AngelRuizMoreno/Jupyter_Dock: Jupyter Dock is a set of Jupyter Notebooks for performing molecular docking protocols interactively, as well as visualizing, converting file formats and analyzing the results.

github.com/AngelRuizMoreno/Jupyter_Dock

AngelRuizMoreno

Molecular Docking integrated in Jupyter Notebooks

Description | **Citation** | **Installation** | **Examples** | **Limitations** | **License**

DOI 10.5281/zenodo.5514956

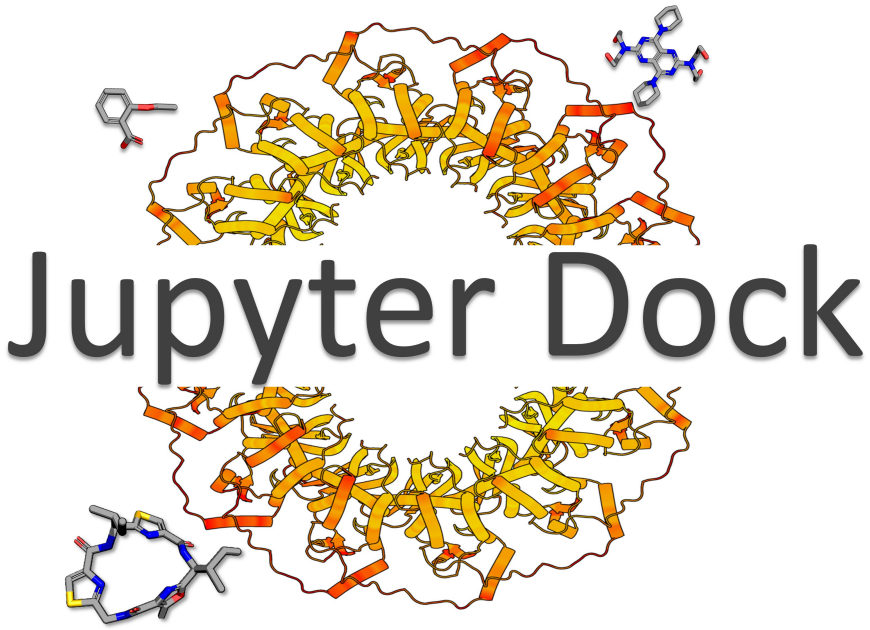


Table of content

Description

Jupyter Dock is a set of Jupyter Notebooks for performing molecular docking protocols interactively, as well as visualizing, converting file formats and analyzing the results.

See Jupyter Docks in action in my personal website: chem-workflows

These notebooks are Python 3 compatible. Each protocol and Jupyter notebook has its own test folder for testing and reproducibility evaluation.

For all notebooks, the demonstration includes the use of AutoDock Vina and Ledock. When available, some alternatives are mentioned in the protocol.

The notebooks includes whole protocols for:

1. Molecular Docking

> For any new user, this is a good place to start. Jupyter Docks' main stages for molecular docking, as well as all functions, methods and codes are described here along with brief explanations, hints, and warnings.

2. Virtual Screening

> Interested in docking multiple ligands into a single target site? This is what you require. This protocol covers all steps from ligand preparation to docking pose visualization in the target site of interest.

3. Blind Docking

> Do you want to dock multiple ligands into whole target surface and/or its pockets? This protocol demonstrates the entire process of pocket search and their use as potential organic molecule binding sites. **(Documentation in progress)**

4. Reverse Docking / Target fishing)

> Interested in docking one or a few molecules into a set of proteins to identify the most promising target(s)? This notebook covers all of the steps required to achieve such a goal in a condensed manner, making the process seem like a walk in the park. **(Documentation in progress)**

5. Scaffold-based Docking

> Do you want to use a molecular substructure as an anchor point for your ligands? This procedure demonstrates an approximation for running molecular docking while constraining the position of a portion of the ligand. This is especially useful for investigating novel ligands with similar structure to known binders. **(In construction)**

