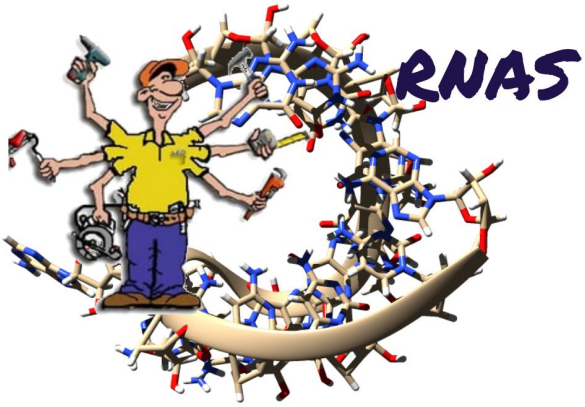
QRNAS

Quick Refinement of Nucleic Acid Structures

User manual version 0.3

(2019)



Content

1. Functionality

QRNAS is a software tool for fine-grained refinement of nucleic acid structures, dedicated to improving the quality of models generated by low- to medium-resolution methods commonly used, e.g., for RNA three-dimensional (3D) structure modeling. QRNAS is capable of handling RNA, DNA or chimeras and hybrids thereof, and enables modeling of nucleic acids containing modified residues. The force-field used by QRNAS is a modified version of AMBER adopted to represent 107 modified nucleotides currently known to be present in RNA.

2. Installation

QRNAS supports both Linux and Mac operating systems. It can also be installed and used under MS Windows, only if MS Windows Subsystem for Linux (WSL) is enabled.

2.1 Installation on Linux machines

2.1.1. Installation of C++ compiler and make

The QRNAS program is written in C++ language using the STL libraries. On GNU/Linux machines it can be compiled using g++ compiler. The g++ compiler can be installed using the following command on a Ubuntu, Debian or their derivatives:

```
$ sudo apt update
$ sudo apt upgrade
$ sudo apt install build-essential
```

On Fedora/Red Hat or its derivatives, the installation can be performed using the following command:

```
$ sudo dnf install make gcc-c++
```

For openSUSE or its derivatives use the following command:

```
$ zypper install make gcc-c++
```

For Arch Linux use the following command:

```
$ pacman -S base-devel
```

Once the compiler is installed QRNAS can be extracted and compiled as described in 2.1.2.

2.1.2 Extraction of QRNAS

The QRNAS source code is provided as a standard tar.gz archive file. The file can be downloaded from the url: http://genesilico.pl/QRNAS/QRNAS.tar.gz. After downloading, the file can be extracted using the following command:

```
$ tar -xvzf QRNAS.tar.gz
```

The above command will create a new directory named QRNAS and extract the source inside it.

2.1.3 Compiling QRNAs in sequential mode

Although the default behavior of make command is to compile QRNAS in sequential mode (in this mode, it can run using only one processor at a time), it is also possible to compile QRNAS in parallel mode (see section 2.1.4).

The QRNAS directory contains all source codes, data directory and other files necessary for compilation and running of QRNAS. After extracting the files as mentioned in 2.1.2 the QRNAS can be compiled by entering the directory and running make command. All the sources will be compiled automatically to generate the binary file QRNAS.

```
$ cd QRNAS
$ make
```

2.1.4 Compiling QRNAS in parallel mode

Calculations of electrostatic interactions in QRNAS are written with the POSIX thread (pthread) library support. This allows the program to the compiled with parallel computing support, which make use of multiple processors at the same time. In order to use this feature, the QRNAS program needs to be compiled in a parallel mode. This can be done by using the Makefile by running the following command:

```
$ make parallel
```

2.1.5 Setting up environment variables for QRNAS

The following environmental variables can be set up for running QRNAS from shell outside the installation directory. The following commands can be run to add the environment variables to user's bashrc from the QRNAS source directory:

```
$ echo export PATH='$PATH':$(pwd) >>~/.bashrc
```

```
$ echo export QRNAS_FF_DIR=$(pwd)/forcefield >>~/.bashrc
```

For changes to take place in the current shell the bashrc file needs to be sourced again. However, this step is not needed if you are opening a new shell environment.

```
$ source ~/.bashrc
```

This step is optional. The `QRNAS_FF_DIR` environment variable can be exported to set path to `forcefield` directory. In the absence of `QRNAS_FF_DIR` environment variable, QRNAS expect `forcefield` directory (or a symbolic link to the `forcefield` directory) in the current working directory.

