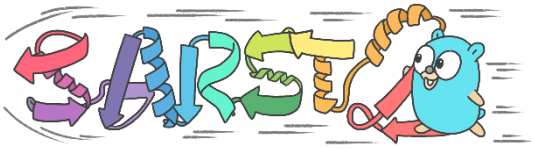
# SARST2 User Manual

## English Version



## Table of contents


Table of contents ..... 1
1. About this software ..... 2
2. Download, decompression, and installation ..... 2
3. Contents of the software folders ..... 3
4. Quick guide ..... 3
5. Manual: sarst2 ..... 4
   5.1 Usage ..... 4
   5.2 Program options ..... 4
   5.3 Example usage: database structural similarity search ..... 7
   5.4 Example usage: one-against-all alignments ..... 7
   5.5 Example usage: pairwise structure alignment ..... 7
   5.6 Output protein structural superimpositions ..... 8
   5.7 Generate interactive HTML result pages ..... 8
6. Manual: formatdb ..... 10
   6.1 Usage ..... 10
   6.2 Program options ..... 10
   6.3 Example usages ..... 11
7. Manual: readdb ..... 12
   7.1 Usage ..... 12
   7.2 Program options ..... 12
   7.3 Example usages ..... 12

# 1. About this software

SARST2 (Structural similarity search Aided by Ramachandran Sequential Transformation, version 2) is a high-performance protein structure alignment algorithm. It supports both database searches for structural similarity using a given query protein, as well as pairwise structure alignments between two protein structures.

This software is published along with the following paper and will be frequently updated via the URLs provided in the publication:

| | |
|---|---|
| Title | SARST2, a high-throughput protein structure alignment algorithm for searching massive databases |
| Authors | Wei-Cheng Lo*, Arieh Warshel, Chia-Hua Lo, Chia Yee Choke, Yan-Jie Li, Shih-Chung Yen, Jyun-Yi Yang and Shih-Wen Weng |
| Institute | Institute of Bioinformatics and Systems Biology, National Yang Ming Chiao Tung University, Hsinchu, Taiwan, Republic of China |

*Corresponding author

Download URLs:
https://github.com/NYCU-10lab/sarst
https://10lab.ceb.nycu.edu.tw/sarst2

# 2. Download, decompression, and installation

The latest version of the SARST2 program and pre-formatted target databases are available at the URLs listed above.

After downloading a compressed archive, use tar, gzip, or zip utilities to extract the files, depending on the format of the archive. Once extracted, the SARST2 program files are provided as pre-compiled executable binaries and do not require installation.

For example, if you download the archive SARST2-v2.0.26-Linux.x86_64.tar.gz, you can decompress it under Linux using the following command:

```
tar xfp SARST2-v2.0.26-Linux.x86_64.tar.gz
```

After extraction, you can run the sarst2 program with the following commands:

```
cd SARST2-v2.0.26-Linux.x86_64/bin
chmod +x sarst2
./sarst2 -h
```

# 3. Contents of the software folders

| | |
|---|---|
| bin | 64-bit executables for Linux, Windows, and macOS. |
| dbs | Pre-formatted target databases: PDB-2022 and SCOP-2.07. |
| doc | User manuals in multiple languages. |

# 4. Quick guide

There are three main programs in this software package, as described below,

| | |
|---|---|
| sarst2 | Implementation of the SARST2 protein structure alignment algorithm. |
| formatdb | A database formatting tool that allows users to create custom target databases for structure search and alignment. |
| readdb | A tool for extracting the protein amino acid and linearly-encoded structure sequences stored in a pre-formatted target database. |

Running any of the above programs without parameters (or with -h) will display a brief text-based help message for that program.

