

Vina Docking Tutorial

Dr. S. Eagon

California Polytechnic State University

Note: This is a tutorial on how to use the Vina program to perform structure-based docking, also called *in silico* docking and/or virtual screening. I will use the term “docking” moving forward to save time. Structure-based docking means that you have a protein target that you hold rigid, then use a program to let a single molecule “wiggle” into the “best” conformation, as determined by an energy scoring function. This is different from ligand-based screening, which means a program tries to add/delete/change groups on a given molecule to find a derivative that best fits a rigid binding pocket. This is also different from molecular dynamics (MD), which allows a molecule AND a protein to move about in a simulated solvent environment.

Software you will need

Before you begin, you need to install the following programs on your computer:

1. ChimeraX
2. Cygwin (for PCs only, Macs will use the terminal line)

Cygwin (for PCs only) can be downloaded here (just do the standard installation, takes around 100MB):

<https://www.cygwin.com/install.html>

Note: Anytime you download something, you take full responsibility for what happens to your computer. While I have used these programs for years and never had any issues, I am not responsible for issues that may arise on your computer due to a bad download. *Caveat emptor*

Unix command line interfaces

In order to use a lot of these programs, we will be using a Unix-style command line interface. Using command lines gives you much more control over programs, and you will find it used extensively on most high performance computing clusters.

Macs OS is based on a Unix kernel, and so behaves much like a Linux system, but with a Windows-like interface. This is why when you bring up the “Terminal” app, you are essentially opening a Unix terminal. So, all Mac users will enter line commands from the terminal window. PCs run DOS instead of Unix, and DOS is much harder to work with. That’s why PC users will install Cygwin. This program opens up a window that gives you a Unix-like environment. It is essentially Windows “pretending” to be Unix. So, all PC users will enter commands from the Cygwin window.

Moving forward, all line commands will be indicated with a \$ sign, like this:

- \$ whoami
 - If you type “whoami” into your command line, you’ll get a response that indicates your user name on the machine

Basic Unix line commands

Take a look at the “**Basic Line Commands**” document for a list of useful commands and examples. It will take some time for you to get used to a Unix environment, but mastery comes with practice. PC users execute these from a Cygwin window, and Mac users from the terminal window.

If you ever want to look up how to do something with a Unix line command, then an online search like “How do I do X in unix?” will usually give a list of results with examples.

Warning: When you execute things by line command, your commands *cannot be undone*. You are, in essence, giving commands to the heart of the system. Do NOT try random commands haphazardly. Be especially careful with the “rm” (remove) command, which deletes things. The most dangerous command is:

- `$ rm -rf *`

This command tells your computer to delete everything in the current folder, and to override any permissions. If you execute this from your C: (or home) drive, **you will destroy every folder & file on your computer, including the operating system files!** With great power comes great responsibility!

Whenever you are using line commands, remember that the computer won’t understand you unless the command is spelled and formatted exactly the right way. Anytime a command fails, be sure to double-check it to make sure it was spelled correctly.

Step 1: Create a folder on the desktop and add the initial files

Make a folder on the desktop called **DockingJob**. You can call this folder something else, but avoid using spaces, as this will cause problems later. If you want to add a space in the name, use the underscore and name it something like **Docking_Job**. For now, I’d recommend you just call it **DockingJob**.

In your **DockingJob** folder, place the following files:

1. The **1stp.pdb** file (can be downloaded from PDB)
2. The **1stp_DOCK.pdbqt** file
3. The correct “tarball” folder containing the Vina program and screening script
 - There’s a different one for Windows vs. Mac. They’ll be named either:
 - **Vina_files_Windows.tar.gz** (for PCs users only)
 - **Vina_files_Mac.tar.gz** (for Mac users only)
 - A “tarball” is a compressed archive that we’ll unpack later. Very similar to a zip file.
 - You don’t need both of these tarballs--just the one for your OS

The PDBQT file is created by me and is not discussed in this tutorial. It is a specific file type only used by docking software, and requires additional programming skills to prepare, as well as the program OpenBabel (which is free). If you are interested in how these are created (or want to create your own), you can visit my research webpage here to learn more:

https://web.calpoly.edu/~seagon/virtual_screening.html

Step 2: Find the coordinates of the ligand using ChimeraX

Before we dock any molecules, we need to determine the coordinates of the ligand in the crystal space. We will use these coordinates when we tell the program where to try to get our molecules to bind.

- Open the **1stp.pdb** file in ChimeraX
- Find the center of the biotin ligand using the following command:
 - \$ measure center :BTN
 - You should get an output in the log pane that looks like this:

```
measure center :BTN
Center of mass of 16 atoms = (11.09, 1.74, -10.63)
```

This result tells us the coordinates of the center of the BTN (biotin) ligand in x,y,z form. In other words, the coordinates of the center of BTN is:

x = 11.09

y = 1.74

z = -10.63

You need to save these numbers in a file somewhere or write them down. We'll need them when we're setting up the parameters for the docking program.

Note: If you're wondering where these numbers come from, they arise from the PDB file. If you open a PDB file with a text editor and scroll down past the REMARKS lines, you'll see long lists of coordinates like this:

```
ATOM  1 N  ALA A 13   22.637  5.768 11.762  1.00 44.60   N
ATOM  2 CA ALA A 13   23.655  4.852 11.146  1.00 44.15   C
ATOM  3 C  ALA A 13   24.276  5.552  9.942  1.00 43.61   C
ATOM  4 O  ALA A 13   23.942  5.192  8.794  1.00 43.25   O
ATOM  5 CB ALA A 13   22.855  3.616 10.672  1.00 44.78   C
ATOM  6 N  GLU A 14   25.105  6.547 10.225  1.00 43.22   N
...(etc)
```

A PDB file is just a collection of points (ATOM 1, ATOM 2, etc.) with names (ALA A 13), coordinates (the first 3 numbers after the name) and the type of atom (at the end). For the example above, ATOM 1 is a Nitrogen atom in residue 13, an alanine, on chain A, with the coordinates x = 22,637, y = 5.768, and z = 11.762. The coordinate grid is centered on the crystal unit center, which is determined by the individual that solved the crystal structure.

At the bottom of the PDB file, you'll see a series of CONECT lines that tell ChimeraX how to stitch these atoms and residues together to form the peptide chain (yes, the word CONECT is spelled incorrectly. Blame the original programmer, who was probably just being lazy)

Step 3: Preparing the Ligands folder

Next, we need to make a folder of ligands that we want docked to our protein. For this tutorial, we are going to try to dock biotin into streptavidin (the 1stp crystal structure). Since we already know that biotin binds streptavidin, this is a good way to make sure our docking simulations are working.

