ProACT2 Manual

Tjelvar Olsson & Mark Williams
March, 2010

Contents

Introduction

ProACT2 (Protein Accessibilities, CaviTies and ConTacts) is a program for structural analysis of proteins, and protein-ligand and protein-protein complexes. Analysis is in terms of solvent accessibility and explicit hydration sites, cavities and polar and apolar contacts within and between the molecules.

ProACT2 is a reimplemented, refined and considerably functionally enhanced Python version of the earlier Fortran program ProACT for the analysis of solvation and cavities in proteins.

Rationale

The main ideas underlying the ProACT2 methodology were first described in

"Buried waters and internal cavities in monomeric proteins"
Williams MA, Goodfellow JM & Thornton JM (1994) Protein Science 3, 1124-1135.

Namely, that distances between atoms in proteins are strongly peaked for a small range of short distances. This non-random distance distribution is attributable to repulsion at very short distances and an attractive interaction that is physically dominant at slightly longer distances. Such distance distributions also show two other features: that the distances over which there is a dominant attractive effect vary depending on the chemical nature (types) of the atoms investigated and that atoms in polar chemical groups (e.g. C=O) show two distinct peaks in their distance distributions with the peak at shorter distances attributable to an enhancement of the attractive interaction in certain local environments by hydrogen-bonding (Coulombic) forces and that at longer distances being where only the van der Waals dispersion is important. Being polar, water molecules around proteins exhibit this double peaked behaviour.

These observations are taken to have several consequences:

Contact can be defined in an empirically derived, physically significant way as the range of distances over which a non-random atom-atom distance distribution is observed.

Contacts in which Coulombic interactions are likely to be significant (polar) can be distinguished from those which are likely to be dominated by van der Waals interaction (apolar) on the basis of distance.

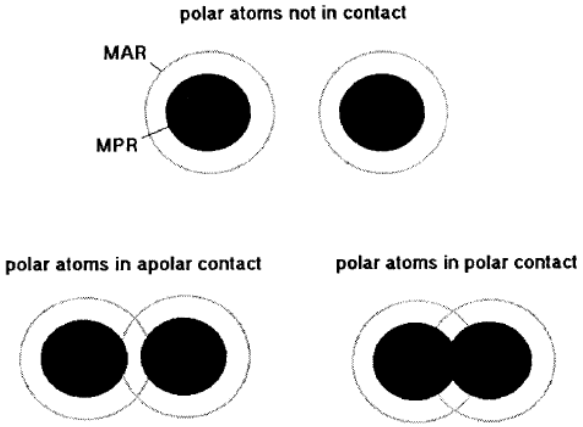
That the traditional approach to determining a solvent accessible surface through 'rolling' a 1.4Å sphere over the surface of spherical protein atoms with fixed radii gives an inaccurate view of likely locations of hydrating

water molecules and that a more realistic model can be provided by varying the effective size of water dependent on local chemical environment.

Atomic radii, contact, cavities and solvation

In a similar way to that in which van der Waals radii are usually derived, analysis of atom-atom distance distributions of atoms of different type, together with an assumption of additive radii, allows us to represent the contact distance criteria as sums of atomic radii.

- Each apolar atom-type is represented by three radii related to the minimum, maximum and most probable (characteristic) observed separation of atoms.
- The furthest separation which can be considered to be a contact is the sum of the atoms' maximum apolar radii.
- Apolar atoms are most likely to be found at a distance equal to the sum of their characteristic apolar radii
- Atoms should not approach closer than the sum of their minimum apolar radii
- Cavities occur in regions of space which lie between atoms but outside the surfaces defined by their maximum atomic radii.
- Cavities can be represented by pseudo-atoms which grow until they 'touch' the surrounding atoms at the surface defined by their characteristic apolar radii. Consequently contacts of atoms with cavities can be identified in a similar way to contacts between atoms.
- Polar atoms (including water) have two groups of radii one describing their behaviour in interaction with apolar atoms and one where the Coulombic interactions are important. Polar atoms are effectively smaller when involved in a hydrogen-bond or Coulombic interaction.



