Comparison of RNA-folding structures, *in-vivo*, *in-vitro* and *in-silico*

Hanieh Roodashty

September 2019

Master Thesis

Department of Informatics
University of Bergen
Norway

To my
husband,
daughters,
parents and
sisters.

Acknowledgements

I would like to sincerely thank my main supervisor, Dr. Eivind Valen for providing me with the opportunity to work in the interesting field of RNA structure probing and for the guidance throughout this master thesis. I am also to my co-supervisor Håkon Tjeldnes for invaluable technical and hands-on advices in the course of the thesis.

I also would like to thank Pål Magnus Gunnestad for all the administrative support he has been willingly offered.

Last but not least I would like to thank my lovely husband Mahdi and my cute daughters Rihanna and Nora for being patient while I was deeply coding day and night. I am also grateful to my family members back home, my lovely parents and sisters for always being there for me.

Hanieh Roodashty

Bergen, September 2019

Summary

This work was performed at the Department of Informatics, University of Bergen between August 2018 and September 2019 with Professor Eivind Valen as supervisor and Håkon Tjeldnes as co-supervisor.

The main objective of the work has been to develop the understanding of the difference between RNA structure probing approaches; *in vivo*, *in vitro* and *in silico*. Similarities and differences between three libraries of data from each of the probing approaches were then studied.

The difference between the results was obtained by coverage-plotting of the three datasets and eventually, match and mismatch status of the RNA structural data for the three approaches were tabulated.

It was shown that the *in vitro* and *in silico* data match for the leader region whereas *in vivo* data mismatch with the *in silico* data for the same region.

Benchmarking the experimental data with computational data, CDS region data for both *in vitro* and *in vivo* show an appropriate match with *in silico* data.

For the trailer region, neither *in vitro* nor *in vivo* datasets did match with *in silico* data.

When *in vivo* and *in vitro* data were compared, it was observed that CDS and trailer regions match while their leader regions mismatch.

In this thesis R coding was used to analyze the data. The source codes are available in the appendix of the thesis as well as an online software development hosting service (GitHub) [1].

Table of Contents

List of Figures

List of Tables

Symbols and Nomenclature

Roman letters

A	Adenine
C	Cytosine
G	Guanine
R	Pearson's correlation coefficient
U	Uracil

Abbreviations

BWT	Burrows-Wheeler transform
CAGE	Cap analysis of gene expression
cDNA	Complementary DNA
CDS	Coding sequence
DMS	Dimethyl sulfate
DNA	Deoxyribonucleic acid
FAI	2-methyl-3-furoic acid imidazolide
FM	Ferragina-Manzini
HISAT	Hierarchical indexing for spliced alignment of transcripts
HTSeq	High-throughput sequencing
MFE	Minimum free energy
mRNA	Messenger ribonucleic acid
NAI	2-methylnicotinic acid imidazolide
ncRNA	Non-coding RNA
NGS	Next generation sequencing
NMIA	N-methylisatoic anhydride

NMR	Nuclear magnetic resonance
NPIA	N-propanone isatoic anhydride
RNA	Ribonucleic acid
rRNA	Ribosome ribonucleic acid
RT	Reverse transcription
SAM	Sequence Alignment/Map
SHAPE	Selective 2’-hydroxyl acylation analyzed by primer extension
TIS	Translation initiation sites
tRNA	Transfer ribonucleic acid
UTR	Untranslated region
1M6	1-methyl-6-nitroisatoic anhydride
1M7	1-methyl-7-nitroisatoic anhydride

Chapter 1

Introduction

An introduction to the master thesis work is provided in this chapter. The chapter begins with an explanation of the background of the work and then describes the drive and motivation for understanding significant differences in RNA secondary structure *in vivo* and *in vitro* data compared to *in silico* data.

1.1 Background

The central dogma of molecular biology is a description of the transfer of genetic information from DNA to RNA and then to protein within biological systems. This concept was first defined by Francis Crick in 1957 [2].

Crick stated that as soon as information have transferred from DNA to RNA (transcription), there will be a route of reverse information from RNA to DNA (reverse transcription process), but when sequential information finally is transferred to protein (translation) there is no reverse process (reverse translation) [2]. Figure 1-1 schematically presents this concept.

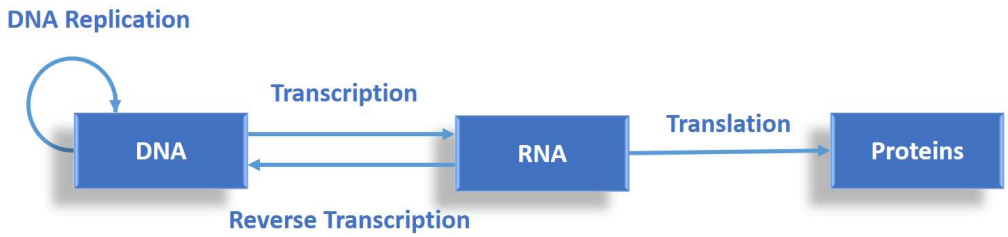

Figure 1-1: The transfer routes in the biological systems

All subjects of biology and microbiology rely on the Crick's notion which makes a framework for understanding the transfer process of genetic information within living organisms.

In genetics, gene expression is the most important process that occurs in all living organisms i.e. eukaryotes (animals and plants), prokaryotes (bacteria and archaea) and viruses. In this process, the genetic information stored in DNA are converted into a functional gene product. The product of a gene is usually a protein, but sometimes it can also be a functional RNA.

There are three key steps in the gene expression process, including *transcription*, *transportation* and *translation*. Transcription is the process of transferring data from DNA to RNA, transportation is RNA transferring from the nucleus into cytoplasm and translation here, is the process of the synthesis of protein out of RNA.

The RNAs are intermediate molecules in gene expression and have different important roles in each step of this cellular process [3]. In the first step, the DNA is transcribed into a type of RNA called messenger RNA (mRNA), which is a copy of the gene's DNA except for having a single strand. In the next step, the mRNA is transported from the nucleus into the cytoplasm and gets attached to a ribosome, where the biological protein synthesis (translation) is carried out.

The RNA in the ribosome is called ribosomal RNA (rRNA) that synthesizes proteins according to the instructions stored in the sequence of mRNA. Eventually in the last step, a type of RNA called transfer RNA (tRNA) transfers corresponding amino acids to the mRNA during protein synthesis [3].

Commonly, those RNA molecules translated into proteins are called messenger RNAs and those that do not encode proteins are referred to as non-coding RNA (ncRNA) [4]. Abundant and functionally important non-coding RNAs include tRNAs and ribosomal RNAs, as well as small RNAs such as miroRNAs and snRNAs.

In addition to typical roles of RNA in protein synthesis, RNA molecules can have many active roles in other cell processes including roles in catalysis, cellular defense and regulation [5]. The structure of RNAs plays an important role in the properties and thus

function of them in cellular processes. It is then crucial to study RNA structures and their effects on the biological roles of RNAs in living cells [6].

In this thesis we looked at regulation of RNA and particularly that of RNA structure.

1.1.1 Problem statement

It is presented that there are several approaches (*in vivo*, *in vitro* and *in silico*) to probe and assess RNA structures. We shall try to demonstrate some of the shortcomings associated with these approaches and examine that to what extent, these approaches agree upon the structural predictions for a zebrafish RNA sample.

1.1.2 Thesis structure

The thesis consists of four chapters, a bibliography and an appendix.

In Chapter 1, relevant concepts of RNA folding and RNA structure together with brief information about *in vivo*, *in vitro* and *in silico* studies are introduced. At the end of this chapter, zebrafish data and their importance for biological studies were discussed.

In Chapter 2 the software and tools used in the master thesis and also the protocol for analyzing *in vivo*, *in vitro* and *in silico* data are presented.

Chapter 3 comprises the results and relevant discussions.

The conclusion and recommendations for future work are reflected in Chapter 4.

Finally, the source code written for the thesis following the protocol in Chapter 2 is presented as an appendix.

The figures in the thesis have been made by the author of the master thesis and are copyrighted.

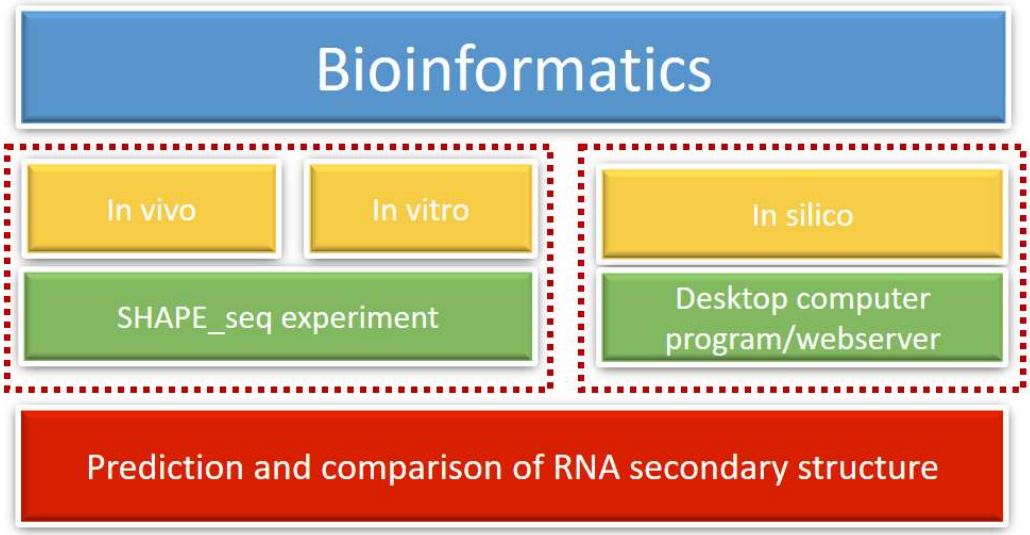

Figure 1-2: Overview of the hierarchy and the project concept

Figure 1-2 shows an overview of the important subjects connected to bioinformatics from one end and the content of this thesis from the other end. We have obtained the *in vivo* and *in vitro* data from SHAPE_seq experiment and *in silico* data via computational approaches. The outcome was then used to predict and compare RNA secondary structures.

1.2 **RNA Structure**

Ribonucleic acids (RNA) are linear polymeric molecules built from 4 possible monomers: Adenine (A), Guanine (G), Uracil (U) and Cytosine (C). These monomers can form complementary interactions to build base-pairings: A with U and C with G.

Unlike DNA which is double-stranded, RNA is most often found as a single-stranded nucleic acid molecule that often contains complementary regions to form double helices when it can fold onto itself [7].

These foldings create a structure that can be divided into three major levels of organization: primary, secondary, and tertiary structure [8].

The primary structure refers to the nucleotide sequence of an RNA, which can be obtained from the DNA sequence of the gene encoding the RNA.

The secondary structure of RNA is a two-dimensional representation of its Watson-Crick C.G and A.U pairs as well as the weaker G.U wobble pairs.

The tertiary structure refers to a three-dimensional representation of the RNA structure that is emerged from a secondary structure. The elements of this structure involve an interaction between two or more secondary structural elements.

In this thesis we have focused on the secondary structure of RNA as it is straightforward to predict and measure its structure.

All other combinations of pairing nucleotides, called non-canonical pairs, are ignored in the secondary structure prediction, although they do occur especially in tertiary structure motifs [9]. RNA secondary structure contains different structural forms, including single-stranded regions, stem, internal loops, hairpin loops, bulge loops, junctions (multibranch loop) and pseudoknot [10]. The single-stranded regions are formed where no base pairs are found.

