RIsearch2: User manual

Ferhat Alkan [1,2], Anne Wenzel [1,2], Oana Palasca [1,2,3],
Peter Kerpedjiev [4], Anders Frost Rudebeck [1,2], and Jan Gorodkin [1,2,*]

[1]Center for non-coding RNA in Technology and Health,

[2]Department of Veterinary Clinical and Animal Science,

University of Copenhagen, Grønnegårdsvej 3, 1870 Frederiksberg C, Denmark,

[3]Novo Nordisk Foundation Center for Protein Research,

University of Copenhagen, Blegdamsvej 3B, 2200 Copenhagen N, Denmark,

[4]Institute for Theoretical Chemistry, University of Vienna, Währingerstraße 17, 1090 Wien, Austria

September 7, 2018

This manual is distributed along with version 2.1rc1.

Contents

*To whom correspondence should be addressed. Tel: +45 353 33578; Fax: +45 353 34704; Email: gorodkin@rth.dk

1 Introduction

RIsearch2 is an RNA–RNA interaction prediction tool which enables quick localisation of potential RNA–RNA interaction sites between given query and target sequences. With its seed-and-extend strategy it promises reasonable running times even for very large scale predictions. It radically differs from its ancestral version RIsearch, particularly by introducing a new suffix array-based seed search method. RIsearch2 applies a two-stage strategy, it uses a suffix array data structure in the first step to locate seeds, stretches of perfect (optionally near-perfect) complementarity, and then extends these seed matches on both ends with a Smith–Waterman-like dynamic programming algorithm, using a simplified version of the Turner nearest-neighbour energy model, introduced by RIsearch.

Rules for the seed matching step of RIsearch2 are defined by the user to a great extent. Attributes of the seeds, such as minimum length, level of complementarity and query-specific positional constraints, are set by the user when running RIsearch2, and they have considerable impact on the running time of the algorithm. There exist only two non-flexible hard-wired rules in the seed localisation step and they should be taken into account by the user when selecting the seed attributes for the run. (1) $G–U$ wobble pairs are valid matches together with canonical Watson–Crick pairs ($G–C$ and $A–U$) and therefore do not break perfect complementarity; and (2) seeds are required to be maximal before they are passed into the seed extension step. A seed reaches maximality if it cannot be extended on either end with a valid base pair due to a mismatch or positional constraint. Consequently, all interactions predicted by RIsearch2 are extensions of maximal seeds.

The dynamic programming step of RIsearch2 can also be tuned by the user. In this step of the algorithm, dynamic programming tables are filled for flanking sequences of located seeds, and backtracking on these tables is used to extend seeds on both ends. Size limits for the dynamic programming tables can be set by the user and these limits determine the length of flanking sequences to consider when extending located seeds. Higher limits increase running time considerably and it is not recommended to set these limits too high. Besides, big bulges and internal loops increase the free energy of the predicted interaction, they destabilise the interaction, and intra-molecular base pairings that might have a stabilising contribution in these loops are not considered by our model.

In the next sections, we explain how to use RIsearch2 in detail, describe all user-defined parameters and give details about the algorithm.

2 How to use RIsearch2?

Execution of RIsearch2 requires the index structure (suffix array) of the target sequence(s) to be already pregenerated as it is the case with many other alignment tools. This is not only mandatory but also very practical for many use case scenarios since it would be a performance loss to regenerate the index structure of large target sequences (*e.g.*, a whole genome) in different runs with different query sequences or parameter settings. Here, we first describe how to generate the index structure with the RIsearch2 executable, whether for single or multiple target sequences, and then describe how to execute the run with all parameter settings.

2.1 Creating the index structure for target sequence(s)

Target sequence(s) are accepted into the program only in FASTA format (or gzip-compressed FASTA files). These sequences are always in 5' to 3' format. The constructed index will include the reverse complement sequences as well. Here is a command line example to show how to create the index file for target sequence(s) in the file *target.fa*.

```
$ risearch2.x -c target.fa -o target.suf
```

The output file path to save the generated index can be replaced with any other path, simply by specifying it after `-o`. The input can also be read from STDIN (by passing `-c -`), which allows for filtration or concatenation from multiple files on the fly. As for example:

```
$ cat target*.fa.gz | risearch2.x -c - -o target.suf
```