2.2 Installation on Windows 10 machines

In order to use QRNAS on Microsoft Windows 10, the first step is to enable Windows Subsystem for Linux (WSL). This can done by running the following command from PowerShell or Windows Command

```
$ Enable-WindowsOptionalFeature -Online -FeatureName Microsoft-Windows-Subsystem-Linux
$ sudo apt update
$ sudo apt upgrade
```

After rebooting the system, open windows command prompt and type `bash`, Follow the on-screen instructions and wait for installation to complete.
The remaining steps for installation are same as described in section 2.1

2.3 Installation on Mac OS

The first step is to install Xcode from the Mac App store. After installing the xcode, open Mac terminal app type the following command to install the command line tools necessary for compiling QRNAS

```
$ xcode-select --install
```

A pop-up windows will open and continue the installation following the on-screen instructions. The remaining steps for compilation and setting environment variables are same as described in section 2.1

3. QRNAS usage

3.1. Running QRNAS without configuration file

Upon successful installation, QRNAS optimization can be performed from the command-line using the default parameters:

```
$ QRNA -i <input_pdb_file>
```

Example:

```
$ QRNA -i 1l2x.pdb
```

It minimizes `1l2x.pdb` and writes the minimized structure in pdb format in every 100 steps as `1l2x_out_`*`XXX`*`.pdb` where *`XXX`* is the number of step (e.g. `1l2x_out_100.pdb` for 100th step).

However user can define the name of the output file:

```
$ QRNA -i <input_pdb_file> -o <output_file>
```

Example:

```
$ QRNA -i 1l2x.pdb -o 1l2x_QRNAS.pdb
```

In this case, the minimized output will be written as `1l2x_QRNAS_`*`XXX`*`.pdb` where *`XXX`* is the number of step (e.g. `1l2x_QRNAS_100.pdb` for 100th step).

QRNAS, with default verbosity, prints the messages including the energy values in console and does not record them in any file. To store this information in a log file, the following command can be used:

```
$ QRNA -i <input_pdb_file> -o <output_file> 2>&1> filename.log
```

Example:

```
$ QRNA -i 1l2x.pdb -o 1l2x_QRNAS.pdb 2>&1> 1l2x_QRNAS.log
```

3.2. Running QRNAS with the configuration file

QRNAS optimization can be run using customized parameters (altering the default values). The alternate parameters should be provided in a configuration file. An example configuration file should be as following:

```
# Sample config file for QRNA 0.2
# Just uncomment the lines you need.

#INPUTPDB   ./1l2x.pdb
#OUTPUTPDB  ./1l2x_refined.pdb
#WRITEFREQ  1000    # Frequency of writing outputpdb (each xxx steps); by default 100
#TRAJECTORY 1       #{0,1}-Each outputPDB is written to a separate file; by default OFF

#VERBOSE     2      # {0,1,2,3} - verbosity; by default 1

#NSTEPS      20000   # Maximal number of steps; by default 100000
#CUTOFF      12.0   # Cutoff for van der Waals [Ang]; by default 12
#USEBORN     1      # {0,1} - Born electrostatics, i.e. implicit solvent (off/on); by
default OFF
#ELECTR      0      # {0,1} Electrostatics (off/on); by default ON
#VDW          0      # {0,1} Van der Waals interactions (off/on); by default ON

#HBONDS      0      # {0,1} Hydrogen bonds (off/on); by default ON
                    # They are necessary to auto-detect secondary structure.

#SSDETECT   0       # {0,1} - Detect base pairs automatically, by default ON
#SSCONSTR   0       # {0,1} - Impose constraints on base pairs (off/on); by default ON

#MAXSTEP     1e-8   # Max step size for golden section search; by default 1e-4
#MINSTEP     1e-10  # Min step size for golden section search; by default 1e-8
                    # (In most cases these parameters are not recommended
                    # to be altered)

#NUMTHREADS  2      # Number of threads (parallel builds only!); by default 4
                    # Takes no effect in sequential builds of QRNA.
                    # Only calculations of electrostatics are currently parallelized.

#BLOWGUARD  0       # Protection against explosion due to high stress; by default ON
                    # It slows the minimization!

#POSRESTRAINTS 1    # {0,1} - Positional restraints (off/on), read from occupancy and
beta-factors;
                    # by default OFF

# Secondary structure can be specified in two ways.
# The deprecated and retarded way is via Vienna notation. It applies to the first chain
only:
#SECSTRUCT   (((....)))
# The kosher and robust way is via pairwise restraints specified in a separate file:
#RESTRFILE file_with_restraints.txt
```

The aforementioned example can be used for the optimization by uncommenting (by removing the '#') the listed parameters in the file:

Example:

```
INPUTPDB   1l2x.pdb
OUTPUTPDB  1l2x_refined.pdb
WRITEFREQ  1000    # Frequency of writing outputPDB (each xxx steps);
by default 100
#TRAJECTORY 1        #{0,1}-Each outputPDB is written to a separate
file; by default OFF
#VERBOSE   2     # {0,1,2,3} - verbosity; by default 1
```

In the above example, the configuration file specifies `1l2x.pdb` as input file (equivalent to `-i` argument in command-line) and `1l2x_refined.pdb` as output file (equivalent to `-o` argument in command-line). The argument `WRITEFREQ 1000`, indicates QRNAS to write optimized structures after 1000 steps (instead of 100 which is default). QRNAS will ignore the last two lines (`TRAJECTORY` and `VERBOSE` since they are commented).

QRNAS can be run with configuration files using `-c` flag:

```
$ QRNA -c <configuration_file>
```

Example:

```
$ QRNA -c config.txt
```

However, if `-i` and `-o` values are provided, QRNAS ignore INPUTPDB and OUTPUTPDB mentioned in configuration file.

```
$ QRNA -i 1l2x.pdb -o 1l2x_QRNAS.pdb -c config.txt
```

4. Running QRNAS with restraints

4.1. Secondary structural restraints

Secondary structure can be specified in two ways:

4.1.1. Vienna (dot-bracket) format

One of the ways to assign secondary-structure is via Vienna notation. It applies to the first chain only and can be mentioned in the configuration file.

Example:

```
SECSTRUCT    (((....)))
```

4.1.2. Explicit restraints on base-pairs

Watson-Crick (canonical) base-pairs can be restrained by specifying appropriate chain identifiers and residue numbers:

```
BASEPAIR    B/1    A/32
BASEPAIR    B/2    A/31
BASEPAIR    B/3    A/30
```

Any non-canonical base-pair will be ignored by QRNAS.

4.2. Long range restraints

QRNAS allows users to restraints a pair of atoms by a certain distance as following:

```
DISTANCE    A/3/C4'    B/4/C3'     3     4     50
DISTANCE    A/4/C2'    B/5/O3'     4     4     100
DISTANCE    A/5/C1'    B/6/H2'     5.5   3.5   100
```

Distance restraints are implemented as simple harmonic forces. The first column indicates the keyword “Distance”, second and third columns indicate the pair of atoms to be restrained. Fourth and fifth columns specify the minimum and maximum values of the distance, respectively. The spring length is considered as the average of the minimum and maximum (in Angstroms). The last column specifies the spring force constant.

In the above example:

```
DISTANCE    A/3/C4'    B/4/C3'     3     4     50
```

the pair of atoms (C4' from residue 3 in chain A & C3' from residue 4 in chain B) are bound by a spring of rest length of 3.5 Å and force constant of 50 kcal/mol/Å^2.
In case of:

```
DISTANCE    A/4/C2'    B/5/O3'     4     4     100
```

the pair of atoms (C2' from residue 4 in chain A & O3' from residue 5 in chain B) are bound by a spring of rest length of 4 Å and force constant of 100 kcal/mol/Å^2.

The restraints can be provided by mentioning the restraints in a file using `-m` flag in command-line:

```
$ QRNA -i <input_pdb_file> -m <restraints_file>
```

Example:

```
$ QRNA -i 1l2x.pdb -m 1l2x_restraints.txt
```

Configuration file (with `-c` flag) can also be used along with the restraints file. Alternatively, restraints file can be specified in configuration file. However, QRNAS will override the restraint file mentioned in configuration file if another restraint file is specified with `-m` flag in the command-line.

4.3 Positional restraints

QRNAS is able to restrain the positions of specified atoms. The two possibilities are “freezing” and/or “pinning down” individual atoms. These restraints can be implemented by altering the occupancy and B-factor column in the input pdb file.
If the occupancy of an atom is set equal to 0.00, its position is fixed / frozen, which means that will not be changed during the optimization.
If the occupancy is set between 0.00 and 1.00 , the residue is “pinned down” to its original position, and the B-factor value is treated as a radius of unrestricted movement from the starting position.
If the occupancy is set equal to 1.00, then the movement of the atom is not restricted (unless it is specified by other restraints).

After modifying the occupancy and B-factor columns of the pdb file, `-P` along with `-i` flag should be used to run QRNAS optimization as follows:

```
$ QRNA -P -i <input_pdb_file>
```

Example:

```
$ QRNA -P -i 1l2x_occu_modified.pdb
```

Here `1l2x_occu_modified.pdb` is the input structure with altered occupancy and B-factor columns.