parKVFinder Guide

Welcome to the **parallel KVFinder (*parKVFinder*)** wiki, this page was built to help you get started with our cavity detection software.

Please read and cite the original paper <`paper title`> (<`doi`>).

---

parKVFinder Installation Guide

Follow these steps to install parKVFinder software on Linux or macOS:

1. Download parKVFinder source code.

Get the latest parKVFinder release from here.

On the terminal, go to the directory containing **parKVFinder.zip** file.

```
$ unzip parKVFinder.zip
$ cd parKVFinder
```

Or, clone the parKVFinder repository into a new directory using `git`.

```
$ git clone https://github.com/LBC-LNBio/parKVFinder.git
$ cd parKVFinder
```

2. Compile parKVFinder from source.

parKVFinder supports GCC (GNU Compiler Collection) versions later than 6. If necessary, refer to this guide for installing GCC versions later than 6 on Linux and macOS.

```
$ make
```

3. Create a symbolic link to run parKVFinder consistently.

```
$ make link
```

Note: `make link` create a soft symbolic link for parKVFinder in your /usr/local/bin directory, using the following command `sudo ln -s` `` `pwd` ``/parKVFinder /usr/local/bin/parKVFinder`.

4. Export *KVFinder_PATH* variable and add to *~/.bashrc* file or similar configuration file.

```
$ export KVFinder_PATH=`pwd`
$ echo "export KVFinder_PATH=`pwd`" >> ~/.bashrc
$ source ~/.bashrc
```

Note: Depending on the operating system, other configuration files are used instead of ~/.bashrc. For example, in macOS ~/.bash_profile are used.

GCC Installation Guide

The installation procedures for GCC (GNU Compiler Collection) are different for Linux and macOS.

First check the GCC version on your operating system.

```
$ gcc --version
```

parKVFinder supports GCC versions later than 6. Otherwise, you must install a newer GCC version.

Linux

To install the standard available version of GCC on your operating system, type:

```
$ sudo apt install gcc
```

Note: check the GCC version again to make sure you have installed a version later than 6.

If the default version GCC is not later than 6, you can install directly any later version by typing:

```
# GCC version 6
$ sudo apt install gcc-6
# GCC version 7
$ sudo apt install gcc-7
# GCC version 8
$ sudo apt install gcc-8
# GCC version 9
$ sudo apt install gcc-9
```

Then, there are two possible paths to prepare GCC to compile the parKVFinder source code.

- Create a symbolic link to your custom GCC

```
$ sudo ln -s /usr/bin/gcc-X /usr/local/bin/gcc
```

Note: **X** is the version of GCC you have installed.

- Replace `gcc` to `gcc-X` in **Makefile** within the **parKVFinder** directory

macOS

The macOS native C compiler is **Clang LLVM compiler**, which is not supported by parKVFinder.

First, you need to install:

- *Homebrew* package manager;
- Command Line Tools (CLT) for *Xcode* (`xcode-select --install`) or *Xcode*.

To install the standard available version of GCC on your *Homebrew* package manager, type:

```
$ brew install gcc
```

Note: check the GCC version, using `brew info gcc`, to make sure you have installed a version later than 6. Also, *Homebrew* installation will not override **Clang LLVM compiler** link.

If the default version GCC is not later than 6, you can install directly any version by typing:

```
# GCC version 6
$ brew install gcc@6
# GCC version 7
$ brew install gcc@7
# GCC version 8
$ brew install gcc@8
```

```
# GCC version 9
$ brew install gcc@9
```

Then, there are two possible paths to prepare GCC to compile the parKVFinder source code.

- Create a symbolic link to your custom GCC

```
$ sudo ln -s $(brew --prefix)/bin/gcc-X /usr/local/bin/gcc
```

Note: **X** is the version of GCC you have installed.

- Replace `gcc` to `gcc-X` in **Makefile** within the **parKVFinder** directory

```
$ sed -i -e 's/gcc/gcc-X/' Makefile
```

Note: **X** is the version of GCC you have installed.

PyMOL Plug-in Installation Guide

PyMOL is required if you wish to use parKVFinder with a graphical user interface (*parKVFinder PyMOL Tools*). If necessary, refer to this guide for installing PyMOL on Linux and macOS.

parKVFinder PyMOL Tools is available to use parKVFinder with PyMOL v1.8 and v2.

However, plug-in requires the installation of toml and future modules and native python do not have them installed. So you need to install them:

```
# PyMOL v1.8 (python 2)
$ pip install toml future
# PyMOL v2 (python 3)
$ pip3 install toml future
# In an conda environment via Anaconda Cloud
$ conda install -c conda-forge toml future
```

Note: pip or Anaconda package management system installation is required.

Then, to install the *parKVFinder PyMOL Tools* on PyMOL, follow these steps:

1. Open PyMOL.
2. Go to **Plugin** menu → **Plugin Manager**.
3. The **Plugin Manager** window will open, go to the **Install New Plugin** tab.
4. Under **Install from local file** Group, click **Choose file...** .
5. The **Install Plugin** window will open, go to the **tools** directory at **parKVFinder** directory and select **parKVFinder_PyMOL_tools.py**.
6. The **Select plugin directory** window will open, select **/home/<user>/.pymol/startup** and click **OK**.
7. The **Confirm** window will open, click **OK**.
8. The **Sucess** window will appear, confirming that the plug-in has been installed.
9. Restart PyMOL.
10. *parKVFinder PyMOL Tools* is ready to use.

PyMOL Installation

The installation procedures for PyMOL are different for Linux and macOS.

Linux

The easiest way is to use your distribution package manager.

```
$ sudo apt install pymol
```

Note: PyMOL version varies according to your distribution version.

Or, you can install PyMOL through Anaconda package management system.

PyMOL 1.8

An open source version of PyMOL 1.8 is available from Anaconda. However, a conda environment must have python 2 installed.

Follow these steps:

1. Create a new environment and activate it:

```
$ conda create --name pymol1.8 python=2.7
$ conda activate pymol1.8
```

2. Install PyMOL 1.8 via Anaconda Cloud:

```
$ conda install -c mw pymol
```

3. Try PyMOL 1.8:

```
$ pymol
```

PyMOL 2

A version of PyMOL 2 is available from Anaconda. However, a conda environment must have python 3 installed.

Follow these steps:

1. Create a new environment and activate it:

```
$ conda create --name pymol2 python=3.7
$ conda activate pymol2
```

2. Install PyMOL 2 via Anaconda Cloud:

```
$ conda install -c schrodinger pymol
```

3. Try PyMOL 2:

```
$ pymol
```

macOS

In macOS, PyMOL requires XQuartz installation.

Install Xquartz via *Homebrew*.

```
$ brew cask install xquartz
```

Or,

An XQuartz installer is provided here.

PyMOL 1.8

An open source version of PyMOL 1.8 is available from Anaconda. However, a conda environment must have python 2 installed.

Follow these steps:

1. Create a new environment and activate it:

```
$ conda create --name pymol1.8 python=2.7
$ conda activate pymol1.8
```

2. Install PyMOL 1.8 via Anaconda Cloud:

```
$ conda install -c mw pymol
```

3. Try PyMOL 1.8:

```
$ pymol
```

PyMOL 2

A version of PyMOL 2 is available from Anaconda. However, a conda environment must have python 3 installed.

Follow these steps:

1. Create a new environment and activate it:

```
$ conda create --name pymol2 python=3.7
$ conda activate pymol2
```

2. Install PyMOL 2 via Anaconda Cloud:

```
$ conda install -c schrodinger pymol
```

3. Try PyMOL 2:

```
$ pymol
```

If the options above does not work, you can try an installation via *Homebrew* package manager. A brief description of how to install open source PyMOL for macOS is provided here.

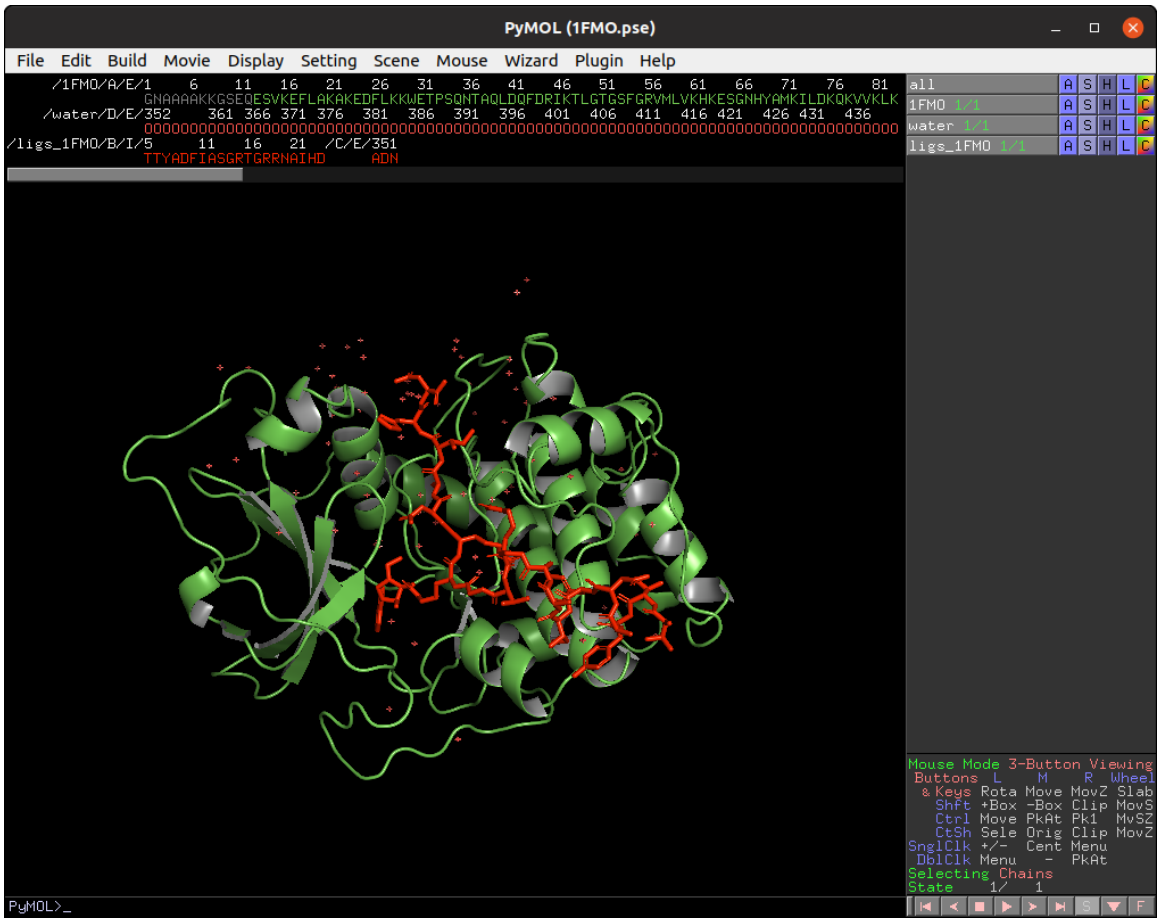
Tutorial

On this tutorial, we are going to demonstrate some features of the parKVFinder, including the graphical user interface (*parKVFinder PyMOL Tools*) and the command-line interface.

All files used on this tutorial can be found under **input** directory, on the **parKVFinder** directory.

parKVFinder PyMOL Tools

First, load **input/1FMO.pse** into PyMOL viewer, which loads two objects in your scene. The **1FMO** is a subunit of a protein kinase A and the **ligs_1FMO** is an adenosine (ADN) and a peptide kinase inhibitor (PKI).