Polar atoms separated by less than the sum of their maximum polar radii (MPR) are in polar contact. If separation is greater than this, but less than the sum of their maximum apolar radii (MAR), they are in apolar contact.

Atomic Radii Sets

The original publication contained six atom types: C, S, O, N, W(ater), P(everything else) with O, N and W able to undergo polar interaction and having two groups of radii. Our set of radii appeared to perform reasonably well when analyses were restricted to protein hydration and cavities. These radii may be used by specifying the *–radii WILLIAMS* option on the command line.

Subsequently, Li & Nussinov carried out a more systematic and extensive analysis of atom-atom distance distributions including a new method for removing the influence of secondary contacts on the distributions (i.e. not including a particular atom-atom distance in the distribution when there is a shorter distance between atoms to which they are directly covalently bonded). The generalized **Li & Nussinov radii are default** for ProACT2.

"A set of van der Waals and coulombic radii of protein atoms for molecular and solvent accessible surface calculation, packing evaluation and docking."
Li AJ & Nussinov R (1998) Proteins 32, 111–127.

Installation

Dependencies

Download proACT2.tgz from http://people.cryst.bbk.ac.uk/~ubcg66a/proact2_summary.html

In order for ProACT2 to work you need:

1. Python 2.4, 2.5 or 2.6 (http://www.python.org)
2. NumPy 1.2 or greater (http://numpy.scipy.org)
2. BioPython 1.50 or greater (http://biopython.org)

Default installation of these packages requires root (Linux & Mac OSX) or Administrator (Windows) privileges. Windows installers are available. An appropriate version of Python is very likely to be available by default on recent Linux and Mac OSX systems and the NumPy and BioPython modules can be also installed in user directories provided appropriate options are specified in the installation commands and appropriate paths set. Please read the installation instructions for these packages carefully.

With 32-bit installations of the above packages, ProACT2 will run quicker if you also have Psyco installed (http://psyco.sourceforge.net/).

Creation of contact maps requires installation of PyX (http://pyx.sourceforge.net)

Linux & Mac OSX

```
tar xvfz proACT2.tgz
cd proACT2
python test.py
```

The last command tests if ProACT2 and dependencies are installed correctly then this will report "*OK*"". Otherwise, some error message will be generated. Most problems are likely to arise from incorrect installation of BioPython or NumPy and this should be clear from the message.

Windows

Extract the contents of proACT2.tgz with WinRAR other other suitable software to a directory e.g. C:\Program Files\ProACT2

At the command prompt, check that the path to python is known with

```
echo %PATH%
```

if not, add location of your python installation with something similar to

```
set PATH=%PATH%;C:\Program Files\Python2.5.1
```

Add location of ProACT2

```
set PYTHONPATH=%PYTHONPATH%;C\Program Files\ProACT2
```

run test.py from your python environment

Running ProACT2

ProACT2 is run from the command line with a variety of option flags. The minimal command

```
python proACT2.py example.pdb
```

acts on a PDB format protein structure file (identified by .pdb or .ent suffix) and will give six outputs.

1. example_summary.txt

```
Number of protein 1 atoms              : 1001
Number of protein 2 atoms              : 0
Number of ligand atoms                 : 0
Number of total contacts               : 4104.0
Number of polar contacts               : 438.0
Number of apolar contacts              : 3666.0
Number of polar water contacts         : 445.0
Number of apolar water contacts        : 821.0
Number of polar surface water contacts : 407.0
Number of apolar surface water contacts: 710.0
Number of polar cleft water contacts   : 33.0
Number of apolar cleft water contacts  : 80.0
Number of polar buried water contacts  : 5.0
Number of apolar buried water contacts : 31.0
Number of polar cavity contacts        : 47.0
Number of apolar cavity contacts       : 141.0
Probe volume                           : 238.368
Surface waters                         : 498
Cleft waters                           : 16
Buried waters                          : 4
Further waters                         : 0
```

2. example_access.pdb – pdb format file annotated with atomic solvent accessibilities
3. example_waters.pdb – computationally modelled water molecules
4. example_cavity.pdb – cavity defining pseudoatoms
5. example_access.rsa – residue by residue apolar and polar accessibility information
6. example_residue_contacts.csv – file containing list of residue to residue/water/cavity contacts

A log file is also generated and summary information is copied to all_runs.csv. This latter file is never overwritten which is helpful when running a series of calculations.

