rDock
Getting Started

rDock Development Team

March 11, 2014

Contents

1 Overview

rDock is a fast and versatile Open Source docking program that can be used against proteins and nucleic acids. It is designed for High Throughput Virtual Screening (HTVS) campaigns and Binding Mode prediction studies.

The rDock program was developed from 1998 to 2006 (formerly known as RiboDock[1]) by the software team at RiboTargets (subsequently Vernalis (R&D) Ltd). In 2006, the software was licensed to the University of York for maintenance and distribution. In 2012, Vernalis and the University of York agreed to release the program as Open Source software. This version is licensed under GNU-LGPL version 3.0 with support from the University of Barcelona - rdock.sourceforge.net.

The major components of the platform now include fast intermolecular scoring functions (van der Waals, polar, desolvation) validated against protein and RNA targets, a Genetic Algorithm-based stochastic search engine, a wide variety of external Structure-Based Drug Discovery (SBDD) derived restraint terms (tethered template, pharmacophore, noe distance restraints), and novel Genetic Programming-based post-docking filtering. A variety of scripts are provided to perform automated validation experiments and to launch virtual screening campaigns.

This introductory guide is aimed at new users of rDock. It describes the minimal set of steps required to build rDock from the source code distribution, and to run one of the automated validation experiments provided in the test suite distribution. The instructions assume that you are comfortable with simple Linux command line administration tasks, and with building Linux applications from make files. Once you are familiar with these steps you should proceed to the User and Reference Guide for more detailed documentation on the usage of rDock.

[1] Validation of an empirical RNA-ligand scoring function for fast flexible docking using RiboDock, SD Morley and M Afshar, J. Comput.-Aided Mol. Des., 18 (2004) 189-208.

2 Prerequisites

Compilers rDock is supplied as source code, which means that you will have to compile the binary files (run-time libraries and executable programs) before you can use them. rDock has been developed largely on the Linux operating system, most recently with the GNU g++ compiler (tested under openSUSE 11.3. The code will almost certainly compile and run under other Linux distributions with little or no modification. For the moment, it has been tested in latest Ubuntu and openSUSE releases for both 32 and 64 bits system architectures (by November 2013) and compilation was possible without any code modification. However, no other distributions or compilers have been tested extensively to date.

For full production use, you would typically compile rDock on a separate build machine and run the docking calculations on a cluster of compute machines. However, for the purposes of getting started, these instructions assume that you will be compiling rDock and running the initial validation experiments on the same machine.

Required packages Make sure you have the following packages installed on your machine before you continue. The versions listed are appropriate for openSUSE 11.3; other versions may be required for your particular Linux distribution.

Table 2.1: Required packages for building and running rDock

Package	Description	Required at	Version
gcc	GNU C compiler	Compile-time	> 3.3.4
g++	GNU C++ compiler	Compile-time	> 3.3.4
popt	C++ command-line argument processing	Run-time	> 1.7-190
popt-devel	Development files for popt library	Run-time	> 1.7-190
cppunit	C++ unit test framework	Compile-time	> 1.10.2
cppunit-devel	Development files for cppunit	Compile-time	> 1.10.2

3 Unpacking the distribution files

The rDock source files and test suite files are provided as independent gzipped tar (.tar.gz) distributions. Depending on your requirements, the two distributions can be unpacked to entirely separate locations, or can be unpacked under the same location. In this example they are unpacked under the same location.

Table 3.1: rDock distribution files

File	Description
rDock_[CODELINE]_src.tar.gz	rDock source distribution
[TEST]_rDock_TestSet.tar.gz	Test suite data files and scripts

where [CODELINE], and [TEST] will vary depending on the release and test set. [CODELINE] represents the major version string (for example, 2013.1) and [TEST] represents the given dataset (ASTEX, RNA or DUD).

Procedure 3.1. Example unpacking procedure

Create a new directory for building rDock.

```
$ mkdir ~/dev
```

The directory you created is referred to as [BUILDDIR] in the subsequents steps.