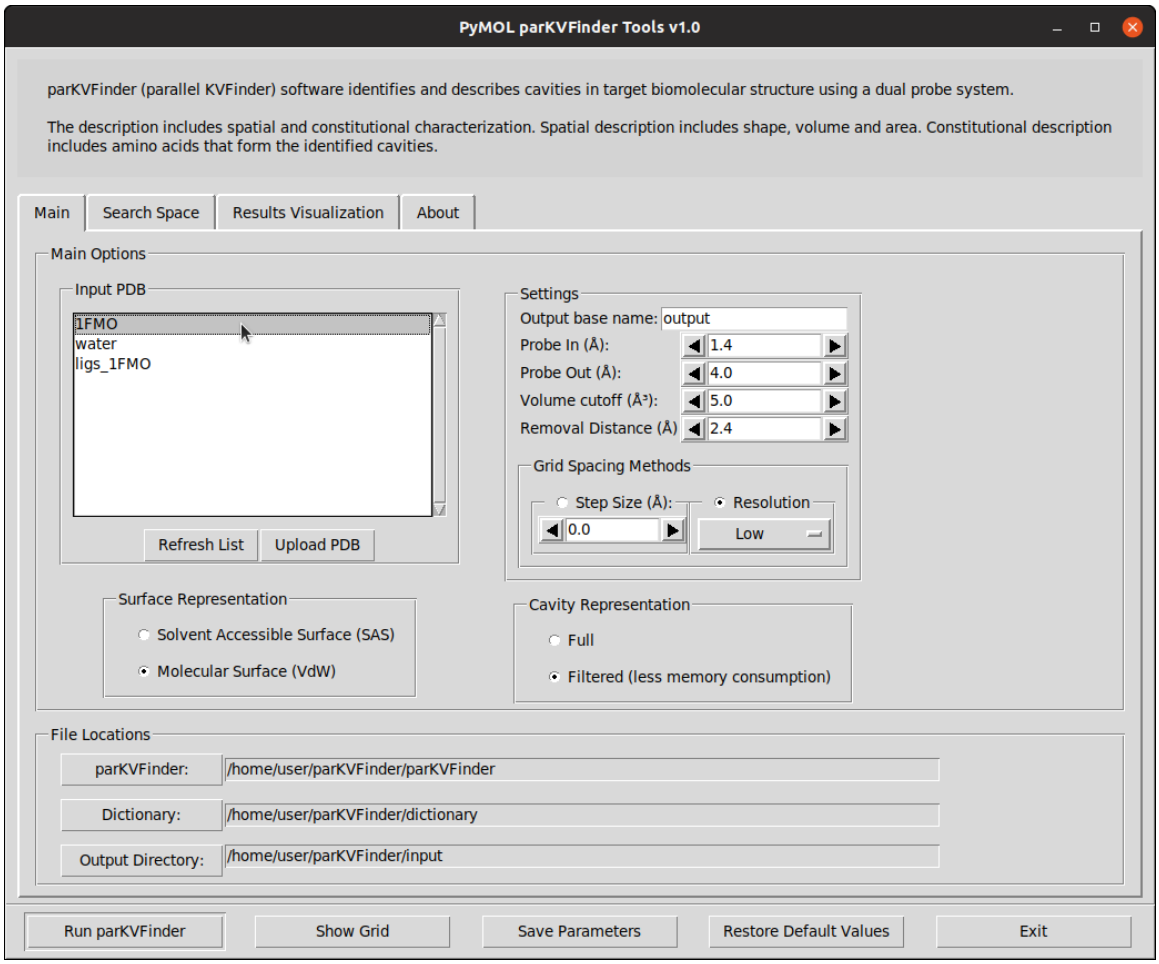


Whole protein detection

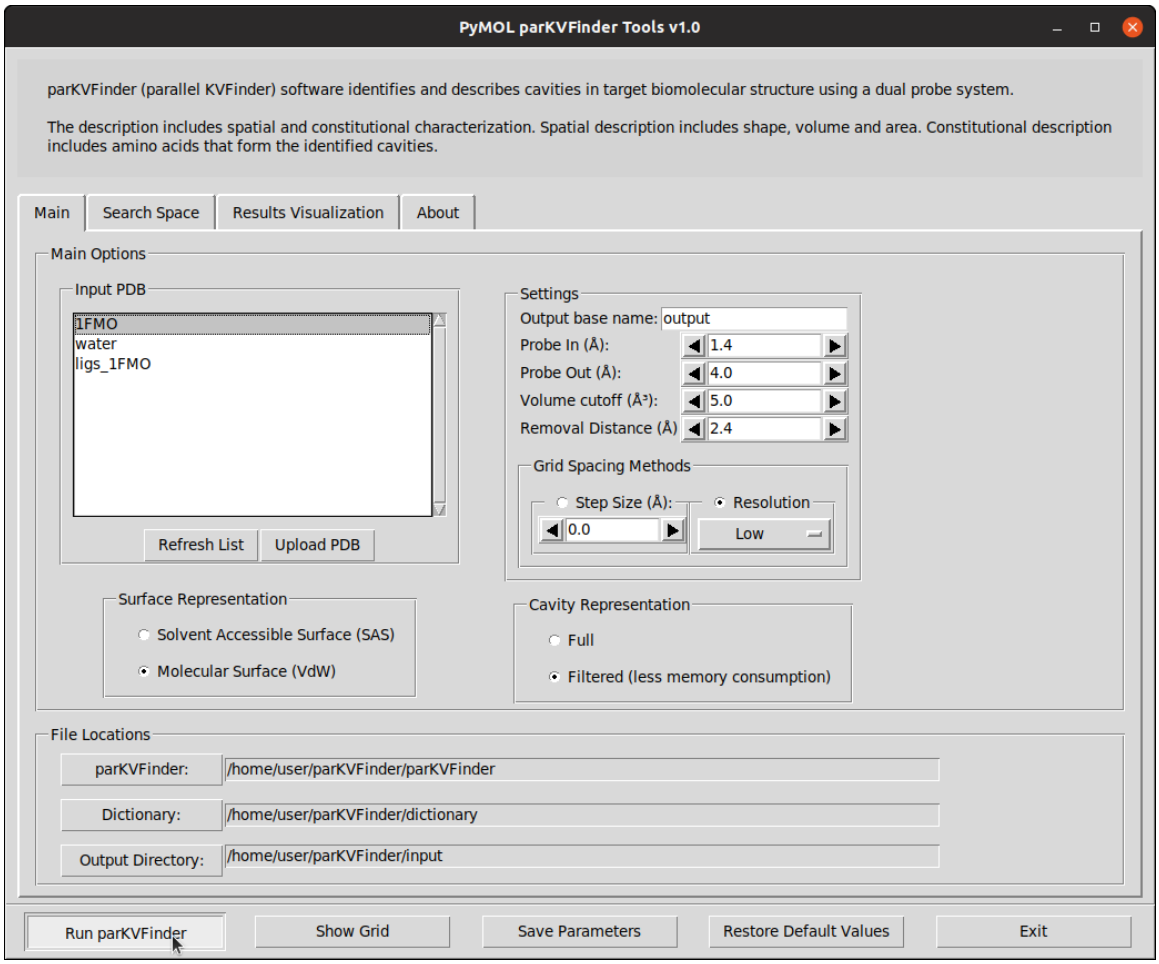
The default parameters are designed to make a simple and fast whole protein detection.

On PyMOL, open **parKVFinder PyMOL Tools** under **Plugin** tab. The objects on the scene will be listed on the **Input PDB** list box, on the **Main** tab. If not, press the **Refresh List**

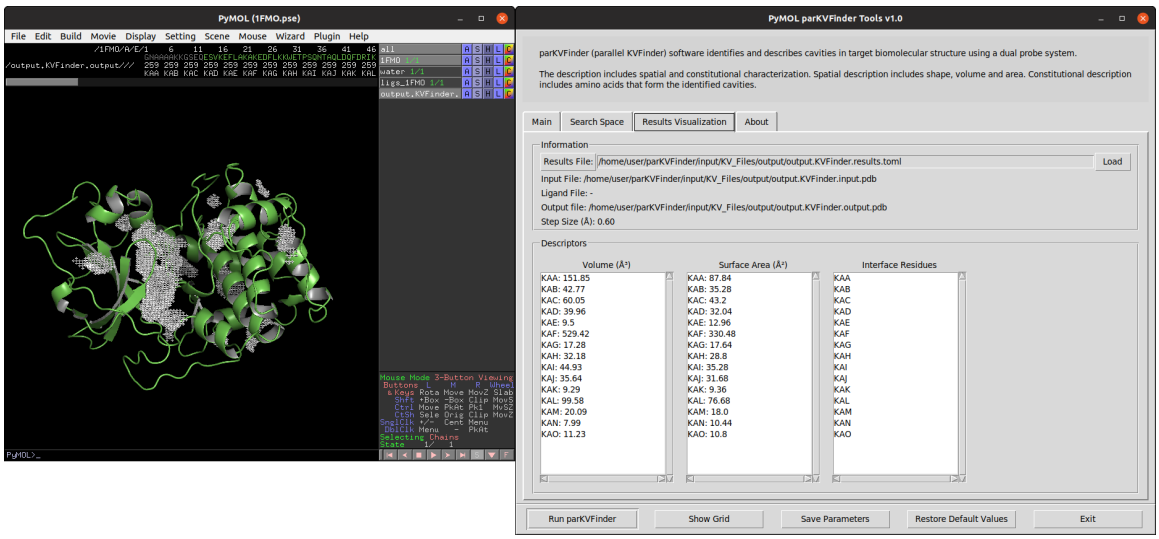
The **Input PDB** selection sets which object will be analyzed by parKVFinder. Select **1FMO** on the list box.



To run parKVFinder with the default parameters, just click **Run parKVFinder** button or press **Enter**.

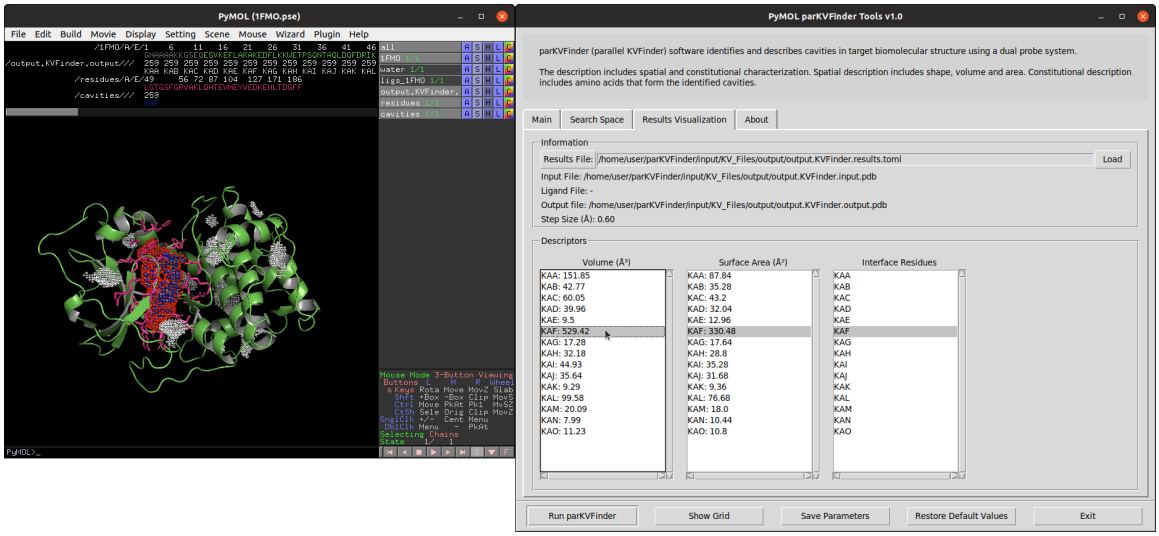


After execution is complete, cavities PDB is loaded into PyMOL viewer as <Output Base Name>.KVFinder.output object and the results file is loaded on the **Results Visualization** tab. In addition, the focus automatically shifts to **Results Visualization** tab.



We can select cavities in the **Volume** or **Surface Area** lists to highlight them on a new object called **cavities**, helping to identify each cavity. Also, we can select cavities in the **Interface Residues** list

to highlight the residues around the cavities on a new object named **residues**.



Changing cavity ceiling

parKVFinder is all about parameter customization. One of parKVFinder's most powerful assets is the ability to manually set the cavity ceiling. parKVFinder works with a double probe system. A smaller probe, called Probe In, and a bigger one, called Probe Out, that defines two molecular surfaces with different molecular accessibility. The space left between these surfaces is considered cavities.

Let's show the effect of varying **Probe Out** and **Removal Distance** on the cavity ceiling.

First, we should copy the adenosine to a new object using the following PyMOL commands:

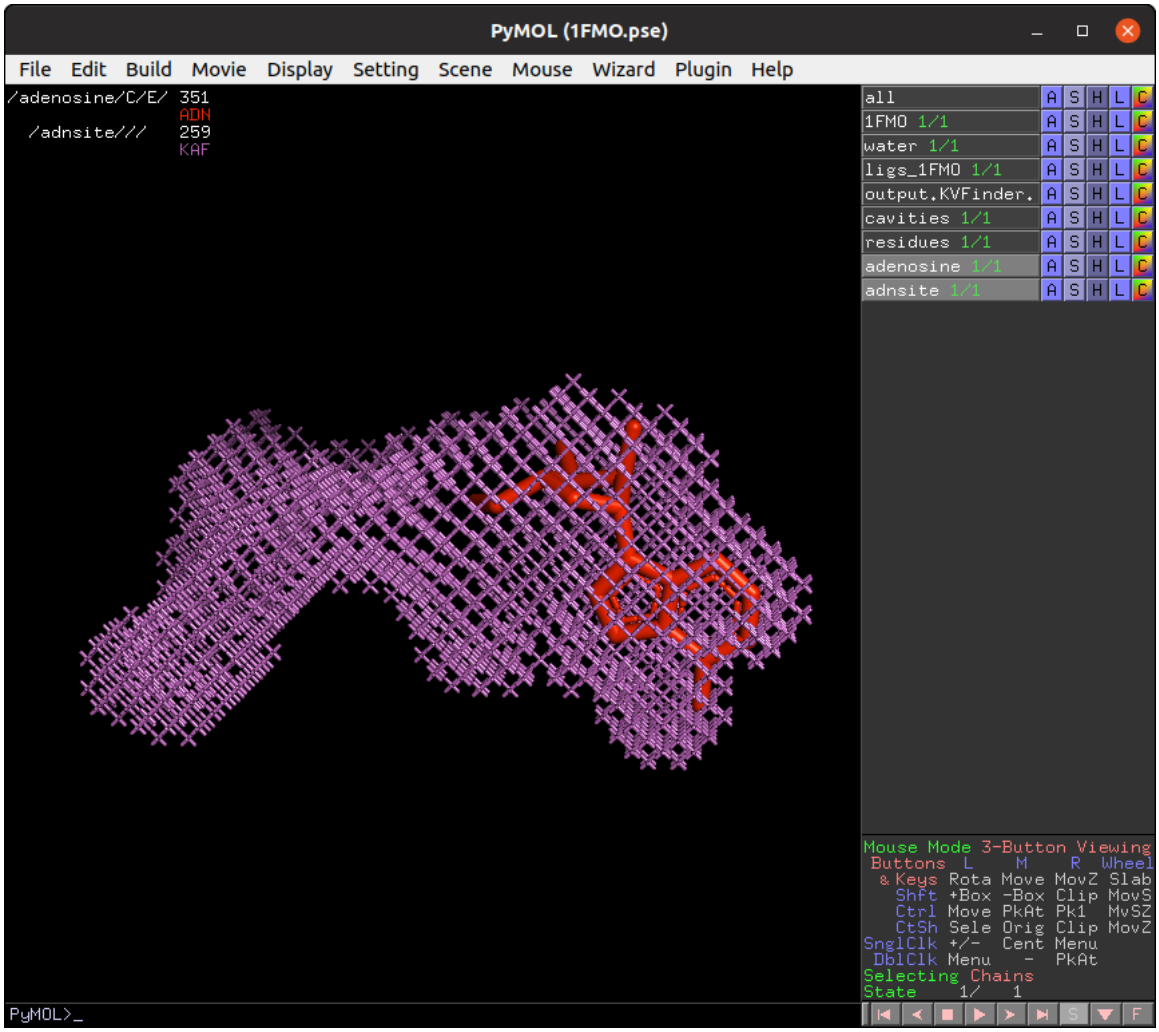
```
# Copy adenosine
select resn ADN
create adenosine, sele
delete sele
```

