Hermes Documentation

Evan Senter

August 18, 2014

Contents

1 Introduction

This documentation aims to outline the basic dependencies, installation, and usage of the Hermes RNA software suite, as well as provide extended commentary on the flags and options available to code. Particular attention is payed to the invocation style of extension programs (using `multi_param`) to dispatch flags to the appropriate submodules. To get started as quickly as possible, see the quickstart one-liner in 2.1. Should you still have questions, you can reach the main author of the source code at evansenter@gmail.com.

2 Installation

2.1 Quick Start

From the root directory of Hermes, execute the following command:

```
cd build && cmake .. && make
```

If you encounter errors in configuring or compiling the software, we recommend checking section 2.4 for common troubleshooting solutions.

2.2 Dependencies

`cmake` (≥ 2.6-patch 4, tested through 3.0.0) `http://www.cmake.org/`

CMake is used as the build system for Hermes.

GNU99 compiler:

Most C compilers should support the `-std=gnu99` flag, which is required for GNU library extensions, particularly `unistd.h`

C++98 compiler:

C++98 support is necessary for proper struct initialization in `FFTbor2D`.

OpenMP support `http://openmp.org/wp/`

OpenMP support comes by default in most modern compilers, and is required for loop-optimization in `FFTbor2D`.

`GSL` (≥ 1.15, tested through 1.16) `http://www.gnu.org/software/gsl/`

GSL is required to compute the eigendecomposition of a (possibly) non-symmetric transition rate matrix for `RNAeq`.

`FFTW3` (≥ 3.3.4) `http://www.fftw.org/`

FFTW3 ([1]) functions are used to compute the inverse discrete Fourier transform in `FFTbor2D`.

`libRNA.a` (≥ 2.0.7, tested through 2.1.7) `http://www.tbi.univie.ac.at/RNA/`

Various ViennaRNA ([2]) functions and data structures are leveraged for homogenous energy model support, as well as `fold_par`, `pf_fold_par`, and `subopt_par`. We additionally make use of ViennaRNA functions to compute the necessary polynomial size for `FFTbor2D`, determined in the following fashion. Parameters K (resp. L) are defined to be the sum of the number of base pairs in reference structure A (resp. reference structure B) plus the number of base pairs in the maximum matching (Nussinov) structure which contains no base pair of A (resp. B).

2.3 Compilation

`cd build && cmake ..`

First, ensure that your system has the dependencies outlined above (the presence of CMake can be verified with `cmake --version`). While not required, is widely considered *best practice* to perform an out-of-source build, where the compilation of the code happens in a separate directory from the location of the source itself. To this end, `hermes` provides an empty `build` directory that can be used for compiling the code. From `hermes/build` execute `cmake` with the path to `hermes/CMakeLists.txt` (`..`) provided as argument.

`make`

Compiles the code, and generates binary executables for `FFTbor2D`, `RNAmfpt`, `RNAeq`, `FFTmfpt`, `FFTeq`, and `RateEq`. Additionally generates both static and shared libraries for `FFTbor2D`, `RNAmfpt`, and `RNAeq`. The output directory for binaries is `hermes/bin` and output directory for libraries is `hermes/lib`.

`make install` (optional)

Installs the executables built with `make` to `$DESTDIR/bin` and copies libraries / archives to `$DESTDIR/lib` (on *nix systems, `$DESTDIR` defaults to `/usr/local`).

2.4 Troubleshooting

While we have done the utmost to try and ensure that CMake is able to infer locations of third-party libraries and add compiler-appropriate flags in an automated fashion, due to the diversity of build environments possible, it is possible that you will need to specify additional command-line flags to `cmake` when generating the Makefiles in order to successfully build Hermes. The following are five useful flags for CMake, and a brief explanation of when they may need to be employed:

- The default compiler I'd like to use for C code is installed in a non-standard location, or not the globally default C compiler.

  CMAKE_C_COMPILER

  i.e. `cmake -DCMAKE_C_COMPILER=/path/to/c/compiler ..`

  This variable sets the path to the compiler to use for configuration and subsequent compilation via `make`. This is the compiler that will be used by CMake to test for the presence of various flags, i.e. `-O3` and `-Wall`.

- The default compiler I'd like to use for C++ code is installed in a non-standard location, or not the globally default C++ compiler.

  CMAKE_CXX_COMPILER

  i.e. `cmake -DCMAKE_CXX_COMPILER=/path/to/cxx/compiler ..`

  Same as above.

- I'm getting a compile-time error indicating undefined symbols for `_get_iindx`, `_maximumMatchingConstraint` or something similar.

  CMAKE_LIBRARY_PATH

  i.e. `cmake -DCMAKE_LIBRARY_PATH=/dir/for/libRNA-2.0.7+/ ..`

  These are `libRNA.a` symbols specific to the 2.0+ release of ViennaRNA. In all cases identified thus far, this error means that the version of `libRNA.a` found by CMake is not out of date, and can be resolved by explicitly providing the library path to a 2.0+ ViennaRNA static library using the `CMAKE_LIBRARY_PATH` flag.

