SOFTWARE TOOL ARTICLE

FreeSASA: An open source C library for solvent accessible surface area calculations [version 1; referees: 2 approved]

Simon Mitternacht

University Library, University of Bergen, Bergen, Norway

v1 **First published:** 18 Feb 2016, **5**:189 (doi: 10.12688/f1000research.7931.1)
Latest published: 18 Feb 2016, **5**:189 (doi: 10.12688/f1000research.7931.1)

Abstract

Calculating solvent accessible surface areas (SASA) is a run-of-the-mill calculation in structural biology. Although there are many programs available for this calculation, there are no free-standing, open-source tools designed for easy tool-chain integration. FreeSASA is an open source C library for SASA calculations that provides both command-line and Python interfaces in addition to its C API. The library implements both Lee and Richards' and Shrake and Rupley's approximations, and is highly configurable to allow the user to control molecular parameters, accuracy and output granularity. It only depends on standard C libraries and should therefore be easy to compile and install on any platform. The library is well-documented, stable and efficient. The command-line interface can easily replace closed source legacy programs, with comparable or better accuracy and speed, and with some added functionality.

Open Peer Review

Referee Status: ✓ ✓

	Invited Referees 1	Invited Referees 2
version 1 published 18 Feb 2016	✓ report	✓ report

1 **Simon Hubbard**, University of Manchester UK

2 **Yaoqi Zhou**, Griffith University Australia, **Yuedong Yang**, Griffith University Australia

Discuss this article

Comments (0)

Corresponding author: Simon Mitternacht (simon.mitternacht@uib.no)

How to cite this article: Mitternacht S. **FreeSASA: An open source C library for solvent accessible surface area calculations [version 1; referees: 2 approved]** *F1000Research* 2016, **5**:189 (doi: 10.12688/f1000research.7931.1)

Grant information: The author(s) declared that no grants were involved in supporting this work.

Competing interests: No competing interests are disclosed.

First published: 18 Feb 2016, **5**:189 (doi: 10.12688/f1000research.7931.1)

Introduction

Exposing apolar molecules to water is highly unfavorable, and minimizing the hydrophobic free energy is an important driving force in the folding of macromolecules (Finkelstein & Ptitsyn, 2002). The solvent accessible surface area (SASA) of a molecule gives a measure of the contact area between molecule and solvent. Although the exact quantitative relation between surface area and free energy is elusive, the SASA can be used to compare different molecules or different conformations of the same molecule, or for example measure the surface that is buried due to oligomerization.

To define the SASA, let a spherical probe represent a solvent molecule. Roll the probe over the surface of a larger molecule. The surface area traced by the center of the probe is the SASA of the larger molecule (Lee & Richards, 1971). Two classical approximations are commonly used to calculate SASA: one by Lee and Richards (L&R) where the surface is approximated by the outline of a set of slices (1971), and one by Shrake & Rupley (1973) (S&R) where the surface of each sphere is approximated by a set of test points. The SASA can be calculated to arbitrary precision by refining the resolution of both. The surface area can also be calculated analytically (Fraczkiewicz & Braun, 1998), which is useful when the gradient is needed, or by various other approximations, tailored for different purposes (Cavallo *et al.*, 2003; Drechsel *et al.*, 2014; Sanner *et al.*, 1996; Weiser *et al.*, 1999; Xu & Zhang, 2009).

There are many tools available to calculate SASA. The most popular program for command line use is probably NACCESS (Hubbard & Thornton, 1993) (freely available for academic use), which is an efficient Fortran implementation of the L&R approximation. Another well-known command line tool is DSSP, which primarily calculates the secondary structure and hydrogen bonds of a protein structure, but provides the SASA as well (Touw *et al.*, 2015) (using S&R, open source). There are also some web services available, for example Getarea, which calculates the surface analytically (Fraczkiewicz & Braun, 1998), and Triforce which uses a semianalytical tessellation approach (Drechsel *et al.*, 2014) (also available for command line use). In addition, most molecular dynamics simulation packages include tools to analyze SASA from trajectories.

FreeSASA is intended to fill the same niche as NACCESS, and a number of other similar programs: a simple and fast command-line tool that "does one thing and does it well" and can be easily integrated into tool chains. The advantage of FreeSASA is that it is open source (GNU General Public License 3), and provides both C and Python APIs in addition to its command line interface. It has sensible default parameters and no obligatory configuration for casual users (the only required input is a structure), but also allows full control over all calculation parameters. Dependencies have been kept to a minimum: compilation only requires standard C and GNU libraries. The library is thread-safe, and some effort has gone into dealing gracefully with various errors. The code ships with thorough documentation, which is also available online at http://freesasa.github.io/doxygen/. Although functionality and availability have been the primary motivating factors for writing this library, performance tests show that FreeSASA is as fast as or faster than legacy programs when run on a single CPU core. In addition, a large portion of the calculation has been parallelized, which gives significant additional speed when run on multicore processors.