The double-stranded secondary structure is defined as a stem. The internal loops in RNA are formed where the double-stranded separates because of no base pairing between the nucleotides.
The most common structure that RNA molecules fold into is called hairpin loop that is formed at the end of a stem (see Figure 1-3). Bulge loop occurs when there are one or more unpaired nucleotides placed in one of the strands of the double-stranded (see Figure 1-3).

Another structural form of RNA is known as a multibranch loop. It is a more complex that is formed when more than two stems intersect. They are major elements of tertiary structure of RNA. One important structural motif of RNA secondary structures is a pseudoknot which occurs when two hairpins or bulges are connected through single-stranded regions. This structural form is very difficult to predict thus most of the RNA secondary structure prediction methods do not consider pseudoknots in the structure of RNA. However, the number of pseudoknots in the RNA structure is small but often take place in highly structured and functional RNAs. Figure 1-3 schematically presents an RNA secondary structure.

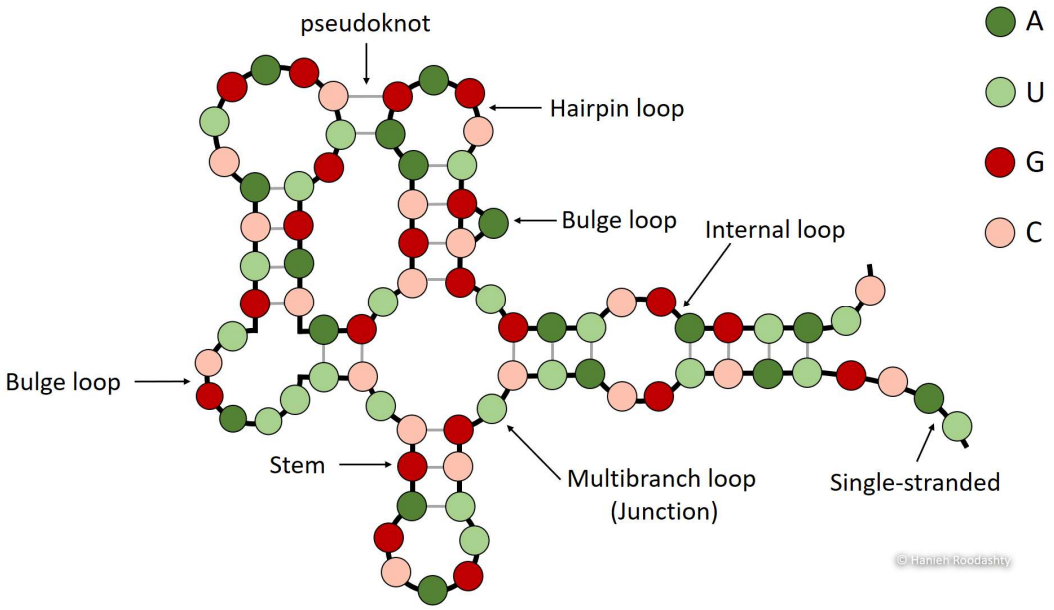

Figure 1-3: RNA secondary structure

RNA molecules perform several cellular functions ranging from translation of genetic information to regulation of the genes activity.

These biological functions of RNAs highly depend on their secondary structures [6] thus, for understanding both the function of the RNA molecules and the mechanism behind that function, the study of RNA secondary structure is necessary. Additionally, predicting RNA secondary can be useful for the interpretation of experiments linked to the mechanism of RNA function and to propose new experiments to probe biological functions[11][12][13].

Because RNA folding process is generally hierarchical, the RNA secondary structure can be predicted without any knowledge of tertiary structure[14] which are typically predicted by complicated and expensive experiments.

The stability of the RNA secondary structure is an important factor to predict and explain the function of the RNA in the cell.
The stability of the secondary structure is calculated based on the amount of free energy released by the forming base pairs in RNA folding process. The number of base pairs specially G.C base pairs which release more free energy (i.e. more negative free energy) when forming, increase the stability of the hairpin loop region. On the other hand, the number of unpaired bases decreases the stability of the interior loops, hairpin loops and bulge loops. Thus, a structure with minimum free energy, is more stable and considered as an optimal structure (most structured). This structure is called the MFE secondary structure[15][16].

There are many applications that compute the free energy of RNA secondary structure to predict the RNA folding structure.

1.3 **Predicting RNA folding structure**

To completely understand the various biological functions of RNAs, including both coding RNAs and non-coding RNAs (ncRNAs), we need to first characterize and predict RNA structures.

These functions are initially encoded in the primary structure [17]. Then the base pairs in the RNA secondary structure that are created by interactions such as hydrogen bond, will be identified by predicting RNA secondary structure methods.

Predicting RNA secondary structure approaches can be classified into physical, computational and experimental methods.

Physical methods use physical properties of RNAs structure for the investigation of the structure. The most powerful tools here are X-ray crystallography and nuclear magnetic resonance (NMR)[18]. However, these methods are expensive and complex, and it led researchers to go after more straightforward and cost-effective methods such as predicting RNA structure by computational approaches [9].

Computational approaches can be classified into two categories: thermodynamic approaches and comparative approaches [9].

Thermodynamic approaches use dynamic programming which is a popular method to compute the secondary structure of RNA [9]. This method involves computing the optimal secondary structure of an RNA sequence which is the structure with minimum free energy of the interaction between nucleotides.

For finding the optimal structure, first the minimum free energy is determined for each possible base pair in the shortest sequence fragments and then for longer fragments until the minimum free energy of the complete sequence is calculated [19].

However, there is a major issue in this method for identifying pseudoknots because the dynamic programing considers only the interactions between the closest nucleotides for predicting secondary structure, while pseudoknotted structures are formed by interactions between distant nucleotides[20]. In general, thermodynamic methods have been appropriate for small RNAs.

In contrast comparative approaches are suitable to manually predict the structure of long RNA sequences. This approach is used when several aligned homologous RNA sequences are available and is useful for the cases that several sequences are used. These methods find pairs that covary to keep Watson-Crick and Wobble complementarities[21][22].

The common practice for studying RNA structures by experimental methods is to use chemical probing experiments [23] that provide information regarding RNA structures by measuring access of nucleotide for RNase cleavage or chemical modification[24] ,[25].

These experiments use chemicals as reagents that react with specific nucleotide positions on the RNA structure to covalently modify them. These reagents can be divided into three classes [25]: 1) base-specific reagents such as DMS (dimethyl sulfate), 2) backbone-cleaving reagents such as hydroxyl radicals [26] and 3) SHAPE (selective 2'-hydroxyl acylation analyzed by primer extension) reagents that modify the 2'-OH of the RNA backbone [25] [27].

After chemical modifications, RNAs are converted into cDNAs through reverse transcription (RT), which either stop [25],[28] or introduce a mutation at each modification positions [27]. Finally, the pool of cDNA molecules is sequenced, and modification frequency at each position is determined by using sequence alignment algorithms.

There are modern chemical probing approaches that by using next generation sequencing (NGS), can probe several RNAs, sequence and analyze the cDNA molecules all in a single experiment [29][30][31][32][33][34]. These datasets can be analyzed in several ways and are used as restraints in computational RNA folding algorithms to improve the accuracy of structure prediction algorithms [33].

Although there are many different chemicals that can be used to probe RNA structure, in this thesis, we looked at the SHAPE class of chemical probes. SHAPE is believed to be a very suitable method for characterizing the structure features of large RNAs [35]. This type of experiment improves the accuracy of RNA secondary structure prediction.

1.3.1 SHAPE

These days, many chemical probing techniques have been suggested to guide the computational predictions of RNA structure as an attempt to improve the accuracy of the RNA secondary structure prediction [36][37][38].

A common chemical method widely used nowadays is to use Selective 2'-Hydroxyl Acylation analyzed by Primer Extension sequencing (SHAPE) as reagent in the chemical probing experiments.

An advantage of SHAPE probing is that it provides information for all positions and that it can be used to probe RNA structure[39] both *in vitro* and *in vivo*.

SHAPE reagents modify the 2' OH (Hydroxyl) group of the RNAs backbone. These chemical probes of RNA structure react with structurally flexible nucleotide positions and will covalently modify them[40][41]. Single-stranded or flexible RNA regions has

high 2’ OH reactivity, while nucleotides in base pairing form or other forms have lower reactivity[42].

After chemical modification, the position of these modifications are detected with reverse transcription (RT) which stops [43][44] at one nucleotide before the RNA modifications (+channel) to create the cDNAs (complementary DNAs).

RT process will go on until reaching to the first modified position where it will be stopped. Therefore, each cDNA can only encode one probe position per RNA.

Additionally, a control RT is performed on an unmodified RNA (-channel) to identify locations where early-stage or the 5’ end termination of RT are carried out.

The pool of cDNA molecules of both unmodified and modified RNAs are sequenced, and modification positions are mapped by using sequence alignment algorithms. These datasets can be analyzed in several ways and are used as restrains in computational RNA folding algorithms to improve the accuracy of structure prediction algorithms [24].

Figure 1-4 schematically presents the SHAPE strategy.

These datasets can then be used to estimate the relative modification frequency or reactivity at each nucleotide of RNA[45][46]. The reactivity is the likelihood of the of modification of nucleotide within an RNA by the SHAPE reagent. Mathematically, a division of the number of modified nucleotide by the total number of nucleotide in the RNA will give the reactivity[46][45]. So, the higher the number of the modification of nucleotide, the higher the reactivity and vice versa.

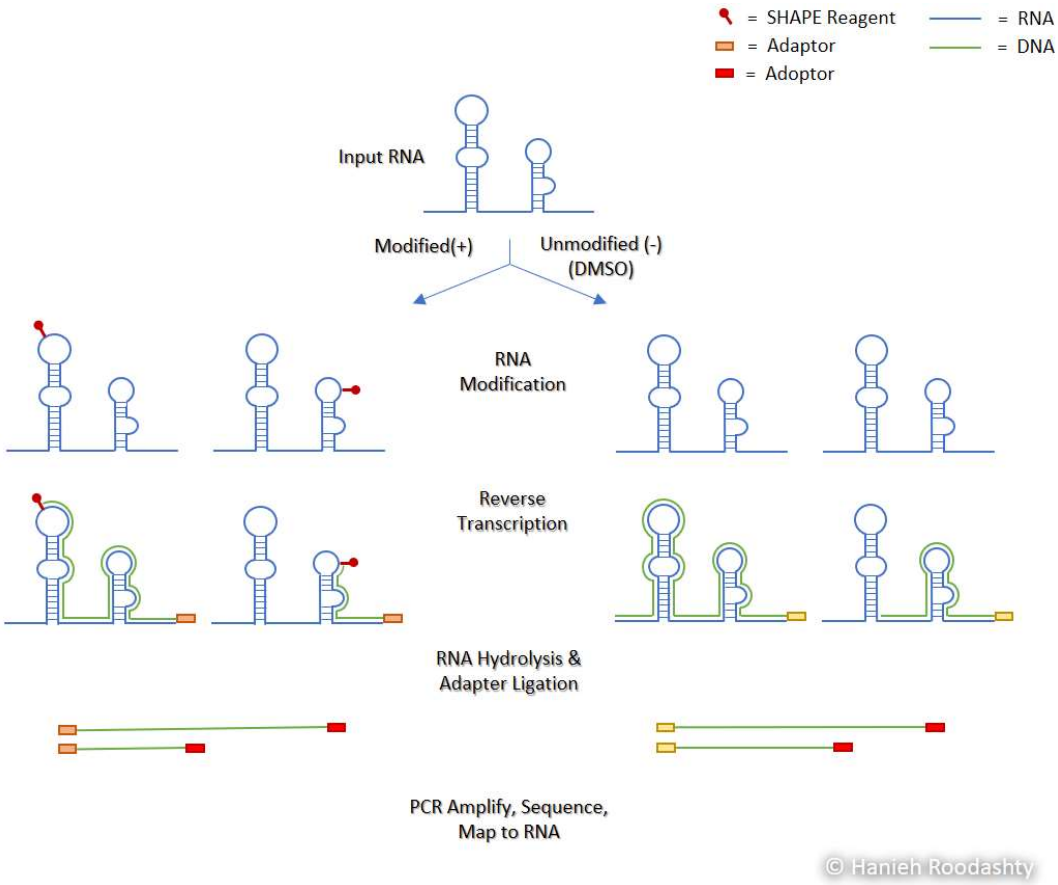

Figure 1-4: SHAPE strategy

High SHAPE reactivities refer to nucleotides that are unstructured (unpaired nucleotides such as loops and bulges), while low reactivities can refer to the nucleotides that have structural inflexibility due to being located in double-stranded helices, bound to a protein (protein interaction) or ligand (ligand binding), tertiary interactions, or other factors that cause reducing nucleotide flexibility[47][48][49]. In addition to this interpretation, SHAPE reactivity data can also be used as restrains in computational RNA folding algorithms to improve the accuracy of structure prediction algorithms [50] [51][52][53][54].