6. Covalent Docking

> Is your hypothesis that your ligand can bind to the target covalently? This protocol describes how to use freely available tools to investigate the covalent binding mode of ligands to protein targets. **(In construction)**

7. Docking Analysis

> Have you completed your docking experiments with Jupyter Dock or another approach and want to conduct a rational analysis? You've come to the right place. This notebook summarizes the most common docking analysis techniques, including score comparisons, z-score calculation between softwares, pose clustering, molecular interactions mapping, and more. **(In construction)**

Question about usage or troubleshooting? Please leave a comment here

Requirements

Jupyter Dock is reliant on a variety of academic software. The Jupyter Dock.yaml file contains all the tools and dependencies, but the following are some of the most noticeable:

- Autodock Vina
- AutoDock Tools
- LeDock and LePro
- Pymol API
- OpenBabel
- RDKit
- py3Dmol
- PDBFixer
- MDAnalysis
- ProLif
- Fpocket
- Meeko
- Smina

Installation

1. Installing all dependencies one by one:

1.1. Create a conda enviroment

```
conda create -n Jupyter_Dock python=3.8
conda activate Jupyter_Dock
```

1.2. Install de dependencies

PyMol

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

py3Dmol

```
conda install -c conda-forge py3dmol
```

AutoDock Vina

```
pip install vina
```

OpenBabel (Pybel)

```
conda install -c conda-forge openbabel
```

Meeko

```
pip install meeko
```

PDBFixer

```
conda install -c conda-forge pdbfixer
```

ProLif, RDKit and MDAnalysis

To achieve a successful installation, the current development of ProLif necessitates a number of dependencies and steps. The quickest way to get a fully functional ProLif is to follow the steps below:

```
conda install rdkit cython
```

Make sure your system has a working GCC before beginning the ProLif installation (needed to compile MDAnalysis).

```
pip install git+https://github.com/chemosim-lab/ProLIF.git
```

Smina, Fpocket,LeDock, LePro and AutoDock Tools executables are provided in the bin folder of this repo.

> It is highly likely that the AutoDock Tools binaries will need to be compiled on your machine before they can be used. In this case, you must download them from https://ccsb.scripps.edu/adfr/downloads/ and then follow the installation instructions.

To address this limitation, the notebooks include the use of Smina binary as an alternative to AutoDock Vina, which offers several advantages for protein and ligand preration and formats, among other things.

2. Available as GitHub repo:

2.1. Clone the repository and create the Jupyter_Dock environment from the Jupyter_Dock.yaml file.

```
$git clone https://github.com/AngelRuizMoreno/Jupyter_Dock.git
$cd Jupyter_Dock
$conda env create -f Jupyter_Dock.yml
```

After installing the prerequisites, you can activate the conda environment and run/modify the Jupyter Notebooks.

3. GoogleColab:

```
Not yet available.
```

Limitations

Jupyter Dock's initial goal was to provide a set of pythonic protocols for molecular docking. Nonetheless, there is a dearth of docking tools in Python for all of the steps and protocols (i.e. pocket search for blind docking). Furthermore, the majority of well-known and widely used molecular docking softwares have been developed as stand-alone executables or as components of software suites. As a result, Jupyter Dock evolved into a notebook, attempting to organize and compile the tools required for the rational implementation of molecular docking.

The following are the main drawbacks of this approach:

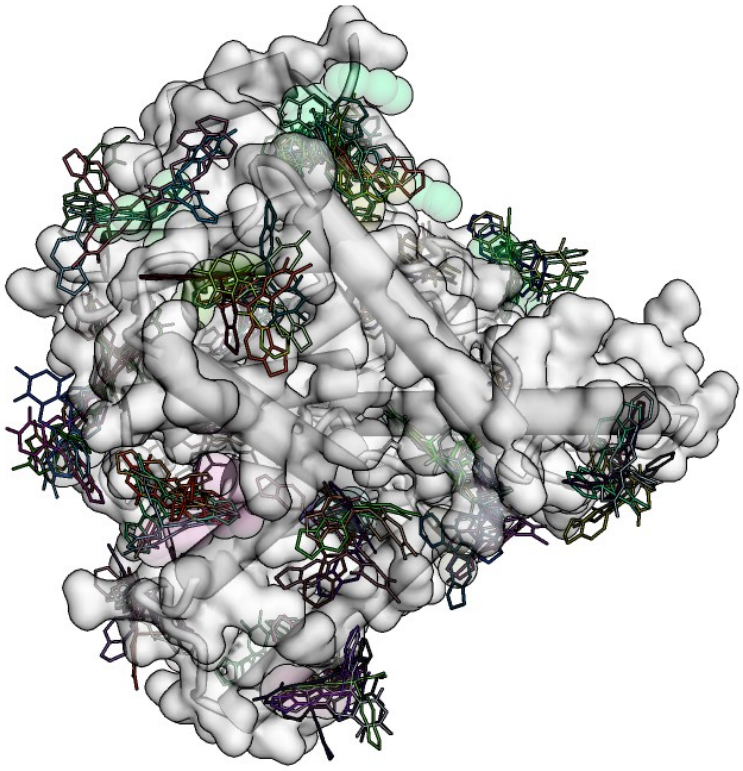
- The AutoDock Tools binary files "prepare_rotein" and "prepare_lgand are still missing for MacOs.
- There is no guarantee that these jupyter notebooks will run on Windows. The alternative for these users would be to run Jupyter Dock through the Windows Subsystem for Linux (WSL)

> Actually, I built Jupyter Dock in a Xiomi Mi Laptop Air using the WSL.

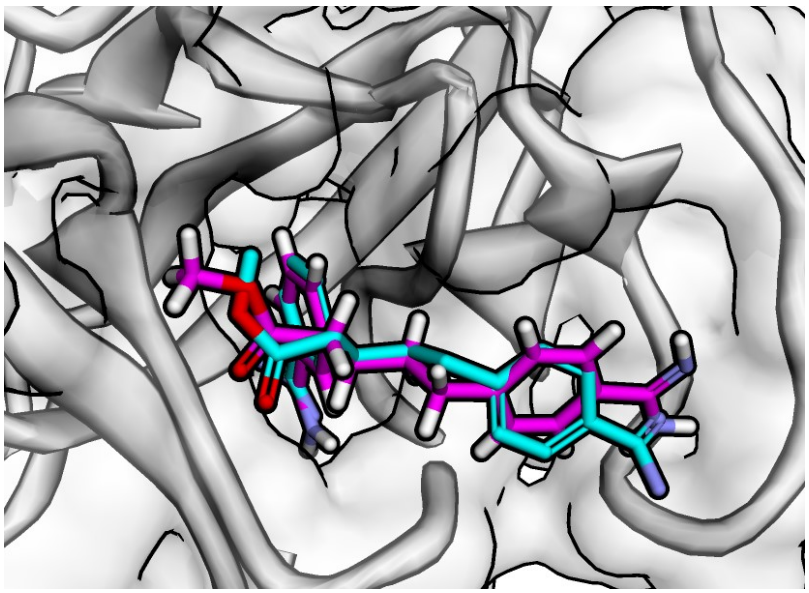
- Ubuntu 18.4 (x86 64) was used to compile the binary files for fpocket, prepare protein, and prepare ligand. As a result, they can function differently in other operating systems and architectures.

