Chapter 10

Hybrid MPI/OpenMP Implementation of RIblast for Transcriptome-Scale lncRNA–RNA Interaction Prediction

Iñaki Amatria-Barral, Jorge González-Domínguez, and Juan Touriño

Abstract

Because the identification of RNA interaction partners for each long noncoding RNA (lncRNA) has proven to be a powerful approach for inferring lncRNA functions, numerous tools have been developed to predict lncRNA–RNA interactions. However, their application at the transcriptome scale has been hindered by high computational costs. In this chapter, we introduce pRIblast, a hybrid MPI/OpenMP implementation of the RIblast algorithm designed for large-scale lncRNA–RNA interaction analyses. pRIblast leverages cluster environments and supercomputing facilities to reduce the computational costs of these studies dramatically. Moreover, pRIblast has been thoroughly optimized, and it can take on huge datasets that were impossible to analyze with the former RIblast algorithm.

Key words lncRNA, lncRNA–RNA interaction, Transcriptome-scale analysis, High-performance computing, MPI, OpenMP

1 Introduction

Long noncoding RNAs (lncRNAs) are transcripts with no less than 200 nucleotides in length and no protein-coding capacity. Even though lncRNAs were first regarded as transcriptional noise, mounting evidence now suggests that these sequences do have functions and that, indeed, lncRNAs play a key role in many biological processes [1]. For instance, lncRNA regulates gene expression [2], immune response [3], and cell cycle [4].

Although more than 58000 lncRNA genes are encoded by the human genome [5], the vast majority of them are still poorly characterized [6]. And what is worse, the dysfunction of many lncRNAs has been associated with severe diseases, such as cardiac pathologies [7], several types of cancer [8, 9], or Parkinson's [10]. Thus, elucidating lncRNA functions is a hot research topic today.

Michiaki Hamada et al. (eds.), *RNA-RNA and RNA-DNA Interactions*, Methods in Molecular Biology, vol. 2949,
https://doi.org/10.1007/978-1-0716-4670-0_10,

lncRNAs lack sequence and structure conservation [11]. Therefore, sequence similarity search and RNA secondary structure search, while they have achieved success in characterizing the function of protein-coding genes and small RNAs [12, 13], are unsuitable for inferring the functions of lncRNAs. In contrast, because lncRNAs work by being assembled with other proteins or RNAs [14], the identification of interaction partners for each lncRNA has proved to be a successful approach for characterizing the functions of lncRNAs [15, 16].

Several sequencing-based technologies have been developed for the experimental discovery of RNA–RNA interactions. These include RIA-seq [17] and RAP-RNA [18], which are able to exhaustively detect interaction targets of specific lncRNAs. Repeating these experiments across many lncRNAs, however, is very labor intensive. On the other hand, there exist methods, such as PARIS [19] and SPLASH [20], that can comprehensively identify RNA–RNA interactions in vivo. Nonetheless, these techniques have found the most success identifying interactions of ribosomal RNAs and small RNAs. Furthermore, lncRNAs show tissue-specific expression patterns [5, 11], meaning that these experiments should be conducted on various tissues to provide extensive results, which, again, requires hard work.

Since the detection of large-scale lncRNA–RNA interactions through experiments is impracticable, computational prediction has emerged as an indispensable technique. Several tools and procedures have been developed with the sole purpose of predicting RNA–RNA interactions [21, 22], aiming to further progress in the characterization of lncRNA transcripts. For instance, Szcześniak and Makałowska predicted lncRNA–RNA interactions across the human transcriptome using sequence similarity search without consideration of the RNA secondary structure [23]. Nevertheless, benchmarking results have shown that tools that omit the RNA secondary structure have lower discriminatory performance [21, 24]. In other words, the most accurate RNA–RNA interaction prediction results are obtained by tools that consider the RNA secondary structure. Notable examples of such tools include IntaRNA [25], RNAplex [26], RactIP [27], ASSA [28], and RRP [29]. Despite their accuracy, these algorithms require extensive computational resources to find promising lncRNA–RNA interacting pairs due to the consideration of the RNA secondary structure, which significantly impacts execution time [30].

