BUDE

1.2.0

Bristol University Docking Engine

authors

Amaurys Avila Ibarra

James Price

Deborah Shoemark

Simon McIntosh-Smith

Richard Sessions

University of Bristol

User Manual

BUDE is released under licence

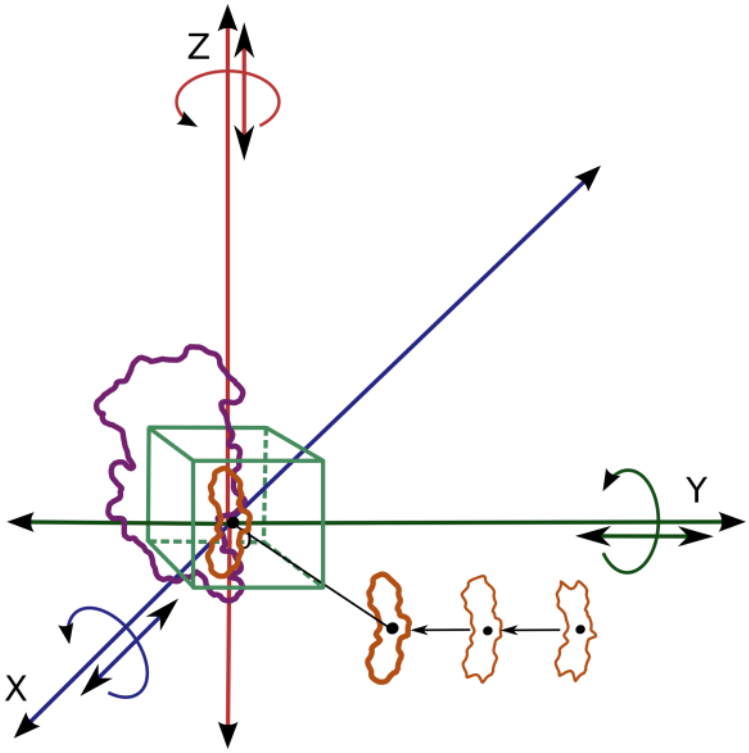
Contents

Introduction

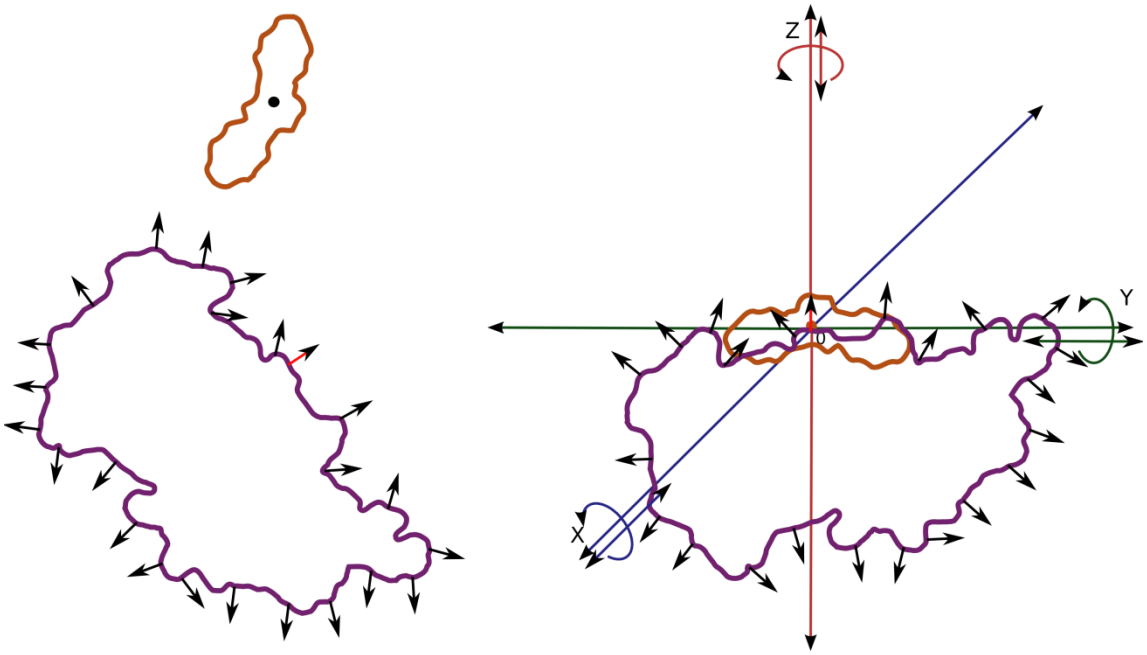
BUDE is a molecular docking program that is straightforward to use in any one of three docking modes, but is also highly customisable by the user. Currently, the program uses a rigid docking method, hence receptor and ligand flexibility can only be handled by multiple docking between different conformations of receptor and ligand. The exception to this general rule is that multiple side chain rotamers of protein receptors or ligands can be selected and combined to give a mean-field interaction energy for these structural elements. The pose search is performed via a limited genetic algorithm (Evolutionary Monte Carlo) and receptor-ligand interaction free energies are estimated via a novel atom-atom based empirical free-energy forcefield. The same forcefield is used for docking as for scoring the best poses.

It is convenient to think of BUDE as providing three distinct docking modes. These are outlined in this section and described in detail in section XXX. The first mode is “virtual screening by docking” whereby a large number of small molecules (typically in the range 1 up to 10^7 compounds) each with multiple conformations (typically in the range 1 to 50) are docked into a known binding site on the receptor (usually a protein). As illustrated in Figure 1a. The second mode is “binding site prediction”. In this case a small number (typically 1 to 100) of small molecules of interest, each with multiple conformations (typically 1 to 50) are docked to small patches of the surface of a receptor (usually a protein). The centres of these surface patches are generated using Connolly’s SASA (solvent accessible surface area) program and the corresponding surface normal is used to define the Z axis of the docking grid. A limited docking search to each patch is performed to locate the best binding sites, as illustrated in Figure 1b. The third docking mode is “protein protein” docking where both the receptor and ligand are protein-sized molecules, both have solvent-accessible-surfaces and these are used to dock every surface patch on the receptor with every surface patch on the ligand. This approach is much slower than methods that use simple surface matching via comparisons of the surfaces in reciprocal space via FFT transformations, but has the advantage that interactions are calculated in real space. Likewise, as the software is developed, sidechain packing will be included in the pose searching.