Residue and chain selection
The default behaviour is to carry out calculations on all structures in the PDB file. Chains and residues from the PDB file may be specifically selected e.g. residues 1 to 83 from chain A.

```
python proACT2.py example.pdb –-chain_sel1 A –-res_sel1 1:83
```

In cases of multiple conformations of protein residues, the conformations with highest occupancy are extracted. Some care should be taken when interpreting results if structural information for some atoms or residues are not given in the PDB file due to disorder.

E**xamples**
Several examples of how to run proACT2 and associated output can be found in the examples subdirectories (the command line in each case is given in the run_me.sh file).

binding_site – calculations restricted to ligand binding site alone
complex – contacts and experimental water categorization on protein-ligand complex
ligand – hydration and accessibility of a free ligand
protein – calculations on a protein
protein_protein – the interface between two protein chains or other subset of residues

Help

For more help run the command: `python proACT2.py -h`

Solvent Accessibility

Atomic accessible surface areas are output as default in the *example_access.pdb* file replacing the B-factor values. In order to turn this off, use the option *--no_access* An additional output file *example_access.rsa* summarises accessibilities on a residue by residue basis. In the case of complexes (see later section), this can optionally include information on the apo and bound states or on changes upon complex formation.

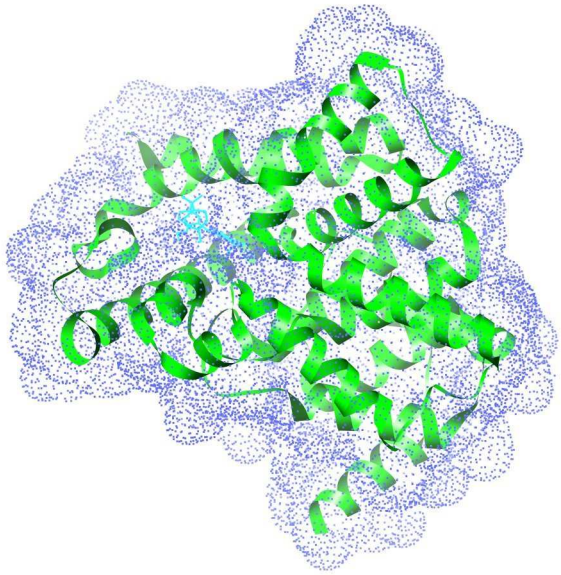
example_access.rsa

```
REM SOLVENT ACCESSIBILITY CALCULATED BY PROACT2
REM RESIDUE SOLVENT ACCESSIBILITY
REM RES _  NUM       TOT    APOLAR     POLAR
RES GLU A  534    137.38     34.26    103.12
RES GLU A  535    117.24     57.30     59.94
RES GLU A  536     69.17     20.90     48.27
RES THR A  537     71.46     45.81     25.64
RES ARG A  538    146.85     56.80     90.04
RES GLU A  539     57.96     24.77     33.19
RES LEU A  540     24.20     24.20      0.00
RES GLN A  541    103.25     24.42     78.84
RES SER A  542     36.85     15.73     21.12
RES LEU A  543      0.75      0.00      0.75
RES ALA A  544     30.38     16.80     13.58
RES ALA A  545     91.46     63.55     27.91
RES ALA A  546     28.44     19.39      9.05
RES VAL A  547    137.63    133.71      3.92
RES VAL A  548     41.26     22.91     18.35
RES PRO A  549     31.02     31.02      0.00
RES SER A  550     53.98     40.86     13.12
RES ALA A  551     13.72     10.56      3.17
```

N.B. As with any surfacing method these values are intended for use in structure description and interpretation, and like-for-like comparison. You cannot sensibly use these surface area values as input to empirical formulae for stability etc. where those formulae were derived using another solvent accessible surface methodology or different radii set.

