FPocketWeb: protein pocket hunting in a web browser

Yuri Kochnev and Jacob D. Durrant

Abstract
Detecting macromolecular (e.g., protein) cavities where small molecules bind is an early step in computer-aided drug discovery. Multiple pocket-detection algorithms have been developed over the past several decades. Among them, fpocket, created by Schmidtke and Le Guilloux, is particularly popular. Like many programs used in computational-biology research, fpocket requires users to download and install an executable file. That file must also be run via a command-line interface, further complicating use. An existing fpocket server application effectively addresses these challenges, but it requires users to upload their possibly proprietary structures to a third-party server. The FPocketWeb web app builds on this prior work. It runs the fpocket3 executable entirely in a web browser without requiring installation. The pocket-finding calculations occur on the user’s computer rather than on a remote server. A working version of the open-source FPocketWeb app can be accessed free of charge from http://durrantlab.com/fpocketweb.

Keywords: Computer-aided drug discovery, Ligand-binding pocket, Fpocket, Browser app, Online resource, WebAssembly

Introduction
Proteins perform various cellular functions, ranging from mediating cell-signaling pathways to catalyzing vital chemical reactions. Many of these functions depend on interactions between protein receptors and endogenous small-molecule ligands that bind in cavities (pockets) typically located on protein surfaces. Carefully designed exogenous molecules can also bind in protein pockets, competing directly or indirectly with endogenous ligands and thus altering protein activity. Such molecules can serve as useful scientific tools (chemical probes) for revealing the basic biology of proper protein function. When properly optimized, they can also become pharmaceuticals that modulate the activity of disease-implicated proteins.

Structure-based computer-aided drug discovery (CADD) leverages known protein structures to identify and design novel exogenous ligands. Central to these efforts is identifying the protein pocket where the ligand will bind. In many cases, the location of the ligand-binding pocket is well-characterized. For example, crystallographic or NMR structures may include a pocket-bound ligand, and biochemical experiments can identify pocket-lining catalytic residues that interact with endogenous small-molecule substrates. But often, the location of the ligand-binding pocket is uncharacterized.

In such cases, cavity detection programs can serve as valuable tools for pocket identification. These programs include (a) “geometric” approaches that search for cavities based on the positions of protein-receptor atoms, and (b) “energy-based” approaches that further consider the physicochemical properties of protein residues to identify sites that might energetically favor ligand binding [1]. Among the few programs that fall into the energy-based category, the open-source tool fpocket is particularly popular. fpocket is written in C, is relatively easy to use, and is notably fast.

As with many CADD programs, the original fpocket implementation requires users to (1) download an
executable program, (2) choose proper configuration parameters, (3) run the program from a Unix- or DOS-like command-line interface, and (4) separately analyze and visualize the program output. The *fpocket* authors have taken several measures to address these challenges. To facilitate output analysis and visualization, *fpocket* saves output files in formats compatible with VMD [2] and PyMOL [3], two popular visualization programs. And to enable use without the command line, the authors created a helpful web server so users can run pocket-hunting calculations in the cloud [4].

These efforts have substantially improved usability, but some challenges remain. VMD- and PyMOL-formatted output files enable more accessible analysis than text-based output. Still, users must separately download and install these third-party visualization programs, and each has a substantial learning curve. Similarly, the *fpocket* server app eliminates command-line use, but it requires users to upload their (possibly proprietary) structures to a third-party system. Furthermore, in principle, hosting such an app requires an extensive and difficult-to-maintain backend computing infrastructure. Long wait queues may be necessary if a remote resource becomes saturated with requests.

To build on these prior efforts to enhance *fpocket* usability, we created FPocketWeb, a WebAssembly (Wasm) enabled web app that runs *fpocket* entirely in users’ browsers, without the need for other plugins or programs. Wasm-compiled code runs locally on the user’s computer, so it does not depend on remote infrastructure to run complex calculations. Instead, a simple web server—or even a “thin server”—is sufficiently powerful to send the Wasm-compiled binaries to the user’s browser.

We release FPocketWeb under the terms of the permissive open-source Apache License, Version 2.0, which is compatible with *fpocket3’s* MIT license. The FPocketWeb source code [5] and browser app [6] are publicly available online without requiring registration.

