J Comput Aided Mol Des (2017) 31:213–218
DOI 10.1007/s10822-016-0006-1

DockingApp: a user friendly interface for facilitated docking simulations with AutoDock Vina

Elena Di Muzio[1] · Daniele Toti[1] · Fabio Polticelli[1]

Received: 4 November 2016 / Accepted: 22 December 2016 / Published online: 6 January 2017

Abstract Molecular docking is a powerful technique that helps uncover the structural and energetic bases of the interaction between macromolecules and substrates, endogenous and exogenous ligands, and inhibitors. Moreover, this technique plays a pivotal role in accelerating the screening of large libraries of compounds for drug development purposes. The need to promote community-driven drug development efforts, especially as far as neglected diseases are concerned, calls for user-friendly tools to allow non-expert users to exploit the full potential of molecular docking. Along this path, here is described the implementation of DockingApp, a freely available, extremely user-friendly, platform-independent application for performing docking simulations and virtual screening tasks using AutoDock Vina. DockingApp sports an intuitive graphical user interface which greatly facilitates both the input phase and the analysis of the results, which can be visualized in graphical form using the embedded JMol applet. The application comes with the DrugBank set of more than 1400 ready-to-dock, FDA-approved drugs, to facilitate virtual screening and drug repurposing initiatives. Furthermore, other databases of compounds such as ZINC, available also in AutoDock format, can be readily and easily plugged in.

E. Di Muzio and D. Toti have contributed equally to this work.

✉ Fabio Polticelli
fabio.polticelli@uniroma3.it

Elena Di Muzio
elena.dimuzio@uniroma3.it

Daniele Toti
daniele.toti@computationalbiology.it

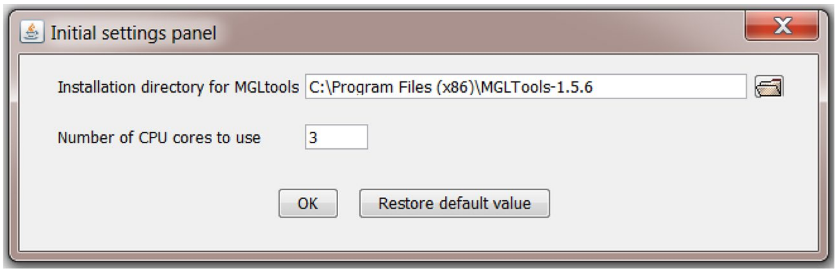
1 Department of Sciences, Roma Tre University, Rome, Italy

Keywords Molecular docking · Virtual screening · Drug repurposing · Graphic interface · Wrapper · AutoDock Vina

Introduction

Collective efforts, such as the Drugs for Neglected Diseases initiative [7], can dramatically speed up the development of novel drugs at a significantly lower cost. Furthermore, repurposing of Food and Drug Administration (FDA)-approved drugs allows to bypass the expensive and time-consuming toxicity assays and clinical trials, greatly reducing the time needed to bring a repurposed drug to the market. In this framework, docking simulations play a central role in the drug development pipeline. However, docking techniques are not readily accessible to researchers outside the structural bioinformatics field, and therefore their potential cannot be fully exploited in community-driven drug discovery initiatives. In the attempt of overcoming the technical difficulties of docking simulations, over the last few years some plug-ins were developed to facilitate them.

Two PyMOL [13] plug-ins exist for docking simulations using AutoDock Vina [14]. The one from the Lill research group is restricted to a Linux environment and requires additional software installation [6], while the one originally developed under Linux by [12] has been adapted for its use in a Windows environment, though apparently tested only on Windows XP. Other examples include AUDocker LE, an AutoDock Vina GUI available under Windows [11] and focused on large scale virtual screening tasks, and DOVIS 2.0, a parallel virtual screening tool for Linux clusters based on AutoDock 4.0 [5]. An AutoDock Vina interface for setting up docking simulations is also available as part of the UCSF Chimera molecular visualization program [10], although it appears more suited to expert than


Initial settings panel

Installation directory for MGLtools: C:\Program Files (x86)\MGLTools-1.5.6

Number of CPU cores to use: 3

Fig. 1 DockingApp's Initial settings panel

novice users. Recently, another PyMOL plug-in, the NRG-suite, has been described, which allows to perform docking simulations in real time using FlexAID [2]. Of note, the latest PyMOL executables are commercial products and the only officially-supported approach for building and installing PyMOL from the source code is under an open-source environment such as Linux.

