CLASTERIT
Users' Manual
Contents

INTRODUCTION..................................................................................................................3
  Referencing CLASTERIT..................................................................................................3
  Installation of CLASTERIT..............................................................................................3
  Installation of additional software..............................................................................4
CLASICO..........................................................................................................................5
  Execution of CLASICO....................................................................................................5
    anal.sh.........................................................................................................................5
    graph.sh......................................................................................................................5
Flow Chart of CLASICO....................................................................................................6
Programs in CLASICO.......................................................................................................7
  ptjm.................................................................................................................................7
  moh.................................................................................................................................7
  mtcmp............................................................................................................................9
  patt.................................................................................................................................10
  ptcmpr........................................................................................................................10
  distgen.........................................................................................................................11
NMRDIST.........................................................................................................................12
  Programs in NMRDIST................................................................................................12
    distavg.......................................................................................................................12
    rmsdcat.....................................................................................................................13
    catdist.....................................................................................................................14
REFERENCES..................................................................................................................15
INTRODUCTION

CLASTERIT is a software to classify and cluster the conformations of proteins and peptides obtained through different computational methods, like molecular dynamics simulations, Monte Carlo or simulated annealing. The package has a collection of programs that are grouped into:

- CLASICO: group of programs that allow to classify structures in patterns based on backbone dihedral angles.
- CLUSTERIT: group of programs that allow to group patterns generated by CLASICO into clusters.
- CLABOND: group of programs that allow to classify structures in patterns based on the presence of H-bonds.
- CLUSBOND: group of programs that allow to group patterns generated by CLABOND into clusters.
- NMRDIST: group of program that allow to compare the distance intervals derived from NOE signals obtained by NMR experiments to those obtained by molecular dynamics trajectories.

The current limitations of the programs do not allow to study more than 200,000 snapshots.

Referencing CLASTERIT

The present program constitutes part of the PhD thesis of Francesc J. Corcho and has been published in:


If you use the program for your work, please, cite this paper to help in the development of this software.

Installation of CLASTERIT

In order to install you will require to use the gfortran package. If you have another fortran95 compiler substitute gfortran by your compiler name in the compile file under the src directory. Then execute:

./compile
If the bin directory is not created, you must created first. It is possible that the files compile and install do not have execution rights. Modify that with the instruction:

```
chmod u+x compile
chmod u+x install
```

**Installation of additional software**

Other software can be installed to visualize the results producing plots automatically. The following programs can be installed:

- `xmgrace`
- `R`
- `gromacs`

All these software are part of LINUX distributions (like Debian) and can be downloaded from the available repositories of these distributions. For instance to install R in debian you will do:

```
apt-get update
apt-get install r-base r-base-dev r-recommended
```
CLASICO

Execution of CLASICO

CLASICO is composed by several binaries files that have to be executed. In order to facilitate the execution of the several binaries the scripts anal.sh and graph.sh is placed under the scripts directory. The script graph.sh will produce the plots necessary to visualize the results of CLASICO.

In order to obtain all the plots automatically the following programs have to be installed:

- xmgrace
- R
- gromacs

All these software are part of LINUX distributions (like Debian) and can be downloaded from the available repositories of these distributions. See the section Installation of additional software above.

In order to facilitate the execution of CLASICO and the visualization of the results two scripts have been added: anal.sh and graph.sh.

anal.sh

This script executes sequentially the CLASICO programs for the classification of the conformational motifs of the snapshots saved for each trajectory: moh, mtcomp, and patt.

graph.sh

This script executes sequentially the programs that will produce the graphs for the visualization of the results of CLASICO. You will need to install xmgrace, R and gromacs if you want to make use of graph.sh or alternatively use any other plotting software.
Flow Chart of CLASICO
Programs in CLASICO

ptjm

In order to process the data the first step is to create the input for CLASICO with ptjm. The program ptjm will convert the data from the files omex, phix, and psix obtained in ptraj to create and input.std file.

moh

This program classifies structures based on structural motifs. The rules to classify the motifs are described in (1). A modification has been introduced from (1) to change the helix definition. Now, 3 consecutive residues which accomplish whatever helical condition (H,310 or PI) are considered an helix. The specific motif assignation is done in a local way depending on the condition accomplished. When different assignations can be done, the order of priority is H > 310 > PI.

The input.std file will be read by moh executable to create the MOHCLAS.mat.