Solvent Accessible Surface

The *--surface* option outputs an *example_surface.dms* file in DMS (dot molecular surface) format, suitable for direct import into a variety of visualization packages (e.g. UCSF Chimera). This surface corresponds exactly to the surface area calculations above.



Hydration of Protein Exterior Surface and Cavities

Having determined solvent accessible surface points, water molecules are placed at these points filling the protein surface. In ProACT2, water molecules are successively added at the position at which they can make the most polar interactions. Because previously placed water molecules can themselves participate in polar interactions, the priority list is iteratively revised during insertion – this slows performance of the analysis, but gives better water positions in comparison to experiment (T.S.G. Olsson – PhD thesis, University of London, 2007).

ProACT2 allows the addition of one or two layers of water e.g. *--layers 2* (default 1). Two layers of water are very often beneficial in investigation of concave ligand binding sites.

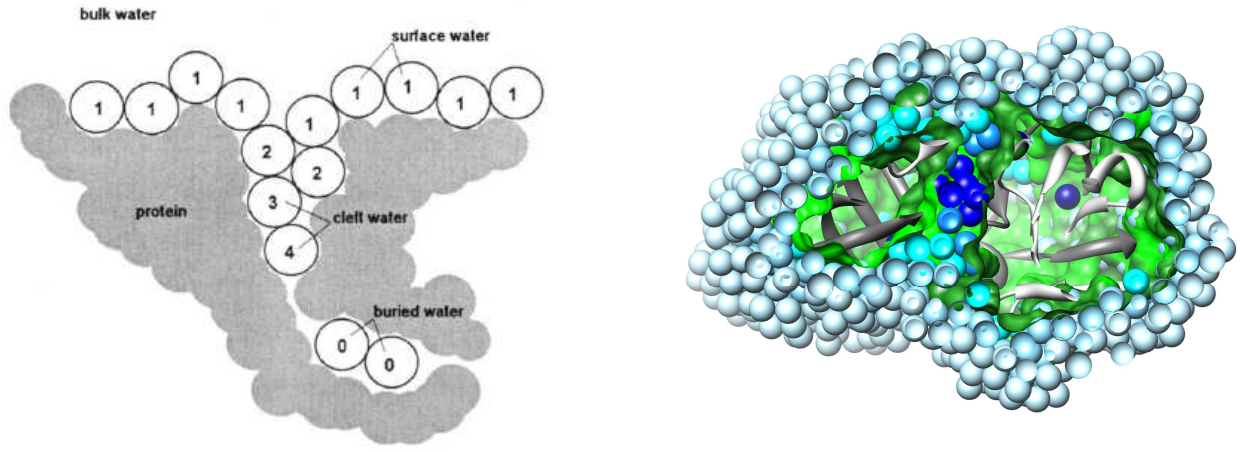
The hydration procedure is not deterministic as there may be many positions at which a particular number of polar contacts can be made. ProACT2 can automatically repeat the hydration process several times so that a statistical view can be taken of the consistency of hydration of any site e.g. *--runs 10* (default 1).

Surface, Cleft and Buried Water Molecules

A key topic of the original publication (Willliams, 1994) was the categorization of experimental surface, cleft and buried waters, following the scheme below

```
python proACT2.py --experimental_waters example.pdb
```

Both experimental and modelled waters are output to *example_waters.pdb*, experimental waters are given a b-factor of 1.00 and modelled waters a b-factor of 0.00. Categorization is denoted in the chain id column of the pdb format output further (F), surface (1), cleft (2,3,4 etc.), buried (0). For other water related output see also example_waters.kin.



