README

ALAscanApp.py

This is a python 3 utility to do alanine scanning using ISAMBARD and BUDE. It is a self-contained utility. It does not need installation. However, it needs to have Python 3.5 or greater. It also needs a reasonably recent C++ compiler. We recommend GCC.

This utility is meant to work in Linux and Mac systems. This document explains how to get it working in these systems.

For the App to work it is required to compile BUDE, the C++ utilities and update/create the App initialisation file (.alascanapp.ini). Hopefully all of that was/will be done with the setup_utils.py script.

If using a different compiler from GCC or Mac system, exporting the CC and CXX environment variables should be done before attempting any compilation procedure or running the setup_utils.py. To use setup_utils.py python 3 must be available.

Definitions for Running Commands

PATH: Environment variable containing information on where to look for executable.
HOME: Environment variable containing the information of the user's home directory.
username: User's name in the system. e.g. ai7766
'>' means prompt from a terminal as ordinary user.
'#' means prompt from a terminal as root or admin user.
'>>>' means prompt for python interpreter.

HOME usually is expanded to '**/home/username**' in Mac systems '**/Users/username**'.

e.g.
>*ls dir*
List content of dir as ordinary user

#zypper install package
Install package as root user.

>>>*quit()*
Quit python interpreter.

Italics show a command and its output.
>*command*
command output

Requirements:

- GCC 4.6 or modern C++ compiler.
- Python 3.5 or greater (Anaconda).
- ISAMBARD
- BUDE

Linux

Most Linux systems have GCC available. Checking if you have GCC, using the command 'which'.

```
>which gcc
/usr/bin/gcc
```
(Or similar, depending on where gcc is installed).

```
>which g++
/usr/bin/g++
```

If it is not available it should display something along these lines:

```
which: no g++ in PATH.
```

If it is not installed in your computer you will need admin rights to install it. Depending on your Linux distribution.

SuSE family
```
# zypper install gcc
```

Red Hat family
```
# yum install gcc
```

Debian family
```
> sudo apt-get install gcc
```

Once GCC is installed you can check with the 'which' command again and it should give you the full path where g++, gcc was installed.

If you have a different compiler, type at the terminal.
```
> export CC=my_C_Compiler CXX=my_C++_Compiler
```

Getting python anaconda

Go to website: https://www.continuum.io/downloads Download the appropriate installer for python 3.X, most likely it will be (64-BIT (X86) INSTALLER), it should download the file Anaconda3-X.X.X-Linux-x86_64.sh where (X.X.X) is the version.

Open a terminal and change directory to wherever the installer was downloaded, run the command and follow the instructions.

```
> bash Anaconda3-X.X.X-Linux-x86_64.sh
```

accept the license, press enter and the letter '**q**', type '**yes**' after the question:

```
Do you approve the license terms? [yes|no]
```

After this you will be asked the location for installing Anaconda3. It suggests

```
/home/username/anaconda3
```

You must have writing access to this location. The default value usually is a good place. If you wish a different location type it. It will output:

```
PREFIX=/home/username/anaconda3
```

and the installation begins.

All going well you should see this message

```
Python 3.6.1 :: Continuum Analytics, Inc.
creating default environment...
installation finished.
Do you wish the installer to prepend the Anaconda3 install
location
to PATH in your /home/username/.bashrc? [yes|no]
[no] >>>
```

Answer '**yes**' to that final question. This will allow you to use that python version whenever you open a terminal. All done with anaconda.

Close that terminal and open a new one. This will ensure that you are using the right version of python. To check it use the 'which' command.

```
> which python
/home/username/anaconda3/bin/python
```
(or wherever you installed it)

Checking python version:

> python --version
Python 3.6.X :: Anaconda X.X.X (64-bit) (according to the downloaded version)

Please check that the version is **greater than 3.5**.

Installing ISAMBARD

>pip install isambard

This should install all required packages. Have a look at this page for more information. https://github.com/woolfson-group/isambard/blob/master/README.md

We will need **Scwrl4** for creating the mutants for the hot constellations. If Scwrl4 is not present only the single alanine mutants will be calculated without any sidechain repacking.