Restraining computational RNA folding algorithms with SHAPE reactivities can be achieved by two approaches: 1) modifying the model parameters in the folding calculation algorithm [51] 2) selecting the structure from a set of generated structures by the folding calculation that highly matches with the SHAPE data [55][56][57].

Two abovementioned approaches perform the same job of calculating a partition function which defines the way that a population of RNA divide into various structures

in equilibrium state, where the occurrence of each structure entails its own probability [58].This partition function can be used for predicting a minimum free energy (MFE) structure which has the maximum probability of occurring in the different structures [58].

The two approaches use SHAPE reactivity data for the folding calculations with different techniques. The first technique converts the SHAPE reactivity data into pseudo-free energy terms that are considered for each nucleotide involved in base stacking in the calculating overall free energies of RNA structure according to the below form:

$$\Delta G_{SHAPE}(\mathrm{i}) = \mathrm{m}\ln[r(i) + 1] + b \qquad (1\text{-}1)$$

Where $\Delta G_{SHAPE}(\mathrm{i})$ is the pseudo-free energy term at nucleotide i, m and b are constant parameters and r(i) is the reactivity at nucleotide i [51].

Finally, the $\Delta G_{SHAPE}(\mathrm{i})$ is included in the calculation of free energy and the MFE structures that are more consistent with the achieved reactivity data are created[51][50].

The second approach to select the best structure from a set of possible structures
first converts SHAPE reactivity data to a binary vector (0,1) and similarly each possible RNA structure is converted into a binary vector (0,1) where 0 and 1 correspond to paired and unpaired bases, respectively. Next step, the distance between the vector of SHAPE reactivity and each possible structure vector is calculated and then a structure that has minimum distance is selected.
Eventually a characteristic centroid structure is calculated by clustering of structures based on the most related minimum distance structure [57].

Recently, some new SHAPE reagents are developed that can be used to global probing of *in vivo* RNA structure such as 1M7 (1-methyl-7-nitroisatoic anhydride) and FAI (2-methyl-3-furoic acid imidazolide). These reagents are based on imidazolide chemistry that can probe RNA structure inside living cells (*in vivo*) [39].
However, *in vivo* SHAPE probing is more difficult and complex than the probing of RNA molecules *in vitro* because, *in vivo* SHAPE reagents encounter the RNA molecules and additionally elements present in the cell.

1.3.2 SHAPES

As mentioned in the SHAPE method, SHAPE reagents react with the 2′ OH group of the ribose of each nucleoside within an RNA to create covalent adducts at flexible regions of RNA structure. This process is called “SHAPE probing”.

After SHAPE probing, probed RNAs are converted to cDNAs by reverse transcriptase that terminate at a position located one nucleotide before SHAPE modifications (adducts).

However, reverse transcriptase can also be stopped at positions of RNA degradation or other structures such as stable RNA structures. Thus, a control RT on a RNA that is not probed with reagents (-channel) is also performed in parallel with a RT on modified RNA. The control data can be used to normalize the SHAPE probing data in order to remove the background signal present in SHAPE probing data caused by natural termination or 5′ end drop-off (run-off).

Performing this strategy to correct the background signal in SHAPE probing data, allows accurate calculation of adduct frequencies for each position in RNA.

Different strategies have been developed for enhancing SHAPE modification and the efficiency of adduct detection, particularly in living cells (*in vivo*) [59][60] [61].

There are some SHAPE reagents such as 1M7 (1-methyl-7-nitroisatoic anhydride), 1M6 (1-methyl-6-nitroisatoic anhydride) and NMIA (N-methylisatoic anhydride) that have little different modification properties [62][54][63] which can be used as probe in chemical probing-sequencing methods inside living cells. Recently two more class of SHAPE reagents have been proposed, consisting NAI (2-methylnicotinic acid imidazolide) and FAI (2-methyl-3-furoic acid imidazolide).

A novel SHAPE Selection (SHAPES) reagent, NPIA (N-propanone isatoic anhydride) have been introduced for enhancing adduct detection [64]. This new reagent has ability to react with RNA and be coupled with biotin. The ability can be helpful for getting only cDNAs that terminate at probed positions by performing a special selection strategy on SHAPE probed RNA[64].

The SHAPES reagent modifies the 2′- OH group of the ribose in each nucleoside of an RNA by NPIA (RNA is probed by NPIA) and afterwards reverse transcription is performed to create cDNA. There are cDNAs that contain structural information, they were created by termination of RT at the NPIA modification and cDNAs that were created by early-stage termination of RT or its termination at the 5′end of the RNA (i.e. background signal). Then the N-propanone group of NPIA is biotinylated with biotin hydrazide. The product is now cDNA/RNA–NPIA–biotin hybrids at the end of this step.

This product will interact with RNase I and cause that all single stranded RNAs be cleaved then this interaction will be followed by performing a CAGE-like (cap analysis of gene expression)[65][66] selection strategy, on streptavidin beads which has high affinity for biotin. This strategy effectively removes the cDNAs caused by premature termination or 5′ end drop off and only cDNAs containing the structural information remain. These are cDNAs with termination at probed positions [64].

The cDNAs with the structural information are sequenced and then the corresponding modification positions are identified. As mentioned for the regular probing method, this dataset can be used to guide computational methods for predicting secondary RNA structure. Figure 1-5 schematically presents the SHAPES strategy.

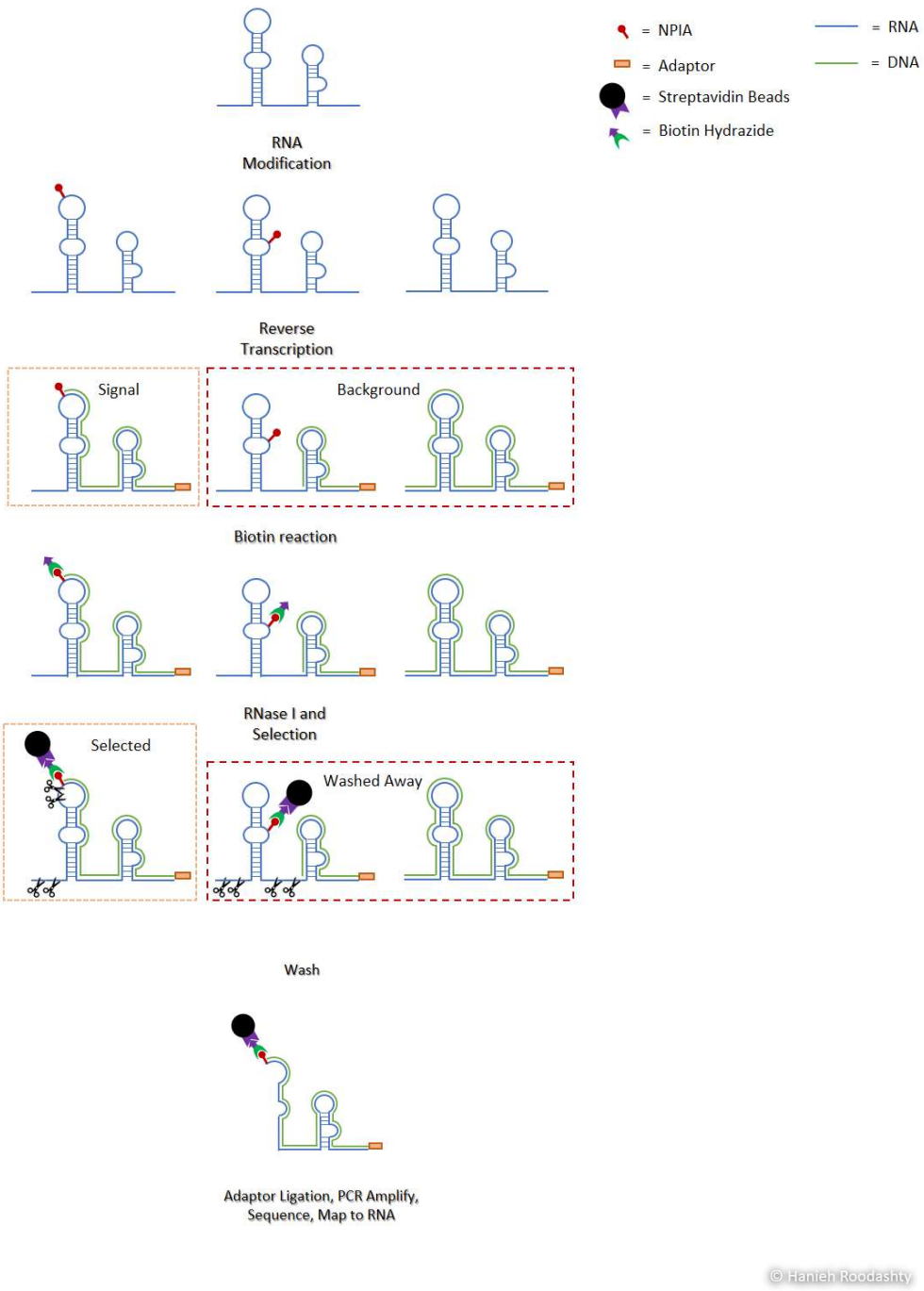

Figure 1-5: SHAPES strategy

As consequence, in SHAPES method, the background signal present in SHAPE probing data, is removed without normalization with a no-reagent control, thus; only one sample

(modified RNA) needs to be sequenced. However, in regular SHAPE methods, a control RT is performed on no-reagent RNA to identify background signal, thus the cost of SHAPE experiment is more than SHAPES method. On the other hand, selection step in SHAPES increases the time needed to perform the experiment but having more data causes to get more accurate result.

Additionally, the SHAPES technology provides an alternative to regular *in vitro* SHAPE probing of RNA structure and will be especially suitable for *in vivo* structure probing which has low probing and high background signal.

1.4 *In vivo*, *in vitro* and *in silico*

The studies of RNA folding structure are categorized into three groups: *in vitro* studies, *in vivo* studies, and *in silico* studies [6].

In vitro (Latin word for "within the glass") studies refers to perform a given procedure in a controlled environment (i.e. in a laboratory environment) outside of a living organism.
Data obtained from *in vitro* experiments may not accurately predict the effects on a whole organism.

