

Tool Independent Web Application for Bioinformatics Analysis

Andrea Di Nenno - Leonardo Tanzi - Alessandro Manfredi

October 2018

1 Abstract

This document presents the construction of a web based application for bioinformatics computations that addresses two main issues we found out being yet uncovered. Firstly the need of an architecture that is independent from the underlying tool, so that even by updating or changing entirely the tool, the amount of work needed to re-set the application is irrelevant. Secondly the need of such architecture to be highly scalable, given the high amount of computational effort required generally by bioinformatic tools. In this specific case, the tool the application is build upon is called isomiR-SEA, which is capable to extract miRnA and isomiR expression levels from high-throughput sequencing data. IsomiR-SEA's innovation consists of performing reads alignment on miRNAs databases by considering the miRNA:mRNA interaction pairing aspects, thus leading to reach and high accuracy in miRNAs expression levels extraction. In order to implement a relevant web application, we needed first to understand the requirements involved in this domain. For this purpose we have analyzed two already deployed frameworks: on the one hand the RNA-seq sequencing technique, a next-generation sequencing technique for transcriptome profiling [7], while on the other hand the genome-wide miRNA expression data, which is used to study miRNA dysregulation comprehensively [6]. Moreover, we looked at a couple of web applications, [8] and [2], that already implement a bioinformatic pipeline, in order to gain some insights and requirements the application should have. In the first section we outline some Bioinformatic's basic notions, in the second one we define the requirements of the application, while in the last one describe our implementation.

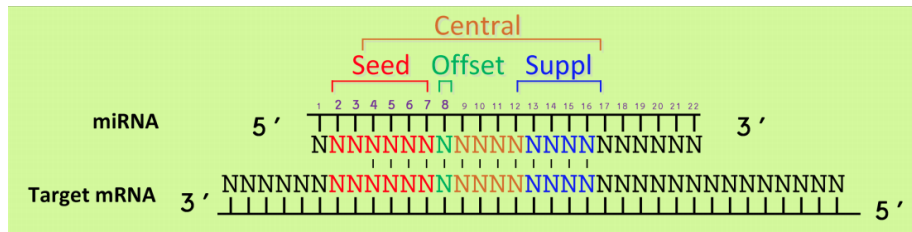


Figure 1: Interaction sites scheme.

2 Overview

This section gives an overview of the theory behind isomiR-SEA and an explanation of the algorithm used by the tool.

2.1 Basic Notions

MiRNAs are composed of 22 to 24 bases, they come from non-coding regions (a.k.a. regions from where you cannot produce mRNA), are double-stranded but during the process the hairpin is cut off and they become single-stranded. Their role is the silencing of the mRNA on the cytoplasm, a regulation function that acts in a post-transcriptional phase. We already know that mRNA are produced in the nucleus of the cell and then move to the cytoplasm, where the ribosomes transform mRNA in proteins. We would expect to have a proportional number of proteins and mRNA, but thanks to the post-transcriptional mechanism it's possible to reduce the number of proteins, inhibiting the mRNA before arriving in the ribosome. The cells must have some way to handle the number of proteins in order to react to outer stimulus and this can be done with microRNAs. miRNAs get attached to some proteins (Argonaut) that move them to the target mRNA. Then miRNAs silence a target mRNA with the bounding. So miRNAs attack to a specific part of the mRNA and this part can't be targetized by the tRNA no more, so mRNA cannot produce proteins. We want to know what miRNAs we have in our sequencing data, comparing them with some short pieces of mRNA. As we can see in figure 1 miRNAs can be divided into seed region (from 2' to 7', most important part to have a perfect bound with the mRNA), offset (8'), supplementary (from 14' to 17') and central (from 5' to 17').

