**Score-based Pharmacophore Modeling Guide**

**2/22/2022**

**Author: Gregory L. Szwabowski**

**Contact: gszwbwsk@memphis.edu**

The following guide outlines the steps required to generate score-based pharmacophore models in any protein structure. Files necessary for this tutorial are located in a .zip file available at:

<https://github.com/gszwabowski/guides/blob/master/score-based_ph4_tutorial_files.zip>

Prerequisites

* .zip file with MOE scripts, python code
* Input structure (experimentally determined structure or homology model)
* MOE installation
* Python installation with the following non-standard modules installed (I suggest installing Anaconda (<https://www.anaconda.com/products/individual>) since it comes with these libraries):
  + *sklearn 0.24.2* (use pip list from the command line or conda list from the Anaconda prompt to check your sklearn version)
    - If your *sklearn* version is above 0.24.2, use the following commands in your command prompt/anaconda prompt to downgrade:
      * Command prompt (non-Anaconda)
        + pip uninstall scikit-learn
        + pip install scikit-learn==0.24.2
      * Anaconda prompt
        + conda install -c conda-forge scikit-learn=0.22.2
  + *pickle*
  + *numpy*
  + *pandas*

|  |  |  |
| --- | --- | --- |
| **Step** | **Description** | **Figure** |
| 1 | First, open MOE and set your current working directory to the folder containing the structure you wish to generate pharmacophore models for. Your input structure can be of any filetype and can be an experimentally resolved structure or homology model of any protein. If it opens in MOE, it can be used as an input for pharmacophore model generation. Make sure that your input structure:   * Consists of only a single chain * Is prepared using the QuickPrep function on the right hand side of the MOE window at default settings. * Is free of ligands, ions, or solvent residues |  |
| 2 | Click on *Compute* 🡪 *Site Finder…* and then click *Apply* in the Site Finder window to find a probable binding site within your target structure. By default, the top ranked binding site is selected in the Site Finder window. If the top ranked binding site looks out of place, you may need to edit your input structure or select a different input structure. Once the top-ranked binding site is adequate, ensure that the *Select Contact Atoms* option is checked. During the steps that follow, ensure that you do not accidentally deselect these atoms. |  |
| 3 | Click on *Compute* 🡪 *Fragments* 🡪 *MultiFragment Search…* to open the panel that is used to perform multiple copy simultaneous searches (MCSS) in MOE. Here, you will be prompted to select which fragments you wish to use to perform MCSS. In the lower half of the panel, ensure that all fragments using the following steps:   * Clicking the first entry (1,2-dimethylpyrrolidine in MOE 2019.01), highlighting it blue. * Scroll down to the last entry. * Shift+click on the last entry (water in MOE 2019.01). All entries should now be colored blue.   Click on *Next >>*. |  |
| 4 | If you are using an experimentally determined structure, click *Next >>* once more to get to the “Start Simulation” page of the MCSS panel. If you are using a homology model, ensure that the value for the “Belly Distance” parameter is set to 10. This will allow for the flexibility of residue side chains within 10 Å of fragments being placed, which aims to offset discrepancies in homology model quality. Click on *Next* >>*.* |  |
| 5 | In the “Start Simulation” page of the panel, the only thing that needs to be adjusted is the “Prefix” parameter, which determines what your MCSS output filenames will be prefixed by. I typically change the name from “mfss” to the name of the target I am working with. Click *Start* to start MCSS. Depending on whether you’re using an experimentally determined structure or a homology model, MCSS will take anywhere from around an hour to a day or so. |  |
