

# uQlust Manual

Rafal Adamczak & Jarek Meller, Cincinnati & Torun, 2014/2015

This document contains the user manual for uQlust **ver. 1.0** (released as of August 2015).

## Section 1: What is uQlust?

uQlust is a software package for ultrafast ranking and clustering of macromolecular structures.

uQlust combines versatile structural profiles of both proteins and nucleic acids with linear time algorithm for comparison of all pairs of models using 1D-Jury (Adamczak and Meller, 2011), and profile hashing for efficient and low memory footprint clustering of macromolecular structures, including hierarchical clustering. While reducing the computation time by orders of magnitude with respect to existing methods, uQlust yields comparable accuracies in protein and RNA clustering and model quality assessment. Even though it has been primarily developed for large scale structure prediction and molecular simulations for both proteins and RNA (in which case all models pertain to just one macromolecule and are assumed to be of the same length), uQlust can also be used in conjunction with an arbitrary profile, such as the FragBag structural motif frequency profile.

uQlust was written (in C#) by Rafal Adamczak, based on a joined work with Jarek Meller (Adamczak & Meller, 2011, 2015). The code is easily portable between different operating systems, and the system independent pre-compiled executables can be run as long as .NET (Windows) or Mono (Linux) are installed (see also the next section). The source code, this user manual and a number of utilities and benchmarks and examples are available as part of this package as well.

### References:

R. Adamczak & J. Meller, *uQlust: Ultrafast Ranking and Clustering of Macromolecular Structures*, to be published (with some luck it should be 2015);

R. Adamczak, J. Pillardy, B.K. Vallat, J. Meller; *Fast Geometric Consensus Approach for Protein Model Quality Assessment*. Journal of Computational Biology 18(12):1807-18 (2011); PMID: 21244273 (for now, please cite the above 1D-Jury paper)

## Section 2: Executing uQlust

System Requirements: **Windows** 64 bit system

1. .NET v.3.5 or higher (if .NET is not installed, Windows will install it automatically)

NOTE: It is possible to run uQlust executables also under 32 bit system; however, when using profiles that are prepared with the DSSP utility, the dssp module must be recompiled, which requires that the boost library is installed as well.

System Requirements: **Linux**

1. Mono v.3.8 or higher
2. boost library

NOTE: The boost library is required to re-compile the DSSP module. The Makefile to compile DSSP is provided, just run the 'make' command in the DSSPModule directory. Check if the file libDSSP.so has been created in the main uQlust directory, which contains the uQlust.exe file (and is referred to as uQlustDistr).

Running uQlust: **Windows**

**GUI version**

uQlust.exe

**Terminal version**

uQlustTerminal.exe -f configuration\_file

Running uQlust: **Linux**

mono uQlust.exe

**Terminal version**

mono uQlustTerminal.exe -f configuration\_file

Examples of configuration files with pre-defined workflows are included in the **workFlows** subdirectory of the uQlustDistr directory. When running uQlust GUI interface for the first time, one needs to set options for uQlust that would define the input file location and type, the type of task to be performed (e.g., ranking vs. clustering), 1D-profile, methods to be used etc.

### **Section 3: Setting up workflows and options in uQlust GUI**

The following refers to setting up options when using uQlust GUI.

#### **STEP 1: DEFINE INPUT**

To run any ranking or clustering method using uQlust one needs to define the source of structural data to be used, in particular the subdirectory that contains structures for analysis. The 'Add' button in the main form that opens automatically when executing uQlust GUI allows the user to

select the source of structures/models. Such selected directory should appear in the current data directory list.

NOTE: One may add multiple directories using 'Add' button; each of these directories will be analyzed separately by running the method selected in the next step (ranking or clustering of some type) on structures from each of those directories separately. One may also remove directory from the list by using the 'Remove' button to remove selected, or all directories by pressing the 'Remove All' button. There is also possibility to add a list of directories that are defined in an external, user generated file by using the radio button "File with list of the directories". Each directory in the file should be listed in a separate line, e.g.,

H:\casp10\T0650  
H:\casp10\T0652  
H:\casp10\T0653

## STEP 1A: DEFINE AUXILLIARY OPTIONS

Use the submenu 'Settings' in the main menu to define the following items (if applicable):

- The type (extension) of files with input structures to be analyzed: from the directory specified only files with this extension will be read by uQlust;
- The name of the directory where profile files to be generated will be stored, and if exist, will be read, so there is no need to generate profiles again;
- Define the maximum number of CPU cores that will be used during calculations;
- uQlust macromolecule mode: there are two possibilities at present, i.e., 'Protein' or 'RNA'. This is used to read the structure files and define applicable profiles - some profiles are only available in Protein mode. For example, in the case of the protein mode it is possible to use only alpha carbon atoms, or all atom models. NOTE: RNA mode is still under development; as an alternative, one can use a user defined profile for RNA structure analysis.