Left: Schematic view of water molecule categorization ('further' water molecules lie beyond the surface water layer) Right: Two layers of hydration are required to fully hydrate the peptide binding cleft in HIV protease (waters coloured according to category)

It is recommended to perform repeated hydration cycles *(e.g. --runs=10*) in order to categorized experimental water molecules. In critical cases categorization of experimental waters (surface, buried, cleft) should be carried out using information from all repeats and the most conservative categorization used e.g. waters should be though of as buried only if they are buried in all repeats. Following this conservative principle, it is also recommended, and perhaps even more important, to use two solvent layers for categorization purposes e.g..

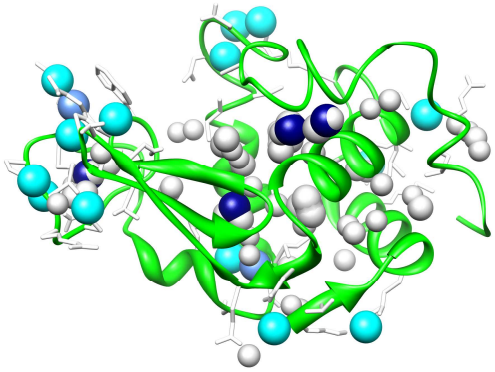
```
python proACT2.py example.pdb --experimental_waters --layers=2 --runs=10
```

N.B. The solvent accessible suface option *--surface* negates the *--experimental_waters* option.

Cavities

Cavities are defined by spherical pseudo-atoms whose centres lie in regions of space outside the surfaces defined by maximum apolar radii of protein and ligand atoms and surface or cleft water molecules.

The cavity pseudo-atoms grow from their centre until their surface touches the surface of surrounding atoms and water molecules defined by their characteristic apolar (van der waals) radius. A similar method has recently been implemented in RosettaScore. In ProACT2, these pseudo-atoms may overlap by a distance of up to half their radius. This gives a closer approximation to the cavity shape.



Cavities in hen egg white lysozyme as white spheres. Blue spheres are experimental cleft and buried (darkest blue) water molecules.

Contacts

A list of the apolar and polar contacts made between protein residues and with ligands, water molecules and cavity pseudoatoms is output in CSV format suitable for import in to common spreadsheet programs. Apolar and polar contacts are not double counted and polar contacts take precedence

```
"resid1","resname1","chain1","resid2","resname2","chain2","polar contacts","apolar contacts"
612,"TYR","A",613,"HIS","A",0,6
612,"TYR","A",617,"HIS","A",1,1
612,"TYR","A",782,"VAL","A",0,2
612,"TYR","A",2,"HOH","1",1,0
612,"TYR","A",13,"PPP","P",0,1
613,"HIS","A",612,"TYR","A",0,1
613,"HIS","A",5,"HOH","1",1,0
617,"HIS","A",612,"TYR","A",0,2
617,"HIS","A",621,"THR","A",0,1
617,"HIS","A",764,"ASP","A",0,2
661,"SER","A",660,"VAL","A",0,1
661,"SER","A",662,"ASN","A",0,1
661,"SER","A",10,"HOH","1",0,1
661,"SER","A",11,"HOH","2",0,1
725,"LEU","A",724,"ASP","A",0,1
725,"LEU","A",726,"ALA","A",0,1
725,"LEU","A",9999,"LIG","Z",0,1
```

The CSV file contains summary information at the ‘residue’ level i.e. in the above example output Tyr612 chain A makes 6 apolar contacts with His613 Chain A, Tyr613 makes a single contact to a cavity pseudo atom PPP13, His613 makes 1 polar contact with water molecule 5, which is at the surface (chain id 1). Usually this would be all of the information required. However, contacts are also listed atom-by-atom when using the Kinemage visualization options for interface analysis (see later section).