Offset, supplementary and seed are named interaction sites, because they are the most important regions for stability. Conservation of interaction sites means perfect match between these regions. We call isoform those miRNA sequences that have been mutated and that have some mismatch with the original miRNA. If we have too many mismatches, the sequence can't be considered no more an isoform of the specific miRNA but it will be a different miRNA. To classify different miRNAs, we can either use a traditional algorithm and try to align them with the DNA sequence or create a database with all the already known

sequences and try to align with them. The problem with the latter is that we are limited because we use only a set of already known sequences. If I have a miRNA of type X in the database and my sequence align perfectly with it, it will be of type X too. If I have more than 3 mismatches, probably I don't have that sequence in my DB, if I have less than three probably is an isoform. The tool identifies not only the miRNA but also their isoforms. Isoforms can be defined as follows:

- **5' iso:** if the first base is different;
- **3' iso:** if the bases after the supplementary regions differs;
- **SNP:** if there is a single mutation on offset or supplementary regions;
- **MSNP:** if there are multiple mutations (maximum 3);

2.2 IsomiR-SEA

2.2.1 Algorithm Explanation

We can divide the algorithm in 3 steps:

1. **Input set up:** we need two input files, the first contains miRNAs sequences and the second stores the input tag table. The file containing miRNAs sequences, downloaded from miRBase database, has to be processed in the so called mature DB seed-based sorting step, where all the seeds are firstly extracted from the miRNAs reference sequences, then classified depending on the species and finally grouped together. The second input file is the standard tags counts file. Generally this file is obtained by means of freely available tools performing the following steps:
 - (a) trimming of adapter sequences on raw fastq or fasta reads, output of sequencing machines;
 - (b) grouping in unique sequences called tags;
 - (c) calculation of the occurrence of reads supporting the various tags.
2. **Elaboration:** each miRNA seed is searched, in this phase, into each stored tags. Tags harbouring the seed are extended without gaps allowed in both directions (ungapped extension). This procedure is iterated for all the miRNAs sharing the same detected seed in a position compatible with miRNA structure (no shifted). Tags not satisfying these conditions are currently discarded and will be not considered in the further steps of the analysis, increasing, in this way, the computational speed. The selected tags are further extended in 3' direction allowing for the presence of a second mismatch. At this point, the distance of the second encountered mismatch with respect to the previous found, is evaluated. Tags characterized by two mismatches in the first 10 aligned nucleotides are discarded, whereas the others are further extended until the end of the alignment or a third mismatch.

3. **Output:** Five output files are generated at each isomiR-SEA run. One file ("align.log") contains all the input parameters previously chosen and the statistics of the computation. Two files ("tagMir-all.tab" and "tagPrxMir-all.tab") provide all the information computed by the program (isoforms, tag quality, tag count, align statistics and many more). The difference between them is that the second analyzes also pre-miRNA. Finally we have two more files ("tagMir-all.gff" and "tagPrxMir-all.gff") concerning the analysis using GFF file (freely downloadable from mirBase).

3 Requirements

This section describes the requirements, both functional and non-functional, that we find important for our scope.

3.1 Functional requirements

These requirements are essential for the pipeline to work properly.

- **Interoperability between components:** one of the most crucial requirement. Indeed since many algorithms have been designed to perform different tasks, there is the need to make all different components work seamlessly in a pipeline. This task is though hard to achieve, as these algorithms may have been written in different programming languages and for sure by different teams and organizations, such that the output from one cannot be used directly as output for others. Bridging scripts are then necessary.
- **Scalability:** ideally the pipeline should be designed to be independent from the bio-informatics tool itself. Furthermore, given the relatively high computation effort that such tools usually require, it's important for the web application to be as lightweight as possible. Otherwise, the front-end wouldn't be as smooth as one would expect, losing the opportunity to scale the system worldwide. Although this requirement could be hard to achieve, it would lead to a major adoption in the industry, increasing so the amount of data publicly accessible by scientists for further analyses.
- **Interactive data visualization:** another crucial requirements, since all bio-informatics tools we have analyzed so far presents a very poor user friendly interface, making hard for simple users other than for experimental scientists to access output data and maybe reuse them for downstream analysis. A valid tool should be up to date to the newest web technologies (e.g. Javascript), in order to organize results in a user-friendly manner, make them fully accessible via a web interface and enable end users to interactively digest analysis results in a user friendly manner.