Here we present DockingApp, a freely-accessible, platform-independent application for setting up, performing and analyzing the results of docking simulations using AutoDock Vina in a painless and extremely user-friendly way. The application comes with a pre-built library of more than 1400 ready-to-dock, FDA-approved drugs for virtual screening and drug repurposing initiatives. In addition, other, much larger databases of small molecule compounds can be easily plugged into DockingApp, such as the renowned ZINC database [4], available in pdbqt AutoDock format at the URL http://zinc.docking.org/pdbqt/.

Methodology

As briefly stated in the Introduction, DockingApp is born as a user-friendly software application meant to allow a variety of differently-skilled users to perform docking simulations, with high confidence on the results produced and minimal effort for setup and configuration. The former feature is guaranteed by relying upon the state-of-the-art docking program AutoDock Vina, which is the "engine" used by DockingApp to carry out the actual docking simulation; the latter feature is provided by a user-friendly graphical interface that on one hand hides all the complexity behind AutoDock Vina's usage, and on the other hand allows for a convenient browsing of the results both in tabular form and via a three-dimensional visualization of the receptor and the identified docking poses. All of this was made possible by the development of a platform-independent graphical user interface or "wrapper" (developed in Java), whose purpose is to both acquire the user's input and process the docking results, and by a Python program that is launched "behind the scenes" and is responsible of interacting with the included AutoDock Vina in the user's behalf. Further details of the application are described in the following subsections.

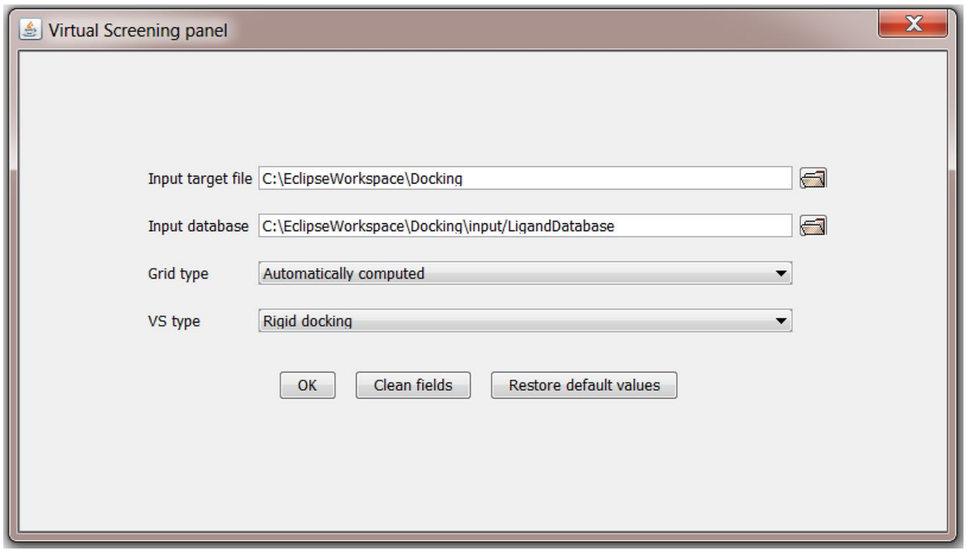
Initial configuration

DockingApp requires a minimal configuration effort, related to the selection of the number of CPU cores to be used for AutoDock Vina's execution and the specification of the installation directory of MGLTools [1, 8], which is a free Python library available for most platforms (Windows, Linux, OSX etc.) and is required for the software to run. The appropriate MGLTools distribution is already included for the respective operating system in DockingApp's packages. This setup can be done via the "Initial settings" panel (Fig. 1), which is automatically loaded at startup when the application is run for the first time, and can be recalled at a later date as the user needs. The default value for the number of CPU cores is set at half of the detected cores on the system; besides, DockingApp tries to automatically detect the location of MGLTools' installation directory on the system via a heuristic search, and if found, the corresponding field is populated with the identified directory.

