consgen manual

Author: James D. R. Knight*

May 27, 2006

*email jaknight@ohri.ca.

Contents

This is a manual on how to use the program consgen. If you have any questions not covered here, any comments or suggestions, feel free to email me at jaknight@ohri.ca.

1 Installation

A precompiled program is supplied (AMD64) and may work on your system. If not, or if you would rather compile a system dependent version, follow these instructions.

1.1 Linux

1. Gunzip and untar the downloaded file

   % tar xvzf consgen-1.0.tar.gz

2. Move into the directory

3. Open the Makefile and specify your C compiler in the CXX macro, and your Fortran compiler in the FXX macro.

4. Type make to compile the program:

   % make

1.2 Mac and Windows

I have worked on these platforms before when programming but no longer do so. Mac should compile much like in Linux. For Windows, try cygwin.

2 Usage

2.1 Input

The program requires a list of PDB files for use and the PDB files themselves. The directory you call the program from must contain the list and the files. It also must contain a parameter file (the default can be found in the source directory called userdefs.data - more below).

The list must be formatted so:

```
1ABC.pdb X
2DEF.pdb Y
3GHI.pdb Z
4JKL.pdb -
```

The first entry is the file name, and the second entry is the chain to be used. If there is no chain identifier, enter a dash.

2.2 Example

A set of .pdb files, along with the list, are included in the Example directory.

2.3 Command line options

The program is run as follows, with optional flags -s and -p:

```
% consgen -l [filename]
```

-l the PDB list is specified with this flag

-s will force the program to perform full pairwise initial alignments to find the optimal order for generating an initial consensus. It is usually not necessary and takes somewhat longer to perform. If it is expected there may be a great deal of difference between samples, this option should be performed. This can also be done after running the program without the flag to see if there is any significant difference.

-p can be used to specify a parameter file (see below). The program automatically searches for a file called userdefs.data in the runtime directory if a parameter file is not specified.

2.4 Input options

There are a number of parameters that can be changed. This is done through the userdefs.data file. This is a default file which can be found in the directory consgen-1.0. It must be available in the runtime directory unless its location is specified on the command line. This file looks as follows:

```
crn 50
rset
trs1 0.01
trs2 2.0
ii 20
im 0.33333
cm 0.5
noi 20
lmkf1 0.70
lmkf2 0.85
lmkf3 1.00
redtrs 0.5
lfree 0.3
trs3 2.0
```

The default values are specified. If an entry is missing in this file the program will use a default.

crn core residue number: when searching for an initial pairwise alignment between two samples, or between a sample and the current consensus, the program looks for highly congruent triangles with matching amino acid vertices within a core or residues. This minimizes memory usage and run time. The ideal area to focus on is around a crucial structural feature, such as an active site. The program, by default searches a core of 50 residues at the heart of the structure. The center is defined as the residue closest to all others, and the 49 closest to it make up the core. To use more residues, change the value of crn. If you wish to use the entire structure, enter 0. The core can be manually defined by specifying a residue (see below).

rset rset defines any categories to be used in the initial pairwise alignments. By default the program only use congruent triangles with matching amino acid vertices and does not consider matching categories, such as hydrophobic. To force the program to consider categories, enter the category number after rset, each separated by a whitespace (ex: rset 21 22 26 29). The categories are as follows: phosphorylated (21), polar (22), hydrophobic (23), small (24), charged (25), aliphatic (26), aromatic (27), basic (28), very small (29) and acidic (30). Using the polar, hydrophobic, small and charged categories will increase demands upon memory since there are a large number of residues in these categories.

trs1 threshold 1: this is the congruency criteria. To be congruent, the sides of the triangles used in generating initial pairwise alignments must be within trs1$\times$100% of each other.

trs2 threshold 2: if two residues are within trs2 angstroms of one another during a pairwise alignment, they are considered to be aligned.

ii initial iteration: congruent triangles are used to initially align two structures. The alignment resulting from such is used to find additionally conserved residues and the alignment is performed again using these new residues. ii specifies the maximum number of iterations this is to be done for.

im initial match: the first two structures to be aligned are considered matched if a set of conserved residues can be found between them that is at least im×100% of the smaller structure. If this cannot be achieved using triangles of trs1 congruency, trs1 is incremented sequentially by 2.5×, 5× and 10×. If it still cannot be met, the best alignment is used.

cm consensus match: similar to im, except cm indicates the degree of similarity sought between a structure and the current consensus set.

noi number of iterations: the multiple alignment and consensus extension steps will be performed for a maximum of noi iterations.

lmkf1 landmark freedom 1: the lowest threshold of residue conservation. Specified in the resulting consensus structure by the color yellow.

lmkf2 landmark freedom 2: the middle threshold of residue conservation. Specified in the resulting consensus structure by the color orange.

lmkf3 landmark freedom 3: the highest threshold of residue conservation. Specified in the resulting consensus structure by the color red.

redthrs redundancy threshold: if consensus points have redthrs or greater residues in common, they are considered redundant.

lfree landmark freedom: this value was introduced to minimize the time spent searching for consensus residues during the multiple alignment step. The location of each residue is used as a potential consensus point. If this point has at least lmkf1×100% samples with an equivalent amino acid, then the mean of this group is used as the consensus point. If a point has (lmkf1-lfree)×100% samples, the mean point of this group is used as a new consensus point to try and add additional samples to meet the lmkf1 threshold.

trs3 threshold 3: during the multiple alignment step, residues from different samples are considered to occupy the same location if they are within trs3 angstroms.