2.2 Executing `RIsearch2` for interaction predictions

Passing the query sequence (`-q FILE`) and understanding `RIsearch2` output

`RIsearch2` accepts query sequence(s) in (gzipped) FASTA format (5'-to-3') and the path is passed to the program with `-q FILE` option. `RIsearch2` generates a gzipped result file for each sequence in the input file (prefixed with *risearch_seqID*). The sequence ID that is used as part of the file name is defined as the first word after '>' (stopping at whitespace). Hence, one should note that if there exist duplicate sequence IDs in the input file, results will be overwritten. Additionally, if there already exists `RIsearch2` result files in the current working directory where `RIsearch2` is executed, this will also cause overwriting result files that were generated in the former run for query sequence IDs shared between the old and new run.

Passing the index file as target (`-i FILE`)

As mentioned in the previous section, the index structure of the target sequence(s) must be pregenerated, and this index structure is passed to the program with `-i` option. Here is a command line example using this to run `RIsearch2` in the simplest way with its default settings.

```
$ risearch2.x -q query.fa -i target.suf
```

Minimum length for seeds (`-s <int>`)

We define seeds as maximal stretches of consecutive base pairs and whatever sequence complementarity is decided for seeds, either perfect or near-perfect. All seeds are composed of two sequences with same length, one from query and other from target (or its reverse complement). As discussed above, a seed is maximal only if it cannot be extended on either end due to positional constraints or invalid base pairs (all but canonical or wobble base pair). In `RIsearch2`, the minimum length $|s|$ of a seed is set by the user with `-s <int>` option; seeds shorter than this are excluded and not passed into the dynamic programming step. For instance, `-s 7` requires seeds to have a minimum of 7 consecutive base pairs without any positional constraints, which will be described later. Under default settings it is set to `-s 6`.

Energy threshold for predicted interactions (`-e <float>`)

`RIsearch2` predicts interactions only around seed regions located by the first step of the algorithm. These interactions are formed by extending the seed on both ends with a dynamic programming approach that approximates the nearest-neighbour energy model. For each located seed, the interaction with minimum free energy is found. In some cases, this will be solely the seed itself. However, all interactions are filtered by applying an energy threshold before being reported. This threshold is set by `-e <float>` option and it is -20 by default, which means predictions with a free energy change higher than -20 kcal/mol are not reported. For miRNA-like interactions, we recommend to use a less restrictive threshold up to -10 kcal/mol. Here is an example command for running `RIsearch2` with such energy threshold together with minimum seed length set to seven.

```
  $ risearch2.x -q query.fa -i target.suf -s 7 -e -10
```

Size limit for dynamic programming tables (`-l <int>`)

This is the most important setting for the seed extension step. It determines the length of the sequences, flanking on both ends of the seeds, to be considered for extension. Besides, together with the seed criterion it has the biggest impact on running time of the algorithm. By default, it is set to 20. Setting it to 0 makes `RIsearch2` report the raw seeds (given they suffice the energy threshold). It is recommended to use values between 10 and 30 for realistic interaction predictions, depending on the type of study. However, results obtained with a smaller size can always be postprocessed to create longer interaction predictions. The size is always limited by sequence boundaries, so setting it to a very high value corresponds to allowing interactions to span the entire query sequence. You can see the use examples in the following sections.

Reporting predicted interactions in different output formats (`-p, -p2, -p3`)

As mentioned before, predicted interactions between query and target sequences are reported in query-specific gzipped result files. Predicted interactions can be reported in two different formats other than default, controlled by the user passing the argument `-p` or `-p2`. In default settings, interactions are reported in tsv format. You can see an example run below.

```
$ risearch2.x -c target.fa -o target.suf
$ risearch2.x -q query.fa -i target.suf -s 7 -e -13 -l 5
$ zcat risearch_query1.out.gz
query1  9       21      ENST00000436685 451     463     +       -20.19
$ zcat risearch_query2.out.gz
query2  9       22      ENST00000534717 281     294     -       -14.54
query2  9       22      ENST00000436685 156     169     -       -14.54
```

In this tab-separated table, columns respectively represent query sequence ID, interaction start position on query, interaction end position on query, target sequence ID, interaction start position on target, interaction end position on target, interaction strand, free energy of the interaction (in kcal/mol). When the strand is negative, it means the interaction is actually predicted between the query and reverse complementary target sequence, start and end positions however always refer to the positions on the target sequence as provided, *i.e.*, the forward strand.