Execution of docking and virtual screening jobs

DockingApp provides the user with the possibility of carrying out docking simulations on a given receptor, either against a single ligand (via the "Docking" panel) or a library of small molecules (via the "Virtual Screening" panel). In the former case, the user needs to specify the receptor and the ligand to be docked either as .pdb or .pdbqt files, whereas in the latter, instead of a single ligand, the user needs to select a folder containing the molecules the input receptor will be screened against in .pdbqt format (see Fig. 2). The automatic conversion of the input receptor and ligand from the .pdb to the .pdbqt file format, required by AutoDock Vina to run, is performed by DockingApp by using the `prepare_receptor4.py` and `prepare_ligand4.py` MGLTools scripts, respectively.

Regardless of the type of execution chosen, the user is given the chance to choose a "Grid type" and a "Docking/VS type" (Fig. 2). As a matter of fact, the search-space grid can be either automatically computed by the application


Virtual Screening panel

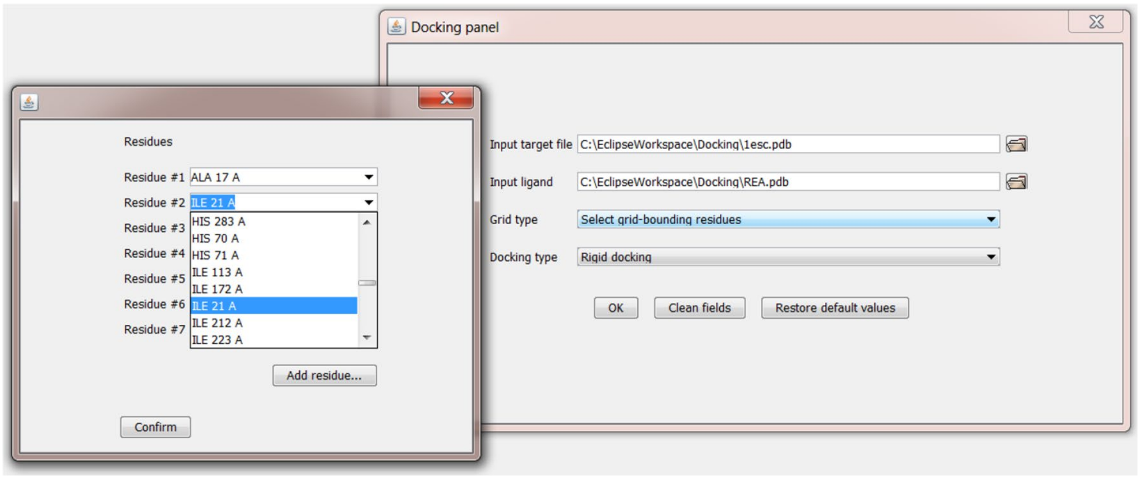
Input target file: C:\EclipseWorkspace\Docking

Input database: C:\EclipseWorkspace\Docking\input/LigandDatabase

Grid type: Automatically computed

VS type: Rigid docking

Fig. 2 Input panel for DockingApp's virtual screening execution


Docking panel

Input target file: C:\EclipseWorkspace\Docking\1esc.pdb

Input ligand: C:\EclipseWorkspace\Docking\REA.pdb

Grid type: Select grid-bounding residues

Docking type: Rigid docking

Residues

Residue #1: ALA 17 A

Residue #2: ILE 21 A

Residue #3

Residue #4

Residue #5

Residue #6

Residue #7

```
HIS 283 A
HIS 70 A
HIS 71 A
ILE 113 A
ILE 172 A
ILE 21 A
ILE 212 A
ILE 223 A
```

Fig. 3 Input panel for DockingApp's docking execution, where the user has chosen to manually specify the docking grid by selecting the appropriate grid-bounding residues, as shown in the corresponding *subpanel*. Here, the user can select any number of residues, either by browsing their list from the input receptor or by starting to type their name in the text fields and taking advantage of the autocomplete features provided for convenience. A similar *subpanel* is displayed when the user opts for a "flexible" docking, allowing the user to select the flexible residues

to encompass the whole receptor molecule, as in a "blind docking" run, or manually specified by the user via the selection of a set of grid-bounding residues. In the former case DockingApp automatically calculates the grid center as the geometric center of all the atoms of the receptor's structure. In the latter case the grid center is calculated in a similar manner using the set of atoms of the manually specified residues. Once the grid center is set, the system calculates the monodimensional distances between the center and each atom along the three x, y and z axes. The size of the grid box along the x, y and z axes is then computed by doubling the maximum value of the relative mono-dimensional distances along each dimension, adding to these values 5 Å to ensure that the grid box encloses all the atoms of either the receptor's structure or of the subset of residues chosen by the user.

