ProBiS Algorithm

2012-2015

User's Guide

November 13, 2015

Laboratory for Molecular Modeling
National Institute of Chemistry
Hajdrihova 19
1000 Ljubljana, Slovenia
www.sicmm.org
Support: konc@cmm.ki.si
Collaborations: dusa@cmm.ki.si

Table Of Contents

Background....4
Installation Instructions....4
Required Libraries....4
Examples....5
  Example 1: Superimpose a pair of protein structures (pairwise alignment)....5
  Example 2: Superimpose a pair of binding sites (pairwise alignment II)....5
  Example 3: Compare a protein against many protein structures....6
  Example 4: Compare a binding site against many protein structures....7
  Example 5: Compare a binding site against many other binding sites....8
Options and modifiers....9
  Options....9
    -align 9
    -compare 9
    -extract 9
    -results 9
    -surfdb 9
    -h 9
  Modifiers....10
    -alno ALIGNMENT_NO....10
    -bsite BSITE, -bsite1 BSITE, or -bsite2 BSITE....10
    -c1, c2....10
    -database 10
    -dist INTER_CHAIN_DIST 10
    -f1, f2....10
    -in INDIR 10
    -local 10
    -longnames....10
    -motif MOTIF, -motif1 MOTIF, or -motif2 MOTIF....10
    -nobb 11
    -nomarkbb 11
    -noclus 11
    -nofp 11
    -noprune 11
    -out OUTDIR 11
    -param PAR_FILE 11
    -sfile SURF_FILE....11
    -super 11
    -verbose 12
    -z_score Z_SCORE 12
Output files....12
    info.json....12
    query.json....12

alignments.json....13
*.cons.pdb, *.bu#.cons.pdb....14
*.rota.pdb....15
Using -align option....15
Using -super modifier....15
FAQ....15
Should I use as input cut-out binding sites (fragments) or complete protein structures?....15
Why .srf file still contains all protein atoms when I used -bsite or -motif to extract only on a binding site?....16
How fast is generation of surfaces (.srf files) with -extract?....16
Is it possible to run local alignment between two proteins. How can I use MOTIF in the parameter file to do this (align two binding sites)?....16
I already have 1phrA.nosql file. However I don't understand how I can get the P-value or significance of match between two structures?....16
I get two json files info.json & query.json. I dont know much about json format, so I don't understand query.json. But cat info.json gives me this {"z_score":1, "qpdb":"1phr", "qcid":"A", "biof": ["1phr.cons.pdb", "1phr.bu1.cons.pdb"]}. Does this means that 1phr has a Z-score = 1 for only self, so it does not matches with anything in proteins.txt?....16
I am trying to run probis on a proteome scale. So a lot of the proteins are models, with UniProt ID but it seems the program reads only first 4 letters of protein name. For example when I run 'mpiexec -host fp167 -np 16 ./probis -surfdb -sfile srfs.txt -f1 P0A626_bs1.srf -c1 A' it generates a nosql file P0A6A.nosql. So I am not sure whether it will correctly read my srfs.txt which contain many modeled structures having long names?....17
How to specify the input or output directory. 17
Can you tell me about the format of alignment.json so that I can parse the output. How do you parse the e-values for your analysis from this file?....18
I was able to figure all the things out and now I can read the JSON files and also find similar binding site in most cases. However in some cases my program dies saying incorrect format of JSON file Perl error -----> malformed JSON string, neither array, object, number, string or atom, at character offset 179 (before "inf, "alignment_score..."). The problem is in alignments.json. Is it a perl problem ?....18
Can I use a protein-protein binding site as input?....18
I am trying to superpose the template's ligand into the query protein based on the rotation and translation matrix that I get from alignments.json. However when I try to superpose the ligand using the transformation matrices, it never lands up inside the predicted binding pocket....18

Background

ProBiS program aligns and superimposes complete protein surfaces, surface motifs, or protein binding sites. It enables pairwise alignments as well as fast database searches for similar proteins or binding sites. The program can find similar binding sites even in proteins with different folds. ProBiS program is parallel, for single or multiple CPU platforms, and implements the latest ProBiS algorithm (Parallel-ProBiS algorithm).