Contact Maps

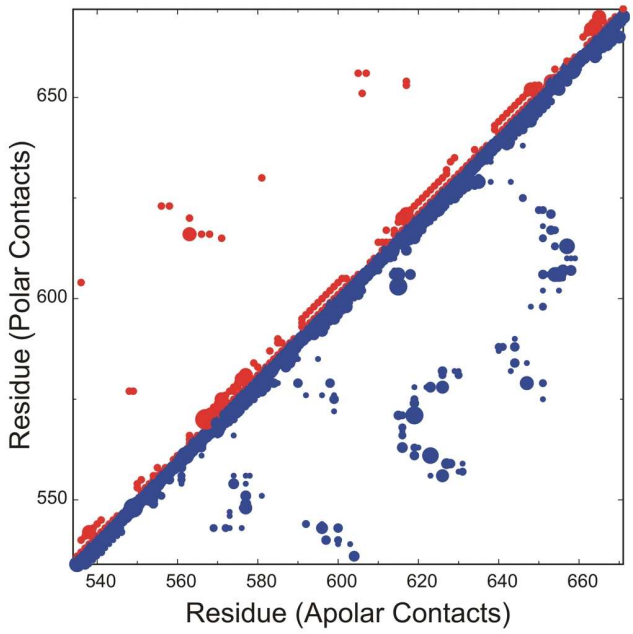
A utility is provided to generate Postscript and PDF (default) figures representing the information held in the contact CSV file. All contacts are plotted with

```
python contact_plot.py example.csv
```

For intermolecular contacts it makes more sense to select subsets of residues by chain

```
python contact_plot.py example.csv -x A-1:30 –y B-26:200
```

i.e. plot contacts between chain A residues 1:30 (on x-axis) and chain B residues 26:200 (on y-axis).
The residue-residue/water/cavity contacts are displayed as circles the area of which depends linearly upon the number of contacts. Given that there are many more apolar atoms and thus contacts than polar, the areas for each type are scaled differently.



Additional details on options can be obtained with

```
python contact_plot.py -h
```

Protein-Ligand and Protein-Protein Complexes

ProACT2 has several new features for analysing the interaction interfaces (binding-sites) of complexes.

An interface is defined as comprising those residues from different molecules whose constituent atoms are in contact or which could contact a common water molecule or cavity defining pseudo-atom.

In identifying protein residues as belonging to an interface, test water molecules are placed at all ligand solvent accessible surface positions i.e. all possible positions for a bridging water molecule to occur are evaluated (these test waters are output in the kinemage file binding_site_waters.kin).

Interactions with small-molecule ligands

```
python proACT2.py --only_binding_site example.pdb example.mol2
```

makes calculations on the complex including only the ligand, solvating water and protein residues in direct or indirect (bridged) contact. Protein residues involved in the binding site are given in the left-hand column of *_contacts.csv file.

Small molecule ligands must be represented as mol2 format files. This is a slight inconvenience for those used only to dealing with PDB format structures, but fits better with the workflows of most small molecule interaction studies e.g. in drug design and docking. mol2 format can be created from PDB by simply using a text editor or script to extract the relevant HETATM records from the PDB file and converting them using babel (http://openbabel.org/wiki/Main_Page)

```
babel -ipdb -omol2 ligand.pdb
```

(N.B. This mol2 file will not necessarily contain correct information on bond-types (single double etc.) since this is not recorded in the originating PDB data, but that is not important for calculations carried out by ProACT2. It may be necessary to use more specialist tools, e.g. Sybyl, ICM, if you also want to use the mol2 file for other purposes.)

```
python proACT2.py --only_binding_site --complex_formation example.pdb example.mol2
```

outputs results of calculations for the complex, the protein and ligand separately, and reports on changes in interactions and hydration upon complex formation in the summary file.

Interactions between proteins

```
python proACT2.py --only_interface exampleA.pdb exampleB.pdb
```

reports on the interactions between the protein in the first pdb file and that from the second. This presumes that relevant components of a complex has been manually separated into different pdb files. Alternatively, chains can be select from the same file

```
python proACT2.py --only_interface --chain_sel1 A --chain_sel2 B example.pdb example.pdb
```

Again the *--complex_formation* flag will lead to calculations on the separate chains in addition to the complex.

Kinemage visualization

Kinemage scripts are output with the *--kinemage* option.