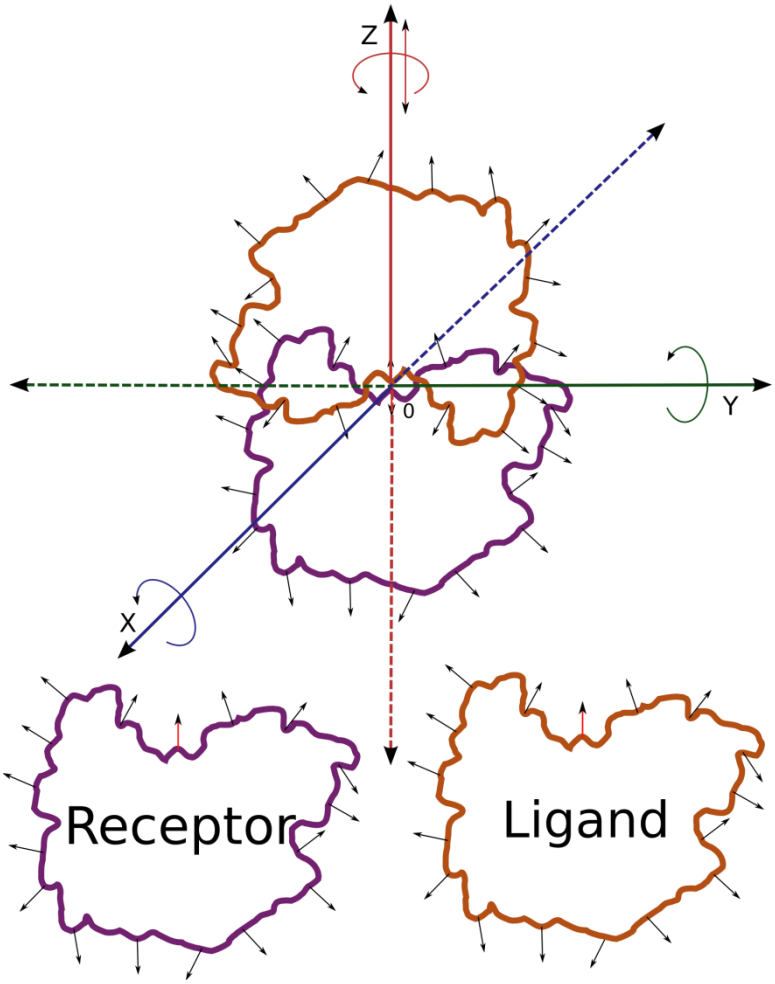
Examples of all three docking modes are provided with the software release. The user should install and make use of these examples as they have the standard docking grids and search definition files for each docking mode. The examples are referred to often in this documentation and form the basis of the tutorial section.



(A)



(B)



(C)

Figure 1 *(a) virtual screening by docking; (b) binding site prediction; (c) protein-protein docking.*

1.1 ***Command line options and the control file:*** the behaviour of BUDE is determined by a set of command line options. Available options can be listed by running **bude -h** and most options have a short and a long version (e.g. **bude -h** is the same as **bude --help**). Typically, BUDE is run by passing a set of options, and arguments to these options, in a text file that, by convention, has the file name extension **.bctl**. A template input control file may be generated by running **bude -B** (or **bude --control-file-template**). The template control file contains comments relevant to each option, and the user may customise these comments as any line beginning with # will be ignored by BUDE. If options are given multiple times, BUDE will use the last instance specified in the control file. Likewise, the presence of an option on the command line will override its value in the control file.

1.2 BUDE docking modes The three distinct docking modes of BUDE are functionally distinguished by the inclusion, or not, of receptor/ligand surface files in the docking procedure:

1.2.1 Virtual Screening by Docking: no surface files are used in this mode. It is assumed that the general location of the binding site on the receptor is known. A file defining the search grid must be provided, via option **-T**, of type **.bltr** which defines the extent of the docking grid in x, y and z (typically a regular 15 x 15 x 15 grid of 1 Å spacing, centred around 0,0,0) and the number of rotational orientations available to the ligand at a grid point (typically 36 x 36 x 36 in 10° increments). The initial (or zero) generation of the EMC is either read from an existing file with the option **-Z** (a transformation descriptor file of type **.tds**) or the number of poses to create for generation zero is specified by the option **-z**. The default method of pose creation for generation zero (random) may be reconfigured by using the appropriate options (**-x** (exhaustive), **--unique-tds-gen0** (all poses unique)). The contents of the EMC file (of type **.bemc**) control the rest of the genetic algorithm parameters and the writing of results at any point during the EMC process. This file is passed with the option **-V**. The receptor filename is passed with **-P** and may be of type **.mol2** or **.pdb**. The ligand file is passed with option **-I** and may be of type **.mol2** or **.pdb** or **.blst** (a list of either pdb or mol2 files (not a mixture)).

These are the basic requirements of a job running a virtual screen or small molecule docking. All other options refine the behaviour of the job. The reader is advised to see the relevant tutorial section(), the bude options reference section (), and the file format section ()

File Formats

General BUDE file formats: All BUDE data and input/output files are ASCII formatted and human readable. Any data appearing after a # (hash) will be ignored allowing for full-line and in-line commenting. Likewise, all blank lines are ignored. All BUDE format files have a header line or lines beginning with % (percent) which determine the file type for the program to read appropriately, and these data are terminated by a line containing just END. The exception to the header/footer rules is the BUDE control file since this is a (commented) ASCII list of options and arguments in any order. All data on lines in these files are space-separated items. Each file type has a 4 letter file type extension (e.g. **.bctl**) note that these are conventions, the program does not require these, but they are designed to make the users life easier.

External formats: BUDE adheres strictly to the fixed column, 80 character wide format for PDB files. A utility called **bude_padPDB** is provided to pad any PDB files that do not conform to the 80 character wide record standard. Sybyl MOL2 format files are read and written, conforming to the standard space-separated data format.

Extended format: An extension to the MOL2 standard has been implemented for packing multiple conformations more efficiently. The new token is @<BUDE>CONF and signals a set of updated coordinates. A utility called **bude_formatter** is provided to pack and unpack multiple MOL2 files. This can reduce the size of large databases of small molecule conformations up to 4 fold.

