MUFOLD Utility Module User’s Guide

Version 2.01

28-April-2015

MUFOLD Protein Prediction Group

University of Missouri – Columbia

Computer Science Department

Engineering Building West

Columbia, MO 65201

**Description**

The MUFOLD Utility Module *User’s Guide* describes how to run and use the various features of the MUFOLD Utility 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 Utiltiy Module (UM) is the software that prepare and convert results generated from Alignment Module (AM). 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. UM design allows one to integrate and test new algorithms easily. This document describes how to use UM, 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>

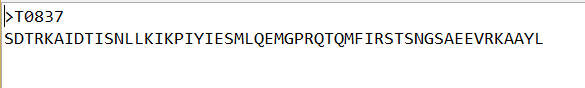
HHSearch software from Bioinformatics Toolkit:

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

**What is needed?**

Before running UM, the following are be needed:

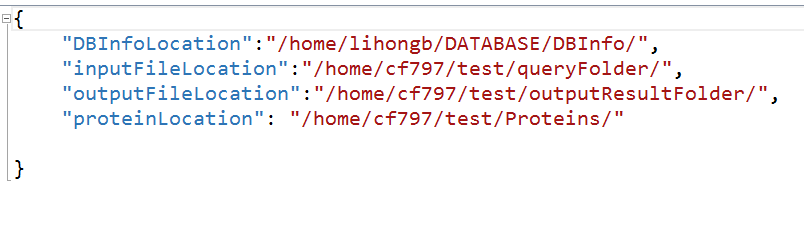
* A protein sequence in fasta format. E.g.



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. This file should be put in a folder that is independent from the source code folder. In my example, all fasta files are put under Protein folder.

* A json configuration file. E.g.:



The json configuration file should contain following parameters:

DBInfoLocation: which is the location of all protein datbase files

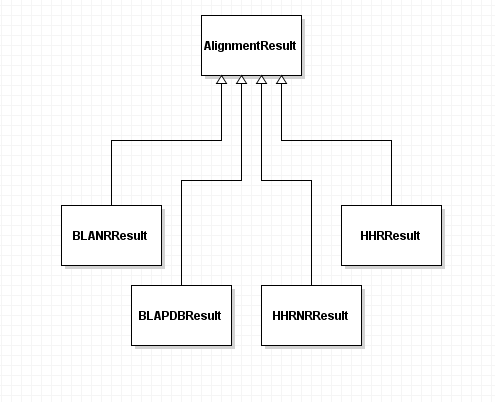
inputFileLocation: all the query results generated from the blast and hhr tools are stored here.

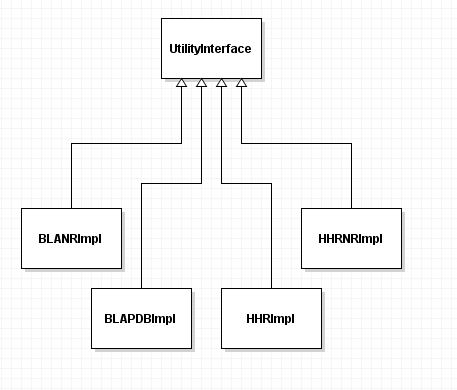
outputFileLocation: in which all the results generated written into it

proteinLocation: which is the path at which the protein sequence file locates

This json configuration file should be put in the same folder where the source code locates. This json configuration file should load once at the beginning of the program and should not be changed after that.

All the source code should be put under one folder, also the json folder should be put within the source code folder. The structure of the code is as follows:





The AlignmentResult class is the base class of all kinds of alignment tool genereated results. All other derived classes should inherent from this class. Since UM servers as the middle module between AM and Model Generation Module (MGM), 3D coordinates from each alignment results should be stored as intermediate results. For convenient purpose, all the coordinates are stored in the vector of floats.

The BLANRResult class is inherited from its base class, i.e. AlignmentResult class. This class can be used to store a record from query.blaNR result file. So this class handles all the information extracted from one blaNR record from query.blaNR file, such as protein name, protein length, and so on.

The other results class follows the same idea.

The UtilityInterface class is the base class for all converting classes. In this class, it defines all the function interfaces and defines all the variables from the configuration file.

The BLANRImpl class is the class that implements the abstract methods from UtilityInterface class. It does major three jobs: convert the results in query.blaNR into json format; find the local alignment coordinates for template; find the global alignment coordinates for template.

The other impl classes follows the same idea.

**How to find the local alignment 3D coordinates?**

Given a query, the local alignment will be the template searched by the blast or hhr alignment tool. The query is partially aligned with the partial template. Sometimes, gaps are involved. The coordinates are taken from the partial template. If the gap exists, use 10000 to be the coordinate.

**How to find the global alignment 3D coordinates?**

Given a query, the global alignment being extended is based on local alignment. The number of points to be extended is depending on how many “head” or “tail” points beyond the local aligned part of the query.

**How to run?**

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

For example:

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

1. Generate the json configuration file for your experiment following the examples provided above. Those json files should be in the same directory as the source code directory
2. Create the queryFolder directory to put all protein alignment results files into it. You should also create individual folder for each protein. For example: I create queryFolder folder under test directory to put all protein query files in. /home/cf797/test/queryFolder/T0837
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 “convert”
2. Here are some valid example to run the experiment

./convert –blaPDB T0837

./convert –blaNR T0837

./convert –hhr T0837

./convert –hhrNR T0837

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

After running the experiment, go to your experiment folder to get the experiment results. The outputResultFolder will be generated and all results will be inside. If you open this folder, inside it, there will be a folder with the protein root name you just enter from the command line. Open that folder, you will find two folders and one file with extension name “.json”. This is when you convert the blaPDB and hhr files. If you convert blaNR and hhrNR, only the json results are generated.

**How to analysis results?**

For the json file result, it is for the whole MUFold system use. If user wants to anaylze the intermediate blast or hhsearch result, the json file are in good format and for other testing program to read in from.

For the global and local alignment results, each template are stored in its independent file that contains the protein name and the 3D coordinates. They can be parsed to the model generation module and other modules for further protein prediction purpose.

For efficient usage, user can use the 3D coordinate vectors directly rather than read the results from json file. This is because it is faster to pass result via memery than I/O from disk.