To install , do this:

mkdir ~/svn/breeder

svn checkout https://simtk.org/svn/breeder   ~/svn/breeder

Your LD\_LIBRARY\_PATH should point to wherever you have libMMBlib.so, e.g. :

export LD\_LIBRARY\_PATH=~/svn/RNAToolbox/trunk/build

You will then need to compile:

cd ~/svn/breeder/build

ccmake ..

[samuelf@tintin1 build]$ pwd

/bubo/home/h17/samuelf/svn/breeder/build

[samuelf@tintin1 build]$ make -j8

Linking CXX executable breeder

[100%] Built target breeder

[samuelf@tintin1 build]$

To update your breeder distribution, do:

cd ~/svn/breeder

svn update

To run, use:

./breeder -ID 14MAY13 -MLF ./mutations-list.txt

Where the file specified by –MLF contains as many mutants as you wish, in the format : X-NNI-S.x-nni-s where X is the chain ID, NNI is the residue ID and insertion code (if any) and S is the substituted residue type. Separate the mutants with “.” For example:

C-128-V.C-147-H.C-163-N

C-128-R.C-147-H.C-163-N

Breeder will fetch all run parameters (except the actual mutations) from the mysql jobInfo table:

mysql> select \* from jobInfo;

+--------------------------------------------------------+--------------------------------------------------------------+------------+----------------+----------------+---------------------------------------+------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------+-------------------+-----------------------+-----------------------------------------------------------+

| runID | batch\_directory | maxMutants | generationSize | maxGenerations | decayConstantsBetweenHighestAndLowest | wildTypeSequence | reportingInterval | numReportingIntervals | wildTypeStructureFileName |

+--------------------------------------------------------+--------------------------------------------------------------+------------+----------------+----------------+---------------------------------------+------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------+-------------------+-----------------------+-----------------------------------------------------------+

| 14MAY13 | /bubo/home/h17/samuelf/glob/projects/protein-A/14MAY13 | 4 | 2 | 1000 | 5 | C:FNKEQQNAFYEILHLPNLNEEQRNGFIQSLKDDPSQSANLLAEA.D:PSVFLFPPKPKDTLMISRTPEVTCVVVDVSHEDPQVKFNWYVDGVQVHNAKTKPREQQYNSTYRVVSVLTVLHQNWLDGKEYKCKVSNKALPAPIEKTISKAKGQPREPQVYTLPPSREEMTKNQVSLTCLVKGFYPSDIAVEWESNGQPENNYKTTPPVLDSDGSFFLYSKLTVDKSRWQQGNVFSCSVMHEALHNHYTQKSLSLS | 1 | 15 | /bubo/home/h17/samuelf/glob/projects/protein-A/last.1.pdb |

+--------------------------------------------------------+--------------------------------------------------------------+------------+----------------+----------------+---------------------------------------+------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------+-------------------+-----------------------+-----------------------------------------------------------+

7 rows in set (0.00 sec)

mysql>

You can change reportingInterval and numReportingIntervals on jobInfo, but as yet not other MMB parameters. If you wish to modify MMB’s other parameters, to go into commands.dat, modify src/Chromosome.cpp method writeCommonPart:

void Chromosome::writeCommonPart(std::ofstream & output, MysqlConnection & mysqlConnection) {

output <<"firstStage 2 "<<std::endl;

output <<"lastStage 2"<<std::endl;

output <<"loadSequencesFromPdb "<<std::endl;

output <<"smallGroupInertiaMultiplier 11 "<<std::endl;

output <<"#reportingInterval .000000000000000000000001"<<std::endl;

output <<"reportingInterval ";

output <<mysqlConnection.getReportingInterval();

output <<std::endl;

output <<"numReportingIntervals "; //150 "<<std::endl;

output << mysqlConnection.getNumReportingIntervals();

output <<std::endl;

output <<"#numReportingIntervals 1 "<<std::endl;

output <<"readAtStage 2"<<std::endl;

output <<"globalBondTorsionScaleFactor 1"<<std::endl;

output <<"globalBondBendScaleFactor 1"<<std::endl;

output <<"globalCoulombScaleFactor 1"<<std::endl;

output <<"globalVdwScaleFactor 1"<<std::endl;

output <<"globalAmberImproperTorsionScaleFactor 1"<<std::endl;

output <<"readBlockEnd"<<std::endl;

output <<"randomizeInitialVelocities false"<<std::endl;

output <<"mobilizer Rigid"<<std::endl;

output <<"constrainChainRigidSegments"<<std::endl;

};

You will then need to recompile:

[samuelf@tintin1 build]$ pwd

/bubo/home/h17/samuelf/svn/breeder/build

[samuelf@tintin1 build]$ make -j8

Linking CXX executable breeder

[100%] Built target breeder

[samuelf@tintin1 build]$

## Starting a new job

First, go to the jobInfo table and specify the parameters for the job, e.g.:

insert into jobInfo (runID , batch\_directory , ,reportingInterval ,numReportingIntervals , wildTypeStructureFileName) VALUES (“17MAY13” , “/bubo/home/h17/samuelf/glob/projects/recA/17MAY13” , 0.1, 150, “/bubo/home/h17/samuelf/glob/projects/recA/last.1.pdb”);

Now put all your mutants in a text file. Name whatever you want (e.g. myMutants.txt). Then run as:

./breeder -ID 17MAY13 -MLF ./myMutants.txt

## Useful MySQL commands:

Include update and delete

update jobInfo set lastStage = 3 where runID = “14MAY13”;

Be very careful with the delete command! Always run it with a carefully selected “where” restriction:

delete from jobInfo where runID = “14MAY13”;