In vivo (Latin for "within the living") refers to experimentation using a whole living organism or cells as opposed to a dead organism.
In contrast to *in vitro* experiments, the effects of various biological objects are tested on living organisms or cells.

In silico is an expression used to mean, "performed on computer or via computer simulation" in reference to biological experiments.

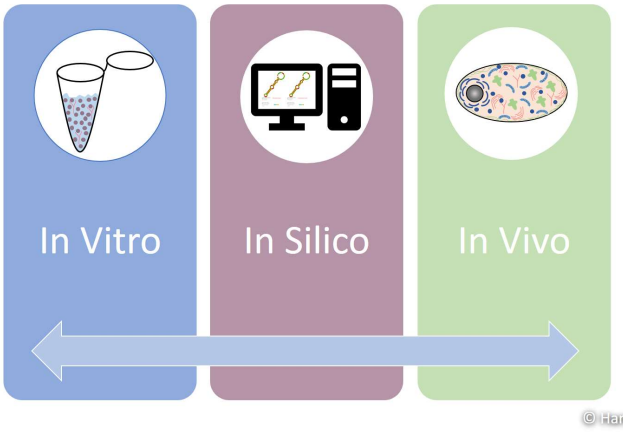

Figure 1-6: Graphical representation of *in vitro*, *in silico* and *in vivo*

There are some advantages and restrictions for probing RNA structure by each of these approaches, and the information of the biological functions of RNAs produced by each of these experiments can be unique [6].

Most of the information of the RNA structure and folding pathway have been discovered by *in vitro* studies and newly some chemical methods have been developed for *in vitro* probing of RNA structure.
In the *in vitro* studies the biological components in cells including proteins, ligands and biological ion compositions are missing *in vitro* experiments and therefore, the laboratory conditions (environment) of this type of experiments are very different from the cellular environment. This appears to be the most important drawback of *in vitro* approach since the mentioned components have influence in the structure and function of RNAs within a cell. Given the limitations of the *in vitro* approach, the development of experiments and techniques for understanding how RNA folds and functions in cellular environmental conditions (*in vivo*) is critical.

In vivo studies can be useful for understanding how the cellular environment affects RNA folding and structure. The RNA structure motifs and interactions between RNA and protein can be verified through *in vivo* experiments. In addition, by some new methods, the probing of RNA structure is possible with *in vivo* conditions. However, studying RNA folding and structure *in vivo* has limitations too. *In vivo* studies provide information about the average RNA structure in a cell and not information on RNA dynamics and RNA folding pathway.

Both *in vivo* and *in vitro* studies of probing RNA structure in genome-wide have shown that the coding regions of RNAs are more structures than the untranslated regions of RNAs[67] [68][69][70][71]. In addition, the start and stop codons have less structures than in the rest of the transcript, which is probably a simplified read-through by the ribosome.

The *in silico* studies in the field of molecular biology were first introduced in the 1980 by Pedro Miramontes [72], a mathematician from National Autonomous University of Mexico (UNAM). He stated that biological experiments can be performed virtually using computers.

The *in vivo* and *in vitro* experiments are used to navigate *in silico* methods to improve the accuracy of *in silico* modeling [73]. For example, the RNA structural probing methods use *in silico* structure prediction tools to predict structure that is guided by the *in vivo* and *in vitro* structure probing data.

In silico experiments make possibility of working on a symbolic gene [74], while *in vivo* and *in vitro* experiments are performed on a material object.

In general, as Wieber mentioned, the *in silico* experiment has been added to the toolbox which molecular biologists use to provide and interpret experimental results [74][75].

Despite a broad use of *in silico* nowadays, the most popular applications that use *in silico* experiments for predicting RNA folding structure have some limitations. For

example, these applications cannot predict pseudoknots in general. However, Hajdin and his teams have shown that there could be a possibility for the prediction of pseudoknots by *in silico* structure prediction that is guided by *in vitro* SHAPE data[76].

1.5 **Zebrafish**

The function of a cell or organism depends on its whole transcriptome (all RNAs). Global transcriptional profiling is a powerful tool for analyzing all transcripts that can exposes the gene activities in the significant biological processes, including developmental processes and genetic disorder.

Embryogenesis is a complex developmental process that requires the interaction of genes and causes changes in gene expression[77]. In recent years, there has been much studies in the transcriptomes of vertebrate embryogenesis to understand the effects of genes interaction and changes in gene expression in embryogenesis at the molecular and cellular levels.

The embryonic development consists of 8 stages including Zygote, Cleavage, Blastula, Gastrula, Segmentation, Pharyngula, Hatching, Early larva.

After fertilization event (a female egg is joined by a male sperm), a zygote cell is formed and then during the periods of embryonic development, the zygote as a single cell is changed to embryo.

The transcriptomes of vertebrate embryogenesis are analyzed by transcriptomics techniques including DNA microarrays and next-generation sequencing technologies called RNA-Seq [78]. This dataset reveals information about the global expression patterns during vertebrate embryogenesis.

The Zebrafish is an excellent model organism to study vertebrate development. The zebrafish, scientifically called “Danio rerio” is a freshwater fish belonging to the minnow family and can be found in the streams of South Asian countries.

Since the 1960, the Zebrafish has been a successful vertebrate model to scientific research as mammalian models. It has several distinct advantages over other vertebrate models that make it an interesting model for investigating human genetics and disease [79].

Some advantages of zebrafish are as follows:

- Zebrafish is typically small (2.5 cm to 4 cm long) so they are cost-effective since they require minor lab space. They also make the use of high-throughput strategies possible.
- Their fertilization is external, and their embryos are transparent. This allows direct visualization of development of internal structures for researchers.

- Zebrafish have a similar genetic structure to humans. In fact, 70% of human genes are found in zebrafish.
- 82 percent of genes involved in the human disease with their corresponding zebrafish genes are known.
- Zebrafish has the same major structures and tissues as humans.
- Entire zebrafish genome has been sequenced with high accuracy [80].

Considering all these advantages, zebrafish is deemed as an excellent vertebrate model to study vertebrate developmental disorders and human diseases [81][82].

In this thesis we used SHAPE-Seq data from zebrafish in both *in vivo* and *in vitro* experiments for Oblong stage of Blastula.

Chapter 2

Methodology

This chapter briefly presents the list of the software used in the thesis with brief introductions for each one, capabilities and output. The chapter eventually presents the protocol for analyzing data *in vivo*, *in vitro* and *in silico* that we performed by different tools and software in this thesis.

2.1 Software and tools used

To perform this project, a personal MacBook Air with a 1.8-GHz Intel Core i5 processor as the hardware and a CentOS as remote server was used for the programming parts.

The software used in this thesis are: HISAT, SAMtools, HTSeq, ViennaRNA and RStudio for using R version 3.6.

2.1.1 HISAT

HISAT (hierarchical indexing for spliced alignment of transcripts) is an efficient tool for alignment of reads from both RNA and DNA sequencing experiments that has faster performance than other methods with better alignment quality[83]. HISAT is designed with an indexing scheme based on the Burrows-Wheeler transform (BWT) and the Ferragina-Manzini (FM) index, employs two types of indexes for alignment: a whole-

genome FM index to anchor each alignment, and numerous local FM indexes for very rapid extensions of these alignments.

Despite its large number of indexes, HISAT requires low memory footprint approximately 4.3 gigabytes of memory for the human genome so HISAT supports genomes of any size, including those larger than 4 billion bases.

After RNA-seq experiment, analysis of data begin by mapping reads to a reference genome to determine the location of reads against the reference genome, it can be done by HISAT, which is available as free, open-source software from reference [83].

HISAT 2 is developed based on the HISAT and Bowtie2 implementations so it is a good replacement to both HISAT and TopHat2 .

HISAT2 outputs alignments in SAM format, enabling interoperation with a large number of other tools (e.g. SAMtools) that use SAM. HISAT2 is distributed under the GPLv3 license, and it can be install on the Linux, Mac OS X and Windows.

After running HISAT2 for the RNAseq, messages summarizing which is an information of mapping the RNAseq to the reference genome, is printed to the "standard error" ("stderr") filehandle.

The summary might look like this:

7465103 reads; of these:
7465103 (100.00%) were paired; of these:
5238926 (70.18%) aligned concordantly 0 times
1198410 (16.05%) aligned concordantly exactly 1 time
1027767 (13.77%) aligned concordantly >1 times

5238926 pairs aligned concordantly 0 times; of these:
8441 (0.16%) aligned discordantly 1 time

5230485 pairs aligned 0 times concordantly or discordantly; of these:
10460970 mates make up the pairs; of these:
6718543 (64.22%) aligned 0 times
2499251 (23.89%) aligned exactly 1 time
1243176 (11.88%) aligned >1 times
55.00% overall alignment rate

2.1.2 SAMtools

SAM (Sequence Alignment Map) format is a common format for storing large biological sequence alignments. The advantage of using sam format:

- Supports all the alignment information produced by different alignment programs
- can be easily converted to other formats

- The data in the sam file can be stored in binary file as a compressed file with the same data
- Is compact in file size
- Can be indexed by genomic position to efficiently recover all reads aligning to a locus.

The SAM format contains a header and an alignment section.

The header section is previous of the alignment section if it exists in the sam file. Heading part begins with the '@' symbol, which separates them from the alignment section. Alignment sections have 11 mandatory fields, as well as a variable number of optional fields[84].

The mandatory fields include QNAME, FLAG, RNAME, POS, MAPQ, CIGAR, RNEXT, PNEXT, TLEN, SEQ and QUAL.

Sam files can be analyzed and edited with the software SAMtools. Some utilities of the SAMtools include sorting, merging, indexing and generating alignments in the sam format.

2.1.3 HTSeq

The HTSeq is a Python package that provides infrastructure to process and analyze high-throughput sequencing (HTS) data.
HTSeq includes parsers for many common file formats in HTS projects and is suitable as a general platform for a various range of projects, as well as classes that simplifies working with data such as genomic coordinates (e.g. read coverage), sequences, sequencing reads, alignments, gene model information and genomic intervals (e.g. genomic features such as exons or genes) [85].

While the main purpose of HTSeq is to allow us to write our analysis scripts, there are also two stand-alone scripts, namely HTSeq are htseq-qa for common tasks that can be used from the shell command line, without any Python knowledge.

The script htseq-qa is used for reading quality assessment and producing plots that summarize the nucleotide compositions of the positions in the read and the base-call qualities.

Another script, the htseq-count is a tool for preprocessing RNA-Seq alignments for differential expression analysis by counting the number of sequencing reads overlap each of the features.

This script is one of the typical use cases for the HTSeq library that takes a SAM/BAM file and GTF/GFF file with gene models as inputs and counts for each gene how many aligned reads map to it.

These counts can then be used for gene-level differential expression analysis using R packages. As the script is designed specifically for differential expression analysis, only reads mapped unambiguously to a single gene are counted. On the other hand, the reads aligned to multiple positions or overlapping with more than one feature, is a special case that should be dealt with using htseq-count available modes as described below.

The htseq-count script allows to choose different three modes: union mode, mode intersection-strict and intersection-nonempty.

The three modes of htseq-count work as follows. For each position i in the sequencing read, a set S(i) is the set of all features mapped to position i. Then, the set S can be the union of all the sets S(i) for mode union, the intersection of all the sets S(i) for mode intersection-strict and the intersection of all non-empty sets S(i) for mode intersection-nonempty.