Important Note: Whenever you do a docking simulation, you should always use a positive control like this. In other words, be sure to dock a molecule you know binds your protein to make sure that everything is working. There's an old saying in computational chemistry that goes "garbage in, garbage out." This means the computer will always spit *something* out, but you need to have a known starting point, otherwise you might just end up with a bunch of garbage and not be able to tell the difference. This has happened to a lot of publications over the years.

- Start by creating a new folder called **Ligands** in the **DockingJobs** folder
 - Be sure to capitalize the "L" in **Ligands** here.
- Now we need to create a file for biotin that the docking program can use and place it in the **Ligands** folder. Our docking program, Vina, looks at molecules that are in PDBQT format. So, we need to make a PDBQT file for biotin. To do this, we are going to use an online server that you can find at this website:

<https://www.cheminfo.org/Chemistry/Cheminformatics/FormatConverter/index.html>

You'll should see something like this:

The screenshot shows the Cheminformatics Format Converter website interface. It features several key components labeled with red arrows and boxes:

- Input File Pane:** Located at the top left, it contains a table with one row and one column.
- Options:** A central panel with dropdown menus for 'Input format' (set to 'smi -- SMILES format') and 'Output format' (set to 'mol -- MDL MOL format'). It also includes checkboxes for 'Generate coordinates' (set to 'None') and 'Add / Delete hydrogens' (set to 'No change'), along with a 'pH to add hydrogens' field.
- Output File Pane:** Located at the top right, it contains a table with one row and one column.
- Chemical Drawing Editor:** A bottom-left panel with a toolbar and a text area for drawing molecules.
- Convert Button:** A button located below the 'Options' panel.
- Log Pane:** A bottom-right panel displaying a log of the conversion process, including a warning about missing 2D or 3D coordinates.

A yellow box in the center provides instructions on how to proceed: 1. Enter an input value, for example a SMILES like "CCCC"; 2. Select the "input format" for example "smi"; 3. Select an output format for example "mol"; 4. Click on "Convert".

This webpage will allow us to turn molecular structures into PDBQT files for docking.

- In your **Ligands** folder (which is in the **DockingJobs** folder), create a text file and call it **ligand_biotin.txt**
 - PC Users: You can create a text file with Notepad or similar program
 - Mac Users: You need to use the BBEdit app to make a text file
 - Mac Users -> Do NOT use the TextEdit app! This will save it as a .rtf file, and not a .txt file. This will cause your docking simulations to fail.
- Copy and paste the contents of the “Output pane” into the **ligand_biotin.txt** file.
- On the first line of your text file, replace the long SMILES string with just the name biotin so it looks like this:

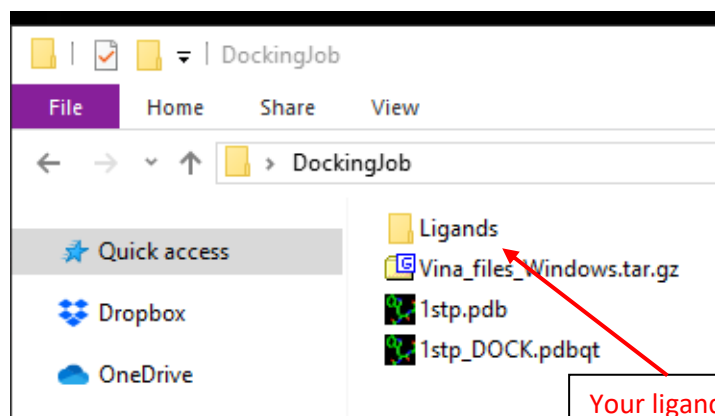
```
REMARK  Name = biotin
REMARK
REMARK
ROOT
ATOM      1  C  LIG      1    -0.410    0.607   -2.819    0.00    0.00    +0.108  C
ATOM      2  C  LIG      1    -0.949    0.363   -1.393    0.00    0.00    +0.069  C
ATOM      3  S  LIG      1    -2.690   -0.061   -1.733    0.00    0.00    -0.154  S
ATOM      4  C  LIG      1    -2.309   -1.013    0.000    0.00    0.00    +0.093  C
ATOM      5  C  LIG      1    -1.058   -0.378    0.000    0.00    0.00    +0.105  C
```

Put the name here

Note: Notice that the PDBQT file looks a lot like a PDB file in some ways! PDB files came first, with the “PDB” standing for “Protein Data Base” file. To run a docking simulation, the program must also know the relative charge at each atom (Q) and the atom type (T). So, PDB + charge (Q) + type (T) gives you a PDBQT file.

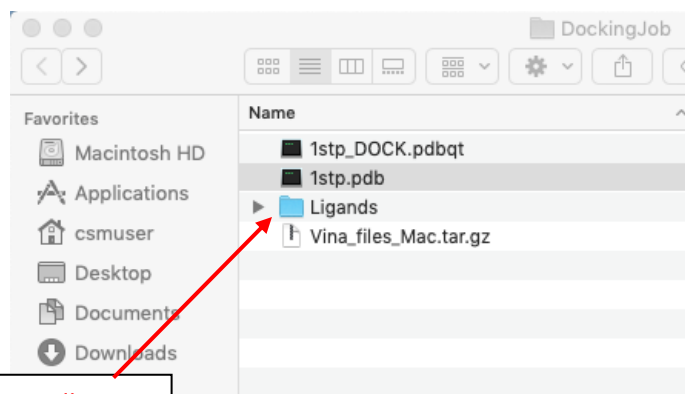
- Lastly, change the file extension of the text file to pdbqt. You can do this several ways:
 - Just change the file name by modifying “**ligand_biotin.txt**” to “**ligand_biotin.pdbqt**”
 - Ignore any warnings the computer gives about doing this.
 - Another way to change the file extension is through using the “Save As” command and changing the file extension
 - If you can’t see the file extensions (the .txt, .doc, .pdb stuff), you’ll need the change the settings on how you view your folder items to see them.
- You should now have a folder that looks like this:

PC Users:



Your ligand_biotin.pdbqt file is in the Ligands folder

Mac Users:

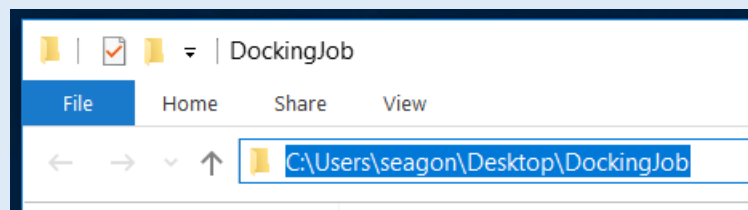


Step 4: Navigating by command line and unpacking the tarball

Now we need to start using line commands. In order to do this, we need to “point” the computer’s attention to the **DockingJobs** folder so it knows what you’re looking at, then execute our commands. Specifically, we’re going to tell it what our directory is (a “directory” is another name for a “folder” or “drive”). This differs depending on whether you’re on a Mac or PC:

PC USERS ONLY:

- We will have to use the “cd” (change directory) command to navigate to the **DockingJobs** folder. To do this, you must know the exact file path of the folder. The easiest way to do this is to click on the directory window in the open **DockingJob** folder to see something like this:



- The highlighted string shows the file path we need to enter!
 - Another way to do this is to right-click on the folder on your desktop then select the “Properties” option. On the first tab, you’ll see a “Location” which lists this same information
- Now, open the Cygwin window
- Use the “cd” command to change the directory. Using the example picture above, it would be this command:
 - `$ cd /cygdrive/c/Users/seagon/Desktop/DockingJob`
 - Pay attention to the direction of the slash there. Unix (and Cygwin) use the forward slash, while the PC (DOS) will display a backslash (like the folder view)
 - You must always start a directory with /cygdrive when using Cygwin. In other words, to reach the C: drive, you must use /cygdrive/c, not just /c.
- The TAB key is your friend here, as it is the autocomplete command. For instance, let’s say I start to type this:
 - `$ cd /cygdrive/Users/seagon/Desktop/Dock`
 - Then hit TAB:
 - `$ cd /cygdrive/Users/seagon/Desktop/DockingJob`
 - Note that the TAB key only works if there is only one autocomplete solution, so it takes a few letters to get it right. It’s not guessing, it’s looking at all possibilities to see if there is only one solution.
 - You can Use the TAB key at each step. For example, it will change “cyg” to “cygdrive” and so on.
- Spaces cause problems in file names, as spaces are interpreted as the end of a command. For instance, let’s say your username is Uncle Grandpa (instead of seagon in the previous example), then adding parenthesis to the file path will fix this like so:
 - `$ cd “/cygdrive/c/Users/Uncle Grandpa/Desktop/”`

- If you are successful, you'll see the directory displayed in Cygwin above the command line:

```
seagon@CSM-SEAGON-01 ~
$ cd /cygdrive/c/Users/seagon/Desktop/DockingJob/

seagon@CSM-SEAGON-01 /cygdrive/c/Users/seagon/Desktop/DockingJob
$ |
```

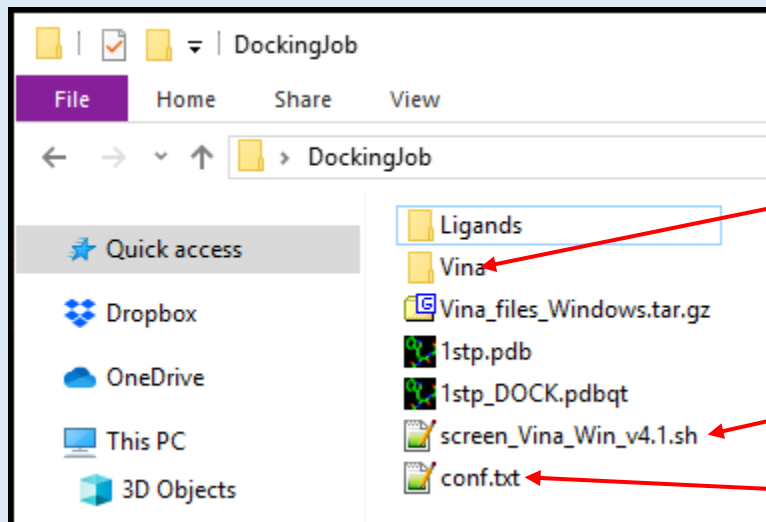
New directory!

- A good way to confirm that you are in the right folder is to list all contents of the current directory. You do that with the following command:

- \$ ls
 - The "ls" here is short for "list"
 - You should get something like this:

```
Dr. Eagon@Megaflop /cygdrive/c/Users/Dr. Eagon/Desktop/DockingJob
$ ls
1stp.pdb 1stp_DOCK.pdbqt Ligands Vina_files_Windows.tar.gz
```

- You can see this lists the contents of the **DockingJob** folder--our crystal PDB and PDBQT file, the tarball, and the **Ligands** folder
- Now we are ready to unpack the tarball! Do that using the following command:
 - \$ tar -xzf Vina_files_Windows.tar.gz
 - The "-xzf" are called flags, and tell the computer what you want to do. The "x" means "extract," the "z" means "g-zip compression" and the "f" means "filename," which is why you list the name of the tarball right after.
 - A fun way to remember this is "Xtract Ze File!"
 - Try using the TAB key to autocomplete the long file name. So, if you type:
 - \$ tar -xzf Vina_fi
 - Then hit the tab, you should get the full command:
 - \$ tar -xzf Vina_files_Windows.tar.gz
 - If you successfully unpacked the tarball, you should be able to see 3 new files in your **DockingJobs** folder, a **conf.txt** file, the **Vina** program folder and a **screen.sh** script:



The Vina program folder

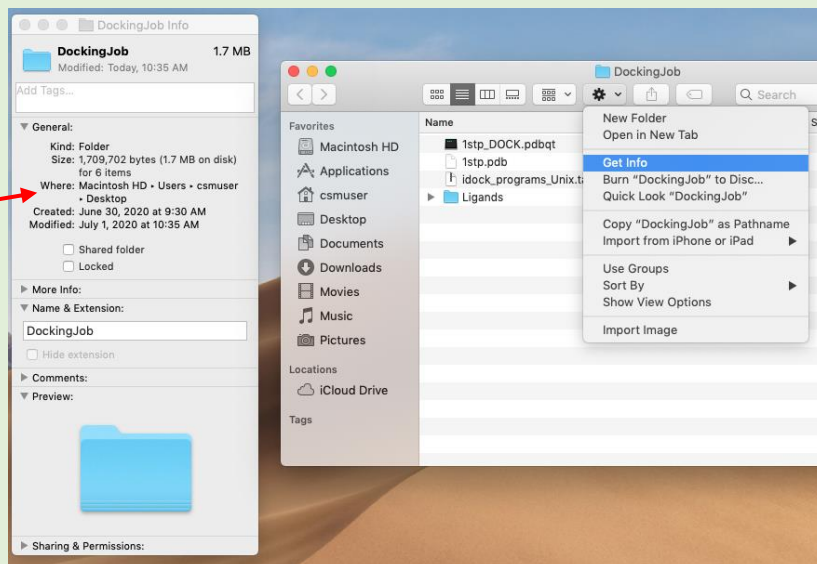
The screening script (will end with .sh)

conf. txt file

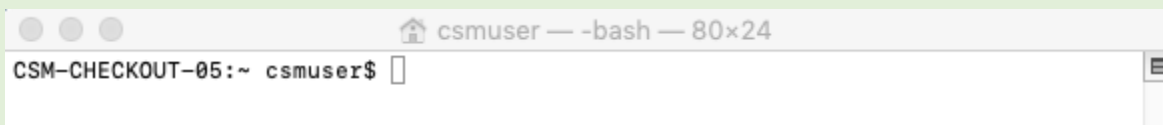
MAC USERS ONLY:

- We will have to use the “cd” (change directory) command to navigate to the **DockingJobs** folder. To do this, you must know the exact file path of the folder. The easiest way to do this is to click on the “settings” symbol (looks like a gear) in the **DockingJobs** folder, and select “Get Info” to see the file path displayed

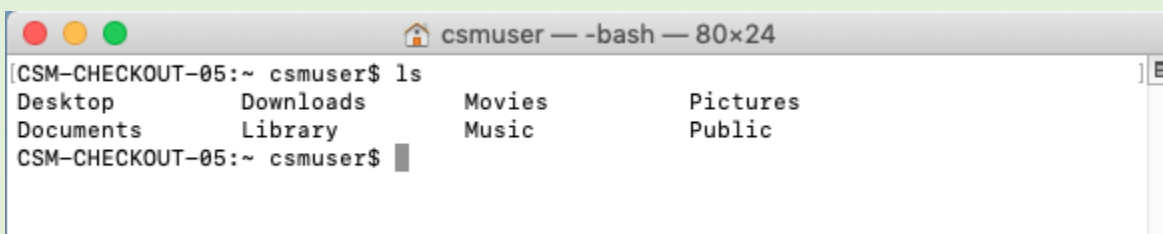
File path is displayed at the “Where” prompt



- The file path is displayed here as “Macintosh HD > Users > csmuser > Desktop”
 - This just tells us that the folder is in the Desktop, as it should be!
- Now, open the terminal window
 - You can find it by searching for “terminal” using the spyglass that appears by the time at the upper right of your screen
- Once the terminal appears, you should see something like this:



- Be default, Mac should start you out in your user directory. Confirm this by using the “list” command:
 - \$ ls



- Note that we can see the Desktop listed, along with other directories. We know that the **DockingJob** folder should be in the Desktop directory

- Now use the “cd” command to change the directory. Since we know the **DockingJob** folder is in the Desktop directory, we will use this command:
 - `$ cd Desktop/DockingJob`
- The TAB key is your friend here, as it is the autocomplete command. For instance, let’s say I start to type this:
 - `$ cd Desktop/Dock`
 - Then hit TAB and you’ll get:
 - `$ cd Desktop/DockingJob`
 - Note that the TAB key only works if there is only one autocomplete solution, so it takes a few letters to get it right. It’s not guessing, it’s looking at all possibilities to see if there is only one solution.
 - You can Use the TAB key at each step. For example, it will change “Desk” to “Desktop” and so on.
- Spaces cause problems in file names, as spaces are interpreted as the end of a command. For instance, let’s say you called your folder **Docking Stuff** (instead of **DockingJob** as instructed), then adding parenthesis to the file path will fix this like so:
 - `$ cd “Desktop/Docking Stuff”`
- If you are successful, you’ll see the directory displayed in front of the command prompt:

New
directory!

```

CSM-CHECKOUT-05:~ csmuser$ ls
Desktop      Downloads    Movies       Pictures
Documents    Library      Music        Public
CSM-CHECKOUT-05:~ csmuser$ cd Desktop/DockingJob/
CSM-CHECKOUT-05:DockingJob csmuser$
  
```

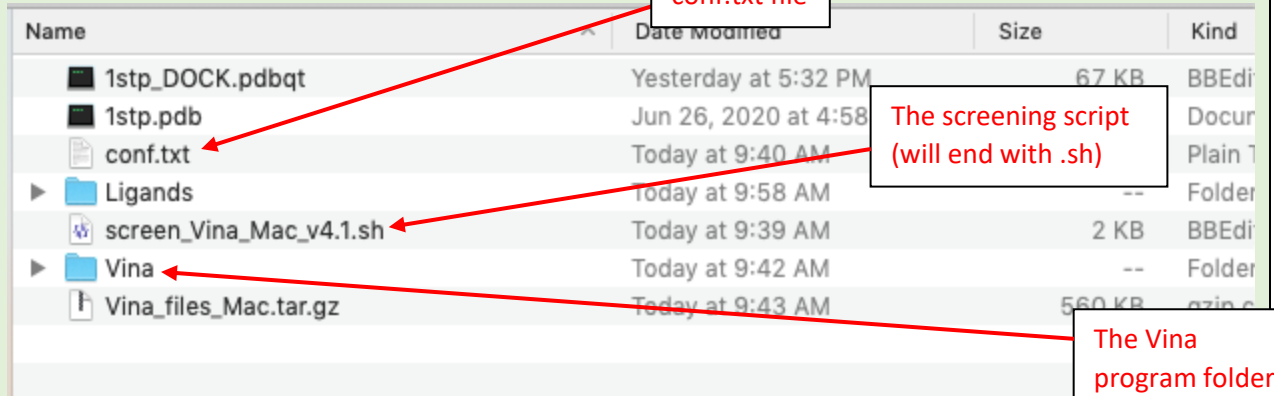
- A good way to confirm that you are in the right folder is to use the “list” command again:
 - `$ ls`
 - You should get something like this:


```

CSM-CHECKOUT-05:DockingJob csmuser$ ls
1stp.pdb          Ligands
1stp_DOCK.pdbqt   Vina_files_Mac.tar.gz
CSM-CHECKOUT-05:DockingJob csmuser$
          
```

 - You can see this lists the contents of the **DockingJob** folder--our crystal PDB and PDBQT file, the tarball, and the **Ligands** folder
- Now we are ready to unpack the tarball! Do that using the following command:
 - `$ tar -xzf Vina_files_Mac.tar.gz`
 - The “-xzf” are called flags, and tell the computer what you want to do. The “x” means “extract,” the “z” means “g-zip compression” and the “f” means “filename,” which is why you list the name of the tarball right after.
 - A fun way to remember this is “**X**tract **Z**e **F**ile!”
 - Try using the TAB key to autocomplete the long file name. So, if you type:
 - `$ tar -xzf Vina_fi`
 - Then hit the tab, you should get the full command:
 - `$ tar -xzf Vina_files_Mac.tar.gz`

- If you successfully unpacked the tarball, you should be able to see 3 new files in your **DockingJobs** folder, a **conf.txt** file, the **Vina** program folder and a **screen.sh** script:



Name	Date Modified	Size	Kind
1stp.Dock.pdbqt	Yesterday at 5:32 PM	67 KB	BBEdi
1stp.pdb	Jun 26, 2020 at 4:58		Docur
conf.txt	Today at 9:40 AM		Plain 1
▶ Ligands	Today at 9:58 AM	--	Folder
screen_Vina_Mac_v4.1.sh	Today at 9:39 AM	2 KB	BBEdi
▶ Vina	Today at 9:42 AM	--	Folder
Vina_files_Mac.tar.gz	Today at 9:43 AM	560 KB	gzi

Step 5: Setting up the configuration file

We are almost ready! The last thing we need to do before we run our simulations is to tell the Vina program what files to use and how we want it to run. Docking programs keep this information in something called the configuration file. The Vina program will automatically look for this information in the **conf.txt** file, so we need to set it up (note that “conf” is short for “configuration”).