Specific BUDE file formats:

bude_control_file.bctl

This type is the exception to the rule, having neither header nor footer. It comprises a list of BUDE options and arguments where appropriate. It is passed to BUDE with **-f**

bude -f *bude_control_file.bctl*

A template can be generated with the option **-B**

bude -B

The following rules of precedence pertain: (1) the last instance of an option in the control file is used. (2) options passed on the command line override those in the control file. See section “BUDE option Reference” for details of the options.

transformation_descriptor_file.tds* or *.bzgn

This is a typical “zero generation” transformation descriptor file:

```
%BUDE TRANSFORMATION DESCRIPTOR FILE
%VERSION  2.0
# The first line of data comprises three values val1 - val3
# subsequent, 'val3' number of lines are one transformation descriptor per line
# the file is terminated with a line containing just END
#
# val1 is redundant. each state should be white space separated
# val2 is the number of states comprising a descriptor. Range = 1-6, this  must
# be the number of states in the transformation library and is never changed,
# namely
#      Rotations: "TILT","ROLL","PAN"  == (XROT, YROT, ZROT)
#      Translations: "XTRANS","YTRANS","ZTRANS"
# val3 is the number of descriptors in this file. Range = 1-4294967295
#
# The descriptors values are right aligned and should have at least one blank
# space in between them.
# The order of the states must not be altered, it MUST always be
# TILT ROLL PAN XTRANS YTRANS ZTRANS
#
# This was generated using transformation library
# inputs/trans_rot_drug_small.bltr
# The seed used was [10142925].
#
# The TDs were NOT uniquely generated.
#
             7             6         20000
     30      22       5      6       1     10
     26      19      23     12       7     12
     13       7      35     14       1      2
......
...... {20000 lines}
......
     15      31      30     13       7      3
     32      13      26     12       4      7
END
```

translation_rotation_definition_file.bltr

This file defines the actual transformations that are applied to the ligand, corresponding to the indices present in the transformation descriptor file. Another way to think about this is it is

the definition of the 6 dimensional docking grid. A single ligand pose is completely defined by: (1) its initial coordindates passed to BUDE; (2) the translation and rotation definition file; (3) the transformation descriptor; as described in section XXX

```
%BUDE TRANSFORMATIONS LIBRARY
%VERSION 2.00
# The above 2 lines are required, only the floating point number can vary.
# Any line with a # as 1st character is a comment line and ignored, but
# no further comments are allowed once the data entries have started.
#
# The file contains a set of translations and rotations from the ligand
# starting point. Translations are given in Angstroms. Rotations are
# given in degrees.
#
# TILT == rotation in X ; ROLL == rotation in Y ; PAN  == rotation in Z
# *IMPORTANT :
#     THE ORDER IN WHICH THE TRANSFORMATIONS ARE LISTED WILL BE THE ORDER
#     IN WHICH BUDE WILL CARRY OUT THE TRANSFORMATIONS.  Make sure that
#     rotations  are first, and translations are second.
# Format is 6 x (TOKEN N, followed by N lines of angles or distances)
TILT   36
-170.0
-160.0
......
...... (36 lines)
......
 170.0
 180.0
ROLL   36
-170.0
-160.0
......
...... (36 lines)
......
 170.0
 180.0
PAN    36
-170.0
-160.0
......
...... (36 lines)
......
 170.0
 180.0
XTRANS 15
  -7.0
  -6.0
......
...... (15 lines)
......
   6.0
   7.0
YTRANS 15
  -7.0
  -6.0
......
...... (15 lines)
......
   6.0
   7.0
ZTRANS 15
  -7.0
  -6.0
......
...... (15 lines)
......
   6.0
```

```
   7.0
END
```

evolutionary_monte_carlo_control_file.emc

This file controls the behaviour of the EMC genetic algorithm. The first line of data, after the header and comments, contains the number of generations. Each subsequent line contains the parameters to describe one generation.

```
%BUDE EMC FILE
%VERSION  2.00
# Remember that the 2 previous lines are required,
# any other lines starting with # are comments.
#
# %VERSION 2.00: Will allow comments at any position within the file. Any part
# of a line after a '#' character will considered a comment. It will also
# ignore blank lines. The tag END will denote the end of the library. No further
# lines will be considered. The line containing the END tag can only have the
# tag and blank characters or a comment.
#
# If the END tag is omitted the file will be processed until EOF and a warning
# will be issued.
#
# Each column in each generation must be space separated.
#
# Column 1: Seed. Is an integer to seed the random number generator.
#
# Column 2: Number of Parents. Is the number of descriptors that will be used
# to generate the Children of the current generation.
#
# Column 3: Parent's Energy. It the number of parents that will have their
# energy information printed.
#
# Column 4: Parent's Coordinates. The number of Parents that will have the
# coordinates printed to a file, PDB or Mol2.
#
# Column 5: Children Number. We tell BUDE how many Children to produce from
# the Parents. If the size of the generation is greater that 50% of the total
# it will be reduced to 50%.
#
# Column 6: Children Energy. Will tell BUDE how many descriptors info should
# be output for the Children.
#
# Column 7: Children Coordinates. Will tell BUDE how many Children will have
# their structural information written to a file.
#
# Column 8: Mutation Method. Will tell BUDE how the descriptor should
# be generated.
# 1 will generate the requested number of descriptors and some maybe repeated.
# 2 will generate the requested number of descriptors and ensure that all are
# unique. If the number requested is greater that 10%, it will be reduced
# to 10% of the total possible number of descriptors.
#
# Column 9: Max Mut States. Is the number of states that can be mutated in a
# descriptor.
#
# Column param2 & param3: are reserved, doing nothing for the moment.
```

```
#_____________________________________________________________________________________________________________
# int     | int      | int      | int      | int       | int       | int       | char     | int      | double   | double   |
# use     | select   | print    | print    | create    | print     | print     | choose   | choose   | reserved | reserved |
# Seed    | Parents  | Parents  | Parents  | Children  | Children  | Children  | Mutation | Max Mut  | param2   | param3   |
#         | Number   | Energy   | Coord    | Number    | Energy    | Coord     | Method   | States   |          |          |
#_________|__________|__________|__________|___________|___________|___________|__________|__________|__________|__________|
 10
#_____________________________________________________________________________________________________________
70772915        300         0          0        6000         0          0         1         4        2.00       3.00
18188928         60         0          0        3000         0          0         1         4        2.00       3.00
19142935         30         0          0        3000         0          0         1         4        2.00       3.00
77772945         30         0          0        3000         0          0         1         4        2.00       3.00
16188955         30         0          0        3000         0          0         1         4        2.00       3.00
15142965         30         0          0        3000         0          0         1         4        2.00       3.00
74772975         30         0          0        3000         0          0         1         4        2.00       3.00
13188985         30         0          0        3000         0          0         1         4        2.00       3.00
72772995         30         0          0        3000         0          0         1         4        2.00       3.00
41442905         30        30          0        3000        10          1         1         4        2.00       3.00
END
```

