MUFOLD Alignment Module User’s Guide

Version 2.0.2

11-October-2015 version 2.0.2

31-March-2015 version 2.0.1(original)

MUFOLD Protein Prediction Group

University of Missouri – Columbia

Computer Science Department

Engineering Building West

Columbia, MO 65201

**Description**

The MUFOLD Alignment Module *User’s Guide* describes how to run and use the various features of the MUFOLD Alignment Module. This guide includes the capabilities of the program, how to use these capabilities, the necessary input files and formats, and how to run the program on Linux Red Hat Server.

MUFOLD Protein Prediction Group, University of Missouri – Columbia

©2015 MUFOLD. University of Missouri. All Rights Reserved

**Introduction**

MUFold Alignment Module (AM) is a software combined third party alignment software. It is for LINUX platforms designed for protein alignment in protein prediction. It is written in C++ programming language and in objected oriented manner so that it is easy to modify and extend. Since protein alignment is a new field, new algorithms and techniques are continually been developed. AM design allows one to integrate and test new algorithms easily. This document describes how to use AM, its features, and the platforms on which it runs.

**Acknowledgement**

BLAST software from NCBI:

<http://blast.ncbi.nlm.nih.gov/Blast.cgi?CMD=Web&PAGE_TYPE=BlastHome>

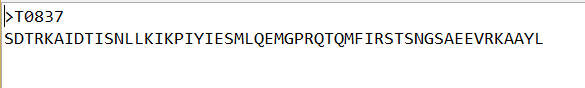
HHSearch software from Bioinformatics Toolkit:

<http://toolkit.tuebingen.mpg.de/hhpred>

**What is needed?**

Before running AM, the following are be needed:

* A protein sequence in fasta format. E.g.



Notice the first row contains the “>” and then the protein name

The second row contains all the protein amino acid, they should be written in one row even if there are a lot of amino acids.

* A json configuration file. If it is blast alignment

There are three blast configuration files

1. configBlastPDB.json



1. ConfigBlastNRC.json



1. ConfigBlastNRR.json



The blast json file should contain following parameters:

alignmentToolLocation: which is the path at which the alignment tool locates

databaseLocation: which is the location of the blast database

experimentLocation: which is the location that user perform the experiment

fastaFileLocation: in which the protein fasta file locates

parameterSetting: which contains all parameter settings user prefer the blast run in some fashion. Here are some parameters used in MUFOLD alignment module:

-o: output file for alignment [File Out]

-R: input file for PSI-BLAST restart [File In]

-Q: output file for PSI-BLAST Matrix in ASCII [File Out]

-d: database [String]

-e: expectation value [Real]

-v: number of database sequences to show one-line descriptions for V [Integer]

-b: number of database sequences to show alignments for B [Integer]

-j: maximum number of passes to use in multi-pass version [Integer]

-h: e-value threshold for inclusion in multi-pass model [Real]

-i: Query File [File In]

-C: Output File for PSI-BLAST Check-pointing [File Out]

Or if it is hhsearch experiment:

configHHSearch should be provided:



alignmentToolLocation: the hhsearch tool location

databaseLocation: the database location that hhsearch tool used

experimentLocation: where user perform experiment

fastaFileLocation: where protein fasta file locate

-a3m: output format: all residues upper case, but gaps aligned to inserts omitted

Notice there are another two parameter settings, they are for HHMake and HHSearch main program, respectively.

For HHMake parameter setting:

-i: query alignment (should in A2M, A3M, or FASTA format), or query HMM

-o: HMM file to be written to

-name: use this name for HMM

For HHSearch main program:

-i: input query alignment

-d: HMM database of concatenated HHMs in hhm

-o: write results in standard format to file

-additional: use global/local alignment mode for searching/ranking

If it is CNFSearch, the configuration file look like this:



AlignmentToolLocation: where executable file locates

databaseLocation: where CNFDB locate

experimentLocation: where user run their experiment

fastaFileLocation: where protein fasta file locates

build feature parameter setting portion is to build feature file for CNFsearch:

-i: input fasta file, here, program will get it from root name

-c: cpu number

-o: output target file, here, program will generate the output target file and the filename should use root name.

CNF search parameter setting are as followed:

-a: number of processors

-q: query protein name. Query name.tgt should be already available in the folder specified by tgt\_root

-p: pval. Keep the results for top templates according to p-value cutoff.

-g: tgt\_root The folder containing the target protein feature files.

-o: output file the file containing a brief summary of the prediction results.

-d: the folder containing the template files

-l: the list of templates

-n: keep the results for top topN temlpates.

**How to run?**

1. Download all the code from <https://github.com/rafaelfang/TestJson> to your directory under server which has Linux platform installed.