Configuring ISAMBARD

We need to run the python interpreter and import isambard

> python

It will show the message below and will open a python prompt

'>>>' Python 3.6.X :: Anaconda X.X.X (64-bit)
... >>>

Once you have the python prompt type:

>>> *import isambard*

This will start isambard configuration. It will ask a few questions. For hot constellations we are interested in the Scwrl4 path. You should be able to see the following information. Bold text is meant to be input by the user. We are particularly interested in the SCWRL path, Rigid packing mode and Bude_2016v1 force field. If some of the executables are not present, pressing 'Enter' should suffice.

```
Running configure.py…
Generating configuration files for ISAMBARD.
All required input can use tab completion for paths.
Setting up SCWRL 4.0 (Recommended)
Please provide a path to your SCWRL executable Tab completion
enabled:
/path/to/scwrl4/Scwrl4
Path set to "/path/to/scwrl4/Scwrl4"
Please choose your packing mode (flexible is significantly slower
but is more accurate). Use a number to select an option:
[1] Flexible
[2] Rigid
2
Setting up DSSP (Recommended)
Please provide a path to your DSSP executable. Tab completion
enabled:
Setting up BUFF (Required)
Please choose the default BUFF force field, this can be modified
during runtime. Use a number to select an option:
[1] Standard
[2] Bude_2016v1
2
Setting up Reduce (optional)
Please provide a path to your reduce executable. Tab completion
enabled:
Setting up naccess (optional)
Please provide a path to your naccess executable. Tab completion
enabled:
Setting up ProFit (optional)
Please provide a path to your ProFit executable. Tab completion
enabled:
Writing settings file to '/HOME/.isambard_settings'
Configuration completed successfully.
...Some C++ messages, safe to ignore ...
>>>
```

After that is done, isambard should be ready to use. We can exit the python interpreter.

```
>>>quit()
```

C++ Utilities

I am assuming that you have the alaScanApp-dist.tar.gz file and it was unpacked and uncompressed creating a directory **alaScanApp-dist** in your home directory with the following structure.

/HOME/**alaScanApp-dist**
/HOME/**alaScanApp-dist**/alaScanApp.tar.gz
/HOME/**alaScanApp-dist**/setup_utils.py
/HOME/**alaScanApp-dist**/**ccpCode**
/HOME/a**laScanApp-dist**/**ccpCode**/**getSD**
/HOME/**alaScanApp-dist**/**ccpCode**/**colourBySD**
/HOME/**alaScanApp-dis**t/**ccpCode**/**dividePDB**
/HOME/**alaScanApp-dist**/**ccpCode**/BUDE-1.2.9.tar.gz
/HOME/**alaScanApp-dist**/**ccpCode**/comp_utils.sh
/HOME/**alaScanApp-dist**/**doc**

We change into that directory from HOME.

>*cd alaScanApp-dist*

We should run the setup_utils.py script. It will display information and waits for user input at three stages. Typing '**yes**' or pressing enter will be understood as 'Yes' and the script will continue. Typing '**no**' will avoid that step. Typing anything else will enter a loop until any of those three options is typed.

Running the script:

>*./setup_utils.py*

```
...
Loads of information,
Do you want to continue [yes|no]? [Yes]
Loads of information,
...
```

Please be patient, it might take a few minutes. Hopefully, all C++ programs were compiled with setup_utils.py. If that was not the case this is how to do them.

BUDE

We need to unpack and uncompress bude (BUDE-1.2.9.tar.gz). There is no need to install it but we will need to compile it. If we have installed a modern version of GCC it should be painless. The compress file should be within the **cppCode** directory.

Change to the C++ directory.

```
>cd cppCode
```

Extract and decompress BUDE

```
>tar -xzvf BUDE-1.2.9.tar.gz
```

This should create a directory called **BUDE-1.2.9.** Let's change into that directory, make a build directory and change into it by executing the commands below.

```
>cd BUDE-1.2.9
>mkdir build
>cd build
```

Once we are in the build directory, we compile BUDE.

```
>../configure --prefix=$HOME
```

A few messages will appear:

```
checking ...
...
```

After the ../configure script is finished we type the command

```
>make
```

Please be patient, the make command produces a healthy amount of output and it will take a few minutes depending on the speed of your machine. All going well there will be a directory called '**bin**' with a few **bude_xxx** utilities and one called **budeScan**.