- Open the **conf.txt** file with any text editor to see something like this:

```
receptor = #.pdbqt

center_x = #
center_y = #
center_z = #

size_x = #
size_y = #
size_z = #

cpu = 2
num_modes = 10
exhaustiveness = 64
```

- Note that the exact appearance of this file will depend on what kind of text editor you use, but you should see all of the same lines (I’m using Notepad++ here)
 - Everywhere you see the “#” signs, we need to provide information for the Vina program to run. Leave the other settings alone for now.
- Fill in the following information for our docking job to get something that looks like this:

```
receptor = 1stp_DOCK.pdbqt

center_x = 11.09
center_y = 1.74
center_z = -10.63

size_x = 20
size_y = 20
size_z = 20

cpu = 2
num_modes = 10
exhaustiveness = 64
```

- **Be sure to save this file so your changes will be preserved!**

- Here's what these parameters mean:
 - The "receptor" is the protein target as a PDBQT file. Since we're docking to the 1stp protein, we're going to use the **1stp_DOCK.pdbqt** file.
 - The "center" inputs tell Vina where the center of the search space is.
 - Note that these are the coordinates of the center of the biotin molecule that we determined in Step 2!
 - The "size" tells Vina how big our cubic search area is in Angstroms. We're telling it to search a 20 x 20 x 20 Å cube, which is large enough to accommodate drug-like molecules (anything less than 500 g/mol or so)
 - The "cpu" line tells the program how many logical cores from the CPU can be utilized to run the simulation.
 - The "num_modes" means the max number of times Vina will try to dock something
 - The "exhaustiveness" tells Vina how intensive to make the computations.

Note: The amount of time Vina takes depends greatly upon the "cpu" setting. The more CPU power you devote to the program, the faster it will go. However, if you utilize 100% of your CPUs to the docking job, this can cause all other programs you may be running (and even your OS) to crash.

A setting of "cpu = 2" is pretty low, but I've picked it so that any computer should be able to run it without too many problems. A good rule of thumb is to not devote more than 50% CPU power to any single job. There is a point, also, where too many cpu threads actually slow down the program. The sweet spot is usually around 4-8 cpu threads. If you want to see how many resources are available for your machine, the easiest way is to see how many logical cores you have at your disposal, and then set the "cpu" parameter to 50% of that value. The commands to figure out your number of logical CPU cores you have differs depending on your operating system:

PC USERS ONLY:

- Use the "number of processors" command from the Cygwin terminal:
 - \$ nproc
 - If the output is "8" then you have the ability to run 8 threads, so consider setting the "cpu" option to 4. If the output is "6" then set the "cpu" to 3, etc.
 - This command can be run in any directory from Cygwin.

MAC USERS ONLY:

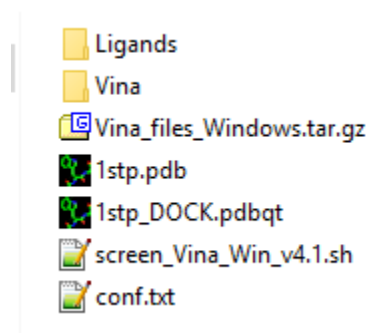
- Use the following command in the terminal window:
 - \$ sysctl -n hw.ncpu
 - If the output is "8" then you have the ability to run 8 threads, so consider setting the "cpu" option to 4. If the output is "6" then set the "cpu" to 3, etc.
 - This command can be run in any directory in the terminal window.

Remember that these numbers correspond to logical core threads, not physical cores. So a quad core with hyperthreading (2 logical cores per physical CPU) will give you a total of 8 logical cores that can thread. If you want to know more about hyperthreading and logical cores vs. physical cores, consider a minor in Computer Science or Computer Engineering.

Step 6: Final check before docking

Before you run a docking job, it's always nice to double-check and make sure everything is in the right place and formatted correctly. Remember, if even a single line in a single file is off, the program will fail. You should have a folder that looks like this:

PC Users:



Mac Users:

Name
1stp_DOCK.pdbqt
1stp.pdb
conf.txt
▶ Ligands
screen_Vina_Mac_v4.1.sh
▶ Vina
Vina_files_Mac.tar.gz

Also confirm the following:

- You have a properly formatted **conf.txt** (all the values added in)
- You have the **ligand_biotin.pdbqt** file in the **Ligands** folder
 - You must start the name of all input files with the word "ligand." In other words, do not call your input file just biotin.pdbqt. The program specifically looks for input files that start with the word "ligand."
- You're not currently streaming video or downloading a big update file.
 - This can draw a lot of RAM depending on your system, and can cause problems during the docking calculations.
 - You can prevent this by disconnecting from the internet if you want. Vina does not need any internet connection to work.
 - If you have a nice machine with a lot of extra RAM, then this probably isn't necessary. The older the computer and the more limited your hardware, the more you need to be careful. Note that a GPU is not used by Vina, so it won't help the process (sorry gamers, but your fancy setup won't help you much here).

Step 7: Running the Docking Job

If everything looks good to go, we're ready to run out docking job at last! All we need to do is to execute the screening script (the file that ends in .sh) by line command.

PC USERS ONLY:

- Navigate to the **DockingJob** folder in Cygwin using the "cd" command.
 - It's always a good idea to use the "ls" command to make sure you can see all of your files to confirm you are in the right directory. It should look something like this:

```
Dr. Eagon@Megaflop /cygdrive/c/Users/Dr. Eagon/Desktop/DockingJob
$ ls
1stp.pdb 1stp_DOCK.pdbqt conf.txt Ligands screen_Vina_Win_v4.1.sh Vina Vina_files_Windows.tar.gz
```

- We see the contents of **DockingJob** listed, so we know we're in the right directory!
- Execute the screening script with the following command:
 - `$./screen_Vina_Win_v4.1.sh`
 - There will be a short pause, but then you should be able to see the output of the program as it runs
 - Depending on the type of computer you have, this will take 1-2 minutes on average
- When you're done, you'll see something like this:

```
Dr. Eagon@Megaflop /cygdrive/c/Users/Dr. Eagon/Desktop/DockingJob
$ ./screen_Vina_Win_v4.1.sh
Processing ligand ligand_biotin.pdbqt
#####
# 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. #
#####

Reading input ... done.
Setting up the scoring function ... done.
Analyzing the binding site ... done.
Using random seed: 762013696
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 -7.4 0.000 0.000
2 -6.6 1.176 1.358
3 -6.2 1.796 2.178
4 -5.9 1.871 2.117
5 -5.8 2.579 6.393
6 -5.8 4.229 6.758
7 -5.8 3.846 7.093
8 -5.7 4.851 7.786
9 -5.5 2.861 3.965
Writing output ... done.
```

