Docking assays of Musashi-1 allosteric RNA-binding sites

Universitat Oberta de Catalunya

Universitat de Barcelona

Lucas Goiriz Beltrán

MU Bioinf. i Bioest.
Área de trabajo final

Nombre Tutor/a de TF

Emmanuel Fajardo

Profesor/a responsable de la asignatura

Emmanuel Fajardo

Fecha Entrega

12 de Enero 2023

Universitat Oberta de Catalunya uoc.edu

FICHA DEL TRABAJO FINAL

Título del trabajo:	*Docking assays of Musashi-1 allosteric RNA-binding sites*
Nombre del autor:	*Lucas Goiriz Beltrán*
Nombre del consultor/a:	*Emmanuel Fajardo*
Nombre del PRA:	*Emmanuel Fajardo*
Fecha de entrega:	*01/2023*
Titulación o programa:	*Máster Universitario en Bioinformática y Bioestadística*
Área del Trabajo Final	*Drug Design and Structural Biology*
Idioma del trabajo:	*Inglés*
Palabras clave:	*Molecular Docking, RNA-binding protein, Allosteric Regulation*

Resumen del Trabajo
Máximo 250 palabras, con la finalidad, contexto de aplicación, metodología, resultados y conclusiones del trabajo
Abstract
A maximum of 250 words, detailing the purpose, context of application, methodology, results and conclusions of the work

Index

List of Figures

1 Introduction

1.1 Context and justification

The Musashi-1 (MSI-1) protein belongs to a family of neural RNA-binding proteins (RBPs) that play a fundamental role in neural development in both vertebrates and invertebrates [2, 3, 4, 5]. In particular, the mouse MSI-1 consists of 362 aminoacids with a molecular mass of 39 kDa [3], and contains 2 sequences of around 80 to 90 aminoacids that interact with RNA (RRMs; short for RNA-recognition motifs) following the consensus sequence $\texttt{RU}_n\texttt{AGU}$[1,2] [5, 7].

Despite multiple efforts, the complete structure of any of the MSI-1 protein homologues remains a mystery. The most notable successes have only been able to completely elucidate MSI-1's RRM1 and RRM2 domains [8, 9, 10, 11], see for example **Figure 1** for the 3D structure of RRM1.

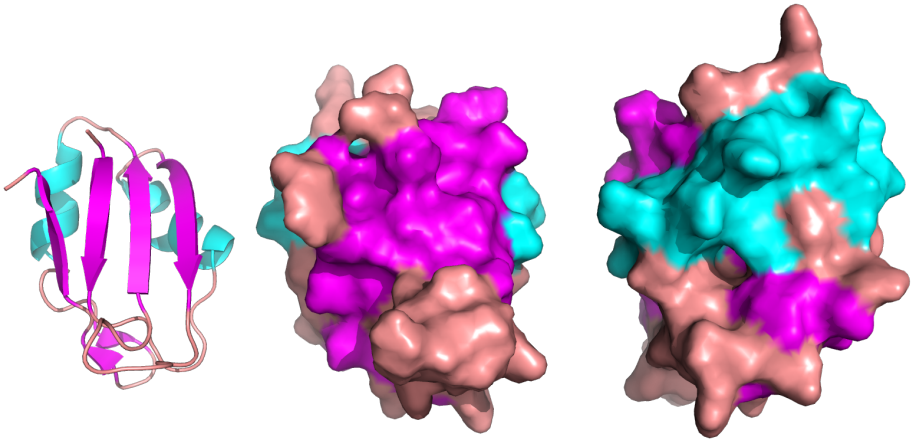

Figure 1: 3D RRM1 structure of the MSI-1 protein constructred by means of protein crystalization and RMN. From left to right: RRM1 in *cartoon* setting, RRM1 in *surface* setting and RRM1 in *surface* setting turned 180°. Colors represent the secondary structures found within the protein: cyan corresponds to α-helices, magenta corresponds to β-sheets and salmon corresponds to *random coil*. The model was retrieved from Uniprot entry Q61474. Visualized through Pymol.

[1] Following the IUPAC standard for degenerate base symbols, R represents both an Adenine (A) and a Guanine (G). Refer to [6] for the complete IUPAC standard.

[2] $\texttt{U}_n$ stands for repeating U within the consensus sequence n times. In most cases, $n \in \{1, 2, 3\}$ [5].

At the moment, the available complete MSI-1 structures only originate from computational predictions, such as the one shown in **Figure 2** which is an aminoacid-sequence-based prediction yielded by AlphaFold [12].

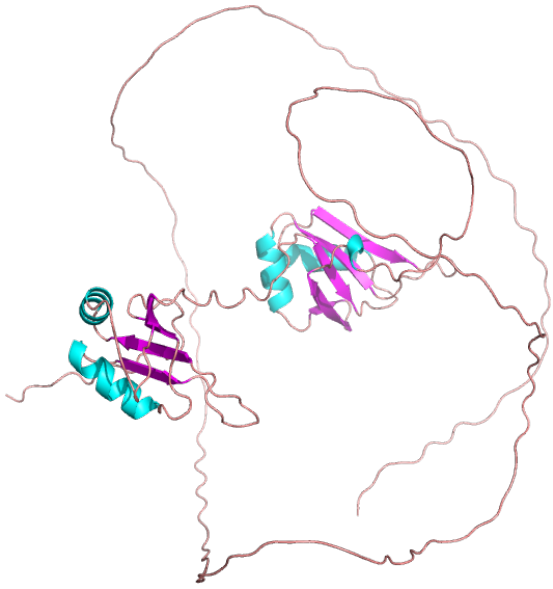

Figure 2: 3D mouse MSI-1 structure as predicted by AlphaFold, in *cartoon* setting. Colors represent the secondary structures found within the protein: cyan corresponds to α-helices, magenta corresponds to β-sheets and salmon corresponds to *random coil*. The model was retrieved from Uniprot entry Q61474. Visualized through Pymol.

With the surge of synthetic biology during the past two decades, RBPs such as MSI-1 have gained a special interest for their applicability as post-transcriptional regulators of gene expression in synthetic gene circuits [13, 14, 15]. In particular, Dolcemascolo and colleagues pioneered by designing a synthetic genetic circuit employing MSI-1 as post-transcriptional regulator [1]. In addition, they also exploited MSI-1's particularity of binding to fatty acids in addition to RNA, out of which the cis-9-Octadecenoic acid (also known as oleic acid) presents a higher binding affinity. The latter interaction weakens the MSI-1 protein's RNA-binding affinity, resulting in a dissociation between MSI-1 and the RNA sequence [1, 16]. In fact, the pocket in which the fatty acids interact with the RMM1 domain of MSI1 can be easily seen in the rightmost protein in **Figure 1**.

Therefore, in order to confidently add MSI-1 as an allosteric regulator to the synthetic biologist's toolbox, there is a need to understand how the underlying interaction mechanisms work. Clingman and colleagues are were the first

who performed docking simulations with the MSI-1 core RNA-binding-motif and several fatty acids [16]. Nevertheless, these results only hold for the aforementioned RNA-motif: it is expected that any mutation on the RNA-motif would drastically impact the manner in which MSI-1 interacts with the RNA.

1.2 Objectives

This Master's Thesis aims to extend the knowledge behind the allosteric interactions occurring in the MSI-1 proteins by performing docking simulations between MSI-1's RRM1 domain, several RNA motifs and fatty acids, with the objective of obtaining a model for each of the combinatorial cases.

This general objective breaks down into the following specific objectives:

1. *In silico* Generate the 3D structures of 5 mutant MSI-1 RNA-binding motifs used in [1]. The 3D structures of the motifs will depend on the sequence-based-prediction of their secondary structures.
2. Assess the structure of MSI-1's RRM1-RNA mutant complex through docking for each of the mutants.
3. Assess the structure of MSI-1's RRM1-fatty acid complex through docking for each of the fatty acids of interest: oleic acid, linoleic acid, palmitoleic acid, arachidonic acid and stearic acid.
4. Keep the top 2 fatty acids whose interaction with MSI-1 is the clearest.
5. Assess the structure of RNA mutant-MSI-1's RRM1-fatty acid complex through the coupling of the docking models obtained.

1.3 Impact on sustainability, social-ethical and diversity

1.3.1 Sustainability

The result of this Master's Thesis has had a negligible but negative impact on sustainability. This impact originates on the energy consumption during the following steps:

1. Design and debugging of the docking pipeline followed in this work.
2. Computation of the different models.

This impact can be considered negligible because the net amount of energy consumed during the computations is very small. Nevertheless, it is important to highlight this aspect as there is room for improvement: the source code of the docking software employed in this work is mainly composed of two programming languages: `C` (72%) and `Python` (28%).