3.2 Non-functional requirements

These requirements don't interfere with the pipeline functioning, but are anyway important for the growth of the technology and more importantly to allow a wide adoption of it by end users.

- **Open-sourceness:** all components of the pipeline must be freely available in the public domain, opening the way for further improvements.
- **Simple parameter setup:** especially for researchers new to this field, the selection of the right tool (between many different) and the specification of the parameters may be non trivial. It's required both experience and a deep knowledge of the algorithms. Our ideal bioinformatic tool should then facilitate this task, by means of a friendly user interface, along with a manual that drives them through the configuration extensively.
- **High speed processing:** this requirement can be met by introducing a high level of parallelism wherever possible. For instance in the very first step, where each sample is processed independently from other, parallelism may be used.
- **Input consistency:** as we have seen that not all tools require the same gene annotation file format (e.g. GTF or BED), we believe that a valid pipeline should avoid any discrepancy or inconsistency between gene annotations formats. Already deployed scripts that convert files from GTF to BED or viceversa are available, and should be incorporated into the data input phase, making the pipeline more fluid.

4 Implementation

Given the requirements we believed important for this application, we designed our system to be as much modular as possible, so that it could be reused with different tools with the minimum amount of effort, other than ensuring scalability, high speed and fluidity of use. The key to obtain these requirements is to separate client context from server one. Usually web application are based on the concept of **server-side rendering** that works by converting HTML files in the server into usable information for the browser. Back in the day, when most web pages were mostly just for displaying static images and text, this approach was fine. Applied to our system, server side rendering would surely lead to a relevant slowdown on the performances because in addition to the high workload given by the isomiR computation, it would require by the server also a continuous re-render of the screen (with plots and an interactive front-end) We adopted instead a **client-side rendering** approach, that rather than fetching all of the content from the HTML document itself, fetches only a bare-bones of it, with a JavaScript engine that will render the rest of the site. This way,

the front-end interface is managed almost entirely by the client's browser, having the back-end server waiting until a computation is needed. We achieved to maintain the highest computation speed while having at the same time lots of interactions from multiple users on the front-end.

4.1 Client Side

For the front-end of the application, which runs on the browser, we used **React** [5], a Javascript framework designed by the Facebook team to create interactive user interfaces, by means of class-based components, including **Chart** [1], another Javascript library that comes up with plot charts components. We have constructed our front-end so that a new update of the tool could be seamlessly be integrated just by working on a JSON file (figure 2), that contains the input parameters of the tool. The HTML will be automatically generated in the browser reading the fields from that JSON file. So rather than creating a static HTML/CSS web-page, that would end up being useless with a new tool or even with minor changes to the current one, we designed an engine that is independent from the specific parameters. Specifically, the Javascript engine we built will parse the JSON file by reading:

- **Type:** will determine the HTML element to render. Currently implemented are types "number" or "text" (will render an HTML text input, accepting only doubles or text), "radio" (will render an HTML radio element, parsing the array "options" with all the possible values), "switch" (will render an HTML checkbox element)
- **Id:** will determine the label to the command line, that will be matched to the value given through the user interface.
- **Label:** will determine the message appearing on the user interface for that specific field.
- **Help:** will determine an additional helper message, appearing as popup after the user clicking the question mark icon.

Given the high amount of parameters available to set, if we had followed the examples of the other two interfaces we analyzed, we would have caused confusion in the user, ending up with a tool that still wouldn't respect the requirements we defined before. Instead we created a tabular screen (Figure 3), by splitting parameters according to their semantic meaning. Other than giving a much more user friendly experience, this solution has a functional utility as well. In fact, HTML elements are rendered on the fly with the tab that is visible, while all the hidden others are not. With an high amount of elements, this means a significant memory saving, and so an higher fluidity.

```

{
  label: "Gap extend cost",
  help: "Gap extend cost for the Smith-Waterman alignment",
  id: "ge",
  type: "number"
},

{
  label: "Match score",
  help: "Score of a match for the Smith-Waterman alignment",
  id: "ma",
  type: "number"
},

