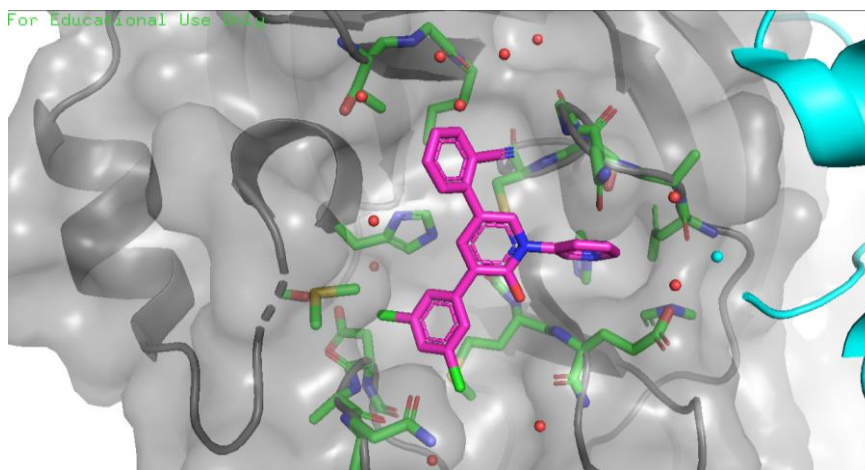
- 9) Represente a superfície da proteína com uma transparência, a partir do objeto **7L10_A**.

```
PyMOL> show surface, 7L10_A  
PyMOL> color gray, 7L10_A  
PyMOL> set transparency, 0.5
```

10) Mude o fundo para branco e amplie na região do ligante.

```
PyMOL> bg_color white  
PyMOL> zoom sitio_ligante
```

Você deverá obter uma imagem semelhante à figura abaixo (se necessário, utilize o mouse para dar zoom e alterar a orientação até obter uma imagem semelhante à mostrada abaixo):

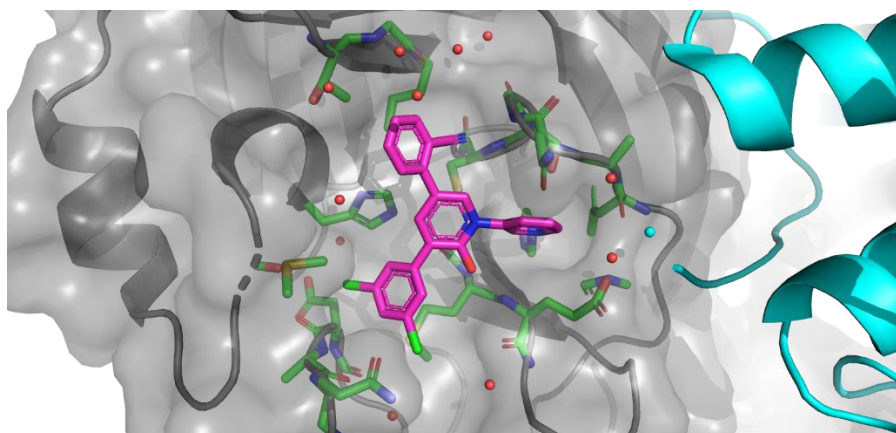


11) Para melhorar a resolução da imagem, defina o tamanho da figura no momento de executar o comando ray:

```
PyMOL> ray
```

12) Salve a imagem com uma resolução específica:

```
PyMOL> png key_chain_A.png, dpi=300
```



13) **Salve a sessão do Pymol** (File > **Save Session**) como **Mpro _parte1.pse**

14) Vamos analisar as interações intermoleculares entre inibidor e enzima.

Calcule as interações entre átomos polares do inibidor e a enzima ou moléculas de água.

- Selecione o ligante no menu de seleção

ligante 1/1	A	S	H	L	C
-------------	---	---	---	---	---

- No menu **Actions (A)** da seleção, clique em **Find – polar contacts – to any atoms**

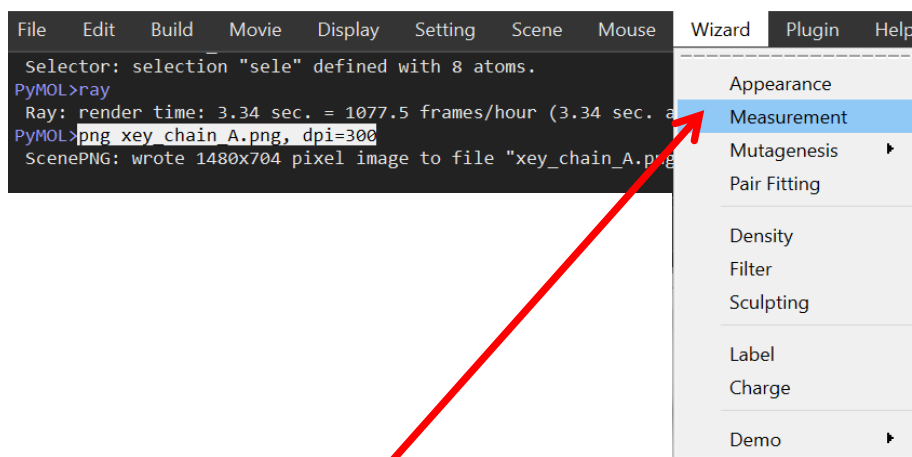
15) Meça a distância entre os átomos envolvidos em cada uma das interações intermoleculares.

- Primeiro esconder a representação da estrutura **7L10_A** clicando no nome no menu lateral.

all	A	S	H	L	C
7L10_A 1/1	A	S	H	L	C
7L10_B 1/1	A	S	H	L	C
ligante 1/1	A	S	H	L	C
sitio_ligante 1/	A	S	H	L	C
ligante_polar_co	A	S	H	L	C

- No Menu **Wizard**, clique em **Measurement**.

Para medir as distâncias entre dois átomos, clique sucessivamente em cada um.



Para sair do menu **Measurement**, clicar em **Done**:

Measurement
Distances
Create New Object
Delete Last Object
Delete All Measurements
Done

16) Vamos alterar as cores das linhas para visualizarmos melhor. Para isso, digite na linha de comando

PyMOL> **set dash_color, black**

- 17) Após medir todas as distâncias, clique em **Done**, no Menu **Measurement**, e salve como **Mpro_dist.pse**.

Parte 2 – *Re-docking* utilizando o Autodock Vina

Vamos inicialmente utilizar a metodologia de reprodução de pose, chamada também de **re-docking**. A metodologia consiste em tentar reproduzir a posição adquirida pelo ligante na estrutura cristalográfica utilizando o programa de atracamento. Portanto, obtemos uma validação do protocolo e um teste do algoritmo de atracamento para o caso da proteína estudada. Ao final, podemos comparar interações e o RMSD entre pose e posição cristalográfica do ligante.