Pereira and colleagues developed a benchmark to order programming languages with respect of their energy consumption when executing well-known computer science algorithms and data structures [17]. Their finding was that the least energy consuming programming language was `C`, while `Python` was one of the most energy consuming.

It can be discussed that, in order to reach minimal energy consumption, the programming language to be used in software that is going to be distributed and executed multiple times should be exclusively `C`. The disadvantages of `C` are that it takes more time to write a full program than with higher level languages (such as `Python`), it is more difficult to mantain a codebase written in `C` and that `C` is not a particularly accessible programming language for non-computer scientists.

There is a trade-off that has be to be considered. Let L be a programming language, E_{W_L} be the energy consumed by a computer during the crafting of a software written in L, E_{E_L} be the energy consumed by a computer during one execution of the software written in L (in this particular case, one docking simulation with some fixed parameters) and n be the number of times the software is executed, then it is possible to approximate $E_{T_L}(n)$ (the total energy consumed by the software written in L at the moment n) as follows:

$$E_{T_L}(n) \approx E_{W_L} + n \times E_{E_L} \tag{1}$$

For the pair `Python` and `C`, we have that $E_{W_{\texttt{Python}}} <<< E_{W_{\texttt{C}}}$ and $E_{E_{\texttt{Python}}} >>> E_{E_{\texttt{C}}}$. So the larger n becomes (the number of times the software is executed), by **Equation 1**, `Python` becomes less and less energy efficient when compared to `C`.

Nevertheless, as mentioned previously, the codebase of the software used is in its majority written in `C`, the energy consumption of a personal computer is low and the number of executions n in this work is small (less than 10^2). Therefore this negative impact on energy consumption can be considered negligible.

1.3.2 Social and ethical responsibility

The technical nature of this work separates it from any positive nor negative impacts on socio-ethical aspects. There are no conflicts of interests with the results, and this work's material (models and source codes) are freely available to anyone through a public `GitHub` repository.

In addition, this work doesn't have any socio-ethical motivations, as the main objective is to gather more technical knowledge for the field of molecular biology.

1.3.3 Diversity, gender and human rights

Again, the technical nature of this work separates it from any positive nor negative impacts on diversity, gender and human rights. The references and materials employed within this work were used without any gender nor race bias.

Furhermore, this work doesn't have any diversity, gender nor human rights motivations. However, this work attempts to be accessible to as many people as possible, in the spirit of the universal right for knowledge and education. Moreover, this work exclusively employed license-free and open sourced software so that all the dependencies are accessible to anyone.

1.4 Approach and methods

This work aims to extend the work of Dolcemascolo and colleagues [1] where they designed a synthetic genetic circuit that employed MSI-1 as post-transcriptional regulator. Thanks to the plasticity of RNA, several mutants of MSI-1's RNA-binding core motif were designed, and it was shown that gene regulation (which is tightly dependent on the MSI-1-RNA interaction) was altered for all mutants: some of them displayed an increase in regulation fold change while others displayed a decrease. In addition, the effects of MSI-1 induced regulation were partially nullified in presence of oleic acid.

The initial step of this work will consist in the retrieval of several 3D structures from the appropriate databases:

- the 3D structure of mouse MSI-1's RMM1 is retrieved from `UniProt`.
- the 3D structures of several fatty acids (oleic acid among them) are retrieved from `chemspider`.

Next, some of the RNA sequences employed in [1] are retrieved and their 3D model structures are computed by means of NUPACK and RNAComposer web server.

Then, all the docking simulations are performed:

1. Between each RNA sequence and MSI-1 RMM1.
2. Between each fatty acid and MSI-1 RMM1.
3. Between the models obtained in the simulations of step 1 and the fatty acids.

The choice of docking software will vary for the type of docking simulation to be performed:

- For protein-RNA docking, the docking software will be LightDock because it is license-free, open source and it is partially written in Python.
- For protein-lipid docking, the docking software will be AutoDock Vina, aided by AutoDock Tools for the preparation step.

The best models will be selected based on the chosen scoring function and biological significancy. In addition, all the models will be visualized with Pymol.

1.5 Work plan

This work is divided in the following tasks (and respective subtasks):

1. Perform docking simulations of MSI-1 and a collection of mutants of the MSI-1 RNA-binding-motif.
   (a) Generate in silico a collection of up to 5 interesting mutant MSI-1 RNA-binding motifs.
   (b) Perform docking simulations between MSI-1 and each of the mutants.
   (c) Discard those mutants whose interaction with MSI-1 is poor.
2. Perform docking simulations of MSI-1 and a collection of fatty acids.
   (a) Perform docking simulations between MSI-1 and several fatty acids of interest: oleic acid, linoleic acid, palmitoleic acid, arachidonic acid and stearic acid.

(b) Keep the top 2 fatty acids whose resulting model is the best.

3. Perform docking simulations of MSI-1 and a subset of mutants of the MSI-1 RNA-binding-motifs in presence of fatty acids.

4. Coupling of the successful docking models to visualize the lipid-MSI1-RNA complex.

5. Defense preparation
   (a) Manuscript
   (b) Presentation

The landmarks of this work are distributed among the following continuous assessment tests:

- Continuous Assessment Test 1: Thesis Definition and Work Plan.
- Continuous Assessment Test 2: Work Development (Phase 1).
  Landmarks:
  - Generate in silico a collection of up to 5 interesting mutant MSI-1 RNA-binding motifs.
  - Perform docking simulations between MSI-1 and each of the mutants *(partially)*.
- Continuous Assessment Test 3: Work Development (Phase 2).
  - Perform docking simulations between MSI-1 and each of the mutants.
  - Discard those mutants whose interaction with MSI-1 is poor.
  - Perform docking simulations between MSI-1 and several fatty acids of interest: oleic acid, linoleic acid, palmitoleic acid, arachidonic acid and stearic acid *(partially)*.
- Continuous Assessment Test 4: Thesis Closure and Presentation.
  - Perform docking simulations between MSI-1 and several fatty acids of interest: oleic acid, linoleic acid, palmitoleic acid, arachidonic acid and stearic acid.
  - Keep the top 2 fatty acids whose resulting model is the best.
  - Manuscript closure.
  - Presentation closure.

The work plan is summarized in **Figure 3**:

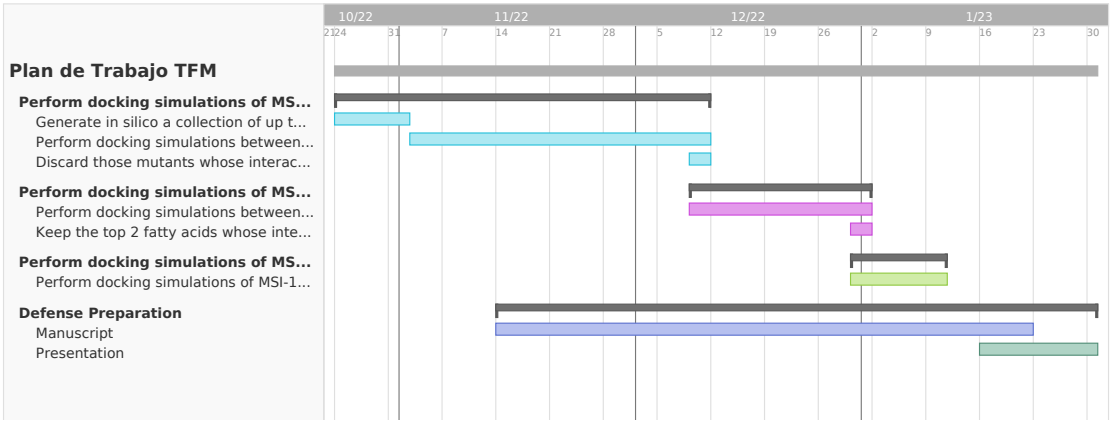

Figure 3: Work plan summarized in a Gnatt chart.

1.6 Summary of obtained products

The main products of this work are:

1. The current Master's Thesis manuscript.
2. 3D structures of 5 MSI-1 RNA-motifs used in [1].
3. *los modelos o lo que salga*
4. *los scripts del pipeline*
5. A step-by-step tutorial on how to perform a simple protein-RNA docking with `LightDock`.