Methods

Implementation

Calculations. Both S&R and L&R are pretty straightforward to implement, and both require first determining which atoms are in contact, and then calculating the overlap between each atom and its neighbors. Finding contacts is done using cell lists, which means the contact calculation is an $O(N)$ operation. Both algorithms then treat each atom independently, making also the second part of the calculation $O(N)$. In addition, this second part is trivially parallelizable.

For L&R, instead of slicing the whole protein in one go, each atom is sliced individually. The L&R calculation is thus parameterized by the number of slices per atom, i.e. small atoms have thinner slices than large atoms.

The Fibonacci spiral gives a good approximation to an even distribution of points on the sphere (Swinbank & Purser, 2006), allowing efficient generation of an arbitrary number of S&R test points. The cell lists provide the first of the two lattices in the double cubic lattice optimization for this algorithm (Eisenhaber *et al.*, 1995), the second lattice (for the test points) is not implemented in FreeSASA, for now.

The correctness of the implementations was tested by first inspecting the surfaces visually. In the two atom case, results were verified against analytical calculations. Another verification came from comparing the results of high precision SASA calculations using the two independent algorithms. In addition, using the L&R algorithm gives identical results to NACCESS when the same resolution and atomic radii are used.

Radius assignment. An important step of the calculation is assigning a radius to each atom. The default in FreeSASA is to use the *ProtOr* radii by Tsai *et al.* (1999). The library recognizes the 20 standard amino acids (plus Sec and Pyl), and the standard nucleotides (plus a few nonstandard ones). Tsai *et al.* do not mention phosphorus and selenium; these atoms are assigned a radius of 1.8 and 1.9 Å respectively.

By default, hydrogen atoms and HETATM records are ignored in Protein Data Bank (PDB) files. If included, the library recognizes three common HETATM entries: the acetyl and NH_2 capping groups, and water, and assigns ProtOr radii to these. Otherwise the van der Waals radius of the element is used, taken from the paper by Mantina *et al.* (2009). For elements outside of the 44 main group elements treated by Mantina *et al.*, or if completely different radii are desired, users can provide their own configuration.

Users can specify their own atomic radii either through the API or by providing a configuration file. The library ships with a few sample configuration files, including one that provides a subset of the NACCESS parameterization, and one with the default ProtOr parameters. In addition, scripts are provided to automatically generate ProtOr configurations from PDB CONECT entries, such as those in the Chemical Component Dictionary (Westbrook *et al.*, 2015). These can then be appended to the default configuration.

Operation

Building the FreeSASA library and command-line interface only requires standard C and GNU libraries and a C99-compliant compiler, and should be straightforward on any UNIX system (has been tested in Mac OS X 10.8 and Debian 8), and not too difficult in Windows (not tested). Building the Python bindings requires Cython (tested with version 0.21). The library ships with an Autotools build configuration, but the source itself is simple enough to be possible to compile "manually", if necessary.

Command-line interface. Building FreeSASA creates the binary `freesasa`. The simplest program call, with default parameters, is

```
$ freesasa 3wbm.pdb
```

using the structure with PDB code 3wbm as an example (a protein-RNA complex). The above produces the following output

```
## freesasa 1.0.1 ##

PARAMETERS
algorithm    : Lee & Richards
probe-radius : 1.400
threads      : 2
slices       : 20

INPUT
source  : 3wbm.pdb
chains  : ABCDXY
atoms   : 3714

RESULTS (A^2)
Total   : 25190.77
Apolar  : 11552.38
Polar   : 13638.39
CHAIN A :  3785.49
CHAIN B :  4342.33
CHAIN C :  3961.12
CHAIN D :  4904.30
CHAIN X :  4156.46
CHAIN Y :  4041.08
```

The numbers in the results section are the SASA values (in Å^2) for the respective groups of atoms.

As an illustration of a few of the other configuration options, and how to use the program as a PDB file filter, the command

```
$ freesasa -n 100 --print-as-B-values
--no-log < 3wbm.pdb > 3wbm.sasa
```

calculates the SASA of a PDB-file passed via `stdin`, using 100 slices per atom. The flag `--no-log` suppresses the regular output. The output will instead, because of the flag `--print-as-B-values`, be the provided PDB-file with the SASA of each atom replacing the temperature factors, and the atomic radii stored in the occupancy factor field.