ProACT2 generates four kinemage files:

1. *_accessibility.kin (displays solvent and probe accessibilities)
2. *_contacts.kin (displays polar and apolar contacts)
3. *_probes.kin (displays voids in structures)
4. *_waters.kin (displays waters in and around structures)

The kinemage (kinetic image) file format allows the build up of 3D diagrams in a simple and comprehensible fashion. It is essentially a mark up language for creating 3D displays (in the same fashion html is a mark up language for creating web pages).

There are two 3D visualisers available for displaying the content of kinemage files: Mage (http://kinemage.biochem.duke.edu/software/mage.php) and the more recent King (http://kinemage.biochem.duke.edu/software/king.php).

A basic kinemage file displaying some points would look like:

```
@kinemage

@dotlist {some_points}
{} 0.0 0.0 0.0
{} 1.0 0.0 0.0
{} 0.0 1.0 0.0
{} 0.0 0.0 1.0
```

The markup text @dotlist represents the type of 3D primitive the coordinates should be represented as, in this case as dots or points. The text in the curly brackets (directly after @dotlist) gives a group name to the set of points defined below it. Alternative representations for visualising coordinates in space are @balllist ("psedo" spheres) and @spherelist ("real" spheres). Lines can be represented using the 3D primitive @vectorlist.

Building on the previous example:

```
@kinemage

@dotlist {some_points}
{origin} 0.0 0.0 0.0
{x} 1.0 0.0 0.0
{y} 0.0 1.0 0.0
{x} 0.0 0.0 1.0

@vectorlist {axis}
{} P 0.0 0.0 0.0 {} 1.0 0.0 0.0
{} P 0.0 0.0 0.0 {} 0.0 1.0 0.0
{} P 0.0 0.0 0.0 {} 0.0 0.0 1.0
```

Notice that we have given the individual points in the @dotlist names in curly brackets (origin, x, y and z). These names will now be displayed when clicking on the points in either Mage or King. Also notice that for each pair of points making up a line the first set of coordinates is preceded by the letter P. This is so that each pair of points will draw one distinct line. If the P was omitted a single line would simply be drawn from point to point until the keyword P was encountered. For example to draw a line from x to y to z and back to x again the following would suffice:

```
@vectorlist {single_line} color= red
{} P 1.0 0.0 0.0
{} 0.0 1.0 0.0
{} 0.0 0.0 1.0
{} 1.0 0.0 0.0
```

In the example above the line has been given the colour red.

More information on the kinemage file format can be obtained from the official documentation (http://kinemage.biochem.duke.edu/php/downlode.php?filename=/downloads/PDFs/format-kinemage.pdf).

In terms of converting PDB files to kinemage representations there are two tools available: molikin (accessible from within King) and prekin (http://kinemage.biochem.duke.edu/software/prekin.php). It is highly recommended to use either of these two programs to generate representations of the protein for viewing in tandem with the files genereated by ProACT2. Note: to view multiple kinemages at the same time in Mage or King use "File>>Append Kin File..." rather than "File>>Open New Kin File...". The latter overwrites any previously loaded kinemages.

The ProACT2 kinemage files:

1. *_accessibility.kin
can be used to display solvent accessible surface (SAS), both polar (PolSAS) and apolar (ApolSAS). In order to conveniently associate surface points with their corresponding atoms both dots representing the actual surface points and lines connecting the surface points with its atom are available. Probe accessible surface (PSAS) and inaccessible surface (PSIAS) can also be displayed. These latter surfaces are split into groups defined by integers. The integer group definitions are only of interest to people who want to look at particular aspects of the surface classification algorithms.