Resulting products are available in different repositories:

1. The direct products of this work are present in this `GitHub` repository.
2. The step-by-step tutorial on how to perform a simple protein-RNA docking with `LightDock` is available at the `LightDock GitHub` repository, and on the `LightDock` tutorials page.

1.7 Summary of other sections of this report

The remaing sections of this manuscript are:

- **State of the art**
- **Materials and methods**
- **Results**
- **Conclusions and further work**
- **Glossary**
- **References**
- **Appendix**

2 State of the art

2.1 Synthetic Biology

Synthetic Biology is a modern niche of Biology in which the characteristic "know-how" from engineering fields is combined with classical Molecular Biology techniques. Guided by physicist Richard Feynman's famous quote "*what I cannot create, I do not understand*", synthetic biologists have developed a deep understanding of how cells control and regulate the processes of gene transcription, translation, metabolite biosynthesis and degradation.

It is "synthetic" in the sense that synthetic biologists engineer novel (absent in nature) genetic constructions with biological "pieces", which usually originate from various different organisms, in order to achieve a phenotype in a target organism that lacks said phenotype in nature. Furthermore, this design usually comes with a mathematical model based in physical first principles of how the system should behave.

One of the first successful constructions of this type, and most likely the most popular one, is Gardner's toggle switch [18] where they engineered a synthetic toggle switch within *E. coli* by using two different promoters that repressed each other, achieving this way bistability (either one promoter is repressed, or the other). Simultaneously, Elowitz' repressilator [19] was published, where they engineered a synthetic "clock" within *E. coli* by chaining 3 repressive promoters that repressed one another in chain, in such a way that the genes controlled by them had periodic expression.

These constructions, and all the ones that followed, would not have been possible without a (by then, small) repository of genetic parts, which in turn would not have been possible without the arduous work of an innumerable amount of traditional molecular biologists that sequenced and characterized said parts.In fact, the iGEM foundation keeps a repository of standarized biological parts [20] along with various data such as the assembly methods used, levels of expression, polymerase used, etc...

However, there is still a need for more biological parts, especially parts that are "orthogonal", i.e. they can co-exist and function correctly within the same organism without deviating from their theoretical behavior. This need, plus the known plasticity of the RNA macromolecule, have led to a growing interest in synthetic constructions where gene expression is either directly controlled by RNA [21, 22] or by an RNA-binding protein [13, 14, 15, 23].

2.2 RNA-binding proteins and the Musashi family

RNA-binding proteins (RBPs) are a huge family of proteins that can be found among all domains of life (Bacteria, Archaea and Eukaryota) [24, 25, 26] and can be traced back to the common ancestor of life LUCA [27]. It is therefore no surprise that RBP are still the cornerstone of the majority of cellular processes [26] as they govern mRNA metabolism: mRNA localization, mRNA translation, mRNA degradation, mRNA editing and mRNA stability [28].

The mechanisms in which RBPs interact with RNA differ vastly from DNA-binding proteins (which usually target the double stranded DNA major groove) because RNA comes in a vast variety of structures [28, 29]. The latter can be observed in the shape or sequence specificity displayed by the great variety of known RNA-binding domains (RBDs) [30].

For instance, a popular RBD is the RNA-Recognition Motif (RRM). RRMs typically have a length between 90 and 100 aminoacids (although this can vary), display a $\beta\alpha\beta\beta\alpha\beta$ topology (where β stands for the β-sheet secondary structure and α stands for the α-helix secondary structure), they target single-stranded RNA and are usually present in multiple copies within the same protein, because their interaction with RNA by themselves is weak [28].

The latter makes RBPs containing RMMs interesting candidates for biological parts in synthetic biology, as they display two levels of "tuneability": the number of RMMs present within the protein and the inherent plasticity of the RNA sequence recongnized by them. Such characteristics are present in the homologues of the Musashi (MSI) family of RBPs. For instance, the mouse MSI-1 which consists of 362 aminoacids with a molecular mass of 39 kDa [3] contains 2 RMMs (dubbed RMM1 and RMM2) that interact with RNA containing the consensus sequence $\mathtt{RU}_n\mathtt{AGU}$ [5, 7].

MSI-1 has an additional particularity that distinguishes it from the common bunch of RBPs, and that is the capability of binding to fatty acids and weakening the MSI-1 protein's RNA-binding affinity, resulting in a dissociation between MSI-1 and the RNA sequence [1, 16]. This property plus the inherent properties of RMM-based RBPs make MSI-1 a valuable biological part for synthetic biology as a highly tuneable allosteric regulator, as shown by Dolcemascolo and colleagues [1]. However, this presumed tuneability is not yet deeply understood.

Clingman and colleagues made great discoveries of MSI-1's interaction mechanisms [16] by means of docking assays between different fatty acids and the MSI-1-RNA complex. Nevertheless, due to the plasticity of RNA, the obtained results aren't extrapolable to other RNA motifs that MSI-1 may recognize.

For that reason, this work attempts to shed light on the allosteric interactions occurring in the MSI-1 proteins by performing docking simulations between MSI-1's RRM1 domain, several RNA motifs and fatty acids.

3 Materials and methods

3.1 Data Acquisition

3.1.1 Mouse MSI-1 and MSI-1's RMM1

Uniprot's database was consulted for the structure of the mouse MSI-1 protein (see **Figure 4**). Unfortunately, there was no NMR crystal structure for the complete protein: the only available complete structure was AlphaFold's prediction of the structure (see **Figure 2**). Said structure had low confidence in the aminoacid sequence not corresponding to the RRMs. Therefore, the crystal structure of MSI-1's RRM1 (see **Figure 1**) was to be employed instead for the docking simulations.

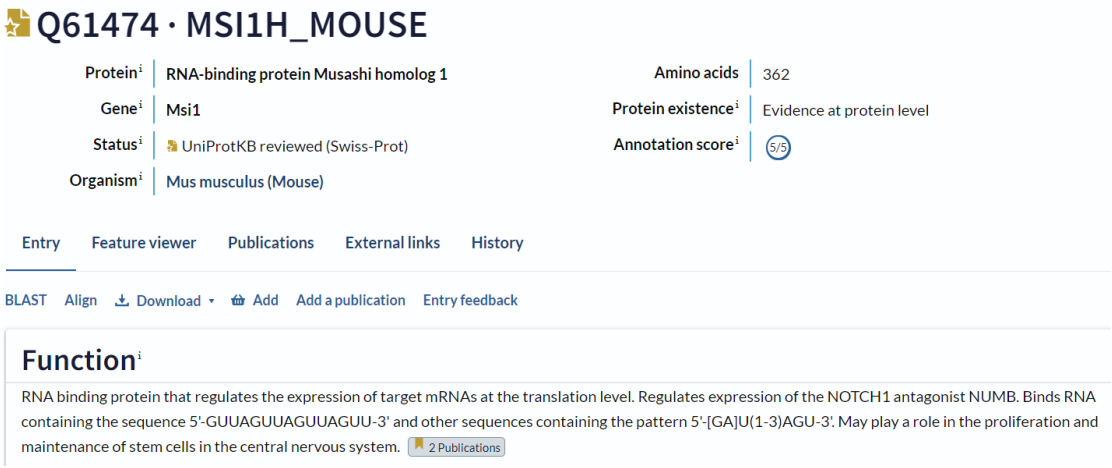

Figure 4: Uniprot entry Q61474 corresponding to the mouse MSI-1 (as seen on 08/01/2023).

The crystal NMR structure was downloaded as a `.pdb` file.

3.1.2 Fatty acids

The fatty acids to be employed down the line in the docking simulations were retrieved from their respective Chemspider entries. The fatty acids downloaded were:

- arachidonic acid
- linoleic acid
- oleic acid

- palmitic acid
- stearic acid

The molecules are shown in **Figure 5**.