bude_forcefield_file.bhff

The forcefield file is used to parameterise each atom. The forcefield file with just a WLD (wild card) residue type defined will parameterise by atom type according to Tripos/Sybyl .mol2 atom types in the coordinate file. Discrete residues types can also be defined (e.g. for amino acids) allowing direct parameterisation of proteins by residue type and atom name (but generally this is deprecated legacy usage). Development is towards a unified atom-type based free energy forcefield.

```
%BUDE HEAVY ATOM AMINO-ACID FORCEFIELD LIBRARY
%VERSION 2.00
#
# Generic Mol2 atom-types base parameters
# *** Generated by FEFFO ***
#
# The above 2 % lines are required, only the floating point number can vary.
# Any line with a # as 1st charcter is a comment line and ignored, but no
# further comments are allowed once the data entries have started.
#
# The first data entry is an integer in the first 10 characters of a new line
# defining the number of residue types in this file.
# Each residue type has a 3 character identifier in column 1 and the number
# of ball types as an integer in the next 10 characters.  The three
# character identifier is used to match amino acids in a pdb file.
#    residue type WLD is a wild card residue, in this case atom types are used
#    to parameterise atoms whose Residue id's do not match any other residue
#    type in the forcfield file
#
#    Entries are space separated
#
# Elect type defines the type of electrostatic energy - i.e.
#     D if it is a dipole induced charge. e.g. for backbone amide hydrogens
#     F if it is a formal charge. e.g. for Lysine NH3+ terminus
#     E if it is capable of being either a donor or an acceptor (eg -OH)
#
#     as _1_ but with Ca and Zn radii 0.42 not 1.42 and N.3 charge 0.20470 from 0.10470
#     and Cl hydrophobic potential to -0.31719 from -0.21719
#     and N.1 re-paramaterised
#     As ff003 but with N.ani added for anionic nitrogens (eg in sulphonamides)
#             Koike et al J. Am. Chem. SOC. 1992, 114,1338-1345
#     And added S.1 as a double-bonded sulphur with a minor -0.2 charge...
#     Added K and Na ions 16 May 2014
#
#
#      Elect          Hydrphbic             Np-Np     Np-P     Radius   Electsttc
# Atom type   Radius  potential  Hardness   dist      dist     scaling  potential
#vvvv   v       v         v         v         v         v         v         v
   1
WLD 42
 C.3    -     1.42000  -0.15007  30.00000   6.50000   1.30000   1.00000   0.00000
 C.2    -     1.42000  -0.24992  30.00000   6.50000   1.30000   1.00000   0.00000
 C.1    -     1.42000  -0.15573  30.00000   6.50000   1.30000   1.00000   0.00000
 C.ar   -     1.64000  -0.09309  30.00000   6.50000   1.30000   1.00000   0.00000
```

```
 C.cat  -     1.42000  -0.22992  30.00000   6.50000   1.30000   1.00000   0.00000
 N.4    F     1.73000   0.43704  30.00000   6.50000   1.30000   1.00000   1.00000
 N.3    E     1.59000   0.13344  30.00000   6.50000   1.30000   1.00000   0.20470
......
......  (42 lines)
......
 B      -     1.42000   0.06440  30.00000   6.50000   1.30000   1.00000   0.00000
 Sn     -     1.42000  -0.15007  30.00000   6.50000   1.30000   1.00000   0.00000
END
```

bude_surface_file.bsrf

The surface file is essentially Connolly (msnew) format with a header:

```
%BUDE CONNOLLY SURFACE FILE
%VERSION 2
# the first three fields are the atom numbers of the atoms the probe was
# touching when it generated the given surface point. The first number is
# the atom whose van der waals surface the point is closest to. The fourth
# field is the number of atoms the probe was touching. If this number is
# less than three, one or both of the second and third fields will be zero.
# The fifth, sixth and seventh fields are the coordinates of the surface point.
# The eighth field is the molecular area associated with the surface point.
# The ninth, tenth and eleventh fields are a unit vector perpendicular to the
# surface pointing in the direction of the probe center. If the buried surface
# flag equals 2, there is a '1' written at the end of the record for buried
# surface points, and a '0' for accessible points. (From comments in msnew.f)
#
   33   49 1818 3   -5.403   -0.863  -14.696   9.4247704  0.701  0.509 -0.500 0
   34 2211 2523 3   -0.262    0.618  -10.661   9.4247704  0.000  0.000 -1.000 0
   48 1817    0 2   -6.263    0.107  -17.597   2.4182255  0.962  0.256  0.090 0
   48  936  952 3   -4.837   -1.590  -19.732   9.4247704  0.701  0.509  0.500 0
......
......  (some number of lines)
......
 2671    0    0 1   28.236   -4.780  -32.365  16.0849552  0.000  0.000 -1.000 0
 2672 2647    0 2   29.216   -3.196  -29.040   9.9129915  0.914  0.126  0.386 0
END
```

rotamer_library.brot

This file contains the Cartesian geometry of the sidechains of amino acid residues comprising the library of rotamers used by BUDE. These atomic coordinates refer to a standard position of the amino acid backbone atoms defined as follows: CA at (0,0,0); N on X axis at (-x,0,0) ; C in XY plane at (x,y,0).