By calling with the option `--chain-groups`,

```
$ freesasa --chain-groups=ABCD+XY 3wbm.pdb
```

two calculations are appended to the original output, one where only the four chains A, B, C and D have been included, and one with only X and Y.

The option `--select` can be used to select a set of atoms using a subset of the selection syntax used in the program Pymol (DeLano, 2002). For example, the command

```
$ freesasa --select="RNA, resn A+U+G+C"
```

will produce the following output (after the regular output shown above)

```
SELECTIONS
RNA :    8197.53
```

where *RNA* is simply the user-defined name of the selection, and the number the contribution to the total SASA from the bases A, U, G and C (which we in this particular case could have got by simply adding the areas for the chains X and Y in the sample output above).

The command

```
$ freesasa -h
```

prints a help message listing all available options, including other ways to redirect output and how to change different calculation parameters (the most detailed information can be found online at http://freesasa.github.io/doxygen/CLI.html).

C API. The C code below illustrates how to perform a SASA-calculation on the same PDB-file as above, using the C API, with default parameters. The functions and types used are all defined in the header `freesasa.h`.

```
   FILE *fp = fopen("3wbm.pdb","r");
   freesasa_structure *structure =
freesasa_structure_from_pdb(fp, NULL, 0);
   freesasa_result *result =
freesasa_calc_structure(structure, NULL);
   printf("Total area : %f A2\n", result->total);
```

The two points where null pointers are passed as arguments are places where atom classifiers and calculation parameters could have been provided. A more elaborate example that includes error handling and freeing of allocated resources can be seen in Figure 1.

```c
#include <stdlib.h>
#include <stdio.h>
#include "freesasa.h"

int main(int argc, char **argv) {
    freesasa_structure *structure = NULL;
    freesasa_result *result = NULL;
    freesasa_strvp *class_area = NULL;

    /* Read structure from stdin */
    structure = freesasa_structure_from_pdb(stdin,NULL,0);

    /* Calculate SASA using structure */
    if (structure) {
        result = freesasa_calc_structure(structure,NULL);
    }

    /* Calculate area of classes (Polar/Apolar/..) */
    if (result) {
        class_area = freesasa_result_classify(result,structure,NULL);
    }

    /* Print results */
    if (class_area) {
        printf("Total area : %f A2\n",result->total);
        for (int i = 0; i < class_area->n; ++i)
            printf("%s : %f A2\n",class_area->string[i],
                   class_area->value[i]);
    } else {
        /* If there was an error at any step, we will end up here */
        printf("Error calculating SASA\n");
    }

    /* Free allocated resources */
    freesasa_strvp_free(class_area);
    freesasa_result_free(result);
    freesasa_structure_free(structure);
    return EXIT_SUCCESS;
}
```

Figure 1. C API. Illustration of how to use the C API including rudimentary error handling.

The API also allows the user to calculate the SASA of a set of coordinates with associated radii. The code below puts two atoms at positions $\vec{x}_1 = (1, 1, 1)$ and $\vec{x}_2 = (2, 2, 2)$ with radii $r_1 = 2$ and $r_2 = 3$, respectively, and outputs the SASA of the individual atoms.

```c
   double coord[] = {1.0, 1.0, 1.0, 2.0, 2.0, 2.0};
   double radius[] = {2.0, 3.0};
   freesasa_result *result =
freesasa_calc_coord(coord, radius, 2, NULL);
   printf("A1 = %f, A2 = %f\n", result->sasa[0],
          result->sasa[1]);
```

Python API. The library includes Python bindings that export most of the C API to Python. The Python code below gives the same output as the example in Figure 1. Error handling is excluded for brevity.

```python
import freesasa

structure = freesasa.Structure("3wbm.pdb")
result = freesasa.calc(structure)
classArea = freesasa.classifyResults(result,
      structure)

print "Total : %.2f A2" % result.totalArea()
for key in classArea:
    print key, ": %.2f A2" % classArea[key]
```

Results

Dataset 1. List of PDB codes used for the performance analysis

http://dx.doi.org/10.5256/f1000research.7931.d112977

This set was generated from the most restrictive list of structures in the PISCES database (Wang & Dunbrack, 2003). 88 PDB files were selected randomly from a set of size intervals in this list, to get an approximately even distribution in size.