To check that all went well type

```
>bin/budeScan -h
```

And the BUDE help should be displayed. If we want to install BUDE in the home directory type

>make install

This will install all bude applications in /**HOME**/**bin** and libraries in /**HOME**/**lib(64)**. If you do not want to install it, take note of the path where the executable is. It will be needed for the ALAscanApp configuration.

Then we should compile the C++ utilities. They are in the **cppCode** directory. If we are doing it after compiling BUDE. Change to the **cppCode** directory.

>cd ../../

Once you are in the **cppCode** directory we use the comp_utils.sh script to compile all C++ utilities. Type:

>./comp_utils.sh

```
Compiling getSD
...
Compiling colourBySD
...
Compiling dividePDB
```

That should be all about compilation.

initialisation file

The initialisation file '**.alascanapp.ini**' will be created or updated in the user's home directory when running ALAscanApp.py or setup_utils.py. The file will be created by either script, if it does not exist. ALAscanApp.py will update the file only if any section or option is missing. setup_utils.py will always update the relevant sections of the compiled utilities.

The file is composed by sections, defined by square brackets **[Section]**. Each section may have several options and their values, '**AnOption = a_value'**.

This file could be edited to alter the ALAscanApp behaviour. Only the values can be modified. Options the values of which contain curly brackets preceded by the sign '**$**' should not be modified. e.g. '**AnOptionX = ${doNoModify}**'.

The sections have self-explanatory names. There are some options that must be updated if the setup_utils.py script was not run.

In section ‘**[Directories]**’, option ‘**CppCode**’ its value should be updated to its proper location. If executables budeScan, getSD and colourBySD are not within that directory then options ‘**BUDE**’, ‘**GetSD**’ and ‘**ColourBySD**’ must also be updated accordingly.

Alternatively, in section ‘**[Executables]**’ the options ‘**BUDE**’, ‘**GetSD**’ and ‘**ColourBySD**’ could be updated with the full path to these executables.

Behaviour Options

Any option’s value within the section **[File Names]** can be modified. They will change the naming of those particular files.

In section **[Directories]** the ‘**Work**’ option should be changed to the directory where the Hot Constellations will be created and ALAscanApp.py will be run from, by default it will be ‘/HOME/**ALAscanApp**’. Directories from the options ‘**Scan**’, ‘**BudeLibs**’, ‘**BudeResults**’, ‘**PdbSources**’ and ‘**Chimera**’ will be created within the work directory. Their values could be modified.

To specify the maximum number of hot constellations that will be calculated in the auto mode, the option ‘**MaxAutoNumber’** in section **[General]** can be modified. Very large numbers must be avoided.

Mac

The procedure is practically the same. The only difference is that we will need to install a proper GCC, the one that comes with Xcode is not a recent one.

To install GCC we will need Xcode and, Homebrew. If they are not installed already we will need to install them first. **For installing these programs we will need admin rights**.

Open terminal. In finder's menu go ⇒ Utilities find the Terminal icon.

Xcode:

>*xcode-select --install*

Follow the instructions on the screen. Please be aware that there are several Web tutorials on how to install Xcode in the different flavours of Mac. A simple search ‘install Xcode Mac’ will return very comprehensive tutorials.

Homebrew:

Please follow the installation instructions in the Homebrew's Website. https://github.com/Homebrew/brew/blob/master/docs/Installation.md

GCC:

Once we have Xcode and Homebrew we can install GCC.

```
>brew install gcc
...
```

This will install GCC, there should be **gcc-X** and **g++-X** where **X** is the version of GCC. We need to export the CC and CXX environment variables.

```
>export CC=gcc-X CXX=g++-X
```

Exporting these variables must be done whenever we open a Terminal. To avoid this it is recommended to add it into **.bash_profile** file, which is in your HOME directory.

Then we follow the same procedure as described from the Linux section for getting anaconda. Bear in mind that you will need the Mac installer and you have two flavours, the command line and a graphical one. The command line behaviour is pretty much like in Linux. The graphical one has a very self-explanatory frontend.