- Libraries required by Hermes are not installed in a location visible by the linker (in LD_LIBRARY_PATH), and CMake is unable to validate their existence.

  CMAKE_LIBRARY_PATH

  i.e. `cmake -DCMAKE_LIBRARY_PATH="/more/libraries;/even/more/libraries" ..`

  In the case when libraries required by Hermes are not visible to CMake, or the global library is an out-of-date version, it is possible to provide hints to the build tool for additional directories to search. Directories specified by the CMAKE_LIBRARY_PATH flag will be prepended onto the linker search path, and thus override global matches (handling the case where default libraries aren't sufficiently up to date). When desiring to provide multiple locations to search for libraries, CMake uses the semicolon (;) character as a separator and the entire string should be quoted to escape the shell environment.

- Headers required by Hermes are not installed in a location visible by the compiler (in CPATH or a derivative), and as a result I'm seeing `undefined reference to` errors.

  CMAKE_INCLUDE_PATH

  i.e. `cmake -DCMAKE_INCLUDE_PATH="/more/includes;/even/more/includes" ..`

  It is generally likely that the CMAKE_LIBRARY_PATH and CMAKE_INCLUDE_PATH will both be necessary, when either one is required. This flag operates in a fashion identical to CMAKE_LIBRARY_PATH described above, and uses the same syntax. Alternatively a user can update their `CPATH` environment variable, but this may have unpredictable results when headers are found, but out of date.

- I don't have permissions to `make install` to the default location (generally `/usr/local` for *nix) on my system.

  CMAKE_INSTALL_PREFIX

  i.e. `cmake -DCMAKE_INSTALL_PREFIX=/make/install/path/prefix ..`

  This variable sets the destination directory of the `make install` command. Binaries will be placed in the `bin` subdirectory and libraries will be placed in the `lib` subdirectory. This is analogous to `./configure --prefix=/make/install/path/prefix` in Autotools and can also be achieved by setting the `DESTDIR` environment variable.

3 Software Organization

3.1 General Principles

The Hermes code is organized into two major directories, `hermes/src` and `hermes/ext`. The conceptual difference between these two directories is that `hermes/src` code (`FFTbor2D`, `RNAmfpt`, `RNAeq`) are all stand-alone pieces of software which achieve specific goals (computing energy grids, hitting time, and equilibrium time respectively). Alternatively, code residing in `hermes/ext` aims to leverage general concepts across the Hermes package to provide a means to ask even more specific questions. In example, `FFTbor2D` allows an investigator to compute the 2D energy grid correspondent to an input RNA sequence s and two structures A, B. `RNAeq` can estimate the population occupancy of A, B for s by either sampling suboptimal structures or exhaustive structural enumeration, but these approaches aren't tractable for non-trivial RNAs.

`FFTeq`, located in `hermes/ext/population_from_fftbor2d` uses functions from `libfftbor_static.a` (derived from `FFTbor2D`) to compute the energy landscape and functions from `librnaeq_static.a` (derived from `RNAeq`) to estimate population occupancy for s, A, B without requiring an investigator to copy pieces of individual packages to achieve their goals, instead using the static libraries automatically produced for all `hermes/src` software, shared headers made available in `hermes/h` and `libmulti_param.a` (from `hermes/src/multi_param`) to dispatch command-line arguments to the appropriate underlying function. The result? The entirety of `FFTeq` is 67 lines of C++ code, and uses native binary data structures the entire way through; there is no command-line funneling of `FFTbor2D` into `RNAeq`.

3.2 multi_param Overview

All of `FFTbor2D`, `RNAmfpt`, and `RNAeq` present a wide selection of command-line arguments to meet the diverse demands of end users. When we moved on to developing *extension* software between these three programs, there were a number of requirements that we came up with to make both implementing and using these programs as easy as possible. We decided that *a*) the developer should not have to reimplement command-line parsing for extensions *b*) all existing flags for underlying libraries (i.e. `FFTbor2D`, `RNAmfpt`, `RNAeq`) would be supported, and *c*) flags would be namespaced to have deterministic targets.

To achieve these goals, the library `multi_param` was developed. This library simply takes a collection of command line arguments and re-dispatches them to the appropriate underlying library. Given a set of command-line flags such as `--all-v --fftbor2d-i GGGAAACCC --fftbor2d-j ‘.........’ --fftbor2d-k ‘(((...)))’ --mfpt-x --mfpt-h`, the code ensures that `FFTbor2D` is passed `-v -i GGGAAACCC -j ‘.........’ -k ‘(((...)))’` as options in `argv` and `RNAmfpt` is passed `-v -x -h` as options in `argv`.

