Mini Review

Open Access CrossMark

How to cluster protein sequences: tools, tips and commands

Abstract

Proteins are the key molecules that facilitate most biological processes within a cell. Therefore, the discovery, annotation and characterization of them, is of great importance. In System Biology, protein clustering by sequence at a large-scale in order to detect homology, orthology, families, common domains or functional similarities is becoming a great challenge, especially when living in the -Omics era where the exponential growth of sequences produced is indisputable. Despite the great plethora of applications with different strengths to serve this purpose that is available today, a steep learning curve to get familiar with such tools is often required. Users often quit when they get lost in the README files prior to any analysis. To help the community overcome this hesitance, this article describes tools and ways to cluster proteins into groups or families and emphasizes on their basic commands that can be executed in a simple Unix terminal. Notably, both graph-based and sequence-based approaches are described..

Keywords: protein clustering, protein sequences, plasmids, genome

Volume 5 Issue 5 - 2017

Georgios A Pavlopoulos
Lawrence Berkeley National Lab, DOE Joint Genome Institute, USA

Correspondence: Georgios A Pavlopoulos, Lawrence Berkeley National Lab, DOE Joint Genome Institute, 2800 Mitchell Drive, Walnut Creek, CA 94598, USA, Email g.pavlopoulos@lbl.gov

Received: April 04, 2017 | **Published:** June 02, 2017

Introduction

The protein landscape changes continuously as new and hypothetical proteins appear every day. IMG[1] today hosts 55,482 Bacterial genomes, 1,580 Archaeal, 258 Eukaryotic, 1,222 Plasmids, 7,521 Viruses, 1,196 genome fragments and 14,265 private and public met genomes and meta transcriptomes. With a very approximate estimation, this corresponds to ~70Million non-redundant proteins at 100% similarity for the isolate side and ~3billion non-redundant proteins for the met genome/metatranscriptome side (coming from scaffolds of length ~500). Release 15-Feb-2017 of UniProtKB/TrEMBL[2] contains 77,483,538 sequence entries. This number corresponds to 1,465,039 (2%) Archaeal proteins, 49,717,238 (64%) Bacterial proteins, 22,299,253 (29%) Eukaryotic proteins, 2,918,867 (4%) Viral proteins and 1,083,141 (<1%) others. Moreover, Uniparc[3] contains 148,791,725 protein entries. The UniProt Archive (UniParc) is a comprehensive and non-redundant database that contains most of the publicly available protein sequences in the world. Protein families can be characterized by molecules which share significant sequence similarity.[4] Notably, this biological problem is very difficult to solve and most available clustering techniques fail in the case of eukaryotic proteins, which contain large numbers of protein domains.[5] Nevertheless, ongoing efforts in detecting the best and more accurate protein clustering are still a very active research field. PFAM[6] version 31.0 for example, a database of a large collection of protein families, organizes proteins in families by similar domains and includes 16,712 entries. Several tools today, follow various methodologies and strategies to perform protein clustering.[7] Outstanding tools such as the CD-HID,[8] UCLUST,[9] kClust[10] and the newly developed MMSEQ/LinClust[11] follow a k-mer and dynamic programming-based sequence alignment approach whereas tools such as the MCL[12] clustering algorithm and others a network topology based clustering.[13–18] In the second case, prior to clustering, a pairwise similarity matrix is required. While such similarities can be calculated in various ways, BLAST+[19] and LAST[20] are the most widely used. In this article, in order to encourage users getting familiar with several tools and avoid troubleshooting, simple command lines to perform such analyses are provided.

Protein clustering

Input File

Prior to any clustering, organization of protein sequences organized in a FASTA file format is required.

Sequence-based clustering

CD-HIT: It clusters proteins into clusters that meet a user-defined similarity threshold. Each cluster has one representative sequence. The input is a protein dataset in fasta format and the output produces two files: a fasta file of representative sequences (centroids) and a text file (.clstr) with a list of clusters. The basic command to cluster the proteins in the example. fasta file with similarity down to 50% is:

```
cd-hit -i example.fasta -o clusters -c 0.5
-n 2
```

UCLUST: Given a user defined identity threshold, the UCLUST algorithm divides a set of sequences into clusters and is more effective in clustering proteins down to 50% similarity. The steps are:

a. Sort sequences by length

```
uclust --sort example.fasta --output seqs_
sorted.fasta
```

b. Cluster de novo at 50% identity

```
uclust --input seqs_sorted.fasta --uc
results.uc --id 0.5
```

c. Like in CD-HIT, clusters can be stored in a *.clstr* file after reformatting the result. uc file:
```
uclust --uc2clstr results.uc --output
clusters.clstr
```

KCLUST: It is a method to cluster large protein sequence databases such as UniProt within days. It can cluster proteins down to 20%-30% maximum pairwise sequence identity. For example, to cluster a set of proteins proteins down to 50% identity, the basic command is:

```
kClust -i example.fasta -d tmp –s 0.5
```

KCLUST will create a */tmp* folder with the clustering results in it. The *headers.dmp* file will contain the index for each protein, and the *clusters.dmp* file the clustering results in a two-column format. The first column contains the index of centroid sequence for each cluster whereas the second column the index of every member of this cluster.