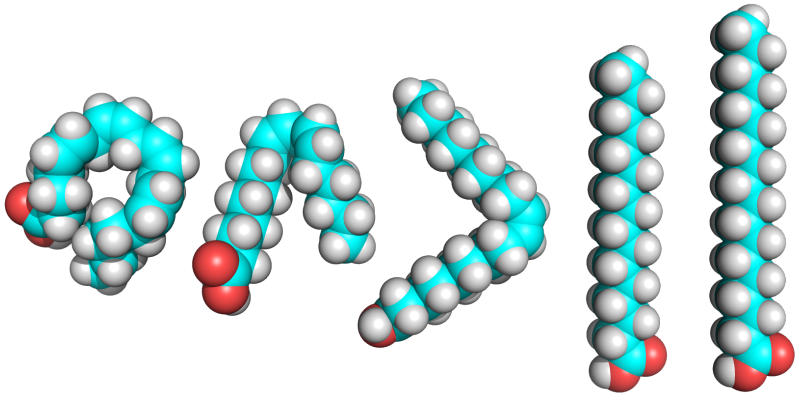

Figure 5: 3D structure of the fatty acids to be employed. From left to right: arachidonic acid, linoleic acid, oleic acid, palmitic acid and stearic acid. Colors represent the atoms present in the structures: cyan is Carbon, red is Oxygen and white is Hydrogen. Visualized through `Pymol`.

The molecules' 3D structures were downloaded as `.mol` files.

3.2 Generation of 3D RNA molecule structures

The raw RNA sequences to be employed were retrieved from [1] and are shown in **Appendix A: RNA motifs employed in this work, as seen in [1] (in `.fasta` format)**, where they crafted small RNA oligos based on a the selex optimal RNA sequence. RNA takes specific spatial conformations when in aqueous media, be it *in vitro* or *in vivo*, which can be computed directly from the RNA sequence by means of pre-measured experimental parameters [31].

To do so, the Nupack software was employed [32] (specifically its associated `Python` package) and the structures in dot-bracket notation[3] were recorded. In addition, the sequences were subjected to pairwise alignments (by means of MAFFT [34]) to highlight the mutations present with respect to the original oligo. The `Python` script employed to perform these computations is found in **Appendix B: `Python` script to treat the RNA motifs**.

[3]Dot-bracket notation is a simplistic manner of representing DNA or RNA secondary structure. For more details, see [33].

The results are shown in **Figure 6**:

```
   name  seq alignment structure
0 orig  GGCAGCGUUAGUUAUUUAGUUCGUAUGCC ***************************** (((((((.(.........)..))).))))
1 mut1  GGCAGCGUUCGUUAUUUAGUUCGUAUGCC ******** ****************** (((((((..(.........)..))).))))
2 mut2  GGCAGCGUUACUUAUUUAGUUCGUAUGCC ********** ****************** (((((((..(((.....))).))).))))
3 mut3  GGCAGCGUUAGCUAUUUAGUUCGUAUGCC ********** ****************** (((((((..(((.....))))))).))))
4 mut4  GGCAGCGUUAGUUAUGUUAGUUCGUAUGCC  ********** ********** (((((((.(.........)..))).))))
5 mut5  GGCAGCGUUACUUAUUUACUUCGUAUGCC ********** ******* ********** (((((((.............))).))))
```

Figure 6: RNA sequences along with the result of the pairwise alingments and Nupack's prediction of their secondary structure.

With this information (RNA sequence and secondary structure), the 3D conformation of each of the RNA motifs was computed by means of the `RNAComposer` web server [35] and downloaded in `.pdb` format. The corresponding 3D structures of the RNA motifs are shown in **Figure 7**.

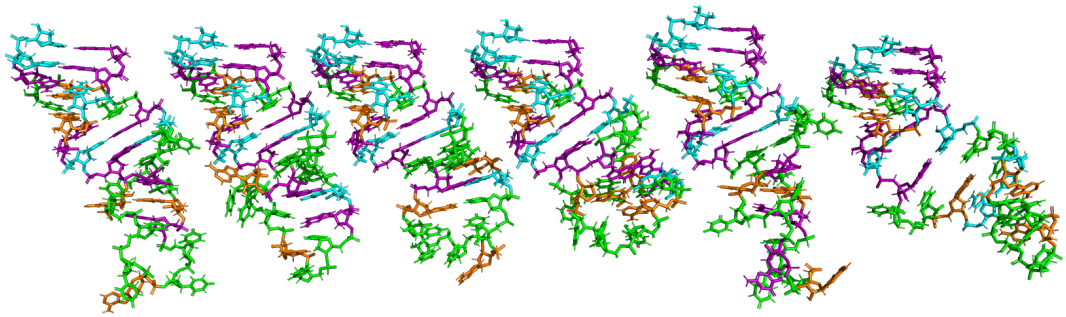

Figure 7: 3D structures of the RNA motifs. From left to right: the original RNA motif followed by the 5 mutants. Colors represent the nucleotidic bases: orange is Adenine, purple is Guanine, cyan is Cytosine and green is Uracil. Visualized through `Pymol`.

With the 3D structures of the RNA motifs, the next step is to prepare the protein-RNA docking simulations.

3.3 Protein-RNA docking simulations with `LightDock`

For the protein-RNA docking simulations, the `LightDock` [36] docking software was chosen. The main reasons are the simplicity of `LightDock`, the quality of the documentation and that the software is written in `Python`, which eases debugging steps in case of complications. Originally, `LightDock` was not meant

for protein-RNA docking simulations. However, by means of some reasearch and auxiliary Python scripts, a simple and novel pipeline was designed which makes it possible to perform such docking simulations with LightDock. This pipeline takes inspiration on the protein-DNA docking example tutorial from the LightDock documentation.

3.3.1 Protein preparation

First, MSI-1's RRM1 .pdb file needs to have their hydrogen atoms modified and relabeled so that they match the AMBER94 force field, which is the parameter set that is going to be used in this docking simulation. To do so, the softwares REDUCE and PDB-Tools [37] were employed.

In addition, the MSI-1's RRM1 .pdb file needed an additional motification related to the AMBER94 force field: regular .pdb files employ the HIS identifier for the histidine aminoacid, which does not appear in the AMBER94 force field. The reason behind this is that for the AMBER94 force field, histidine can be one of 3 possible residues [38]:

1. HID: histidine with a single hydrogen on the delta nitrogen
2. HIE: histidine with a single hydrogen on the epsilon nitrogen
3. HIP: histidine with hydrogens on both nitrogens (delta and epsilon). This histidine has a positive charge

Therefore, one has to check which kind of histidine residue is present in the molecule, and change it for the corresponding AMBER94-compatible identifier. This can be done with a simple Python script called fixHIS, which is found in **Appendix C: Python script to relabel HIS residues for their corresponding AMBER94 compatible residues**.

3.3.2 RNA preparation

Similar to the protein, the RNA structures' .pdb file need to have their hydrogen atoms modified and relabeled. Again, the softwares REDUCE and PDB-Tools [37] were employed.

In addition, REDUCE adds the 3' and 5' hydroxile groups to the RNA structures. These atoms aren't present in the AMBER94 force field, therefore it is mandatory to remove them in order to avoid issues downstream. This can be done with a simple Python script called removeEnds.py, which is found in **Appendix D:**

Python **script to remove 3' and 5' hydroxile groups in the RNA structures**.

Next, due to particularities of REDUCE for nucleic acids, it is necessary to rename and remove incompatible atom types present in the RNA structure. Again, this can be done with a simple Python script called reduce_to_amber.py, which is found in **Appendix E:** Python **script to rename and remove incompatible atom types from the RNA srtructure**. Note that the aforementioned Python script is an adaptation of the original version provided in the protein-DNA docking example tutorial from the LightDock documentation.

Finally, in order to sucessfully run LightDock's setup phase, it is mandatory to remove the R tags within the RNA structures. The R tags are important because they indicate which AMBER94 parameters to use during the docking simulations, however one of LightDock's dependencies in the setup phase (ProDy) has conflicts with this notation. This removal can be done with a simple Python script called manipulateRNA.py, which is found in **Appendix F:** Python **script to remove the incompatible** R **tags from the RNA structures**.

3.3.3 LightDock setup and simulation

LightDock's setup itself is simple, as it only consists of executing the setup script with the structures prepared in the previous steps, as shown in **Figure 8**:

```
1 lightdock3setup.py protein.pdb rna.pdb -anm
```

Figure 8: LightDock setup command.

The setup will generate, among other files, the simulation-ready structures lightdock_protein.pdb and lightdock_rna.pdb. For the docking simulation to be successful, it is crucial to add the R tags within the lightdock_rna.pdb RNA structure, which is basically reversing the tag removal process performed earlier. This can be achieved with a simple Python script called manipulateRNAagain.py, which is found in **Appendix G:** Python **script to recover the** R **tags from the RNA structures**.

3.3.4 LightDock docking simulation, clustering and ranking

If the previous steps executed without issues, it is possible to run LightDock as shown in Figure 9:

```
lightdock3.py setup.json 100 -s dna -c 4
```

Figure 9: LightDock execution command, using the DNA scoring function for nucleic acids and 4 processing cores.