Further information can be found in the following articles:

- Konc,J., Depolli,M., Trobec,R., Rozman,K., Janezic,D. Parallel-ProBiS: Fast parallel algorithm for local structural comparison of protein structures and binding sites. *J. Comp. Chem.*, 2012, doi: 10.1002/jcc.23048.
- Konc,J. and Janezic,D. ProBiS algorithm for detection of structurally similar protein binding sites by local structural alignment. *Bioinformatics* 2010, **26**, 1160-1168.

Installation Instructions

For installation of the ProBiS source code on Ubuntu (or similar system) do the following:

- Unzip the distribution. A directory is created, in which there is the source code of ProBiS algorithm. Change to this directory after the unpacking is complete.

```
$ unzip probis-{version_number}.zip
$ cd probis-{version_number}
```

- Compile ProBiS source code into executable program.

```
$ make probis
```

- When compiled, type 'probis' to start.

```
$ ./probis
```

Required Libraries

ProBiS requires Open MPI library, GNU scientific library and GNU C++ compiler. Installation of these libraries for Ubuntu is easy:

```
$ sudo apt-get install openmpi-bin libopenmpi-dev libgsl0-dev g++
```

Examples

Files for each example are in a directory ‘example{number}’. Just ‘cd’ to an example directory, and you are ready to start! New terms, options, and modifiers, introduced in each example, are marked in bold.

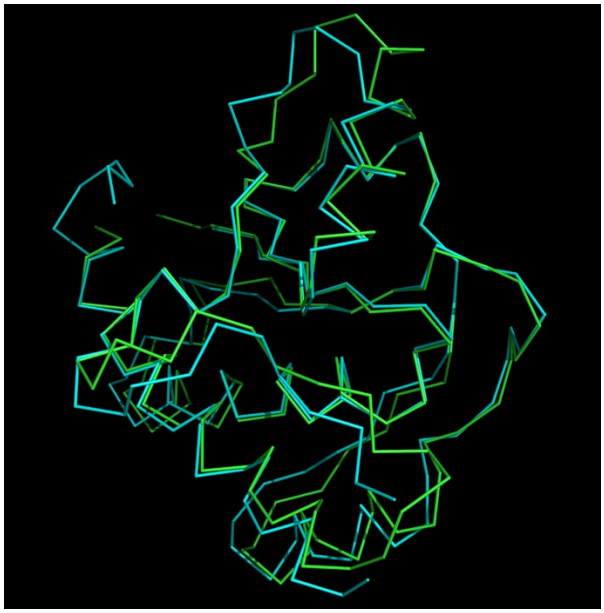
Example 1: Superimpose a pair of protein structures (pairwise alignment).

ProBiS will find all local structural alignments of the compared protein surfaces, and superimpose the second protein’s coordinates onto the first protein’ coordinates according to the found alignments.

- Superimpose two proteins as follows:

  $ ../probis **-compare -super -f1** 1phr.pdb **-c1** A **-f2** 3jvi.pdb **-c2** A

  Output files '*.rota.pdb' contain 3jvi’s coordinates (only!) superimposed onto 1phr according to the three different alignments found in this case. The best superimposition (with highest z_score) is in file ‘1phrA_3jviA.0.rota.pdb’! Alignment scores are in 'REMARK PROBIS' lines in the '.rota.pdb' files.



Superimposed proteins.

Example 2: Superimpose a pair of binding sites (pairwise alignment II).

Superimposing two binding sites is super easy (for correct scoring use .pdb files of complete proteins with ligands!):

- Superimpose the two binding sites defined as residues in a 3.0 Angstrom radius around the SO4 ligands as follows (see -bsite modifier):

  $ ../probis -compare -super **-dist 3.0** -f1 1phr.pdb -c1 A **-bsite1 SO4.158.A** -f2 3jvi.pdb **-bsite2 SO4.201.A** -c2 A

  Alternatively, you can also directly select the SO4 binding site residues by their residue numbers and chain identifier(s) using the '-motif' modifier:

  $ ../probis -compare -super **-motif1 "[:A and (12-19,129-131)]" -motif2 "[:A and (7-15)]"** -f1 1phr.pdb -c1 A -f2 3jvi.pdb -c2 A

  Output file '1phrA_3jviA.0.rota.pdb' contains 3jvi's coordinates (only!) superimposed onto 1phr according to the found alignment of the two binding sites. Alignment scores are in 'REMARK PROBIS' lines in '.rota.pdb' files.

Example 3: Compare a protein against many protein structures.