There are three (related) ways implemented for classifying the patterns:

1. Directly classify the residue in agreement with the tables (c.f. definition_table document). The acronym definitions for the motifs are:

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>I1</td>
<td>first residue of a beta turn type I</td>
<td></td>
</tr>
<tr>
<td>I2</td>
<td>second residue of a beta turn type I</td>
<td></td>
</tr>
<tr>
<td>i1</td>
<td>first residue of a beta turn type I'</td>
<td></td>
</tr>
<tr>
<td>i2</td>
<td>second residue of a beta turn type I'</td>
<td></td>
</tr>
<tr>
<td>II1</td>
<td>first residue of a beta turn type II</td>
<td></td>
</tr>
<tr>
<td>II2</td>
<td>second residue of a beta turn type II</td>
<td></td>
</tr>
<tr>
<td>ii1</td>
<td>first residue of a beta turn type II'</td>
<td></td>
</tr>
<tr>
<td>ii2</td>
<td>second residue of a beta turn type II'</td>
<td></td>
</tr>
<tr>
<td>III1</td>
<td>first residue of a beta turn type III</td>
<td></td>
</tr>
<tr>
<td>III2</td>
<td>second residue of a beta turn type III</td>
<td></td>
</tr>
<tr>
<td>iii1</td>
<td>first residue of a beta turn type III'</td>
<td></td>
</tr>
</tbody>
</table>
This classification generates the MOHCLAS.mat file, where the pattern information is encoded and MOHCLAS.stat2, with statistics about the presence of each motif per residue.

2. The second way tries to solve some redundancies in the motif classification. Some decisions have been taken and all of them should be retained in mind:
   a) bI/bIII indetermination always solved assigning bI.
   b) If a residue is assigned as helical then it can not be assigned as b-turn
   c) If beta strand, not polyproline helix (take care, this decision absolutely delete all polyprolines helices, so it should be checked when multiple consecutive prolines exists in your peptide)
   d) If alpha helix, not 3_10-helix

This classification generates the MOHCLAS.mat2 file, where the pattern information is encoded and MOHCLAS.stat3, with statistics about the presence of each motif per residue.

3. The third (and last way) simplifies the information from the motif assignment (once the redundancies are solved) by just considering one b-turn type (called beta).
This classification generates the MOHCLAS.mat3 file, where the pattern information is encoded and MOHCLAS.stat4, with statistics about the presence of each motif per residue.
The second and third ways matrices files (MOHCLAS.mat2 & MOHCLAS.mat3) can be used for the next steps in cluster analysis by copying to a MOHCLAS.mat file.

**mtcomp**

The next step is to convert the MOHCLAS.mat into a compressed matrix with the *mtcomp* executable. The format of MOHCLAS.comp contains in the first column the number of the snapshot. The second column contains the number of following items in the same row. The following columns are the residue, the number of conformational motifs and which ones there are, following the numbering in the table below.

<table>
<thead>
<tr>
<th>Number</th>
<th>Conformational motif</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>I1</td>
</tr>
<tr>
<td>2</td>
<td>I2</td>
</tr>
<tr>
<td>3</td>
<td>i1</td>
</tr>
<tr>
<td>4</td>
<td>i2</td>
</tr>
<tr>
<td>5</td>
<td>II1</td>
</tr>
<tr>
<td>6</td>
<td>II2</td>
</tr>
<tr>
<td>7</td>
<td>ii1</td>
</tr>
<tr>
<td>8</td>
<td>ii2</td>
</tr>
<tr>
<td>9</td>
<td>III1</td>
</tr>
<tr>
<td>10</td>
<td>III2</td>
</tr>
<tr>
<td>11</td>
<td>iii1</td>
</tr>
<tr>
<td>12</td>
<td>iii2</td>
</tr>
<tr>
<td>13</td>
<td>Via1</td>
</tr>
<tr>
<td>14</td>
<td>Via2</td>
</tr>
<tr>
<td>15</td>
<td>Vib1</td>
</tr>
<tr>
<td>16</td>
<td>Vib2</td>
</tr>
<tr>
<td>17</td>
<td>vib1</td>
</tr>
<tr>
<td>18</td>
<td>vib2</td>
</tr>
<tr>
<td>19</td>
<td>H</td>
</tr>
<tr>
<td>20</td>
<td>310</td>
</tr>
<tr>
<td>21</td>
<td>Pi</td>
</tr>
</tbody>
</table>
The program patt classifies the structures in patterns based on the conformational motifs present.

*MOHCLAS.comp* and *PATT.sum* are input files

*PATT.patt* is the pattern file (exactly equal to *MOHCLAS.comp* but only with non-repeated patterns)

*PATT.group* is the file for record the appearing of new patterns in fact we are assigning the pattern number to the structure

*PATT.perclass* records the number of times that a given pattern appears.

*PATT.relev* records the number of times that a given pattern appears if it appears more than the 0.1% of structures.

---

**ptcmpr**

Compares the patterns in two *MOHCLAS.comp* from different simulations of the same molecule (it is strictly necessary to have the same number of residues that generate the two *MOHCLAS.comp*). This comparison only makes sense if the two *MOHCLAS.comp* files are generated from the same way of the third available after mohclas (c.f in mohclas discussion above)

The usage is: **“ptcmpr fileA fileB”**

where fileA is the first “*MOHCLAS.comp* like” file and fileB is the second one. Thus, all the output files just labeled with A correspond to fileA information, all of them labeled with B correspond to fileB information and those labeled with AB have information related with both files.