(NB rotamers are only used in the sidechain mean-field calculations in this version of the software)

```
%BUDE ROTAMER LIBRARY
%VERSION 2.00
# Remember that the 2 previous lines are required,
#
# Each configuration set will start with the 3 character representation of the
# amino acids plus 3 integers, the first one is the index of the configuration
# the second is the number of configurations, and the third the number of atoms
# in the amino acid. The number of atoms MUST be present in the start of the
# configuration.
#
# Each configurations is followed by the atom name and the XYZ coordinates.
# The data after the Z coordinate will be ignored.
#
# %VERSION 2.00: Will allow comments at any position within the file. Any part
# of a line after a '#' character will considered a comment.It will also
# ignore blank lines.
#
# %VERSION 2.00: The tag END will denote the end of the library. No further
```

```
# lines will be considered. The line containing the END tag can only have the
# tag and blank characters or a comment.
#
# If the END tag is omitted the file will be processed until the EOF and a warning
# will be issued.
#
#amino  index  conf   atoms
ALA      1      1      1
CB       0.465990805    -0.817742225    -1.217112304
ARG      1      7      7
CB       0.503425220    -0.800236441    -1.232645708
CG       2.017135691    -1.141228346    -1.216638172
CD       2.461444346    -1.904547743    -2.471084280
NE       3.910064123    -2.206189968    -2.345988163
CZ       4.640620496    -2.858715298    -3.240078948
NH1      4.181279652    -3.324822899    -4.363548823
NH2      5.885100083    -3.039551498    -2.971324526
ARG      2      7
CB       0.503425220    -0.800236441    -1.232645708
CG       2.039603523    -1.011181228    -1.291889039
CD       2.480279515    -1.776736649    -2.546253867
NE       3.957297762    -1.925411662    -2.508188404
......
...... (further entries)
......
VAL      1      3      3
CB       0.536149580    -0.807939518    -1.247819175
CG1      2.071963320    -0.775006353    -1.482573129
CG2      0.157614719    -2.308695819    -1.233398918
VAL      2      3
CB       0.536149586    -0.807939458    -1.247819175
CG1      0.183540298    -2.320728305    -1.293144468
CG2      0.071113770    -0.239518791    -2.610318888
VAL      3      3
CB       0.536149580    -0.807939518    -1.247819175
CG1      0.307054055    -0.157764981    -2.640552508
CG2      2.050569511    -1.122191502    -1.188293556
GLY      1      1      0
END
```

rotamer_choice.brch

This file defines which of the rotamers specified in the rotamer library (.brot) file are actually used during the BUDE docking. (NB rotamers are only used in the sidechain mean-field calculations in this version of the software)

```
%BUDE ROTAMER CHOICE
%VERSION 2.00
# Remember that the 2 previous lines are required,
# This file is dependant on the rotamer library.
# This file allows you to specify which rotamers of a particular amino acid
# you want to use for the energy calculations.
#
# %VERSION 2.00: The first line of data must have two intergers. The number of
# amino acids and the number of rotamers.The rotamers must not exceed the
# MAX_ROTAMER_CHOICE_NUMBER, currently set to 200.
# Amin oacids must not have 1 in any rotamer higher than those declare in the
# rotamer library.
#
# 1 means use the rotamer, 0 means do not use the rotamer
#
# %VERSION 2.00: Will allow comments at any position within the file. Any part
# of a line after a '#' character will considered a comment.It will also
# ignore blank lines.
#
# %VERSION 2.00: The tag END will denote the end of the library. No further
# lines will be considered. The line containing the END tag can only have the
# tag and blank characters or a comment.
#
# If the END tag is omitted the file will be process till the EOF and a warning
# will be issued.
#
```

```
#    1  2  3  4  5  6  7  8  9  10 11 12 13 14 15 16 17 18 19 20
20 20
ALA  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
ARG  1  1  0  1  1  1  1  0  0  0  0  0  0  0  0  0  0  0  0  0
ASN  1  1  1  1  1  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0
ASP  1  1  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
CYS  1  1  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
GLN  1  1  1  1  1  1  1  1  1  1  1  1  1  0  0  0  0  0  0  0
GLU  1  1  1  1  1  1  1  0  0  0  0  0  0  0  0  0  0  0  0  0
HIS  1  1  1  1  1  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0
ILE  1  1  1  1  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
LEU  1  1  1  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
LYS  1  1  1  1  1  1  1  0  0  0  0  0  0  0  0  0  0  0  0  0
MET  1  1  1  1  1  1  1  0  0  0  0  0  0  0  0  0  0  0  0  0
PHE  1  1  1  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
PRO  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
SER  1  1  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
THR  1  1  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
TRP  1  1  1  1  1  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0
TYR  1  1  1  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
VAL  1  1  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
GLY  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
END
```

BUDE options Reference

Progam options are typically provided in the control file (**.bctl**), which is merely a text file containing a list of options and arguments and is passed to BUDE with the option **-f** . Where appropriate, options have default values. Options may also be passed to BUDE on the command line. Options given on the command line override any options that might have been passed in a control file.

Control file rules:

Every line that begins with '#' is treated as a comment. In-line comments are allowed such that all text in a line after a '#' character is ignored. Any empty line or on containing only white spaces will be ignored. The option and its value must be on the same line separated by a white space. If options are duplicated in the control file, the last definition found is used. The END tag denotes the end of data. If the END tag is omitted the file will be processed until EOF.

Options:

Many options have a short and long form, both are given here

-h, **--help**
: Display this help and exit.

-v, **--default-values**
: Print default values and exit.

-B, **--control-file-template**
: Creates a template for the control file and exits.

--cl-info
: Display the OpenCL Platforms, Devices and exits.

-**f**, **--control-file** file.name
Reads file to initialise the control variables.

-d, **--debug-flag** true|false
Set whether to print debugging, default is false.

-p, **--print-flag**: true|false
Set whether to print details of the run to the standard output, default is false.

-i, **--dummy-run** true|false
Set whether to do a dummy run, default is false. A dummy run does the input pre-processing but stops just before the actual docking.

-b, **--carbon-alpha-beta-only** true|false
Set whether to do “Carbon Alpha and Beta only” docking, default is false. I.e only Cα and Cβ atoms are used in the docking, all other atoms are ignored. This option is only appropriate for docking two polypeptides, and requires an appropriate “coarse-grained” forcefield file.

-n, **--multi-conformer-dock** true|false
Treat conformers as single molecules, default is false. This option applies only to Bude-Mol2 format files of ligands where there may be multiple conformers defined (via the @BUDE<CONF> records). If this option is set to false, all atoms will be used to give a molecular mean-field and a single docking done for this Bude_Mol2 file. If the option is set to true then each conformer will be treated as a single molecule and a docking performed for each conformation (the latter is typically used for virtual screening).