**Implementation**

**Wasm compilation**

We downloaded the *fpocket3* codebase, written in the C programming language, in 2022 [7]. We then used Emscripten 3.1.4 to compile it to a Wasm module. The *fpocket* source code required only minor adjustments to be Wasm-compilable. To create the output directories and subdirectories, the original *fpocket* uses a system call to execute the `mkdir` command. This command is readily available on operating systems such as Linux, but it is not available in the browser environment. We therefore commented out the offending line before Wasm compilation. We separately create the required directories on the virtual file system using JavaScript instead.

We also did not implement the full functionality of the VMD molfile plugin [8]. Specifically, our compilation excludes references to the netCDF library, which allows command-line *fpocket3* to load netCDF files. FPocketWeb therefore supports only PDB files.

**FPocketWeb browser app**

The FPocketWeb graphical user interface (GUI) allows users to quickly load protein structures into the browser’s memory, specify FPocketWeb parameters, run calculations using the Wasm-compiled *fpocket* module, and visualize/download the results. To create the interface, we followed the same approach we have used previously [9–12]. The FPocketWeb interface is written in the Microsoft TypeScript programming language, which compiles to JavaScript. The open-source Vue.js framework [13] provides consistently styled components (e.g., buttons, text fields, etc.). Many of these components are derived from BootstrapVue [14], an open-source library that implements the Bootstrap4 framework [15]. We also adapted our existing molecular-visualization Vue.js component [9, 11, 12], which leverages the 3Dmol.js library [16]. Finally, we used Webpack [17] and Google Closure Compiler [18] to compile, assemble, and optimize the code for size and speed.

**Results and discussion**

**FPocketWeb use**

**Input parameters tab**

To run FPocketWeb, users need only visit the FPocketWeb website [6]. The “Input Parameters” tab then appears, which includes the “Input File” and “Advanced Parameters” subsections (Fig. 1A, B). In the “Input File” subsection (Fig. 1A), users can specify the protein file (PDB format) for pocket hunting. This file is opened locally, but it is never uploaded to any third-party server. Users can also click the “Use Example File” button to load *H. sapiens* heat shock protein 90 (Hsp90, PDB 5J2V [19]) for testing. Loaded files are displayed in the “PDB Preview” subsection (not shown) using the 3Dmol.js molecular viewer [16].

The “Advanced Parameters” subsection (Fig. 1B) allows users to specify the same parameters available through the *fpocket* command-line executable so they can fine-tune the underlying pocket-finding method. Those interested in further details should consult the original *fpocket* manuscripts [1, 4]. FPocketWeb initially hides these parameters...
because most users will prefer to use the default values. Once ready, users can click the “Start FPocketWeb” button (Fig. 1C) to initiate the FPocketWeb run.

**Output tab**

FPocketWeb displays the “Output” tab once the calculations finish (Fig. 1). This tab includes the “Visualization” and “Detected Pockets” subsections. The “Visualization” subsection shows the protein and detected cavities (Fig. 1C). Initially, only the pocket with the highest \(fpocket\) score is shown. When running FPocketWeb on a device with a sufficiently wide screen, a list of the detected pockets appears on the right (not shown), allowing the user to easily toggle individual pocket visualizations on and off. The “Detected Pockets” subsection (Fig. 1D) contains a table with the detailed output for each detected pocket, one pocket per row. Initially, only the score, the druggability score, the number of alpha spheres, and the volume are displayed, but the “Show Details” toggle allows users to display all other \(fpocket\) metrics for each detected pocket. Users can also change the color used to visualize each pocket.

The “Output Files” subsection (not shown) allows users to view the FPocketWeb output PDB file, which includes the original protein structure and all the detected pockets. Users can press the “Download” button to save the file. Finally, the “Run FPocket from the Command Line” subsection (also not shown) provides a code snippet so users can run \(fpocket\) from the command line with the same FPocketWeb parameters used in the browser.