Also, copy the adenosine cavity (KAF) to a new object (adnsite) to compare the cavity ceiling from the previous execution.

```
# Copy adenosine cavity
select resn KAF and output.KVFinder.output
create adnsite, sele
delete sele
```

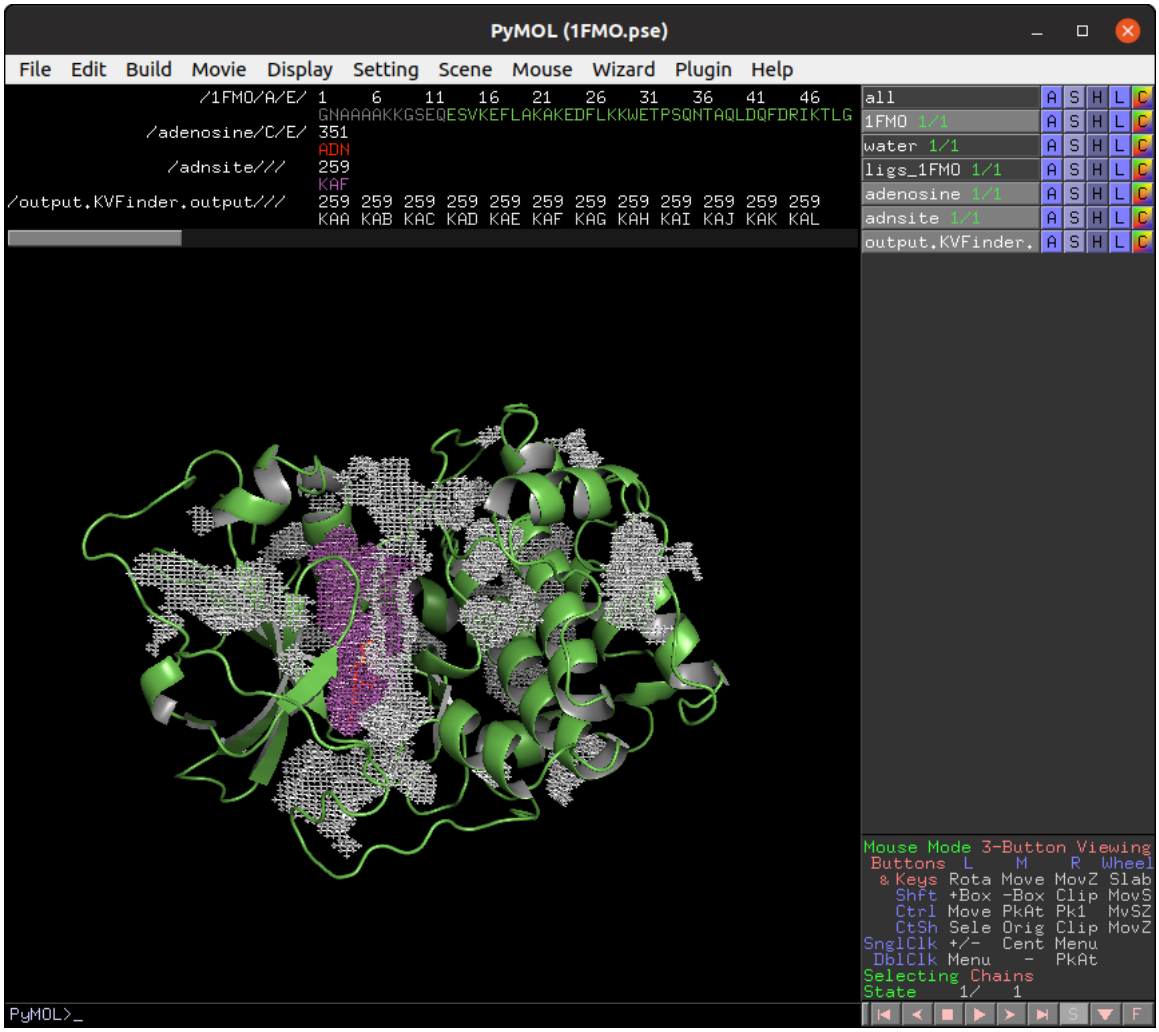
Finally, prepare the new PyMOL scene.

```
# Prepare PyMOL scene
color magenta, adnsite
disable
enable (adnsite, adenosine)
```



Adjusting Probe Out

As mentioned above, adjusting the Probe Out size changes the level of the cavity ceiling. So let's go back on the **Main** tab and change the **Probe Out** size to 8.0 Å. Run parKVFinder again.

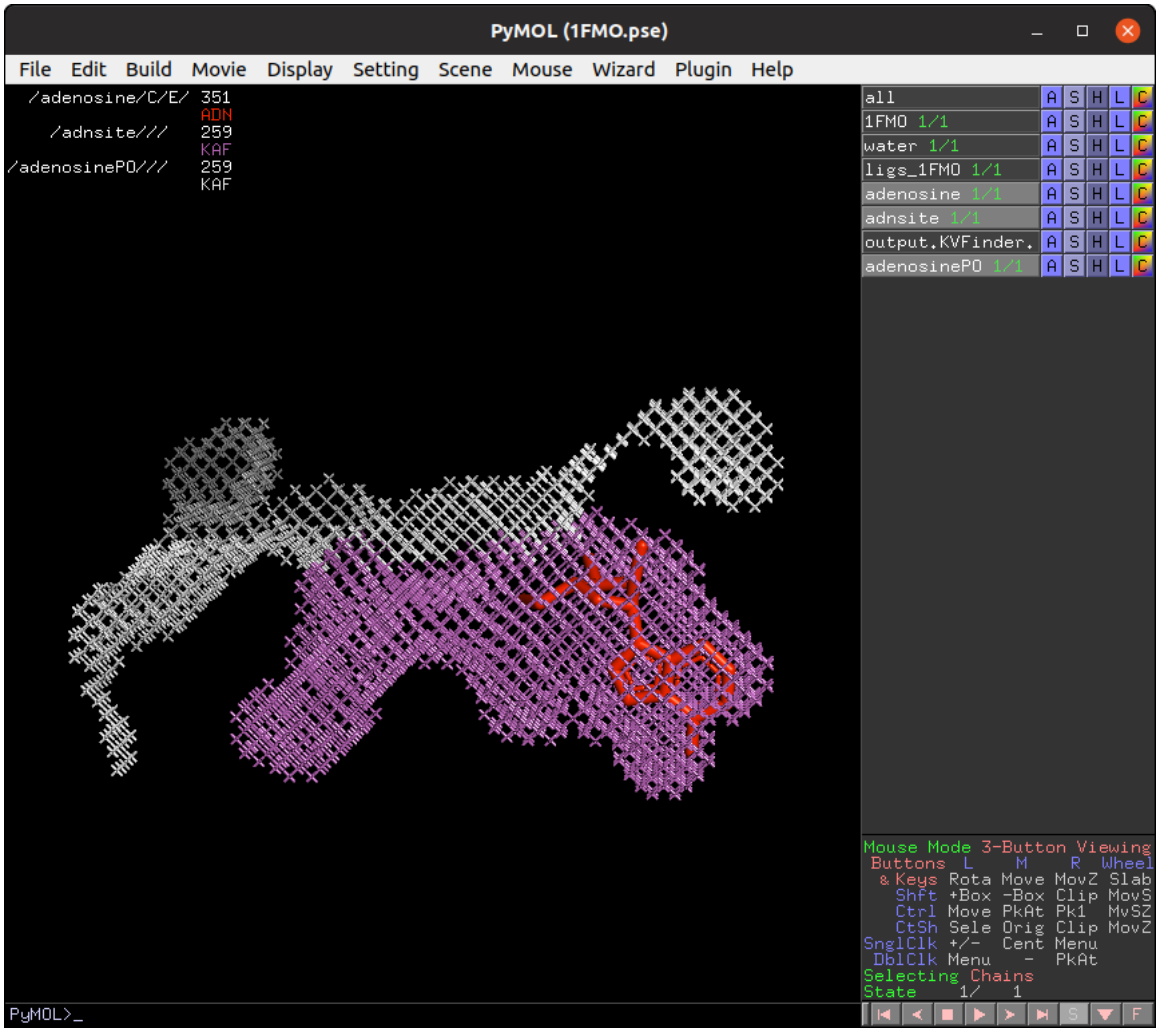


Again, copy the adenosine cavity (KAF) to a new object (adnsitePO).

```
# Copy new adenosine cavity
select resn KAF and output.KVFinder.output
create adnsitePO, sele
delete sele
```

Finally, prepare the PyMOL scene.

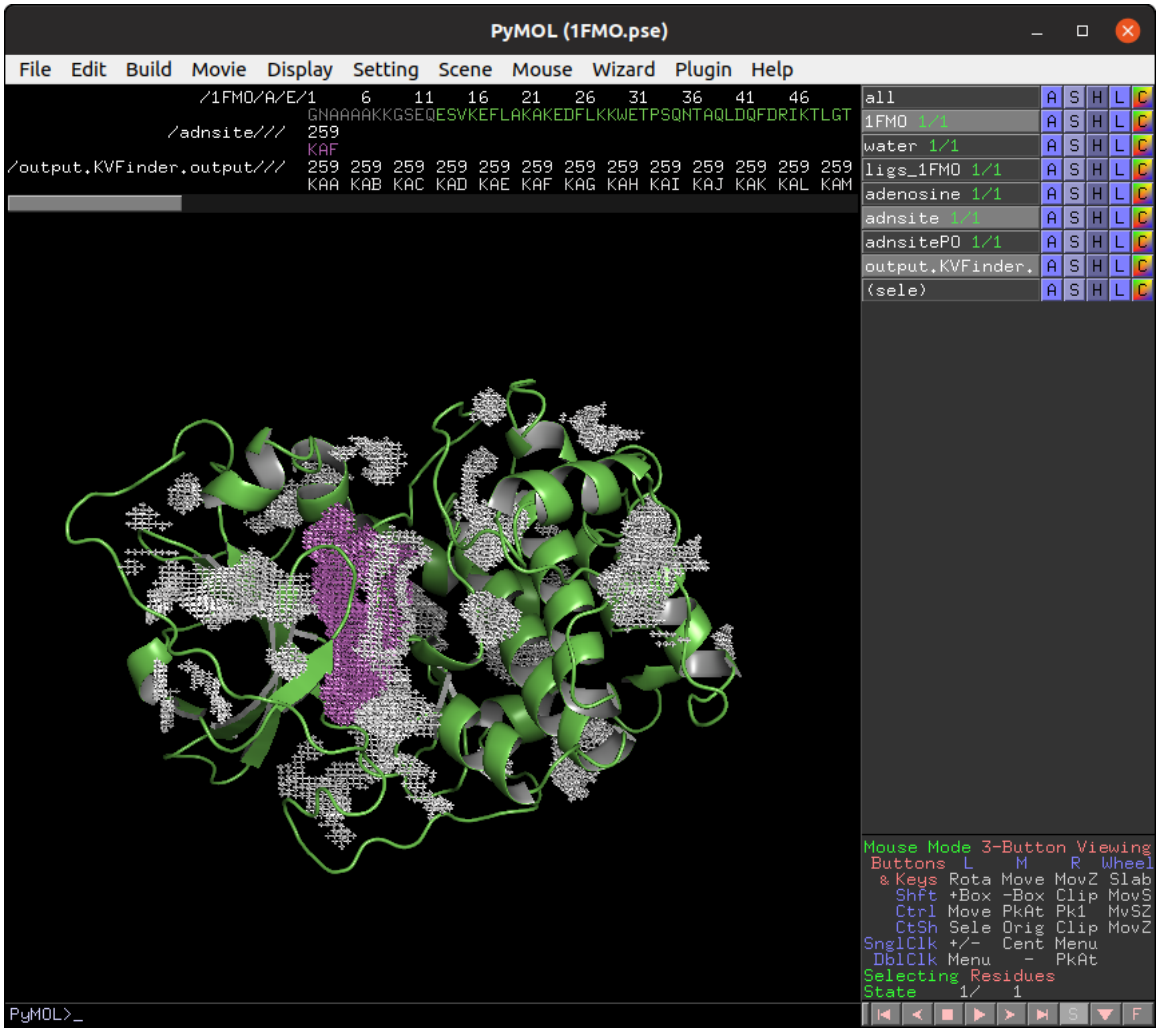
```
# Prepare PyMOL scene
disable
enable (adenosine, adnsite, adnsitePO)
```



Note that the adenosine cavity detected with the 4 Å Probe Out (magenta) has a lower ceiling than that detected with the 8 Å probe (white). Therefore, by increasing the size of the Probe Out, the cavity ceiling is also raised.