*FileA.out* and *fileB.out* have 3 columns: The first one is the pattern number, the second one is the first conformation in which this pattern appears and the third one is the number of times that this pattern appears.

*FileA.relev.out* and *fileB.relev.out* have the same structure, but only with those patterns with a weight of 0.1% or more.

*FileAB.out* has a four column structure. If a given pattern is shared, then it appears in this file in the next format:
nconf(a) numele(a) nconf(b) numele(b)

nconf(a) is the first conformation in which this pattern appears in file A and nconf(b) is the first conformation in which this pattern appears in file B. numele(a) and numele(b) are the number of times that the pattern appears in each file.

SharingAB.out has a summary of the statistics of shared patterns between file A and B. It has two different parts, shared patterns (the statistics only has into account the number of patterns shared) and shared conformations (the statistics also has into account the number of times that a given pattern is shared). Furthermore, the percentage is considered in front the number of conformations present in file A and in file B.

**distgen**

This program transforms the input.std file in a input file for kleiweg clustering programs (http://www.let.rug.nl/kleiweg/clustering/).

Each phi/psi value is split in cos and sin descriptors.

Take care when using that program, similar angles not necessary lead to similar structures (a big change in one angle but maintaining the rest of them equal can lead to a low distance between both structures with these descriptors but being the structures intuitively quite dissimilar).
NMRDIST

Programs in NMRDIST

distavg

1. Use ptraj (AMBER) to obtain separate files for each of the distances you want to compare. When two or three hydrogens are equivalent set the ptraj input to compute the distance between the single hydrogen and the center of mass of the two or three equivalent hydrogens.

2. Write a file containing all the names of the distance files ordered (distfile.ord). The first line should contain the first and last structure that you want to include in the computation of the average distance:

   1 200000
   dist-aN1_1
   dist-aN1_2

3. Execute:

   distavg distfile.ord 95.0

   The first argument correspond to the name of the file that contains the names of distance files, and the second argument is the interval of confidence desired for the distances.

   Three outputs are generated:
   - For each distance a file called histo-00x.dat is generated. If you want to visualize the results you can use xmgrace and execute:
     xmgrace histo-*.dat
   - The file disavg_numeric.out contains for each distance the number of intervals that contains the percentage of structures specified by the second argument. We will use 95.0 or 50.0 to define intervals that contain 95% or 50% of the structures of the trajectory
   - The file disavg_numeric_plot.out contains for each interval the middle value and half the width interval so the intervals can be represented with xmgrace or other application as a plot type xydx.
The file `disavg_generic.out` represents an interval containing always 95% of the structures and considering a normal distribution. The output contains 10 columns:

- **N**: the distance number
- **mean**: the average distance value
- **intwidth**: 1.96*times the standard deviation
- **stdev**: the standard deviation of the distances
- **skwns**: the skewness, a measure of the asymmetry of the probability distribution of a real-valued random variable
- **kurts**: the kurtosis, a measure of the "peakedness" of the probability distribution of a real-valued random variable.
- **low**: the low end of the confidence of interval (mean-intwidth)
- **high**: the high end of the confidence of interval (mean+intwidth)
- **min**: the minimum distance value
- **max**: the maximum distance value.

4. Create a file manually called `distNOE.in` containing:
   - first column: higher end of the NOE distance interval
   - second column: distance number
   - third column: '0'
   - fourth column: the width of the NOE distance interval

5. Compile `overlap.f` to `overlap`

6. Execute `overlap`

**rmsdcat**

Given a file containing the number of intervals and the ends of the intervals, and another file where for each snapshot of the trajectory the rmsd has been computed in respect to a reference, the program `rmsdcat` groups the structures and print their number in separate files based on the rms distance.

**catdist**

Given a number of files, output of the program `rmsdcat` that contains the structures that fulfil each of the rmsd intervals defined in the file `intinfo`
and given the file distfile.ord containing the list of files for which the distances corresponding to NOE's atoms have been computed, the program will select those distances and save them in separate files for executing distavg on them.

Usage: catdist distfile no.of intervals intervalstat

eg: catdist distfile.ord 6 intervalstat.dat

This will group all the distances present in all the distance files contained in the file distfile.ord in the first 6 interval distances present in intervalstat.dat. If you will only one to select those distance in one of the intervals in intervalstat.dat, you should create a file containing only the interval that you want to select the distances from and then execute:

eg: catdist distfile.ord 1 new_intervalstat.dat

As a result of the execution of catdist you will create as many files for each of the distance files as intervals you have. Then you can execute again distavg, but a file distfile.ord_x has to be created, with x being the number of the interval of rmsd you want to calculate the average for. The name of files contained in distfile.ord_x should end their name with _x.int.

eg. dist-NN1_1 file should be renamed in distfile.ord_1 to dist-NN1_1_1.int.

Note: The number of intervals is limited to a number below 10 in the current release.
REFERENCES