Dataset 2. Zip-archive with raw data for the performance analysis in Figure 2 and Figure 3

http://dx.doi.org/10.5256/f1000research.7931.d112978

See the file explanation.txt for an overview of the contents of the archive.

The computational efficiency of the two algorithms was compared by running the FreeSASA command-line program with different parameters on a set of 88 PDB structures selected from the PISCES database (Wang & Dunbrack, 2003) (see Dataset 1 for a list). PISCES specifies a specific chain in each structure, but in the following all chains were used, which resulted in the largest structure having over 30,000 atoms (1jz8). To average out some variation in the running time in these rather short calculations (in some cases < 10 ms), the fastest calculations were run two to five times. As we will see below, error bars are relatively small along that axis.

To measure the accuracy of the two algorithms, a reference SASA value, A_{ref}, was calculated using L&R with 1000 slices per atom for each structure. The error of a given SASA-value, A, is then $\varepsilon = |A - A_{\mathrm{ref}}|/N$, where N is the number of atoms in the structure. Calculation time T is measured as the wall time of the entire calculation including reading and writing files. Dataset 2 contains the values of A, A_{ref}, N and T used to produce Figure 2 and Figure 3.

Figure 2 shows ε versus T/N, averaged over the 88 PDB structures. At low resolution S&R is considerably faster than L&R, and at high resolution L&R is faster, with a crossover at around 1000 test points or 20 slices per atom (20 slices is the default setting in FreeSASA).

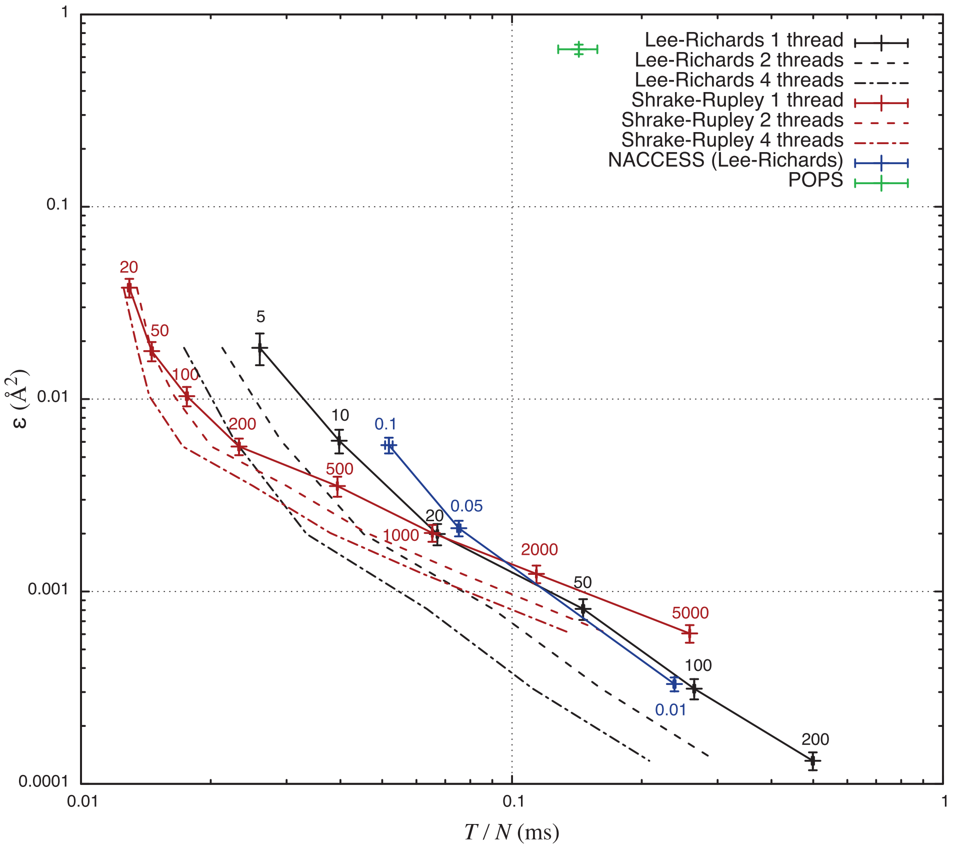

Figure 2. Precision and calculation time. The mean of the error ε in SASA vs T/N, for the two algorithms in FreeSASA plus the programs NACCESS and POPS. Labels indicate the resolution used for each set of calculations, and error bars the standard error along both axes. The solid lines are only there to guide the eye, and the dashed lines indicate the analogous lines when using 2 and 4 threads in FreeSASA. An L&R run with 1000 slices was used as A_{ref} when calculating ε for both approximations. NACCESS uses L&R and was run with three values of the z-parameter (0.1, 0.05 and 0.01, corresponding to 10, 20 and 100 slices per atom), a run with z-parameter 0.005 was used as A_{ref} (using even lower z-values gave inconsistent results). The NACCESS reference calculation was also used as reference for POPS. All programs were compiled using GCC 4.9.3 with the optimization flag "-Ofast" and the tests were run on an Intel Core i5-2415M CPU at 2.30 GHz. The raw data for this figure can be found in Dataset 2.