To address the execution time challenge, Fukunaga and Hamada developed RIblast [30], a C++ algorithm that exploits the seed-and-extension heuristic [12]. This approach accelerates the prediction of lncRNA–RNA interacting pairs without compromising prediction accuracy, rivaling the performance of the most accurate tools in the state of the art [24, 30]. However, even with their thorough optimization efforts, RIblast still falls

short when it comes to computing lncRNA–RNA interactions at the transcriptome scale. Indeed, Fukunaga and Hamada have already emphasized the need for further acceleration to effectively identify lncRNA–RNA interactions at such a large scale [30]. Therefore, in this chapter, we present pRIblast [31, 32], an enhanced version of RIblast developed to overcome the current execution time limitations. By optimizing the algorithm's runtime and leveraging parallel computing techniques, pRIblast aims to enable the identification of lncRNA–RNA interactions on a broader scale, specifically at the transcriptome level. We demonstrate the improved efficiency and performance of pRIblast and provide practical guidelines for its execution on workstations and computing clusters.

1.1 RIblast

1.1.1 RNA–RNA Interaction Prediction Algorithm

RIblast enumerates potentially interacting segments between a query RNA, q, and a target RNA, t. RIblast uses two energies to determine if two segments, q_s and t_s, interact. These energies are:

- The accessible energy, which indicates segments that are likely to form intermolecular base pairs with other segments. When the accessible energy is low, a segment is more likely to form intermolecular base pairs as it avoids intramolecular base pairing.
- The hybridization energy, which is the free energy released by the intermolecular base pairs between two segments. The hybridization energy is computed as the sum of the stacking energies and loop energies in the intermolecular base pair structure. Intramolecular base pairs are ignored by the hybridization energy algorithm of RIblast.

The interaction energy between q_s and t_s is then computed by adding together the accessible energy of q_s, the accessible energy of t_s, and the hybridization energy between q_s and t_s. RIblast will output segments with particularly low interaction energy as a detected RNA–RNA interaction.

For all-to-all predictions, the calculation time of accessible energies scales linearly with the number of sequences. However, the calculation time of the hybridization energies scales quadratically with the number of sequences. This poses a challenge for comprehensive prediction of lncRNA–RNA interactions. And this is where pRIblast focuses its performance optimization efforts, leveraging parallel computing technologies to reduce the execution time and enable the identification of lncRNA–RNA interactions at the transcriptome scale.

1.1.2 RIblast Overview

RIblast functions in two steps: a database construction step and an RNA interaction search step.

In the database construction step, RIblast precalculates the accessible energies and the results of searching for short substrings

of a target RNA dataset. This is done by first computing approximate accessible energies of the RNA sequences and then encoding the sequences, building a suffix array, and constructing a lookup table of search results. All these data (i.e., the accessible energies, the encoded sequences, the suffix array, and the lookup table) are stored in a database that will later be used in the RNA interaction search step. Building a database of RNA sequences allows RIblast to:

- Accelerate the subsequent RNA interaction search step. The accessibility and lookup table computations do not need to be repeated for every query sequence, resulting in substantial computational time savings.
- Enable the efficient reuse of data. Once a database is built, the RNA interaction search can be performed with different query sequences on the same target dataset without having to recompute accessibility and lookup table information. This proves particularly useful when analyzing large datasets, such as the complete human RNA transcriptome.

In the RNA interaction search step, RIblast finds interactions between the target RNA sequences in a database and a given set of query sequences. Here, RIblast first computes approximate accessible energies and constructs a suffix array for every sequence in the query RNA dataset. Second, RIblast finds seed regions with a hybridization energy that does not exceed an energy threshold, T_1. This is done based on the suffix array of the queries and the target database. Third, RIblast computes the interaction energy of the seeds and removes those with energies that exceed 0 kcal/mol. Fourth, seeds are extended to find interactions with no gaps. Fifth, overlapping interactions are removed, as well as interactions with energies exceeding a threshold, T_2. And sixth, interactions are extended from seed regions with a gap and stored in a comma-separated output text file.

For further information on the functioning and design of the RIblast algorithm, we refer the interested readers to the original paper of RIblast [30], especially those seeking to optimize the search parameters of the algorithm for their specific use case.