- 1) Adicione hidrogênios (Estruturas PDB originadas de cristalografia normalmente não possuem hidrogênios).

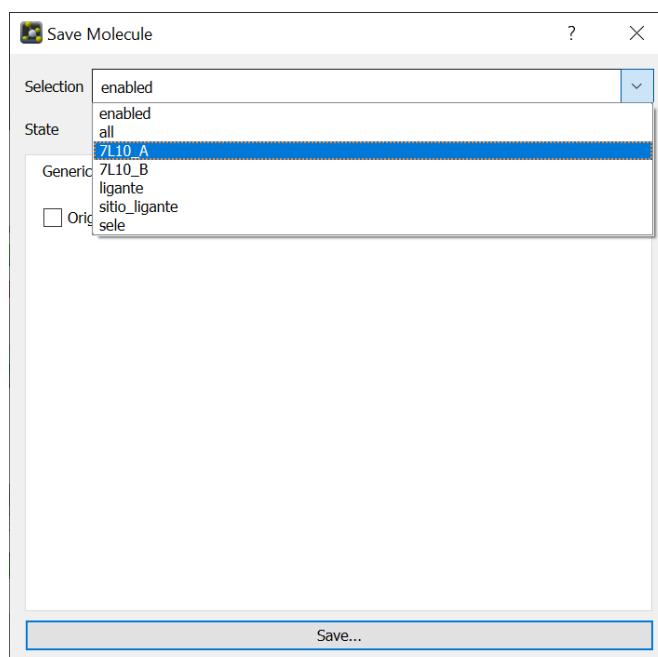
PyMOL> **h_add**

OBS. 1: Para simplificar o tutorial as protonações dos resíduos ASP, CYS, HIS, LYS, e TYR, não foram exploradas. O correto é buscar na literatura a protonação correta para o pH de ativada ou utilizar um programa/servidor de predição com o PROPKA (<https://github.com/jensengroup/propka>) ou H++ (<http://biophysics.cs.vt.edu/>).

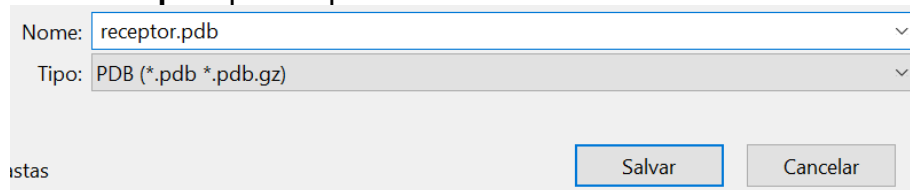
- 2) Para o atracamento vamos primeiro remover as moléculas de água (em alguns casos podemos manter algumas moléculas de águas, mas essas moléculas precisam ser analisadas com cuidado):

PyMOL> **remove resn hoh**

- 3) Apesar da estrutura ativa ser o dímero, vamos trabalhar apenas com cadeia A. Vamos salvar arquivos separados para o **receptor** e para o **ligante**. Em **File > Export Molecule**



Dar o nome de **receptor** para a proteína.

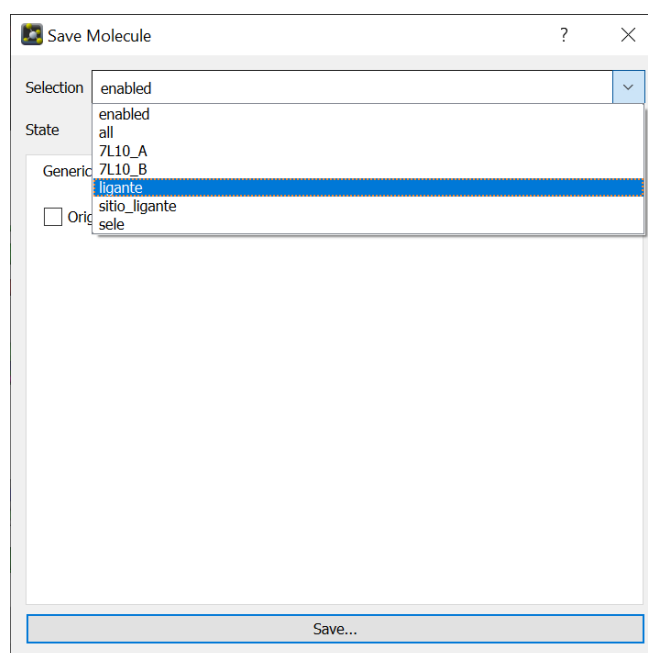


Nome: receptor.pdb

Tipo: PDB (*.pdb *.pdb.gz)

Salvar Cancelar

Faça o mesmo para o **ligante**, salve como **ligante**.

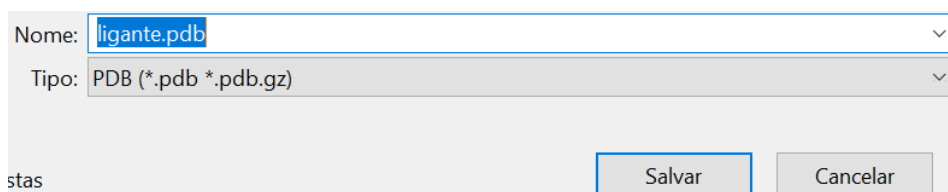


Save Molecule

Selection: enabled

State: enabled, all, 7L10_A, 7L10_B, **ligante**, sitio_ligante, sele

Save...



Nome: ligante.pdb

Tipo: PDB (*.pdb *.pdb.gz)

Salvar Cancelar

- 4) O Vina não faz um pré-cálculo de Grid. Porém, uma demarcação da área de docking (sítio ativo ou até mesmo a proteína toda) precisa ser estabelecida. Como estamos usando uma estrutura que já possui o ligante podemos calcular o centro de massa do ligante para ser o centro da nossa Grid.

PyMOL> **centerofmass** ligante

```
PyMOL>centerofmass ligante
Center of Mass: [ 8.417, -0.060, 22.273]
```

Anote esses números. Vamos utilizá-los logo adiante.

- 5) Podemos fechar o **PyMOL** não vamos precisar dele para os próximos passos. A preparação para o docking será feita por **linha de comando**. Já que o programa de docking que vamos utilizar, **Vina**, foi projetado para não ter uma interface e com isso ser um programa mais leve.

- 6) Para o docking com o **Vina** precisamos dos arquivos de entrada no formato PDBQT. Esse formato é uma adaptação do formato PDB. Nele além das coordenadas atômicas temos informação de carga e descrição dos átomos, específico para os programas **Vina** e **Autodock4**.