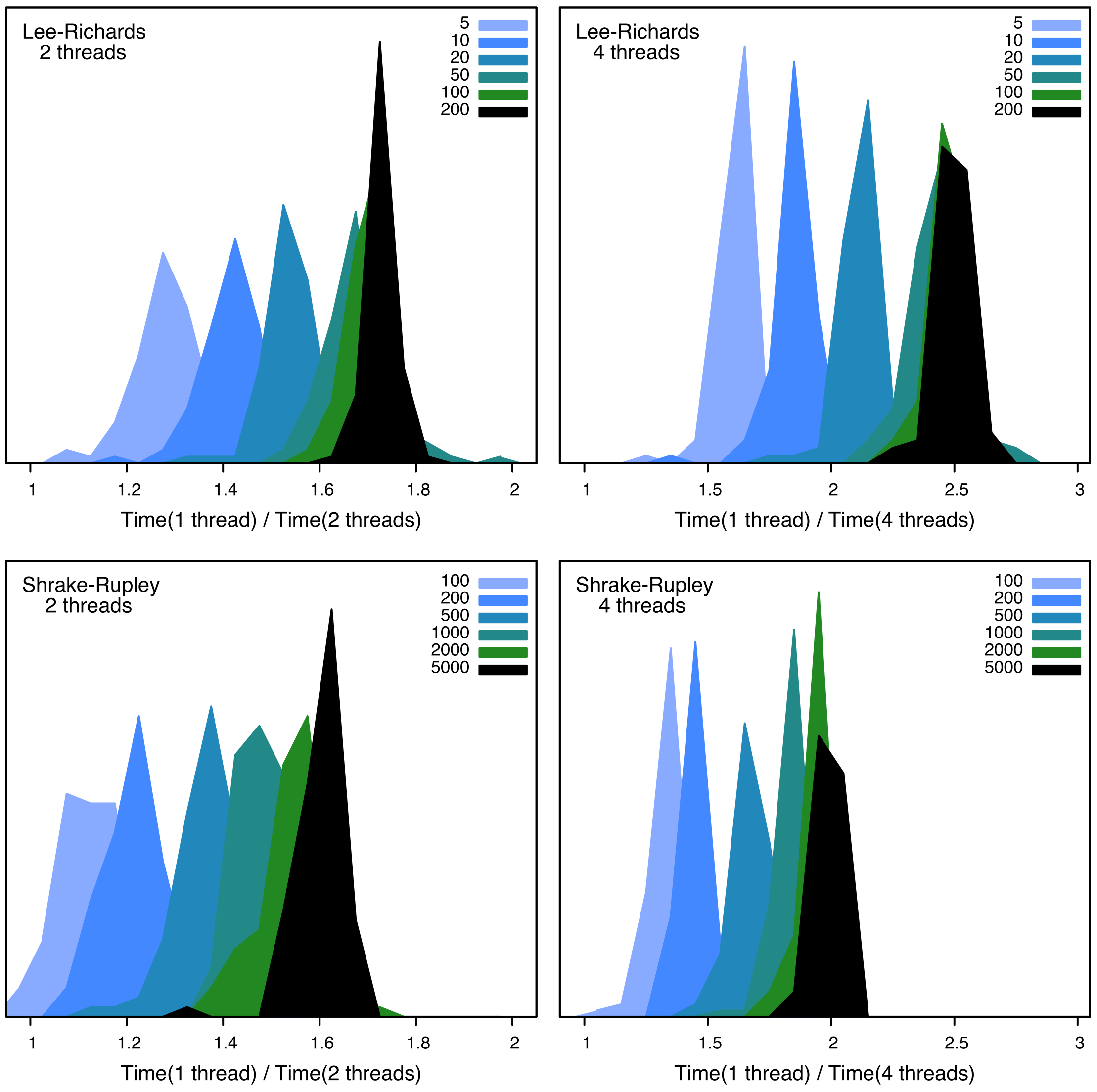

Figure 3. Parallelization. The histograms shows the distribution of the calculation time using two or four threads divided by the time using one thread. Thus if this fraction is two or four, respectively, we have "perfect" parallelization. The legends indicate the resolution of the calculation: for L&R, slices per atom, and for S&R, number of test points. The raw data for this figure can be found in Dataset 2.