## STEP 2: DEFINE ANALYSIS TYPE

### Ranking methods:

1. **'1DJury'**: the algorithm ranks structures based on the similarity between their profiles. It calculates similarity of particular structure to all other available structures, as in 3D-jury algorithm. The difference is that profiles with smart preprocessing are used here, reducing the complexity of the algorithm to  $O(n)$ , where  $n$  is the number of structures;
2. **'3DJury'**: the algorithm ranks structures based on their 3D superposition-based distance with  $O(n^2)$  time complexity. NOTE: this can be very slow, and not significantly more accurate than 1D-jury; to be used primarily for evaluation purposes on small data sets;
3. **'Sift'**: in this algorithm, score for each of the structure is calculated based on the analysis of packing of amino acid residues within the first and second contact shell in terms of

radial distribution function; this is very useful when assessing if protein models are physically plausible, and can be used for ranking as well.

### Clustering methods:

- 1) **‘Hash clustering’**: available options under this form include **‘uQlust:Hash’** (uQlust:Hash(K,F)) and **‘uQlust:Rpart’** (uQlust:Rpart(K,F));
- 2) **‘Hierarchical clustering’**: available options include **‘uQlust:Tree’** (or approximate hierarchical clustering starting from hash key-based initial slicing of data and subsequent aggregation and tree building) or **‘Agglomerative’** (or traditional full hierarchical clustering) with either Hamming distance or RMSD, plus **‘Fast top-down’** clustering using 2-means to split the data, and **‘Davies-Bouldin top-down’** approach that uses repeated runs of K-means to define an optimal split of the data at each level; NOTE: while uQlust:Tree is very fast and is meant to be applied to large data sets, the remaining approaches may turn out to be rather slow – use with caution;
- 3) **‘Baker’**: a heuristic developed in D. Baker’s group that uses a form of distance-based clustering with a random reference state (not very well tested).

NOTE: In hash clustering, one must provide the number of target clusters, K, and the fraction of data, F, these K target clusters should contain. NOTE: the choice of K and F should be carefully assessed, just like the choice of K for K-means. It is one of the goals of uQlust to make it possible to run repeatedly fast clustering with the goal of assessing the quality of results with different parameters and effectively estimate the optimal number of clusters K.

NOTE: In order to perform ‘hash clustering’ one has to specify a profile to generate hash keys and a method to define a consensus profile (see also the next section). Consensus profile may be defined by a consensus state on each position of the profile, or by using 1d-jury ranking to define a 1D-jury centroid as a reference vector. Subsequently, each profile is transformed into a hash key based on consensus profile as follows: each position in the profile is compared with the corresponding position in the consensus profile, if these are the same states, then 0 is put into the key (otherwise 1 is added).

Consensus states	CCCCCCCCCCCCHEEE
Current profile	CCCCHCCCHHEEHEEH
Key	0000100011001001

The hash keys built in this way are used in the hash table as a basis for fast aggregation of data into micro-clusters: if the keys are the same, the structures are put in the same cluster. Such defined hash keys can be shortened (and thus leading effectively to a better aggregation) by using regularization options. For regularization size of the window, profile (can be different than

the profile for keys generation) and the threshold distance must be provided. Symbols with red color represent current regularization window:

Current key                    0000100011001001

Distance between consensus states and current profile in the considered window is 1, so if the defined threshold distance  $\leq 1$  than the window after regularization looks like this:

After regularization        0000000011001001

NOTE: In order to achieve the required number of target clusters in `uQlust:Hash(K,F)` one can use one of the following two heuristics:

- Combine clusters by column selection – this method adjusts the length of the key by removing from the key positions that in all considered structures have almost the same states. There are two possibilities available:
  - Entropy - for each key position entropy is calculated. Positions with the lowest entropy are removed from the key. The removing procedure is repeated as long as the number of relevant clusters is reached.
  - Meta columns – this method decreases the length of the key by 3. It analyze 3 position in side by side e.g. for  $i$ -th position in the key  $i+1$  and  $i+2$  is considered and based on their states new key is built. For example if those 3 positions have the same state then in the key 0 will be put to the key otherwise 1.
- Combine clusters by hamming distance – instead changing key, clusters are joint together based on the hamming distance to the consensus state profile. For each of the key distance, number of 1 in the key, to the consensus profile is calculated and all distances are sorted. The key with minimal distance is selected and the other structures that are in similar distance from the consensus profile are checked if there are close enough to the selected one. Once cluster is set, next key with the lowest distance, so far not used, is selected and procedure is repeated. Phrases “similar distance”, “close enough” means parameters that are found in automatic way to fulfilled clustering requirements (numbers of relevant clusters and their coverage).

NOTE: In the traditional, full hierarchical clustering (referred to as ‘**Agglomerative**’) **one can select any of** the typical linkage criteria: single, average or complete linkage. Moreover, one can select among distance measures: Rmsd and Maxsub (for proteins/RNAs, using their 3D structures), and Weighted Hamming (for any profile).

NOTE: For each of the clustering methods, centroids/reference structures at each level may be found be averaging structures that belong to the cluster with the selected distance measure, or by using 1d-jury. In this case one has to check “Use 1d-jury to find reference structure” checkbox and specify profile that will be used by the 1d-jury method.