| 6 | At this point, you are now ready to generate subset databases from the MCSS output database that are used to generate additional pharmacophore models. First, open the *mfss\_subset.svl* script (located in the “scripts” folder) using the *Edit…* button and then save and load the script by clicking *SVL 🡪 Save and Load*. Next, move the files suffixed \_*output.mdb* and \_*minrec.moe* to a folder titled “moe”. After, create 4 additional empty subdirectories in the directory where your “moe” folder is located, titled as follows:   * “ef” * “gh” * “rec\_ef * “rec\_gh”   With the *mfss\_subset.svl* script loaded and the 5 subdirectories (“moe”, “ef”, “gh”, “rec\_ef”, “rec\_gh”) created, navigate to the first subset folder (“ef”) in the MOE file browser and set it as your CWD. You will now use the mfss\_subset function to generate a database containing a subset of the original set of 39 fragments placed with MCSS. Since your CWD is set to the “ef” directory, you will be creating a subset of the original MCSS output database that only contains fragments in the EF fragment subset. Run the command  mfss\_subset [mfss\_output, receptor, frag\_db, prefix]  to generate the subset database in the “ef” folder, where   * mfss\_output: original MCSS output database located in the “moe” folder (suffixed *output.mdb*) * receptor: minimized receptor contained in the “moe” folder (suffixed *minrec.moe*) * frag\_db: database in the “fragment\_subsets” folder containing a reference fragment set * prefix: desired name prefix for subset files   Example: With the filenames that have been shown in screenshots thus far, the command to generate the MCSS subset in the “ef” folder would be:  mfss\_subset ['../moe/gpr37\_hm10\_output.mdb', '../moe/gpr37\_hm10\_minrec.moe', '../../../../guides/score-based\_ph4\_tutorial\_files/fragment\_subsets/ef.mdb', 'GPR37\_ef']  Note that the frag\_db argument is set to reference the *ef.mdb* file containing the EF set fragments and the prefix argument is formatted as <target name>\_<fragment subset>. Next, navigate to the “gh” folder you created and set it as your CWD. Run the mfss\_subset command again, this time changing the 3rd and 4th arguments of your command. Using the files shown in screenshots as an example, the command would be:  mfss\_subset ['../moe/gpr37\_hm10\_output.mdb', '../moe/gpr37\_hm10\_minrec.moe', '../../../../guides/score-based\_ph4\_tutorial\_files/fragment\_subsets/gh.mdb', 'GPR37\_gh']  Continue this process in the “rec\_ef” and “rec\_gh” directories, making sure to change the 3rd and 4th arguments of your command to match the fragment subset being generated. Once complete, you should have 5 subdirectories with MCSS outputs and minimized receptor structures: 1 containing the original MCSS output in the “moe” folder and 4 containing subsets of the original output in the “ef”, “gh”, “rec\_ef”, and “rec\_gh” folders. | “moe” folder:    Directory structure for fragment set subdirectories:    Outputs for each fragment subset directory: |
| 7 | You can now perform score-based pharmacophore model generation. Prior to generating pharmacophore models, the *ph4\_edit\_2.svl* file needs to be moved to the “svl/run” folder located in your MOE installation directory. Copy this file and paste it into the “svl/run” folder located in your MOE installation directory. To start, set the directory containing the “moe”, “ef”, “gh”, “rec\_ef”, and “rec\_gh” subdirectories as your CWD. Next, navigate to the “scripts” folder and *Edit…* then *Save and Load* the *scorebased\_ph4gen.svl* script. Although this script can be used to generate pharmacophores in each fragment subset directory, it is much easier to sequentially generate pharmacophores in each subset directory. After saving and loading the pharmacophore model generation script, ensure that no MOE windows other than the main MOE window, SVL commands window, and Sequence Editor windows are open. Run the following command:  scorebased\_ph4gen [fragment\_sets]  where fragment\_sets is an integer 0 or 1 representing whether you have set up directories for each fragment set. In this case, the command to use is:  scorebased\_ph4gen [1]  Automated pharmacophore model generation will then commence. This process should take about an hour, make sure to leave MOE alone while the script is running. When it is done, a MOE window will pop up informing you so. When the script is complete, you should have 4 pharmacophore model files (each named after the 1 of the 4 scores used to sort the fragments in the MCSS output database) in each of the 5 subdirectories you created. | Moved *ph4\_edit\_2.svl* file:    Setting CWD:    Command to start pharmacophore model generation:    Pharmacophore files: |