For example:

All the code are on /home/cf797/test/AlignmentModule

1. Generate the json configuration file for your blast/hhsearch/cnfSearch experiment following the examples provided above. Those json files should be in the same directory as the program directory
2. Create the experiment directory as you mentioned in the json file. For example: I create testAlignment folder under test directory to put all alignment results in.
3. Compile the code and generate an executable file by typing the following command:

Make clean

Make

1. If no compile errors, there will be a executable file generated named “mufoldAlignment”
2. Here are some valid example to run the experiment

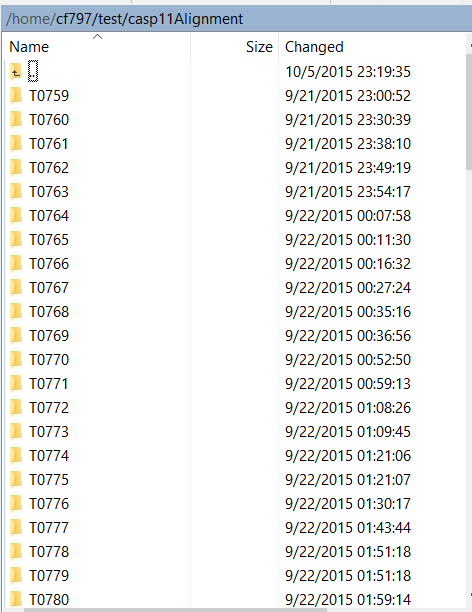
./mufoldAlignment -blast T0759

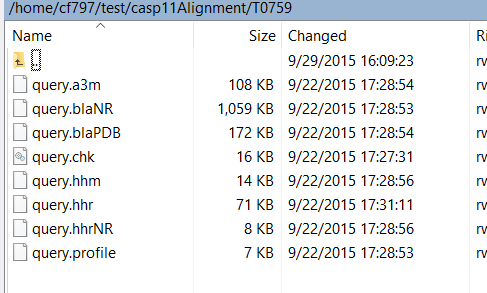
./mufoldAlignment -hhsearch T0759

./mufoldAlignment -cnfsearch T0759

**What is the output you can get?**

After running the experiment, go to your experiment folder to get the experiment results. The blast will generate the results which look like: “something.blaPDB”. The hhsearch will generate the results which look like “something.hhr”. There should also be some other intermediate results such as “something.a3m” (the alignhit command), “something.chk” (blast command), and “something.chk” (blast command), etc.





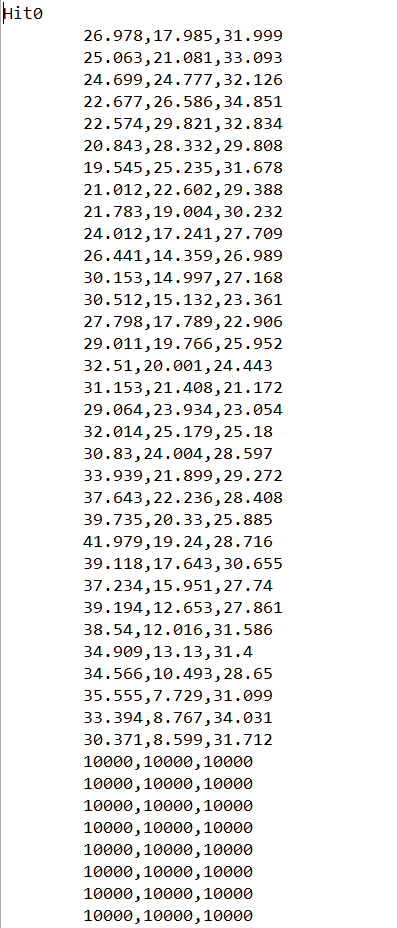
**How to analysis results?**

The alignment results are stored in the txt file and all alignment hits are one by one stored together. The CalculateTMScore module is used to parse the result files and change it into json format so that it is convenient to use. It will also generate 3D coordinates for the alignment hit of the alignment result file.

The alignment result file is parsed to the NFA (Nondeterministic finite automaton) to get each individual hit. They are then stored into json format.



For future protein structure prediction purpose, the 3D coordinates for the subject part are taken from the PDB database and then used as the template for query. So their 3D coordinates are stored into text file.



Each hit represent each alignment. For example, the hit0 is the 3D coordinate list for T0759 alignment hit with hitName 1LM7\_A. The subject part is used to fetch 3D coordinates and then used for query part.

**The class diagram of alignment module**

CNFSearch

HHSearch

Blast

Alignment Tool

Query sequence

Alignment module

Template 3D coordinates

Json format

Alignment Result