Docking/virtual screening (VS) can be either "rigid" or "flexible": in the latter case, the user needs to choose the flexible residues of the input receptor. Specific panels are provided for the selection of the grid-bounding residues and for the flexible residues, featuring comboboxes where residues can be either selected from the provided drop-down lists or directly typed in the auto-completable text fields, as depicted in Fig. 3. For a default execution, the grid is automatically computed and the docking/VS is rigid, and thus the user needs only to select the input receptor and the input ligand (in the case of one-to-one docking) or the input database (for virtual screening; one is provided by DockingApp as described below), minimizing the complexity of the simulation. In the case of a flexible docking, the `prepare_flexreceptor4.py` MGLTools script is used by DockingApp, in a transparent fashion with respect to the user, to prepare the additional input file required by AutoDock Vina for flexible docking simulations.

Once all the selections are made and confirmed, DockingApp starts the corresponding execution by launching the included AutoDock Vina, which is responsible of performing the selected docking/virtual screening operations. Specifically, this is done by using a Python script, which interacts and exploits MGLTools to produce the configuration files needed by AutoDock Vina and then launches the actual call to the latter, in a completely transparent fashion with respect to the user. Obviously, virtual screening executions may take a long time to complete based on the processing power of the machine used and the processor cores available, especially when screening against a large set of molecules; it must be noted that DockingApp must be kept open during the process. At the present time, a dataset of 1466 FDA-approved small molecules is provided bundled with the application. Such dataset was downloaded from DrugBank [15] and was originally composed of 1584 small molecule drugs. Since each compound was represented in two dimensions, i.e. its atomic coordinates were given in a bidimensional reference system instead of a three-dimensional one, a procedure to convert the bidimensional coordinates to three-dimensional ones was carried out. For this purpose the MolConverter[1] utility was used. Afterwards, the resulting PDB files were in turn converted to the .pdbqt format via the aforementioned `prepare_ligand4.py` MGLTools script. It must be noted that, of the original 1584 molecules, 118 could not be converted due to "unknown atom type" errors encountered during the process (*e.g.* platinum, arsenic, etc.). The resulting set is provided within the "input/LigandDatabase" folder of the application to be used for virtual screening analyses. As already mentioned, other databases of small molecules in .pdbqt format can be easily plugged into DockingApp by just placing them in a corresponding folder within the input directory.

Visualization of results

Results produced by the docking/virtual screening process are made available by DockingApp via an interface showing several elements.

Firstly, a table is displayed listing the identified poses with their corresponding affinity values[2], RMSD upper and lower bounds, the corresponding ligand and file name, as well as an additional value dubbed SILE (Size-Independent Ligand Efficiency). The latter has been introduced by [9] in order to provide a measure of the docking energy unbiased by the size of the respective molecule, useful to evaluate the potential effectiveness of a compound as a drug and to guide the drug-design optimization process. The SILE is defined as follows:

$$SILE = \frac{affinity}{N^{0.3}}$$

where N denotes the number of heavy atoms in the considered molecule.

Secondly, a panel for filtering the results by different criteria is provided beside the results table, as detailed in Fig. 4. Finally, a three-dimensional view of the input receptor is shown, with the identified docking poses highlighted in different colors, whose display can be turned on or off by the corresponding checkboxes in the table records.

Results can be saved by the user as .dck files (DockingApp's file format) to be stored and re-opened at a later date at the user's convenience. A screenshot of the results displayed after a virtual screening execution is shown in Fig. 4.