Examples

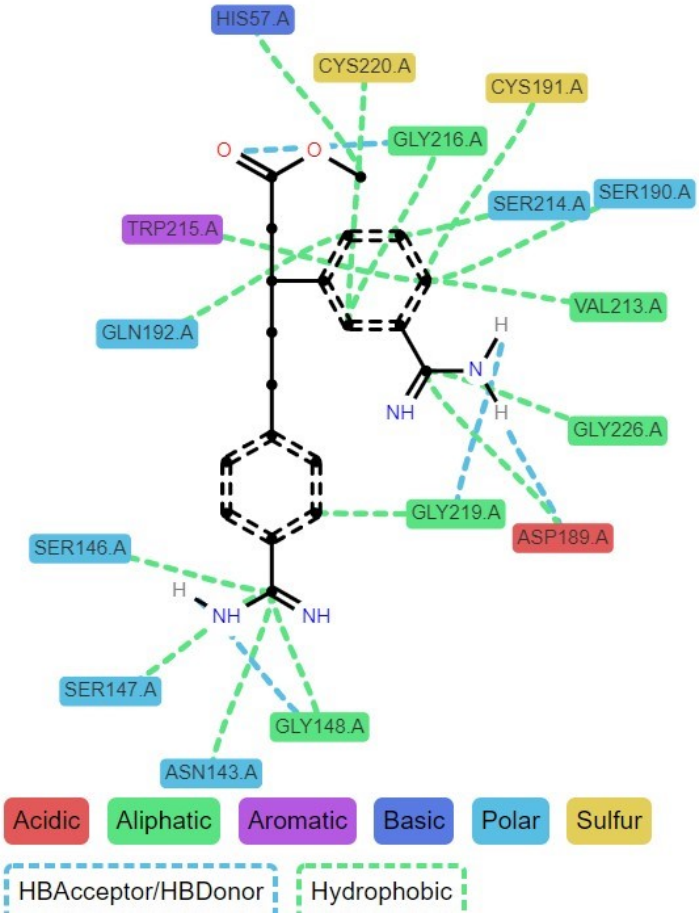
Docking powered by AutoDock Tools and LeDock



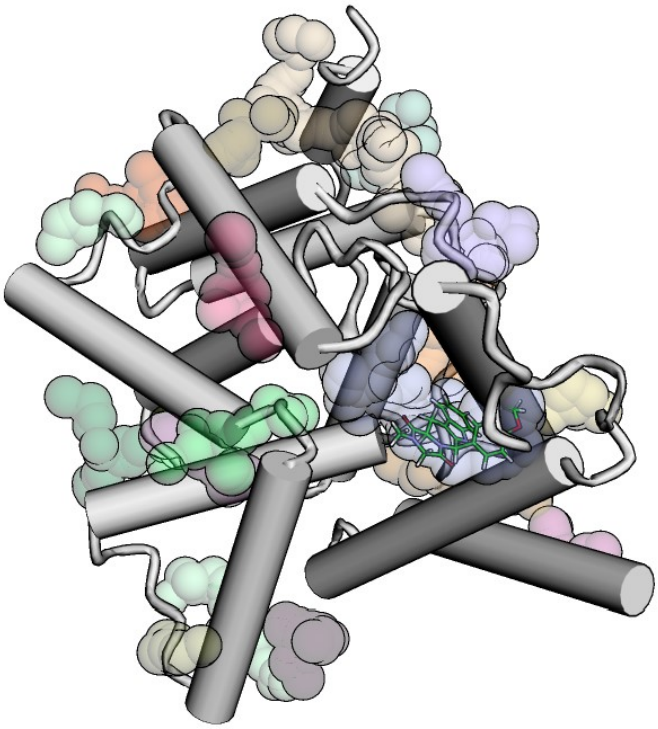
3D visualization of Docking results powered by py3Dmol



2D interaction maps powered by ProLif



Pocket search powered by Fpocket



Citation

If you use these notebooks, please credit this repository and the required tools as follows:

1. **Jupyter Dock**

> Ruiz-Moreno A.J. Jupyter Dock: Molecular Docking integrated in Jupyter Notebooks. https://doi.org/10.5281/zenodo.5514956

2. **Autodock Vina**

> Eberhardt, J., Santos-Martins, D., Tillack, A.F., Forli, S. (2021). AutoDock Vina 1.2.0: New Docking Methods, Expanded Force Field, and Python Bindings. Journal of Chemical Information and Modeling.