| 8 | With 20 pharmacophore models generated, you can now begin extracting the attributes necessary for pharmacophore model classification (more information regarding the attributes calculated can be found in the header of the *scorebased\_datacollection.svl* script). To do this, save and load the *scorebased\_datacollection.svl* script in the “scripts” folder. Next, navigate to the “ef” subdirectory and set it as your CWD. Run the following command:  scorebased\_datacollection [rec\_name, receptor]  where:   * rec\_name is the name of your target, entered as a token (e.g. ‘GPR37’) * receptor is the file containing the structure used during pharmacophore model generation   While this command is being run, the Site Finder window will automatically open and close itself to find the geometric centroid of the top-ranked binding site in your receptor. When this script is done running, a database titled *ph4\_data.mdb* will be present in the directory you’re working in. This database will contain 4 entries, with each entry describing various attributes of each of the 4 pharmacophore models present in the “ef” subdirectory. In the output database, the “score\_type” field describes the fragment-receptor interaction score used to sort the fragments in the MCSS output database prior to pharmacophore feature annotation. In addition, this field also describes the filename of each pharmacophore model.  Note: The “match\_features” and “Hits” columns will be left blank when the script is done running and will later be manually filled. | Example command:    Output database: |
| 9 | Set your CWD to the next fragment set subdirectory (“gh”) and run the scorebased\_datacollection command to extract attributes for each pharmacophore model generated with the GH fragment set. Continue this process in each subsequent subdirectory (“moe”, “rec\_ef”, “rec\_gh”). Next, your goal is to import the data contained in each fragment set’s *ph4\_data.mdb* file into a single database. Set your MOE CWD to the directory containing the 5 fragment set subdirectories and click *New* 🡪 *Database…* Name the new database *ph4\_data\_all.mdb* and make sure to delete the ‘mol’ string from the Fields portion of the File Prompt. Click *OK* to create an empty database.  In the newly created database, click *File* 🡪 *Import…*.In the Database Import menu, click on the + icon located on the right-hand side of the menu. In the File Prompt, double-click into each fragment subset folder and add *ph4\_data.mdb* to the list of files to be imported into *ph4\_data\_all.mdb*. Once each fragment set’s *ph4\_data.mdb* file has been added to the list, click *OK* in the File Prompt. Back in the Database Import menu, click *OK* to import the files you’ve selected into *ph4\_data\_all.mdb*. Once data for all 20 entries are imported into *ph4\_data\_all.mdb*, click the blank box above the first entry number to select all cells in the database. Next, click *Edit* 🡪 *Copy* and paste this data into a blank Excel spreadsheet. | *ph4\_data\_all.mdb* creation:    Database Import menu:    File Prompt:    Final file list prior to import containing each fragment set’s *ph4\_data.mdb* file:    Selecting all entries: |
| 10 | Your next objective is to use your generated pharmacophore models to search our internal test database containing conformations of 569 ligands with varying activities for 30 class A GPCR. This database is titled *pbd\_conf10\_updated.mdb* and is included in this tutorial’s .zip file. To accomplish this, first save and load the *feature\_search\_dir\_7feats.svl* file located in the “scripts” folder. Just like the previous step, this step involves setting your CWD to 1 of 5 fragment set subdirectories and running a command in each fragment set subdirectory. In each fragment set subdirectory, run the command  feature\_search\_dir\_7feats [compound\_db, mseq\_field]  where   * compound\_db is the location of the internal test database relative to your CWD * mseq\_field is the field containing the molecule sequence number for each molecule in compound\_db   For example, I would use the command feature\_search\_dir\_7feats [‘../pbd\_conf10\_updated\_uniq.mdb’, ‘mseq’]if the internal test database was located in the directory *above* the fragment set subdirectory I have my CWD set to. Running this command will use each pharmacophore model present in your CWD to search the internal test database at 3, 4, 5, 6, and 7 partial match features. The number of hits for each pharmacophore model when searching at each partial match feature number will be printed to the SVL commands window as well as saved to a file called *ph4\_searchlog.txt*. | Example database search command:    feature\_search\_dir\_7feats output: |