2. *_contacts.kin
contains information on all contacts in the form of vectors between the contacting atoms/waters/voids as well as points representing the point in space midway in between the centres of the two intereacting atoms/waters/voids. Below are some lines from a *_contacts.kin file:

```
@text
This file contains vectors representing:
1) Polar contacts
2) Apolar contacts
3) Polar water contacts
4) Apolar water contacts
5) Polar void contacts
6) Apolar void contacts
as well as dots representing the midpoint of contacts.

@kinemage
@vectorlist {Polar Contacts} color=red width=3
{ atmName= O   atmType=O resId=735 resType=PHE chain=A } P -43.5340 36.3370
63.5630{ atmName= N   atmType=N resId=739 resType=ILE chain=A } -45.8590
37.6750 62.0330
{ atmName= O   atmType=O resId=643 resType=ILE chain=A } P -30.9350 49.2930
75.6670{ atmName= N   atmType=N resId=647 resType=LEU chain=A } -28.9470
47.0870 75.2210
...
@dotlist {Polar Contacts} color=red width=3
{} -44.6965 37.0060 62.7980
{} -29.9410 48.1900 75.4440
...
```

3. *_probes.kin
contains information on voids in the structure. In this file the radius of the probes is easily accessible both from the name representing the individual probes but also from the radius (r) given at the end of each coordinate line.

```
@text
This file contains the cavity-defining pseudo-atoms.

@kinemage
@spherelist {Probes} color=purple
{ 1 r=1.7106 } -35.0240 33.2173 71.8311 r=1.7106
{ 2 r=1.6532 } -19.0322 47.6213 77.4679 r=1.6532
...
```

4. *_waters.kin
contains information on the waters in the structure.The waters are split into four groups: buried, surface, cleft and further. Note that the name representing the individual water molecules contains information on: the

water type (watType=), whether the water was taken from the pdb file or not (experimental=True/False) and in which layer of hydration the water was added (hydLay=).

```
@text
This file contains the water atoms. Waters can be:
Buried  - 0 or -1
Cleft   - 2, 3, ..., n
Surface - 1
Further - -1, -2, ..., -n

@kinemage
@spherelist {Buried} color=gold radius=1.37
{1 hydLay=1 watType=0 experimental=False} -32.7298 31.3388 57.7065
{85 hydLay=1 watType=0 experimental=False} -33.6900 42.9858 58.4144
...
@spherelist {Surface} color=pinktint radius=1.37
{4 hydLay=1 watType=1 experimental=False} -40.6547 35.2514 80.1172
{6 hydLay=1 watType=1 experimental=False} -30.4076 29.7910 46.3844
...
@spherelist {Cleft} color=peachtint radius=1.37
{2 hydLay=1 watType=3 experimental=False} -26.5654 54.6840 87.6737
{3 hydLay=1 watType=2 experimental=False} -15.5580 15.6760 60.4820
...
@spherelist {Further} color=lilactint radius=1.37
```

Licence

This software is distributed under a Creative Commons licence which allows non-commercial use and redistribution of the software or a modified version of the software provided that authorship of the software is attributed to the Authors below and a link or reference to the original version of the software on the Authors website is prominently displayed. Any modified software may only be distributed under the same or similar license to this one.

Full licence terms are available at

http://creativecommons.org/licenses/by-nc-sa/3.0/

Warranties and Liabilities
Unless otherwise mutually agreed to by the parties in writing and to the fullest extent permitted by applicable law, licensor offers the work as-is and makes no representations or warranties of any kind concerning the work, express, implied, statutory or otherwise, including, without limitation, warranties of title, merchantability, fitness for a particular purpose, noninfringement, or the absence of latent or other defects, accuracy, or the presence of absence of errors, whether or not discoverable.

Except to the extent required by applicable law, in no event will licensor be liable to you on any legal theory for any special, incidental, consequential, punitive or exemplary damages arising out of this license or the use of the work, even if licensor has been advised of the possibility of such damages.

Authors

Tjelvar S.G Olsson	main program - algorithms, coding & design
Antony Churchill	contact maps - coding & design
Mark A. Williams	algorithms & design

Substantial parts of the code are derived from the previous Fortran program PRO_ACT

Julia M. Goodfellow	algorithms & design
Janet M. Thornton	algorithms & design
Mark A. Williams	algorithms, coding & design

Help & Comments

m.williams@mail.cryst.bbk.ac.uk