Adjusting Removal Distance

Besides adjusting the Probe Out size, we can also adjust the Removal Distance to change the cavity ceiling. So let's go back to the **Main** tab and change the **Removal Distance** to 1.2 Å and the size of **Probe Out** back to 4.0 Å. Run parKVFinder again.

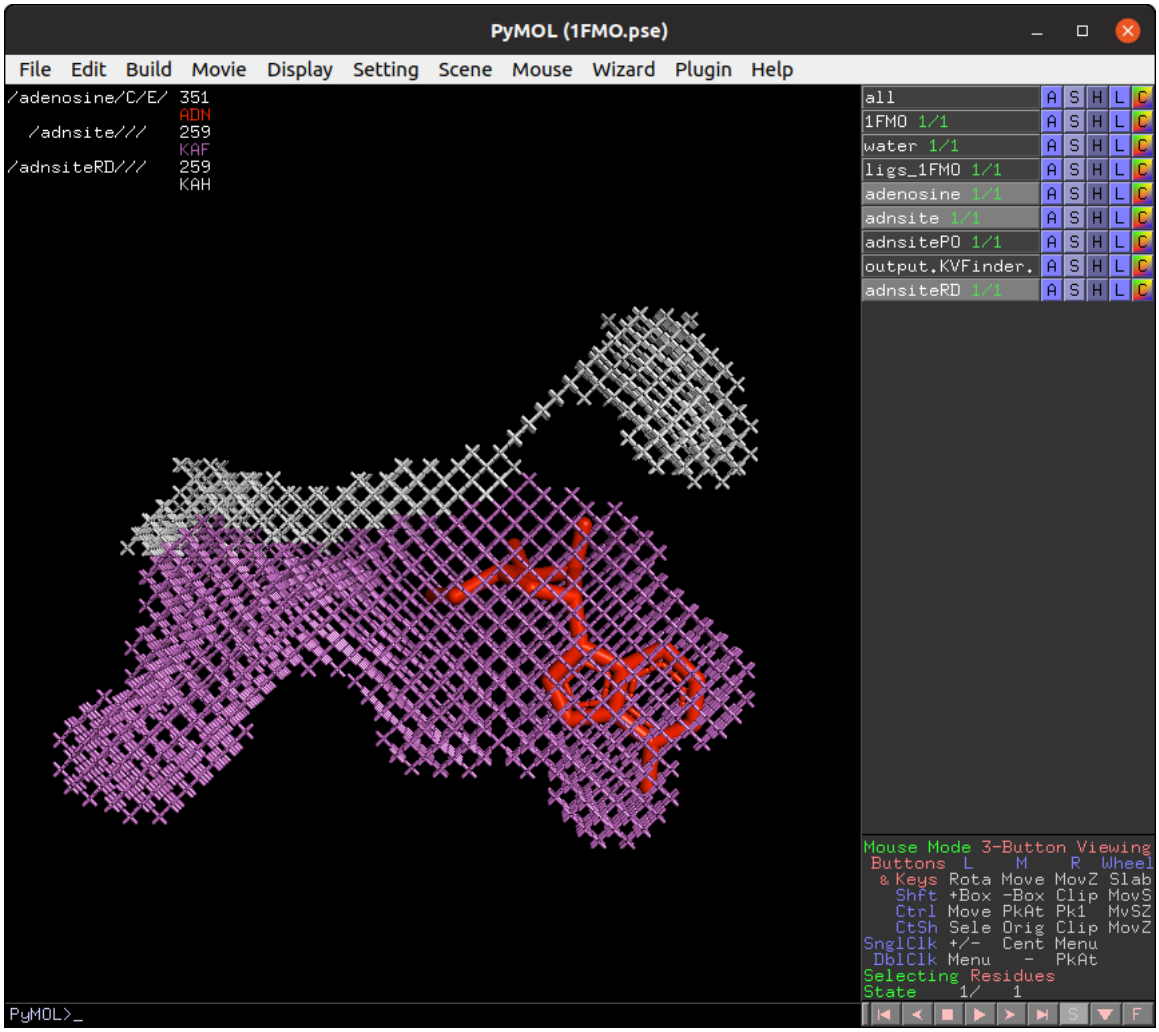


Again, copy the adenosine cavity (KAH) to a new object (adnsiteRD).

```
# Copy new adenosine cavity
select resn KAH and output.KVFinder.output
create adnsiteRD, sele
delete sele
```

Finally, prepare the PyMOL scene.

```
# Prepare PyMOL scene
disable
enable (adenosine, adnsite, adnsiteRD)
```



Note that the adenosine cavity detected with the 2.4 Å Removal Distance (magenta) has a lower ceiling than that detected with the 1.2 Å (white). Therefore, by decreasing the Removal Distance, the cavity ceiling is also raised.

Furthermore, changing the cavity ceiling by varying Probe Out and Removal Distance also affects cavity segregation.

Note: Usually the Removal Distance adjustment is less time consuming than the Probe Out adjustment for similar effects.

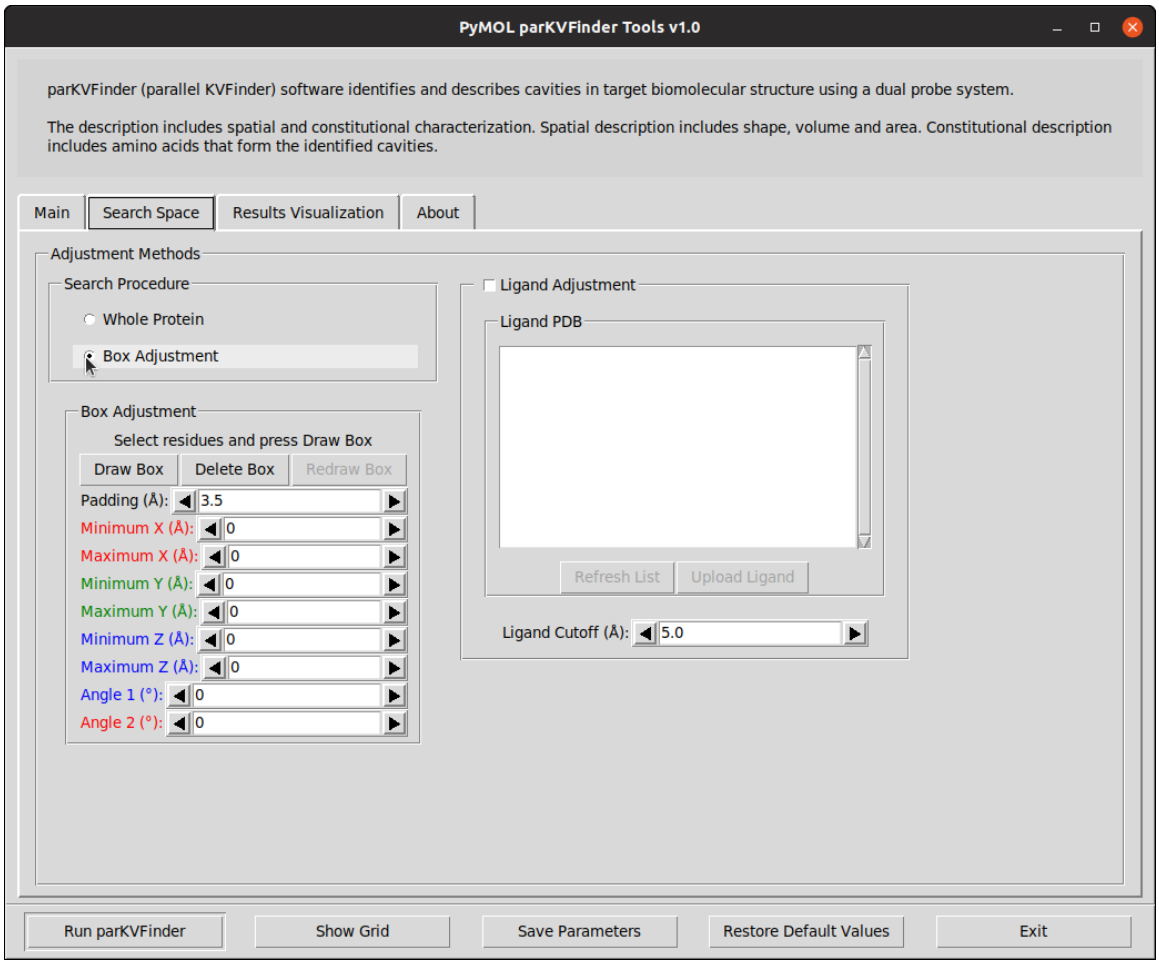
Steered detection

An important feature of parKVFinder is the steered detection of cavities. We continue our tutorial illustrating two distinct methods of cavity segmentation.

Box adjustment mode

Box adjustment mode explores closed regions with a custom box, which can be drawn via the GUI.

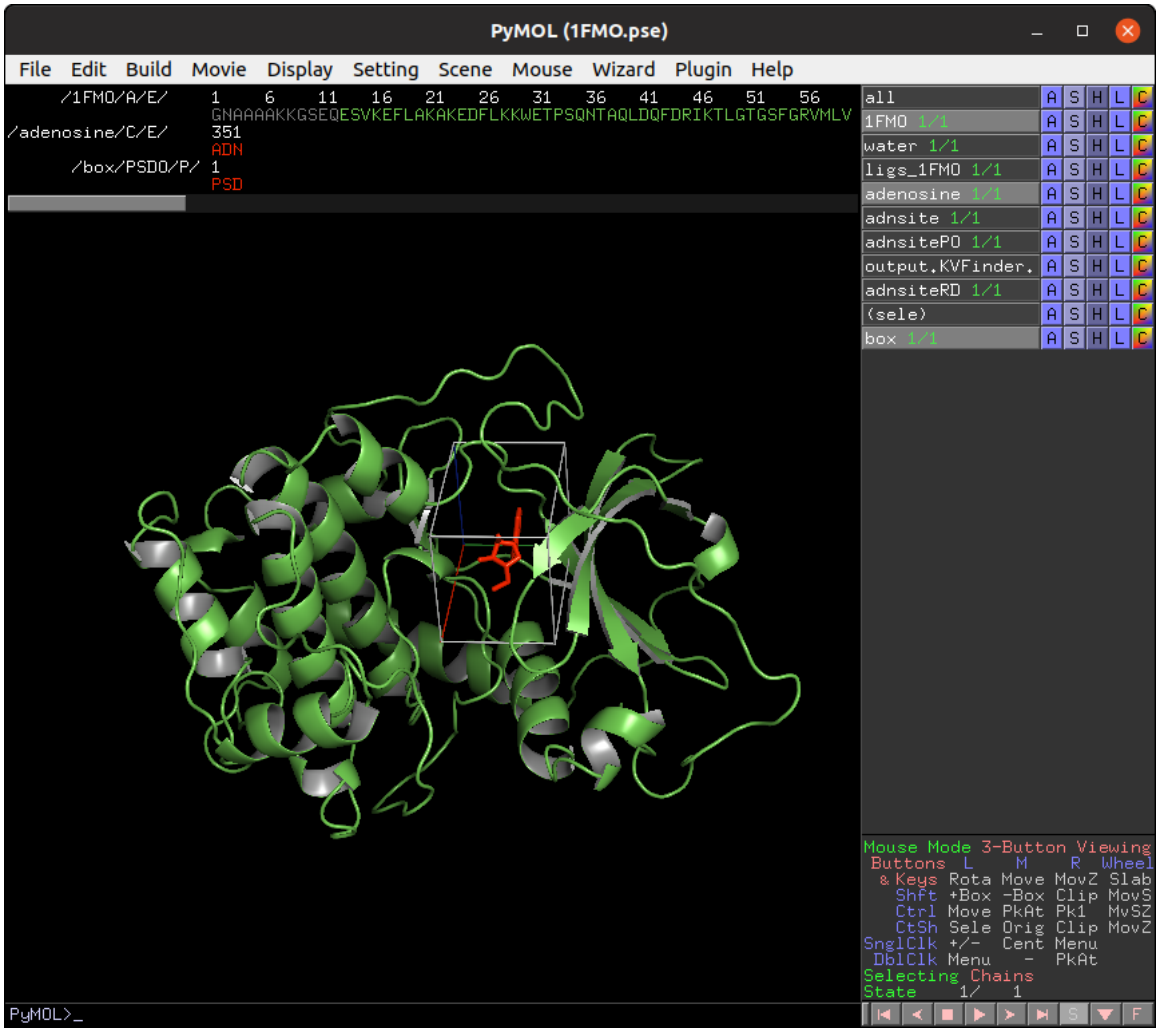
On the **Search Space** tab, select **Box Adjustment** option under **Search Procedure** group. This will enable a **Box Adjustment** frame, which handles the custom box in PyMOL viewer.