3.3 multi_param Details

An example from `hermes/ext/mfpt_from_fftbor2d/mfpt_from_fftbor2d.cpp`:

```
PARAM_CONTAINER* params;
FFTBOR2D_PARAMS fftbor2d_params;

/* ...omitted for clarity... */

char* subparams[] = { "fftbor2d", "mfpt" };
```

```
params = split_args(argc, argv, subparams, 2);

fftbor2d_params = init_fftbor2d_params();
parse_fftbor2d_args(
  fftbor2d_params,
  params[0].argc,
  params[0].argv,
  &mfpt_from_fftbor2d_usage
);
```

The way this code works is by first declaring which packages can be used by the extensions, out of `fftbor2d`, `mfpt`, or `population`. In the snippet above the selection is saved in the variable `subparams`. All of the `FFTbor2D`, `RNAmfpt`, and `RNAeq` libraries have `init_*_params` functions available, which return an object with the default parameters for that package. They also all have `parse_*_args` functions, which take three arguments, *1*) a pointer to the parameters object *2*) `argc`, and *3*) `argv` .

`split_args` takes the prefixed command-line arguments (see the example in 3.2) and bins them by their prefix, with the special `--all` prefix being supplied to all declared subparams. The prefixes are then removed and the grouped arguments are returned from the function in a `PARAM_CONTAINER` array, with the same order as the subparams were passed to the function. It is then trivial to call the `parse_*_args` functions with the corresponding subarrays from `PARAM_CONTAINER` to get a final parameters object.

Leveraging the example from 3.2 a final time, the variable `params` would look as follows after invoking `split_args`:

```
[
  {
    argv: ['-v', '-i', 'GGGAAACCC', '-j', '.........', '-k', '(((...)))'],
    argc: 7
  }, {
    argv: ['-v', '-x', '-h'],
    argc: 3
  }
]
```

4 Core Programs

4.1 FFTbor2D

4.1.1 Applications

FFTbor2D ([3]) computes the 2D energy landscape of an arbitrary but fixed sequence s, parameterized by base pair distance from input structures A and B.

4.1.2 Example Usage

```
FFTbor2D -m -i GGGAAACCC -j '.........' -k '(((...)))' -e ./misc/rna_turner2004.par

+0.00000000  +0.00000000  +0.00000000  +0.10047371  +0.00000000  +0.00000000  ...
```

```
+0.00000000  +0.00000000  +0.00019384  +0.00000000  +0.00181492  +0.00000000  ...
+0.00000000  +0.02666435  +0.00000000  +0.00036362  +0.00000000  +0.16642614  ...
+0.70406341  +0.00000000  +0.00000000  +0.00000000  +0.00000000  +0.00000000  ...
+0.00000000  +0.00000000  +0.00000000  +0.00000000  +0.00000000  +0.00000000  ...
+0.00000000  +0.00000000  +0.00000000  +0.00000000  +0.00000000  +0.00000000  ...
+0.00000000  +0.00000000  +0.00000000  +0.00000000  +0.00000000  +0.00000000  ...
```

In the above example, position $(0, 0)$ is located in the upper-left of the output matrix, and base pair distance from A (resp. B) moves along the rows (resp. columns). This can be verified using the `-c` flag for CSV output instead of matrix formatting (using `-m`):

```
FFTbor2D -c -i GGGAAACCC -j '.........' -k '(((...)))' -e ./misc/rna_turner2004.par

0,3,+0.10047371
1,2,+0.00019384
1,4,+0.00181492
2,1,+0.02666435
2,3,+0.00036362
2,5,+0.16642614
3,0,+0.70406341
```

4.1.3 Options

- `-i` *(requried)* The RNA sequence s to be used by `FFTbor2D`.
- `-j` *(requried)* The first structure A to be used by `FFTbor2D`. Any base pairs in A incompatible with s (i.e. not a Watson-Crick or GU wobble) are ignored by `FFTbor2D`.
- `-k` *(requried)* The second structure B to be used by `FFTbor2D`. The same restrictions apply as with `-j`.
- `-t` The temperature at which the energy landscape is computed. The provided value is used both in computing Boltzmann factors ($\beta = -1/\text{RT}$) and for energy table lookups. The default is 37°C.
- `-e` The energy file to be used by `FFTbor2D`, in the Vienna 2.0 format. The default is `rna_turner2004.par` located in the same directory as the `FFTbor2D` executable.
- `-p` The precision m of the probabilities output by `FFTbor2D`, in base 2 format. The default is system specific (you can call `FFTbor2D` with no parameters to see the precision range available on your platform), and using 0 disables any precision control—not recommended. The default is 27, which corresponds to 8 decimal places of precision.
- `-b` Enables the output of performance related benchmarking for all major subroutines in `FFTbor2D`.

Output formatting flags (mutually exclusive):

> The default format is a verbose version of `-s` which also includes the user-input s, A, B, and column headers.

  - `-c` CSV-formatted output, where only non-zero positions are emitted. The first (resp. second) column corresponds to base pair distance from A (resp. B), and the final column is the Boltzmann probability ($p(Z_{k,l}/Z)$) for that position.
  - `-m` Output formatted as a matrix, where $(0, 0)$ is in the lop-left and base pair distance from A (resp. B) moves along the rows (resp. columns).
  - `-s` Output formatted in the same fashion as `-c`, but delimited with tab characters. The rightmost column is the ensemble free energy ($-\mathrm{RT}\log(Z_{k,l})$) in $\frac{\mathrm{kcal}}{\mathrm{mol}}$.
- `-v` Enables verbose output.