- The program is done once a new command line prompt appears

MAC USERS ONLY:

- Navigate to the **DockingJob** folder in Cygwin using the “cd” command.
 - It’s always a good idea to use the “ls” command to make sure you can see all of your files to confirm you are in the right directory. It should look something like this:

```
CSM-CHECKOUT-05:DockingJob csmuser$ ls
1stp.pdb          Vina          screen_Vina_Mac_v4.1.sh
1stp.Dock.pdbqt   Vina_files_Mac.tar.gz
Ligands           conf.txt
CSM-CHECKOUT-05:DockingJob csmuser$
```

- We see the contents of **DockingJob** listed, so we know we’re in the right directory!
- Execute the screening script with the following command:
 - `$. /screen_Vina_Mac_v4.1.sh`
 - There will be a short pause, but then you should be able to see the output of the program as it runs
 - Depending on the type of computer you have, this will take 1-2 minutes on average
- When you’re done, you’ll see something like this:

```
CSM-CHECKOUT-05:DockingJob csmuser$ ./screen_Vina_Mac_v4.1.sh
Processing ligand ligand_biotin.pdbqt
#####
# 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.   #
#####

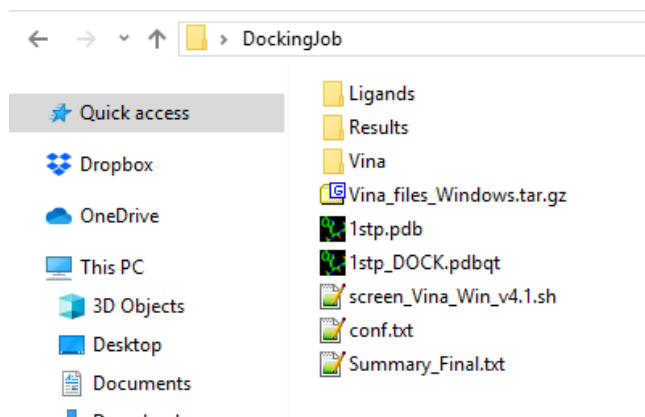
Reading input ... done.
Setting up the scoring function ... done.
Analyzing the binding site ... done.
Using random seed: 871988762
Performing search ...
0% 10 20 30 40 50 60 70 80 90 100%
|---|---|---|---|---|---|---|---|---|---|
*****
done.
Refining results ... done.

mode |  affinity | dist from best mode
    | (kcal/mol) | rmsd l.b. | rmsd u.b.
-----+-----+-----+-----
1      -7.5      0.000      0.000
2      -6.2      2.361      2.751
3      -6.0      1.519      1.753
4      -6.0      5.048      7.645
5      -5.8      3.936      7.152
6      -5.8      2.816      3.595
7      -5.8      1.903      2.208
8      -5.8      4.227      6.765
9      -5.6      4.325      6.381
Writing output ... done.
```

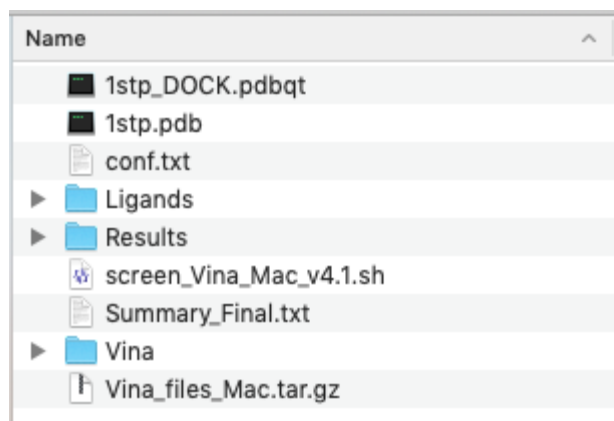
- The program is done once a new command line prompt appears.

- You'll note that 2 new items have appeared in your **DockingJob** folder:
 - A **Results** folder that contains your docked molecule
 - A **Summary_Final.txt** file that lists the calculated energy for your docked molecule

PC Users:



Mac Users:



- If you open the **Summary_Final.txt** file, you should see an output that looks like this:

```

1  ligand_biotin.pdbqt.txt  -7.4  ligand_biotin.pdbqt.txt  biotin
2

```

- Your file may look a little different, depending on the text file viewer you are using. The file should have all of the same entries (file name, energy value, file name, name)
- Note that the energy value here (-7.4) is the calculated binding energy in kcal/mol for biotin. The value is negative because it is favorable.
 - Your exact value may be slightly different, but it should be somewhere around the range of -7.0 to -7.5 kcal/mol

Troubleshooting: If your **Summary_Final.txt** file doesn't look like this, then something went wrong. Delete the **Results** folder and the **Summary_Final.txt** file, then look for a mistake and try running the script again. The most common errors are because:

- The **conf.txt** file wasn't formatted properly (some entry is misspelled, like the receptor line)
- The **Ligands** folder wasn't spelled correctly
- The biotin ligand wasn't named **ligand_biotin.pdbqt**
- The biotin ligand wasn't in the **Ligands** folder
- You get a "Permission denied" warning at the command line, or it says it can't find a file. You can fix this by typing the following from the command line in your **DockingJob** directory:
 - \$ `chmod -R 775 ./`
 - After you enter this command, run the screening script again
- A "parse" error in a line means something is misspelled. Take a look at the line it indicates.

Be sure to always delete the **Results** folder and the **Summary_Final.txt** file before you run the script again, otherwise the results will start to compile and create new errors.

Step 8: Processing your docked ligand into a PDB file

To properly examine our output ligand in ChimeraX, we need to convert the output of the Vina program into a PDB file that we can see.