When S contains exactly one feature, the read or read pair is counted for this feature, likewise, when S is empty; the read or read pair is counted as no_feature. If S contains more than one feature, htseq-count behaves according to the two available options below:

- --nonunique none (default): the read or read pair is considered as ambiguous and not counted for any features. Also, if the read or read pair aligns to more than one genomic position, it is scored as alignment_not_unique.
- --nonunique all: the read or read pair is considered as ambiguous and is counted in all features where it was originally mapped. When the read or read pair is aligned to more than one position in the reference, it is scored as alignment_not_unique and it will be also individually counted for each position.

Here it should be noted that since the reads with multiple alignments or overlaps get scored multiple times, the sum of total counts will not be equal to the number of reads or read pairs.

If none of the abovementioned modes of htseq-count addresses the specific need in hand, own script can be written with HTSeq. Chapter A tour through HTSeq in reference [86] provides a step-by-step guideline on HTSeq.

Since the emerge of htseq-count in 2010, it has been widely used by the bioinformatics society. Lately, competing tools for counting reads have been published such as the summarizeOverlap function in the GenomicRanges Bioconductor package [87] and featureCount[88]. The latter possesses fast runtimes due to be run in C.
In 2014 Fonseca et al. [89] made a comparison between the accuracy counting reads of htseq-count and these other tools. They concluded that htseq-count is of higher

accuracy in this matter. However, none of these tools including htseq-count was flexible enough to properly deal with special cases. To compensate for that, HTSeq offers own script writing possibility for users which is explained in detail in reference [86].

2.1.4 ViennaRNA

The ViennaRNA is a useful tool for the RNA bioinformatics community which contains a C code library and several stand-alone programs for the prediction and comparison of RNA secondary structures. The first version of this package was released by HOfacker et al. in 1994[25].

The computer codes of ViennaRNA are based on dynamic programming algorithms introduced by Zuker [90] to compute such as predict minimum free energy structures, the equilibrium partition functions of RNA molecules, base pairing probabilities and other functions (a comprehensive programs included in the viennaRNA package as well as the reference manual are presented in reference[91]).

Amongst all implementation of viennaRNA package, RNAfold which computes the RNA secondary structure prediction through minimum free energy is the most used function in the package.

The RNAfold use RNA sequences as an input and the output is a string representation of the minimum free energy (MFE) structure and MFE values which are written in the standard output stream. RNAfold also can compute the partition function, the matrix of base pairing probabilities, and the centroid structure by using the -p option in command line.

RNAfold for representing the secondary structures, uses the dot-parentheses format i.e the three characters (,), and. to denote nucleotides that are paired or unpaired with a upstream and downstream. The dot-parentheses format is also used in many independent tools e.g. [92][93][94].

The tools included in the viennaRNA package and additional tools are available in the viennaRNA webserver [91]for public use through a web browser interface. All computations of webserver are according to the performance of the new version of viennaRNA package so the viennaRNA webserver always has been upgraded.

In this thesis we used RNAfold program. It has several command line options and we used the option –noPS in this work. By using this option one can have an output of only minimum free energy and skip producing postscript drawing of the MFE structure.

2.1.5 R

R is a powerful programming language and free software environment for analysis data and producing graphical displays of data [95].

R is free software and can run on a variety of operating systems, so it is accessible and comfortable for students and researchers.

What is most interesting about R is its flexibility which allows for many different packages to be added on its structure [96]. Several packages are being developed to address specific sets of biologic tasks.

The Bioconductor community developed several R packages [97] which include tools for analyzing biological data and addressing many bioinformatic problems.

By flexible and extensible capability of R, most bioinformatics data analysis tasks can be done in one program with add-on packages so researchers can use one programming environment for many tasks rather than using multiple tools.

The motivation of the use of R language in this thesis were also its extensibility and the growing use in bioinformatics by biological researcher.

R is accessible by command line interface and sever graphical user interfaces, such as RStudio and an integrated development environment.

In this thesis we used RStudio as open source and enterprise-ready professional software for R since it makes working with R more efficient.

2.1.6 RStudio

RStudio is a free and open source integrated development (IDE) for R. It was developed by JJ Allaire [98] who created the programming language ColdFusion.

RStudio is accessible in two ways: RStudio Desktop which is a standalone desktop application and it works with the R that installed on local Windows, Mac OS X or Linux workplace and RStudio Server which is provided a browser-based interface (the RStudio IDE) to a version of R running on a remote Linux server.

In this thesis we used RStudio server because of addressing the problems of below:

- Access to R workspace from any computer
- Access to the more powerful computer resources (larger CPU and memory footprints)
- Easy sharing of code, data, and other files

Logging to RStudio server depends on type of authentication system. By default, RStudio uses PAM authentication which can log in by username and password of Linux server for other authentication system, it is needed to create local PAM accounts.

2.2 **Data processing**

Figure *2-1* below shows the protocol used in this master thesis to utilize different tools for SHAPE-seq data analysis. Different software used in the assignment are also mentioned in the protocol. In the figure, straight lines and curvy lines indicate data input and data output to and from software respectively. The rectangulars in the figure represent data with the type of data mentioned in a bracket, and the ovals represent the tools used.

HISAT, SAMtools, HTSeq, ViennaRNA package and Rstudio are free, open-source software tools for analysis of SHAPE-seq experiments. They are useful for alignment of reads to a reference genome, computing the abundance of reads per gene in each sample, and comparing *In vivo*, *In vitro* and *In silico* data samples.
The protocol describes all the steps necessary to process a large set of raw sequencing reads and to create different kind of plots to properly compare the data samples. The execution time of this protocol highly depends on the available computing resources, but approximately takes under 720 minutes for the resources used in this master thesis which were explained in detail in section 2.1.

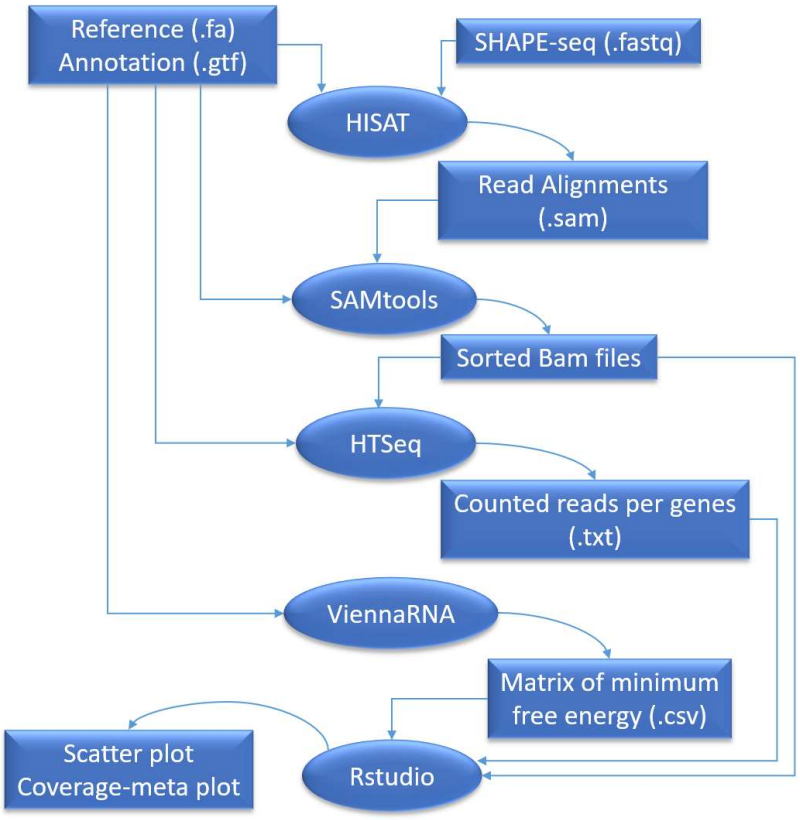

Figure 2-1: An overview of the Shape-seq data analysis protocol

2.2.1 Collecting the FASTQ files of the Oblong stage from in vitro and in vivo experiments

The FASTQ files can be reached in the directory of remote system that is described below:

/export/valenfs/data/raw_data/SHAPE/SHAPES_June2018/raw

- ✓ Connecting to ssh kjempetuja, copying the files to a directory that was created to store all data of this thesis

In command line:
$ ssh haniehr@login.ii.uib.no
$ ssh kjempetuja
$cd /export/valenfs/data/raw_data/SHAPE/SHAPES_June2018/raw

In vivo data:
$ cp {NAI-N3_Oblong_S21_L008_R1_001.fastq.gz, NAI-N3_Oblong_S21_L008_R2_001.fastq.gz, NAI-N3_old_Oblong_S24_L008_R1_001.fastq.gz,NAI-N3_old_Oblong_S24_L008_R2_001.fastq.gz} /Home/ii/haniehr/

In vitro data:
$ cp {invitro_MCE_Oblong_S26_L008_R1_001.fastq.gz, invitro_MCE_Oblong_S26_L008_R2_001.fastq.gz, invitro_old_Oblong_S25_L008_R1_001.fastq.gz, invitro_old_Oblong_S25_L008_R2_001.fastq.gz} /Home/ii/haniehr/

$logout (from kjempetuja)
$logout (from haniehr)

$ scp haniehr@login.ii.uib.no:/Home/ii/haniehr/{NAI-N3_Oblong_S21_L008_R1_001.fastq.gz, NAI-N3_Oblong_S21_L008_R2_001.fastq.gz, NAI-N3_old_Oblong_S24_L008_R1_001.fastq.gz, NAI-N3_old_Oblong_S24_L008_R2_001.fastq.gz} /Users/haniehroodashty/bin

$ scp haniehr@login.ii.uib.no:/Home/ii/haniehr/{invitro_MCE_Oblong_S26_L008_R1_001.fastq.gz, cp invitro_MCE_Oblong_S26_L008_R2_001.fastq.gz, invitro_old_Oblong_S25_L008_R1_001.fastq.gz, invitro_old_Oblong_S25_L008_R2_001.fastq.gz} /Users/haniehroodashty/bin

2.2.2 Using Hisat2 to map paired-end reads to the transcriptome/genome

We used HISAT2, which uses algorithmic improvements that give higher accuracy than the original HISAT system.

2.2.2.1 Downloading and installing

- ✓ Requirements for running Hisat2
  - Download and setup xcode form apple store
  - Install GNU GCC compiler on mac OS X (gcc version 7.1.0)
- ✓ A directory was created to store all the executable programs and data used and created in the thesis.

In command line:
$ mkdir $HOME/bin
Add the above directory to my PATH
$ export PATH=$HOME/bin:$PATH

- ✓ The binary package of Hisat2 was downloaded from [1]
- ✓ Hisat2 executables was copied to a directory in our PATH

In command line:
$ cp hisat2-2.1.0/hisat2* $HOME/bin

2.2.2.2 Procedure

- ✓ A Genome Index from danRer11.fa was created

The GRCz11 zebrafish genome (danRer11.fa) was downloaded from [2].

In command line:
$ Hisat2-build danRer11.fa danRer_index
The output is:
danRer_index.1.ht2, danRer_index.2.ht2, . . . and danRer_index.8.ht2

Splice sites from a GTF annotation file was extracted and the GRCz11 annotation was downloaded as a GTF file from [2].

We needed to have the same name chromosome in fa and gtf file when using them in different tools so we downloaded them from one source and also to change the gene_Id in gtf file because the gene_Id and transcript_Id fields had the same name so we changed the gene_Id names in Excel by removing dot(.) and value after that (For example, if the gene_Id is "XM_021475941.1" after changing it is "XM_021475941").