1.2 pRIblast

pRIblast is a hybrid MPI/OpenMP implementation of the RIblast algorithm specifically designed to enable large-scale lncRNA–RNA interaction analyses. While pRIblast shares the same prediction algorithm as RIblast, it leverages the power of high-performance computing to address the increasing demand for comprehensive lncRNA–RNA interaction analysis.

pRIblast also functions in two steps: a database construction step and an RNA interaction search step. These steps have undergone extensive optimization efforts to maximize performance,

including static analysis improvements, data storage and memory management optimizations, and the utilization of advanced parallelization techniques. These optimizations collectively enhance the efficiency and scalability of pRIblast, enabling it to handle huge lncRNA–RNA datasets.

By incorporating pRIblast into their research workflow, scientists can benefit from its ability to dramatically reduce the computational costs associated with large-scale lncRNA–RNA interaction analysis. This empowers researchers to explore the functional implications of lncRNAs at the transcriptome scale and uncover novel insights into their roles in cellular processes.

1.2.1 MPI, OpenMP, and pRIblast's Target Architecture

pRIblast is designed to target distributed-memory architectures, such as computing clusters or supercomputers. These architectures consist of multiple computing nodes, with each node comprising a complete individual computer, connected through a high-performance interconnection network.

In this distributed-memory scenario, pRIblast divides the workload among multiple MPI (Message-Passing Interface) processes that run on the different nodes of the cluster. MPI [33] is the de facto standard for developing parallel programs that run on distributed-memory systems. It provides a portable, highly optimized, and flexible message-passing protocol, facilitating efficient communication and coordination between processes.

Once the workload is distributed among the nodes of the cluster using MPI, each process spawns OpenMP threads to exploit all the CPU cores within each node and accelerate the computation, ideally by a factor of $N \times C$ (where N is the number of nodes and C is the number of cores per node). OpenMP [34] is a collection of compiler directives and functions that offers a scalable, up-to-date, and portable application programming interface for shared-memory systems programming.

For example, in a computing cluster consisting of two nodes, each with four CPU cores, MPI would be used to coordinate two processes running on the two nodes. Each process would then spawn four OpenMP threads to fully exploit the eight cores within this two-node setup, theoretically accelerating a given computation by a factor of 8. In the context of RNA–RNA interaction prediction, pRIblast uses MPI to distribute RNA sequences among the nodes of the cluster, and OpenMP is employed to process these sequences in parallel across all the CPU cores of the cluster.

This paradigm is flexible and allows pRIblast to not only exploit distributed-memory architectures but also standard desktop computers and workstations. Therefore, pRIblast can accelerate the prediction of lncRNA–RNA interacting pairs even on a single machine with multiple CPU cores, providing a more modest configuration for users who do not have access to large-scale computing resources. This demonstrates the versatility of pRIblast, making

it suitable for a wide range of computing environments and accommodating users with varying resources and requirements.

1.2.2 Database Paging Mechanism

pRIblast introduces a highly effective optimization known as the database paging mechanism to overcome the memory limitations of the former RIblast algorithm. In RIblast, results are written to the output file only after comparing all target sequences against a specific RNA query sequence. This approach can lead to memory overflow if the target database is large or if a query segment produces a significant number of prediction candidates.

To address this limitation, pRIblast divides the target database into smaller, independent subsets called "pages." These pages are designed to fit within the available memory space of a single compute node and prevent memory overflow. Instead of comparing query sequences against the entire target database in a single round, pRIblast makes predictions against subsets of the database represented by these pages.

Implementing the database paging mechanism required slight modifications to the database construction step described before. Specifically, the database construction step was adjusted to read chunks of a fixed size that correspond to the pages used in the RNA interaction search step. Nevertheless, databases constructed without the modified version of the construction step remain fully compatible with the new pRIblast workflow.

The database paging mechanism in pRIblast significantly improves the efficiency, scalability, and memory management of the algorithm. By dividing the target database into manageable subsets, pRIblast minimizes memory pressure and prevents overflow, enabling researchers to analyze large-scale lncRNA–RNA interactions without compromising computational resources.