-j, **--use-hydrogen** true|false
Set whether to use hydrogen atoms, default is false. Option true is **never** used for the BUDE forcefield but is available for future compatibility development for other forcefields or scoring methods. N.B. using hydrogens and setting their interaction parameters to 0 in the forcefield file will **not** ignore them due to the parameter summing/averaging used in the BUDE pose energy calculations.

-l, **--use-mass-centre** true|false
Set whether to use centre of mass as reference, default is true. This option ensures that rotations are applied around the centre of mass of the ligand. Typically this is a sensible thing to do in order to avoid spurious translations resulting from a non centred molecule. (N.B. it is actually the centre of coordinates rather than the centre of mass that is calculated and used)

-r, **--read-rotamer-mode** unsigned short
Rotamer Read Options. This option is not currently used.

-e, **--print-energy-mode** unsigned short
Print Energy Mode. This option is not currently used.

-o, **--emc-route** unsigned short

EMC route, values are 1–3, Default is 1. This option determines how the minimisation is performed.

1 EMC is minimised by Energy.

2 EMC is minimised by RMSD.

3 Not implemented currently, it will default to Energy.

-y, **--rmsd-type** unsigned short

RMS type, values are 0 and 1, Default is 1. This option determines how the RMSD is calculated.

0 RMSD will user Carbon Alpha only.

1 RMSD will use all atoms.

-s, **--sort-mode** unsigned short

Sorting mode. Determines which type of sorting is performed. Not currently used.

-a, **--residue-activation-mode** unsigned short

Activate residues mode, values 00 – 22, Default is 00. This is a two digit, base 3, value. The first digit is used for the receptor and the second for the ligand.

0 - means do not activate.

1 - means activate by sequence file.

2 - means activate by surface file.

e.g. 12 - means activate receptor by sequence and the ligand by surface.

-c, **--energy-calculation-route** unsigned short

The route used for energy calculations. This option may take values 0 - 2 and the default is 0. Essentially, this command controls the whether the program uses the OpenCL kernels for the energy calculation (options 0 and 1). There is also an extra option 2 for reporting residue-residue interaction energies in the non-accelerated code path. The behaviour of these options depends on whether BUDE was compiled with OpenCL enabled or not, as follows:

OpenCL disabled:

0 the energy calculation will be performed using all available CPU cores

1 the program will terminate because it is not OpenCL aware

2 the energy calculation will be performed using one CPU core and all residue-residue interaction energies reported. This option can generate a lot of data so use with caution!

OpenCL enabled:

0 the energy calculation will be performed using one CPU core and the host code

1 the energy calculation will be performed using the OpenCL kernels running on all available OpenCL devices (e.g. CPUs and GPUs) as defined by the option **--cl-dev**

2 the energy calculation will be performed using one CPU core and all residue-residue interaction energies reported. This option can generate a lot of data so use with caution!

-g, **--generation-zero-seed** unsigned

Generation zero seed. This should be a large positive integer and it is used as the random number generator seed for producing the random, zero-generation set of transformation descriptors for the start of the EMC minimisation.

-u, **--cutoff-distance** double

Cut off distance in Ångstroms. Must be a real number greater than 0, the Default is 22.0 . It is the cutoff distance between residues, measured between Cα atoms (or atom 2 in the residue if Cα not present). Entire residues are excluded from calculation based on these criteria.

-m, **--fractional-multiplier** double

Fractional Multiplier. Designed for mixed RMS and Energy target functions Not implemented in this BUDE version.

-x, **--exhaustive-generation**: true|false

Set whether to do an exhaustive descriptor generation, default is false. NB this will generate Nxtrans*Nytrans*Nztrans*Nxrot*Nyrot*Nzrot transformation descriptors, so should only be used on *small* grids. Otherwise the total size of generation zero will be unreasonably large.

-z, **--generation-zero-descriptor-number**: unsigned

This is the number of randomly generated transformation descriptors for generation zero of the EMC genetic algorithm. If this is set to 0 they are read from the file passed with --zero-generation-filename. If this is not 0, then this filename must be empty (i.e. no argument passed to the option --zero-generation-filename). Likewise, the option --exhaustive-generation must be set to false.

-Z, **--zero-generation-filename**: file.name

Zero generation file name. This is the file containing a set of zero generation transformation descriptors to be read by BUDE. If this file name is given, the option --generation-zero-descriptor-number must be set to zero and the option --exhaustive-generation must be set to false.

--unique-tds-gen0: true|false

Set whether generation zero will be formed with unique transformation descriptors or not, default is false. This is only used in special circumstances, since duplication by random generation is unlikely to be common in the typical case where zero generation size << total possible states (possible poses). Likewise, unique generation becomes computationally costly if the number of zero generation descriptors is a significant proportion of the total number of states (possible poses). Use with caution!

-R, **--output-file-base**: file

This is the base name for the coordinate file output. It can include full or relative paths. Note, the directories specified in the path must exist, they will not be created (this behaviour is a safety precaution). The filename base is used as a prefix to the standard output file naming (see section XXX).

-E, **--output-energy-filename**: file.name

This is the base name for the energy file output. It can include full or relative paths. Note, the directories specified in the path must exist, they will not be created (this behaviour is a safety precaution). The filename base is used as a prefix to the standard output file naming (see section XXX). If this option is not defined, the --output-file-base is used.

--output-coordinates-format: coordinate output format

Specifies the format of the output coordinate files. Valid options are PDB and MOL2 (these are not case sensitive)

-P, **--receptor-coordinates-filename**: file.name

Valid formats are PDB and MOL2. NB PDB files must have 80 character records, use the utility bude_padPDB to fix if necessary.

--docking-grid-centres: X Y Z Coordinates

The centre of the docking grid will be moved to these coordinates in the receptor coordinate system. If nothing is specified here and a standard grid (i.e. centred on 0,0,0) is used then the centre of the receptor binding pocket must also be at 0,0,0.

-O, **--receptor-surface-filename**: file.name

This holds the name of the receptor Connolly surface file used for surface docking (binding site detection and protein-protein docking).

-S, **--receptor-sequence-filename**: file.name

The receptor sequence file name is used for selecting sidechains of residues for use in flexible sidechain docking. The only rotamer flexibility currently implemented is the rotamer-mean-field energy calculation

-I, **--ligand-coordinates-filename**: file.name

Valid formats are PDB, MOL2, Bude-Mol2 and Bude-list. NB PDB files must have 80 character records, use the utility bude_padPDB to fix if necessary. Bude-mol2 allows the easy inputting of multiple ligand conformations. The Bude-list format file allows a list of ligand files to be passed to the program and is typically used in virtual screening. The list file may be either a list of Mol2/Bude-mol2 files or PDB files but *not* a mixture of both.