Linux, macOS

```
./sarst2
./formatdb
./readdb
./sarst2 -h
./formatdb -h
./readdb -h
```

Windows (Cmd or PowerShell)

```
.\sarst2.exe
.\formatdb.exe
.\readdb.exe
.\sarst2.exe -h
.\formatdb.exe -h
.\readdb.exe -h
```

# 5. Manual: sarst2

## 5.1 Usage

```
./sarst2    query structure    subject structure(s)   [Options]
            ---------------    --------------------
            > one PDB/CIF file  > can be
                                1. -db + a pre-formatted database
                                2. multiple PDB/CIF files
                                3. folders of PDB/CIF files
                                4. one PDB/CIF file (pairwise)
```

## 5.2 Program options

| | | |
|---|---|---|
| -db | [str] | The target database of subject structures to search.<br>(default: none) |
| -brief | [int] | Number of subject structures to show one-line summaries.<br>(default: 500) |
| -detail | [int] | Number of subject structures to show detailed alignment data.<br>(default: 500) |
| -t | [int] | Number of threads.<br>It must be ≥ 0; 0 means all processors will be used.<br>(default: 0, all processors) |
| -w | [int] | Word size.<br>(default: 5) |
| -orderby | [int] | Sort the hit list by one of the following factors, 1: Conf-score--, 2: TM-score--, 3: sequence identity--, or 4: RMSD++, where --/++ means descending/ascending order.<br>(default: 1, Conf-score--)<br>(ignored in a pairwise alignment) |
| -mode | [int] | Searching mode, 1: accurate, 2: balanced, 3: quick,<br>or other values: auto.<br>(default: auto) |
| -f | [int] | Enable minor filters, 0: off, 1: on, or other values: auto.<br>(default: auto)<br>(always 0, disabled, in a pairwise alignment) |

| | | |
|---|---|---|
| -C | [float] | Conf-score (confidence score) threshold.<br>It must be between 0 and 1; 0 means no threshold is applied.<br>(default: 0.5)<br>(always disabled in a pairwise alignment) |
| -pC | [float] | Cutoff of the final pC-value, i.e., -log2(C).<br>It must be ≥ 0; 0 means no cutoff is applied.<br>(default: 1.0, equivalent to -C = 0.5)<br>(always disabled in a pairwise alignment) |
| -e | [float] | Cutoff of the pC-value, applied to each filter and refinement step.<br>Given the same -e and -pC, -e discards more irrelevant hits.<br>It must be ≥ 0; 0 means no cutoff is applied.<br>(default: 1.0)<br>(always disabled in a pairwise alignment) |
| -tmcut | [float] | TM-score cutoff.<br>It must be ≥ 0; 0 means no cutoff is applied.<br>A TM-score ≥ 0.7 by SARST2 might imply family-level homology.<br>(default: 0.15)<br>(always disabled in a pairwise alignment) |
| -mem | [T/F] | Cache all subject protein data in memory.<br>(default: T)<br>(always T, enabled, in a pairwise alignment) |
| -q | [T/F] | Quick output style.<br>Display results in a simplified, parser-friendly format.<br>(default: F) |
| -sa | [T/F] | Display the structure-based sequence alignment.<br>(default: T) |
| -mat | [T/F] | Display the transformation matrix for superimposition.<br>(default: F) |
| -nmsbj | [T/F] | Normalize the TM-score by the size of the subject structure.<br>(default: F) |
| -nmavg | [T/F] | Normalize the TM-score by the average size of the query structure and each subject structure.<br>(default: F) |
| -nmusr | [float] | The protein size for normalizing the TM-score.<br>It should be ≥ minimum size of the two structures; otherwise, the TM-score may be > 1. |