1.2.3 pRIblast in Figures

To demonstrate the enhanced performance of pRIblast compared to RIblast, we present a comparison of execution times for predicting lncRNA–RNA interactions on five noncoding RNA transcriptomes. These experiments were conducted using a computing cluster comprising 16 nodes, each equipped with 16 CPU cores. Table 1 provides an overview of the execution times obtained from these experiments, showcasing the efficiency gains achieved by pRIblast.

The table shows that pRIblast consistently outperforms RIblast across all five transcriptomes. Leveraging an optimized algorithm and parallel computing techniques, pRIblast enables significantly faster database construction, completing the task up to 318 times faster than RIblast. Similarly, pRIblast accelerates the prediction of RNA–RNA interactions by up to 135 times compared to RIblast. These advancements capitalize on the full computational power of pRIblast, facilitating rapid and comprehensive analysis of lncRNA–RNA interactions.

Table 1
Execution time comparison between RIblast and pRIblast on five non-coding RNA transcriptomes and a computing cluster with 16 nodes, each equipped with 16 CPU cores

	RIblast		pRIblast	
Transcriptome	DB const.	RNA–RNA search	DB const.	RNA–RNA search
Lepidothrix C.	8 m 0 s	2 h 6 m 42 s	6 s	3 m 37 s
Ursus A.	13 m 56 s	4 h 38 m 34 s	5 s	1 m 59 s
Drosophila M.	41 m 36 s	14 h 43 m 49 s	30 s	9 m 55 s
Anser B.	2 h 8 m 13 s	22d 8 h 13 m 46 s*	24 s	3 h 56 m 6 s
Anas P. P.	3 h 14 m 41 s	101 d 10 h 52 m 55 s*	36 s	20 h 7 m 45 s

* Estimate of the runtime, RIblast cannot compute these datasets because it runs out of memory

Furthermore, the database paging mechanism employed in pRIblast proves to be a crucial optimization, as highlighted in Table 1. This mechanism enables pRIblast to successfully complete two transcriptome-scale analyses that would have been infeasible with RIblast due to memory limitations.

For further information on the technical details behind pRIblast, we refer the interested readers to the original papers of pRIblast [31, 32].

2 Materials

2.1 Hardware Requirements

Choose one of the following:

- An x86/x86_64, SSE2-equipped processor, i.e., virtually any modern x86 processor (year 2000 onward)
- A cluster of x86/x86_64, SSE2-equipped processors, i.e., virtually any computing cluster

2.2 Software Requirements

1. A Linux-/Unix-like operating system, e.g., Ubuntu 20.04
2. Wget, e.g., GNU Wget v1.20.3
3. Gzip, e.g., GNU Gzip v1.10
4. Git, e.g., Git v2.25.1
5. Make, e.g., GNU Make v4.2.1
6. A C++17 compliant and OpenMP-enabled C++ compiler, e.g., GCC v9.4.0
7. An MPI-3 compliant MPI library, e.g., OpenMPI v4.0.3
8. Python 3, e.g., Python v3.8.10

For users executing pRIblast on an Ubuntu desktop/workstation, all these dependencies may be installed at once using the following command:

```
$ sudo apt update && \
  sudo apt install -y wget gzip git build-essential
openmpi-bin libopenmpi-dev python3
```

Users executing pRIblast on a computing cluster will likely have all these packages installed by default. However, they will need to load them explicitly using the appropriate module management tool in place at their system. In the computing cluster used to demonstrate the methods in this chapter, this software is Environment Modules[1] (as it is in most computing clusters worldwide), and we load the required software by executing the following command:

```
$ module load gcc/system openmpi/4.1.4 python/
3.9.9
```

Please note that different installations of Environment Modules may have different module names. The provided command is specific to our system. If you are unsure about how to load the required software, please consult your system administrator.

3 Methods

This section will show how to:

- Install pRIblast on your system.
- Download a noncoding RNA transcriptome and prepare it for analysis.
- Run pRIblast on a:
  - Desktop computer or workstation.
  - Computing cluster.

3.1 Installing pRIblast

This section provides step-by-step instructions for installing pRIblast on your system. It ensures that users have pRIblast installed and ready to use for the subsequent steps.