- Open your **Results** folder
- You will see several files, but we're looking for one called **ligand_biotin_OUTPUT.pdbqt**
 - This file contains the results of docking the biotin molecule
 - The docking script will always add the word OUTPUT to the processed files to help tell the difference between this file and the original input file (**ligand_biotin.pdbqt**)
- Open the **ligand_biotin_OUTPUT.pdbqt** file with a text editor
 - You should notice that this file contains ~10 different models of biotin, which each molecule the result of a different simulation run
 - The number of attempts is set by the "num_modes" setting in the **conf.txt** file
 - The results are always organized as #1 = best
- We want to take the best result (model #1 from the **ligand_biotin_OUTPUT.pdbqt** file) and turn it into a PDB file. We are going to use the same online server we used in Step 3 to do this.
- Copy everything from REMARK with the name of the ligand to the TORSDOF line:

```
MODEL 1
REMARK VINA RESULT:      -7.4      0.000      0.000
REMARK Name = biotin
REMARK                   x      y      z      vdW Elec      q      Type
REMARK
ROOT
ATOM      1  C  LIG      1      10.810      1.227     -10.431      0.00      0.00      +0.069 C
ATOM      2  C  LIG      1      11.218     -0.154     -9.864      0.00      0.00      +0.108 C
ATOM      3  N  LIG      1      12.669     -0.330     -9.859      0.00      0.00     -0.294 N
ATOM      4  H  LIG      1      13.225     -0.322     -10.672      0.00      0.00      +0.151 HD
ATOM      5  C  LIG      1      13.163     -0.506     -8.613      0.00      0.00      +0.300 C
ATOM      6  O  LIG      1      14.382     -0.663     -8.386      0.00      0.00     -0.253 OA
ATOM      7  N  LIG      1      12.147     -0.481     -7.743      0.00      0.00     -0.294 N
ATOM      8  H  LIG      1      12.259     -0.598     -6.772      0.00      0.00      +0.151 HD
ATOM      9  C  LIG      1      10.853     -0.262     -8.388      0.00      0.00      +0.105 C
ATOM     10  C  LIG      1      10.196      1.046     -7.956      0.00      0.00      +0.093 C
ATOM     11  S  LIG      1      10.957      2.356     -8.986      0.00      0.00     -0.154 S
ENDROOT
BRANCH   1  12
ATOM     12  C  LIG      1      11.539      1.498     -11.684      0.00      0.00      +0.016 C
BRANCH  12  13
ATOM     13  C  LIG      1      11.664      2.888     -12.305      0.00      0.00      +0.001 C
BRANCH  13  14
ATOM     14  C  LIG      1      10.398      3.633     -12.433      0.00      0.00      +0.006 C
BRANCH  14  15
ATOM     15  C  LIG      1      10.376      4.607     -13.663      0.00      0.00      +0.052 C
BRANCH  15  16
ATOM     16  C  LIG      1      9.095      4.501     -14.412      0.00      0.00      +0.042 C
ATOM     17  O  LIG      1      8.914      3.325     -14.982      0.00      0.00     -0.550 OA
ATOM     18  O  LIG      1      8.077      5.169     -14.181      0.00      0.00     -0.550 OA
ENDBRANCH 15  16
ENDBRANCH 14  15
ENDBRANCH 13  14
ENDBRANCH 12  13
ENDBRANCH 1  12
TORSDOF 5
ENDMDL
MODEL 2
REMARK VINA RESULT:      -6.6      1.176      1.358
REMARK Name = biotin
REMARK                   x      y      z      vdW Elec      q      Type
REMARK
ROOT
```

REMARK with
the name line

Copy everything you
see highlighted

TORSDOF
line

<https://www.cheminfo.org/Chemistry/Cheminformatics/FormatConverter/index.html>

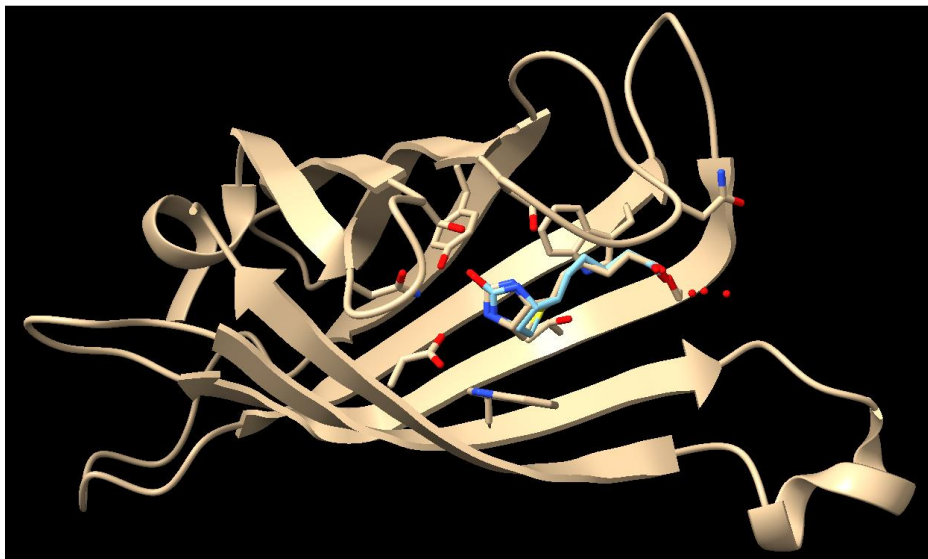
- In the “OPTIONS” pane, select the following:
 - “Input format” should be “pdbqt -- AutoDock PDBQT format”
 - “Output format” should be “pdb -- Protein Data Bank format”
 - “Generate coordinates” should be set to “None”
 - “Add / Delete hydrogens” should be set to “Delete”
 - “pH to add hydrogens” should be blank (empty)
- Hit the “Convert” button to get your new PDB file in the “Output” pane.

OUTPUT											
1	REMARK	Name = biotin									
2	REMARK					x	y	z	vdW	Elec	q
3	REMARK										
4	COMPND	biotin									
5	AUTHOR	GENERATED BY OPEN BABEL 2.3.2									
6	ATOM	1	C	LIG	1	10.810	1.227	-10.431	1.00	0.00	C
7	ATOM	2	C	LIG	1	11.218	-0.154	-9.864	1.00	0.00	C
8	ATOM	3	N	LIG	1	12.669	-0.330	-9.859	1.00	0.00	N
9	ATOM	4	C	LIG	1	13.163	-0.506	-8.613	1.00	0.00	C
10	ATOM	5	O	LIG	1	14.382	-0.663	-8.386	1.00	0.00	O
11	ATOM	6	N	LIG	1	12.147	-0.481	-7.743	1.00	0.00	N
12	ATOM	7	C	LIG	1	10.853	-0.262	-8.388	1.00	0.00	C
13	ATOM	8	C	LIG	1	10.196	1.046	-7.956	1.00	0.00	C
14	ATOM	9	S	LIG	1	10.957	2.356	-8.986	1.00	0.00	S
15	ATOM	10	C	LIG	1	11.539	1.498	-11.684	1.00	0.00	C
16	ATOM	11	C	LIG	1	11.664	2.888	-12.305	1.00	0.00	C
17	ATOM	12	C	LIG	1	10.398	3.633	-12.433	1.00	0.00	C
18	ATOM	13	C	LIG	1	10.376	4.607	-13.663	1.00	0.00	C
19	ATOM	14	C	LIG	1	9.095	4.501	-14.412	1.00	0.00	C
20	ATOM	15	O	LIG	1	8.914	3.325	-14.982	1.00	0.00	O
21	ATOM	16	O	LIG	1	8.077	5.169	-14.181	1.00	0.00	O
22	CONECT	1	10	2	9						
23	CONECT	2	1	3	7						
24	CONECT	3	2	4							
25	CONECT	4	3	5	6						
26	CONECT	5	4								
27	CONECT	6	4	7							
28	CONECT	7	2	8	6						
29	CONECT	8	9	7							
30	CONECT	9	1	8							
31	CONECT	10	11	1							
32	<										>