**Benchmarks and compatibility**

We performed benchmark calculations on a Linux laptop (HP Pavilion 15.6”) running Fedora 34 (1.3 GHz Intel Core i7 processor and 16 GB 2400 MHz DDR4 memory) to compare FPocketWeb and command-line \(fpocket\) (Table 1). The two programs had comparable execution times and produced identical scores, druggability scores, numbers of alpha spheres, and volumes. After these initial benchmarks, we further tested FPocketWeb on the browser/operating-system combinations shown in Table 2. The app is compatible with desktop and mobile operating systems, as well as all major browsers (e.g., Chrome, Edge, Safari, and Firefox).

**Example of use: TEM-1 \(\beta\)-lactamase**

As a demonstration of use, we first considered TEM-1, a bacterial \(\beta\)-lactamase. TEM-1 contributes to penicillin and cephalosporin resistance in Gram-negative bacteria [24] by hydrolyzing the four-carbon ring common to all \(\beta\)-lactam antibiotics. Aside from the orthosteric pocket where the ring is hydrolyzed, a recently discovered allosteric cryptic pocket [25–28] is collapsed and hidden (PDB ID: 1FQG [29]) unless co-crystallized with
a bound ligand (PDB ID: 1PZP [28]). TEM-1 is often used to benchmark computational tools for cryptic-pocket identification [25, 30, 31].

We applied FPocketWeb to TEM-1 crystal structures with (Z)-3-[(4-phenylazanylphenyl)amino]-2-(2 H-1,2,3,4-tetrazol-5-yl)prop-2-enenitrile (FTA, PDB ID: 1PZP [28]) and penicillin G (open form; PDB ID: 1FQG [29]) bound in the allosteric and orthosteric pockets, respectively. We first used FPocketWeb to remove all non-protein atoms before detecting pockets on the protein surfaces.

This analysis shows the dramatic impact that the allosteric inhibitor FTA has on the TEM-1 structure. The top-ranked pocket detected when we applied FPocketWeb to the FTA-bound structure (PDB ID: 1PZP [28]) encompassed the allosteric and orthosteric sites (Fig. 2A), with a total volume of 2161 Å³. In contrast, the allosteric pocket was not detected when we applied FPocketWeb to the penicillin-bound structure (PDB ID: 1FQG [29]) because the allosteric pocket was unoccupied and collapsed. Even the orthosteric pocket ranked only fifth (Fig. 2B, 350 Å³).

Example of use: influenza neuraminidase
As a second demonstration of use, we consider the enzymatic pocket of influenza neuraminidase (N1). N1 is an exosialidase that prevents the aggregation of newly formed viral particles on infected-cell surfaces [32]. The primary N1 enzymatic pocket [33, 34] is persistent even

Table 1  FPocketWeb/fpocket benchmarks

| PDB   | Protein               | FPocketWeb Time (s) | fpocket Time (s) | Score |
|-------|-----------------------|---------------------|------------------|-------|
| 1HWL  | HMG-CoA reductase      | 5.6±0.095           | 5.1±0.149        | 0.437 |
| 2P16  | factor Xa             | 0.7±0.032           | 0.6±0.009        | 0.961 |
| 3LN1  | COX-2                 | 15.9±0.514          | 16.0±0.145       | 0.630 |
| 4LL3  | HIV protease          | 0.4±0.052           | 0.3±0.015        | 0.862 |
| 4TZ4  | Cereblon              | 5.2±0.048           | 5.3±0.075        | 0.301 |

Times are reported in seconds. FPocketWeb and fpocket produced the same scores; we report the scores associated with the top-ranked pockets. The calculations were performed on PDB files downloaded directly from the Protein Data Bank, without further processing.

Table 2  Browser compatibility

| Browser        | Operating system |
|----------------|------------------|
| Chrome 101.0.4951.41 | macOS 12.1       |
| Firefox 99.0.1  | macOS 12.1       |
| Safari 15.2     | macOS 12.1       |
| Chrome 101.0.4951.41 | Windows 10 Enterprise |
| Firefox 99.0.1  | Windows 10 Enterprise |
| Edge 100.0.1185.50 | Windows 10 Enterprise |
| Chrome 100.0.4896.127 | Android 12     |
| Firefox 99.2.0  | Android 12       |
| Safari 15       | iOS 13.4.1       |
| Firefox 99.0    | Ubuntu 20.04.4 LTS |
| Chrome 101.0.4951.41 | Ubuntu 20.04.4 LTS |

We have tested FPocketWeb on all major browsers and operating systems.