- Convert proteins from '.pdb' to '.srf' (surface) format, which allows faster computation.

  $ ../probis **-extract** -f1 1phr.pdb -c1 A -srffile 1phrA.srf
  $ for i in $(cat proteins.txt); do ../probis **-extract** -f1 ${i:0:4}.pdb -c1 ${i:4:1} -srffile $i.srf; done

  Expression ${i:0:4} is replaced by first four letters on each line in proteins.txt (I'm using Bash notation, which might differ from yours).

- Run ProBiS on 8 processors for query protein 1phr.A. Resulting pairwise alignments are saved in an '.nosql' file. The hostfile 'hosts' is a text file with hosts specified, one per line. To run the non-parallel version, just omit the 'mpiexec' command and its parameters.

  $ **mpiexec -v -hostfile hosts -np 8** ../probis **-surfdb -sfile srfs.txt** -f1 1phrA.srf -c1 A -nosql example.nosql

- Read a '.nosql' file and convert alignments to Json format. Output the query protein PDB with residues marked by degrees of structural conservation (see *.cons.pdb).

  $ ../probis **-results** -f1 1phr.pdb -c1 A -nosql example.nosql -json example.json

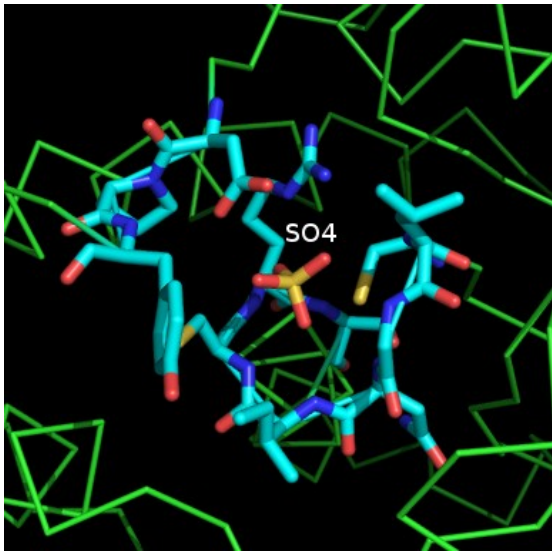
- To get superimposed proteins as .pdb files.

  $ for i in $(cat proteins.txt); do ../probis **-align -alno 0** -nosql example.nosql -f1 1phr.pdb -c1 A -f2 ${i:0:4}.pdb -c2 ${i:4:1}; done

  Output are '*.alno.rota.pdb' files with rotated coordinates of the aligned proteins and the original coordinates of the query protein. To get alignment no. 2 or 3, use '-alno 1' or '-alno 2'; etc. Alignment numbers start with 0!

Example 4: Compare a binding site against many protein structures.

- Extract the SO4 ligand's binding site and save it to '.srf' file. This binding site is defined as all surface residues in a radius of 3.0 Angstroms around the SO4 ligand. Use an intact PDB file, containing the whole protein and ligands, with ProBiS! (Do not cut out binding sites into a separate file by hand, because ProBiS will do this for you.)



Binding site for SO4 ion. Residues lining the binding site are cyan sticks.

Ligand is identified by its residue name, residue number, and chain identifier. Use the '-bsite' and '-dist' modifiers as follows:

```
$ ../probis -extract -bsite SO4.158.A -dist 3.0 -f1 1phr.pdb -c1 A -srffile 1phrA.srf
```

Alternatively, you can also select the SO4 binding site residues by their residue numbers and chain identifier(s) using the '-motif' modifier:

```
$ ../probis -extract -motif "[:A and (12-19,129-131)]" -f1 1phr.pdb -c1 A -srffile
1phrA.srf
```

Syntax is the same as for selecting residues in Jmol. If the binding site was on the interface of two chains (e.g., chain A and B), you could do the following: "[:A and (residue_ numbers_A) or :B and (residue_numbers_B)]", and also provide two chain id's such as '-c1 AB' .

Then:

```
$ for i in $(cat proteins.txt); do ../probis -extract -f1 ${i:0:4}.pdb -c1 ${i:4:1} -srffile
$i.srf; done
```

- Run ProBiS on 8 processors for query binding site on 1phr.A. Resulting pairwise alignments are saved in an '.nosql' file. The hostfile 'hosts' is a text file with hosts specified, one per line.

```
$ mpiexec -v -hostfile hosts -np 8 ../probis -surfdb -local -sfile srfs.txt -f1
1phrA.srf -c1 A -nosql example.nosql
```

Use '-local' modifier if you want to get only alignments in the selected binding site region (it's also a bit faster). Otherwise, the alignments will be extended to whole proteins.