1. Navigate to your `HOME` directory and create an `opt` folder where you will store your local installation of pRIblast.

```
$ cd ~
$ mkdir -p opt
```

[1] https://lmod.readthedocs.io/en/latest/

2. Navigate to the newly created folder and clone pRIblast into your local machine.

```
$ cd opt
$ git clone https://github.com/UDC-GAC/pRIblast
```

3. Navigate to the root folder of pRIblast and compile the application (*see* **Note 1** below).

```
$ cd pRIblast
$ make -j$(getconf _NPROCESSORS_ONLN)
```

4. Update your `PATH` variable so that your system can find the pRIblast executable file automatically. We recommend changing the name of the executable file, too.

```
$ mv target/pRIblast.release target/pRIblast
$ echo "export PATH=\$PATH:$(pwd)/target" >> ~/.
bashrc
```

5. Restart your shell's session, and you will be able to invoke pRIblast from anywhere in your system.

```
$ cd ~
$ source .bashrc
$ pRIblast
```

3.2 Data Preparation

In this section, the focus is on preparing the data for analysis with pRIblast. It guides users through creating a data directory, downloading a specific noncoding RNA transcriptome, decompressing the downloaded file, and splitting the transcriptome into separate files for RNA and lncRNA sequences.

1. Navigate to your `HOME` directory and create a `data` folder where you will store the transcriptome executed in the following sections of the chapter.

```
$ cd ~
$ mkdir -p data/pRIblast/input
```

2. Navigate to the newly created folder and download the *Lepidothrix coronata* noncoding RNA transcriptome from the Ensembl genome browser 109 [35].

```
$ cd data/pRIblast/input
$ wget http://ftp.ensembl.org/pub/release-109/
fasta/lepidot\
      hrix_coronata/ncrna/Lepidothrix_coronata.
      Lepidothrix\
       _coronata-1.0.ncrna.fa.gz -O lepidothrix.
       fa.gz
```

3. Decompress the transcriptome.

```
$ gzip -d lepidothrix.fa.gz
```

4. Run the script provided alongside pRIblast to split `lepidothrix.fa` into two separate files: `lepidothrix_RNA.fa`, which will hold all the RNA sequences in the transcriptome and will be used later to construct a database, and `lepidothrix_lncRNA.fa`, which will hold only the lncRNA sequences in the transcriptome and will be used later to predict interactions against `lepidothrix_RNA.fa`.

```
$ python3 ~/opt/pRIblast/scripts/split_transcriptome.py lepidothrix.fa
```

3.3 Running pRIblast on a Desktop Computer or Workstation

This section provides instructions for running pRIblast on a desktop computer or workstation.

1. Navigate to your `HOME` directory and create an `output` folder inside the `data` folder where you will store the results of pRIblast.

```
$ cd ~
$ mkdir -p data/pRIblast/output
```

2. Create a temporary directory inside `/tmp` for pRIblast to store intermediate result files.

```
$ mkdir -p /tmp/pRIblast
```

3. Build a database of RNA sequences, `lepidothrix-db`, with the complete RNA transcriptome downloaded in the previous section.

```
$ pRIblast db -i data/pRIblast/input/lepidothrix_RNA.fa \
        -o data/pRIblast/output/lepidothrix-db \
        -p /tmp/pRIblast
```

4. Predict interactions between the lncRNA sequences in the *Lepidothrix coronata* transcriptome and the RNA sequences in the newly created database (*see* **Note 2** below). Predicted interactions will be stored in the file `lepidothrix-pred.txt`.

```
$ pRIblast ris -i data/pRIblast/input/lepidothrix_lncRNA.fa \
       -o data/pRIblast/output/lepidothrix-pred.txt \
       -d data/pRIblast/output/lepidothrix-db \
       -p /tmp/pRIblast
```

The previous commands will launch one pRIblast process and spawn as many threads as there are CPU cores available in your computer (*see* **Note 3** below). This demonstrates the flexibility

provided by pRIblast and how it seamlessly scales from standard desktop computers to supercomputers.

3.4 Running pRIblast on a Computing Cluster

This section provides instructions for running pRIblast on a computing cluster. But please note that the method described here is specifically designed for clusters managed with SLURM[2] (about 60% of the supercomputers in the TOP500 list) and Environment Modules. However, it can serve as a solid foundation for creating execution scripts in clusters managed with other clustering software.