Technology, requirements and availability

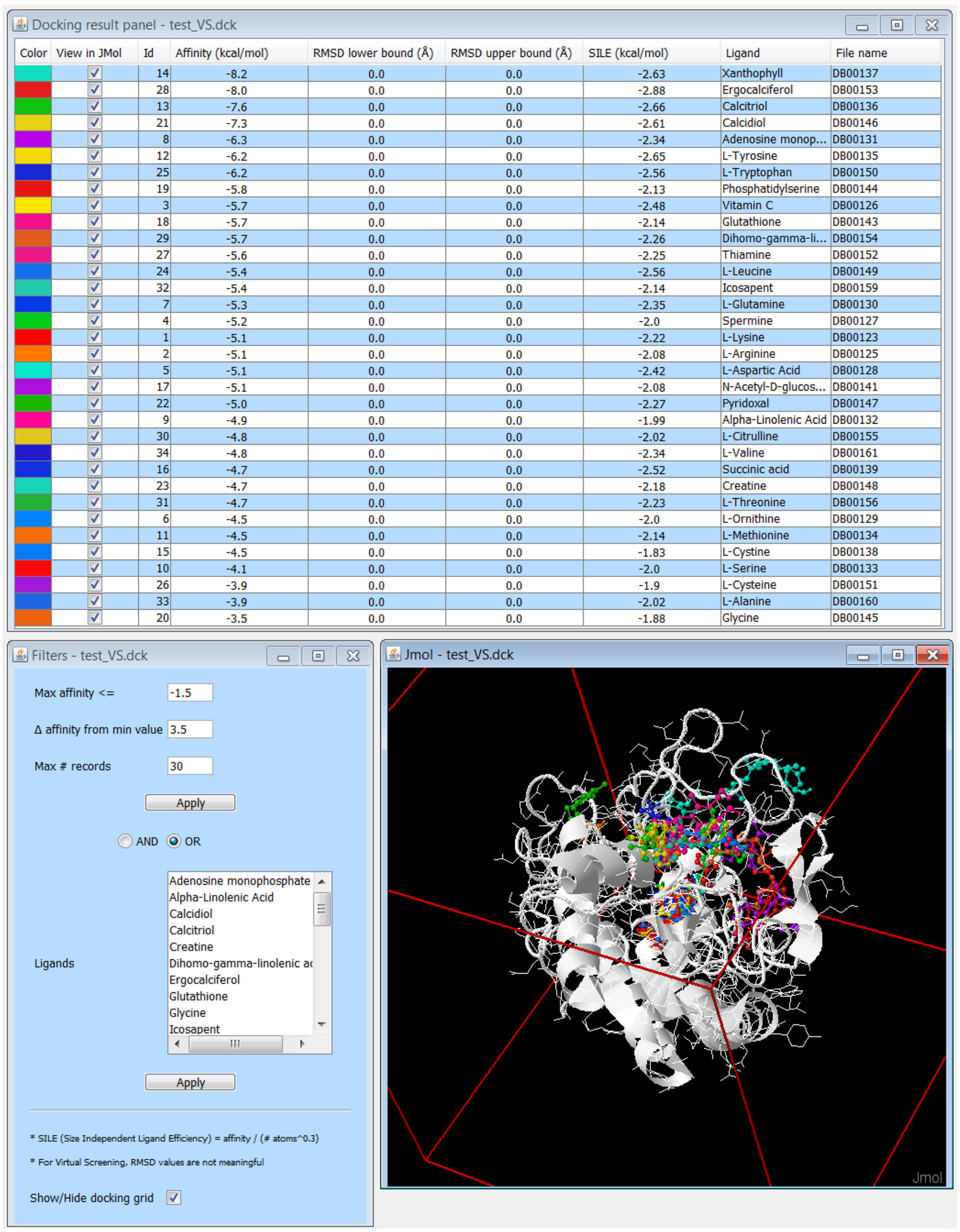
DockingApp is a 64-bit stand-alone software application developed in Java SE 7 and Python, with a Java Swing Graphic User Interface (GUI). It embeds AutoDock Vina and the JMol visualization program [3], and can be run locally on any major 64-bit OS equipped with a suitable Java Virtual Machine (JVM). Packages for Linux, Mac OSX and Windows are provided, each including the required MGLTools distribution (ver. 1.5.6) for the corresponding operating system. DockingApp is available for download at the following URL: http://www.computationalbiology.it/software.html. DockingApp is provided via a GPL license and its source code is available upon request to the authors.