Copy or download the distribution files to [BUILDDIR].

```
$ cp ~/mydownloads/rDock_2013.1_src.tar.gz ~/dev/
```

Extract the distributions.

```
$ cd ~/dev/
$ tar -xvzf rdock_2013.1_src.tar.gz
```

The distributions contain files with relative path names, and you should find the following subdirectories created under rDock_[CODELINE]_src. Note that the ./rDock_2013.1_src subdirectory may have a different name depending on the major version string (see above).

```
$ find . -type d
.
./fw
./src
./src/daylight
./src/lib
./src/exe
./src/GP
./build
./build/test
./build/test/RBT_HOME
./build/tmakelib
./build/tmakelib/linux-pathCC-64
./build/tmakelib/linux-g++-64
./build/tmakelib/linux-g++
./build/tmakelib/unix
./data
./data/filters
./data/sf
./data/pmf
./data/pmf/smoothed
./data/scripts
./lib
```

```
./popt
./popt/intl
./popt/po
./popt/.deps
./popt/test3-data
./popt/autom4te.cache
./import
./import/tnt
./import/tnt/include
./import/simplex
./import/simplex/src
./import/simplex/include
./docs
./docs/images
./docs/newDocs
./include
./include/GP
./bin
```

Make a note of the following locations for later use.

The rDock root directory is [BUILDDIR]/rDock_[CODELINE]_src and will be referred to as [RBT_ROOT] in later instructions. In this example, [RBT_ROOT] is /dev/rDock_2013.1_src/.

4 Building rDock

rDock is written in C++ (with a small amount of C code from Numerical Recipes) and makes heavy use of the C++ Standard Template Library (STL). The majority of the source code is compiled into a single shared library (libRbt.so). The executable programs themselves are relatively light-weight command-line applications linked with libRbt.so.

The tmake build system (from Trolltech) is used to generate makefiles automatically for a particular build target (i.e. combination of operating system and compiler). The source distribution comes with tmake templates defining the compiler options and flags for three Linux build targets (linux-g++ and linux-g++-64). The build targets have been tested under openSUSE 11.3 (2.6.34.10-0.2 kernel) with GNU g++ (versions 3.3.4, 4.5.0 and 4.7.2).

Table 4.1: Standard tmake build targets provided

Target Name	Architecture	Compiler	Compiler flags (release build)
linux-g++	32-bit	g++	-m32 -O3 -ffast-math
linux-g++-64	64-bit	g++	-m64 -O3 -ffast-math

Customising the tmake template for a build target. If none of the tmake templates are suitable for your machine, or if you wish to customise the compiler options, you should first customise one of the existing templates. The tmake template files are stored under [RBT_ROOT]/build/tmakelib/. Locate and edit the tmake.conf file for the build target you wish to customise. For example, to customise the linux-g++ build target, edit [RBT_ROOT]/build/tmakelib/linux-g++/tmake.conf and localise the values to suit your compiler.

Procedure 4.1. rDock build procedure

To build rDock, first go to the [RBT_ROOT]/build/ directory.

```
$ cd [RBT_ROOT]/build
```

4.1.1. Compile

Make one of the build targets listed below.

```
$ make linux-g++
$ make linux-g++-64
```

4.1.2. Test

Run the rDock unit tests to check build integrity. If no failed tests are reported you should be all set.

```
$ make test
```

4.1.3. Cleanup (optional)

To remove all intermediate build files from [RBT_ROOT]/build/, leaving just the final executables (in [RBT_ROOT]/bin/) and shared libraries (in [RBT_ROOT]/lib/):

```
$ make clean
```

To remove the final executables and shared libraries as well, returning to a source-only distribution:

```
$ make distclean
```

5 Validation experiments

In this section (in rDock webpage) you will find the instructions about how to reproduce our validation experiments using different test sets. Three different sets were analyzed for three different purposes:

- ASTEX set for binding mode prediction in Proteins.
- RNA set for assess RNA-ligand docking.
- DUD set for database enrichment.

5.1 Binding Mode Prediction in Proteins

First of all, please go to sourceforge download page to download a compressed file with the necessary data.

After downloading the file `ASTEX_rDock_TestSet.tar.gz`, uncompress the file with the following command, which will create a folder called `ASTEX_rDock_TestSet`:

```
tar -xvzf ASTEX_rDock_TestSet.tar.gz
cd ASTEX_rDock_TestSet/
```

Here you will have the instructions for one of the systems (1sj0), to run with the rest of the systems, just change the pdb code with the one desired. Then, make sure that the necessary environmental variables for running rDock are well defined and run the following commands for entering to the folder and running rDock with the same settings that we have used:

```
cd 1sj0/

#first create the cavity using rbcavity
rbcavity -r 1sj0_rdock.prm -was > 1sj0_cavity.log

#then use rbdock to run docking
rbdock -r 1sj0_rdock.prm -p dock.prm -n 100 -i 1sj0_ligand.sd \
-o 1sj0_docking_out > 1sj0_docking_out.log

#sdsort for sorting the results according to their score
sdsort -n -f'SCORE' 1sj0_docking_out.sd > 1sj0_docking_out_sorted.sd

#calculate rmsd from the output comparing with the crystal structure of the ligand
sdrmsd 1sj0_ligand.sd 1sj0_docking_out_sorted.sd
```

5.2 Binding Mode Prediction in RNA

In a similar way of the section above, here you will find a brief tutorial on how to run rDock with the RNA TestSet used in the validation. As in the first section, please go to sourceforge download page to download a compressed file with the necessary data.