4.2 RNAmfpt

4.2.1 Applications

`RNAmfpt` computes the mean first passage time (hitting time) of an user-provided input matrix, in CSV format. The user can provide as input either a 2D energy grid (such as those produced by `FFTbor2D` with the `-c` format) or a transition probability matrix. The default expectation is a 2D energy grid composed of Boltzmann probabilities, though ensemble free energies are alternatively supported with `-e`.

If providing a transition probability matrix M as input (with the `-t` flag), the format is still expected to be in CSV form, where columns i, j, p represent the 0-indexed row-based probabilities for M, s.t. $M_{i,j} = p_{i \to j} = (i, j, p_{i \to j})$. Only non-zero $M_{i,j}$ entries are required.

We also provide an extension of `RNAmfpt` through `FFTmfpt` (5.1) which leverages the 2D energy landscape output by `FFTbor2D` (4.1) to estimate mean first passage time with good results.

4.2.2 Example Usage

These trivial examples make use of the file `./src/mfpt/example.csv`, which is the output of calling:

```
FFTbor2D -c -i GGGAAACCC -j ’.........’ -k ’(((...)))’ -e ./misc/rna_turner1999.par
```

```
0,3,+0.10531278
1,2,+0.00213850
1,4,+0.00509026
2,1,+0.28092942
2,3,+0.00032964
2,5,+0.15262286
3,0,+0.45357654
```

A simple invocation of `RNAmfpt` appears like:

```
RNAmfpt -c ./src/mfpt/example.csv -xh
```

```
+699.65148561
```

In the above example, the value output by `RNAmfpt` is the approximate hitting time for the 2D energy grid provided with the `-c` flag. `RNAmfpt` converts this input to a transition probability matrix where only single base pair substitutions are valid (diagonal moves only using `-x`), and the Hastings correction is applied with `-h`.

4.2.3 Options

`-c` *(requried)* Path to the input CSV file.

Input format flags (mutually exclusive):

The default expectation is a 0-indexed 2D probability landscape in the CSV format described within 4.2.1. In this case, the transition probability is defined:

$$p_{a \to b} = \min(1, \exp(-(p_b/p_a)/\mathrm{RT}))/N_a \tag{1}$$

where p_a, p_b are Boltzmann probabilities and N_a is the number of neighbors of a.

`-e` The input matrix is comprised of ensemble free energies rather than Boltzmann probabilities. If this flag is provided, the transition probability is defined:

$$p_{a \to b} = \min(1, \exp(-(E_b - E_a)/\mathrm{RT}))/N_a \tag{2}$$

where definitions follow as above.

`-t` The input CSV file is a transition probability matrix already, and `RNAmfpt` should not try to convert it into one. In this case, the first two columns in the CSV file are 0-indexed row-order indices into the transition probability matrix, and the third (final) column is the transition probability $p_{a \to b}$.

Transition probability matrix manipulation:

`-x` Only permit single base pair moves: $(i, j) \to (k, l) \implies (k, l) \in (i \pm 1, j \pm 1)$. This option assumes that the input is not already a transition probability matrix, the matrix is parameterized by base pair distance to some fixed, implicit A, B and that the matrix already satisfies the triangle inequality and parity condition.

`-f` The transition probability matrix generated by `RNAmfpt` should be fully connected.

`-t` The input CSV file is a transition probability matrix already, and `RNAmfpt` should not try to convert it into one. In this case, the first two columns in the CSV file are 0-indexed row-order indices into the transition probability matrix, and the third (final) column is the transition probability $p_{a \to b}$.

`-h` Enables the usage of the Hastings adjustment in formulating the transition probability matrix. Has no effect if the input is already a transition matrix (using `-t`) or fully connected (using using `-f`). When enabled, the transition probabilities are defined as:

$$p_{a \to b} = \min(1, (N_a/N_b) \times \exp(-(p_b/p_a)/\mathrm{RT}))/N_a \tag{3}$$

where variable definitions are the same as described above. Computation of N_a, N_b respects grid boundaries and the triangle inequality, with base pair distance $d_{bp}(A, B)$ inferred from the input energy landscape by looking for a non-zero $(0, d_{bp}(A, B))$ or $(d_{bp}(A, B), 0)$ position. Also works with the `-e` flag to use energies instead of probabilities.