-Q, **--ligand-sequence-filename**: file.name

The ligand sequence file name is used for selecting sidechains of residues for use in flexible sidechain docking. The only rotamer flexibility currently implemented is the rotamer-mean-field energy calculation

-N, **--ligand-surface-filename**: file.name

This holds the name of the ligand Connolly surface file used for surface docking when performing protein-protein docking.

-G, **--target-coordinates-filename**: file.name

This holds the name of the ligand target used for docking by RMSD to a target rather than by energy. It must hold the same molecule as the ligand file, but in the target

position with respect to the receptor (e.g. in the position determined experimentally by crystallography). Only used for testing the EMC by doing an RMSD minimisation.

-T, **--transformations-filename**: file.name
This file contains the grid positions and available rotations that are used as the 6-D search space during docking.

-F, **--forcefield-filename**: file.name
This contains the forcefield file name. NB a WLD residue is used to parameterise all atoms by atom type. The parameterisation by residue type requires the coordinates to be in PDB format and every residue type to have corresponding parameters in the forcefield file.

-V, **--emc-filename**: file.name
The evolutionary Monte Carlo file contains the parameters that control the EMC pose search and the output of energy and coordinate data.

-L, **--rotamer-library-filename**: file.name
This is the file defining the geometry of all sidechain rotamers available to residue types. Mean-field energies only implemented currently.

-C, **--rotamer-choice-filename**: file.name
This file selects which of the rotamers from the library are actually used during this docking run. This holds the name of the ligand

--mol2-translations-filename: file.name
If the user wishes to use a PDB file input but parameterise the atoms using teh “by atom type” force field, this file can be used to translate the PDB residue and atom type into the corresponding Mol2 type. More typically the user would convert the PDB file to Mol2 format using babel, and then use the Mol2 input directly.

--cl-kernel-file: file.name
This is the file name of the BUDE OCL kernel file. Default is ‘’. This OpenCL source code is loaded and compiled “on the fly” by the man program. Hence, it is absolutely required for OpenCL accelerated jobs.

--cl-dev: String
This option allows the user to choose which OpenCL devices to use. Valid options are, 'all', 'gpu', 'cpu', 'default', 'accelerator'.

--cl-dlb: true|false
This turns dynamic workload balancing on and off.

--cl-dbl-wload: unsigned short
When dynamic load balancing is turned on, this value is the threshold percentage to trigger rebalancing. The default value is 5.

--cl-select: short:short[,short:short]

Platform:Device pairs to select for use during this BUDE run, default value is '' (i.e. blank, meaning no devices). It is a comma separated list of colon separated pairs. E.g. 1:1,1:2,2:1 instructs BUDE to use platform 1 devices 1 and 2, and platform 2 device 1. If you are running on a workstation, and you still want to use the graphics during the BUDE run, do not select the GPU device driving the screen!

--cl-loads: double[,double]

This comma separated list distributes the initial workload for the devices chosen with the option --cl-select, and specifies the percentage work performed on each device. The sum total of this list should be 100.

--cl-wgsize: int[,int]

This comma separated list of integers defines the OpenCL workgroup size for each device defined with --cl-select. This will affect the OpenCL performance MORE HERE XXXX

Utilities

Data handling, pre and post processing utilities are provided. These are all standalone programs that can be run with the **-h** option for further information. Their use is summarised here.

bude_bild

This utility will read a BUDE Connolly surface file and produce three Chimera bild format files for visualising the surface points and their normal vectors. One bild file is generated for the points, one for the vectors and one for a combination of both.
Receptor files will be coloured red, ligands blue.

programName [option] [value]

Options:

-h	Prints this help and exits.
--help	Prints this help and exits.
--receptor-surface-filename	Connolly Surface Points File name for the receptor.
--ligand-surface-filename	Connolly Surface Points File name for the ligand.

bude_formatter

This utility provides some re-formatting options for small-molecule databases and files. Its principal function is to pack and unpack multiple conformations of a small molecule in mol2 format into a single extended mol2 format file for reading into BUDE. This can reduce the size of mol2 files of multiple conformations by up to four-fold while still providing human readable ASCII data files. The extension to mol2 format is the addition of a record tag '@<BUDE>CONF N' where N is the conformer number. This tag must be placed after the first complete molecule description in the mol2 file and comprise exactly one line per atom of updated x, y and z coordinates. This record tag and data are repeated for each new conformation. Alternatively, it can produce several files, one per molecule from Multiple Molecule Mol2 or SDF files.

The input file can be a Multiple Molecule file or a list of file names.
The format can be Mol2, SDF, BUDE_Mol2. It will be Mol2 by default.
The multiple file option will false by default, thus producing BUDE-Mol2.

It is assumed that the name of the molecule follows this pattern in the file name:
PREFIX_molecule_name_SUFFIX
The pattern 'molecule_name' will be added to the molecule name in the Mol2 file. If the PREFIX & SUFFIX are empty the whole file name, excluding the extension, will be added into the molecule name field.

eg. file name: XX_006_arg_ZZ.mol2
The molecule name will be:
'006_arg', if PREFIX=XX_ & SUFFIX=_ZZ.mol2
'XX_006_arg_ZZ', if PREFIX & SUFFIX were empty or do not occur in the file name.

When fragmenting multiple molecule SDF files. The option '--sdf-id-tag' will be use to extract the molecule id and use it as part of the file name for single files.

If the format is BUDE_Mol2 the utility will create a single multi-molecule-Mol2 file or multiple Mol2 files one per molecule, according to the option '--multi-files'.

Usage: programName [option] [value]

eg. bude_formatter --in-file-name prefix-molecule_name-suffix.mol2 --in-file-format BUDE_Mol2

Options:

-h
: Prints this help and exit.

--help
: Prints this help and exit.

--in-file-name
: Input file name.

--in-bude-list
: List of input file names, one per line.
All files in the list should be the same format.

--out-file-basename
: Output file basename.