> Trott, O., & Olson, A. J. (2010). AutoDock Vina: improving the speed and accuracy of docking with a new scoring function, efficient optimization, and multithreading. Journal of computational chemistry, 31(2), 455-461.

3. **LeDock**

> Wang Z, Sun H, Yao X, Li D, Xu L, Li Y, et al. Comprehensive evaluation of ten docking programs on a diverse set of protein–ligand complexes: the prediction accuracy of sampling power and scoring power. Phys Chem Chem Phys. 2016;18: 12964–12975. https://doi.org/10.1039/C6CP01555G.

4. **AutoDock Tools**

> Morris, G. M., Huey, R., Lindstrom, W., Sanner, M. F., Belew, R. K., Goodsell, D. S., & Olson, A. J. (2009). AutoDock4 and AutoDockTools4: Automated docking with selective receptor flexibility. Journal of computational chemistry, 30(16), 2785–2791. https://doi.org/10.1002/jcc.21256.

5. **Pymol API**

> The PyMOL Molecular Graphics System, Version 1.2r3pre, Schrödinger, LLC.

6. **OpenBabel**

> O'Boyle, N.M., Banck, M., James, C.A. et al. Open Babel: An open chemical toolbox. J Cheminform 3, 33 (2011). https://doi.org/10.1186/1758-2946-3-33.

7. **RDKit**

> RDKit: Open-source cheminformatics; http://www.rdkit.org

8. **py3Dmol**

> Keshavan Seshadri, Peng Liu, and David Ryan Koes. Journal of Chemical Education 2020 97 (10), 3872-3876. https://doi.org/10.1021/acs.jchemed.0c00579.

9. **PDBFixer**

> P. Eastman, M. S. Friedrichs, J. D. Chodera, R. J. Radmer, C. M. Bruns, J. P. Ku, K. A. Beauchamp, T. J. Lane, L.-P. Wang, D. Shukla, T. Tye, M. Houston, T. Stich, C. Klein, M. R. Shirts, and V. S. Pande. 2013. OpenMM 4: A Reusable, Extensible, Hardware Independent Library for High Performance Molecular Simulation. Journal of Chemical Theory and Computation. ACS Publications. 9(1): 461-469.

10. **MDAnalysis**

> R. J. Gowers, M. Linke, J. Barnoud, T. J. E. Reddy, M. N. Melo, S. L. Seyler, D. L. Dotson, J. Domanski, S. Buchoux, I. M. Kenney, and O. Beckstein. MDAnalysis: A Python package for the rapid analysis of molecular dynamics simulations. In S. Benthall and S. Rostrup, editors, Proceedings of the 15th Python in Science Conference, pages 98-105, Austin, TX, 2016. SciPy, doi:10.25080/majora-629e541a-00e.

> N. Michaud-Agrawal, E. J. Denning, T. B. Woolf, and O. Beckstein. MDAnalysis: A Toolkit for the Analysis of Molecular Dynamics Simulations. J. Comput. Chem. 32 (2011), 2319-2327, doi:10.1002/jcc.21787. PMCID:PMC3144279.

11. **ProLif**

> chemosim-lab/ProLIF: v0.3.3 - 2021-06-11.https://doi.org/10.5281/zenodo.4386984.

12. **Fpocket**

> Le Guilloux, V., Schmidtke, P. & Tuffery, P. Fpocket: An open source platform for ligand pocket detection. BMC Bioinformatics 10, 168 (2009). https://doi.org/10.1186/1471-2105-10-168.

13. **Smina**

> Koes, D. R., Baumgartner, M. P., & Camacho, C. J. (2013). Lessons learned in empirical scoring with smina from the CSAR 2011 benchmarking exercise. Journal of chemical information and modeling, 53(8), 1893–1904. https://doi.org/10.1021/ci300604z

License