```

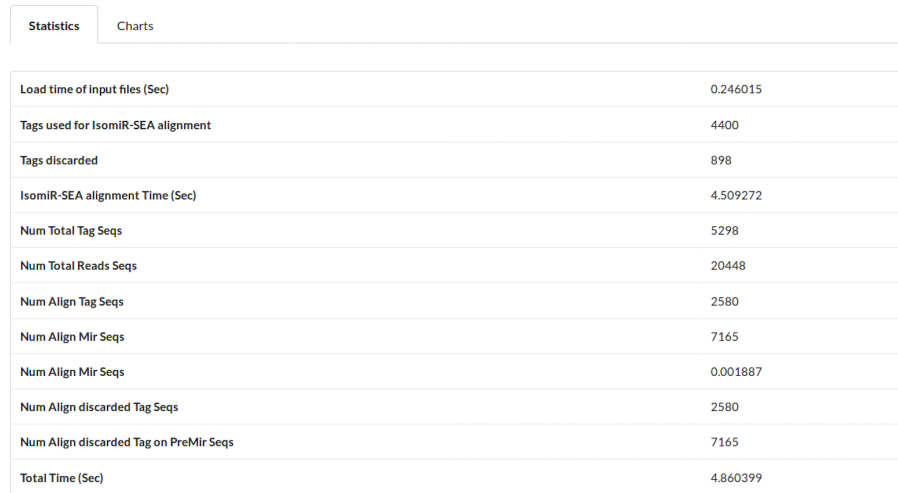
Figure 2: Snippet of the JSON file containing the tool parameters

The screenshot shows the 'isomiR-SEA' web application interface. At the top, there is a teal header bar with the application name and navigation links: 'Start', 'Docs', 'GD', 'Contacts', and a user icon. Below the header, a message states: 'The tool can be run simply by setting mandatory parameters. All the others are optional'. To the right of this message is a green 'RUN' button. The main content area features a tabbed interface with five tabs: 'Mandatory Params' (selected), 'Alignment Params', 'Input Params', 'Database Params', and 'Performance Params'. Under the 'Mandatory Params' tab, there are two input sections. The first is labeled 'Input file storing Tags' and contains a text input field with an 'UPLOAD' button to its right. The second is labeled 'Name of input file storing mature/star miRs' and also contains a text input field with an 'UPLOAD' button to its right.

Figure 3: Tabular front end

4.1.1 Plot design

As long as the output representation is concerned, we created an elegant, flexible and super fast interface. The chart we designed is input responsive, it re-renders as soon as a new miRNA ID is typed without even the need of reloading the page. First off, as shown in figure 4, we show the user some statistics about the computation performances.



Statistics	Charts
Load time of input files (Sec)	0.246015
Tags used for IsomiR-SEA alignment	4400
Tags discarded	898
IsomiR-SEA alignment Time (Sec)	4.509272
Num Total Tag Seqs	5298
Num Total Reads Seqs	20448
Num Align Tag Seqs	2580
Num Align Mir Seqs	7165
Num Align Mir Seqs	0.001887
Num Align discarded Tag Seqs	2580
Num Align discarded Tag on PreMir Seqs	7165
Total Time (Sec)	4.860399

Figure 4: Computation statistics

Then (Figures 5 and 6) the user has the possibility to analyze the Isoform Distribution of a single miRNA ID rather than of multiple miRNA in a stacked bar chart, possibly with an infinite amount of IDs. Furthermore, for each Isoform the Tag Count value is reported on the y2 axis. As you can see from the pictures, we implemented also a function to export the plots in .jpg and download them. It is also possible to download the five output files generated by isomiR-SEA.