Para criar o PDBQT do receptor basta:

```
$ pythonsh $mglscripsts/prepare_receptor4.py -r receptor.pdb
```

OBS. 2: pythonsh é uma versão (antiga) compilada do python para o pacote MGLTools. O caminho dele é:

```
/programs/mgltools_x86_64Linux2_1.5.6/bin/python
```

OBS. 3: \$mglscripsts é um PATH criado para não precisar digitar todo o caminho até o script. O caminho é:

```
/programs/mgltools_x86_64Linux2_1.5.6/MGLToolsPkgs/AutoDockTools/Utilities24/prepare_receptor4.py
```

- 7) Para criar o PDBQT do ligante basta:

```
$ pythonsh $mglscripsts/prepare_ligand4.py -l ligante.pdb -U "
```

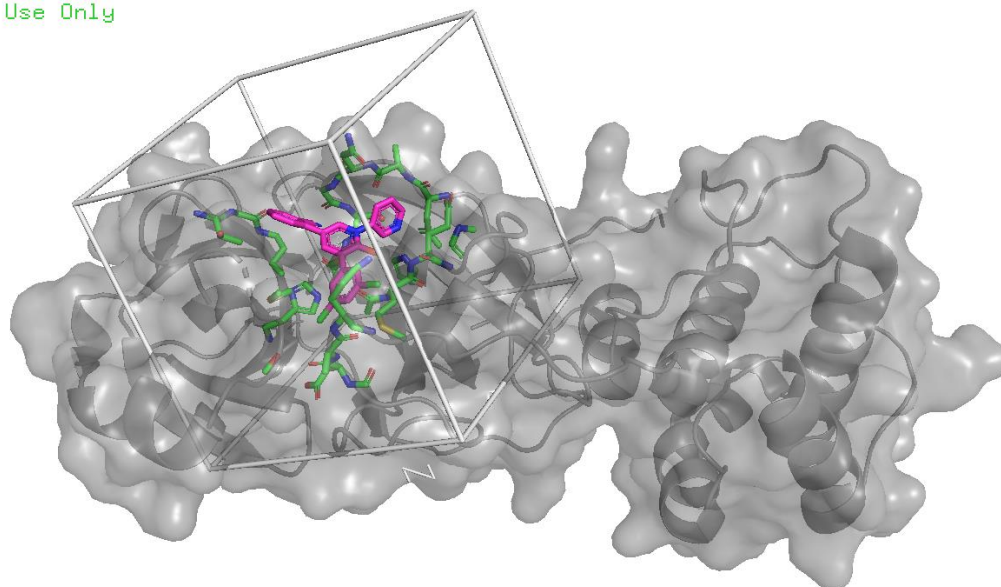
OBS. 4: -U " pede para que o script mantenha todos os hidrogênios do ligante. Tirando essa "flag", apenas os hidrogênios polares permaneceriam.

- 8) O último arquivo de entrada necessário é o arquivo com os parâmetros do **Vina**. É um simples arquivo de texto (vamos chamar de **config.txt**) contendo os seguintes parâmetros:

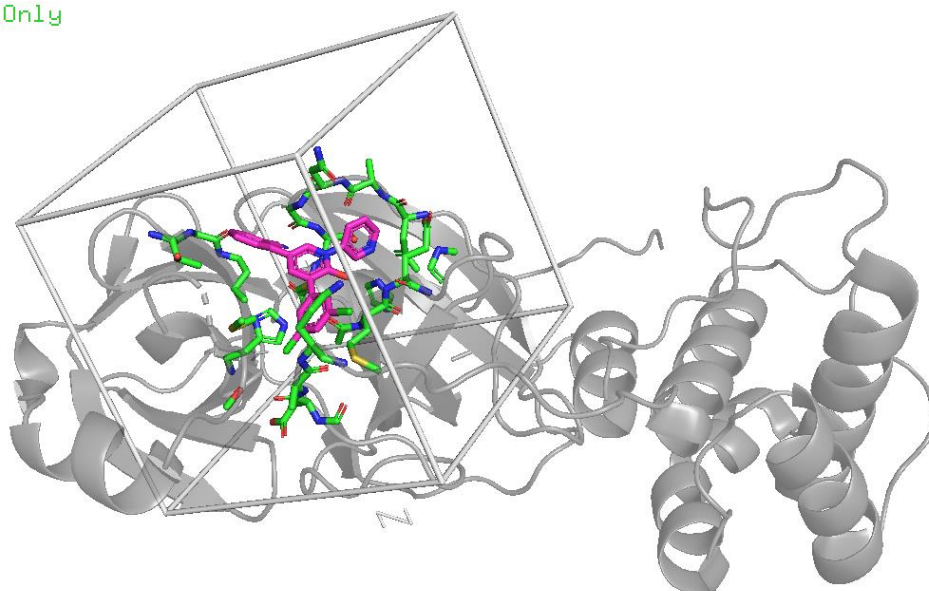
receptor = receptor.pdbqt	#Nome do arquivo PDBQT do receptor
ligand = ligante.pdbqt	#Nome do arquivo PDBQT do ligante
center_x = 8.417	#Coordenada X do centro da caixa (aqui o centro do ligante)
center_y = -0.06	#Coordenada Y do centro da caixa (aqui o centro do ligante)
center_z = 22.273	#Coordenada Z do centro da caixa (aqui o centro do ligante)
size_x = 25.00	#Tamanho da caixa para o eixo X
size_y = 25.00	#Tamanho da caixa para o eixo Y
size_z = 25.00	#Tamanho da caixa para o eixo Z
out = docked_ligante.pdbqt	#Nome do arquivo de saída com as poses do docking
log = docked.log	#Arquivo de log com informações da saída do docking
num_modes = 10	#Quantidade de poses

Center_x, center_y e center_z são as coordenadas do centro de massa do ligante calculados anteriormente no Pymol. Isso quer dizer que o programa terá sua "região de busca" centrada nessas coordenadas. Para delimitar uma região que compreenda parte da proteína ou um sítio de interesse temos os parâmetros size_x, size_y e size_z, que no nosso caso, estende **25 Å** para cada eixo desse ponto formando uma caixa cúbica nessa região da proteína. A região para docking será então:

For Educational Use Only



For Educational Use Only



9) Vamos criar o arquivo config.txt usando o programa **Vi**:

```
$ vi config.txt
```

Para entrar em modo de edição temos que apertar a **letra I** no teclado. Vai aparecer -- **INSERT** -- no canto inferior da tela. Copie (ou digite) as informações anteriores. Para copiar **Ctrl+C** da linha receptor até a linha num_modes, para colar clicar com **o direito do mouse**.

```

receptor = receptor.pdbqt      #Nome do arquivo PDBQT do receptor
ligand = ligante.pdbqt         #Nome do arquivo PDBQT do ligante
center_x = 8.417              #Coordenada X do centro da caixa (aqui o centro do ligante)
center_y = -0.06              #Coordenada Y do centro da caixa (aqui o centro do ligante)
center_z = 22.273            #Coordenada Z do centro da caixa (aqui o centro do ligante)
size_x = 25.00               #Tamanho da caixa para o eixo X
size_y = 25.00               #Tamanho da caixa para o eixo Y
size_z = 25.00               #Tamanho da caixa para o eixo Z
out = docked_ligante.pdbqt    #Nome do arquivo de saída com as poses do docking
log = docked.log              #Arquivo de log com informações da saída do docking
num_modes = 10                #Quantidade de poses

"config.txt" 12L, 700C

```

Para salvar, temos que sair do modo de inserção com **ESC** e depois digitar : **(Shift+:)** e **wq**.