### STEP 3: DEFINE STRUCTURAL PROFILES TO BE USED

Structural profiles are used in uQlust to provide a 1D projection of the structure, with the goal of achieving efficiency with a minimal loss of accuracy. For example, protein structure may be characterized by secondary structures or solvent accessibilities. Each of the profiles consists of states, in case of secondary structures one can use three states (helix, beta strands or coils) or 8 states. The number of states depends on the profile used. However, for some algorithms, including uQlust:Hash and uQlust:Rpart, less is generally better. In uQlust, there are number of predefined profiles (internal profiles) that can be used. Although they are predefined the user can adjust/tune-up them by using the graphical profile editor available in uQlust. It allows one to change the number of states in the profile by combining some of the states into one state, and to define weights for each type of transition between the states, reflecting the implied similarity between the states and effective penalty for not being in the same state. For example, for secondary structure states H, E and C, states H and E can be considered as the most opposite states, while C could be regarded as somewhat similar to both H and E. Consequently, observing a difference between H and E would be penalized much more strongly, compared to a transition between H and C. Such flexibility allows one to define profiles with some implicit notion of similarity between the states, while still providing a linear complexity solution to the ranking problem (see also Adamczak and Meller, 2011). For example, in order to calculate the hamming distance between two secondary structures profiles

```
1: C C C H H H C C C
2: H C C E H H C C C
```

One can consider that the difference between H-E is more important than C-H, it is possible to add weight for the transition H-E higher than C-H. It is also possible to use a number of profiles at the same time. For example protein structure can be represented by secondary structure profile and solvent accessibility profile (0 – fully buried state, 9 – fully exposed state)

```
C C C H H H C C C
9 9 9 3 3 3 7 7 7
```

All selected profiles will be combined into one profile that combines together states on each position, so the final product looks like this:

```
C9 C9 C9 H3 H3 H3 C7 C7 C7
```

Also transition table will be merged. Depends on type of transition table, there are two types: similarity – used by 1djury or distance, combination of tables will be different. For similarity table merging is made by multiplying scores, for distance by adding them.

Similarity transition tables

SS transition table

	C	H	E
C	2	1	1
H	1	2	0

SA transition table

	0	1	2	...
0	0.4	0.3	0.2	...
1	0.3	0.4	0.3	...
2	0.2	0.3	0.4	...
...	...	...	...	...

Result

	C0	C1	C2	...
C0	0.8	0.6	0.4	...
C1	0.6	0.8	0.6	...
C2	0.4	0.6	0.8	...
...	...	...	...	...

Distance transition tables

	C	H	E
C	0	1	1
H	1	0	2

	0	1	2	...
0	0	0.2	0.3	...
1	0.2	0	0.2	...
2	0.3	0.2	0	...
...	...	...	...	...



	C0	C1	C2	...
C0	0	1.2	1.3	...
C1	1.2	0	2.2	...
C2	1.3	2.2	0	...
...	...	...	...	...

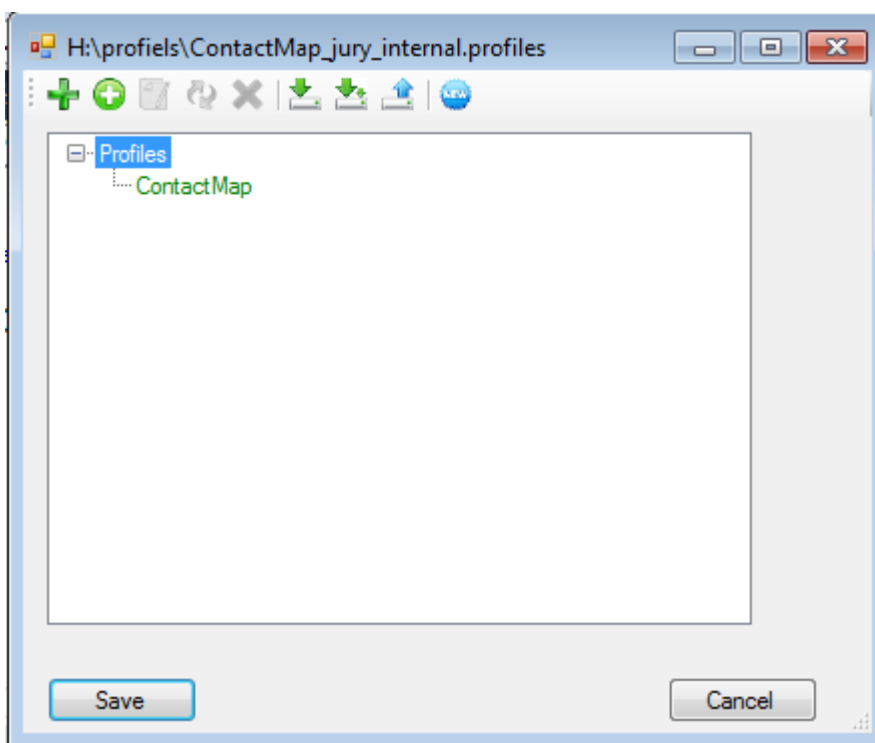
uQlust has the following predefined profiles:

- SS – secondary structure profile is generated by dssp module that is build in the uQlust, there are 8 states, but defining the profile one may change it to the lower number of states
- SA – solvent accessibility also generated by dssp module, there are 9 states
- Contact – contact-based profile that is based on distance between side chains atoms, the state represents number of residues that are in contact (distance smaller than 8.5Å) with a particular residue. There are 10 states if the number of residues in contact is higher than 9, than the state remain 9.
- User defined profiles – profiles that are generated by user outside of uQlust, inside uQlust one has to define all states that could appear in the profile, and transition table
- Contact CA-CM – the same as Contact but only c alpha atoms are used
- SS CA - secondary structure, states are defined based on the distance between C alpha atoms. The distance is calculated between i-th and i+2 (dist2) residue and i-th and i+4 (dist4) residue. There are 9 states including state U (unphysical).
- ContactMap – binary contact map represents the structure of the protein, the threshold for contacts vs non contact is 8.5Å and the sequence distance between considered residues is 12. In all previous profiles the length of the profile was the same as the number of residues in the protein. In this case we have full contact map so theoretically the length should be roughly  $(n-12)*(n-12+1)/2$ , but because the profile for long protein could be very long, so to decrease amount of needed storage space contact map profiles are preprocessed. Contact map profiles are

sparse, so whole ensemble of profiles of structures used for clustering is checked to find position in the profiles that has everywhere 0, this kind of position is removed from the profile. The disadvantage is that when new structure will be added to the ensemble (after profiles where generated) all contact map profiles must be generated again. To force regenerating profiles you have to delete profile file from the storage or change directory where the profiles are stored.







### Profile manager

To configure profiles that should be used for each structure profile manager should be used. Below is a window of profile manager with ContactMap profile added to the profiles list. To add additional profile to the list, "Profiles" node should be selected and then  press button . If the ContactMap node will be selected, then the new profile will be created as a child, this may be useful if to create some profile existing of some other profile is required. In typical situation new profile should be added when "Profile" node is selected.





To make any change on the profile first it must be selected. There are following operations available in profile manager:

-  • Edit profile to change state definition or transition table
- 
  - Allows to change selected profile from active to non active state and vice versa. Active profiles are depicted in green color non active in red. Only active profiles are used to represent structure.
- 
  - Remove profile from the list.
- 
  - Save as, save, load
- 
  - Remove all profiles from the list
- Add  profile that will be generated by external program

In case when non standard profiles (that do not exists in uQust) need to be used one can load them to the uQust. Profiles must be stored in the file. The structure of the file is following:

>name of structure

name\_of\_profile profile definition of profile

Example 1:

>Run8\_2\_627.pdb

ContactMap profile 0 0 0 0 1 0 1 0 1 0 1 1 1 1 1 0 0 0 0 1 1 0 0 0 1 0 1 0 0 0 0 1 1 0 0 0 0

>Run0\_1\_191.pdb

ContactMap profile 0 1 1 1

Example 2:

>d4579.pdb

SS profile C C H H H H H - T T S S S S H H H H T T T S

SA profile 9 8 4 6 7 4 0 4 6 2 1 5 6 6 1 1 1 8 6 4 9 7 2 9 6 9

>d11939.pdb

SS profile C C C - H H H H H T T S S S S S S G G G S

SA profile 9 9 8 7 6 0 2 6 2 1 4 9 3 8 3 0 6 8 2 7 2 0 0 2 6 9

The external profiles that are stored in the file must be aligned, so each of the profile must be of the same length. The gaps in the alignment are represented by character '-', as can be seen in Example 2. Moreover for each of the structure one can define more than one profile, as shown in Example 2, the names of the profiles must be different. To use external profile one has to select "File with aligned profiles" radio button in the main frame.

### **Distance measures**

There are 3 available distance measures:

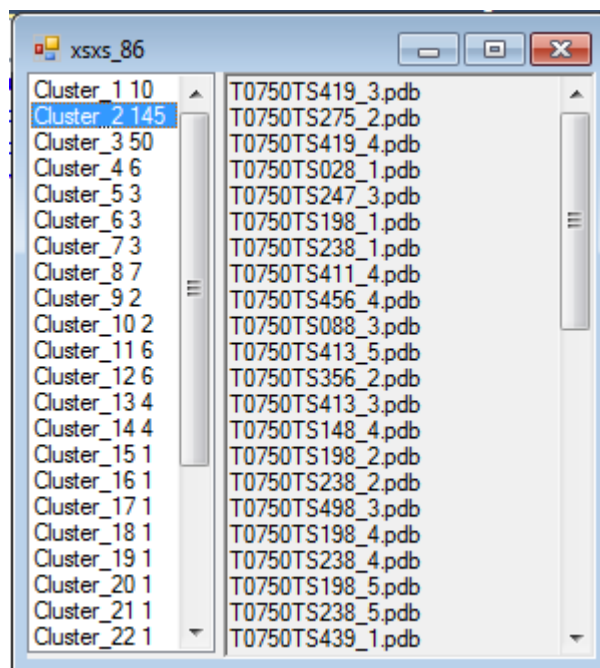
- 1) Rmsd – root means square deviation of atomic positions. There are two options (radio buttons)
  - a) All atoms – calculate rmsd using all available atoms
  - b) Only CA – calculate rmsd using only c alpha atoms
- 2) Maxsub – only c alpha atoms are used
- 3) Weighted Hamming – hamming distance calculated for specified profile.