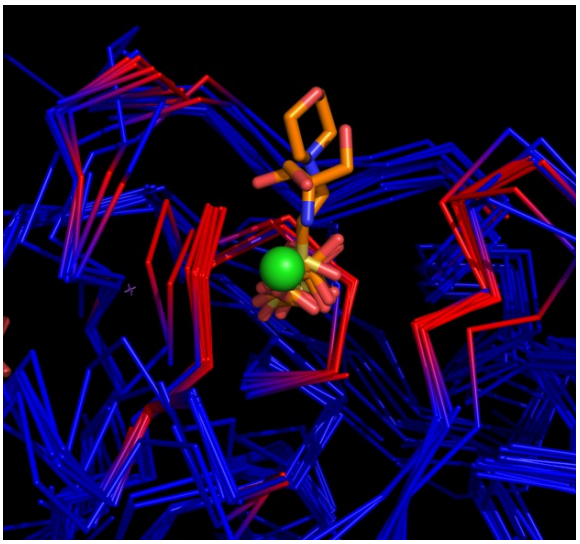
- Read an '.nosql' file and convert alignments to Json format. Output the query protein PDB with residues marked by degrees of structural conservation (see *.cons.pdb).

```
$ ../probis -results -f1 1phr.pdb -c1 A -nosql example.nosql -json example.json
```

- To get PDB files of the superimposed proteins.

```
$ for i in $(cat proteins.txt); do ../probis -align -alno 0 -nosql example.nosql -f1
1phr.pdb -c1 A -f2 ${i:0:4}.pdb -c2 ${i:4:1}; done
```

Output are *.alno.rota.pdb files with rotated coordinates of aligned proteins. To get alignment no. 2 or 3, use '-alno 1' or '-alno 2'; etc. Alignment numbers start with 0!



Multiple superimposed binding sites (red) and ligands in the center.

Example 5: Compare a binding site against many other binding sites.

- Everything is the same as in Example 4, except that you have to prepare a file with ligand codes (format them according to rules described with '-bsite' modifier) that will define the binding sites you want to compare (see 'proteins.txt' in this example's directory). This time, the command for converting .pdb to .srf files is:

```
$ for i in $(cat proteins.txt); do ../probis -extract -bsite ${i:6} -dist 3.0 -f1 $
{i:0:4}.pdb -c1 ${i:4:1} -srffile ${i:0:5}.srf; done
```

Options and modifiers

Run ProBiS with the following command:

probis {OPTION} [MODIFIERS] -f1 PDB_FILE1 -c1 CHAIN_ID1 [-f2 PDB_FILE2 -c2 CHAIN_ID2]

Options

-align

Read a rotational matrix of an alignment from an .nosql file and superimpose the two given proteins accordingly (first run -compare or -surfdb). Output the superimposed proteins' coordinates in a .pdb file. You need to provide both .pdb files that you want to superimpose (see -f1, -c1, -f2, c2 modifiers) and an alignment number (see -alno modifier).

-compare

Compare two protein surfaces (.pdb or .srf files). (If you use .pdb files, surfaces will be computed first.) Output their local structural alignments in an .nosql file. Each alignment consists of a rotational matrix, alignment scores, and aligned residues of the compared proteins.

-extract

Calculate surface of a protein. Redirect the output (which is the surface) to a surface (.srf) file. Surface files can be used instead of .pdb files together with -compare or -surfdb options, which improves performance when doing repetitive comparisons (because surface does not need to be recalculated for each comparison). Option -surfdb works with .srf files exclusively!

-results

Read alignments from an .nosql file, filter them according to their scores, and calculate fingerprint residues (which can also be used as filter). Output results in Json format. In addition, replace B-factors in the query protein’s PDB file with the degrees of structural conservation. If used with -ligdir modifier, output ligands in Json format as well.

-surfdb

Compare the query protein surface (.srf) with other protein surfaces listed in the SURF_FILE (see -sfile modifier). This does the same calculation as the -compare option, but faster, because protein surfaces are precalculated (see -extract option). This options also supports parallel computation on multiple CPUs. Output is the same as with -compare.