- Back in your **DockingJob** folder, create a text file called **biotin_BEST.txt**
 - Mac users--be sure to use BBedit when you make this text file
- Paste the contents of the “Output” pane into the **biotin_BEST.txt** file.
- Change the file extension of the **biotin_BEST.txt** file to **biotin_BEST.pdb**
 - You can also do this by renaming the file using the Save As command

Step 9: Visualizing your docked ligand and calculating RMSD

- Open ChimeraX
- Using the “Open” command under the “Home” tab, open the file **1stp.pdb**
- Use the “Open” command to also open your **biotin_BEST.pdb**
 - Ignore any warnings that ChimeraX gives about bad lines
- You should see something like this:



- Your docked biotin is the light blue colored chain. We can visually see that the docked molecule is in the same pocket and more or less lines up with original biotin ligand.
 - Your exact orientation may be slightly different than what you see, but the biotin molecules should “stack” pretty well.
- To confirm our accuracy, we are going to calculate the RMSD (Root Mean Square Deviation) with ChimeraX. This gives us a measure of the variance (how closely we matched the known answer)
- Calculate the RMSD between the two biotin molecules with the following command:
 - `$ align :BTN toAtoms #2 move nothing`

```
align :BTN toAtoms #2 move nothing  
RMSD between 16 atom pairs is 2.878 angstroms
```

- The align command only works if you have the exact same set of atoms
 - The “toAtoms” means to use the different atom types as a comparison. In other words, N to N, C to C, etc.
 - The “move nothing” tells ChimeraX not to physically move the molecules.
 - Docked ligands can usually be selected using the # tag, like the “#2” above. You can see these tags if you hover your mouse over the docked ligand.
- As far as RMSD values go, < 1.5 is great, 1.5-3.0 is good, 3.0-5.0 is okay, and > 5.0 is bad.
 - Our number (2.878) is good, and indicates success! Your number may be a bit different, but should be less than 5.
- **Save this final ChimeraX session file!** This is everything we have worked for!

Extra stuff for the curious

If you want to learn more about the Vina program, you can find the publication describing it here:

<https://doi.org/10.1002/jcc.21334>

Vina was funded through the NIH by taxpayers, so that is why it's freely available.

Frequently asked questions

1) Are there other docking programs out there? Is Vina special?

There are more than a dozen programs out there that do the same thing. A few are free and open source, while the rest are commercial. Vina is like the Mazda of docking programs. It's been around a long time and it's not pretty, but it works well and it's easy to use (works on Linux, PC and Mac). I have picked Vina because it's free and is one of the few that runs on both PC and Mac.

2) Can I play around with other docking programs to see how they work?

Not easily, no. Most are only available for Linux systems, and only a few for PC. Mac versions are rare. Other free, somewhat modern programs out there include Smina (a fork of Vina) and idock.

3) Where did that docking script come from? What are those other weird files I see in the Results folder, like the ZINC_ID.txt file?

I wrote the docking script. Normally, Vina only runs one molecule at a time. However, my program sets it up so that it will run as many ligands as you put in the Ligands folder. Vina also doesn't summarize all results. The other files you see are part of the processing script that allows for the generation of the Summary_Final.txt file. You're welcome.

4) What does the calculated energy value mean?

Energy values calculated by these programs mean very little from a thermodynamic point of view. Remember, you're holding the protein rigid (not letting it move), placing it in empty space, then trying to force a molecule into a pocket. This is not representative at all of an actual protein in a cell, which is a moving system surrounded by water molecules. The energy values do tell you the relative predicted ranking of one molecule versus another. So, the bigger the binding energy, the more likely it is predicted by the program to bind.

5) What is a "good" energy value to have?

Remembering that these numbers are predictions, you can use the following rules of thumb:

- -11 kcal/mol or better = great!
- -9 to -11 kcal/mole = good
- -7 to -9 kcal/mole = okay
- -5 to -7 kcal/mole = bruh...
- -5 kcal/mole or lower = call the ambulance, because this thing has no pulse

6) Can I play around with the num_modes and exhaustiveness settings?

Yes, you can. Keep in mind that the default values for these parameters are:

num_modes = 9

exhaustiveness = 8

You can change these settings by just adding in the value you want. For example, let's say you want 15 poses and you want Vina to work 10x the default setting computationally. Then you just add those values to your **conf.txt** file. Using the conf.txt file we used for this tutorial as an example, your new **conf.txt** file would look like this:

```
receptor = 1stp_DOCK.pdbqt

center_x = 11.09
center_y = 1.74
center_z = -10.63

size_x = 20
size_y = 20
size_z = 20

cpu = 2
num_modes = 15
exhaustiveness = 80
```

Be careful when changing the “exhaustiveness” value, as this isn’t just a linear multiplication. In other words, doubling this value doesn’t mean that Vina will take double the time—it could be much more! The actual time depends on your available RAM and your cpu and num_modes settings. If this number is higher than your system resources, it can crash your computer. I wouldn’t recommend going above 80 with this number. That, and there’s a paper out there showing that it isn’t worth it with Vina to go higher than 10x the default anyways. (<https://doi.org/10.1016/j.jmgm.2020.107532>)

Keep in mind that the “num_modes” is a maximum setting, and Vina may not always output that many solutions. Depending on the exhaustiveness setting, Vina has a threshold from which it will stop trying to generate new poses. In fact, some simulations may only output a single conformation, because any change whatsoever results in something with a significantly lower binding energy. For instance, Vina will give up on anything that outputs a low docking energy.

Also note that by changing these numbers, you’re not necessarily more likely to get the “correct” answer here. Beyond a certain point, adding more conformations just ends up forcing Vina to find more “bad” conformations, and higher “exhaustiveness” just causes Vina to spend a lot of time making tiny tweaks to small groups to find just the “right” orientation (should the methyl be twisted 12.43° or 12.42° relative to the plane of the ring?...still insufficient data....I’m sorry, I can’t do that Dave.)

Remember: Any computer simulation will be limited by our understanding of these systems, and the “shortcuts” that we take to calculate an output value in a reasonable amount of time.