## **STEP 3: ANALYSIS AND VISUALIZATION OF THE RESULTS**

### **Visualizing results**

Results are available on the main form in the section "Clustering results". If clusterization is finished the font color is green otherwise is red. One must click LMB on the row with results that should be visualized. If row is selected click RMB on it and context menu will appear where one of the option is "Show Results". There are following visualization methods available:

1. **Text list**  
Available for non-hierarchical clustering methods.



Left panel of the presented window contains list of clusters that consist of the label “Cluster”, its number, and the numbers of structures in the cluster. Right panel contains of the list of structures that are in the selected cluster.

## 2. Order View

Available for non-hierarchical clustering methods. Allows to see transitions between clusters. File with defined order of structure is needed, so when this method is chosen open file dialog will appear and one has to select file with defined order. The format of this file is following:

```
structure_name order_number
```

For example:

Run0\_0.pdb 0

Run0\_1.pdb 1

Run0\_2.pdb 2

Run0\_3.pdb 3

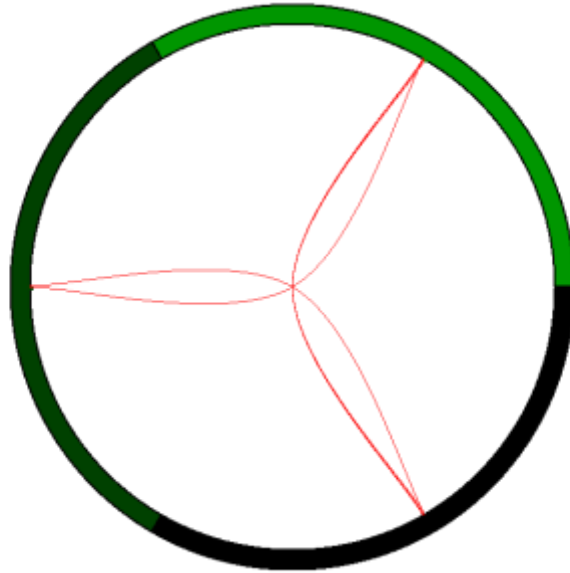
Run0\_4.pdb 4

Run0\_5.pdb 5

Run0\_6.pdb 6

Run0\_7.pdb 7

If first three structures will belong to cluster\_1 and the next two structures to cluster\_2 and last three to cluster\_3 one will observe transition from cluster\_1 to cluster\_2 and from cluster\_2 to cluster\_3. Picture below shows results for 3 clusters. Clusters are positioned on a circle, the bigger the cluster the more space on the circle it takes. Transitions between clusters are denoted by red lines. One may click LMB on the cluster to see only transitions from and into this cluster.



### 3. Circle View

Available only for hierarchical clustering methods. Moreover for this visualization file with label (class) definition is required. The format of the label definition file is following:

structure\_name label

For example:

Run0\_0.pdb Run0

Run0\_100.pdb Run0

Run0\_101.pdb Run1

Run0\_102.pdb Run1

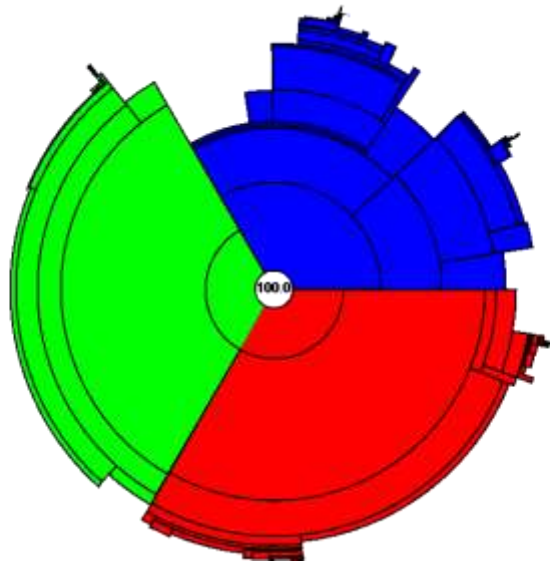
Run0\_103.pdb Run2

Run0\_104.pdb Run2

Clusters, denoted by black lines, are represented as a part of a wheel, the bigger cluster the bigger part it takes. The distances from the middle (white) circle represent the distances of the representatives of the clusters from each other. The number in the middle shows the percentage of the available data currently shown on the visualization, one may click LMB on any cluster to see it in more details and click

RMB to return to the previews visualization. One may change the definition of colors for each label by clicking LMB on the region where color and the label appears (on the picture left upper corner).