| | | |
|---|---|---|
| -d | [float] | The d0 for scaling the TM-score, e.g., 5.0 Angstroms (Å). |
| -ml | [T/F] | Apply machine learning.<br>(default: T)<br>(always F, disabled, in a pairwise alignment) |
| -fdp | [str] | Dynamic programming algorithm for the filtering steps.<br>Supported options: NW (Needleman-Wunsch), SW (Smith-Waterman)<br>(default: NW) |
| -rdp | [str] | Dynamic programming algorithm for the refinement step.<br>Supported options: NW (Needleman-Wunsch), SW (Smith-Waterman)<br>(default: NW) |
| -swp | [str] | Path to the user-specified swap file.<br>Using a swap file can reduce the memory cost.<br>(default: none) |
| -Sout | [str] | Folder to output the structure superimposition files.<br>The folder will be created if it does not exist.<br>The number of superimposition files is confined by the -detail option.<br>(default: none) |
| -html | [str] | Make an HTML output folder.<br>Superimposed structure files will also be generated in the HTML folder.<br>(default: none) |
| -jsmol | [str] | Set the path to the JSmol JavaScript package for displaying superimposed structures in the HTML output.<br>It can be a local disk folder or an HTTP(S) URL.<br>(default: none)<br>(trial URL: "https://10lab.ceb.nycu.edu.tw/ext/jsmol") |
| -pssm_out | [str] | File to store the PSSMs of the structural and sequence codes applied in this algorithm.<br>(default: none) |
| -pssm_pC | [float] | The pC-value cutoff for PSSM construction.<br>(default: 0.05) |
| -h | | Print the help message (quick guide). |

## 5.3 Example usage: database structural similarity search

Search the query structure against a pre-formatted target database

```
./sarst2 Qry.pdb -db my_db/my_proteins.db -brief 10 -w 7 -e 0.1
./sarst2 Qry.pdb -db my_db/my_proteins.db -brief 10 -d 5.0 -sa F
```

In the above example, the target database is stored in the folder "my_db", and "my_proteins.db" is the file stem of the target database files. See the **Manual: formatdb** for the actual database file list.

## 5.4 Example usage: one-against-all alignments

Search the query structure against listed subject structures

```
./sarst2 Qry.pdb Sbj1.pdb Sbj2.cif Sbj3.pdb -mat T
```

Search the query structure against subject structure files specified with wildcard patterns

```
./sarst2 Qry.pdb "set1/*.pdb" "set2/1a???.cif" -nmavg T
```

In this example, "set1/*.pdb" and "set2/1a???.cif" are enclosed in quotation marks and contain wildcard characters. The sarst2 program will automatically expand these wildcard patterns and retrieve the matching file names internally. If the patterns are not enclosed in quotation marks, the operating system will expand the wildcards instead. When the number of matching files is large, the resulting command-line argument list may exceed system limits and cause the command to fail. Therefore, we recommend enclosing wildcard patterns in quotation marks to allow sarst2 to handle file listing internally, rather than relying on the operating system's default behavior.

Search the query structure against several folders containing subject structures

```
./sarst2 Qry.pdb set1 set2 -nmavg T
```

In this example, set1 and set2 are folders that may contain protein structure files. The sarst2 program will automatically retrieve all files in these folders (equivalent to set1/* and set2/*). Files identified as PDB or CIF format will be selected and aligned against the query structure.

## 5.5 Example usage: pairwise structure alignment

Align the query structure with one subject structure

```
./sarst2 Qry.pdb Sbj.pdb -sa F
./sarst2 Qry.pdb Sbj.cif -mat T
```

## 5.6 Output protein structural superimpositions

Generate query-subject structural superimposition PDB files

```
./sarst2 Qry.pdb -db prot/myDb -detail 100 -Sout output_folder
./sarst2 Qry.pdb "set1/*.cif" -detail 100 -Sout output_folder
```

Using the -Sout output_folder option, the superimposed protein structures in PDB format will be output to the user-specified folder. The number of superimposed structures generated is defined by the -detail option. Each output file is named Qry-SbjSN.pdb, where SN represents the serial number of the subject protein in the hit list. In each superimposition file, the chain IDs for the query and subject protein structures are Q and S, respectively. The two chains are separated by a TER record, as illustrated below:

```
ATOM    150  CA  LEU Q 150      29.000  -8.400   0.800                       C
ATOM    151  CA  GLY Q 151      26.000  -9.600   2.600                       C
ATOM    152  CA  TYR Q 152      25.400  -6.800   5.000                       C
ATOM    153  CA  GLN Q 153      23.600  -3.800   3.600                       C
ATOM    154  CA  GLY Q 154      22.800  -2.800   7.200                       C
TER
ATOM      1  CA  MET S   1      24.400   9.800 -10.000                       C
ATOM      2  CA  VAL S   2      27.200  11.800 -11.400                       C
ATOM      3  CA  LEU S   3      28.800  15.200 -10.400                       C
ATOM      4  CA  SER S   4      29.800  17.800 -13.000                       C
ATOM      5  CA  GLU S   5      33.400  19.200 -13.000                       C
ATOM      6  CA  GLY S   6      32.000  22.400 -11.600                       C
```