1. Navigate to your `HOME` directory and create an `output` folder inside the `data` folder where you will store the results of pRIblast.

```
$ cd ~
$ mkdir -p data/pRIblast/output
```

2. Create a `scripts` directory where you will store the batch script used to submit pRIblast for execution on the cluster.

```
$ mkdir -p scripts/pRIblast
```

3. Navigate to the `scripts` folder and create the following batch script with the name `lepidothrix.sbatch`. This script will first load all the required software to run pRIblast. Then, it will create a database of RNA sequences, `lepidothrix-db`, using all the RNA sequences in the *Lepidothrix coronata* transcriptome. And finally, it will predict lncRNA–RNA interactions between the RNA sequences in the database and the lncRNA sequences in the transcriptome.

```
$ cd scripts/pRIblast
$ cat <<EOF > lepidothrix.sbatch
#!/bin/bash
#SBATCH --nodes=16
#SBATCH --ntasks-per-node=1
#SBATCH --cpus-per-task=16
#SBATCH --mem-per-cpu=4G
#SBATCH --time=00:20:00

module load gcc/system openmpi/4.1.4

export OMP_NUM_THREADS=\$SLURM_CPUS_PER_TASK
srun pRIblast db -i ~/data/pRIblast/input/lepido-
thrix_RNA.fa \
      -o ~/data/pRIblast/output/lepidothrix-db \
      -p \$TMPDIR
```

[2] https://slurm.schedmd.com/documentation.html

```
srun pRIblast ris -i ~/data/pRIblast/input/lepi-
dothrix_lncRNA.fa \
    -o ~/data/pRIblast/output/lepidothrix-pred.
txt \
    -d ~/data/pRIblast/output/lepidothrix-db \
    -p \$TMPDIR
EOF
```

4. Use `sbatch` to submit the script for execution on the cluster (*see* **Notes 2** and **4** below).

```
$ sbatch lepidothrix.sbatch
```

This last command initiates the execution of `lepidothrix.sbatch` on the cluster in batch mode (i.e., noninteractive, offline execution). The script launches pRIblast with 16 processes, each spawning 16 threads, distributed across 16 different nodes of the cluster, thereby using a total of 256 CPU cores. Additionally, the script allocates 4 GB of main memory per thread and sets a time limit of 20 minutes for the execution. To optimize performance and reduce input and output latencies, the `$TMPDIR` environment variable points to a local temporary directory on each node's local file system.

4 Notes

The following notes provide additional information and troubleshooting tips for potential issues that users may encounter during the installation and execution of pRIblast.

1. If the compilation fails and throws any of the following error messages:
   - `make: command not found`, or
   - `make: mpic++: command not found`,

   make sure you have installed all the required software listed in Subheading 2.2 before compiling pRIblast. Users compiling pRIblast in a cluster make sure you have loaded all the required software using the appropriate module management tool.
2. If the interaction prediction step fails with an error message of the form:
   - `segmentation fault (core dumped)`,
   - `fatal error: out of memory`, or
   - `error: detected oom-kill event(s)`,

   this means that pRIblast ran out of memory while finding RNA interactions. Rebuild the target database, but this time set a page size (i.e., append the option `-c INT`) to reduce the

maximum number of sequences that are processed simultaneously by the algorithm. We have found that a page size of 250 sequences reduces the memory consumption of the algorithm below 4 GB per thread. If you still encounter out-of-memory errors, reconstruct the database with a smaller page size (i.e., `-c 150`).

3. To control the number of threads used by pRIblast on a desktop computer, users may set the environment variable `OMP_NUM_THREADS` accordingly. For instance, to limit the number of threads spawned by pRIblast to 4, execute the application as follows:

```
$ OMP_NUM_THREADS=4 pRIblast ...
```

4. If the submission fails with an error message of the form `sbatch: error: batch job submission failed`, please check that the resources requested by the batch script do not violate any hard limit or accounting/quality of service policy of your system (e.g., job submission limits or time limits). For instance, it may happen that your cluster does not allow configurations where memory per CPU core exceeds a threshold of 2 GB. If you are unsure, please contact your system administrator for further assistance.