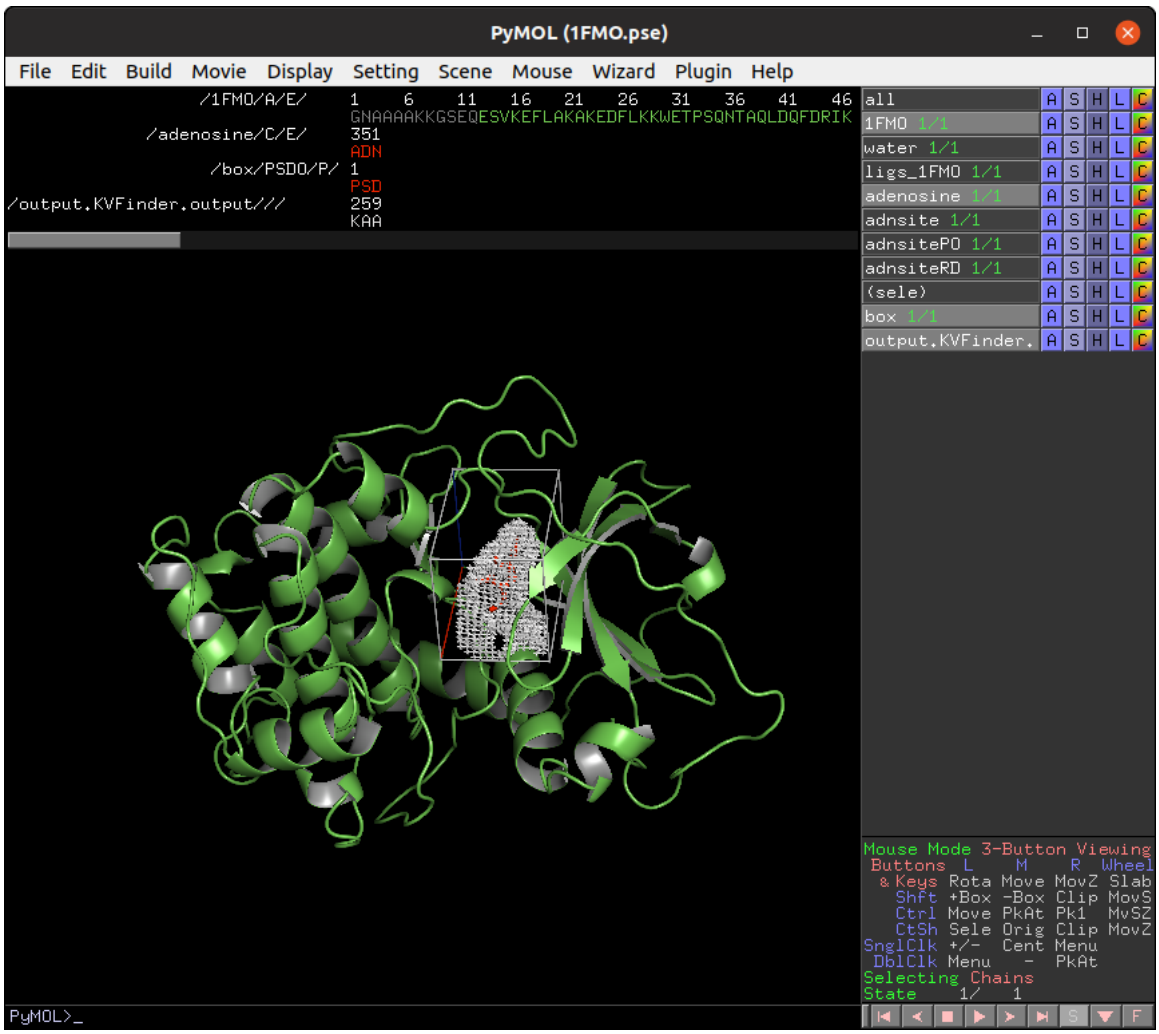
The custom box is drawn based on the (sele) object in the PyMOL viewer.

Then, select the adenosine ligand. This can be made on the PyMOL viewer by clicking on the ligand structure or using `select resn ADN` PyMOL command.

Click on **Draw Box** Button. This will create a custom box that limits the search space. It is fully customizable, but we will not change it for now.



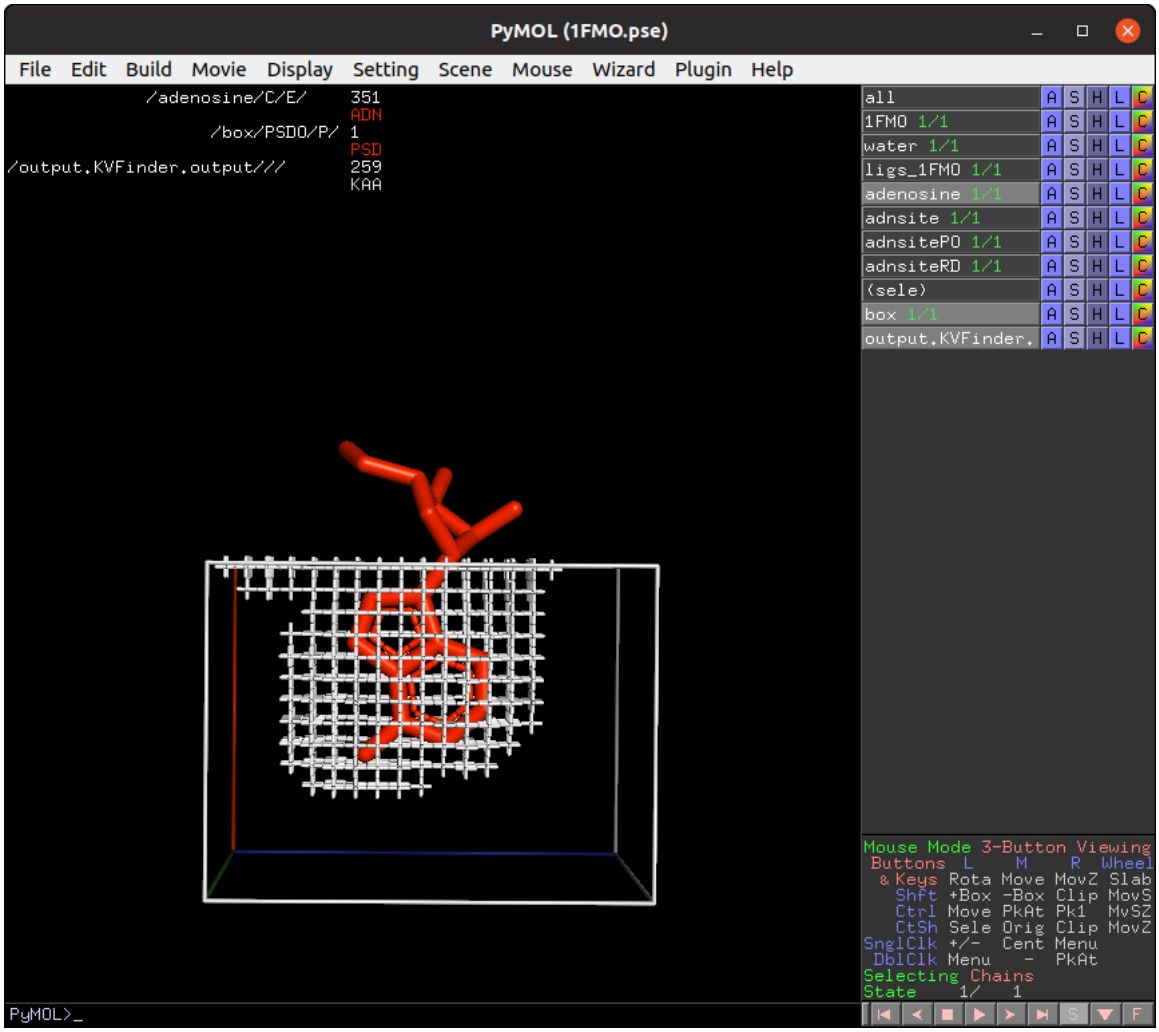
On the **Main** tab, change **Removal Distance** back to 2.4 Å and rerun parKVFinder.



Now, let's customize the box parameters to segment the binding site of our target protein.

Each axis is associated with one color (red with X, green with Y and blue with Z). The adjustment is made by the arrows or directly setting the value in the entry on the **Search Space** tab in the **Box Adjustment** group. We can also adjust the box angles by the same procedure. After altering the values, just click on **Redraw Box** button to redraw the box object using the new values.

Then, on the **Search Space** tab, reduce **Maximum X** to 1.0 Å and click **Redraw Box**. Rerun parKVFinder.



Lastly, click on **Delete Box** button to delete the custom box.

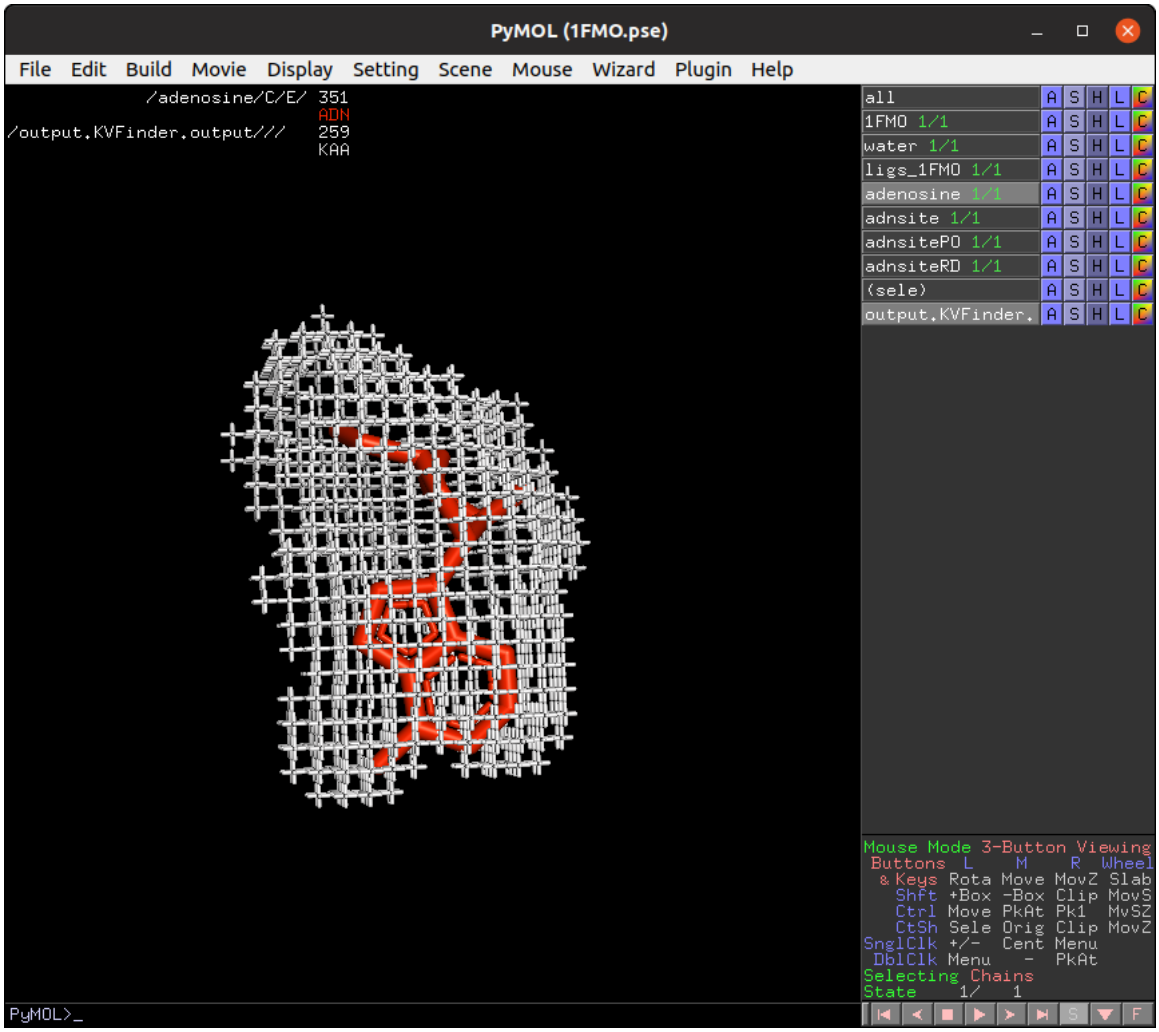
Ligand adjustment mode

A last feature is to limit the search around a structure. In this last example, let's do a whole protein prospection again, but limiting the search space around ligands.

First, on the **Search Space** tab, select **Whole Protein** option under **Search Procedure** group. This will disable the previous enabled **Box Adjustment** frame.