In command line:
$ Hisat2_extract_splice_sites.py annotation_danRer11_edited.gtf >
danRer11_splice_sites.txt

- ✓ Running Hisat2 to Map the reads to the reference genome for each sample (*In vivo* and *In vitro*)

In command line:
$ hisat2 -x danRer11_index --known-splicesite-infile danRer11_splice_sites.txt -p 2 -1 invitro_MCE_Oblong_S26_L008_R1_001.fastq.gz -2 invitro_MCE_Oblong_S26_L008_R2_001.fastq.gz -S invitro_MCE_Oblong_S26_L008.sam

The output is:
7542803 reads; of these:
7542803 (100.00%) were paired; of these:
4830582 (64.04%) aligned concordantly 0 times
1326801 (17.59%) aligned concordantly exactly 1 time
1385420 (18.37%) aligned concordantly >1 times

4830582 pairs aligned concordantly 0 times; of these:
10609 (0.22%) aligned discordantly 1 time

4819973 pairs aligned 0 times concordantly or discordantly; of these:
9639946 mates make up the pairs; of these:
5966484 (61.89%) aligned 0 times
2320425 (24.07%) aligned exactly 1 time
1353037 (14.04%) aligned >1 times
60.45% overall alignment rate

In command line:
$ hisat2 -x danRer11_index --known-splicesite-infile danRer11_splice_sites.txt -p 2 -1 invitro_old_Oblong_S25_L008_R1_001.fastq.gz -2 invitro_old_Oblong_S25_L008_R2_001.fastq.gz -S invitro_old_Oblong_S25_L008.sam

The output is:
7465103 reads; of these:
7465103 (100.00%) were paired; of these:
5238926 (70.18%) aligned concordantly 0 times
1198410 (16.05%) aligned concordantly exactly 1 time
1027767 (13.77%) aligned concordantly >1 times

5238926 pairs aligned concordantly 0 times; of these:
8441 (0.16%) aligned discordantly 1 time

5230485 pairs aligned 0 times concordantly or discordantly; of these:
10460970 mates make up the pairs; of these:
6718543 (64.22%) aligned 0 times
2499251 (23.89%) aligned exactly 1 time

1243176 (11.88%) aligned >1 times
55.00% overall alignment rate

In command line:
$ hisat2 -x danRer11_index --known-splicesite-infile danRer11_splice_sites.txt -p 2 -1 NAI-N3_Oblong_S21_L008_R1_001.fastq.gz -2 NAI-N3_Oblong_S21_L008_R2_001.fastq.gz -S NAI-N3_Oblong_S21_L008.sam

The output is:
7471263 reads; of these:
7471263 (100.00%) were paired; of these:
5027424 (67.29%) aligned concordantly 0 times
1784490 (23.88%) aligned concordantly exactly 1 time
659349 (8.83%) aligned concordantly >1 times

5027424 pairs aligned concordantly 0 times; of these:
11035 (0.22%) aligned discordantly 1 time

5016389 pairs aligned 0 times concordantly or discordantly; of these:
10032778 mates make up the pairs; of these:
6368605 (63.48%) aligned 0 times
2899903 (28.90%) aligned exactly 1 time
764270 (7.62%) aligned >1 times
57.38% overall alignment rate

In command line:
$ hisat2 -x danRer11_index --known-splicesite-infile danRer11_splice_sites.txt -p 2 -1 NAI-N3_old_Oblong_S24_L008_R1_001.fastq.gz -2 NAI-N3_old_Oblong_S24_L008_R2_001.fastq.gz -S NAI-N3_old_Oblong_S24_L008.sam

The output is:
39509059 reads; of these:
39509059 (100.00%) were paired; of these:
28691477 (72.62%) aligned concordantly 0 times
8428212 (21.33%) aligned concordantly exactly 1 time
2389370 (6.05%) aligned concordantly >1 times

28691477 pairs aligned concordantly 0 times; of these:
61445 (0.21%) aligned discordantly 1 time

28630032 pairs aligned 0 times concordantly or discordantly; of these:
57260064 mates make up the pairs; of these:
37479137 (65.45%) aligned 0 times
16067137 (28.06%) aligned exactly 1 time
3713790 (6.49%) aligned >1 times
52.57% overall alignment rate

2.2.3 Sorting and converting sam files using SAMtools

2.2.3.1 Downloading and installing

- ✓ SAMtools were downloaded from [3], SAMtools tar file was unpacked, cd to the SAMtools source directory and SAMtools binary was copied to our path

In command line:

```
$ tar jxvf samtools-0.1.19.tar.bz2
$ cd samtools-0.1.19
$ make
$ cp samtools-0.1.19/samtools $HOME/bin
```

2.2.3.2 Procedure

- ✓ Sort and convert the sam files to bam files

In command line:

```
$ samtools sort -@ 2 invitro_MCE_Oblong_S26_L008.sam -o
invitro_MCE_Oblong_S26_L008.bam
$ samtools sort -@ 2 invitro_old_Oblong_S25_L008.sam
-o invitro_old_Oblong_S25_L008.bam
$ samtools sort -@ 2 NAI-N3_Oblong_S21_L008.sam -o NAI-
N3_Oblong_S21_L008.bam
$ samtools sort -@ 2 NAI-N3_old_Oblong_S24_L008.sam -o NAI-
N3_old_Oblong_S24_L008.bam
```

2.2.4 Counting reads of genes by HTSeq

2.2.4.1 Downloading and installing

- ✓ Requirements for running HTSeq [4]
  - Install Python 2.7 or above

In command line:

```
$ brew install python
```

- Install NumPy, a commonly used Python package for numerical calculations (We needed as well as to install pip3)

In command line:

```
$ pip3 install --upgrade pip
$ python3 -m pip install --user numpy scipy matplotlib ipython jupyter pandas sympy
nose
```

- Install Pysam, a Python interface to SAMtools

In command line:
$ pip install pysam

- ✓ The source package of HTSeq was downloaded from [5], unpacked the tar file, cd to the HTSeq source directory and compiled HTSeq

In command line:
$ tar jxvf HTSeq-0.11.2.tar
$ cd HTSeq-0.11.2
$ python3 setup.py build
$ python3 setup.py install

2.2.4.2 Procedure

- ✓ Counting how many reads map to each gene by htseq-count tool for all bam files.

In command line:
$python3 -m HTSeq.scripts.count -f bam -r pos -s yes --nonunique all invitro_MCE_Oblong_S26_L008.bam annotation_danRer11_edited.gtf > result_HTSeq_count_invitro_MCE_Oblong_yes.txt

The output is:
__no_feature 198532
__ambiguous 527094
__too_low_aQual 4916712
__not_aligned 1283637
__alignment_not_unique 2361465

$ python3 -m HTSeq.scripts.count -f bam -r pos -s yes --nonunique all invitro_old_Oblong_S25_L008.bam annotation_danRer11_edited.gtf > result_HTSeq_count_invitro_old_Oblong_yes.txt

The output is:
__no_feature 161400
__ambiguous 484373
__too_low_aQual 4661359
__not_aligned 1592509
__alignment_not_unique 1916912

$ python3 -m HTSeq.scripts.count -f bam -r pos -s yes --nonunique all NAI-N3_Oblong_S21_L008.bam annotation_danRer11_edited.gtf > result_HTSeq_count_NAI-N3_Oblong_S21_L008_yes.txt

The output is:
__no_feature 173066
__ambiguous 788940

__too_low_aQual 4268109
__not_aligned 1404581
__alignment_not_unique 1206926

$ python3 -m HTSeq.scripts.count -f bam -r pos -s yes --nonunique all NAI-N3_old_Oblong_S24_L008.bam annotation_danRer11_edited.gtf > result_HTSeq_count_ NAI-N3_old_Oblong_S24_L008_yes.txt

The output is:
__no_feature 754269
__ambiguous 3725121
__too_low_aQual 21941000
__not_aligned 9061528
__alignment_not_unique 5134797

2.2.5 Calculating minimum free energy secondary structures and getting csv matrix of them by Vienna RNA

2.2.5.1 Downloading and installing

- ✓ Requirements for installing viennaRNA package:
  - Install conda with the Miniconda package from [6]
  - After installing conda we need to add the bioconda channel as well as the other channels bioconda depends on

In command line:
$ conda config --add channels defaults
$ conda config --add channels bioconda
$ conda config --add channels conda-forge

- ✓ Install ViennaRNA package

In command line:
$conda install viennarna

In this thesis it was tried to install viennaRNA package according to the instruction from [7], it should be noted that after the installation of binary package when [./configure] is typed this error message appeared:
Gcc-ar cannot find liblto-plugin.so
We found that mac has an issue with gcc-ar which has not solved yet. The solution was to install vennaRNA by conda.

- ✓ Copying the RNAfold executable file from vennaRNA package which is in annaconda3 package to a directory that was created to store all data of this thesis

2.2.5.2 Procedure

- ✓ Connect to ssh kjempetuja and download three files (parse_vf.pl, fasta_windows.pl and minimum free energy.sh).

In command line:
$ cd /export/valenfs/projects/adam/final_results/scripts/repo3/giess-scripts/TIS_prediction/prediction_scripts/05_prediction_plots/plots_novel
$ cp {parse_vf.pl, fasta_windows.pl, minimum free energy.sh} /Home/ii/haniehr/
$logout (from kjempetuja)
$logout (from haniehr)
$ scp haniehr@login.ii.uib.no:/Home/ii/haniehr/ {parse_vf.pl, fasta_windows.pl} /Users/haniehroodashty/bin

- ✓ Run minimum free energy.sh shell script [1] for input tx.fasta file (sequence of filtered transcripts)

In the shell script, those two downloaded files are used to split fasta file in sliding windows of 39 bases and RNAfold with option –noPS run on all split fasta files to create MFE for each window of transcripts. Eventually the matrix of minimum free energy from all transcripts is obtained in csv format.

The tx.fasta is created by viennaScript.R script that can be found in the appendix of this thesis and also in GitHub [1].

In command line:

$ minimum free energy.sh -f tx.fasta –o ./Vienna

(This command also can run in R)

After running this script, three folders are created in thesis directory with names csvs, viennas and windows. The csv file is on csvs folder with name am.csv.

It is noteworthy that the file am.csv has some lines that do not have any information about minimum free energy, so we removed (skip) 23479 first lines, because these lines are just the names of the transcripts without any values. After these lines, the values for the transcripts start to appear. We called this new csv file “am2.csv”.

2.2.6 Using Rstudio to create scatter plot and coverage meta plot of data

We used Rstudio server in this thesis since the files had so large size that could not be processed on Rstudio desktop.

2.2.6.1 Downloading and installing

- ✓ The Rstudio server has been installed in the kjempetuja (the instruction of the installation of Rstudio server can be found from [98]).

2.2.6.2 Procedure

- ✓ Connect to Rstudio server

In command line:
$ ssh -L 20002:kjempefuru.cbu.uib.no:8787 -N haniehr@login.ii.uib.no

Then we go to browser and write localhost:20002.

The main page of Rsudio server will be open after using valid credential to login.

- ✓ Loading annotation file, csv, bam and txt files

Loading the annotation file and extract the exons grouped by filtered transcript:

```
Library (GenomicFeatures)
txdb <- makeTxDbFromGFF (file = "annotation_danRer11_edited.gtf")
tx <- exonsBy (txdb, use.names = TRUE)
tx <- tx [widthPerGroup(tx) > 200]
```

Load the csv file:

```
library(data.table)
hits <- fread ("am2.csv" ,nrow = length(tx), header = FALSE , fill=TRUE )
```

It was tried to load csv file in R with fread function, but the error message appeared:

"Expected 10082 fields but found 14663 fields".

This issue was related to the different fields (number of columns) that existed in some lines.