Using ϵ to inflate the energy landscape:

In instances where no direct path exists from A to B in single base pair steps, or one would like to adjust the energy landscape such that all valid positions are accessible, we provide a number of flags to make an ϵ adjustment to the input 2D probability landscape. Valid positions are defined as:

$$V = \{(i,j) \mid 0 \leq i,j \leq n \land i+j \geq d_{bp}(A,B) \land i+j \equiv d_{bp}(A,B) \pmod 2\} \qquad (4)$$

where n is set with `-n` and $d_{bp}(A,B)$ with `-d` (if it can't be inferred). All probabilities $\{p_{i,j} \mid (i,j) \in V\}$ are then ϵ-adjusted such that $p^*_{i,j} = \frac{p_{i,j}+\frac{\epsilon}{|V|}}{1+\epsilon}$.

`-d` Flag to explicitly provide $d_{bp}(A,B)$, as defined in `-h`. Only required when $d_{bp}(A,B)$ can't be inferred from the input probability matrix by identifying a non-zero $(0, d_{bp}(A,B))$ or $(d_{bp}(A,B), 0)$ position in the input.

`-n` Maximum base pair distance n to use when including all accessible positions.

`-o` The *total* ϵ probability to add to the input probability grid before renormalization. The *per-position* contribution will be $\frac{\epsilon}{|V|}$, where definitions are as above.

Mean first passage time options:

`-a` The 0-indexed position of $(0, d_{bp}(A,B))$ in the input CSV file, representing the starting state for folding. If this flag is not provided, `RNAmfpt` will attempt to infer it by looking for a non-zero $(0, d_{bp}(A,B))$ position in the input and present an error to the user if it can't be found. If the input is already a transition matrix (using `-t`), `-a` should be the index of the row / column correspondent to the start state.

`-z` The 0-indexed position of $(d_{bp}(A,B), 0)$ in the input CSV file, representing the target state for folding. Expectations are identical as with `-a`.

`-r` Compute a transition rate matrix rather than a probability matrix. The rate matrix M is defined such that $q_{i,i} = -\sum_{j \neq i} q_{i,j}$. This flag is provided so that `RNAeq` and extensions involving rate matrices can use a unified codebase for their generation, and is not intended for typical calls to `RNAmfpt`.

`-l` Prints all mean first passage times $\{\text{mfpt}_{X \to B} \mid X \in V \land X \neq B\}$ where V is the set of valid positions for the given input.

`-v` Enables verbose output.

4.3 `RNAeq`

4.3.1 Applications

`RNAeq` is a tool to compute the population occupancy and equilibrium time for an RNA sequence, transitioning from a starting distribution wholly occupied by an arbitrary input structure A (the empty structure by default). From this starting distribution, occupancy curves can be generated for any arbitrary structure B, which is taken to be the MFE by default. The resulting population occupancy curves can be used to estimate the equilibrium time as well using a sliding window approach.

By default, this code takes an input RNA sequence s and uses `RNAsubopt` from the Vienna package to generate all secondary structures for s, making its use practical only for trivially small sequences. Alternatively an energy band can be enabled to only sample structures within $\Delta \frac{\text{kcal}}{\text{mol}}$ of the MFE structure using `-c`. Even then, it is quite possible that exponential number of secondary structures makes a sampling-based approach intractable for anything but small

RNAs, highlighting the difficulty of performing computational kinetics. In this case it's recommended to use a notion of macrostates, whereby the granularity of the transition space is decreased to make software more performant, at the expense of accuracy. The extension program `FFTeq` (5.2) uses the 2D energy landscape from `FFTbor2D` (4.1) to good effect, and more generally `RateEq` (5.3) allows the user to input an arbitrary rate matrix to perform population kinetics on, allowing for a user-defined generalization of macrostates.

4.3.2 Example Usage

A simple invocation of `RNAeq` appears like:

```
RNAeq -o -s GGGAAACCC -l '(((...)))' -p 0.25

-3.250000 +0.00056212 +0.99856051
-3.000000 +0.00099929 +0.99744193
-2.750000 +0.00177604 +0.99545654
-2.500000 +0.00315519 +0.99193786
-2.250000 +0.00560102 +0.98571804
-2.000000 +0.00992932 +0.97477461
-1.750000 +0.01756010 +0.95567878
-1.500000 +0.03092313 +0.92284200
-1.250000 +0.05404704 +0.86782054
-1.000000 +0.09322305 +0.77973465
-0.750000 +0.15714791 +0.64949189
-0.500000 +0.25475008 +0.48144406
-0.250000 +0.38729984 +0.30830977
+0.000000 +0.53394003 +0.18159969
+0.250000 +0.64774409 +0.12138539
+0.500000 +0.69618181 +0.10316965
+0.750000 +0.70383144 +0.10055275
```

Invoking `RNAeq` in this fashion produces output in three columns representing *a)* $\log_{10} t$, where t is arbitrary time *b)* $p(B, \log_{10} t)$, the population occupancy of B (defaulting to the MFE) at time $\log_{10} t$, and *c)* $p(A, \log_{10} t)$, the population occupancy of A (defaulting to the empty structure) at time $\log_{10} t$. The `-o` flag indicates that we will permit `RNAsubopt` to sample structures having lonely base pairs, and `-p` indicates that we would like $\log_{10} t$to increment in steps of 0.25. As before, all available options are described in detail below.