Acknowledgements

We sincerely thank Dr. Michiaki Hamada, Dr. Tsukasa Fukunaga, and Dr. Chao Zeng for inviting us to write this chapter for Methods in Molecular Biology. This work was supported by the Ministry of Science and Innovation of Spain (PID2019-104184RB-I00/AEI/10.13039/501100011033), by the Ministry of Education of Spain (FPU21/00491), and by Xunta de Galicia (Consolidation Program of Competitive Reference Groups, ref. ED431C 2021/30).

References

1. Zhu JJ, Fu HJ, Wu YG, Zheng XF (2013) Function of lncRNAs and approaches to lncRNA-protein interactions. Sci China Life Sci 56:876–885
2. Fatica A, Bozzoni I (2014) Long non-coding RNAs: new players in cell differentiation and development. Nat Rev Genet 15(1):7–21
3. Ouyang J, Hu J, Chen J-L (2016) lncRNAs regulate the innate immune response to viral infection. Wiley Interdiscip Rev RNA 7(1): 129–143
4. Kitagawa M, Kitagawa K, Kotake Y, Niida H, Ohhata T (2013) Cell cycle regulation by long non-coding RNAs. Cell Mol Life Sci 70:4785–4794
5. Iyer MK, Niknafs YS, Malik R, Singhal U, Sahu A, Hosono Y, Barrette TR, Prensner JR, Evans JR Zhao S et al (2015) The landscape of long noncoding RNAs in the human transcriptome. Nat Genet 47(3):199–208
6. de Hoon M, Shin JW, Carninci P (2015) Paradigm shifts in genomics through the FANTOM projects. Mammalian Genome 26(9–10):391–402