10) Agora que temos todos os arquivos necessário, podemos rodar o **Vina**. Utilizamos o comando:

```
$ vina --config config.txt
```

A seguinte tela vai aparecer:

```

#####
# If you used AutoDock Vina in your work, please cite:
#
# O. Trott, A. J. Olson,
# AutoDock Vina: improving the speed and accuracy of docking
# with a new scoring function, efficient optimization and
# multithreading, Journal of Computational Chemistry 31 (2010)
# 455-461
#
# DOI 10.1002/jcc.21334
#
# Please see http://vina.scripps.edu for more information.
#####

Detected 4 CPUs
Reading input ... done.
Setting up the scoring function ... done.
Analyzing the binding site ... done.
Using random seed: 1941012439
Performing search ...
0% 10 20 30 40 50 60 70 80 90 100%
|----|----|----|----|----|----|----|----|----|
*****
done.
Refining results ... done.

mode |  affinity | dist from best mode
    | (kcal/mol) | rmsd l.b. | rmsd u.b.
-----+-----+-----+-----
1      -8.7      0.000      0.000
2      -8.6      1.052      2.447
3      -8.2      1.623      5.531
4      -8.2      1.723      5.610
5      -8.1      1.900      2.074
6      -8.0      2.909      6.183
7      -8.0      3.143      6.524
8      -7.9      3.091      6.643
9      -7.8      4.194      6.756
10     -7.5      3.382      7.239
Writing output ... done.

```

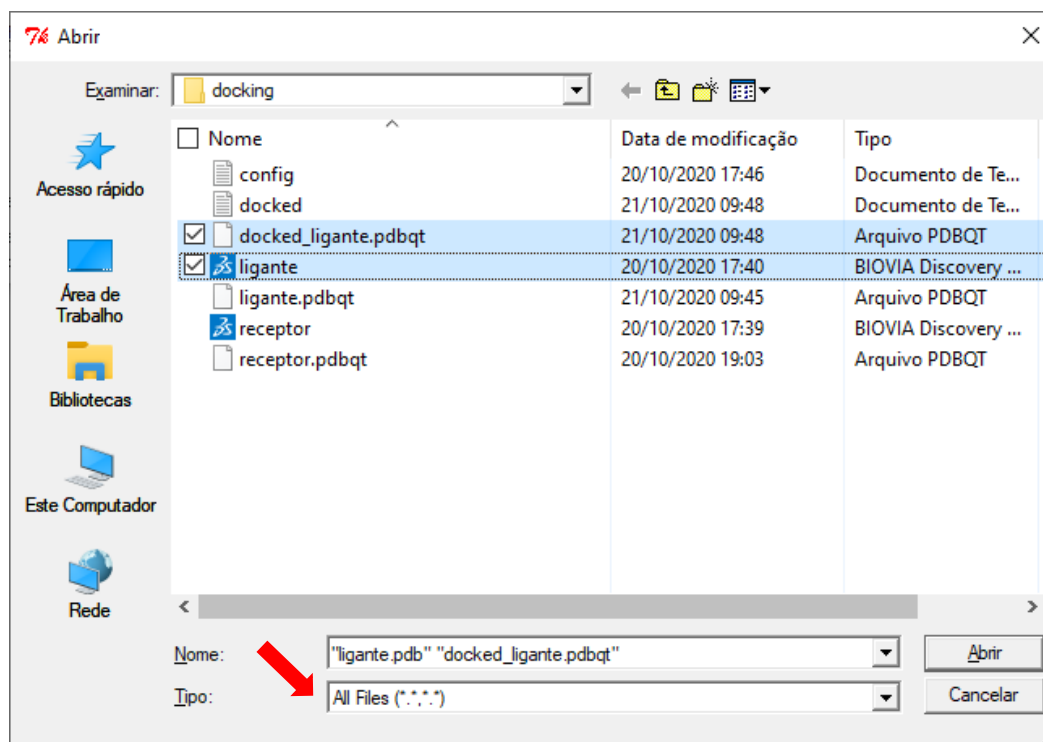
OBS. 5: O programa Vina é aleatório e a cada rodada ele escolhe um número aleatório,

dados pelo parâmetro chamado **seed**. Portanto, os resultados podem ser diferentes em cada rodada.

11) Pronto! Já temos os modos calculados pelo **Vina**.

Para verificar os resultados, vamos abrir o arquivo **docked_ligante.pdbqt** pelo PyMOL, junto com o ligante cristalográfico **ligante.pdb**.

No PyMOL vamos em **File > Open** navegar até a pasta onde estão os arquivos:



Caso o arquivo PDBQT não aparecer, mude o **Tipo** para **All Files**. E selecione os arquivos.

As poses abrem todas juntas na visualização **docked_ligante.pdbqt**. Para ir passando por elas clique no botão **>** no canto direito inferior.



12) Olhando todas as **poses**, a que mais se aproximou do ligante cristalográfico foi a primeira (melhor valor de pontuação **-8.7 kcal/mol**). Ou seja, temos aqui um exemplo de sucesso no **re-docking**, onde o programa encontra a pose cristalográfica e ranqueai como sendo o melhor de todos os modos calculados.

13) Para confirmarmos o sucesso do **re-docking**, calculamos o valor de RMSD das primeiras poses. Se o valor do RMSD estiver abaixo de 2.0 Å para a primeira pose, temos um sucesso tanto da função de pontuação quando do algoritmo de busca. Porém, se a primeira pose estiver acima de 2.0 Å, enquanto as poses de colocação próxima possuírem RMSD abaixo, temos uma falha na pontuação.