Comparisons were done with NACCESS 2.1.1 (Hubbard & Thornton, 1993), DSSP 2.2.1 (Touw *et al.*, 2015), NSOL 1.7 (Masuya, 2003), POPS 1.6.4 (Cavallo *et al.*, 2003) and Triforce 0.1 (Drechsel *et al.*, 2014). The list could potentially have been a lot longer; some programs were left out on the basis of being closed source, poorly documented or no longer available. NACCESS was included in spite of its limiting license due to its popularity. The SASA facilities in molecular dynamics packages were not considered since these cater to a different use case.

NACCESS allows the user to choose arbitrary resolution and can therefore be used as a reference for itself, and POPS was optimized with NACCESS as reference. NACCESS uses L&R and performs very similarly to FreeSASA using L&R. The POPS algorithm is intended as a fast coarse-grained approximation; its authors state an average error of 2.6 Å^2 per atom (Cavallo *et al.*, 2003). In Figure 2 the mean ε is lower than that, which is expected, since this error is measured over the total SASA, not atom by atom. A fit showed that POPS runs in $O(N^2)$ time (using the data in the file `pops.dat` in Dataset 2), which to some extent explains the relatively long mean calculation time per atom.

The other programs listed above were left out of Figure 2 because they can not be compared under the same premises. DSSP calculates many different things in addition to its 200 test-point S&R-calculation, and the total running time is therefore naturally longer than the corresponding calculation in FreeSASA, although the accuracy should be comparable for the same number of test points. The program NSOL uses S&R, but does five different SASA calculations on the same input using different parameters. The NSOL code was modified to only do one of the five calculations, and is then only slightly slower than FreeSASA using the same number of test points. Lastly, Triforce is not suitable for comparison in this particular use case because it has a high initialization cost, which makes it slow for calculating the SASA of an isolated structure.

In single-threaded mode, FreeSASA using L&R is almost indistinguishable from NACCESS in Figure 2, but it is significantly faster when 2 or 4 threads are used. The effect of spreading the calculation over several threads is shown in more detail in Figure 3. Since the generation of cell lists is not parallelized, using more than one thread only gives a significant performance benefit in the high resolution limit. Based on these results, the default has been set to two threads. Depending on the nature of the calculations, this speedup can make a noticeable difference.

Summary

FreeSASA is an efficient library for calculating the SASA of protein, RNA and DNA structures. Its main advantages over other commonly used tools is that it is open source, easily configurable and can be used both as a command line tool, a C library and a Python module. The tests above demonstrate that it runs as fast as, or faster than, some popular tools at a given resolution, and can be boosted further by parallelizing the calculation.

Data and software availability

Data

F1000Research: Dataset 1. List of PDB codes used for the performance analysis, 10.5256/f1000research.7931.d112977 (Mitternacht, 2016a).

F1000Research: Dataset 2. Zip-archive with raw data for the performance analysis in Figure 2 and Figure 3, 10.5256/f1000research.7931.d112978 (Mitternacht, 2016b).

Software

Latest source code and software available from http://freesasa.github.io/

Archived source code as at the time of publication http://dx.doi.org/10.5281/zenodo.45239 (Mitternacht, 2016c).

License GNU GPL v3 (http://www.gnu.org/licenses/gpl-3.0.en.html).

Competing interests

No competing interests are disclosed.

Grant information

The author declared that no grants were involved in supporting this work.

Acknowledgments

Thanks go to Edvin Fuglebakk for comments on the code and documentation. Thanks to Sandhya P. Tiwari and Anders Irbäck for comments on the manuscript. Thanks to João Rodrigues for suggestions for improved functionality.