4.3.3 Options

`-g` `RNAeq` will only produce the eigenvalues for the rate matrix, rather than produce population occupancy values or equilibrium times. The is much faster because the eigenvector matrix does not need to be inverted.

`-a` Indicates the 0-indexed position in the rate matrix corresponding to the structure S for which $p(S, t_0) = 1$. By default this is taken from `-k`, but this flag is of use in extensions where precomputed rate matrices are provided as command-line input.

`-z` Indicates the 0-indexed position in the rate matrix corresponding to the target structure S for which population occupancy curves should be computed or equilibrium times estimated. By default this is taken from `-l`, but this flag is of use in extensions where precomputed rate matrices are provided as command-line input.

`-r` Serialization direction, can be one of -1, 1. When 1, the eigensystem and its inversion will be serialized using `TPL` to the binary file specified named by `-f`. When -1, the file named by `-f` will be loaded and used as the eigensystem of interest for the invocation of `RNAeq`. This allows repeated analysis of the kinetic characteristics of a given rate matrix without redundant computation of the eigendecomposition.

`-f` The filename for the binary serialization of the eigensystem described by `-r`. Depending on the value of `-r`, either specifies a filename to read from (when `-r` is -1) or write to (when `-r` is 1).

Input format flags:

`-s` *(requried)* The RNA sequence s to be used by `RNAeq`.

`-k` The starting structure for `RNAeq` refolding. Represents the structure S for which $p(S, t_0) = 1$. If not explicitly provided, the empty structure is used.

`-l` The target structure to be used by `RNAeq` when computing population occupancy curves or equilibrium time. If not explicitly provided, the MFE structure is computed using `RNAfold` and used.

`-c` The `RNAsubopt` sampling threshold Δ (in $\frac{\text{kcal}}{\text{mol}}$) such that only suboptimal structures with energy less than $E_{MFE} + \Delta$ are generated. If not provided, all suboptimal structures are generated.

`-o` Allow `RNAsubopt` to sample structures containing lonely base pairs.

Managing the population occupancy time window:

As described in 4.3.2, all time values presented by `RNAeq` are in $\log_{10}$-time, so a time point of 10^6 would be presented in the output as 6. This format is maintained across all of `RNAeq`, and thus any flags dealing with time accept values in $\log_{10}$-time, similar to how the output times for population occupancy and equilibrium are in $\log_{10}$-time.

By default, `RNAeq` requires no configuration to reasonably estimate the time range of interest for arbitrary population curves. This is done by using what we refer to as a *soft bounding* heuristic. Due to our observed numeric instability when working with floating point numbers beyond $\log_{10}$-time $(-12, 12)$, `RNAeq` sets two global timepoints, $t_0 \approx -10$ (`-i`) and $t_{\text{inf}} \approx 10$ (`-j`). It is unreasonable to expect that the time range $[t_0, t_{\text{inf}}]$ to always be of interest to the investigator, regardless of input sequence. On the contrary, there are many cases where an RNA rapidly reaches equilibrium, and large regions to the left (resp. right) of the sigmoidal curve have a $\frac{dp}{dt} \approx 0$, perhaps due to insufficient time to start refolding (resp. equilibrium having been achieved).

`RNAeq` prunes out these regions by identifying two positions, a t_{start} value for which $|p(t_0) - t_{\text{start}}| \approx \delta$, and corresponding t_{stop} for which $|p(t_{\text{inf}}) - t_{\text{stop}}| \approx \delta$. In both instances, δ is taken to be 10^{-3} by default. In our experience, the time range $[t_{\text{start}}, t_{\text{stop}}]$ is an effective *soft bound* for the population occupancy curve which helps to present only the window of interest to the investigator. If desired, it is possible to disable the soft bounding feature using `-n`, which will present the entire timespan $[t_0, t_{\text{inf}}]$ to the user, where t_0 can be set in $\log_{10}$-time with `-i` (default -10), and t_{inf} can be set in $\log_{10}$-time with `-j` (default 10).

`-i` The $\log_{10}$-time to represent t_0, -10 by default. Serves as a hard bound to prevent numeric instability, values ≤ -12 are not recommended.

`-j` The $\log_{10}$-time to represent t_{inf}, 10 by default. Serves as a hard bound to prevent numeric instability, values ≥ 12 are not recommended.

`-p` The step size i to use when computing population occupancy curves. Population occupancy is computed at discrete times $(t_0, t_0 + i, t_0 + 2i, \cdots, t_{\text{inf}} - i, t_{\text{inf}})$.

`-d` δ value used to estimate *soft bounds* t_{start} and t_{stop} for the computed population kinetics. Defaults to 10^{-3}.

`-n` Disables the usage of soft bounds (estimated with `-d`). The full time range $[t_0, t_{\text{inf}}]$ (`-i` to `-j`) is output.