The solution was to use skip argument in fread. We used this argument 3 times. In other words, every time the error was experienced fread function was used again by starting from the problematic fields had problem and eventually we received 3 data table (hits2, hits3, hits4) and then bound them into hit file.

Loading the aligned reads of *in vitro* and *in vivo* (bam file):

For example, loading first replicate of *in vitro* data

```
Library (GenomicAlignments)
invitroRep1<- GRanges (readGAlignments ("invitro_MCE_Oblong_S26_L008.bam"))
```

Load txt file:
For example, loading the output of HTSeq of first replicate for *in vitro* data

```
htseq_invitro_rep1 <- read.delim
("result_HTSeq_count_invitro_MCE_Oblong_yes.txt",header = FALSE)
```

- ✓ Creating scatter plot and coverage meta plot for comparing data

For creating these plots, it was required to load two libraries below:

```
Library (ggplot2)
Library (ORFik)
```

Now, it is possible to use ggscatter function for making scatter plot of *in vitro* and *in vivo* data and then by windowCoveragePlot function from ORFik package, make coverage meta plot of transcripts and transcript regions i.e. Leader (5' UTR), CDS and Trailer (3' UTR).

For more details of data prepared analysis and plotting see R code in the appendix of the thesis or the entry for this master project in GitHub [1].

Chapter 3

Results and Discussion

The output of the generated code for analyzing and comparing data for both *in vivo* and *in vitro* and eventually *in silico* is presented in this chapter. Relevant discussions and observations are also mentioned together with the results.

3.1 **Introduction**

As explained in Chapter 2, the data for *in vivo*, *in vitro* and *in silico* were collected and prepared for analysis in the first step. Then the coverage plots along transcript and also along regions per transcript were plotted separately for each set of data.

A comparison of structurality between the plotted data was performed afterwards. It was eventually discussed how and why the structure of various regions are different and what implications the number of reads in each case have in connection with the RNA structure.

In all following data processing, the transcripts were filtered based on size restrictions: leader > 100 bases, CDS > 100 bases and trailer > 100 bases and also the longest transcript per gene is selected, therefore the number of transcripts is equal to the number of genes.

3.2 **Processing *in vivo* data and results**

3.2.1 Scatter plot of observed reads per gene for in vivo replicates data

The number of genes in the data is shown in relation to the number of the mapped reads per gene for both *in vivo* replicates “Invivo_NAI-N3_Oblong” and “Invivo_NAI-N3_old_Oblong”.

Here the X-axis is log2 number of observed reads per gene and the Y-axis is log2 frequency of genes.

These plots were made to control the quality of the data and the depth of sequencing which come from *in vivo* experiment.

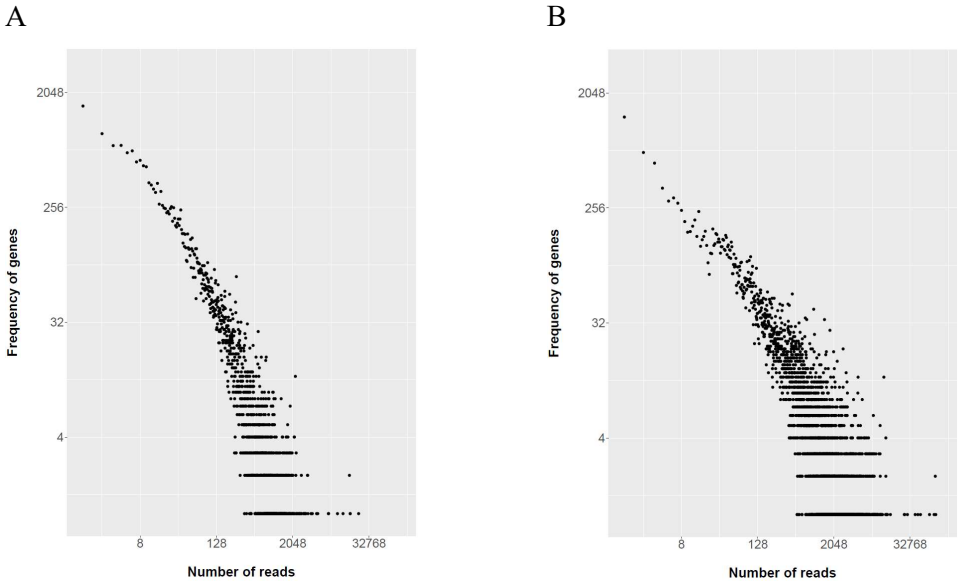

Figure 3-1 : A) Scatter plot of Invivo_NAI-N3_Oblong (replicate one). B) Scatter plot of Invivo_NAI-N3_old_Oblong (replicate two)

It can be seen from

Figure 3-1 that there are many genes with numerous reads that indicates high enough number of reads for both replicate 1 and 2, vouching for a trustworthy *in vivo* library.

3.2.2 Correlation plot of in vivo replicate one with in vivo replicate two

Pearson's coefficient (R) of number of reads is mapped to per gene between *in vivo* replicates is shown in the following plot (Figure *3-2*). The number of reads were transformed to logarithmic scale in base 2 and shown as scatter plot. Here each dot represents the number of reads of a specific transcript.

In the figure below, the X-axis is log2 number of reads per gene for REP1 (Invivo_NAI-N3_Oblong) and the Y-axis is the log2 number of reads per gene for REP2 (Invivo_NAI-N3_old_Oblong).

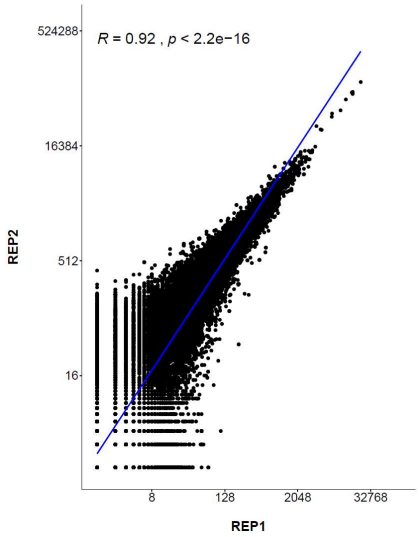

Figure 3-2: Correlation of scatter plot of *in vivo* REP1 and REP2

It can be observed from Figure *3-2* that the two replicates *in vivo* experiment agree at abundance and have a high correlation (R=92%). It will however be explained in the next section that a high correlation here can not necessarily guarantee for high accuracy of the *in vivo* experiment results, unless the error for the specific transcript regions are also examined and proved to be low enough.

3.2.3 Plot of percentage difference between replicate one and replicate two in vivo

The correlation between the two *in vivo* replicates data determines the error between the total numbers of the reads per transcript in each replicate. It does not then specify the error for each transcript region (leader, CDS and trailer) in the 'total error'. Therefore, we require to determine the difference between the number reads per transcript region (regional error) for replicate 1 and 2 and look at the summation of these errors that we call 'global error'.

Figure 3-3 is an attempt to show the role of the regional errors in the overall correlation of two replicates. In this case, the total error of the transcript is 0 between two replicates while errors for transcript regions are significant values.

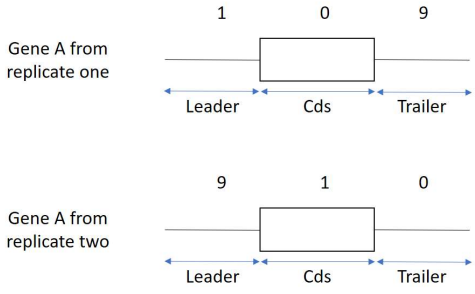

Figure 3-3 Illustration of the role of regional error in correlation of replicates

In Figure 3-3, the numbers above each transcript region correspond to their number of mapped reads. In this case, that the numbers are chosen to be extreme for the sake of clarity, while the total number of reads for a specific transcript are the same, the number of reads per transcript regions are totally different and the 'regional error' are significant. Then we consider working with the summation of regional errors, i.e. global error, as a more meaningful parameter rather than total error to understand the difference between the two replicates. In the example above the different type of errors between replicate 1 and 2 will be as follows:

Total error: $[1 + 0 + 9]_{\text{replicate1}} - [9 + 1 + 0]_{\text{replicate 2}} = 0$

Regional errors:

- % Leader-error: $[1/10]_{\text{leader - replicate 1}} - [9/10]_{\text{leader - replicate 2}} = 80\%$
- %CDS-error: $[o/10]_{\text{CDS - replicate 1}} - [1/10]_{\text{CDS- replicate 2}} = 10\%$
- %Trailer-error: $[9/10]_{\text{Trailer - replicate 1}} - [0/10]_{\text{Trailer - replicate 2}} = 90\%$

Global errors: 80% + 10% + 90% = 180%

Thereafter, global errors for all transcripts between the two replicates were calculated. Then the frequency of transcripts was plotted versus corresponding global errors.

In Figure *3-4* the X-axis is the global errors between two *in vivo* replicates in percent and the Y-axis is frequency of genes.

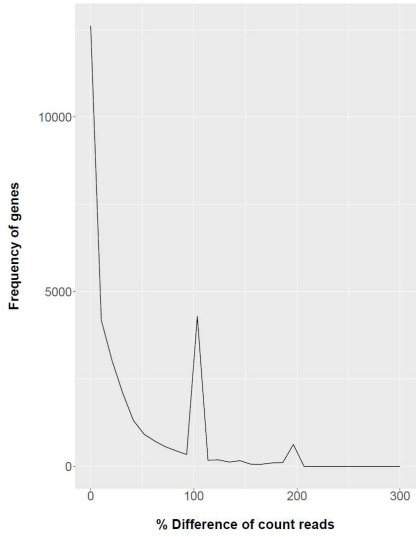

Figure 3-4: Plot of global error between *in vivo* REP1 and REP2

The peak at 100% shows around 4000 transcripts that have 100% global error. Such global error can appear in the conditions of the following example are met. In one of the replicates (replicate one in this example), 50% of the reads of a certain transcript is placed on one of the transcript regions (for example leaders). 50% of the remaining reads is then placed on one of the other two transcript regions (for example trailer) the last transcript region (i.e. CDS in this example) has no reads. Following the same example, in replicate two, 50% of the reads of transcript is placed on the CDS region of the transcript and 50% of the remaining reads is placed on trailer region. Here the last region, leader in this example, has no reads. Then the global error between the two replicates is equal to 100%.

In Figure 3-4 the peak at 200% shows around 1000 transcripts, in other words, there are ~1000 transcripts that have 200% global error. An example of such error can be when in replicate one, 100% of the reads of a given transcript is placed on the leader region of the transcript while in replicate two, 100% of the reads of the transcript is placed on the CDS region of the transcript. Then the global error between two replicates for the transcript is equal to 200%.

As we can see most of the transcripts have less than half a percent difference in the count reads. In other words, the global error is not significant here between the corresponding transcripts of the two replicates, i.e. the two replicates have a proper correlation. We can then pool both replicates and make up one library as *in vivo* data to be later used for comparison with *in silico* and *in vitro* data. This will be shown in the Comparison and Discussion section.

3.2.4 Coverage metaplot of the number of reads along the transcripts for in vivo replicates.

The number of normalized reads over transcripts (count read /sum of counts per transcript) is represented in relation to scaled position along transcript and along regions per transcript (leader, CDS and trailer) separately.

We used a 100-nt sliding window approach to scale all positions to the width of 100.

For each position, the count read was calculated as the average of the count reads obtained in each of the overlapping windows.