Once the simulation ends, and in order to be able to do the clustering and ranking step, it is mandatory to remove the R tags added to lightdock_rna.pdb for the simulation step. For that, once again the script manipulateRNA.py shown in **Appendix F: Python script to remove the incompatible R tags from the RNA structures** is employed.

Next, the clustering and ranking can be performed in the same way as specified in the protein-DNA docking example tutorial from the LightDock documentation. The bash script performing the ranking and clustering steps is shown in **Appendix H: Bash script to cluster and rank the models generated by LightDock**.

The ranking will generate a solutions.list file where the different models generated by LightDock are scored. For this work, the top 10 best scoring models were taken.

The entire pipeline followed in this section has been implemented as a bash script called run1.sh, which is found in **Appendix I: Bash script to run a complete protein-RNA docking simulation with LightDock.** Given a target_protein.pdb file and a ligand_rna.pdb file, run1.sh is executed as shown in **Figure 10**:

```
bash run1.sh target_protein.pdb ligand_rna.pdb
```

Figure 10: Execution of the run1.sh script.

A refined version of this pipeline has been contributed to the official LightDock tutorials section. In said version, the Python scripts for file manipulation have been cleaned-up and applied to model the 1A1T protein-RNA complex. This tutorial can be directly accessed here.

3.4 Lipid-protein docking simulations with AutoDock Vina

For the lipid-protein docking simulations, AutoDock Vina docking software was employed. The main reasons for changing the docking software at this step

are that for now `LightDock` does not feature full support of lipid-protein docking and that `AutoDock Vina` is license free. For the preparation steps, the `AutoDock Tools` is used for simplicity, although other software could have been used instead.

3.4.1 Protein preparation

First, MSI-1's RRM1 `.pdb` is loaded into `AutoDock Tools`, which displays all the models present in said file, as show in **Figure 11**:

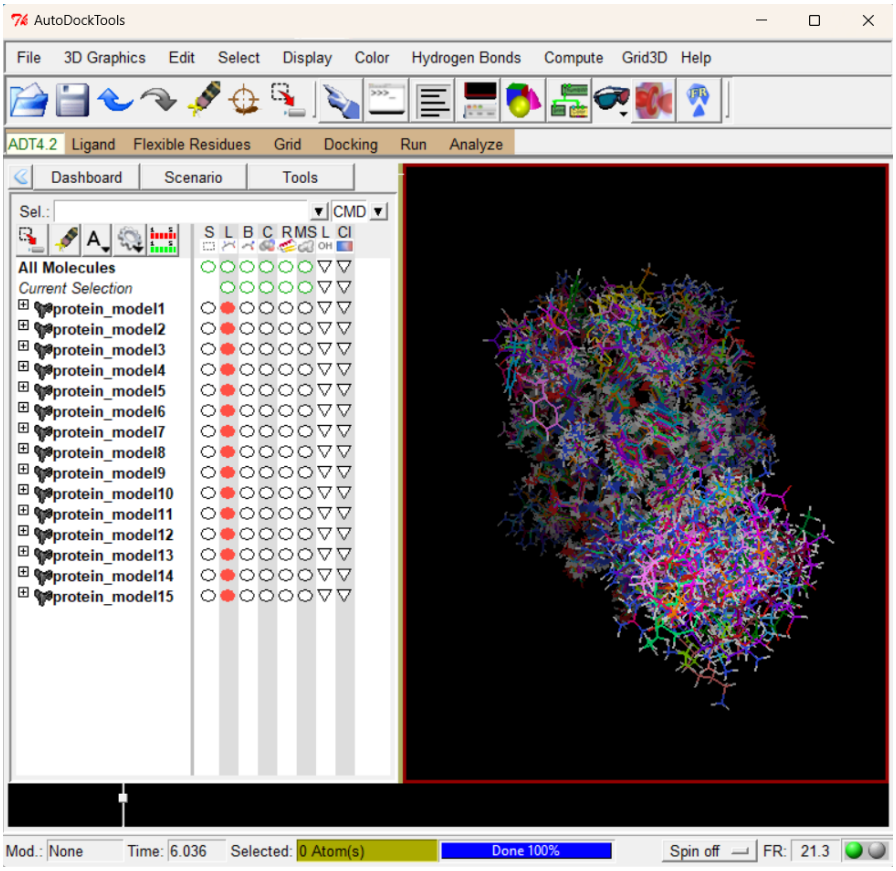

Figure 11: Protein models in `AutoDock Tools`.

It is mandatory to discard the models that are not going to be used in the docking simulations. For simplicity, in this work the model to be kept is the first one.

Next, there are a series of procedures to prepare the protein model for docking:

1. Remove the water molecules by clicking `Edit>Delete Water`. This step may be skipped if there are no water molecules present.

2. Add polar hydrogen atoms by clicking `Edit>Hydrogens>Add` and then select `Polar Only`. Again, this step may be skipped if the hydrogen atoms are already present.

3. Add Kollmann charges by clicking `Edit>Charges>Add Kollmann Charges`. This step is unskippable and it is essential for the docking simulation to be successful.

Once these steps are completed, the protein model is ready for docking and can be saved in `.pdbqt` format by clicking `Grid>Macromolecule>Choose...`.

3.4.2 Lipid preparation

The fatty acid molecules also need a preparation step prior to the docking simulation. First, it is necessary to convert the fatty acid models from `.mol` format to `.pdb` format. One way to do this is to open the `.mol` file in `Pymol` and then saving it in `.pdb` format. An alternative way would be to use an online file converter, such as this one.

Next, the fatty acid in `.pdb` format is loaded into `AutoDock Tools`, where preparation is done by clicking `Ligand>Input>Choose...` and selecting the fatty acid by clicking `Select Molecule for AutoDock4`. This will remove hydrogens and compute charges, which renders the fatty acid ready for codking. The model is then exported by clicking `Ligand>Output>Save`.

3.4.3 Grid preparation

One requirement for running `AutoDock Vina` is knowing the “pocket” which the ligand molecule interacts with. If this is un

3.5 Lipid-protein-RNA

4 Results

4.1 MSI-1's RRM1-orig docking model

The first docking simulation performed with `LightDock` corresponded to the RRM1-orig RNA-motif complex. The simulation yielded several models out of which 10 were visualized, as shown in **Figure 12**:

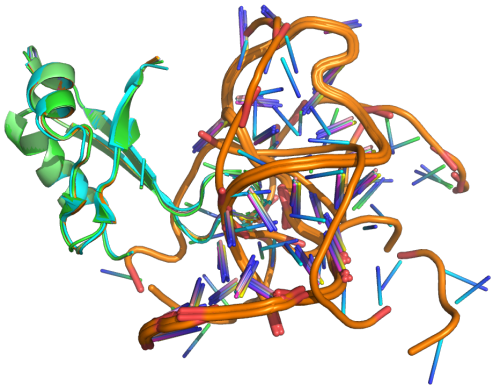

Figure 12: Top 10 scoring models for RRM1-orig RNA-motif complex. Visualized through `Pymol`.

It is observed that there is no consensus among the top 10 scoring models, as it seems that the RNA-motifs adopts different conformations. To be more precise, in **Figure 12** there are 3 different RNA conformations. Their separate conformations are shown in **Figure 13**:

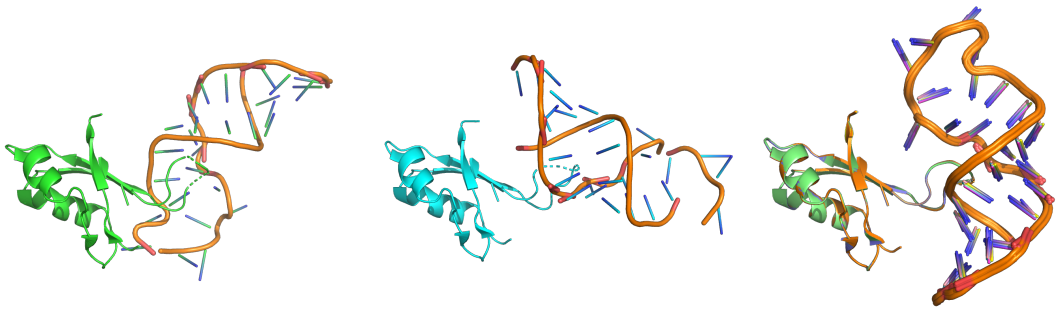

Figure 13: Top 10 scoring models for RRM1-orig RNA-motif complex grouped by RNA conformations. From left to right: best scoring model, second best scoring model and top 3 to 10 best scoring models. Visualized through `Pymol`.