Conclusion

In this work the implementation and features of DockingApp has been described, which is a Java-based

[1] https://www.chemaxon.com/products/marvin/molconverter/.

[2] It must be noted that affinity is the term used in the Autodock Vina output to indicate the predicted binding energy (lower values indicating tighter binding).



Docking result panel - test_VS.dck

Color	View in JMol	Id	Affinity (kcal/mol)	RMSD lower bound (Å)	RMSD upper bound (Å)	SILE (kcal/mol)	Ligand	File name
🟦	✓	14	-8.2	0.0	0.0	-2.63	Xanthophyll	DB00137
🟥	✓	28	-8.0	0.0	0.0	-2.88	Ergocalciferol	DB00153
🟩	✓	13	-7.6	0.0	0.0	-2.66	Calcitriol	DB00136
🟨	✓	21	-7.3	0.0	0.0	-2.61	Calcidiol	DB00146
🟪	✓	8	-6.3	0.0	0.0	-2.34	Adenosine monop...	DB00131
🟨	✓	12	-6.2	0.0	0.0	-2.65	L-Tyrosine	DB00135
🟦	✓	25	-6.2	0.0	0.0	-2.56	L-Tryptophan	DB00150
🟥	✓	19	-5.8	0.0	0.0	-2.13	Phosphatidylserine	DB00144
🟨	✓	3	-5.7	0.0	0.0	-2.48	Vitamin C	DB00126
🟥	✓	18	-5.7	0.0	0.0	-2.14	Glutathione	DB00143
🟧	✓	29	-5.7	0.0	0.0	-2.26	Dihomo-gamma-li...	DB00154
🟥	✓	27	-5.6	0.0	0.0	-2.25	Thiamine	DB00152
🟦	✓	24	-5.4	0.0	0.0	-2.56	L-Leucine	DB00149
🟦	✓	32	-5.4	0.0	0.0	-2.14	Icosapent	DB00159
🟦	✓	7	-5.3	0.0	0.0	-2.35	L-Glutamine	DB00130
🟩	✓	4	-5.2	0.0	0.0	-2.0	Spermine	DB00127
🟥	✓	1	-5.1	0.0	0.0	-2.22	L-Lysine	DB00123
🟧	✓	2	-5.1	0.0	0.0	-2.08	L-Arginine	DB00125
🟦	✓	5	-5.1	0.0	0.0	-2.42	L-Aspartic Acid	DB00128
🟪	✓	17	-5.1	0.0	0.0	-2.08	N-Acetyl-D-glucos...	DB00141
🟩	✓	22	-5.0	0.0	0.0	-2.27	Pyridoxal	DB00147
🟪	✓	9	-4.9	0.0	0.0	-1.99	Alpha-Linolenic Acid	DB00132
🟨	✓	30	-4.8	0.0	0.0	-2.02	L-Citrulline	DB00155
🟦	✓	34	-4.8	0.0	0.0	-2.34	L-Valine	DB00161
🟦	✓	16	-4.7	0.0	0.0	-2.52	Succinic acid	DB00139
🟦	✓	23	-4.7	0.0	0.0	-2.18	Creatine	DB00148
🟩	✓	31	-4.7	0.0	0.0	-2.23	L-Threonine	DB00156
🟦	✓	6	-4.5	0.0	0.0	-2.0	L-Ornithine	DB00129
🟧	✓	11	-4.5	0.0	0.0	-2.14	L-Methionine	DB00134
🟦	✓	15	-4.5	0.0	0.0	-1.83	L-Cystine	DB00138
🟥	✓	10	-4.1	0.0	0.0	-2.0	L-Serine	DB00133
🟪	✓	26	-3.9	0.0	0.0	-1.9	L-Cysteine	DB00151
🟦	✓	33	-3.9	0.0	0.0	-2.02	L-Alanine	DB00160
🟧	✓	20	-3.5	0.0	0.0	-1.88	Glycine	DB00145

Filters - test_VS.dck

Max affinity <=	-1.5
Δ affinity from min value	3.5
Max # records	30

Apply

AND [✓ OR]

Ligands: Adenosine monophosphate, Alpha-Linolenic Acid, Calcidiol, Calcitriol, Creatine, Dihomo-gamma-linolenic ac, Ergocalciferol, Glutathione, Glycine, Icosapent