-h

Show list of all parameters and their current values. You can copy/paste the parameters into a separate file and change their values (see -param modifier).

Modifiers

-alno ALIGNMENT_NO

Each comparison of a pair of proteins may result in many different local structural alignments.The alignment number can be 0 to 4 (see default CLUS_TO_OUTPUT parameter).

-bsite BSITE, -bsite1 BSITE, or -bsite2 BSITE

This selects protein residues in a certain radius (set by -dist) around the given ligand, and takes these residues as input (use with -extract or -compare options). If used with -compare, it only works with .pdb files (not .srf).
For example:
‘-bsite ATP.305.A’ - ATP (residue name), 305 (residue number), A (chain id)
‘-bsite *.*.B’ - chain B is the ligand (so you can select protein-protein binding sites as input)

-c1, c2

Chain identifiers of the compared proteins. You may give multiple chains, e.g., '-c1 ABC'.

-database

Used by the web server (with -surfdb and -compare options). It will output an .nosql file as usual, and additional .nosql file with inversed rotational matrices, whose lines are marked with asterisks.

-dist INTER_CHAIN_DIST

The distance between protein chains or between ligand and protein. Use with -bsite modifier or -mark and -results options.

-f1, f2

Compared proteins' files (.pdb or .srf).

-in INDIR

Directory where input files are.

-json JSON_FILE

Output json alignments to this file.

-local

Use this to perform local alignments search (with -compare or -surfdb options). By default, after the local alignment is found (with maximum clique algorithm), an attempt is made to extend this alignment along the backbones of the compared proteins. In this way, parts of proteins that adopt different conformations (e.g., loops) can be aligned (these aligned residues are marked with 'flx' in alignments.json).

-longnames

Use this to allow long file names. By default the protein names are trimmed down to 4 letters.

-motif MOTIF, -motif1 MOTIF, or -motif2 MOTIF

This selects residues to be used as a query instead of the whole protein structure (use with -extract or -compare options). This will generate a .srf file with only the selected residues. To select some residues on chains A and B of the input protein, use -motif "[:A and (14,57,69-71) or :B and (33,34,50)]". Note that chain Iids are case sensitive. Square brackets are mandatory!

-nobb

Do not include descriptors originating from backbone atoms.

-nomarkbb

Turns off the default action, which is to mark (not delete) backbone descriptors. In the first step of filtering, only non-backbone descriptors are used, while in the maximum clique step, all descriptors are used.

-noclus

Local structural alignments found (maximum cliques) are not clustered.

-nofp

Do not calculate fingerprint residues. Do not filter by fingerprint residues (use with -results option).

-noprune

Alignments are not pruned. By default bad scoring cliques are deleted (see SURF_VECTOR_ANGLE, BLOSUM_SCORE and CALPHA_RMSD parameters).

-nosql NOSQL_FILE

Output / read nosql alignments to / from this file.

-out OUTDIR

Directory to write output files to.

-param PAR_FILE

Read parameters from the specified parameter file.

-sfile SURF_FILE

Specify file that contains names of .srf files to be compared with the query protein (see -surfdb option). Each line must contain one .srf file-name.
Example:
protein1.srf A
protein2.srf B
protein3.srf A
etc.

-super

Find local structural alignments between two proteins (use with -compare option) and superimpose the two proteins according all alignments found. For each alignment, output the '.rota.pdb' file with the proteins superimposed according to this alignment.

-verbose

Output debugging information. Use when testing the program.

-z_score Z_SCORE

The cutoff value for z_score. Low z_score (<2) means that more insignificant alignments will be outputted (these can also occur by chance), higher z_score (>2) means only significant alignments will be outputted (use with -results option).

Output files

For explanation of the Json format see http://www.json.org/. Json is supported in all major programming languages.

info.json

Example:
{"z_score":1,"qpdb":"bs3","qcid":"A","biof":["bs3.cons.pdb"]}

Legend:

z_score	the cutoff z_score (see -z_score modifier); only alignments with their z_scores above this value are kept in 'alignments.json' file
qpdb	query pdb identifier (the first four letters of the input .pdb file) (see -longnames modifier)
qcid	query chain identifier
biof	names of the biological-assembly files of the query protein (if they exist)

query.json

Lists residues of the query protein and their degrees of structural conservation.