5 Conclusions and further work

6 Glossary

MSI-1

RBP

RRM

NMR

7 References

[1] R. Dolcemascolo, M. Heras-Hernández, L. Goiriz, R. Montagud-Martínez, A. Requena-Menéndez, R. Márquez-Costa, R. Ruiz, A. Pérez-Ràfols, R. A. Higuera-Rodríguez, G. Pérez-Ropero, W. F. Vranken, T. Martelli, W. Kaiser, J. Buijs, and G. Rodrigo, "Repurposing the mammalian rna-binding protein musashi-1 as an allosteric translation repressor in bacteria," *bioRxiv*, 2022.

[2] M. Nakamura, H. Okano, J. A. Blendy, and C. Montell, "Musashi, a neural rna-binding protein required for drosophila adult external sensory organ development," *Neuron*, vol. 13, no. 1, pp. 67–81, 1994.

[3] S. Sakakibara, T. Imai, K. Hamaguchi, M. Okabe, J. Aruga, K. Nakajima, D. Yasutomi, T. Nagata, Y. Kurihara, S. Uesugi, T. Miyata, M. Ogawa, K. Mikoshiba, and H. Okano, "Mouse-musashi-1, a neural rna-binding protein highly enriched in the mammalian cns stem cell," *Developmental Biology*, vol. 176, no. 2, pp. 230–242, 1996.

[4] P. Good, A. Yoda, S.-i. Sakakibara, A. Yamamoto, T. Imai, H. Sawa, T. Ikeuchi, S. Tsuji, H. Satoh, and H. Okano, "The humanmusashi homolog 1(msi1) gene encoding the homologue of musashi/nrp-1, a neural rna-binding protein putatively expressed in cns stem cells and neural progenitor cells," *Genomics*, vol. 52, no. 3, pp. 382–384, 1998.

[5] T. Imai, A. Tokunaga, T. Yoshida, M. Hashimoto, K. Mikoshiba, G. Weinmaster, M. Nakafuku, and H. Okano, "The neural rna-binding protein musashi1 translationally regulates mammalian numb gene expression by interacting with its mrna," *Molecular and Cellular Biology*, vol. 21, no. 12, pp. 3888–3900, 2001.

[6] A. Cornish-Bowden, "Nomenclature for incompletely specified bases in nucleic acid sequences: Rcommendations 1984," *Nucleic Acids Research*, vol. 13, no. 9, pp. 3021–3030, 1985.

[7] N. R. Zearfoss, L. M. Deveau, C. C. Clingman, E. Schmidt, E. S. Johnson, F. Massi, and S. P. Ryder, "A conserved three-nucleotide core motif defines musashi rna binding specificity," *Journal of Biological Chemistry*, vol. 289, no. 51, pp. 35530–35541, 2014.

[8] T. Nagata, R. Kanno, Y. Kurihara, S. Uesugi, T. Imai, S.-i. Sakakibara, H. Okano, and M. Katahira, "Structure, backbone dynamics and interactions with rna of the c-terminal rna-binding domain of a mouse neural rna-binding protein, musashi1," *Journal of Molecular Biology*, vol. 287, no. 2, pp. 315–330, 1999.

[9] Y. Miyanoiri, H. Kobayashi, T. Imai, M. Watanabe, T. Nagata, S. Uesugi, H. Okano, and M. Katahira, "Origin of higher affinity to rna of the n-terminal rna-binding domain than that of the c-terminal one of a mouse neural protein, musashi1, as revealed by comparison of their structures, modes of interaction, surface electrostatic potentials, and backbone dynamics," *Journal of Biological Chemistry*, vol. 278, no. 42, pp. 41309–41315, 2003.

[10] T. Ohyama, T. Nagata, K. Tsuda, N. Kobayashi, T. Imai, H. Okano, T. Yamazaki, and M. Katahira, "Structure of musashi1 in a complex with target rna: The role of aromatic stacking interactions," *Nucleic Acids Research*, vol. 40, no. 7, pp. 3218–3231, 2011.

[11] L. Lan, M. Xing, M. Kashipathy, J. Douglas, P. Gao, K. Battaile, R. Hanzlik, S. Lovell, and L. Xu, "Crystal and solution structures of human oncoprotein musashi-2 n-terminal rna recognition motif 1," *Proteins: Structure, Function, and Bioinformatics*, vol. 88, no. 4, pp. 573–583, 2019.

[12] J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Žídek, A. Potapenko, and et al., "Highly accurate protein structure prediction with alphafold," *Nature*, vol. 596, no. 7873, pp. 583–589, 2021.

[13] B. J. Belmont and J. C. Niles, "Engineering a direct and inducible protein-rna interaction to regulate rna biology," *ACS Chemical Biology*, vol. 5, no. 9, pp. 851–861, 2010.

[14] J. Cao, M. Arha, C. Sudrik, A. Mukherjee, X. Wu, and R. S. Kane, "A universal strategy for regulating mrna translation in prokaryotic and eukaryotic cells," *Nucleic Acids Research*, vol. 43, no. 8, pp. 4353–4362, 2015.

[15] N. Katz, R. Cohen, O. Solomon, B. Kaufmann, O. Atar, Z. Yakhini, S. Goldberg, and R. Amit, "Synthetic 5' utrs can either up- or downregulate expression upon rna-binding protein binding," *Cell Systems*, vol. 9, no. 1, pp. 93–106.e8, 2019.

[16] C. C. Clingman, L. M. Deveau, S. A. Hay, R. M. Genga, S. M. Shandilya, F. Massi, and S. P. Ryder, "Allosteric inhibition of a stem cell rna-binding protein by an intermediary metabolite," *eLife*, vol. 3, p. e02848, 2014.

[17] R. Pereira, M. Couto, F. Ribeiro, R. Rua, J. Cunha, J. P. Fernandes, and J. Saraiva, "Energy efficiency across programming languages: How do energy, time, and memory relate?," *Proceedings of the 10th ACM SIGPLAN International Conference on Software Language Engineering*, 2017.

[18] T. S. Gardner, C. R. Cantor, and J. J. Collins, "Construction of a genetic toggle switch in escherichia coli," *Nature*, vol. 403, no. 6767, pp. 339–342, 2000.

[19] M. B. Elowitz and S. Leibler, "A synthetic oscillatory network of transcriptional regulators," *Nature*, vol. 403, no. 6767, pp. 335–338, 2000.

[20] The iGEM Foundation, "igem registry of standard biological parts." http://parts.igem.org/Main_Page, 2022.

[21] V. Siciliano, I. Garzilli, C. Fracassi, S. Criscuolo, S. Ventre, and D. di Bernardo, "Mirnas confer phenotypic robustness to gene networks by suppressing biological noise," *Nature Communications*, vol. 4, no. 1, p. 2364, 2013.

[22] H. Seok, H. Lee, E.-S. Jang, and S. W. Chi, "Evaluation and control of mirna-like off-target repression for rna interference," *Cellular and Molecular Life Sciences*, vol. 75, no. 5, pp. 797–814, 2017.

[23] P. Babitzke, C. S. Baker, and T. Romeo, "Regulation of translation initiation by rna binding proteins," *Annual Review of Microbiology*, vol. 63, no. 1, pp. 27–44, 2009.

[24] J. Lykke-Andersen, "Rna-protein interactions of an archaeal homotetrameric splicing endoribonuclease with an exceptional evolutionary history," *The EMBO Journal*, vol. 16, no. 20, pp. 6290–6300, 1997.

[25] D. E. Draper and L. P. Reynaldo, "Rna binding strategies of ribosomal proteins," *Nucleic Acids Research*, vol. 27, no. 2, pp. 381–388, 1999.

[26] E. Holmqvist and J. Vogel, "Rna-binding proteins in bacteria," *Nature Reviews Microbiology*, vol. 16, no. 10, pp. 601–615, 2018.

[27] E. V. Koonin, M. Krupovic, S. Ishino, and Y. Ishino, "The replication machinery of luca: Common origin of dna replication and transcription," *BMC Biology*, vol. 18, no. 1, p. 61, 2020.

[28] A. Re, T. Joshi, E. Kulberkyte, Q. Morris, and C. T. Workman, "Rna–protein interactions: An overview," *Methods in Molecular Biology*, pp. 491–521, 2013.