7. Hobuß L, Bär C, Thum T (2019) Long non-coding RNAs: at the heart of cardiac dysfunction? Front Physiol 10:30
8. Tornesello ML, Faraonio R, Buonaguro L, Annunziata C, Starita N, Cerasuolo A, Pezzuto F, Tornesello AL, Buonaguro FM (2020) The role of microRNAs, long non-coding RNAs, and circular RNAs in cervical cancer. Front Oncol 10:150
9. Dong X, He X, Guan A, Huang W, Jia H, Huang Y, Chen S, Zhang Z, Gao J, Wang H (2019) Long non-coding RNA Hotair promotes gastric cancer progression via miR-217-GPC5 axis. Life Sci 217:271–282
10. Elkouris M, Kouroupi G, Vourvoukelis A, Papagiannakis N, Kaltezioti V, Matsas R, Stefanis L, Xilouri M, Politis PK (2019) Long non-coding RNAs associated with neurodegeneration-linked genes are reduced in Parkinson's disease patients. Front Cell Neurosci 13:58
11. Cabili MN, Trapnell C, Goff L, Koziol M, Tazon-Vega B, Regev A, Rinn JL (2011) Integrative annotation of human large intergenic noncoding RNAs reveals global properties and specific subclasses. Genes Dev 25(18): 1915–1927
12. Altschul SF, Gish W, Miller W, Myers EW, Lipman DJ (1990) Basic local alignment search tool. J Mol Biol 215(3):403–410
13. Nawrocki EP, Eddy SR (2013) Infernal 1.1: 100-fold faster RNA homology searches. Bioinformatics 29(22):2933–2935
14. Hirose T, Mishima Y, Tomari Y (2014) Elements and machinery of non-coding RNAs: toward their taxonomy. EMBO Rep 15(5): 489–507
15. Abdelmohsen K, Panda AC, Kang M-J, Guo R, Kim J, Grammatikakis I, Yoon J-H, Dudekula DB, Noh JH, Yang X et al (2014) 7SL RNA represses p53 translation by competing with HuR. Nucleic Acids Res 42(15):10099–10111
16. Gong C, Maquat LE (2011) lncRNAs transactivate STAU1-mediated mRNA decay by duplexing with 3′ UTRs via Alu elements. Nature 470(7333):284–288
17. Kretz M, Siprashvili Z, Chu C, Webster DE, Zehnder A, Qu K, Lee CS, Flockhart RJ, Groff AF, Chow J et al (2013) Control of somatic tissue differentiation by the long non-coding RNA TINCR. Nature 493(7431):231–235
18. Engreitz JM, Sirokman K, McDonel P, Shishkin AA, Surka C, Russell P, Grossman SR, Chow AY, Guttman M, Lander ES (2014) RNA-RNA interactions enable specific targeting of noncoding RNAs to nascent Pre-mRNAs and chromatin sites. Cell 159(1): 188–199
19. Lu Z, Zhang QC, Lee B, Flynn RA, Smith MA, Robinson JT, Davidovich C, Gooding AR, Goodrich KJ, Mattick JS et al (2016) RNA duplex map in living cells reveals higher-order transcriptome structure. Cell 165(5): 1267–1279
20. Aw JGA, Shen Y, Wilm A, Sun M, Lim XN, Boon K-L, Tapsin S, Chan Y-S, Tan C-P, Sim AY et al (2016) In vivo mapping of eukaryotic RNA interactomes reveals principles of higher-order organization and regulation. Mol Cell 62(4):603–617
21. Lai D, Meyer IM (2016) A comprehensive comparison of general RNA–RNA interaction prediction methods. Nucleic Acids Res 44(7): e61
22. Umu SU, Gardner PP (2017) A comprehensive benchmark of RNA–RNA interaction prediction tools for all domains of life. Bioinformatics 33(7):988–996
23. Szcześniak MW, Makałowska I (2016) lncRNA–RNA interactions across the human transcriptome. PLoS One 11(3):e0150353
24. Antonov IV, Mazurov E, Borodovsky M, Medvedeva YA (2019) Prediction of lncRNAs and their interactions with nucleic acids: benchmarking bioinformatics tools. Briefings Bioinform 20(2):551–564
25. Busch A, Richter AS, Backofen R (2008) IntaRNA: efficient prediction of bacterial sRNA targets incorporating target site accessibility and seed regions. Bioinformatics 24(24): 2849–2856
26. Tafer H, Hofacker IL (2008) RNAplex: a fast tool for RNA–RNA interaction search. Bioinformatics 24(22):2657–2663
27. Kato Y, Sato K, Hamada M, Watanabe Y, Asai K, Akutsu T (2010) RactIP: fast and accurate prediction of RNA-RNA interaction using integer programming. Bioinformatics 26(18): i460–i466
28. Antonov I, Marakhonov A, Zamkova M, Medvedeva Y (2018) ASSA: fast identification of statistically significant interactions between long RNAs. J Bioinform Comput Biol 16(1): 1840001
29. Terai G, Iwakiri J, Kameda T, Hamada M, Asai K (2016) Comprehensive prediction of lncRNA–RNA interactions in human transcriptome. BMC Genom 17:153–164
30. Fukunaga T, Hamada M (2017) RIblast: an ultrafast RNA–RNA interaction prediction system based on a seed-and-extension approach. Bioinformatics 33(17):2666–2674

31. Amatria-Barral I, González-Domínguez J, Touriño J (2023) pRIblast: a highly efficient parallel application for comprehensive lncRNA–RNA interaction prediction. Fut Gener Comput Syst 138:270–279
32. Amatria-Barral I, González-Domínguez J, Touriño J (2023) Parallel construction of RNA databases for extensive lncRNA–RNA interaction prediction. In: Proceedings of the 38th ACM/SIGAPP symposium on applied computing, SAC '23 (Tallinn, Estonia), pp 555–558
33. The MPI Forum (2012) MPI: A Message-Passing Interface Standard (v3.0). https://www.mpi-forum.org/docs/mpi-3.0/mpi30-report.pdf (visited on 22/06/2023)
34. OpenMP Architecture Review Board (2015) OpenMP Application Programming Interface (v4.5). https://www.openmp.org/wp-content/uploads/openmp-4.5.pdf (visited on 22/06/2023)
35. Martin FJ, Amode MR, Aneja A, Austine-Orimoloye O, Azov AG, Barnes I, Becker A, Bennett R, Berry A, Bhai J et al (2023) Ensembl 2023. Nucleic Acids Res 51(D1): D933–D941