Example:
[
{"resi":1,"resn":"G","chain_id":"A","cons":0,"fp":0},
…
{"resi":129,"resn":"R","chain_id":"A","cons":4,"fp":0}
]

Legend:

resi	residue number
resn	residue name (one-letter notation)
chain_id	chain identifier
cons	degree of structural conservation between 0 and 9
fp	is it a fingerprint (highly conserved) residue? 0 - no, 1 - yes

alignments.json

Contains a list of aligned proteins sorted by their scores (sorting is done only by the z_score of alignment number 0 for each protein).

Example:


```
[
	{
		"pdb_id":"1eyv",
		"chain_id":"A",
		"nfp":0,
		"protein_name":"-",
		"alignment":
		[
			{
				"scores":
				{
					"alignment_no":0,
					"aligned_vertices":182,
					"e_value":5.45e-144,
					"rmsd":0.1,
					"sva":1.01,
					"z_score":4.74,
					"alignment_score":12.45
				},
				"rotation_matrix":
				[
					[1.00,-0.00,0.00],
					[-0.00,1.00,-0.00],
					[-0.00,0.00,1.00]
				],
				"translation_vector":[-0.00,-0.00,0.00],
				"aligned_residues":[
					{"resn1":"L","resi1":13,"chain_id1":"A","cl":"","resn2":"L","re
				s
					i2":13,"chain_id2":"A"},
					...
					{"resn1":"F","resi1":129,"chain_id1":"A","cl":"","resn2":"F","r
				e
					si2":14,"chain_id2":"A"}
				]
			},
			...
```

]
},
{ aligned protein #2 (similar as above) },
{ aligned protein #3 (similar as above) },
…
{ aligned protein #N (similar as above) }
]

Legend:

pdb_id	pdb id of the aligned protein
chain_id	chain id of the aligned protein
nfp	number of fingerprint (highly conserved) residues in all alignments of the aligned protein
protein_name	name of the protein (not used)
alignment	array of all alignments for one aligned protein
scores	object that holds all different scores for one alignment
alignment_no	alignment number (starts with 0!)
aligned_vertices	number of aligned graph vertices (the compared proteins are represented as vertices on the level of their functional groups)
e_value	expectation value for an alignment (the lower the better)
rmsd	RMSD of the aligned vertices (not atoms!)
sva	surface vector angle is the angle between the two normal vectors of the superimposed protein surfaces (lower the better)
z_score	Z-score for this alignment
alignment_score	a compound score calculated from sva, rmsd, e_value, and aligned_vertices
rotation_matrix	use rotation matrix (R) and translation vector (Tvec) to superimpose coordinates of query (X) to the template protein (Xrot): Xrot=R*X + Tvec. This will superimpose query -> template! For inverse superimposition (template->query) see FAQ below!
translation_vector	see rotation_matrix
aligned_residues	object that holds corresponding aligned residues of the two compared proteins
resn1, resn2	residue names of the aligned residues: 1 - first protein, 2 - second protein
resi1, resi2	residue numbers of the aligned residues: 1 - first protein, 2 - second protein
chain_id1, chain_id2	chain ids of the aligned residues: 1 - first protein, 2 - second protein
cl	if this is flexible alignment (flx), then this alignment was found by extension of local alignments along backbones (see -local modifier)

*.cons.pdb, *.bu#.cons.pdb

Coordinates of the query protein (or its biological units) with degrees of structural conservation in B-factors.

Degrees of struct. conservation [0-9]
|

```
                                                                  V
ATOM    164  N   ASP A  25      -5.417  22.585   4.112  1.00  0.40
ATOM    165  CA  ASP A  25      -4.752  21.401   3.553  1.00  0.40
```

*.rota.pdb

Using -align option

Superimposed coordinates of the query and compared proteins. Aligned residues are marked with ones in B-factors.

```
                                                  Aligned residues have 1.00 here
                                                                  |
                                                                  V
MODEL        1     1d1qB   ← Superimposed protein
…
ATOM   1417  N   CYS B  18       8.287  29.028  27.733  1.00  1.00
ATOM   1418  CA  CYS B  18       8.912  28.194  28.757  1.00  1.00
…
ENDMDL
MODEL        2     1phrA   ← Query protein (original coordinates)
…
ATOM     97  N   CYS A  17       8.075  28.921  27.850  1.00  1.00
ATOM     98  CA  CYS A  17       8.756  28.117  28.832  1.00  1.00
…
ENDMDL
END
```