Equilibrium computation:

`RNAeq` allows for the estimation of equilibrium time from an implicitly computed population occupancy curve for a target structure (`-l`), refolding from a starting structure (`-k`). The equilibrium time is estimated using a sliding window approach, starting from the *right soft bound* t_{stop} (see documentation above for definition). The sliding window (default size: 5, starting at t_{stop}) is used in the following fashion.

If the probabilities at *all* positions $(i+1, i+2, \cdots, i+\texttt{window\_size}-1)$ are within ϵ of $p(i)$, the window starting at i is said to be in equilibrium (we will call this *satisfying the equilibrium condition*), and moved to the left. This process is continued until a window starting at some j is found that is not at equilibrium, in which case $j + 1$, the last time the window equilibrium was satisfied, is returned. Should the window starting at t_{stop} not be in equilibrium already, the window is instead moved to the right until a position j is encountered for which the equilibrium condition is satisfied, and returned. Should the sliding window ever encounter the hard bound t_0 (resp. t_{inf}), the equilibrium time is said to be $-\infty$ (resp. ∞).

`-q` Compute the estimated equilibrium time for the refolding of input sequence s from A to B.

`-h` Equilibrium is considered for all macrostates, rather than just the refolding of A to B. A window starting at index i is then said to be in equilibrium if the population occupancy curves for all macrostates M^* s.t. $M^* = M - \{A\}$ are in equilibrium at i.

`-e` The maximum deviation ϵ that all window indexes may have with the left-most position while still being considered *in equilibrium.* Defaults to 10^{-4}.

`-w` The window size used for predicting equilibrium. Equilibrium is considered as having been achieved when all indexes $(i + 1, i + 2, \cdots, i + \texttt{window\_size} - 1)$ have values within ϵ (`-e`) of the population proportion at index i.

`-t` The temperature at which structures as sampled via `RNAsubopt`. The default is 37°C.

`-b` Enables the output of performance related benchmarking for all major subroutines in `RNAeq`.

`-v` Enables verbose output.

5 Extensions

5.1 `FFTmfpt`

5.1.1 Applications

`FFTmfpt` can be used to estimate the mean first passage time of a given RNA in the same fashion as `RNAmfpt`, but using the 2D energy landscape computed by `FFTbor2D` as the underlying graph for the Markov chain. We expect this extension to be particularly useful to investigators interested in leveraging the speed of `FFTbor2D` as a coarse-grained representation of the refolding landscape in order to quickly estimate hitting times for a large collection of sequences, where speed is a major factor.

`FFTmfpt` uses the `multi_param` framework to accept all command line flags available for `FFTbor2D` (prefixed by `--fftbor2d-`) and `RNAmfpt` (prefixed by `--mfpt-`). The only required flags are `--fftbor2d-i` (input sequence s), `--fftbor2d-j` (starting structure A), `--fftbor2d-k` (target structure B), and a move type for the Markov chain, one of `--mfpt-x` or `--mfpt-f`. See the command-line options documentation for `FFTbor2D` (4.1.3) and `RNAmfpt` (4.2.3) for a full description of all available settings.

5.1.2 Example Usage

In the following example, we compute the estimated hitting time of a toy 9 nt. sequence permitting only single base pair moves and applying the Hastings adjustment to the transition probability matrix.

```
FFTmfpt --fftbor2d-i GGGAAACCC --fftbor2d-j "........." --fftbor2d-k "(((...)))" \
--mfpt-x --mfpt-h

6786.738209
```

To highlight the speed of computing kinetics in this fashion, consider the following 76 nt. tRNA sequence for which the mean first passage time is estimated in approximately one second of wall time on a laptop!

```
time FFTmfpt --mfpt-x --mfpt-h \
--fftbor2d-i \
’GCCUCCUUAGCGCAGUAGGUAGCGCGUCAGUCUCAUAAUCUGAAGGUCCUGAGUUCGAACCUCAGAGGGGGCACCA’ \
--fftbor2d-j \
’............................................................................’ \
--fftbor2d-k \
’(((((((..((((........)))).((((.........)))).....(((((.......))))))))))))....’

4509.578593

real 0m1.146s
user 0m3.743s
sys 0m0.039s
```

5.2 FFTeq

5.2.1 Applications

The intention of FFTeq is to present an efficient estimator of population occupancy and equilibrium time for a given RNA sequence in a similar fashion to FFTmfpt (5.1.1), making use of the energy grid computed by FFTbor2D as a coarse grained graph for the Markov process. As with FFTmfpt (5.1.1), the only required flags are --fftbor2d-i (input sequence *s*), --fftbor2d-j (starting structure *A*), and --fftbor2d-k (target structure *B*).

FFTmfpt uses the multi_param framework to accept all command line flags available for FFTbor2D (prefixed by --fftbor2d-), RNAmfpt (prefixed by --mfpt-), and RNAeq (prefixed by --population-).