■ Road  
■ Road  
■ Road



#### 4. Dendrogram

The black squares on the dendrogram denote the points where one may LMB or RMB click. The LMB click will show the text list with structures that belongs to the cluster. The RMB click allow to zoom dendrogram, so on the dendrogram only clusters below the selected one will appear, and the selected one will be on the top of hierarchy. To return to the previews picture one may click RMB, pointing mouse outside of any square region.

## APPENDIX: FORMAL DEFINITION OF MAIN METHODS AND PROFILES

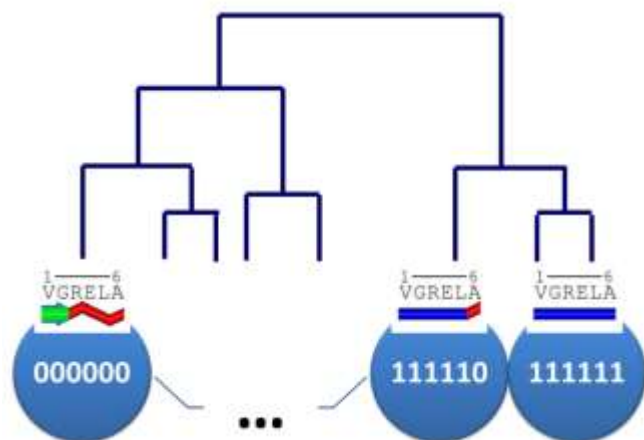
Based on Adamczak & Meller, to be published

### S2. Profile hashing and clustering heuristics

Geometric consensus-based model ranking with 1D profiles can be performed in linear time. Similar simplifying assumptions are made for the problem of explicit clustering: i) the input set consists of alternative 3D structures of the same macromolecule (fixed length); and ii) 3D structures can be efficiently projected into suitable 1D structural profiles. With those assumptions, an approximate hierarchical clustering approach for large data sets is introduced that achieves effective linear scaling (although it is not strictly linear in this case). The proposed hierarchical approach overcomes the limitations of K-means type methods, including the problem of selecting the number of clusters  $K$ , numerical convergence, and the lack of recursive partitioning structure, while maintaining computational efficiency of K-means algorithms.

**Profile hashing** is used in conjunction with 1D-Jury as a basis for low memory footprint and ultrafast clustering heuristics available in uQlust, including approximate hierarchical clustering. Binary hash keys are generated with a 1D profile of choice by comparing each profile with a reference profile that obtains the maximum 1D-Jury score (which can be computed in linear time). The hash key for a profile is defined at each position as follows: 0 is added to the key if a given profile is in the same state as the reference profile at that position, 1 is added otherwise (note that the number of ones in a key is equal to the Hamming distance from the reference profile). Since the best 1D-jury score reference profile is expected to represent a natural geometric consensus for a substantial subset of models, one can also expect that many of such partially (locally) consistent models will likely obtain the same hash key, resulting in less granular partitioning into micro-clusters (subsets of profiles with the same value of the hashing function) compared with a random reference structure, or a direct use of multi-state profile as opposed to binary ‘geometric consensus’ keys.

The first heuristic discussed here is a profile hashing-based clustering that directly draws from the above considerations. It is referred to as **uQlust:Hash(K,F)**, where  $K$  defines the number of target clusters, and  $F$  denotes the fraction of data that should be contained within those  $K$  clusters. Hash(K,F) starts by initial data ‘slicing’ into micro-clusters with the same value of the hashing function. Subsequent agglomeration into  $K$  clusters (comprising  $F\%$  of data) is obtained by simply changing the granularity of hash keys, which is achieved by removing sufficient number of profile hash key positions with lowest entropy across all data



vectors.

**Figure 1S.** Schematic representation of approximate hierarchical clustering with profile hashing to define ‘micro-clusters’ (lower level in the figure, with hashing keys in terms of consensus structure) to be subsequently hierarchically clustered, starting from the representative structures (shown above) in each micro-cluster using some convenient similarity measure, such as Hamming or RMSD distance.

Another heuristic in uQlust is a form of reference-based partitioning, which is referred to as **uQlust:Rpart(K,F)**. As before, this new heuristic relies on the initial identification of 1D-jury ‘centroid’ for the entire data set, as a suitable reference conformation. For efficiency and granularity, Rpart also represents all profiles in terms of binary hash keys. However, the subsequent partitioning of data proceeds very differently. Rather than aggregating hashing-based micro-clusters, Rpart identifies recursively macro-clusters centered on a reference profile by adjusting radius of clustering to achieve certain  $K$  clusters comprising  $F\%$  of data. Specifically, Hamming distances to the reference profile hash key are computed to identify a central inner sphere that

contains data points closer than the radius of clustering (initially set to  $\frac{1}{4}$  of the maximum distance from the 1D-jury reference) to the reference vector. Such defined sphere constitutes a candidate for a macro-cluster. The profile (structure) with the highest 1D-Jury score in the outer layer is then selected as the next reference structure, the radius of clustering is reset to its distance from the original 1D-jury reference, and the process is repeated on the remainder of the data, such that only points at distances less than twice the current radius of clustering from the original 1D-jury reference are considered. If after K iterations less than F% of data points are covered by such defined K candidate macro-clusters, then the size of the radius of clustering is increased (or decreased if more than F% of data points are covered), and the process is repeated. The process stops when no further improvement towards the targeted F is observed.