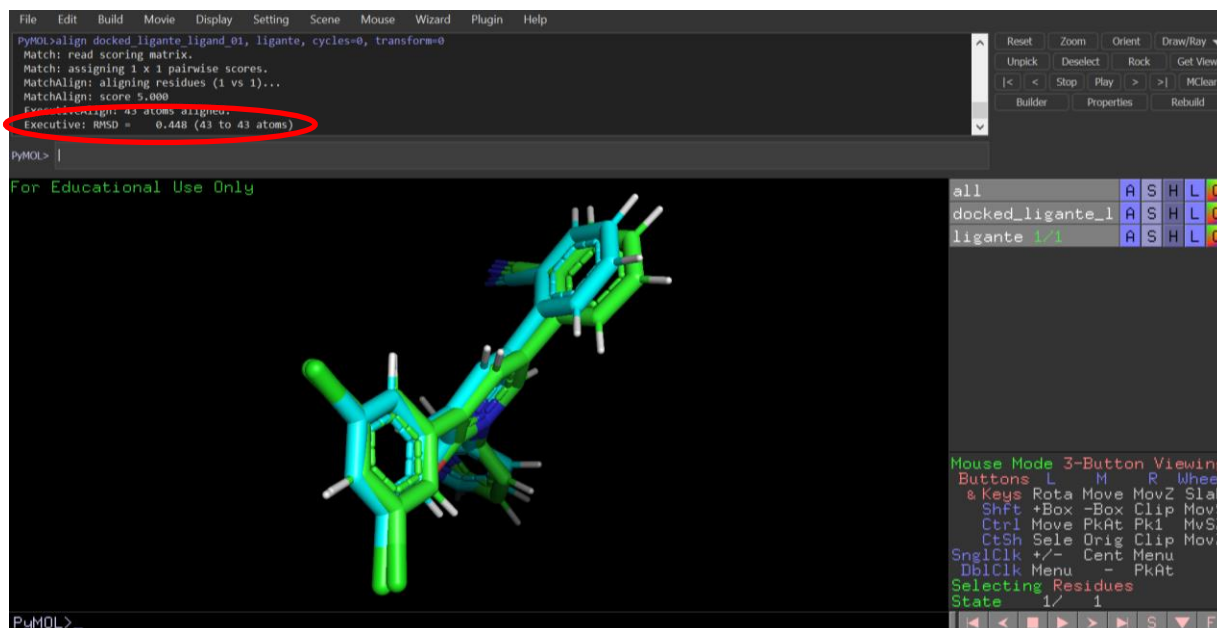
Voltando ao **TERMINAL**, ou seja, fora do **PyMOL**. Vamos primeiro separar cada uma das poses com o programa **vina_split**:

```
$ vina_split --input docked_ligante.pdbqt
```

```
docked_ligante_ligand_01.pdbqt
docked_ligante_ligand_02.pdbqt
docked_ligante_ligand_03.pdbqt
docked_ligante_ligand_04.pdbqt
docked_ligante_ligand_05.pdbqt
docked_ligante_ligand_06.pdbqt
docked_ligante_ligand_07.pdbqt
docked_ligante_ligand_08.pdbqt
docked_ligante_ligand_09.pdbqt
docked_ligante_ligand_10.pdbqt
```

Vamos abrir os arquivos **ligante.pdbqt** e o arquivo **docked_ligante_ligand_01.pdbqt** no **PyMOL**. Podemos utilizar o seguinte comando:

```
PyMOL> align docked_ligante_ligand_01, ligante, cycles=0, transform=0
```



OBS. 6: A verificação de RMSD só pode ser feita quando temos um ligante cristal para comparar. Em um experimento de *docking* com ligantes novos esse parâmetro não pode ser calculado. Porém, as interações formadas entre proteína e ligante podem (e devem) ser analisadas e exploradas.

- 14) Vamos analisar as interações intermoleculares entre essa e a enzima. Abra o arquivo do receptor. Em **File > Open** navegar até a pasta onde estão os arquivos. Selecione o arquivo **receptor.pdb**

Selecione a **pose** no menu de seleção. No menu **Actions (A)** da seleção, clique em

Find – polar contacts – to any atoms

Alguma interação foi igual a vista entre ligante do cristal e receptor?

Você pode fazer a mesma coisa para o ligante do cristal e mudar a cor no menu **Color (C)**.

- 15) Salve (**File > Save Session**) como **Mpro_dock1.pse**. Feche o **PyMOL**.

Parte 3 – Cross-docking utilizando o Autodock Vina

Continuando exemplos de atracamento molecular, vamos utilizar a metodologia de **cross-docking**. A metodologia consiste em tentar reproduzir a posição adquirida pelo ligante em uma estrutura cristalográfica diferente. Ou, trocar os ligantes de suas estruturas nativas para verificar se o programa consegue reproduzir a mesma pose em ambientes diferentes. Similar ao **re-docking**, também podemos comparar interações e o RMSD entre pose e posição cristalográfica do ligante.

- 1) Abra o **PyMOL** e a estrutura **receptor.pdb**.
- 2) Abra a estrutura PDB **5R82**, um outro complexo da protease principal do SARS-CoV-2 com um fragmento de ligante:

```
PyMOL> fetch 5R82
```

- 3) Vamos alinhar as duas estruturas para as coordenadas do novo receptor coincidir com o receptor.pdb:

```
PyMOL> align 5R82, receptor
```

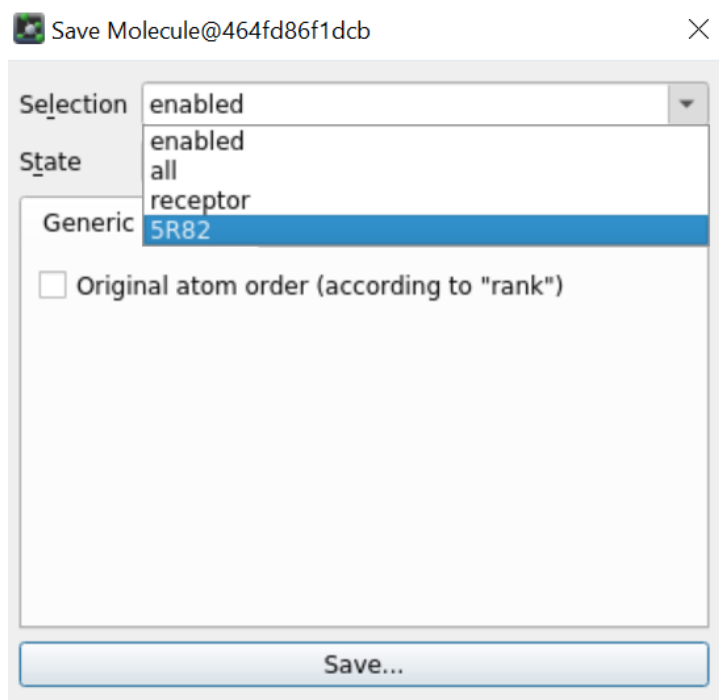
- 4) Como queremos apenas o receptor, vamos remover as águas e os ligantes. E adicionar hidrogênios:

```
PyMOL> remove resn hoh
```

```
PyMOL> remove het
```

```
PyMOL> h_add
```

- 5) Vamos salvar esse novo **receptor** em **File > Export Molecule**



Dar o nome de **novo_receptor** para a proteína.