![Fig. 2](image-url)  Screenshots of FPocketWeb applied to TEM-1. A TEM-1 when the ligand FTA is bound in the allosteric site (PDB ID: 1PZP). The top-ranked pocket, shown in surface representation, encompasses both the allosteric and orthosteric pockets, marked with a dagger and double dagger, respectively. B TEM-1 when penicillin G (open form) is bound in the orthosteric site (PDB ID: 1FQG). The fifth-ranked pocket, shown in surface representation, corresponds to the orthosteric pocket (double dagger). The allosteric pocket was not detected because it is collapsed.
in the absence of a bound ligand. But a flexible pocket-
adjacent “150 loop” [35–37] enables a pocket extension
(the 150-cavity) that is collapsed in many crystal struc-
tures. Others have designed N1 inhibitors that exploit
this cavity [38].

We applied FPocketWeb to N1 crystal structures in
open (PDB ID: 2HTY [37]) and closed (PDB ID: 2HU4
[37]) 150-cavity conformations. We again removed any
non-protein atoms before detecting pockets on the pro-
tein surfaces. The top-ranked pocket detected when we
applied FPocketWeb to the open-150-cavity structure
included the 150-cavity (Fig. 3A, marked with a dagger
and shown as a white surface, 911 Å³). The fifth-ranked
pocket corresponded to the known sialic-acid binding
site (Fig. 3A, marked with a double dagger and shown as a
red surface, 491 Å³). In contrast, when we applied FPock-
etWeb to the closed-150-cavity structure, the 150-cavity
was not detected. The top-ranked pocket did include the
known sialic-acid binding site (Fig. 3B, marked with a
double dagger, 1142 Å³).

Limitations
In compiling fpocket to Wasm, we also compiled the
related programs specified in the same makefile, includ-
ing mdpocket (a tool for analyzing pocket conformations
sampled by molecular dynamics simulations), dpocket (a
tool for extracting physicochemical pocket descriptors),
and tpocket (a tool for evaluating cavity scoring func-
tions). However, we subsequently focused our web-based
implementation and testing on fpocket alone. Implemen-
ting mdpocket as a browser app is also appealing, but it
would require loading an entire MD trajectory into the
browser’s memory, which is not presently practical. For
those interested in using an online version of mdpocket,
we recommend the original fpocket creators’ useful server
application, which allows users to upload their trajectory
files and run mdpocket in the cloud [4].

Another limitation is that FPocketWeb only accepts
files in the PDB format. The original fpocket3 executable
uses the molfile plugin [8] to load structure and trajectory
files in many formats. In some cases (e.g., the netCDF
format), the molfile plugin leverages other libraries with
many dependencies. Given that PDB is the de facto
standard for static protein structures and that FPock-
etWeb is not meant to work with trajectory files (i.e., it
do not implement mdpocket), we opted to focus on the
PDB format alone. Users with files in other formats can
easily convert them to the PDB format using programs
such as VMD [2] and Open Babel [39].
Conclusion
FPocketWeb is an easy-to-use pocket-finding tool that allows users to run the f pockets executable entirely in a web browser, without the need for explicit download and installation. The pocket-finding calculations occur on the user’s computer rather than on a remote server, so F PocketWeb does not require substantial remote infrastructure, nor does it require users to upload their possibly proprietary structures. And yet it retains the usability of a web-based interface, and the identified pockets are visualized in the browser itself, without requiring third-party software such as VMD. In our tests of five proteins, F PocketWeb and the original f pockets had comparable run times and produced identical pocket scores.

We are hopeful that F PocketWeb’s utility and usability will make it a broadly useful open-source tool for those interested in computational structural biology and computer-aided drug discovery. We release it under the Apache License, Version 2.0. The source code [5] and browser app [6] are available free of charge online, without requiring registration.

Availability and requirements
Project name: F PocketWeb.
Project home page: http://durrantlab.com/f pocketweb-download.
Operating system(s): Windows, macOS, Linux, Android, iOS.
Programming language: TypeScript, JavaScript, HTML, CSS, WebAssembly.
Other requirements: Any modern web browser.
License: Apache License, Version 2.0.
Any restrictions to use by non-academics: None.

Supplementary Information
The online version contains supplementary material available at https://doi. org/10.1186/s13321-022-00637-0.