Passing either `-p` or `-p2` will also return the interaction structure, which requires backtracking through the dynamic programming matrices.

When `-p` is passed, the binding site is visualised additionally, and the output takes four lines per interaction: (1) query in 5'-to-3' direction, (2) base pairs (a colon for G-U, a bar for C-G and A-U pairs), (3) target sequence in 3'-to-5' direction, (4) summary of the prediction (format as above). Here is how result files look with this long output format for the same interactions as reported above.

```
$ risearch2.x -q query.fa -i target.suf -s 7 -e -13 -l 5 -p
$ zcat risearch_query1.out.gz
GGUUGAGAGGGCG
||  |||||||||
ccuucucucccgc
query1  9       21      ENST00000436685 451     463     +       -20.19
$ zcat risearch_query2.out.gz
UGAUGCCUUUCUUC
||  ||||:|||:|
acgccggagagagg
query2  9       22      ENST00000534717 281     294     -       -14.54
UGAUGCCUUUCUUC
||  ||||:|||:|
acgccggagagagg
query2  9       22      ENST00000436685 156     169     -       -14.54
```

For a format that is easier to handle (smaller file size and straightforward parsing), but still provide information on the binding site itself, -p2 can be passed to the program. An extra column with the condensed interaction structure is then added to the default output table. It is essentially a transcript of the second line of the long format and additional information on gaps using letter codes as follows; P: canonical base pair, W: G-U wobble pair, U: unpaired, Q: bulge in query (a nucleotide in the query across a gap in the target), and T: bulge in target. Together with the input sequences, this information is enough to recreate the full interaction as given in the long format (-p). Example output with -p2 for the same interactions as before:

```
$ risearch2.x -c target.fa -o target.suf
$ risearch2.x -q query.fa -i target.suf -s 7 -e -13 -l 5 -p2
$ zcat risearch_query1.out.gz
query1  9       21      ENST00000436685 451     463     + -20.19 PPUUPPPPPPPPP
$ zcat risearch_query2.out.gz
query2  9       22      ENST00000534717 281     294     - -14.54 PPUUPPPPWPPPWP
query2  9       22      ENST00000436685 156     169     - -14.54 PPUUPPPPWPPPWP
```

A more detailed -p3 format additionally includes the sequence of the target interaction site together with the sequences of its upstream and downstream region. When -p3 is selected, the program outputs the predictions in -p2 format with three additional columns: the target binding site (3'-to-5' direction), 5' end upstream (3'-to-5' direction), and 3' end downstream (5'-to-3' direction) sequences of the target interaction site. This is the required format for post-processing of CRISPR Cas9 off-target predictions with CRISPR-OFF webserver[1].

```
$ risearch2.x -c target.fa -o target.suf
$ risearch2.x -q query.fa -i target.suf -s 7 -e -13 -l 5 -p3
$ zcat risearch_query1.out.gz
query1  9       21      ENST00000436685 451     463     + -20.19 PPUUPPPPPPPPP
ccuucucucccgc   ccgagaccucgacucuacuu    ugcagccugggaacuucagc
$ zcat risearch_query2.out.gz
query2  9       22      ENST00000534717 281     294     - -14.54 PPUUPPPPWPPPWP
acgccggagagagg  augggccugugacuacagua    gucgaucauagucuuccugc
query2  9       22      ENST00000436685 156     169     - -14.54 PPUUPPPPWPPPWP
acgccggagagagg  augggccugugacuacagua    gucgaucauagucuuccugc
```

[1]https://rth.dk/resources/crispr/crisproff/

Per-nucleotide extension penalty (`-d <int>`)

The per-nucleotide penalty (given in dacal/mol) mimics the energy required to free a nucleotide from any intramolecular binding. Without this, predictions tend to be overly long and interspersed with loops. A value of 30, corresponding to 0.3 kcal/mol, produces shorter and more stable interactions. However, it is not required for short queries such as miRNAs. If this parameter is specified, the penalty is directly integrated into the scoring matrix for that run. Before reporting the energy in the results, it is corrected for the penalty.