File name:

Files of type:

6) Podemos então rodar no terminal a criação do PDBQT desse PDB

```
$ pythonsh $mglscripsts/prepare_receptor4.py -r novo_receptor.pdb
```

Ao invés de criar um novo arquivo config.txt, vamos copiar o existente com um novo arquivo:

```
$ cp config.txt novo_config.txt
```

Vamos editar apenas a linha de receptor desse novo arquivo de configuração:

```
$ vi novo_config.txt
```

```
receptor = novo_receptor.pdbqt      #Nome do arquivo PDBQT do receptor
ligand = ligante.pdbqt              #Nome do arquivo PDBQT do ligante
center_x = 8.417                    #Coordenada X do centro da caixa (aqui o centro do ligante)
center_y = -0.06                     #Coordenada Y do centro da caixa (aqui o centro do ligante)
center_z = 22.273                   #Coordenada Z do centro da caixa (aqui o centro do ligante)
size_x = 25.00                      #Tamanho da caixa para o eixo X
size_y = 25.00                      #Tamanho da caixa para o eixo Y
size_z = 25.00                      #Tamanho da caixa para o eixo Z
out = novo_docked_ligante.pdbqt     #Nome do arquivo de saída com as poses do docking
log = novo_docked.log               #Arquivo de log com informações da saída do docking
num_modes = 10                     #Quantidade de poses
```

-- INSERT --

Lembrando que para entrar de modo de edição temos que digitar a tecla I (-- **INSERT** --). Modificar as linhas:

receptor = receptor.pdbqt para receptor = novo receptor.pdbqt

out = docked_ligante.pdbqt para out = novo_docked_ligante.pdbqt

log = docked.log para log = novo_docked.log

Sair de modo de edição com **ESC** e salvar com **:wq**.

7) Depois de modificar os arquivos para não sobrescrever os anteriores podemos rodar o **Vina**:

```
$ vina --config novo_config.txt
```



```
#####
# If you used AutoDock Vina in your work, please cite: #
# #
# O. Trott, A. J. Olson, #
# AutoDock Vina: improving the speed and accuracy of docking #
# with a new scoring function, efficient optimization and #
# multithreading, Journal of Computational Chemistry 31 (2010) #
# 455-461 #
# #
# DOI 10.1002/jcc.21334 #
# #
# Please see http://vina.scripps.edu for more information. #
#####

Detected 8 CPUs
Reading input ... done.
Setting up the scoring function ... done.
Analyzing the binding site ... done.
Using random seed: 487241060
Performing search ...
0% 10 20 30 40 50 60 70 80 90 100%
|----|----|----|----|----|----|----|----|----|
*****
done.
Refining results ... done.

mode | affinity | dist from best mode
      | (kcal/mol) | rmsd l.b. | rmsd u.b.
-----+-----+-----+-----
1      -7.7      0.000      0.000
2      -7.6      0.064      1.690
3      -7.6      2.793      5.906
4      -7.6      1.802      5.578
5      -7.1      6.891     10.335
6      -7.0      3.763      6.196
7      -6.7      3.488      5.882
8      -6.7      5.246      7.087
9      -6.6      6.788      9.446
10     -6.6      6.808     10.094
Writing output ... done.
```

Notemos que o valor de score aumentou da pontuação de **-8.7 kcal/mol** para **-7.7 kcal/mol**. Portanto podemos esperar que a pose ou as interações alcançadas seja diferente do resultado anterior.

- 8) Vamos voltar para o **PyMOL** e para verificar os resultados. Vamos abrir o arquivo **novo_docked_ligante.pdbqt**, junto com o ligante cristalográfico **ligante.pdb**.



Olhando todos os resultados podemos ver que nenhuma das poses reproduziu a

posição cristalográfica. Apesar, do composto terem docado em uma posição muito próxima, as poses de melhor pontuação estão invertidas.

- 9) Vamos voltar ao **TERMINAL**, ou seja, fora do **PyMOL** para separar cada uma das poses com o programa **vina_split** e calcular o RMSD:

```
$ vina_split --input novo_docked_ligante.pdbqt
```

Vamos abrir os arquivos **ligante.pdbqt** e o arquivo **novo_docked_ligante_ligand_01.pdbqt** no **PyMOL**. Podemos utilizar o seguinte comando:

```
PyMOL> align novo_docked_ligante_ligand_01, ligante, cycles=0, transform=0
```



Podemos ver que o RMSD passa um pouco do corte de 2.0 Å, mostrando que para um novo ambiente o programa não teve muito sucesso em reproduzir a pose original. Porém, o *cross-docking* é uma avaliação mais complicada para o programa, pois com o receptor rígido se perde o efeito de encaixe induzido produzido pelo ligante no ambiente nativo.

Parte 4 – Triagem virtual com Autodock Vina

Para essa parte do Tutorial, vamos fazer uma pequena triagem virtual com poucos compostos para exemplificar de forma rápida o processo. Porém, uma triagem precisa de no mínimo centenas de compostos. Existem muitas maneiras de escolher os compostos para fazer uma triagem. Os compostos podem ser originados do próprio laboratório, de catálogos disponíveis para compra, ou compostos já testados para outros fins. Com isso, a primeira etapa de uma triagem é escolher e preparar todos esses compostos de forma uniforme. Como normalmente são muitos compostos, se faz necessário a utilização de scripts de preparação ou de base de dados especializadas onde a estrutura 3D (ou 2D) do ligante pode ser baixada. Base de dados como ZINC15, PubChem e ChEMBL são muito utilizadas para esse propósito. Nesse exemplo vamos usar o ZINC15.

1) Abra o **ZINC15** (<https://zinc15.docking.org/>)

ZINC
Substances
Catalogs
Tranches
Biological
More
About

ZINC15

Welcome to ZINC, a free database of commercially-available compounds for virtual screening. ZINC contains over 230 million purchasable compounds in ready-to-dock, 3D formats. ZINC also contains over 750 million purchasable compounds you can search for analogs in under a minute.

ZINC is provided by the [Irwin](#) and [Shoichet](#) Laboratories in the Department of Pharmaceutical Chemistry at the University of California, San Francisco (UCSF). We thank [NIGMS](#) for financial support (GM71896).

To cite ZINC, please reference: Sterling and Irwin, *J. Chem. Inf. Model.*, 2015 <http://pubs.acs.org/doi/abs/10.1021/acs.jcim.5b00559>. You may also wish to cite our previous papers: Irwin, Sterling, Mysinger, Bolstad and Coleman, *J. Chem. Inf. Model.*, 2012 DOI: [10.1021/ci3001277](https://doi.org/10.1021/ci3001277) or Irwin and Shoichet, *J. Chem. Inf. Model.* 2005;45(1):177-82 [PDF](#), [DOI](#).

Em **Catalogs**, digitar **FDA** para buscar apenas compostos aprovados pelo *U.S. Food and Drug Administration* e clicar em **Search**:

Catalogs

Help
Examples
Browse
Table
Subsets


fda

Search


Catalogs contain [substances](#), which inherit properties of purchasability, biogenicity and bioactivity from their catalog membership.