2.5 Specifying a core

The center of each protein is defined as the residue closest to all others, and the default core is then equated with the crn-1 residues closest to this. This is an important feature to consider as it impacts the initial pairwise alignment step and ultimately the resulting consensus. The program uses the core residues for generating test alignments, on the basis that the core will have the highest degree of conservation between samples. The geometric core may not always corespond with the feature of interest. For instance, if interest is on similarity around a ligand binding domain on the surface of a protein, aligning the geometric cores may not find potential similarity. Specifying a core will allow for the region of concern to be concentrated upon. The center of the core can be specified by residue number after the chain identifier in the list of PDB files to be used:

```
1ABC.pdb X 123
2DEF.pdb Y 456
3GHI.pdb Z 789
```

2.6 Points of note

If all of the residues in the input samples are standard amino acids or either a phosphorylated serine, threonine or tyrosine residue, the program will be able to properly label and use them. Residues that are not in this group will be ignored. If you wish to use these residues, you'll have to modify several subroutines in the resid.c file.

3 Output

Output consists of four things: PDB files modified according to the alignment, a consensus structure in PDB format, a csv file for viewing the consensus residues and a PyMol script for viewing the structures.

The PDB files are pretty straightforward. Residue conservation is specified in the occupancy columns. 1.00 indicates lmkf3 conservation of a specific amino acid, 0.75 lmkf2 conservation and 0.50 lmkf1 conservation. 0.25 indicates lmkf3 conservation of an amino acid category. PyMOL can easily handle commands to colour residues by occupancy, and I expect other molecular viewers can do the same. The three letter identifiers of amino acid categories in the consensus structure are: ACD (a-acidic), ALI (l-aliphatic), ARO (r-aromatic), BAS (b-basic), CHG (c-charged), HYD (h-hydrophobic), POL (p-polar), SML (ssmall) and VSM (v-very small).

The csv file can be viewed in any spreadsheet program. It has four principal columns (see example below). The first identifies the consensus point; the second, the amino acid or category found at the consensus point; the third, the number of samples with a residue at the consensus point (the number after the + sign indicates the number of samples with a residue at the consensus point which has a different amino acid but shares a category); and one column for each sample indicating the residue identifier. The consensus points are numbered based on the ordering of residues in the first sample of the PDB list, therefore choose the first entry based upon the structure of greatest familiarity. In the sample columns four types of entries may be found: a number indicating the conserved residue; an x indicating there is no conserved residue in this sample; a capital letter followed by a number - this is used for category consensus points to make known what actual amino acid from each sample is at the consensus point; and a capital letter, followed by a number, followed by a lower case letter. The last is used for samples that do not have the specific amino acid of the consensus point, but have an amino acid that shares a category (specified by the lower case letter).

3 landmarks

	type	3 samples	1ABC	2DEF	3GHI
1	A	2	123	x	456
2	h	3	G789	A234	P567
3	C	2+1	N891s	345	678

The alignment.pml script can be used for viewing the structures with PyMOL. Instructions on installing and using PyMOL can be found at http://pymol.sourceforge.net. To use the script (must be done in the directory containing the modified PDB files):

% pymol alignment.pml

Any molecular visualization program capable of handling PDB files can be used. My preference is for PyMOL.

4 Troubleshooting

4.1 No alignment

If no alignment can be generated for the input structures, try using the -s flag. If this fails, try reducing the number of samples. Input parameters could also be changed to allow for more freedom in positioning of equivalent residues, the congruency criteria could be reduced and the core could be changed and/or expanded.

4.2 Core dump

I have tested the program extensively but there is the chance you may get a core dump. Three main reasons: the input instructions were not followed, the parameters were set to values difficult to handle, and/or the PDB files have incorrect formatting. If unsure, follow input instructions, use default parameters and ensure PDB values have standard formatting (see http://pdbbeta.rcsb.org/pdb/file_formats/pdb/pdbguide2.2/guide2.2_frame.html). If the error is in the parameters and you feel the parameters you have chosen should work, e-mail me. Or if you can't find the source of the core dump, e-mail me. I'll respond quickly to such problems.

5 Additional functionality

There is great scope for additional functionality which I may make available. If there is something you think could be improved, done differently or added, feel free to do so yourself, or send me an e-mail. If the suggestion is valid or does not take much time, I'll make the changes and send you the modified program.

6 For the programmers

The program is written entirely in C and Fortran. I have made source code available to allow system dependent compilation; for others to make improvements, catch bugs, or hack if necessary; and because I believe in the philosophy of open source software. The program depends on several mathematical subroutines from the slatec library (http://www.netlib.org/slatec/).