As shown in the figure, only alpha carbon (Cα) atoms appear in the superimposition file. This is because SARST2 performs all computations based solely on the Cα coordinates. The orientation of the query structure remains fixed across all superimposition files, while each subject structure is transformed (rotated and translated) according to its alignment with the query structure to achieve superimposition.

To visualize the superimposed structures, we recommend using RasMol or RasWin (http://www.openrasmol.org/). Since only Cα atoms are present in the superimposition files, the display mode in RasMol should be set to "backbone".

## 5.7 Generate interactive HTML result pages

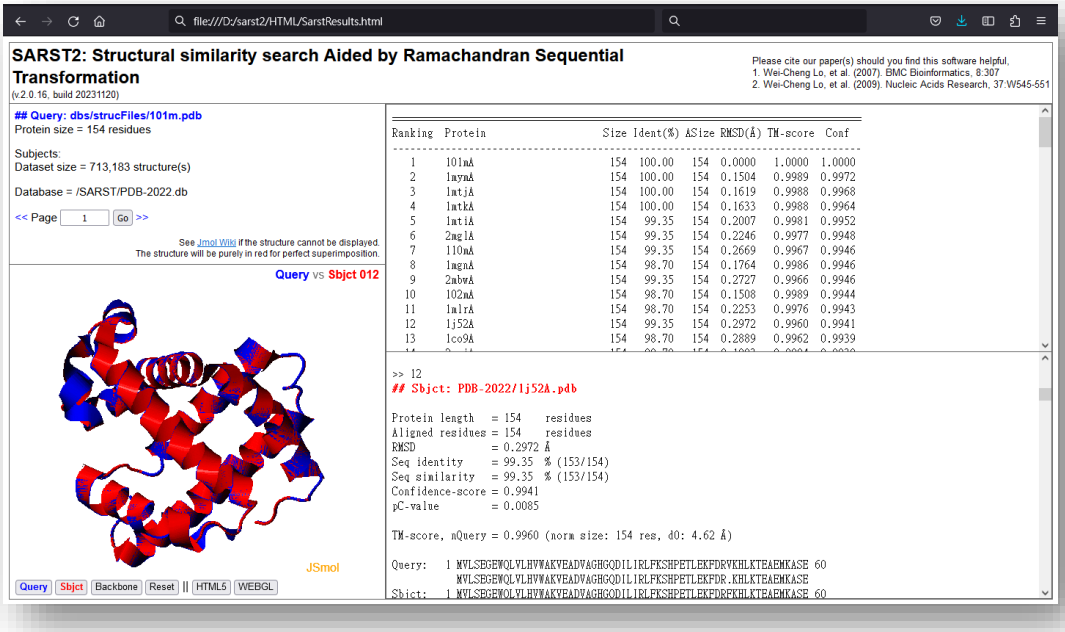
Generate an HTML document with online JSmol scripts

```
./sarst2 Qry.pdb -db my_db/my_proteins.db -html output_folder
-jsmol "https://10lab.ceb.nycu.edu.tw/ext/jsmol"
```

Generate an HTML document with a local JSmol script folder (Windows)

```
./sarst2 Qry.pdb "set1/*.cif" -detail 100 -html output_folder
-jsmol "file:///D:/bioinfo/software/jsmol"
```

Using the "-html output_folder" option, an HTML result document and corresponding superimposed protein structure files will be generated in the specified folder. The "-html" option must be used together with "-jsmol", which specifies the URL or local path of the JSmol package (version 2013). JSmol is an interactive 3D molecular structure viewer that runs in web browsers and supports most major modern browsers.