Vai aparecer dois catálogos:

«
1
»
2
Download
Home / catalogs
Filters
fda



FDA-approved drugs (via DSSTOX)



DrugBank FDA only

«
1
»
2
Download
Home / catalogs
Filters
fda

Vamos escolher e clicar no segundo **DrugBank FDA Only**.

DrugBank FDA only
In: annotated fda

Contact Information

Phone	no phone
Fax	no fax
Website	www.drugbank.ca
Email	not for sale

Catalog Properties

Purchasability	Annotated
Building Blocks	No
Activity Level	FDA Approved
Biogenicity Level	Unspecified

Last ZINC Import

Version	2018-02-20
Last Loaded On	2018-02-20
Original Catalog Size	1495
Compounds Removed	249
Est. Sole Supplier	0
Depleted Entries	Unknown (Browse)

Useful Links

Browse Substances

Browse Catalog Items

Browse Protomers

Genes (With active compounds in this catalog)

Unique Substances

No canto direito vamos clicar em **Browse Substances**:

The interface shows a grid of 12 chemical compounds, each with a name and a chemical structure:

- ZINC53 Aspirin
- ZINC61 Baclofen
- ZINC83
- ZINC96 Dexbrompheniramine
- ZINC122 Carbinoxamine
- ZINC128 Carteolol
- ZINC196 Cyclopentolate
- ZINC215 Diethylpropion
- ZINC242 Doxylamine
- ZINC257 Esmolol
- ZINC271 Ethotoin
- ZINC301 Fenoprofen

Vamos escolher 3 substâncias e baixar cada uma delas.

- 2) Podemos escolher **Aspirin (ZINC53)**, **Baclofen (ZINC61)** e **Dexbrompheniramine (ZINC96)**. Clique em cada uma das páginas e em **Available 3D Representations** clicar no **ícone de Download**:

ZINC53 (Aspirin)

In: anodyne bb fda for-sale in-stock natural-products
Google Wikipedia PubMed

Added	Availability	Since	Mwt	logP	Download
2005-09-27	In-Stock	2015-08-07	180.159	1.31	

Mol Formula	Rings	Heavy Atoms	Hetero Atoms	Fraction sp ³	Tranche
C ₉ H ₈ O ₄	1	13	4	0.11	ADAA

SMILES: CC(=O)Oc1ccccc1C(=O)O

InChI: InChI=1S/C9H8O4/c1-6(10)13-8-5-3-2-4-7(8)9(11)12/h2-5H,1H3,(H,11,12)

InChI Key: BSYNRYMUTXBXSQ-UHFFFAOYSA-N

Available 3D Representations

pH range	Net charge	H-bond donors	H-bond acceptors	tPSA	Rotatable bonds	Apolar desolvation	Polar desolvation	Download
Reference	-1	0	4	66	2	6.58	-56.82	

Vai aparecer o seguinte:

The screenshot shows a download menu with the following options:

- 6.82
- DB2
- Mol2
- Solvation

Clique em Mol2. Os arquivos baixados vão estar no formato mol2.gz, ou seja, compactados. Faça o download de todos os compostos escolhidos.

OBS.: Talvez seja necessário dar permissão para o navegador baixar esses

arquivos.

- 3) Copie ou mova os arquivos para a pasta de trabalho utilizada nas etapas anteriores. E no terminal vamos descompactar os arquivos:

```
$ gunzip 166296378.mol2.gz
```

```
$ gunzip 386595858.mol2.gz
```

```
$ gunzip 563211153.mol2.gz
```

O arquivo 166296378.mol2 corresponde a **Aspirin (ZINC53)**, 386595858.mol2 é **Baclofen (ZINC61)**, e 563211153.mol2 é o **Dexbrompheniramine (ZINC96)**. Vamos renomear para

```
$ mv 166296378.mol2 ligante1.mol2
```

```
$ mv 386595858.mol2 ligante2.mol2
```

```
$ mv 563211153.mol2 ligante3.mol2
```

- 4) Vamos utilizar em nossa triagem o primeiro receptor preparado, o da estrutura **7L10**. Com isso, podemos copiar o arquivo de configuração para o **Vina** anterior e modificar apenas algumas linhas:

```
$ cp config.txt triagem.txt
```

Vamos retirar as linhas **ligand**, **out** e **log**. E modificar o **num_modes = 10** para **num_modes = 1**. Abra o arquivo com o **vi**, modifique e salve.

```
$ vi triagem.txt
```

```
receptor = receptor.pdbqt          #Nome do arquivo PDBQT do receptor
center_x = 8.417                  #Coordenada X do centro da caixa (aqui o centro do ligante)
center_y = -0.06                  #Coordenada Y do centro da caixa (aqui o centro do ligante)
center_z = 22.273                 #Coordenada Z do centro da caixa (aqui o centro do ligante)
size_x = 25.00                    #Tamanho da caixa para o eixo X
size_y = 25.00                    #Tamanho da caixa para o eixo Y
size_z = 25.00                    #Tamanho da caixa para o eixo Z
num_modes = 1                     #Quantidade de poses
```

- 5) Para essa versão do **Vina**, a triagem é feita composto a composto. Então, vamos criar um script que vai preparar os compostos selecionados (nesse caso apenas 3) e fazer o docking com o **Vina**. Criamos um **bash script**:

```
$ bash vina_screen_local.sh
```

Esse script vai ler todos os ligantes e depois fazer o *docking* de cada um, colocando os resultados em uma pasta separada. Podemos digitar ou copiar as seguintes linhas:

```
#!/bin/bash
```