Using -super modifier

```
REMARK PROBIS ALIGNED_VERTICES 91  ← Alignment scores
REMARK PROBIS E_VALUE 6.62986e-41
REMARK PROBIS RMSD 0.715179
REMARK PROBIS SVA 0.947438                Superimposed protein
REMARK PROBIS Z_SCORE 3.34832                     |
REMARK PROBIS ALIGNMENT_SCORE 9.37345             V
ATOM      1  N   SER A   0      -3.462  48.359  42.685  1.00 24.99
ATOM      2  CA  SER A   0      -2.739  47.051  42.825  1.00 24.45
```

FAQ

Should I use as input cut-out binding sites (fragments) or complete protein structures?

ProBiS does better if you take whole protein structures, and then define residues to be compared (binding sites, motifs) with the '-motif or '-bsite' modifiers (see examples 2, 4, and 5). If you still want to use fragments of '.pdb' files as input, you should add to the parameters.inp the following lines:

```
Z_SCORE -1.0 # allow insignificant alignments
SURF_VECTOR_ANGLE 4.0 # turns off checking for equal surface vector angles
```

Then use '-param' modifier at the command line, when you run probis -surfdb or -compare, e.g.:

```
$ ../probis -surfdb -param parameters.inp -f1 …..
```

Why .srf file still contains all protein atoms when I used -bsite or -motif to extract only on a binding site?

If using whole structures together with -motif, the .srf file will still hold the whole protein, but only the binding site residues will be used for comparison (they will be marked as surface). Use the -motif or -bsite modifiers on target structure or on both target and template structures.

How fast is generation of surfaces (.srf files) with -extract?

Generation of .srf files should be lightning-fast (also if -motif and -bsite modifiers are used). I regularly use it for database of few 10k proteins, which is completed in minutes.

Is it possible to run local alignment between two proteins. How can I use MOTIF in the parameter file to do this (align two binding sites)?

You can run local alignment between two protein binding sites, you should use the -motif (or -bsite) modifier together with --extract option (see also examples 2, 4, and 5), to create '.srf' files, that contain only selected surface residues :

For query protein, e.g.:

```
$../probis -extract -motif "[:A and (14,57,69-71) or :B and (33,34,50)]" -f1 1phr.pdb -c1 A > 1phrA.srf
```

Similar for the compared protein(s):

```
$ for i in `cat proteins.txt`; do ../probis -extract  -motif "some valid selection of residues" -f1 ${i:0:4}.pdb -c1 ${i:4:1} > $i.srf;done
```

Don't forget to use "" around the residue selection!

I already have 1phrA.nosql file. However I don't understand how I can get the P-value or significance of match between two structures?

You can get E-value or Z-score (the last shows you the significance of match) by reading the '.nosql' file with:

$../probis -results -param parameters.inp -f1 1phr.pdb -c1 A

This will generate a '.json' file, in which e_value and z_score will be for each alignment generated (there can be more alignments, identified by alignment_no, per each pair of compared proteins).

I get two json files info.json & query.json. I dont know much about json format, so I don't understand query.json. But cat info.json gives me this {"z_score":1, "qpdb":"1phr", "qcid":"A", "biof": ["1phr.cons.pdb", "1phr.bu1.cons.pdb"]}. Does this means that 1phr has a Z-score = 1 for only self, so it does not matches with anything in proteins.txt?

In 'info.json', the line {"z_score":1, "qpdb":"1phr", "qcid":"A", "biof": ["1phr.cons.pdb", "1phr.bu1.cons.pdb"]} means that the cutoff z_score is set to 1. All alignments with z_score < 1.0 will be deleted. You can change this (and other parameters) in parameters.inp file: to tighten the filter, just add line Z_SCORE 2.0.

File 'query.json' just gives degress of conservation for every residue of the query protein, e.g., {"resi":1,"resn":"A","chain_id":"A","cons":0,"fp":0}, means that residue number 1, chain A, has a conservation of 0 (could be [0-9]).

I am trying to run probis on a proteome scale. So a lot of the proteins are models, with UniProt ID but it seems the program reads only first 4 letters of protein name. For example when I run 'mpiexec -host fp167 -np 16 ./probis -surfdb -sfile srfs.txt -f1 P0A626_bs1.srf -c1 A' it generates a nosql file P0A6A.nosql. So I am not sure whether it will correctly read my srfs.txt which contain many modeled structures having long names?