Finally, in the case of approximate hierarchical clustering, which is referred to as **uQlust:Tree**, the first step is analogous to that used for Rpart(K,F) or Hash(K,F), except that a large K is used to induce a large number of small clusters (micro-clusters, see Figure 1S) and F is set to 100% to include all data. While K can be modified by the user to set the tradeoff between speed and accuracy, its default value is set to K=1,000 to provide sufficient granularity in both: conformational space partitioning and model clustering for quality assessment (note that K and F are effectively fixed and can be dropped in references to uQlust:Tree). In the next step, a 1D-jury centroid is computed for each micro-cluster, and from this level traditional average distance agglomerative (bottom-up) hierarchical clustering with either Hamming distance (for arbitrary profiles), or RMSD (only for proteins or RNAs) is applied. As a result, effectively linear complexity in the number of structures,  $N_{\text{struct}}$ , is achieved when  $N_{\text{struct}} \gg K$  (see running times in Table 2).

### S3. Definition of structural profiles

An arbitrary, residue level profile generated by using an external application can be used for ranking or clustering in conjunction with Hamming distance-based ranking and clustering approaches. Internally computed profiles (starting from a set of all-atom or reduced PDB or DCD files) include:

- SS-SA or secondary structure (SS) – solvent accessibility (SA) profile, which assigns each amino acid residue to one of up to  $N_{\text{SS}} = 8$  secondary structures (by default  $N_{\text{SS}} = 3$ ), and one of up to  $N_{\text{SA}} = 10$  solvent accessibility states (by default  $N_{\text{SA}} = 2$  with the threshold of 20% relative solvent accessibility (RSA) separating ‘buried’ from ‘exposed’ states); the DSSP utility (Touw et al., 2015) is implemented internally to provide the definition of SS and RSA;
- CA(SS)-NC(SA) or approximate distance dependent secondary structure (SS) – solvent accessibility (SA) profile, which can be used for  $C_{\alpha}$  models, and assigns pseudo-secondary structure states based on distances (in Ang) between  $C_{\alpha}$  atoms (CA); Specifically, a combination of  $d4 = d(C_{\alpha,i}, C_{\alpha,i+4})$  and  $d2 = d(C_{\alpha,i}, C_{\alpha,i+2})$  is used to define the states: intervals (4.0,6.0) and (6.0,8.0) are considered for d2, and intervals (4.0,7.0), (7.0,9.0), (9.0,11.0) and (11.0,13.0) are considered for d4, such that  $4.0 < d2 < 6.0$  and  $4.0 < d4 < 7.0$  defines a pseudo-helix, while  $6.0 < d2 < 8.0$  coupled with  $7.0 < d4 < 9.0$  defines a pseudo-strand, and six additional pseudo-secondary structure states are defined as other combinations of d2 and d3 intervals, resulting in the total of 9 states (together with ‘OTHER’ state, hence  $N_{\text{PSS}} = 9$ ); solvent accessibility is approximated by the number of contacts (NC) within 8.5 Ang radius around  $C_{\alpha,i}$  (which is capped at 10, hence  $N_{\text{Cont}} = 10$ , although further coarse graining is possible);
- CA-CM (contact map), which is also applicable to both atomistic and reduced models, and consists of the top triangle of the binary contact map, where  $d(C_{\alpha,i}, C_{\alpha,j}) < 8.5$  Ang.

In addition, a specialized RNA secondary structure – torsional angles (TA) or RNA- SS-TA profile is defined in conjunction with the DSSR utility in order to generate the initial secondary structure and torsional angles states for further aggregation by the user.

**Table 2S.** Structural profiles implemented in uQlust.

Name	Type	State Assignment	Number of States	Size
SS-SA	Prot	uQlust	$N_{\text{SS}} * N_{\text{SA}}$	$N_{\text{res}}$
CA(SS)-NC(SA)	Prot	uQlust	$N_{\text{PSS}} * N_{\text{Cont}}$	$N_{\text{res}}$
CA-CM	Prot	uQlust	2	$N_{\text{res}}(N_{\text{res}}+1)/2$
RNA-SS-TA	RNA	DSSR	$N_{\text{SS}} * N_{\text{TA}}$	$N_{\text{base}}$
User generated	Prot/RNA	User defined	User defined	User defined

For each profile, its type (as defined by the macromolecule it applies to, i.e. either protein or RNA), the source of state assignment (uQlust if computed internally, DSSR is used to provide initial secondary structure and torsional angle state assignment for RNAs), the number of states and the size (length) of the profile are reported.