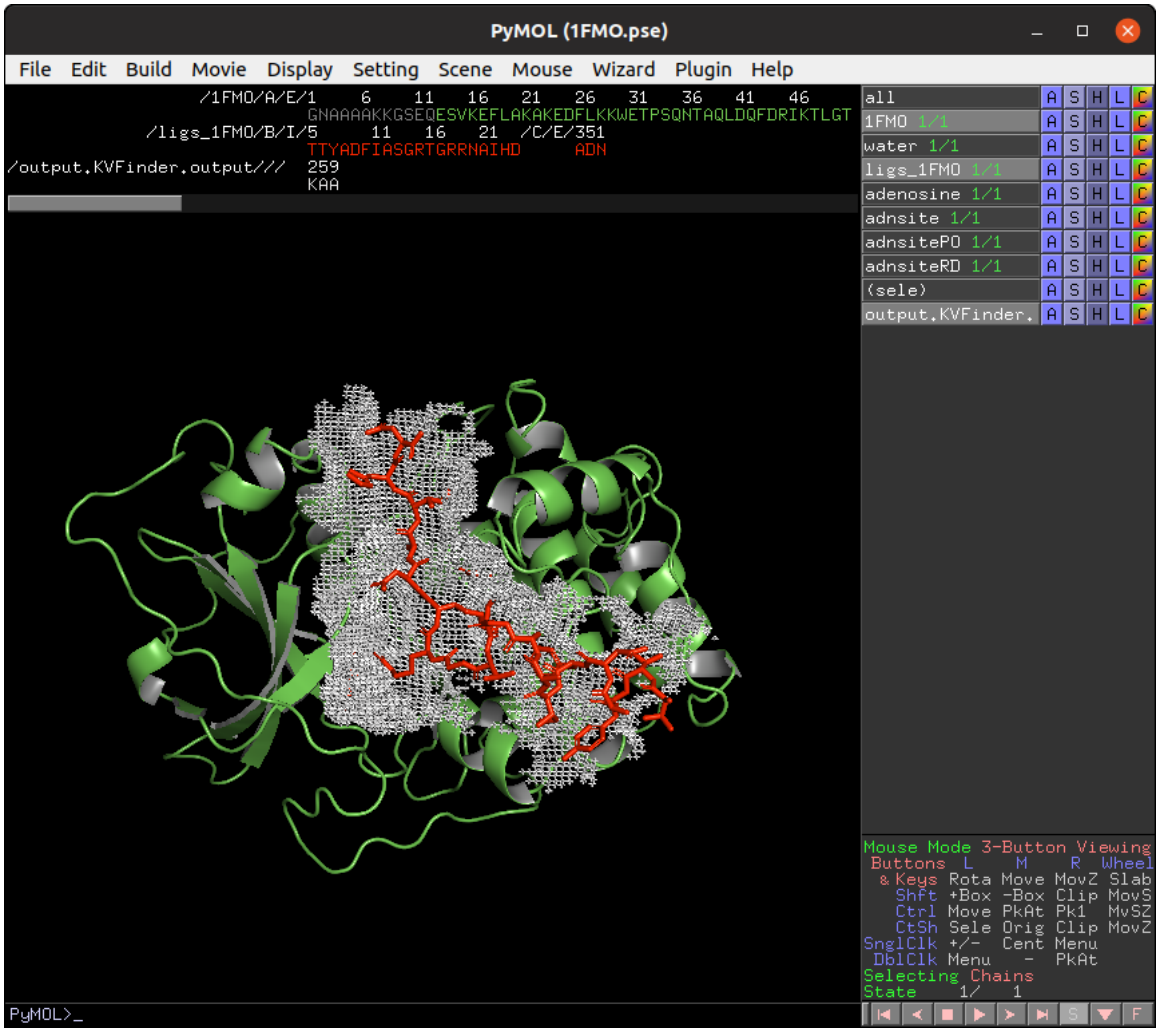
Still on the **Search Space** tab, click on the check button **Ligand Adjustment**. This will enable the buttons **Refresh List** and **Upload Ligand**.

Click the **Refresh List** button to display all objects in the scene in the **Ligand PDB** list box. Select the adenosine on the list box and reduce **Ligand Cutoff** to 3.0 Å. Run parKVFinder again.



Now, let's shift focus to the two ligands (adenosine and PKI) in the ligs_1FMO object.

On the **Search Space** tab, select the ligs_1FMO on the **Ligand PDB** list box and increase **Ligand Cutoff** back to 5.0 Å. Back on the **Main** tab, increase **Probe Out** to 10.0 Å and reduce **Removal Distance** to 0.0 Å. Run parKVFinder again.



Command line interface

parKVFinder has a command-line interface, which can be useful for molecular dynamics and high-throughput analysis. It also handles the same parameters available in parKVFinder PyMOL Tools, except for box rotations in box adjustment mode.

```
/home/user/parKVFinder$ parKVFinder
parKVFinder (parallel KVFinder) software identifies and describes cavities in
target biomolecular structure using a dual probe system.

The description includes spatial and constitutional characterization. Spatial
description includes shape, volume and area. Constitutional description includes
amino acids that form the identified cavities.

Usage: parKVFinder <.pdb> [options],
    where PDB is a path to a target PDB file.

Options:
  -h, --help
      Display this help message.
  -v, --version
```

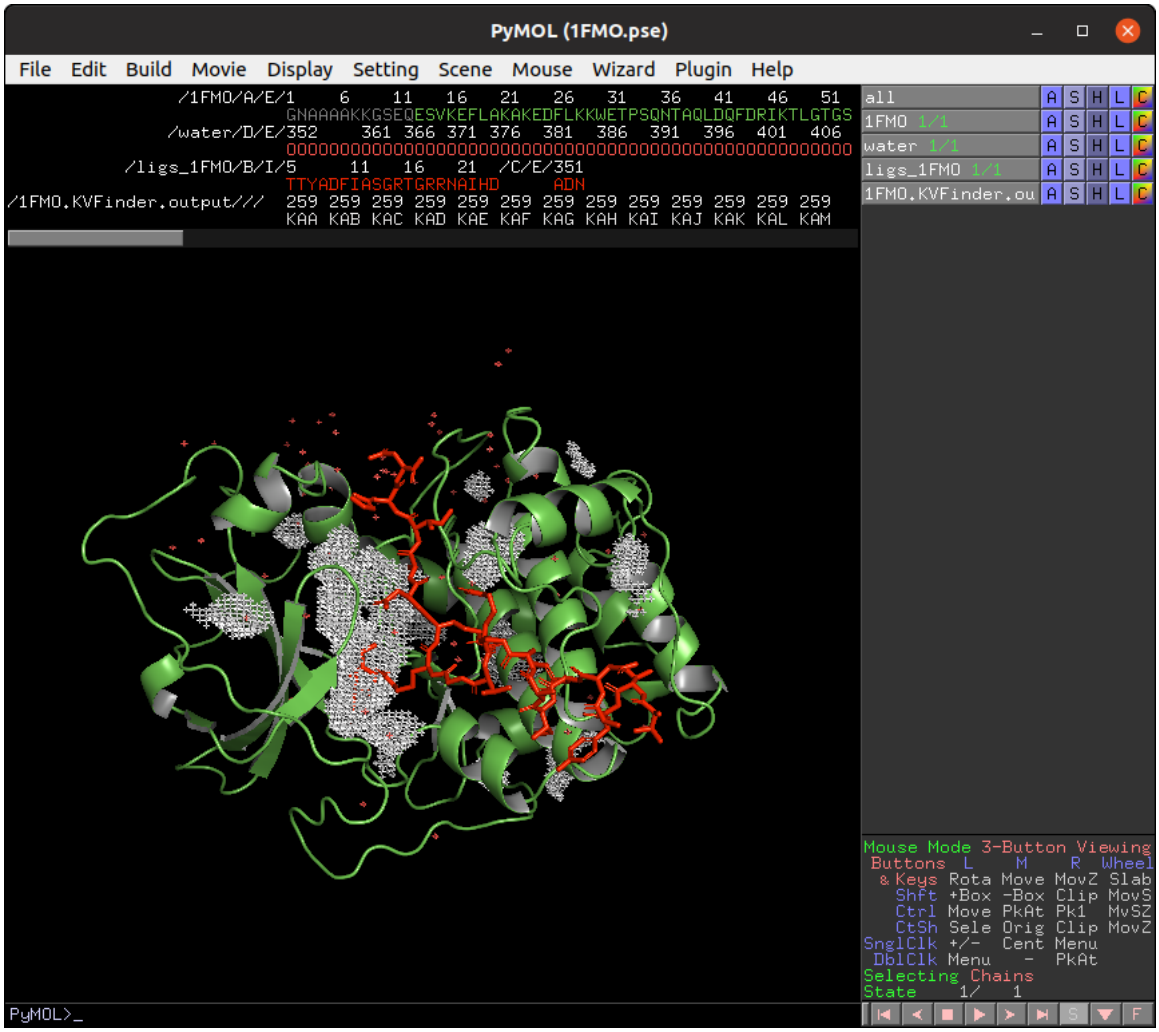
```
      Display parKVFinder version.
  --verbose
      Print extra information to stdout.
```

Whole Protein detection

Standard execution of the command line interface only requires the PDB file path of the target protein. So let's repeat the execution of 1FMO protein with default parameters.

```
/home/user/parKVFinder$ parKVFinder$ parKVFinder input/1FMO.pdb
[PID 8504] Running parKVFinder for: /home/user/parKVFinder/input/1FMO.pdb
done!
Elapsed time: 0.78 seconds
```

To view the cavities, the cavities PDB file must be loaded into PyMOL.



Custom execution

There a set of options for customizing parKVFinder's command line execution. These options are displayed in a help menu with their default values when applicable.

```
/home/user/parKVFinder$ parKVFinder$ parKVFinder -h
================================================================================
============================ parKVFinder help menu ============================
parKVFinder (parallel KVFinder) software identifies and describes cavities in
target biomolecular structure using a dual probe system.

The description includes spatial and constitutional characterization. Spatial
description includes shape, volume and area. Constitutional description includes
amino acids that form the identified cavities.

Usage: parKVFinder <.pdb> [options],
    where PDB is a path to a target PDB file.

Options:
  -h, --help
      Display this help message.
  -v, --version
      Display parKVFinder version.
  --verbose
      Print extra information to stdout.

General options:
  -p, --parameters  [<.toml>]
      Define path to parameters file.
  -d, --dictionary  [<dictionary>]
      Define path to a custom dictionary file.
  -r, --resolution  <enum>      (Low)
      Define resolution mode. Options include: Off, Low, Medium and High.
  -s, --step        <real>      (0.0)
      Define step size (grid spacing).
  -i, --probe_in    <real>      (1.4)
      Define probe in size.
  -o, --probe_out   <real>      (4.0)
      Define probe out size.
  --volume_cutoff   <real>      (5.0)
      Define cavities volume filter.
  --removal_distance    <real>      (2.4)
      Define removal distance when comparing probes surfaces.
  -t, --template            (paramters.toml)
      Create a parameter file template with defined parameters in current
      working directory.

Box adjustment options:
  -B, --box
      Define a search box mode where parKVFinder will detect cavities.
  --custom_box      [<file>]
      Define a custom search box based on a file containing the minimum and
      maximum cartesian values of each axis in angstrom.
  --residues_box    [<file>]
      Automatically set a search box based a file containing a tab-separated
      list of residues.
  --padding     <real>      (3.5)
      Define residues box padding. Adds a length in each box direction.
```

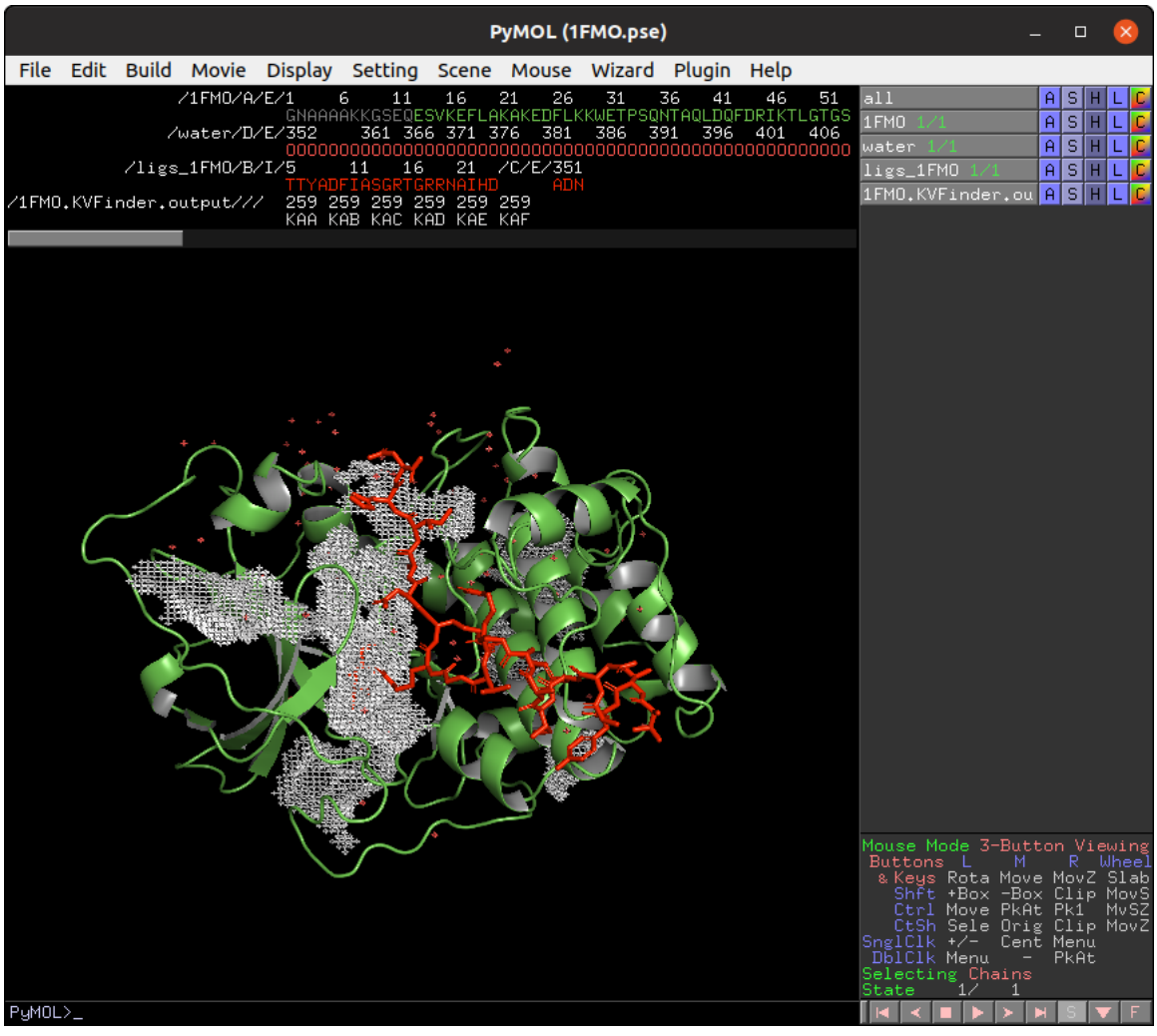
```
Surface options:
  -S, --surface     <enum>      (VdW)
      Define a surface representation. Options include: SAS and VdW. SAS
      specifies solvent accessible surface. VdW specifies van der Waals
      molecular surface.

Ligand options:
  -L, --ligand      [<.pdb>]
      Define path to ligand PDB file.
  --ligand_cutoff   <real>      (5.0)
      Define ligand radius distance cutoff.

================================================================================
================================================================================
```