USEARCH: It is a unique sequence analysis tool with thousands of users worldwide. USEARCH offers search and clustering algorithms that are often orders of magnitude faster than BLAST. A typical command to run USEARCH would be:

```
usearch -cluster_fast example.fasta -id 0.5 -centroids out.fasta –uc clusters.uc
```

In this case, example. fasta is the input file whereas out. fastacontains the centroids for each cluster and *clusters. uc* the final clusters.

MMSEQ/LINCLUST: It claims to be able to cluster billion proteins down to 50% sequence identity in two days. It runs in linear time and has been tested against UCLUST and CD-HIT. The basic commands are:

```
mmseqs createdb example.fasta DB
mkdir tmp
mmseqs cluster/linclust DB clu tmp --min-seq-id 0.5 --target-cov 0.5
mmseqs createtsv DB DB clu clu.tsv
```

All results are stored in *clu.tsv* file. The first column represents the centroid of each cluster whereas the second column every member of this cluster. Thus by using command: `cut -f1 clu.tsv | wc -l`

one can count the number of clusters.

Graph-based clustering

Another approach to cluster proteins into groups is to take advantage of the topological features of a constructed network containing all pair wise similarities in a binary form. To do that, BLAST+ and LAST tools are recommended. BLAST+ does not scale for huge datasets whereas LAST and mega blast[21] tools do. BLAST+ is more sensitive though. For smaller datasets and for much higher quality results at the cost of a much slower speed, ssearch36 or glsearch36 implementation of Smith-Waterman algorithm is recommended.

Similarities

LAST: Step 1: Database building. Assuming that one has a fast a file called example. fasta. First users need to build a database (DB) with the following command:

```
lastdb -p -C 2 DB example.fasta
```

Step 2: A pair wise similarity matrix must be created and used as an input for graph-based clustering. A typical representation would be a tab-delimited format such as:

```
proteinA ProteinB 0.3
proteinA ProteinC 0.2
proteinB ProteinD 0.3...
```

By using last, in order to create such matrix holding information about proteins down to 50% identity the command must look like: `lastal –m200 -pBLOSUM62 -P 0 -f blasttab DB example.fasta | awk '{if($1!~/^#/ && $3>=50) print $1"\t"$2"\t"($3/100)}' > hits.list`

The *hits.list* will be a 3-column tab delimited file (1st column: source, 2nd column: target, 3rd column: identity). Another way to create an adjacency similarity matrix is to use BLASTP instead of LAST.

Step 1: Users must create a blast database using the command: `makeblastdb -in example.fasta -dbtype 'prot' -out DB`

Step 2: The binary adjacency list can be created by querying the *DB* database like: `blastp -query example.fasta -db DB -outfmt '6 std qlen slen' -out output_tmp.list -evalue 1.0e-05`

Hits are firstly stored in a BLASTAB file format called output_tmp.list

Step 3: Similarly to the previous example, users can create the similarity matrix, which will be used as a graph clustering input like:

```
awk '{if($1!~/^#/ && $3>=50) print $1"\t"$2"\t"($3/100)}' output_tmp.list>hits.list
```

Note that if one wants to take into account the best hits only, then he/she might want to filter the file by best similarity first like:

```
sort -nrk 3 hits.list | sort -k 1,2 –u >best_hits.list
```

Network clustering

While many graph clustering algorithms exist, MCL algorithm is by far the most widely used for protein clustering. Notably, MCL is inflation-value sensitive and often memory greedy when it comes to Millions of sequences. To Run MCL with inflation value 1.8, a typical command is:

```
mcl hits.list –I 1.8 --abc -o clusters.txt
```

Clusters are stored in clusters.txt. Each line is a cluster and the members in each line are tab separated. Thus, the number of lines corresponds to the number of clusters. To filter singletons and keep the clusters with two or more members for example, users can filter as: `awk '{if(NF>=2) print}' clusters.txt > clusters_after_threshold.txt`

Similarly to MCL, the highly scalable SPICi density-based clustering algorithm[22] can be utilized like:

```
spici -ihits.list -o out.spici
```

Density is by default set to 0.5. To keep members of clusters greater than a number, the *–s* parameter must be used.