Changing energy matrix (`-z mat`)

By default, the energy scheme approximating the Turner 2004 parameters[2] is used (`-z t04`), alternatively the scoring scheme based on the previous Turner 1999 model[3] can be specified (`-z t99`). Other scoring schemes might be added in the future.

Query-specific positional constraints for seeds (`-s n:m/l`)

It is also possible to create query-specific positional constraints for seeds with the `-s` option. For example, setting `-s 5:12/4` requires a seed of minimum length 4 to be located between bases 5 and 12 (including) of the query. Negative positions correspond to indexing from the 3' end, *i.e.*, `-s -10:-1/5` requires a seed of minimum length 5 within the last ten bases of the query sequence. Note that, all seeds are maximised only within the interval $[n, m]$ before being passed on to the second stage. Omitting l enforces all bases in the specified range to be the seed, *i.e.*, setting `-s 2:7` is equal to `-s 2:7/6` (widely recognised as the seed region in microRNAs). In this case, seeds are not further maximised.

Disabling G–U wobble pair within seeds (`--noGUseed`)

When finding the seeds of interactions, it is possible to disable the G–U wobble pairs within the seed regions. This is done by passing `--noGUseed` option. However, it should be noted that the simplified energy model and seed extension is not affected by this option. G–U wobble pairs are still considered as a match when extending the seeds and computing the free energy of the interaction with `t99` and `t04` simplified energy models.

2.3 Experimental options

Level of complementarity for seeds (introducing symmetric mismatches on seeds) (`-m c:p`)

NOTE: Implementation/behavior of parameter `-m` changed in comparison to version 2.0.
Under default settings and in many use scenarios, seed sequences are required to be perfect complementary (wobble pair allowed or not, depending on the `--noGUseed` option), however, the user can additionally allow near-perfect complementary seeds with `-m c:p` option. These additional seeds are defined as "seeds with mismatches" since they are required to contain a limited number of invalid base pairs, that is set by the given c parameter. By definition, "seeds with mismatches" have complementary stretches around the introduced mismatches and a seed with mismatch can have up to $(c + 1)$ of them.

[2]Mathews DH, Disney MD, Childs JL, Schroeder SJ, Zuker M, and Turner DH. Incorporating chemical modification constraints into a dynamic programming algorithm for prediction of RNA secondary structure. *Proc. Natl. Acad. Sci. U.S.A.*, 101(19):7287–7292, 2004.

[3]Mathews DH, Sabina J, Zuker M, and Turner DH. Expanded sequence dependence of thermodynamic parameters improves prediction of RNA secondary structure. *Journal of Molecular Biology*, 288(5):911–940, 1999.

These maximal complementary stretches must always be shorter than l, the minimum seed length. Note that, this definition for additional near-perfect complementary seeds avoids overlaps between perfect and near-perfect complementary seeds. In other words, no seed with mismatch will ever include a valid perfect complementary seed as part of itself. The other parameter p, given together with the parameter c, can limit the positions of the mismatches by forcing maximal complementary stretches at the beginning and end of the seed. The value assigned to p sets the minimum length of these two maximal complementary stretches (first and last) where their maximum length is always l, the seed length. When number of allowed mismatches in the seed, parameter c, is set to a positive integer and the parameter p is set to 0, the program naturally returns some overlapping seeds, therefore, it is always advised to assign a positive integer to the parameter p when using this option.

Having seeds with mismatches enables to reanalyse and join overlooked seeds, that are shorter than the seed length threshold, to increase the sensitivity of the interaction predictions. On the other hand, this might result in generating many low quality seeds and increase the running time. On default settings, mismatches are not allowed at all in the seed. This option is only used for experimental purposes, therefore there are no recommended values yet.

Energy per length threshold for seeds (`-x <float>`)

This is also an experimental option in `RIsearch2`. When this parameter is set, `RIsearch2` computes the free energy of the seeds and filter out those with energy per length values higher than the given threshold value (kcal/mol). This option enables ignoring low quality seeds where quality metric is defined by the user and this can increase the specificity of interaction predictions, and reduce the runtime as well. Runtime decrease is due to skipping the seed extension step for filtered out seeds. When `-x <float>` is set, we first compute the free energy of all seeds together with their subseeds equal or longer than the user-defined seed length l (by using the simplified energy model of `RIsearch`). Then, best subseed with the lowest energy per length score, which might be the whole seed itself, is passed to the next stage as the original seed only if its energy per length value is lower than the given threshold. Otherwise, we ignore the seed (and all its subseeds) for good. Once again, this is an experimental option and it is not activated under default settings, and there are no recommended values yet.