So let's repeat the previous analysis, but changing **Probe Out** and **Volume Cutoff** parameters.

```
/home/user/parKVFinder$ parKVFinder$ parKVFinder input/1FMO.pdb --probe_out 8.0 --volume_cutoff 100.
[PID 12089] Running parKVFinder for: /home/user/parKVFinder/input/1FMO.pdb
done!
Elapsed time: 1.99 seconds
```



If we set the verbose flag, extra information will be printed in stdout.

```
/home/user/parKVFinder$ parKVFinder$ parKVFinder input/1FMO.pdb --probe_out 8.0 --volume_cutoff 100.
[PID 12198] Running parKVFinder for: /home/user/parKVFinder/input/1FMO.pdb
> Setting 'dictionary_name' to default file: /home/user/parKVFinder/dictionary
> Setting 'probe_in' to default value: 1.40
> Setting 'removal_distance' to default value: 2.40
> Setting 'step' (grid spacing) to 0.00.
> Setting 'resolution' to flag: Low
> Setting 'ligand_cutoff' to default value: 5.00
> Chosen 'surface' representation: VdW
> Running parKVFinder for whole biomolecular structure
> Loading atomic dictionary file
> Calculating grid dimensions
> Reading PDB coordinates
> Creating grid
> Filling grid with probe in surface
> Filling grid with probe out surface
> Defining biomolecular cavities
> Calculating volume
> Calculating surface points
> Calculating area
> Retrieving residues surrounding cavities
> Writing cavities PDB file
> Writing results file
done!
Elapsed time: 1.99 seconds
```

Box Adjustment modes

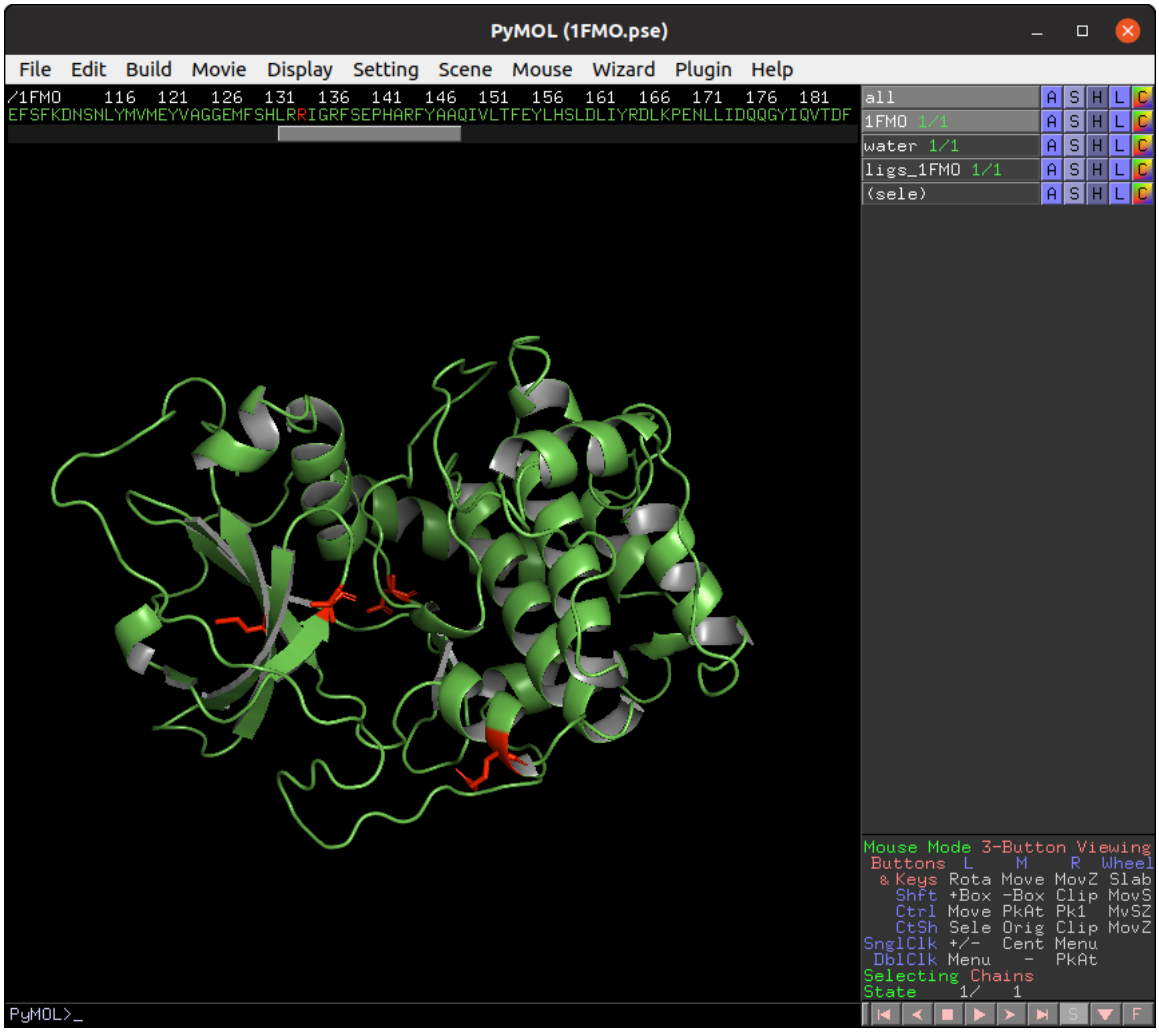
The command line interface takes a different approach to box adjustment mode compared to PyMOL parKVFinder Tools. As the GUI is not available to drawn the custom box, the command line interface set the custom box based on a list of residues (residues_box) or a set of cartesian coordinates (custom_box).

So let's execute parKVFinder using both box approaches.

Residues box

First, we need to define a list of residues in a tab-separated file, in which we specify the residues as the residues number and chain identifier separated by an underscore character (_).

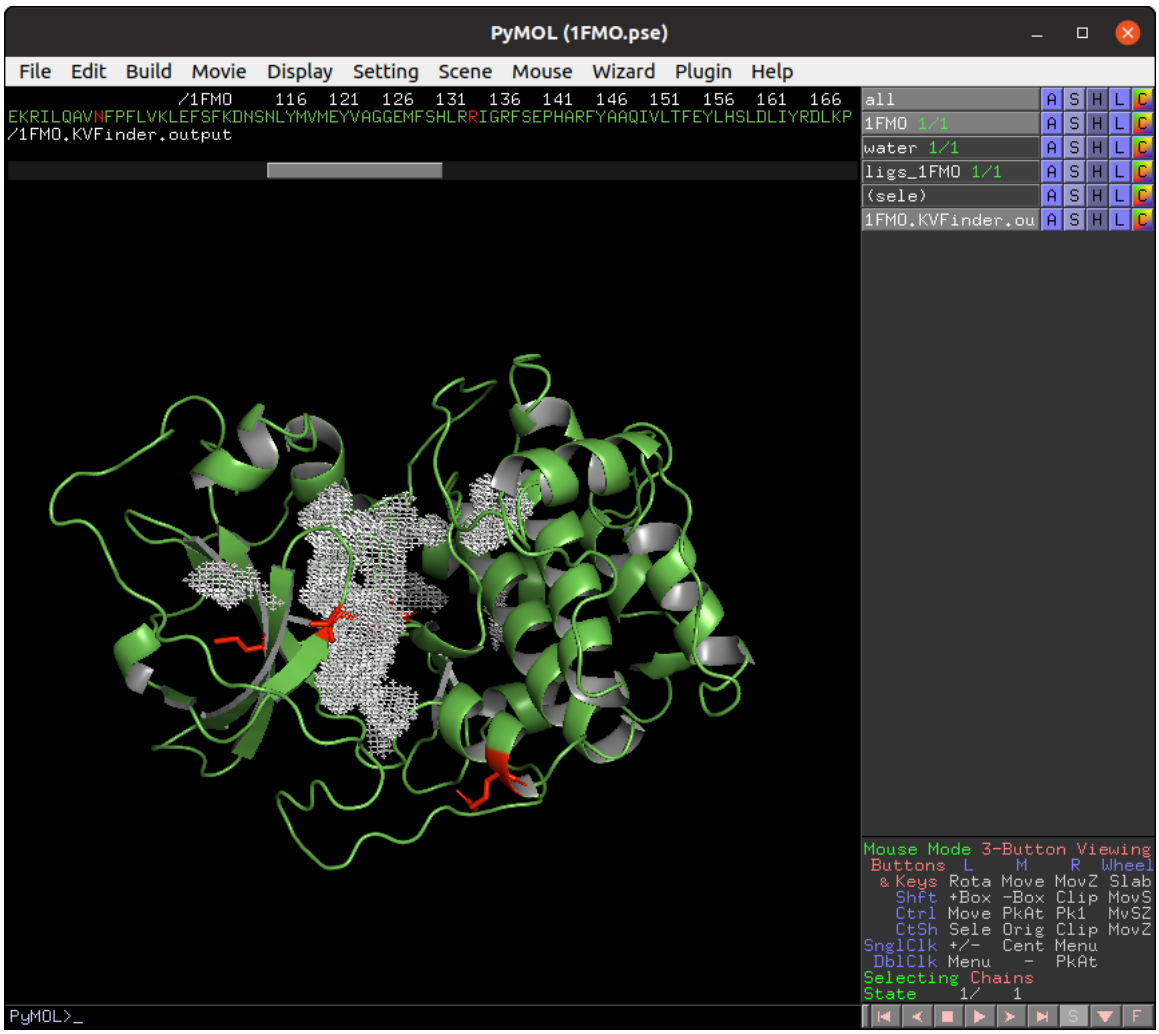
Considering the example file called 1FMO.residues.KVFinder.in, located inside input directory, we select the residues 51E, 71E, 99E and 134E (highlighted in red) to draw the custom box.




```
/home/user/parKVFinder$ parKVFinder$ cat input/1FMO.residues.KVFinder.in
51_E    71_E    99_E    134_E
```

So let's run parKVFinder for the 1FMO protein using a custom box based on this set of residues. A custom box will be drawn around these residues with a default padding, which can be changed using the `--padding <real>` option.

```
/home/user/parKVFinder$ parKVFinder$ parKVFinder input/1FMO.pdb -B --residues_box input/1FMO.residue
[PID 15230] Running parKVFinder for: /home/user/parKVFinder/input/1FMO.pdb
done!
Elapsed time: 0.27 seconds
```



Note: This method uses the same procedure as PyMOL parKVFinder Tools to draw the custom box in Box Adjustment mode.

Cartesian box

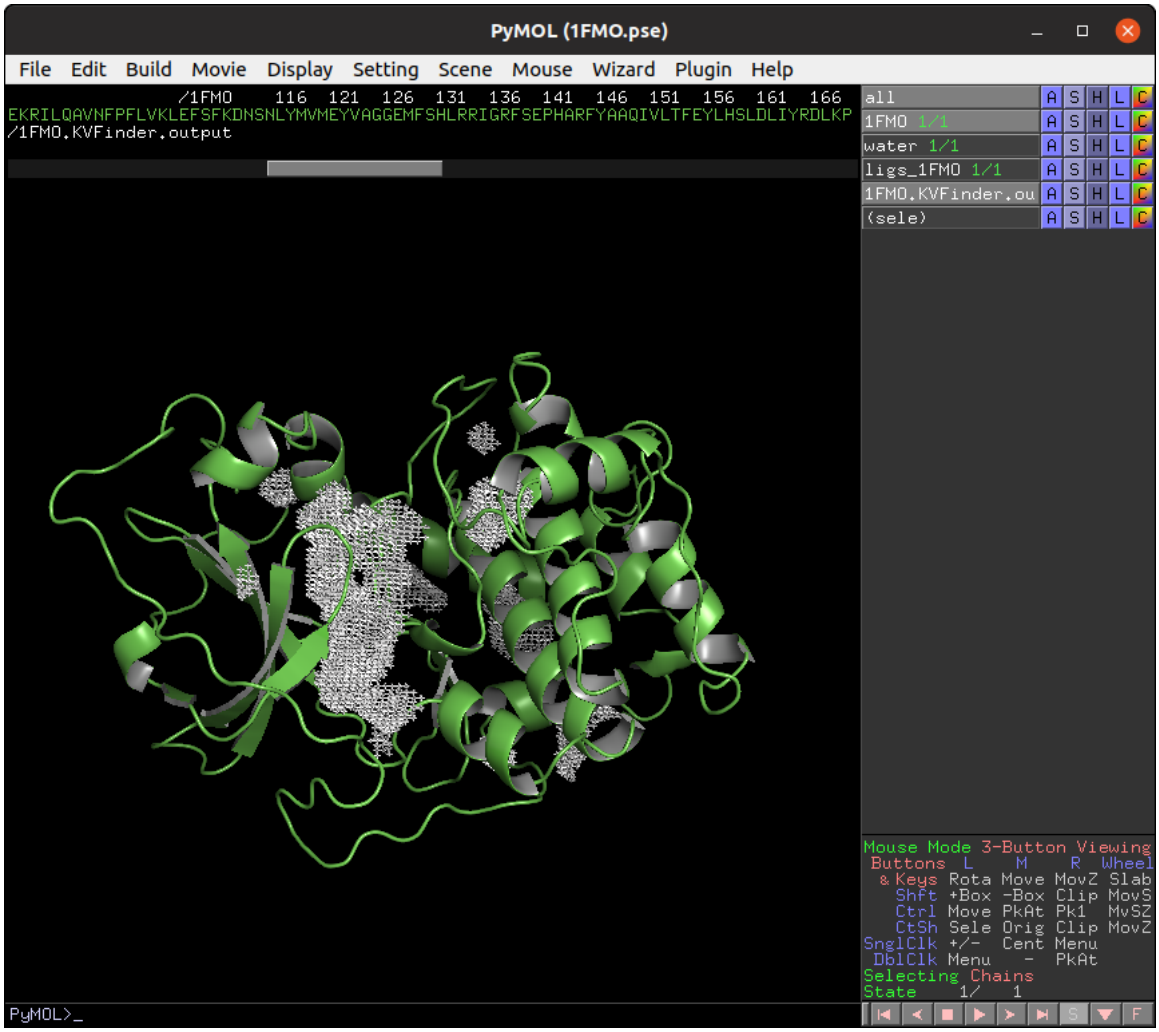
First, we need to define a tab-separated file containing the minimum and maximum cartesian values of each axis in the following order: Xmin, Xmax, Ymin, Ymax, Zmin and Zmax.