Apply

* SILE (Size Independent Ligand Efficiency) = affinity / (# atoms^0.3)

* For Virtual Screening, RMSD values are not meaningful

Show/Hide docking grid ✓

Jmol - test_VS.dck

Fig. 4 Screenshot of DockinApp's output window for a virtual screening execution. Users can turn on or off the differently-colored poses in the three-dimensional visualization by simply checking/unchecking their corresponding checkboxes. The table of the results can be reordered by different sorting parameters by clicking on their corresponding table header. Besides, the "Filters" *panel* enables the user to dynamically filter those results below a certain affinity threshold, or those with a given affinity difference with respect to the best pose, or simply those that exceed a certain maximum number of records, and/or those results involving specific ligands. Furthermore, three-dimensional visualization is provided via a full-fledged JMol window retaining all JMol's functionalities, including the input console for specifying further commands in the JMol syntax. The docking grid used in the process, displayed as a red cubic wireframe, can be toggled on and off via its corresponding checkbox. Results can be also saved in DockingApp's file format and re-opened when needed

AutoDock Vina "wrapper" that can be used to speed up and facilitate docking simulations and the analysis of their results in an intuitive, painless and user-friendly fashion. With DockingApp, even "naive" users, non expert in structural bioinformatics, can perform docking simulations and virtual screening tasks, thus increasing the number of researchers working in the biomedical field who can take advantage of these techniques for the development of new drugs. Finally, the extremely user-friendly character of DockingApp makes it also an excellent tool to be used in educational settings to teach the basics of molecular docking and attract students towards the structural bioinformatics field.

References

1. Dallakyan S (2010) MGLTools. http://mgltools.scripps.edu/
2. Gaudreault F, Morency LP, Najmanovich RJ (2015) NRGsuite: a PyMOL plugin to perform docking simulations in real time using FlexAID. Bioinformatics 31(23):3856–8
3. Hanson RM (2010) Jmol-a paradigm shift in crystallographic visualization. J Appl Crystallogr 43:1250–1260
4. Irwin JJ, Sterling T, Mysinger MM, Bolstad ES, Coleman RG (2012) ZINC: a free tool to discover chemistry for biology. J Chem Inf Model 52(7):1757–68
5. Jiang X, Kumar K, Hu X, Wallqvist A, Reifman J (2008) DOVIS 2.0: an efficient and easy to use parallel virtual screening tool based on AutoDock 4.0. Chem Central J 2:18
6. Lill MA, Danielson ML (2011) Computer-aided drug design platform using PyMOL. J Comput Aided Mol Des 25:1319
7. Maxmen A (2016) Busting the billion-dollar myth: how to slash the cost of drug development. Nature 536:388–390
8. Morris GM, Huey R, Lindstrom W, Sanner MF, Belew RK, Goodsell DS, Olson AJ (2009) Autodock4 and AutoDockTools4: automated docking with selective receptor flexiblity. J Comput Chem 16:2785–2791
9. Nissink JW (2009) Simple size-independent measure of ligand efficiency. J Chem Inf Model 49(6):1617–1622
10. Pettersen EF, Goddard TD, Huang CC, Couch GS, Greenblatt DM, Meng EC, Ferrin TE (2004) UCSF Chimera-a visualization system for exploratory research and analysis. J Comput Chem 25(13):1605–1612
11. Sandeep G, Nagasree KP, Hanisha M, Kumar MMK (2011) AUDocker LE: a GUI for virtual screening with AUTODOCK Vina. BMC Res Notes 4:445
12. Seeliger D, de Groot BL (2010) Ligand docking and binding site analysis with PyMOL and Autodock/Vina. J Comput Aided Mol Des 24:417422
13. The PyMOL Molecular Graphics System, Version 1.8. 2016. Schrdinger, LLC
14. Trott O, Olson AJ (2010) AutoDock Vina: improving the speed and accuracy of docking with a new scoring function, efficient optimization and multithreading. J Comput Chem 31:455–461
15. Wishart DS, Knox C, Guo AC, Shrivastava S, Hassanali M, Stothard P, Chang Z, Woolsey J (2006) DrugBank: a comprehensive resource for in silico drug discovery and exploration. Nucleic Acids Res 34(Database issue):668–672, 16381955