3 Algorithm Details

3.1 Finding seeds in query and target with suffix arrays

Using suffix arrays as an index structure is at the core of the our algorithm in order to easily locate the seed regions of putative RNA–RNA interactions. A suffix array is defined as the lexicographically sorted list of all suffixes in a string. We extend this definition by annotating the list indices with their originating sequence thereby allowing concatenation of multiple sequences into a single suffix array.

Index construction

The FASTA formatted target (subject) sequences (could be a whole genome of interest) are converted into a suffix array using the libdivsufsort 2.0 library[4]. We do not use a generalised suffix array where each sequence is padded with unique terminator symbols, but instead build a regular suffix array of the concatenated sequences. Each sequence is stored in the original orientation back-to-back with its own

[4]http://libdivsufsort.googlecode.com/ Copyright (c) 2003-2008 Yuta Mori All Rights Reserved.

reverse complement $(s_1, revcomp(s_1), s_2, revcomp(s_2), ..., s_k, revcomp(s_k))$ and each entry in the suffix array is annotated with a serial number identifying the sequence from which the suffix originated. The index structure is constructed only once (in $\mathcal{O}(n \log n)$ time) and stored in a binary format on disk to be reused for a number of runs, with the following format specification:

- N : number of entries in suffix array (all nucleotides)
- K : number of sequences
- $L[1..K]$: length of sequences
- Each entry of next K entries contains length of name (uint16_t) followed by the sequence name (or ID) as null-terminated string
- $SA[1..N]$: the suffix array itself

Here, N, K, L[...] and SA[...] are 64 bits and bit packing in the suffix array is as follows:
| 63-38 (index) | 37-34 (nucleotide) | 33-0 (suffix array) |.

For each element of the suffix array, the 26 left-most bits hold the index, *i.e.*, serial number of originating sequence of a suffix. The 34 right-most bits hold the starting position of this suffix in the concatenated sequence. These limit the number of sequences and total nucleotides that can be packed into one index. The 4 bits in the middle are used to store the original nucleotides sequentially and independent of the suffix array. The way they are encoded leaves room for extension of the energy model towards modified bases, such as LNA. The last three bits are used to denote the type (ACGUN), while the first bit could be used to indicate a modified base (upper/lower case in the input FASTA). The initial step of locating seeds is independent of such modifications and thus only looks for the last three bits, while the thermodynamics and so the look-up in the scoring matrix depends on the modification and hence all four bits. As there currently is no energy model for modified bases, all nucleotides are converted to lower case.

Query Preparation

The query sequences are processed sequentially, in parallel if multiple threads are used (enabled by the OpenMP library[5]). Each query is then converted into a partial suffix array, that only allows finding the seeds that can satisfy the seed length and position criteria. Valid start positions are calculated based on the given query and seed specification (command line option). All positions outside this range are excluded from the matching stage.

Parallel Suffix Array Matching

The matching is performed by traversing the two suffix arrays in parallel. In each step, the intervals for each nucleotide are determined using a binary search and base pairs, *i.e.*, Watson–Crick or wobble, are recursively explored in depth-first order. With option `-m` also mismatches are considered. For memory-constrained systems, the FASTA file backing the index can be split into several smaller FASTA files by cutting at sequence boundaries. The merged output of querying the individual indexes will be identical to using one large index but some speed is sacrificed due to the fact that less sequence similarity can be exploited when constructing the suffix array and locating the seeds.

[5]http://www.openmp.org Copyright (c) 1997-2015 OpenMP Architecture Review Board.

3.2 Extending Seeds

For all identified seeds, the matching region (up to a user-defined distance from the seed match) is extended and the binding energy is computed by using the simplified energy model of `RIsearch`. This is done by filling the usual dynamic programming matrices for a limited region up- and downstream of the seed region, with the special requirement of having the first (last) base pair of the seed set. Parameter `-l L` determines the limits of this dynamic programming window by setting the maximum length of the up(down)-stream region to L. After the extension step, for all hits passing the energy threshold, the actual interaction *alignment* is found via backtracking and results are written in one of three different formats.