References

Cavallo L, Kleinjung J, Fraternali F: **POPS: A fast algorithm for solvent accessible surface areas at atomic and residue level.** *Nucleic Acids Res.* 2003; **31**(13): 3364–3366.
PubMed Abstract | Publisher Full Text | Free Full Text

DeLano WL: **The PyMOL molecular graphics system**. 2002.
Reference Source

Drechsel NJ, Fennell CJ, Dill KA, *et al.*: **TRIFORCE: Tessellated Semianalytical Solvent Exposed Surface Areas and Derivatives.** *J Chem Theory Comput.* 2014; **10**(9): 4121–4132.
PubMed Abstract | Publisher Full Text | Free Full Text

Eisenhaber F, Lijnzaad P, Argos P, *et al.*: **The double cubic lattice method: efficient approaches to numerical integration of surface area and volume and to dot surface contouring of molecular assemblies.** *J Comput Chem.* 1995; **16**(3): 273–284.
Publisher Full Text

Finkelstein AV, Ptitsyn O: **Protein physics: a course of lectures**. Academic Press, London, 2002.
Reference Source

Fraczkiewicz R, Braun W: **Exact and efficient analytical calculation of the accessible surface areas and their gradients for macromolecules.** *J Comput Chem.* 1998; **19**(3): 319–333.
Publisher Full Text

Hubbard SJ, Thornton JM: **NACCESS**. *Computer Program, Department of Biochemistry and Molecular Biology*, University College London, 1993.
Reference Source

Lee B, Richards FM: **The interpretation of protein structures: estimation of static accessibility.** *J Mol Biol.* 1971; **55**(3): 379–400.
PubMed Abstract | Publisher Full Text

Mantina M, Chamberlin AC, Valero R, *et al.*: **Consistent van der Waals radii for the whole main group.** *J Phys Chem A.* 2009; **113**(19): 5806–5812.
PubMed Abstract | Publisher Full Text | Free Full Text

Masuya M: **NSOL: A numerical calculation program of molecular surface area, volume, and solvation energy**. 2003.
Reference Source

Mitternacht S: **Dataset 1 in: FreeSASA: An open source C library for solvent accessible surface area calculations.** *F1000Research.* 2016a.
Data Source

Mitternacht S: **Dataset 2 in: FreeSASA: An open source C library for solvent accessible surface area calculations.** *F1000Research.* 2016b.
Data Source

Mitternacht S: **FreeSASA 1.0.1: Solvent Accessible Surface Area Calculations.** *Zenodo.* 2016c.
Data Source

Sanner MF, Olson AJ, Spehner JC: **Reduced surface: an efficient way to compute molecular surfaces.** *Biopolymers.* 1996; **38**(3): 305–320.
PubMed Abstract | Publisher Full Text

Shrake A, Rupley JA: **Environment and exposure to solvent of protein atoms. Lysozyme and insulin.** *J Mol Biol.* 1973; **79**(2): 351–371.
PubMed Abstract | Publisher Full Text

Swinbank R, Purser RJ: **Fibonacci grids: A novel approach to global modelling.** *Q J R Meteorol Soc.* 2006; **132**(619): 1769–1793.
Publisher Full Text

Touw WG, Baakman C, Black J, *et al.*: **A series of PDB-related databanks for everyday needs.** *Nucleic Acids Res.* 2015; **43**(Database issue): D364–D368.
PubMed Abstract | Publisher Full Text | Free Full Text

Tsai J, Taylor R, Chothia C, *et al.*: **The packing density in proteins: standard radii and volumes.** *J Mol Biol.* 1999; **290**(1): 253–266.
PubMed Abstract | Publisher Full Text

Wang G, Dunbrack RL Jr: **PISCES: a protein sequence culling server.** *Bioinformatics.* 2003; **19**(12): 1589–1591.
PubMed Abstract | Publisher Full Text

Weiser J, Shenkin PS, Still WC: **Approximate atomic surfaces from linear combinations of pairwise overlaps (LCPO).** *J Comput Chem.* 1999; **20**(2): 217–230.
Publisher Full Text

Westbrook JD, Shao C, Feng Z, *et al.*: **The chemical component dictionary: complete descriptions of constituent molecules in experimentally determined 3D macromolecules in the Protein Data Bank.** *Bioinformatics.* 2015; **31**(8): 1274–1278.
PubMed Abstract | Publisher Full Text | Free Full Text

Xu D, Zhang Y: **Generating triangulated macromolecular surfaces by Euclidean Distance Transform.** *PLoS One.* 2009; **4**(12): e8140.
PubMed Abstract | Publisher Full Text | Free Full Text

Open Peer Review

Current Referee Status: ✓ ✓

Version 1

Referee Report 29 February 2016

doi:10.5256/f1000research.8538.r12528

✓ **Yaoqi Zhou**, **Yuedong Yang**
Institute of Glycomics, Griffith University, Gold Coast, QLD, Australia

Solvent accessible surface area is a frequently calculated quantity in structural bioinformatics. Although many tools are available, an open source, easy-to-use program is certainly welcome. The author did a careful test and performed comparative studies to existing tools. The code is clearly written and well documented. There are no major changes required for this clearly written manuscript. Here are some minor questions requiring further clarifications and additions.