--in-file-format Default: 'Mol2':
: File format for the input file. [Mol2 | SDF | BUDE_Mol2]

--multi-files Default: 'false':
: Whether we want multiple files, one per molecule. [true | false]

--infile-molecule-name Default: 'false':
: Whether we want get molecule name from file name.[true | false]

--override-residue-label Default: 'true':
: Whether we want override residue name and ID number.[true | false]

--prefix
Prefix in the file name.

--suffix
Suffix in the file name.

--sdf-id-tag Default: '<mol.id>':
SDF Tag for molecule id.

The option '--in-file-name' has higher precedence than '--in-bude-list'.
If both are given the '--in-bude-list' option will be ignored.

b_gensrf.sh

This script reads a standard PDB format file and generates a BUDE format surface file using msnew. If the input file is 1ABC.pdb the output will be 1ABC.pdb.bsrf and 1ABC.pdb.bsrf.log. Details of surface area, point density and probe size are written to the log file.

Usage: b_gensrf.sh [pdb_files_list] [pdb_dir]

Where:
pdb_files_list is a file containing a list of PDB files names, one per line.
pdb_dir is the directory containing the PDB files.
Surface files will be written to directory: bude_surface_files/.

Options:

-h	Prints usage and exits.
--help	Prints usage and exits.

bude_genZero

This utility will read a BUDE transformation library file (.bltr) and produce a BUDE transformation descriptor file (.tds) suitable for use as the zero generation population for the EMC minimisation. The .tds file created will ***only*** be compatible with the corresponding transformation library. The “exhaustive” and “number of descriptors to generate” flags cannot be used simultaneously.

programName [option] [value]

Options:
-h, **--help**
Prints this help and exits.

--unique-tds-gen0 [true/false]
Generate unique descriptors.

-x, **--exhaustive-generation** [true/false]

Whether to produce an exhaustive generation or not.

-z, **--generation-zero-descriptor-number** [positive integer]
Number of descriptors to generate.

-g, **--generation-zero-seed** [positive integer]
Seed to use for the random generator.

-T, **--transformations-filename** [filename.ext]
BUDE Transformation Library file name.

-R, **--output-file-base** [filename]
Basename for the output file.

bude_padPDB

This program will read PDB files and pad every line that has less than 80 characters. The default padding character will be ' ' (i.e. a blank character).

programName [option] [value]

Options:

-h	Prints this help and exits.
--help	Prints this help and exits.
--pdb-file	PDB File name.
--multi-pdb-file	PDB File names.
--padding-char	Padding character, default ' '.
--padding-length	Padding Length, default 80.

e.g.
bude_padPDB --pdb-file fileName1 --pdb-file fileName2
bude_padPDB --pdb-file fileName1 --pdb-file fileName2 --padding-char ' ' --padding-length M

Where:
fileNameX is the name of the PDB files to pad.
' ' is the character to pad with.
M is the length at which the lines will be padded to.

e.g.
bude_padPDB --multi-pdb-file fileName1 fileName2
bude_padPDB --padding-char ' ' --padding-length M --multi-pdb-file fileName1 fileName2

Where:
fileNameX is the name of the PDB files to pad.
' ' is the character to pad with.
M is the length at which the lines will be padded to.

Please note that if --multi-pdb-file option is given.
It MUST be the LAST option, only file names must be after it.

bude_point_select
This program will select N number of Connolly surface points from a BUDE Connolly Surface Points File (.bsrf) and create a new, smaller surface file.

programName [option] [value]

Options:

-h	Prints this help and exits.
--help	Prints this help and exits.
--bude-surface-file	Connolly Surface Points File name.
--receptor-surface-filename	Connolly Surface Points File name for the receptor.
--ligand-surface-filename	Connolly Surface Points File name for the ligand.
--points-number	Total number of Surface Points to select for one or two files.
--min-per-molecule	Minimum number of point to be selected per molecule, in weighted selection.
--random-selection	Select them randomly or not.

e.g.
bude_point_select --bude-surface-file fileName --points-number N --random-selection true/false

Where:
fileName is the name of the BUDE Connolly Surface File.
N is the number of points you want to select.
true/false whether you want to select them randomly or not, default is true.

e.g.
bude_point_select --receptor-surface-filename receptorFile.name --ligand-surface-filename ligandFile.name --points-number N --min-per-molecule M --random-selection true/false

Where:
receptorFile.name is the name of the BUDE Connolly Surface File for the receptor.
ligandFile.name is the name of the BUDE Connolly Surface File for the ligand.
N is the number of points you want to select.
M is the minimum number of points you want to select per molecule.
true/false whether you want to select them randomly or not, default is true.

Selection:
If set to randomly select the points. The program will divide the total number of points into N number of fragments as and then it will randomly select a point from each fragment. If the random flag is not set then the program will select the first point.Then it will jump total/N number of points before selecting the next point.

Weighted Option:
If option --receptor-surface-filename is used, then --ligand-surface-filename must be used as well and vice versa. These options are used for the weighted selections of points and the number of points must be the total of points to select from the two files.

If the minimum number of points per molecule is specified, the weighted selection will be modified to fit this requirement. E.g. If a receptor-ligand pair has a ratio of 70:30 surface points respectively, the total number of points to select is 100 and minimum per molecule is 40. The selection will be 60 for the receptor and 40 for the ligand instead of 70 for the receptor and 30 for the ligand.

bude_sequence

This program will read PDB or MOL2 files and produce a BUDE Sequence list file per file. PDB and Mol2 files must comply with BUDE requirements. E.g. Each line of the PDB file must be at least 80 characters long. For PDB format, any residue which is comprised of HETATMs only will NOT be included in the BUDE Sequence List.

programName [option] [value]

Options:

-h	Prints this help and exits.
--help	Prints this help and exits.
--molecule-file	PDB File name or Mol2 File name.
--multi-file	PDB and/or Mol2 File names.
--residues-active	Whether residues will be set to active or not. False by default.

e.g.
bude_sequence --molecule-file fileName1 --molecule-file fileName2
bude_sequence --molecule-file fileName1 --molecule-file fileName2 --residues-active true

Where:
fileNameX is the name of the PDB and/or Mol2 files to create sequence from.
true means that all residues will be set to active.

e.g.
bude_sequence --multi-file fileName1 fileName2
bude_sequence --residues-active false --multi-file fileName1 fileName2

Where:
fileNameX is the name of the PDB and/or Mol2 files to create sequence from, and false means that all residues are set to inactive.

Please note that if [--multi-file] option is given.
It MUST be the LAST option, only file names must be after it.