[29] M. Corley, M. C. Burns, and G. W. Yeo, "How rna-binding proteins interact with rna: Molecules and mechanisms," *Molecular Cell*, vol. 78, no. 1, pp. 9–29, 2020.

[30] R. Stefl, L. Skrisovska, and F. H.-T. Allain, "Rna sequence- and shape-dependent recognition by proteins in the ribonucleoprotein particle," *EMBO reports*, vol. 6, no. 1, pp. 33–38, 2005.

[31] J. SantaLucia, "A unified view of polymer, dumbbell, and oligonucleotide dna nearest-neighbor thermodynamics," *Proceedings of the National Academy of Sciences*, vol. 95, no. 4, pp. 1460–1465, 1998.

[32] R. M. Dirks and N. A. Pierce, "An algorithm for computing nucleic acid base-pairing probabilities including pseudoknots," *Journal of Computational Chemistry*, vol. 25, no. 10, pp. 1295–1304, 2004.

[33] Theoretical Biochemistry Group, Universität Wien, "Representations of rna secondary structures." https://www.tbi.univie.ac.at/RNA/ViennaRNA/doc/html/rna_structure_notations.html, 2020.

[34] K. Katoh, K. Misawa, K. Kuma, and T. Miyata, "Mafft: A novel method for rapid multiple sequence alignment based on fast fourier transform," *Nucleic Acids Research*, vol. 30, no. 14, pp. 3059–3066, 2002.

[35] M. Biesiada, K. Purzycka, M. Szachniuk, J. Blazewicz, and R. Adamiak, "Automated rna 3d structure prediction with rnacomposer," *RNA Structure Determination: Methods and Protocols*, pp. 199–215, 2016.

[36] B. Jiménez-García, J. Roel-Touris, M. Romero-Durana, M. Vidal, D. Jiménez-González, and J. Fernández-Recio, "Lightdock: A new multi-scale approach to protein-protein docking," *Bioinformatics*, vol. 34, no. 1, pp. 49–55, 2017.

[37] J. Rodrigues, J. Teixeira, M. Trellet, and A. Bonvin, "pdb-tools: a swiss army knife for molecular structures," *F1000Research*, vol. 7, no. 1961, 2018.

[38] Amber biomolecular simulation package, "Amber histidine residues." http://ambermd.org/Questions/HIS.html, 2005.

8 Appendix

Appendix A: RNA motifs employed in this work, as seen in [1] (in `.fasta` format)

```
>orig
GGCAGCGUUAGUUAUUUAGUUCGUAUGCC
>mut1
GGCAGCGUUCGUUAUUUAGUUCGUAUGCC
>mut2
GGCAGCGUUACUUAUUUAGUUCGUAUGCC
>mut3
GGCAGCGUUAGCUAUUUAGUUCGUAUGCC
>mut4
GGCAGCGUUAGUUAUGUUAGUUCGUAUGCC
>mut5
GGCAGCGUUACUUAUUUACUUCGUAUGCC
```

Appendix B: `Python` script to treat the RNA motifs

```python
import os
import datetime
import Bio.SeqIO
import pandas as pd
import nupack as npk
from hashlib import sha256
from subprocess import Popen, PIPE

# Global vars
modelRNA = npk.Model(
    material="rna95-nupack3",
    ensemble="some-nupack3",
    celsius=37,
    sodium=1.0,
    magnesium=0.0
)

# Functions

def alignment(reference:str, case:str) -> str:

    # Absolutely unique name
    name = (
        sha256(bytes(str(datetime.datetime.now()), "UTF-8")).
    hexdigest()
        + ".fasta"
    )
```

```
27
28      with open(name, "w") as file:
29          file.write(
30              f">reference\n{reference}\n>case\n{case}\n"
31          )
32
33      process = Popen(
34          ["mafft", "--clustalout", name],
35          stdout=PIPE, stderr=PIPE
36      )
37
38      # Should never fail, so I won't check the stderr
39      stdout = process.communicate()[0]
40
41      os.remove(name)
42
43      alignInfo = stdout.decode("UTF-8").split("\n")
44
45      return alignInfo[5][-max(len(reference), len(case)):]
46
47  def folding(seq:str) -> str:
48
49      mfe_structures = npk.mfe(
50          strands=[seq],
51          model=modelRNA
52      )
53
54      return str(mfe_structures[0].structure)
55
56  def get_data_RNAcomposer(df:pd.DataFrame) -> None:
57
58      series = df.apply(
59          lambda row:
60          "\n".join([
61              ">" + row["name"],
62              row["seq"],
63              row["structure"]
64          ]),
65          axis=1
66      )
67
68      for el in series:
69          print(el)
70
71      return None
72
73
74  def main():
75
```

```
    # Read sequences into data frame
    seqs = pd.DataFrame([
        (record.id, str(record.seq))
        for record in
        Bio.SeqIO.parse("../data/motifs.fasta", "fasta")
    ])

    seqs.columns = ("name", "seq")

    seqs["alignment"] = seqs["seq"].apply(
        lambda seq: alignment(seqs.loc[0, "seq"], seq)
    )

    seqs["structure"] = seqs["seq"].apply(
        lambda seq: folding(seq)
    )

    return seqs



if __name__ == "__main__":
    s = main()
    print(s)
```

Appendix C: `Python` script to relabel `HIS` residues for their corresponding `AMBER94` compatible residues

```
import sys
from lightdock.scoring.dna.data.amber import atoms_per_residue

"""http://ambermd.org/Questions/HIS.html

AMBER Histidine residues

Histidine (HIS in normal pdb files) is really one of three
   possible residues:
HID: Histidine with hydrogen on the delta nitrogen

HIE: Histidine with hydrogen on the epsilon nitrogen

HIP: Histidine with hydrogens on both nitrogens; this is
   positively charged.

It is up to the user to inspect the environment of each histidine
   and identify the type that is appropriate.
"""

```

```
18
19 hists = {
20     k:sorted([atom for atom in v if atom.startswith("H")])
21     for k,v in atoms_per_residue.items()
22     if k.startswith("HI")
23 }
24
25 content = ""
26
27 with open(sys.argv[1]) as file:
28
29     isHis = False
30     HisH = []
31     histinfo = ""
32
33     for line in file:
34         elems = line.split()
35
36         if elems[0] == "ATOM":
37
38             if elems[3] == "HIS":
39
40                 isHis = True
41                 histinfo += line
42
43                 if elems[-2] == "H":
44                     HisH.append(elems[2])
45
46                 continue
47
48             elif isHis:
49                 isHis = False
50                 key = [k for k,v in hists.items() if v == sorted(
   HisH)][0]
51
52                 content += (
53                     histinfo.replace("HIS", key)
54                     + line
55                 )
56
57                 HisH = []
58                 histinfo = ""
59                 continue
60
61             else:
62                 content += line
63                 continue
64
65         content += line
```

```

print(content)
```

Appendix D: Python script to remove 3' and 5' hydroxile groups in the RNA structures

```
import sys

with open(sys.argv[1]) as file:
    filelines = filter(
        lambda line: (
            (line.find("HO5'") < 0)
            and (line.find("HO3'") < 0)
        ),
        file.readlines()
    )

print("".join(filelines))
```