5.2.2 Example Usage

FFTeq can be used to generate population occupancy curves in the following manner:

```
FFTeq --fftbor2d-i GGGAAACCC --fftbor2d-j "........." --fftbor2d-k "(((...)))" \
--population-p 0.5

-1.500000 +0.00000001 +0.99937772
-1.000000 +0.00000029 +0.99809540
-0.500000 +0.00000753 +0.99449426
+0.000000 +0.00014023 +0.98557561
+0.500000 +0.00145114 +0.96352653
+1.000000 +0.00733188 +0.90182966
+1.500000 +0.02430557 +0.74360243
+2.000000 +0.06459245 +0.47569262
+2.500000 +0.14819884 +0.31072886
+3.000000 +0.33391730 +0.23696313
+3.500000 +0.60157943 +0.13826409
+4.000000 +0.70229737 +0.10112492
```

It can also estimate equilibrium time, just as RNAeq can:

```
FFTeq --fftbor2d-i GGGAAACCC --fftbor2d-j "........." --fftbor2d-k "(((...)))" \
--population-q

3.995000
```

To highlight the speed of FFTeq, consider the following real-world example, using the same tRNA as in the FFTmfpt (5.1.2). The equilibrium time is estimated from the population occupancy curve in approximately 10 seconds on a laptop!

```
time FFTeq --population-q \
--fftbor2d-i \
'GCCUCCUUAGCGCAGUAGGUAGCGCGUCAGUCUCAUAAUCUGAAGGUCCUGAGUUCGAACCUCAGAGGGGGCACCA' \
--fftbor2d-j \
'............................................................................' \
--fftbor2d-k \
’(((((((..((((........)))).((((.........)))).....(((((.......))))))))))))....’

3.111000

real 0m10.550s
user 0m13.258s
sys 0m0.067s
```

5.3 RateEq

5.3.1 Applications

The program `RateEq` allows an investigator to compute population occupancy curves and equilibrium times using the underlying functions in `RNAeq`, but providing their own transition rate matrix. This allows users to easily use the Hermes framework for kinetics applications beyond its current scope, and allows for arbitrary generalizations of macrostates or exhaustive kinetics through a single tool, similar to the generality of `RNAmfpt`.

`RateEq` accepts command-line options for both `RNAmfpt` (using `--mfpt-`) and `RNAeq` (using `--population-`), where `RNAmfpt` functions are used to convert the rate matrix (provided via CSV with the `--mfpt-c` flag) to a generalized data structure, and `RNAeq` performs the required eigendecomposition. The only required flags are `--mfpt-c` to provide the input rate matrix, in the CSV format described in 4.2.1, and `--population-a` (resp. `--population-z`) to indicate the 0-indexed position in the rate matrix correspondent to the starting state (resp. ending state), described in greater detail in 4.3.3.

5.3.2 Example Usage

These trivial examples make use of the following CSV file.

```
./ext/population_from_rate_matrix/all_structures_for_gggaaaccc\
__19_5_rate_move_with_hastings.csv
```

This file is the rate matrix derived from all compatible structures for the toy sequence GGGAAACCC, similar to the matrix computed by `RNAeq` when invoked like `RNAeq -s GGGAAACCC -o`. The index representing the empty structure (starting state) is 19 and the MFE structure is located at index 5.

```
RateEq --population-a 19 --population-z 5 --population-p 0.5 --mfpt-c \
./ext/population_from_rate_matrix/all_structures_for_gggaaaccc\
__19_5_rate_move_without_hastings.csv

-2.000000 +0.00000000 +0.99878366
-1.500000 +0.00000014 +0.99619920
-1.000000 +0.00000415 +0.98841027
-0.500000 +0.00010582 +0.96692649
+0.000000 +0.00184091 +0.91694937
+0.500000 +0.01529280 +0.80912513
+1.000000 +0.05796537 +0.57473609
+1.500000 +0.13220381 +0.27178670
```

```
+2.000000 +0.23416181 +0.16602654
+2.500000 +0.41551053 +0.12445470
+3.000000 +0.57547984 +0.08935250
+3.500000 +0.59543269 +0.08497569
```

The above example computes the population occupancy curves for the given rate matrix (in $\log_{10}$-time increments of 0.5, specified by the `--population-p 0.5` flag), while the invocation below estimates the equilibrium time for state 5 (representing the MFE structure), starting from state 19 (representing the empty structure).

```
RateEq --population-a 19 --population-z 5 --population-q --mfpt-c \
./ext/population_from_rate_matrix/all_structures_for_gggaaaccc\
__19_5_rate_move_without_hastings.csv
```

```
3.235000
```

6 References

[1] M. Frigo and S.G. Johnson. The design and implementation of fftw3. *Proceedings of the IEEE*, 93(2):216231.

[2] R. Lorenz, S. H. Bernhart, C. Honer Zu Siederdissen, H. Tafer, C. Flamm, P. F. Stadler, and I. L. Hofacker. ViennaRNA Package 2.0. *Algorithms Mol Biol*, 6:26, 2011.

[3] E. Senter, I. Dotu, and P. Clote. RNA folding pathways and kinetics using 2D energy landscapes. *J Math Biol*, Feb 2014.