1. For multiple chains or a protein-ligand complex structure, does Freesasa yield the ASA for isolated chains or chains in the complex structure? It would be better if the program has an option to calculate the change of ASA on a residue before and after binding to another molecule (ligand, RNA, DNA, another chain) assuming that there is no structural change upon binding. This would allow to identify the functional residues.

2. Please clarify the command to produce a residue-level ASA, rather than the atomic level ASA.

3. What are default recommendations for the ASA calculation in term of resolution required or choice of LR and SR approximations?

4. Are the default atomic radii employed here the same as used in DSSP and/or NACCESS? If different, what are the main differences?

5. Is this method faster than analytical methods? If so, by how much?

6. Can authors provide a table that lists all possible commands?

7. The program would be more useful if it can directly read in Biopython data structure and work with pymol.

We have read this submission. We believe that we have an appropriate level of expertise to confirm that it is of an acceptable scientific standard.

Competing Interests: No competing interests were disclosed.

Author Response 29 Feb 2016

Simon Mitternacht,

Thanks for your comments and suggestions. I interpret most of them as being better solved by improving the functionality and/or documentation of the program itself, rather than revising the paper. For the sake of brevity not all the functionality is described in the paper, but is available in the online documentation.

1. The default behavior is to calculate the SASA of the chains in the complex. The command option '--chain-groups' described in the paper allows the user to look at combinations of chains in isolation. The option '--separate-chains', is not discussed in the paper, but can be used to treats each chain separately (i.e. for a four chain protein '--chain-groups=A+B+C+D' would be equivalent to '--separate-chains'). My intention was to give the user the ability to calculate the total ASA of different combinations of chains, and then do the arithmetic of the changes themselves. I will look into adding command-line options to do this automatically in future versions of the program.

2. The option '--foreach-residue' (not described in the paper) can be used to print the SASA of each residue.

3. It is hard to give a general recommendation here, but I chose to use the same default resolution as NACCESS since this seems to be an accepted standard. If one is looking for changes due to minor conformational changes, a higher resolution might be needed. The average of A_{ref}/N in Dataset 2 is 5.6 $Å^2$, and the average error in Figure 2 for the default resolution is around 0.002 $Å^2$, i.e. the error in the total SASA (A) is on average $0.002/5.6 < 1/1000$.

4. As mentioned in the paper, the ProtOr atomic radii are used. I think the easiest way of comparing these with those employed in NACCESS and DSSP is to compare the radii in the configuration-files supplied with the program (in the directory share/).

5. I was not able to get hold of FANTOM or any other analytical program, so I have unfortunately not been able to compare directly. As mentioned I compared with the semi-analytical Triforce, which was significantly slower when run on a single structure due to high initialization cost. I assume Triforce performs better when the same instance of the program is run on many structures, but that is a different use case.

6. The command 'freesasa -h' lists all options. I decided not to provide this list in the paper because it will probably expand as time goes, and then the paper would quickly become outdated.

7. Thank you for these suggestions, I will look into ways of adding more functionality to the Python bindings.

Competing Interests: No competing interests were disclosed.

Referee Report 26 February 2016

doi:10.5256/f1000research.8538.r12524

Simon Hubbard
Faculty of Life Sciences, University of Manchester, Manchester, UK

This is a simple and effective implementation of the two main heuristics used for calculating solvent accessible surface area of atomic structures such as proteins. The algortihm, FreeSASA, is suitably described and tested in the article. I see no real need to ask for any changes, as the author has done a very professional and nice job here. It downloads and installs very smoothly and offers most of the functionality that NACCESS offers, so will be appreciated by legacy users I would imagine (though some things have not been implemented, probably with good reason - either way, not a problem for me - and the author can react to requests or users can do a little work to get what they want in terms of formatting). I am sure the ability to thread across multiple cores will be beneficial and speed things up, and the API will make it slot into python pipelines etc. It's possible that it won't directly supplant some NACCESS dependencies which rely on its rather old formatting (.rsa file for example perhaps) but this is probably only a minor concern and easily fixed if needed (since its fully open source, its probably reduced to changing a few print statements anyway).

I have read this submission. I believe that I have an appropriate level of expertise to confirm that it is of an acceptable scientific standard.

Competing Interests: The reviewer as one of the original co-authors of NACCESS, one of the software tools this program is compared with. This is freely available to academic users but licenced, via UCL e-lucid, to commerical users.