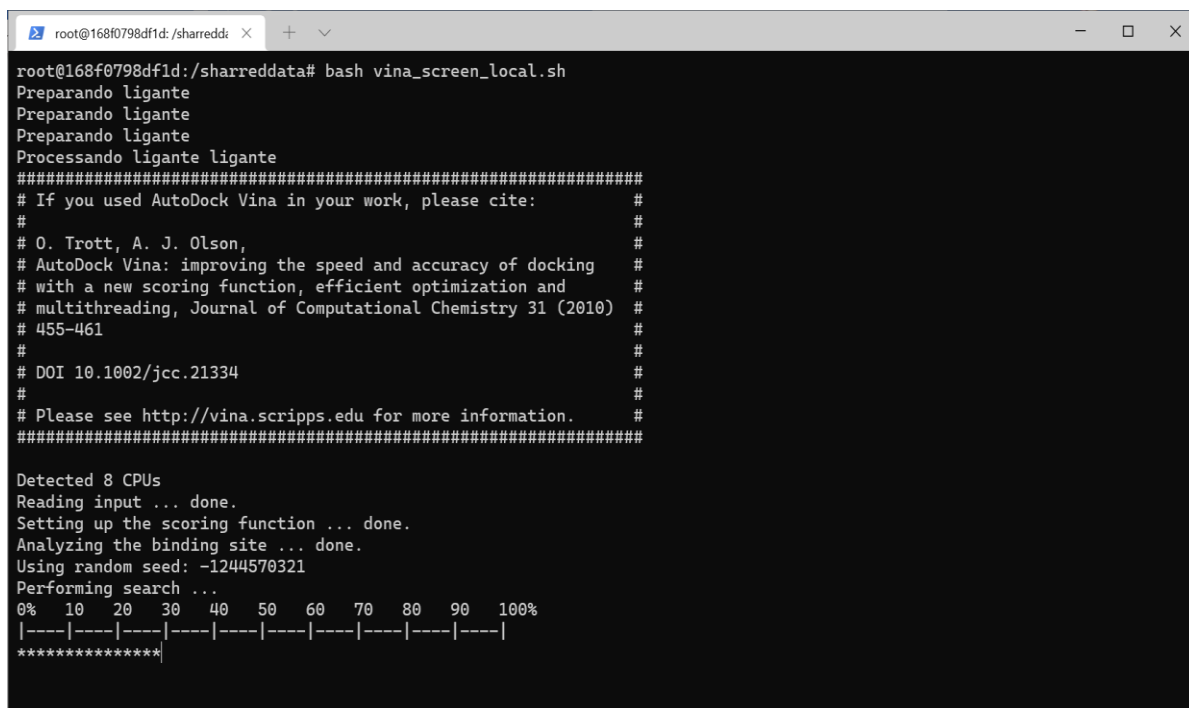
```
minhapasta=$(pwd) #recebe o caminho onde se encontram os arquivos e o script
n=3               #numero de ligantes no arquivo de ligantes
pythonsh='/programs/mgltools_x86_64Linux2_1.5.6/bin/python' #seta o python para
rodar os scripts MGLTools
```

```
for (( i=1;i<=n;i++ )); do
    echo Preparando ligante $i
    $pythonsh $mglscripts/'prepare_ligand4.py -l ligante$i.mol2 -U '' #prepara
cada ligante em pdbqt
done
```

```
#Faz o docking de cada um dos ligantes por vez e coloca o resultado em pastas
com o nome dos ligantes
for f in ligante*.pdbqt; do
    b=`basename $f .pdbqt`
    echo Processando ligante $b
    mkdir -p $b
    vina --config triagem.txt --ligand $f --out ${b}/out.pdbqt --log ${b}/log.txt
done
```

Depois de salvo, podemos rodar o script com

```
$ bash vina_screen_local.sh
```



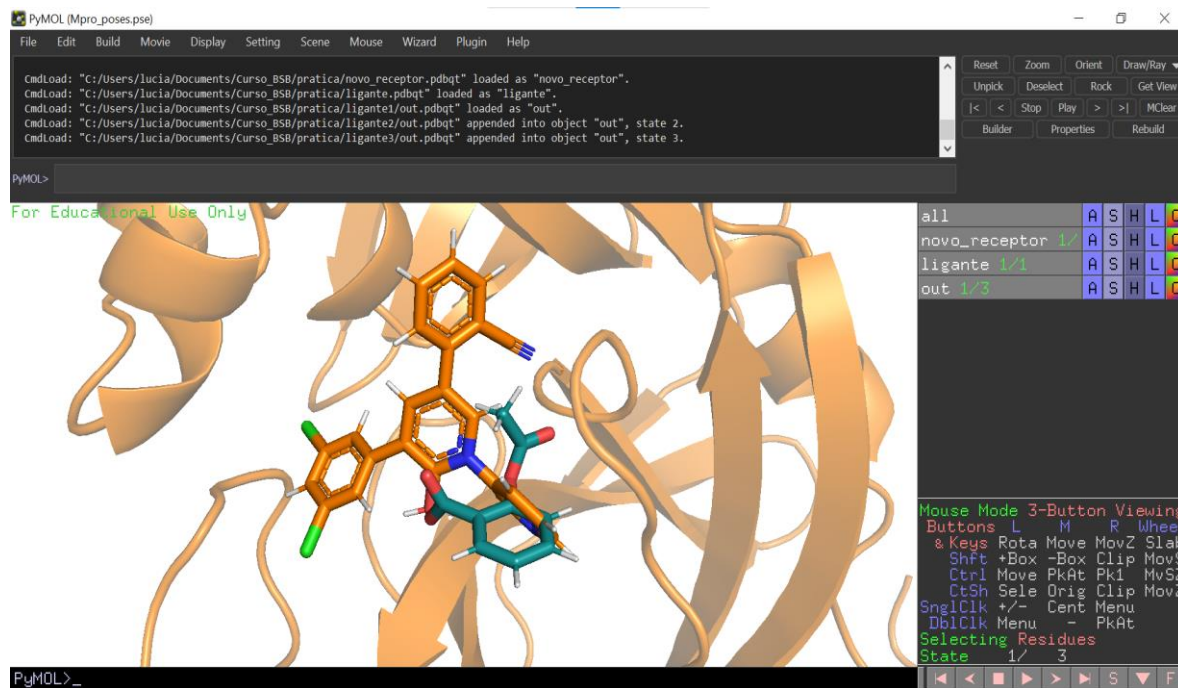
```
root@168f0798df1d: /sharreddata# bash vina_screen_local.sh
Preparando ligante
Preparando ligante
Preparando ligante
Processando ligante ligante
#####
# If you used AutoDock Vina in your work, please cite:      #
#                                                            #
# O. Trott, A. J. Olson,                                     #
# AutoDock Vina: improving the speed and accuracy of docking #
# with a new scoring function, efficient optimization and    #
# multithreading, Journal of Computational Chemistry 31 (2010) #
# 455-461                                                      #
#                                                            #
# DOI 10.1002/jcc.21334                                       #
#                                                            #
# Please see http://vina.scripps.edu for more information.   #
#####
Detected 8 CPUs
Reading input ... done.
Setting up the scoring function ... done.
Analyzing the binding site ... done.
Using random seed: -1244570321
Performing search ...
0% 10 20 30 40 50 60 70 80 90 100%
|----|----|----|----|----|----|----|----|
*****
```

OBS.: O script não está muito refinado (ou perfeito) então acabamos fazendo o *docking* do ligante.pdbqt junto. Um bom exercício é refazer esse último loop.

6) Pronto. Notemos que todos os compostos novos alcançaram score abaixo do

ligante cristalográfico de **~5.5 kcal/mol**. Podemos abrir cada um dos resultados no **PyMOL** junto com a estrutura do receptor (e o ligante cristalográfico para comparação) para ver a como eles foram posicionados e as interações formadas:

7)



OBS.: Como os compostos têm o mesmo nome, eles são inseridos na mesma representação no **PyMOL** (mais uma coisa a melhor no script).