RNA profiles are built by considering a combination of secondary structure/base pairing states with discrete projections of (pseudo-) torsional angles, as generated by DSSR (Lu and Olson, 2003). Specifically, epsilon-zeta BI and BII backbone states are combined with chi syn- and anti- states (plus ‘other’ state), resulting in  $N_{\text{TA}} = 5$  distinct torsional angle states. This is further



combined with either a simplified secondary structure state assignment (stem vs. loop,  $N_{SS} = 2$ ), or a higher resolution assignment of each of the nucleotides forming a bp contact into one of  $N_{SS} = 5$  states (canonical Watson-Crick, Wobble, Platform, Shear, Other), resulting in 5 times 2 (or 5 times 5), i.e., 10 (or 25 distinct states). The former of these two profiles was used for evaluation of RNA-SS-TA profile based clustering and selection of best FARNAs models (Table 3S).

## S5. Implementation

uQlust is written in C# and should be easily portable between different operating systems (system independent pre-compiled executables that require .NET ver. 4.5 or higher, or Mono ver. 3.8 on 64-bit Windows or Linux operating systems, respectively, are provided with the distribution). Multithreading is implemented to speed-up profile pre-processing, ranking and clustering. Work is in progress to enable the use of uQlust (in particular, for profile pre-processing) in conjunction with Hadoop Map/Reduce framework, using the Microsoft Azure plugin for C#. For vector hashing, C# Dictionary Type with a hash function default method GetHashCode() is used. The source code, pre-compiled executables, user manual and a number of utilities and benchmarks/examples are available from <http://github.com/uQlust>.

## S6. Summary of features and functions available in uQlust:

- Traditional K-means and hierarchical clustering using RMSD or MaxSub (Siew et al., 2000) as distance/similarity measures;
- An enhanced K-means approach with 1D-jury based selection of centroids to improve convergence (denoted as uQlust:K-means);
- Several fast and low memory footprint clustering heuristics, including uQlust:Hash(K,F), uQlust:Rpart(K,F), and approximate hierarchical clustering, uQlust:Tree;
- Easy mode: user selects macromolecule, input format and profile type; 1D-jury ranking, Hash, Rpart and Tree clustering are performed automatically (and compared using Rand index to assess consistency of results obtained using these different heuristics) with default setup;
- Ultrafast ranking using an arbitrary, user defined structural profile and a number of well performing predefined profiles;
- Linear time 1D-jury (Adamczak and Meller, 2011) ranking with the contact map (CA-CM) profile, using sparse implementation and hashing key pruning to reduce memory usage;
- Fast RMSD implementation using quaternion approach (Coutsias et al., 2004);
- MaxSub (Siew et al., 2000) and the original 3D-jury approach (Ginalski et al., 2003) for convenient comparison with 3D geometric consensus methods;
- Internal DSSP function for efficient assignment of secondary structure (SS) and solvent accessibility (SA) states based on (Touw et al., 2015);
- Fast, contact-based pseudo-SS-SA profiles for reduced models of proteins, or to speed-up analysis of all-atom models;
- DSSR-based (Lu and Olson, 2003) profile pre-processing and state aggregation for RNA models;
- In addition to applications to ranking and clustering of models of the same length, uQlust can be used in conjunction with arbitrary 1D structural profiles, such as FragBag structural motif frequency projection (Budowski-Tal et al., 2010) for clustering structures of different length;
- Parallelization of profile pre-processing, ranking and clustering using multi-threading;
- Versatile visualization and assessment of clustering results.

## References:

- Adamczak, R. and Meller, J. (2011) *Fast geometric consensus approach for modeling quality assessment*, J Comp Biol 18(12): 1807-1818
- Budowski-Tal, I. et al. (2010) *FragBag, an accurate representation of protein structure, retrieves structural neighbors from the entire PDB quickly and accurately*, Proc. Natl. Acad. Sci. USA 107 (8): 3481-3486
- Coutsias, E. A. et al. (2004) *Using quaternions to calculate RMSD*, J. Comput. Chem. 25: 1849–1857
- Das, R., Baker, D. (2007) *Automated de novo prediction of native-like RNA tertiary structures*, Proc. Natl. Acad. Sci. U.S.A. (104): 14664–14669
- Elmer, S. et al. (2005) *Foldamer dynamics expressed via Markov state models. II. State space decomposition*, J. of Chem. Phys. 123 (11), 114903
- Ginalski, K. et al. (2003) *3D-Jury: a simple approach to improve protein structure predictions*, Bioinformatics 19: 1015-1018
- Jamroz, M. et al. (2013) *CABS-flex: server for fast simulation of protein structure fluctuations*, Nucl. Acids Res. 41: W427-W431
- Lu, X. and Olson, W. K. (2003) *3DNA: a software package for the analysis, rebuilding and visualization of three-dimensional nucleic acid structures*, Nucleic Acids Res. 31(17): 5108-21
- Nugent, T. et al. (2014) *Evaluation of predictions in the CASP10 model refinement category*, Proteins (82) 98–111
- Siew, N. et al. (2000) *MaxSub: an automated measure for the assessment of protein structure prediction quality*, Bioinformatics 16, 776-785
- Touw, W. G. et al. (2015) *A series of PDB related databases for everyday needs*, Nucleic Acids Research 43(Database issue): D364-D368

Wu, S. et al. (2007) *Ab initio modeling of small proteins by iterative TASSER simulations*, BMC Biology (5): 17