In "molecule.cc" the program trims protein names to 4 characters. This is an old legacy behavior required by the web server... If you want to use long names, just use '-longnames' modifier.

How to specify the input or output directory.

You can try to use '-indir' and '-outdir' modifiers. If these don't work (they are still experimental), you can try this:

Output:
put the directory where "probis" program is (e.g. /usr/local/bin) in the path (in your .bashrc file PATH=$PATH:/usr/local/bin), make a directory where you want your output files to be written, 'cd' to that directory, and run probis from there. All output files will appear in that directory.

Input: if your .srf files are in e.g. '/tmp' directory, use full
path in srfs.txt
for example:

/tmp/P71662_bs1.srf A
/tmp/P71662_bs2.srf A

/tmp/O69689_bs1.srf A
/tmp/O69689_bs3.srf A
/tmp/O69689_bs4.srf A

Also, use full path on the command line for example:

```
$ probis -extract -f1 /pdbbank/P0A626_bs1.pdb -c1 A > /tmp/P0A626_bs1.srf
$ mpiexec -v -hostfile /full-path/hosts -np 8 probis -surfdb -param /full-
path/parameters.inp --sfile /full-path/srfs.txt -f1 P0A626_bs1.srf -c1 A
```

This will generate P0A626_bs1A.nosql.

```
$ probis -results --param /full-path/parameters.inp -f1 /pdbbank/P0A626_bs1.pdb -c1 A
```

Can you tell me about the format of alignment.json so that I can parse the output. How do you parse the e-values for your analysis from this file?

The easiest way to parse this file might be to use some existing JSON parser (e.g., http://docs.python.org/library/json.html - you can get json parser for almost every programming language), because this allows you to see structure of the data in the file. You can then access data similar to the following:

print simprot[2].alignment[0].scores.z_score

In every .json file [] is an array and {} is an object. File 'alignments.json' is organized as an array of similar proteins, where the highest scoring (according to z_score in alignment_no:0) are first: [simprot1, simprot2, simprot3, ...] then each protein is an object: simprot = {"pdb_id":"1all", ... "alignment":[ali0, ali1, ...]} each simprot can have up to five alignments which are array: alignment = [ali0,ali1, ...] etc. the relevant score for you is probably z_score or e_value.

I was able to figure all the things out and now I can read the JSON files and also find similar binding site in most cases. However in some cases my program dies saying incorrect format of JSON file Perl error -----> malformed JSON string, neither array, object, number, string or atom, at character offset 179 (before "inf, "alignment_score..."). The problem is in alignments.json. Is it a perl problem ?

In 'alignments.json' file, try replacing "inf" with some negative value (-99.0). The real problem is probably in your procedure (check!).

Can I use a protein-protein binding site as input?

Yes. See usage of '-bsite' modifier in this guide.

I am trying to superpose the template's ligand into the query protein based on the rotation and translation matrix that I get from

alignments.json. However when I try to superpose the ligand using the transformation matrices, it never lands up inside the predicted binding pocket.

The problem is that you are trying to rotate in the wrong direction, that is, template.pdb -> query.pdb... However, the current matrix in alignments.json codes for rotation query.pdb -> template.pdb ! One way to solve this (using alignments.json file) is to do the following:
Considering that this is your current equation:

> v' = R*v+t

where v is the vector with coordinates and v' are the rotated coordinates (R and t are rot. matrix and vector, respectively).
However, if you want to rotate in the opposite direction, i.e., v' -> v, then to calculate v, you have to multiply equation by R^-1:

> R^-1*v'=R^-1*R*v + R^-1*t
then
> v = R^-1*v' - R^-1*t = R^-1*(v'-t)

(R^-1 equals Rtransposed; R^-1*R = 1)

then to get coordinates v (template->query) you have to use:

> v = Rtransposed*(v'-t)

This was the hard way.. It's much easier to use '.nosql' file instead of '.json', and let probis program do all the math for you, like this:

> probis -align -alno 0 -f1 query.pdb -c1 A -f2 template.pdb -c2 A

This will read the alignment number 0 in the query.nosql file, superimpose template.pdb -> query.pdb, and write the resulting superimposition to "query_tempA.0.rota.pdb"!