Appendix E: Python script to rename and remove incompatible atom types from the RNA srtructure

```
#!/usr/bin/env python3

# NOTE BY LUCAS (NOVEMBER 2022):
# use this on .pdb files corresponding to RNA structures only!

"""
ATOM      23  H5'   DG B   1      -15.347   -3.940   -5.934   1.00   0.00
               H    new
ATOM      24 H5''   DG B   1      -15.852   -4.851   -7.094   1.00   0.00
               H    new
ATOM      25  H4'   DG B   1      -15.214   -3.222   -8.238   1.00   0.00
               H    new
ATOM      26  H3'   DG B   1      -13.483   -5.075   -8.732   1.00   0.00
               H    new
ATOM      27  H2'   DG B   1      -11.785   -4.224   -7.287   1.00   0.00
               H    new
ATOM      28 H2''   DG B   1      -11.314   -3.732   -8.690   1.00   0.00
               H    new
ATOM      29  H1'   DG B   1      -12.430   -1.717   -8.491   1.00   0.00
               H    new
ATOM      30  H8    DG B   1      -10.897   -3.080   -5.447   1.00   0.00
               H    new
ATOM      31  H1    DG B   1      -10.070    2.951   -6.160   1.00   0.00
               H    new
ATOM      32  H21   DG B   1      -12.087    2.806   -8.716   1.00   0.00
               H    new
```

```
ATOM      33   H22   DG B    1      -11.235    3.696   -7.879   1.00   0.00
                H    new
"""

import os
import argparse
from lightdock.scoring.dna.data.amber import atoms_per_residue
from lightdock.pdbutil.PDBIO import read_atom_line


def _format_atom_name(atom_name):
    """Format ATOM name with correct padding"""
    if len(atom_name) == 4:
        return atom_name
    else:
        return " %s" % atom_name


def write_atom_line(atom, output):
    """Writes a PDB file format line to output."""
    if atom.__class__.__name__ == "HetAtom":
        atom_type = "HETATM"
    else:
        atom_type = "ATOM  "
    line = "%6s%5d %-4s%-1s%3s%2s%4d%1s   %8.3f%8.3f%8.3f%6.2f%6.2f%12s\n" % (
        atom_type,
        atom.number,
        _format_atom_name(atom.name),
        atom.alternative,
        atom.residue_name,
        atom.chain_id,
        atom.residue_number,
        atom.residue_insertion,
        atom.x,
        atom.y,
        atom.z,
        atom.occupancy,
        atom.b_factor,
        atom.element,
    )
    output.write(line)


translation = {
    "H5'": "H5'1",
    "H5''": "H5'2",
    "H2'": "H2'1",
    "H2''": "H2'2",
```

```
    "HO2'": "HO'2", # These ones were added by Lucas in November 2022
    "HO3'": "HO'3", #
    "HO5'": "HO'5"  #
}


if __name__ == "__main__":

    parser = argparse.ArgumentParser()
    parser.add_argument("input_pdb_file")
    parser.add_argument("output_pdb_file")
    args = parser.parse_args()

    with open(args.input_pdb_file) as ih:
        with open(args.output_pdb_file, 'w') as oh:
            for line in ih:
                line = line.rstrip(os.linesep)
                if line.startswith("ATOM  "):
                    atom = read_atom_line(line)

                    # The following lines were added by Lucas in November 2022 #
                    if (
                #
                        not(atom.residue_name.startswith("R"))
                #
                        and not(atom.residue_name.startswith("D"))
                #
                        ):
                #
                        atom.residue_name = "R" + atom.residue_name            #
                #
    ############################################################
                    if atom.residue_name not in atoms_per_residue:
                        print(f"Not supported atom: {atom.residue_name}.{atom.name}")
                    else:
                        if atom.name not in atoms_per_residue[atom.residue_name] and atom.is_hydrogen():
                            try:
                                atom.name = translation[atom.name]
                                write_atom_line(atom, oh)
                            except KeyError:
                                print(f"Atom not found in mapping: {atom.residue_name}.{atom.name}")
                        else:
                            write_atom_line(atom, oh)
```

Appendix F: Python script to remove the incompatible R tags from the RNA structures

```python
import sys

def reemplazar(line):
    line = list(line)
    line[line.index("R")] = " "
    return "".join(line)

with open(sys.argv[1]) as file:
    lines = file.readlines()


S = list(map(
    reemplazar,
    lines
))

with open(sys.argv[1], "w") as file:
    file.write("".join(S))
```

Appendix G: Python script to recover the R tags from the RNA structures

```python
import sys

with open(sys.argv[1]) as file:

    filelines = file.readlines()


nufilelines = list(map(
    lambda line: (
        line
        .replace("   A A", "  RA A")
        .replace("   U A", "  RU A")
        .replace("   G A", "  RG A")
        .replace("   C A", "  RC A")
        .replace("   A B", "  RA B")
        .replace("   U B", "  RU B")
        .replace("   G B", "  RG B")
        .replace("   C B", "  RC B")
    ),
    filelines
))

with open(sys.argv[1], "w") as file:
```

```
    file.write("".join(nufilelines))
```

Appendix H: `Bash` script to cluster and rank the models generated by `LightDock`

```
## Calculate the number of swarms
s=‘ls -d ./swarm_* | wc -l‘
swarms=$((s-1))

## Create files for Ant-Thony
for i in $(seq 0 $swarms)
  do
    echo "cd swarm_${i}; lgd_generate_conformations.py ../protein.pdb ../rna.pdb gso_100.out 200 > /dev/null 2> /dev/null;" >> generate_lightdock.list;
  done

for i in $(seq 0 $swarms)
  do
    echo "cd swarm_${i}; lgd_cluster_bsas.py gso_100.out > /dev/null 2> /dev/null;" >> cluster_lightdock.list;
  done

### Generate LightDock models
ant_thony.py -c 4 generate_lightdock.list;

### Clustering BSAS (rmsd) within swarm
ant_thony.py -c 4 cluster_lightdock.list;

### Generate ranking files for filtering
lgd_rank.py $s 100;
```

Appendix I: `Bash` script to run a complete protein-RNA docking simulation with `LightDock`

```
# Parse inputs
proteinPDBpath=$1
ligandPDBpath=$2

IFS='/' read -r -a tmp <<< "$proteinPDBpath"
IFS='.' read -r -a proteinPDBfilename <<< "${tmp[${#tmp[@]}-1]}"

IFS='/' read -r -a tmp <<< "$ligandPDBpath"
IFS='.' read -r -a ligandPDBfilename <<< "${tmp[${#tmp[@]}-1]}"

echo ">>>Inputs used are: $proteinPDBfilename and $ligandPDBfilename"

```

```
# Protonation
## Case 1: protein.

echo ">>>CASE 1: protein."
echo ">>>Remove the previous hydrogens and them rebuild them according to reduce"

reduce -Trim $proteinPDBpath > "${proteinPDBfilename}_noh.pdb"
reduce -BUILD "${proteinPDBfilename}_noh.pdb" > "${proteinPDBfilename}_h.pdb"

echo ">>>Renumber the atoms of the protein receptor partner using PDB-Tools"

pdb_reatom "${proteinPDBfilename}_h.pdb" > protein_pre.pdb

echo ">>>Fix HIS residues so that they work with AMBER force field"

python3.9 ../../../fixHIS.py protein_pre.pdb > protein.pdb

## Case 2: RNA motif.

echo ">>>CASE 2: RNA ligand."
echo ">>>Remove the previous hydrogens and them rebuild them according to reduce"

reduce -Trim $ligandPDBpath > "${ligandPDBfilename}_noh.pdb"
reduce -BUILD "${ligandPDBfilename}_noh.pdb" > "${ligandPDBfilename}_h.pdb"

echo ">>>Remove possible top overhang OH"

python3.9 ../../../removeEnds.py "${ligandPDBfilename}_h.pdb" > "${ligandPDBfilename}_nvh.pdb"

echo ">>>Rename and/or remove incompatible atom types of RNA"

python3.9 ../../../reduce_to_amber.py "${ligandPDBfilename}_nvh.pdb" rna.pdb

python3.9 ../../../manipulateRNA.py rna.pdb

# Setup

echo ">>>Make lightdock setup"

# Without restraints
lightdock3_setup.py protein.pdb rna.pdb -anm
```

```
54
55
56 # Simulation
57 echo ">>>Recover RNA tags in lightdock_rna.pdb"
58
59 python3.9 ../../../manipulateRNAagain.py lightdock_rna.pdb
60
61 echo ">>>Make lightdock simulation"
62
63 lightdock3.py setup.json 100 -s dna -c 4
64
65 # Clustering
66 echo ">>>Do clustering and filtering"
67
68 python3.9 ../../../manipulateRNA.py lightdock_rna.pdb
69
70 ## Calculate the number of swarms
71 s=`ls -d ./swarm_* | wc -l`
72 swarms=$((s-1))
73
74 ## Create files for Ant-Thony
75 for i in $(seq 0 $swarms)
76   do
77     echo "cd swarm_${i}; lgd_generate_conformations.py ../protein.
    pdb ../rna.pdb gso_100.out 200 > /dev/null 2> /dev/null;" >>
    generate_lightdock.list;
78   done
79
80 for i in $(seq 0 $swarms)
81   do
82     echo "cd swarm_${i}; lgd_cluster_bsas.py gso_100.out > /dev/
    null 2> /dev/null;" >> cluster_lightdock.list;
83   done
84
85 ### Generate LightDock models
86 ant_thony.py -c 4 generate_lightdock.list;
87
88 ### Clustering BSAS (rmsd) within swarm
89 ant_thony.py -c 4 cluster_lightdock.list;
90
91 ### Generate ranking files for filtering
92 lgd_rank.py $s 100;
```