The main file in the HTML output folder is SarstResults.html, which should be opened in a web browser. Other HTML files are embedded within the main page using inner frames. A subfolder named "sup" will also be created; it stores the structural superimposition files between the query and each subject protein in the hit list.

Depending on your operating system, browser, or antivirus software, you may need to adjust the security settings to allow the browser to execute JavaScript and access the superimposed structure files in the "sup" folder, so that the JSmol 3D viewer can function properly.

# 6. Manual: formatdb

## 6.1 Usage

```
./formatdb      subject structure(s)   -db database   [Options]
                --------------------
                > can be
                1. multiple PDB/CIF files
                2. folders containing PDB/CIF files
                3. a plain text file listing PDB/CIF file paths
```

## 6.2 Program options

| | | |
|---|---|---|
| -db | [str] | The target database of subject structures to create.<br>(default: none) |
| -flist | [str] | Plain text file listing PDB/CIF file paths. This argument can be used along with common subject file arguments.<br>(default: none) |
| -t | [int] | Number of threads.<br>It must be ≥ 0; 0 means all processors will be used.<br>(default: 0, all processors) |
| -split | [int] | Split the database into subsets, each with the number of subject structures specified by this option. Database splitting helps prevent the database files from exceeding the file size limit of the disk.<br>(default: none) |
| -save_disk | [T/F] | Round the coordinates of atoms from three decimal places into one decimal place to reduce disk usage.<br>(default: F) |
| -keep_order | [T/F] | Keep the order of the subject structures stored in the database as their input order. Setting T may slow down database creation.<br>(default: F) |
| -h | | Print the help message (quick guide). |

## 6.3 Example usages

Create a target database for listed subject structure files

```
./formatdb Sbj1.pdb Sbj2.cif Sbj3.pdb -db myDb -keep_order T
```

Several database files with filenames starting with myDb will be created. Enabling -keep_order will preserve the order of subject structures in the target database according to how they were listed in the command-line arguments.

Create a target database for listed subject structure files with wildcards

```
./formatdb "set1/*.pdb" "set2/*.cif" Sbj1.pdb Sbj2.cif -db myDb
```

When listing subject files, it is fine to mix arguments with and without wildcards. It is recommended to enclose wildcard arguments in quotation marks, so that the program can correctly handle file expansion.

Create a target database based on a file list of subject structures

```
./formatdb -flist protlist.txt Sbj1.pdb Sbj2.cif -db myDb
```

The file protlist.txt should contain a list of file paths, with one file path per line.

Create a target database from folders containing subject structure files

```
./formatdb folder1 folder2 -db myDb -save_disk T -split 50000
```

Enabling -save_disk will round the Cα coordinates to one decimal place to save storage space. The "-split 50000" option will result in multiple subset databases, each containing at most 50,000 structures. This split option is particularly useful when the size of the formatted database files may exceed the maximum file size supported by some operating systems or disk formats.

# 7. Manual: readdb

## 7.1 Usage

```
./readdb  target database  output file  [-seq sequence type]
          ---------------  -----------  -------------
          > must be SARST2  > will be in         > can be
            pre-formatted     FASTA format       1. AA
                                                 2. AAT
                                                 3. SARST
                                                 4. Four-symbol SSE
```

## 7.2 Program options

| | | |
|---|---|---|
| -seq | [str] | The output sequence type.<br>AA amino acid sequence<br>AAT five-symbol amino acid type sequence<br>SARST SARST Ramachandran-code sequence<br>SSE four-symbol secondary structure element sequence<br>(default: AA) |
| -h | | Print the help message (quick guide). |

## 7.3 Example usages

Extract subject sequences from a SARST2 target database

```
./readdb my_db/my_proteins.db seqs.fasta
./readdb my_db/my_proteins.db seqs.fasta -seq SARST
./readdb my_db/my_proteins.db seqs.fasta -seq AAT
```

When -seq is not specified, the default output sequence type is amino acid sequences. The output sequence file (seqs.fasta) will be in FASTA format. If the output file already exists before running readdb, it will be overwritten.