| 11 | While this script performs database searches at varying levels of specificity, we are interested in the number of hits obtained when attempting to match 5 or 6 of the 7 features present in each generated pharmacophore model (suffixed “\_5” and “\_6” in the script’s output, respectively). Go back to the Excel spreadsheet you pasted *ph4\_data\_all.mdb* into and save 2 versions of it as .csv files: one suffixed *\_5feats.csv* (for searches matching 5 of 7 pharmacophore features) and one suffixed *\_6feats.csv* (for searches matching 6 of 7 pharmacophore features). In each of these .csv files, you will need to manually fill in the “match\_features” and “Hits” columns present in each database. For the “match\_features” column, enter 5 for all entries in the *\_5feats.csv* file and enter 6 for all entries in the *\_6feats.csv* file. For the “Hits” column, you will have to manually enter the number of hits retrieved when using each pharmacophore model to search the internal test database at 5 or 6 features. This can be a bit tricky, as you must keep track of which fragment set, pharmacophore model, and partial match feature you are entering data for.  The following is an example case: running the feature\_search\_dir\_7feats command while your CWD is set to the “ef” fragment set subdirectory will result in a text output describing the hits obtained for the 4 pharmacophore models in the directory (*dE(class).ph4, dE.ph4, dU(class).ph4, dU.ph4*) when searching the internal test database at different partial match feature number values (suffixed “\_3” through “\_7” in the text output). For each entry with an “ef” value in the “subset” column of the *\_5feats.csv* spreadsheet (i.e. the pharmacophore models generated with the EF fragment subset), I would enter the results from the feature\_search\_dir\_7feats output that were suffixed with “\_5” while ensuring that I am matching the score type in the results to the score type in the spreadsheet. I would then repeat this process for the remaining fragment set subdirectories. | Example .csv files:    Data entry: |
| 12 | The data collected into your *\_5feats.csv* and *\_6feats.csv* files can now be fed into the machine learning model. In either your command prompt or Anaconda prompt, cd to the directory containing the *PH4\_classifier.py* script. At this point, you’ll be using the *PH4\_classifier.py* script to separate your data into 5 clusters and then classify data in the 1st cluster using logistic regression. Use the following command to use the script with your data:  python PH4\_classifier.py <.csv file>  where   * .csv file points to the location of your *\_5feats.csv* or *\_6feats.csv* file, relative to your current directory   This command needs to be run twice, once for each of your .csv files. Results will be printed to the command line as well as written to a file named *clusterI\_ph4\_preds.csv*. The *clusterI\_ph4\_preds.csv* file will be overwritten when the command is run a second time, so be sure to appropriately rename it prior to running the command again. To determine which of your pharmacophore files have been classified as quality with the *PH4\_classifier.py* script, take a look at the output that is printed after running the command. In the output in the first figure on the right as an example, 2 pharmacophore models are classified as quality:   1. the dE(class) pharmacophore model generated with the receptor EF fragment set (the *dE(class).ph4* file located in the “rec\_ef” subdirectory) 2. the dE(class) pharmacophore model generated with the receptor GH fragment set (the *dE(class).ph4* file located in the “rec\_gh” subdirectory)   The script may classify pharmacophore models with zero internal test database hits (the “hits\_actual” column in the output), which are likely to be of little use in external database searches. Therefore, I suggest the using only pharmacophore models that returned hits during internal test database searches (ex. pharmacophore models with index 0, 2, and 4 in the bottom figure to the right) prior to external database searches. | Example command and output:    Another example command and output: |
| 13 | Hopefully, you now have at least 1 pharmacophore model classified as quality with the *PH4\_classifier.py* script. When performing external database searches with your predicted quality pharmacophore models, ensure that these searches are performed with the correct number of partial match features. For example, a pharmacophore model classified as quality with your *\_5feats.csv* data should require prospective ligands to match 55 of the 7 features present in the pharmacophore model.  If none of your generated pharmacophore models are classified as quality with the *PH4\_classifier.py* script, I suggest generating pharmacophore models in another structure of your target (whether experimentally determined or predictive) and classifying them as a last resort. |  |