Insert one or multiple miRNAID, divided by ', ' (ex MIMAT0000070 or miR-17-5p)

MIMAT0000070

Plot

Download as PDF

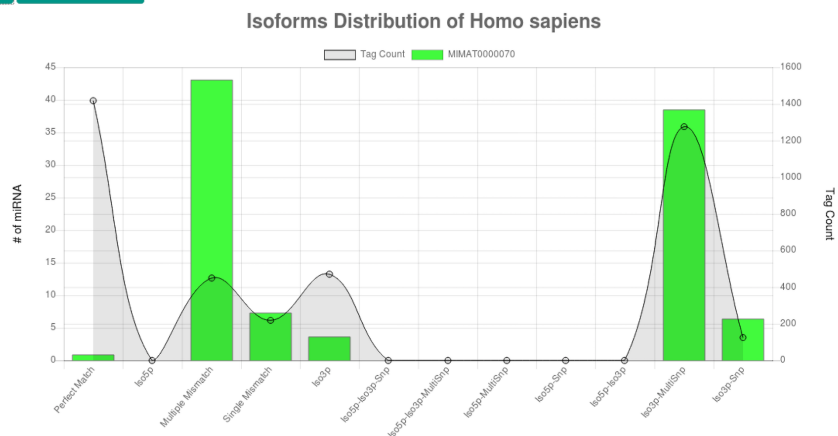


Figure 5: Single miRNA ID input

Insert one or multiple miRNAID, divided by ', ' (ex MIMAT0000070 or miR-17-5p)

MIMAT0000334, MIMAT0000358, let-7-5p, miR-313-3p

Plot

Download as PDF

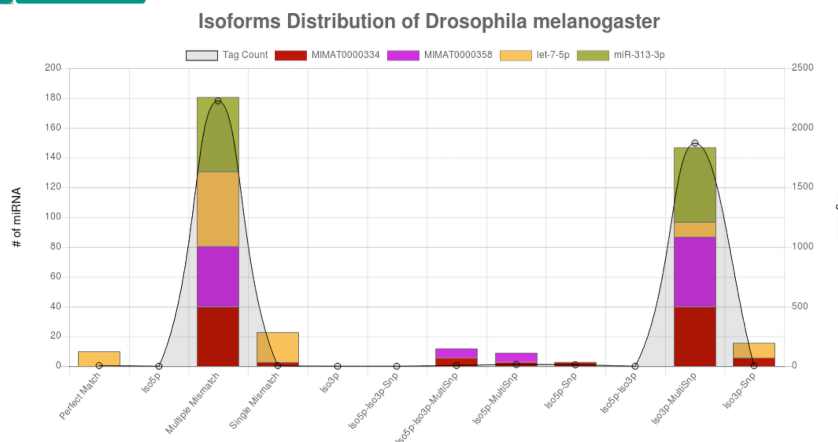


Figure 6: Multiple miRNA ID input

4.2 Server Side

Other than a friendly and responsible user interface, it was important and crucial for this work to setup a robust server side, given the high amount of computation and operations required. Of course server is responsible of getting requests of new computation coming from clients, running the isomiR-SEA script, and returning the results back to client. So we implemented our back-end server such that even when encountering errors during the computation (for example caused by bad input format), instead of crashing it handles the error updating the front-end.

To achieve this, we have used **Node and Express** [4] [3]. Node is an asynchronous event driven JavaScript runtime, designed to build scalable network applications. Indeed almost no function in Node directly performs I/O, so the process never blocks. Because nothing blocks, scalable systems are very reasonable to be developed. Express instead is a flexible and lightweight framework for Node.js web-based applications, that provides an easy way to creates API. In our case, it was useful for managing HTTP communications and data transfer between client and server.

References

- [1] Multiple authors. <https://www.chartjs.org/>.
- [2] National Center for Biotechnology Information. <https://blast.ncbi.nlm.nih.gov/blast.cgi>, August 2018.
- [3] Linux foundation. <https://expressjs.com/it/>.
- [4] Linux foundation. <https://nodejs.org/it/>.
- [5] Facebook Inc. <https://reactjs.org/>.
- [6] Sarah Du Shanrong Zhao, William Gordon. Quickmirseq: a pipeline for quick and accurate quantification of both known mirnas and isomirs by jointly processing multiple samples from microrna sequencing. 2017.
- [7] Sarah Du Shanrong Zhao, William Gordon. Quickrnaseq lifts large-scale rna-seq data analyses to the next level of automation and interactive visualization. 2017.
- [8] Kengo Saton Yuki Kato. <https://rtips.dna.bio.keio.ac.jp/ipknot/>, March 2014.