Cluster centroids

For the network-based approach, a potential centroid of a cluster could be determined by the number of hits within a cluster above a certain similarity.

Conclusion

In this article, the very basic commands from various tools to perform protein sequence clustering are presented. While this guide might be of a great help for many users, the commands should not be used blindly as all tools are sensitive upon parameterization and results can vary significantly. USEARCH, UCLUST and CD-HIT should perform nearly the same algorithm. They all cluster the shorter sequences to the longer if the global alignment identity >0.5% is fulfilled. However, kClust, MMSeqs and LAST compute sequence identity in a local way. In order to directly compare clustering between each other, Rand Index, Variation of Information, *F*-Score and other metrics can be used. Such metrics are extensively explained elsewhere.[23]

Acknowledgements

This work was partially supported by the U.S. Department of Energy Joint Genome Institute, a DOE Office of Science User Facility, under Contract No. DE-AC02-05CH11231.

Conflict of interest

The author declares no conflict of interest.

References

1. Chen IA, Markowitz VM, Chu K, et al. IMG/M: integrated genome and Meta genome comparative data analysis system. *Nucleic Acids Res*. 2017;45:D507–D516.
2. Bairoch A, Apweiler R. The SWISS–PROT protein sequence data bank and its supplement TrEMBL in 1999. *Nucleic Acids Res*. 1999;27(1):49–54.
3. Leinonen R, Diez FG, Binns D, et al. UniProt archive. *Bioinformatics*. 2004;20(17):3236–3237.
4. Dayhoff MO. The origin and evolution of protein superfamilies. *Fed Proc*. 1976;35(10):2132–2138.
5. Apic G, Gough J, Teichmann SA. "Domain combinations in archaeal, eubacterial and eukaryotic proteomes. *J Mol Biol*. 2001;310(2):311–325.
6. Finn RD, Coggill P, Eberhardt RY, et al. The Pfam protein families database: towards a more sustainable future. *Nucleic Acids Res*. 2016;4(44):D279–D285.
7. Li W, Fu L, Niu B, et al. Ultrafast clustering algorithms for metagenomic sequence analysis. *Brief Bioinform*. 2012;13(6):656–668.
8. Li W, Godzik A. Cd–hit: a fast program for clustering and comparing large sets of protein or nucleotide sequences. *Bioinformatics*. 2006;22(13):1658–1659.
9. Edgar RC. Search and clustering orders of magnitude faster than BLAST. *Bioinformatics*. 2010;26(19):2460–2461.
10. Hauser M, Mayer CE, Soding J. kClust: fast and sensitive clustering of large protein sequence databases. *BMC Bioinformatics*. 2013;14:248.
11. Steinegger M, Soding J. Linclust: clustering protein sequences in linear time. *bio Rxiv preprint*. 2017.
12. Enright J, Van Dongen S, Ouzounis CA. An efficient algorithm for large–scale detection of protein families. *Nucleic Acids Res*. 2002;30(7):1575–1584.
13. Moschopoulos AN, Pavlopoulos GA, Iacucci E, et al. Which clustering algorithm is better for predicting protein complexes? *BMC Res Notes*. 2011;4:549.
14. Pavlopoulos GA, Malliarakis D, Papanikolaou N, et al. Visualizing genome and systems biology: technologies, tools, implementation techniques and trends, past, present and future. *Gigascience*. 2015;4:38.
15. Pavlopoulos GA, Moschopoulos CN, Hooper SD, et al. jClust: a clustering and visualization toolbox. Bioinformatics. 2009;25(15):1994–1996.
16. Pavlopoulos GA, Secrier M, Moschopoulos CN, et al. Using graph theory to analyze biological networks. *BioData Min*. 2011;4:10.
17. Frech A, Chen N. Genome–wide comparative gene family classification. *PLoS One*. 2010;5(10):e13409.
18. Bader GD, Hogue CW. An automated method for finding molecular complexes in large protein interaction networks. *BMC Bioinformatics*. 2003;4:2.
19. Camacho C, Coulouris G, Avagyan V, et al. BLAST+: architecture and applications. *BMC Bioinformatics*. 2009;10:421.
20. Frith MC. Gentle masking of low–complexity sequences improves homology search. *PLoS One*. 2011;6(12):e28819.
21. Zhang Z, Schwartz S, Wagner L, et al. A greedy algorithm for aligning DNA sequences. *J Comput Biol*. 2000;7(1–2):203–214.
22. Jiang P, Singh M. SPICi: a fast clustering algorithm for large biological networks. *Bioinformatics*. 2010;26(8):1105–1111.
23. Wagner S, Wagner D. *Comparing Clusterings– An Overview*. Universität Karlsruhe (TH); 2007.