After downloading the file `RNA_rDock_TestSet.tar.gz`, uncompress the file with the following command, which will create a folder called `RNA_rDock_TestSet`:

```
tar -xvzf RNA_rDock_TestSet.tar.gz
cd RNA_rDock_TestSet/
```

Here you will have the instructions for one of the systems (1nem), to run with the rest of the systems, just change the pdb code with the one desired. Then, make sure that the necessary environmental variables for running rDock are well defined and run the following commands for entering to the folder and running rDock with the same settings that we have used (if you have run the previous set, the variables should already be correctly defined):

```
cd 1nem/

#first create the cavity using rbcavity
rbcavity -r 1nem_rdock.prm -was > 1nem_cavity.log
```

```
#then use rbdock to run docking
rbdock -r 1nem_rdock.prm -p dock.prm -n 100 -i 1nem_lig.sd \
-o 1nem_docking_out > 1nem_docking_out.log

#sdsort for sorting the results according to their score
sdsort -n -f'SCORE' 1nem_docking_out.sd > 1nem_docking_out_sorted.sd

#calculate rmsd from the output comparing with the crystal structure of the ligand
sdrmsd 1nem_lig.sd 1nem_docking_out_sorted.sd
```

5.3 Database Enrichment (actives vs decoys - for HTVS)

In this section you will find a brief tutorial on how to run rDock with the DUD TestSet used in the validation and how to perform different analysis of the results. As in the sections above, please go to sourceforge download page to download a compressed file with the necessary data.

After downloading the file `DUD_rDock_TestSet.tar.gz`, uncompress the file with the following command, which will create a folder called `DUD_rDock_TestSet`:

```
tar -xvzf DUD_rDock_TestSet.tar.gz
cd DUD_rDock_TestSet/
```

Here you will have the instructions for one of the systems (hivpr), to run with the rest of the systems, just change the DUD system code with the one desired. Then, make sure that the necessary environmental variables for running rDock are well defined and run the following commands for entering to the folder and running rDock with the same settings that we have used (if you have run the previous sets, the variables should already be correctly defined):

```
cd hivpr/

#first create the cavity using rbcavity
rbcavity -r hivpr_rdock.prm -was > hivpr_cavity.log
```

As the number of ligands to dock is very high, we suggest you to use any distributed computing environments, such as SGE or Condor, and configure rDock to run in multiple CPUs. Namely, split the input ligands file in as many parts as desired (very easy using sdsplit tool) and run independent rDock docking jobs for each "splitted" input file. However, for this example purpose, you will have the instructions for running all set of actives and decoys in one docking job:

```
#uncompress ligand file
gunzip hivpr_ligprep.sdf.gz

#use rbdock to run docking
rbdock -r hivpr_rdock.prm -p dock.prm -n 100 -i hivpr_ligprep.sdf \
-o hivpr_docking_out > hivpr_docking_out.log

#sdsort with -n and -s flags will sort internally each ligand by increasing
#score and sdfilter will get only the first entry of each ligand.
sdsort -n -s -fSCORE hivpr_docking_out.sd | sdfilter -f'$_COUNT == 1' >  hivpr_1poseperlig.sd

#sdreport will print all the scores of the output in a tabular format and,
#with command awk, we will format the results
sdreport -t hivpr_1poseperlig.sd | awk '{print $2,$3,$4,$5,$6,$7}' > dataforR_uq.txt
```

At this point, you should have a file called `hivpr_docking_out.sd` with all docking poses written by rDock (100 * input ligands), a file called `hivpr_1poseperlig.sd` with the best scored docking pose for each ligand and a file called `dataforR_uq.txt` that will be used for calculating ROC Curves using R. The next step is to calculate ROC Curves and other statistics. To do so, please visit section How to calculate ROC curves and jump to the subsection "R Commands for generating ROC Curves".