Considering the example file called 1FMO.box.KVFinder.in, located inside input directory, to draw the custom box.

```
/home/user/parKVFinder$ parKVFinder$ cat input/1FMO.box.KVFinder.in
-20.0    20.0    -20.0    20.0    -20.0    20.0
```

So let's run parKVFinder for the 1FMO protein using a custom box based on this set of coordinates.

```
/home/user/parKVFinder$ parKVFinder$ parKVFinder input/1FMO.pdb -B --custom_box input/1FMO.box.KVFin
[PID 16070] Running parKVFinder for: /home/user/parKVFinder/input/1FMO.pdb
done!
Elapsed time: 0.42 seconds
```


PyMOL (1FMO.pse)
File Edit Build Movie Display Setting Scene Mouse Wizard Plugin Help
/1FMO 116 121 126 131 136 141 146 151 156 161 166
EKRILQAVNFPFLVKLEFSFKDNSNLYMVMEYVAGGEMFSHLRRIGRFSEPHARFYAAQIVLTFEYLHSLDLIYRDLKP
/1FMO.KVFinder.output
all
1FMO 1/1
water 1/1
ligs_1FMO 1/1
1FMO.KVFinder.ou
(sele)
Mouse Mode 3-Button Viewing
Buttons L M R Wheel
& Keys Rota Move MovZ Slab
Shft +Box -Box Clip MovS
Ctrl Move PkAt Pk1 MvSZ
CtSh Sele Orig Clip MovZ
SnglClk +/- Cent Menu
DblClk Menu - PkAt
Selecting Chains
State 1/ 1
PyMOL>_

Manual

This is a comprehensive list of all PyMOL parKVFinder Tools and parKVFinder command line interface commands.

parKVFinder PyMOL Tools

Run parKVFinder: Executes parKVFinder. First, it checks the consistency of parKVFinder parameters. If any of these conditions fail, an appropriate warning message will be displayed. The cavity PDB file is loaded into the PyMOL viewer and the results information is displayed on the Results Visualization tab.

Show Grid: Creates a box of the search space for a target structure (selected on Input PDB list box). It depends on Probe Out size and slightly on grid spacing (Step Size or Resolution options).

Save Parameters: Saves a parameters file (parameters.toml) to the current output directory without running parKVFinder.

Restore Default Values: Restores the initial values of all parameters at startup of PyMOL parKVFinder Tools.

Exit: Exits PyMOL parKVFinder Tools.

Main tab

This tab features the main parameters and file paths for customizing parKVFinder's detection capabilities.

Main Options

Input PDB: This list box defines which structure will be used as input in parKVFinder execution. The list is generated using molecular objects loaded in the PyMOL viewer.

Refresh List: Refreshes the Input PDB list box of objects loaded in PyMOL viewer.

Upload PDB: Loads a PDB file into PyMOL viewer and automatically refreshes the Input PDB list box.

Output Base Name: This entry field defines the prefix of the output files (cavities PDB file, results file, parameters file and output directory name).

Probe In: A smaller probe that rolls around the target protein. Usually set to a water molecule (1.4 Å).

Probe Out: A larger probe that rolls around the target protein, which depending on the characteristics of the target structure, user may adjust probe size.

Volume Cutoff: Defines a filter to exclude cavities with smaller volumes than this limit. Useful to exclude uninteresting cavities from results.

Removal Distance: Defines a distance to be removed from the cavity ceiling. This length is converted to grid units by dividing it by grid spacing.

Step Size: Directly defines the grid spacing used in parKVFinder execution, which defines the number of voxels in the grid. Users can define only one grid spacing method (Step size or Resolution).

Resolution: Indirectly defines the grid spacing used in parKVFinder execution, which defines the number of voxels in the grid. There are three possible resolutions: Low (0.6 Å), Medium (0.50 Å) and High (0.25 Å). Users can define only one grid spacing method (Step size or Resolution).

Surface Representation: Selects surface type to be considered, Solvent Accessible Surface (SAS) or Molecular Surface (VdW).

Cavity Representation: Selects how parKVFinder exports its cavities PDB file. The Filled option exports all cavity points, and the Filtered option filters out some internal points before exporting cavities. This parameter affects the cavities visualization and memory requirement in PyMOL.

File Locations

parKVFinder: Defines the path to the parKVFinder executable.

Dictionary: Defines the path to the parKVFinder van der Waals dictionary, which can be customized. The dictionary file contains the Van der Waals radii for atoms of the recognized residues. This file has a mandatory format:

```
--------------------------------------------------------------------------------
# This is a comment line

#>RES (residue name in 3 letter code)
#Atom_name\t\tAtom_radius

>ALA
N       1.8240
H       0.6000
HB1     1.4870

>ARG
CB      1.9080
HB2     1.4870
2HB     1.4870

>GEN
AC      2.00
AG      1.72
AL      2.00
--------------------------------------------------------------------------------
```

Output Directory: Path to a specific output directory where all files created by parKVFinder will be saved.

Search Space tab

This tab features options for customizing parKVFinder's search space.

Search Procedure: Selects search space to be considered, Whole Protein or Box Adjustment options. The Whole Protein mode defines the search space as the target structure. The Box Adjustment mode defines the search space as a custom box. When selecting the Box Adjustment mode, a Box Adjustment frame is enabled, which allows custom box adjustment.

Box Adjustment

Draw Box: Creates a box around a user selection - (sele) object. When the box object is drawn, Draw Box button is disabled.

Delete Box: Deletes the box object from PyMOL viewer and enable Draw Box button.

Redraw Box: Redraw box based on a user selection - (sele) object - and other box parameters (padding, minimum X, maximum X, minimum Y, maximum Y, minimum Z, maximum Z, angle 1 and angle 2).

Padding: Defines padding around the selection which the box will be created.

Minimum/Maximum vertices: These entries adjust the box size. Each axis is associated with a color (red with X, green with Y and blue with Z). The base reference of the box is the vertex where the colored edges meet. Minimum vertices move the edge next to the base, while maximum vertices control the opposite side of the edge.

Angle 1: Rotates X axis.

Angle 2: Rotates Z axis.

Ligand Adjustment

Ligand Adjustment: When enabled, this mode limits the search space around a ligand. The ligand is defined in the Ligand PDB list box, while the search radius is defined by the Ligand Cutoff entry.

Ligand PDB: This list box defines which structure will be used as ligand in parKVFinder execution. The list is generated using molecular objects loaded in the PyMOL viewer.

Refresh List: Refreshes the Ligand PDB list box of objects loaded in PyMOL viewer.

Upload Ligand: Loads a PDB file into PyMOL viewer and automatically refreshes the Ligand PDB list box.

Ligand Cutoff: Defines the search radius from the ligand structure.

Results Visualization tab

This tab displays parKVFinder results.

Information

Results File: Button to select a results file to load.

Load: Loads the information from results file.

Input File: Path to input PDB file.

Ligand File: Path to ligand PDB file.

Output File: Path to output PDB file.

Step Size: Results grid spacing.

Descriptors

Volume: Displays the calculated volume for each cavity, identified by a three-letter code. If the user selects a cavity, an active object is created in PyMOL viewer for visual inspection.

Surface Area: Displays the calculated surface area for each cavity, identified by a three-letter code. If the user selects a cavity, an active object is created in PyMOL viewer for visual inspection.

Interface Residues: Displays residues around the selected cavities on this list on an active object in the PyMOL viewer.

Command line interface

parKVFinder command line interface handles the same parameters available in parKVFinder PyMOL Tools, except for a different box adjustment treatment.

```
# Usage
parKVFinder <.pdb> [options]
```

where <.pdb> is a path to a target PDB file.

All customizable options are set using short and/or long command line arguments.

List of options:

- `-h` or `--help`: Display this help message.

```
# Usage
parKVFinder -h
parKVFinder --help
```

- `-v` or `--version`: Display parKVFinder version.

```
# Usage
parKVFinder -v
parKVFinder --version
```

- `--verbose`: Print extra information to stdout.

```
# Usage
parKVFinder <.pdb> --verbose
```

General options

- `-p` or `--parameters` <.toml>: Define path to parameters file.

```
# Usage
parKVFinder <.pdb> -p <.toml>
parKVFinder <.pdb> --parameters <.toml>
```

- `-d` or `--dictionary` <dictionary>: Define path to a custom dictionary file.

```
# Usage
parKVFinder <.pdb> -d <file>
parKVFinder <.pdb> --dictionary <file>
```

Default: dictionary file in parKVFinder directory (KVFinder_PATH location).

The dictionary file contains the Van der Waals radii for atoms of the recognized residues. This file has a required format:

```
--------------------------------------------------------------------------------
# This is a comment line

#>RES (residue name in 3 letter code)
#Atom_name\t\tAtom_radius

>ALA
N       1.8240
H       0.6000
HB1     1.4870

>ARG
CB      1.9080
HB2     1.4870
2HB     1.4870

>GEN
AC      2.00
```

```
AG      1.72
AL      2.00
--------------------------------------------------------------------------------
```

- `-r` or `--resolution`: Define resolution mode. Options include: Off, Low, Medium and High.

```
# Usage
parKVFinder <.pdb> -r <enum>
parKVFinder <.pdb> --resolution <enum>
```

Default: `Low`.

- -s, –step <real>: Define step size (grid spacing).

```
# Usage
parKVFinder <.pdb> -s <real>
parKVFinder <.pdb> --step <real>
```

- -i, –probe_in <real>: Define probe in size.

```
# Usage
parKVFinder <.pdb> -i <real>
parKVFinder <.pdb> --probe_in <real>
```

Default: `1.4`.

- -o, –probe_out <real>: Define probe out size.

```
# Usage
parKVFinder <.pdb> -o <real>
parKVFinder <.pdb> --probe_out <real>
```

Default: `4.0`.

- –volume_cutoff <real>: Define cavities volume filter.

```
# Usage
parKVFinder <.pdb> --volume_cutoff <real>
```

Default: `5.0`.

- –removal_distance <real>: Define removal distance when comparing probes surfaces.

```
# Usage
parKVFinder <.pdb> --removal_distance <real>
```

Default: `2.4`.

- -t, –template: Create a parameter file template (parameters.toml) with defined parameters in the current working directory.

```
# Usage
parKVFinder <.pdb> -t
parKVFinder <.pdb> --template
```

Box adjustment options

- -B, –box: Define a search box mode where parKVFinder will detect cavities.

```
# Usage
parKVFinder <.pdb> -B --<method> <file>
parKVFinder <.pdb> --box --<method> <file>
```

- –custom_box <file>: Define a custom search box based on a file containing the minimum and maximum cartesian values of each axis in angstrom.

```
# Usage
parKVFinder <.pdb> -B --custom_box <file>
parKVFinder <.pdb> --box --custom_box <file>
```

The file must contain the minimum and maximum cartesian values of each axis in the following order: Xmin, Xmax, Ymin, Ymax, Zmin and Zmax. This file has a required format:

```
--------------------------------------------------------------------------------
Xmin    Xmax    Ymin    Ymax    Zmin    Zmax
--------------------------------------------------------------------------------
```

Note: The values are separated by tab.

- –residues_box <file>: Automatically set a search box based a file containing a tab-separated list of residues.

```
# Usage
parKVFinder <.pdb> -B --residues_box <file>
parKVFinder <.pdb> --box --residues_box <file>
```

The file must contain a list of residues, in which we specify the residues as the residues number and chain identifier separated by an underscore character (_). This file has a required format:

```
--------------------------------------------------------------------------------
resnum1_chain    resnum2_chain    [...]    resnumN_chain
--------------------------------------------------------------------------------
```

Note: The values are separated by tab.

- –padding <real>: Define residues box padding. Adds a length in each box direction.
  - Usage:
  - Default:

Surface options

- -S, –surface : Define a surface representation. Options include: SAS and VdW. SAS specifies solvent accessible surface. VdW specifies van der Waals molecular surface.
  - Usage:
  - Default:

Ligand options

- -L, –ligand <.pdb>: Define path to ligand PDB file.
  - Usage:
- –ligand_cutoff <real>: Define ligand radius distance cutoff.
  - Usage:
  - Default:

About

parKVFinder and PyMOL parKVFinder Tools was developed by:

- <author 1> (<e-mail 1>)
- <author 2> (<e-mail 2>)
- ...
- <author N> (<e-mail N>)

<Institution> (<website>).

If you have any further questions, inquires or if you wish to contribute to parKVFinder project, please contact us at <e-mail>.