Additional file 1. The F PocketWeb source code, version 1.0.1. See http://durrantlab.com/f pocketweb-download for the latest version.

Acknowledgements
We acknowledge the University of Pittsburgh’s Center for Research Computing for providing valuable computer resources.

Author contributions
YK: Compiled the f pocket3 codebase to WebAssembly, performed benchmark tests, helped create the F PocketWeb GUI, and contributed to the writing of the manuscript. JDD: Helped create the GUI, provided two examples of use, contributed to the writing of the manuscript, and provided research supervision. Both the authors read and approved the final manuscript.

Funding
This work was supported by the National Institute of General Medical Sciences of the National Institutes of Health [R01GM132553 to JDD]. The content is solely the responsibility of the authors and does not necessarily represent the official views of the National Institutes of Health.

Availability of data and materials
The F PocketWeb source code is available free of charge from http://durr antlab.com/f pocketweb-download, and the web app can be accessed at http://durrantlab.com/f pocketweb. Both URLs are publicly available, without requiring registration. A copy of the source code is also provided as Additional file 1.

Declarations
Competing interests
Not applicable.

Received: 10 June 2022 Accepted: 4 August 2022
Published online: 26 August 2022

References
1. Le Guilloux V, Schmidtke P, Tuffery P (2009) F pockets: an open source platform for ligand pocket detection. BMC Bioinformatics 10:168. https://doi. org/10.1186/1471-2105-10-168
2. Humphrey W, Dalke A, Schulten K (1996) VMD: visual molecular dynamics. J Mol Graph 14:33–38. https://doi.org/10.1016/0263-7855(96)00018-5
3. Schrödinger L (2015) The PyMOL Molecular Graphics System, Version, 1.8 edn.
4. Schmidtke P, Le Guilloux V, Maupetit J, Tuffery P (2010) F pockets: online tools for protein ensemble pocket detection and tracking. Nucleic Acids Res 38:W582-W589. https://doi.org/10.1093/nar/gkq383
5. Durrant J (2022) F PocketWeb-Durrant Lab. https://doi.org/https://durr antlab.pitt.edu/f pocketweb-download. Accessed 9 June 2022
6. Durrant J (2022) F PocketWeb. https://doi.org/http://durrantlab.com/ f pocketweb. Accessed 9 June 2022
7. Discngine/f pocket (2022) https://doi.org/https://github.com/Discn gine/f pocket. Accessed 2022
8. Molfile Plugin Documentation (2021) University of Illinois at Urbana-Champaign. https://doi.org/https://www.ks.uiuc.edu/Research/vmd/plugins/molfile/. Accessed 28 April 2022
9. Young J, Garikipati N, Durrant JD (2022) BINANA 2: characterizing receptor/ligand interactions in Python and JavaScript. J Chem Inf Model 62:753–760. https://doi.org/10.1021/acs.jcim.1c01461
10. Durrant JD (2022) Prot2Prot: a deep learning model for rapid. Photorealist-tic macromolecular visualization. bioRxiv https://doi.org/10.1101/ bioRxiv.62:753–760. https://doi.org/10.1021/acs.jcim.1c01461
11. Green H, Durrant JD (2021) DeepFrag: an open-source browser app for deep-learning lead optimization. J Chem Inf Model 61:2523–2529.
12. Kochnev Y, Hellemann E, Cassidy KC, Durrant JD (2020) Webina: an open-source library and web app that runs AutoDock Vina entirely in the web browser. Bioinformatics 36:4513–4515. https://doi.org/10.1093/bioinformatics/btu829
13. Vue.js—The Progressive JavaScript Framework. https://doi.org/https:// vuejs.org/. Accessed 9 June 2022
14. BootstrapVue: https://bootstrap-vue.org/. Accessed 9 June 2022
15. Bootstrap: The most popular HTML, CSS, and JS library in the world. https://https://bootstrap-vue.org/. Accessed 9 June 2022
16. Rego N, Koes D (2015) 3Dmol.js: molecular visualization with WebGL. Bioinformatics 31:1322–1324. https://doi.org/10.1093/bioinformatics/ btu829
17. webpack. https://webpack.js.org/. Accessed 9 June 2022