In Figure 3-5 A and C, the X-axis is the scaled position along transcript and along regions per transcript respectively, for *in vivo* replicate one and the Y-axis is the total number of normalized reads for each position in all transcripts. Figure 3-5 A and C appear different in terms of peak distribution while both are from *in vivo* replicate one. The reason is that we scaled the position of all transcripts to 100 in A but in C each region of transcripts is scaled to 100. It means that each region in Figure 3-5 C has a certain number of positions (100 positions). It is known that in general the number of nucleotides of leader and trailers are less than CDS and in case of zebrafish, the leader is less than trailer. When we scale the leader and trailer to 100, we actually stretch the positions of these two regions. Likewise, scaling the CDS to 100 means that the

positions of CDS are compressed. Based on these notes, we can see that the peak appearing in the early positions of the transcripts in Figure *3-5* A cannot be seen in leader region of Figure 3-5 C. It is also noteworthy that for Figure 3-5 A, the start positions of leader, CDS and trailer are not known since the transcripts can have different start position for each transcript regions. This is the case for every dataset that the coverage transcript plots and coverage leader, CDS and trailer are plotted in this thesis.

In Figure 3-5 B and D, the X-axis is the scaled position along transcript and regions per transcript respectively, for *in vivo* replicate one and Y-axis is the total number of normalized reads for each position in all transcripts.

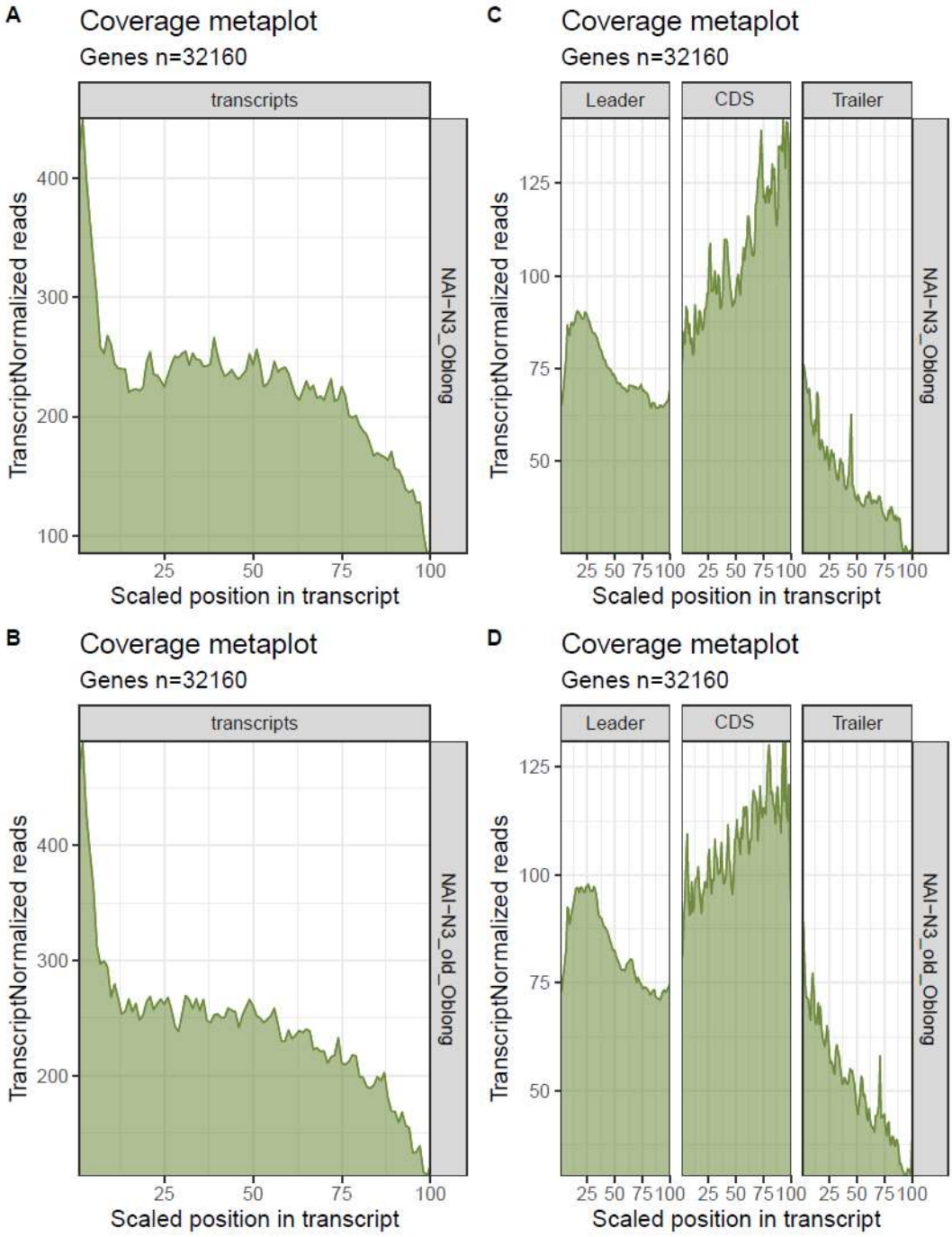

Figure 3-5: A) Coverage meta plot of transcripts in Invivo_NAI-N3_Oblong (replicate one). B) Coverage meta plot of transcripts in Invivo_NAI-N3_old_Oblong (replicate two). C) Coverage meta plot of leader, CDS and trailer in Invivo_NAI-N3_Oblong (replicate one). D) Coverage meta plot of leader, CDS and trailer in Invivo_NAI-N3_old_Oblong (replicate two)

As expected from the results in Figure 3-4; the coverage meta plot of in vivo replicates are similar in Figure 3-5. This is a reassured sign of satisfactory accuracy for in vivo experiments. Also, from Figure 3-5, CDS region of transcript appears to be less structure compared to other regions. The implications and results related to these observations will be discussed in detail in Comparison and Results section.

3.3 **Processing *in vitro* data and *results***

3.3.1 Scatter plot of observed reads per gene for in vitro replicates data

The number of genes in the data is shown in relation to the number of mapped reads per gene for both *in vitro* replicates "Invitro_MCE_Oblong" and "Invitro_old_Oblong".

Similar to data processing for *in vivo* data, these plots here also were made to control the quality of the data and the depth of sequencing.

In the plots below, the X-axis is log2 number of observed reads per gene and the Y-axis is log2 frequency of genes.

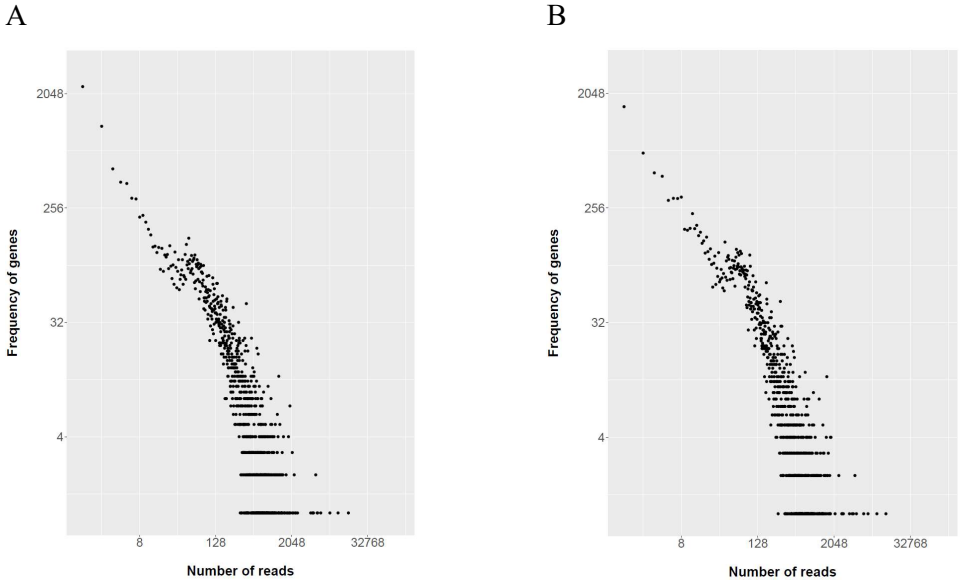

Figure 3-6: A) Scatter plot of Invitro_MCE_Oblong (replicate one). B) Scatter plot of Invitro_old_Oblong (replicate two)

In Figure 3-6 there are many genes with numerous reads that indicates high enough number of reads for both replicate 1 and 2. This will also guarantee for a reliable *in vitro* library.

3.3.2 Correlation plot of in vitro replicate one with in vitro replicate two

Pearson's coefficient (R) of the number of reads is mapped to per gene between *in vitro* replicates is shown in Figure 3-7. The number of reads were transformed to logarithmic scale in base 2, are shown as scatter plot. Each dot represents the number of reads of a specific transcript.

In Figure 3-7, X-axis is log2 number of reads per gene for REP1 (Invitro_MCE_Oblong) and Y-axis is log 2 number of reads per gene for REP2 (Invitro_old_Oblong).

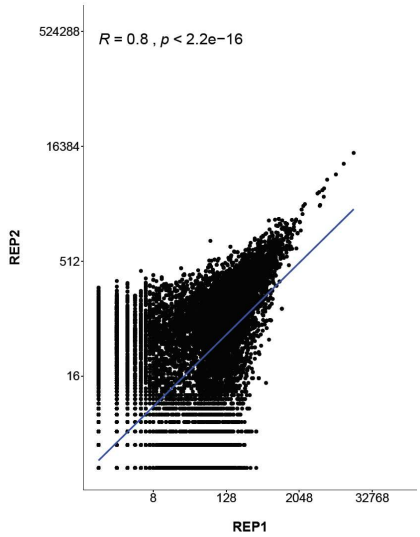

Figure 3-7: Correlation of scatter plot of REP1 and REP2 *in vitro*

3.3.3 Plot of percentage difference between REP1 and REP2 in vitro

In Figure 3-8 the X-axis is the global errors between two *in vitro* replicates data and Y-axis is frequency of genes.

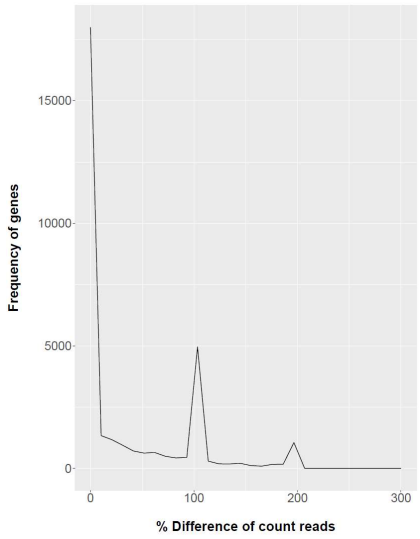

Figure 3-8: Plot of percentage difference between REP1 and REP2 *in vitro*

The majority of the transcripts in Figure 3-8 have also high similarity in count reads for both replicates. This means that two replicates have a proper correlation. We can then pool both replicates and make up one library as *in vitro* data to be later used for comparison with *in silico* and *in vivo* data. This will be shown in the Comparison and Discussion section.

3.3.4 Coverage metaplot of the number of reads along the transcripts for in vitro replicates.

The number of normalized reads over transcript is represented in relation to scaled positions along transcript and along regions per transcript separately in Figure 3-9 for *in vitro* replicates.

The positions of transcripts and each region of them are scaled to 100.

In Figure 3-9 A and C, the X-axis is the scaled position in all transcripts for *in vitro* replicate one and the Y-axis is the number of normalized reads for each position in all transcripts.

In Figure 3-5 B and D, X-axis is the scaled position in all transcripts for *in vitro* replicate one and Y-axis is the total number of normalized reads for each position of regions in all transcripts.

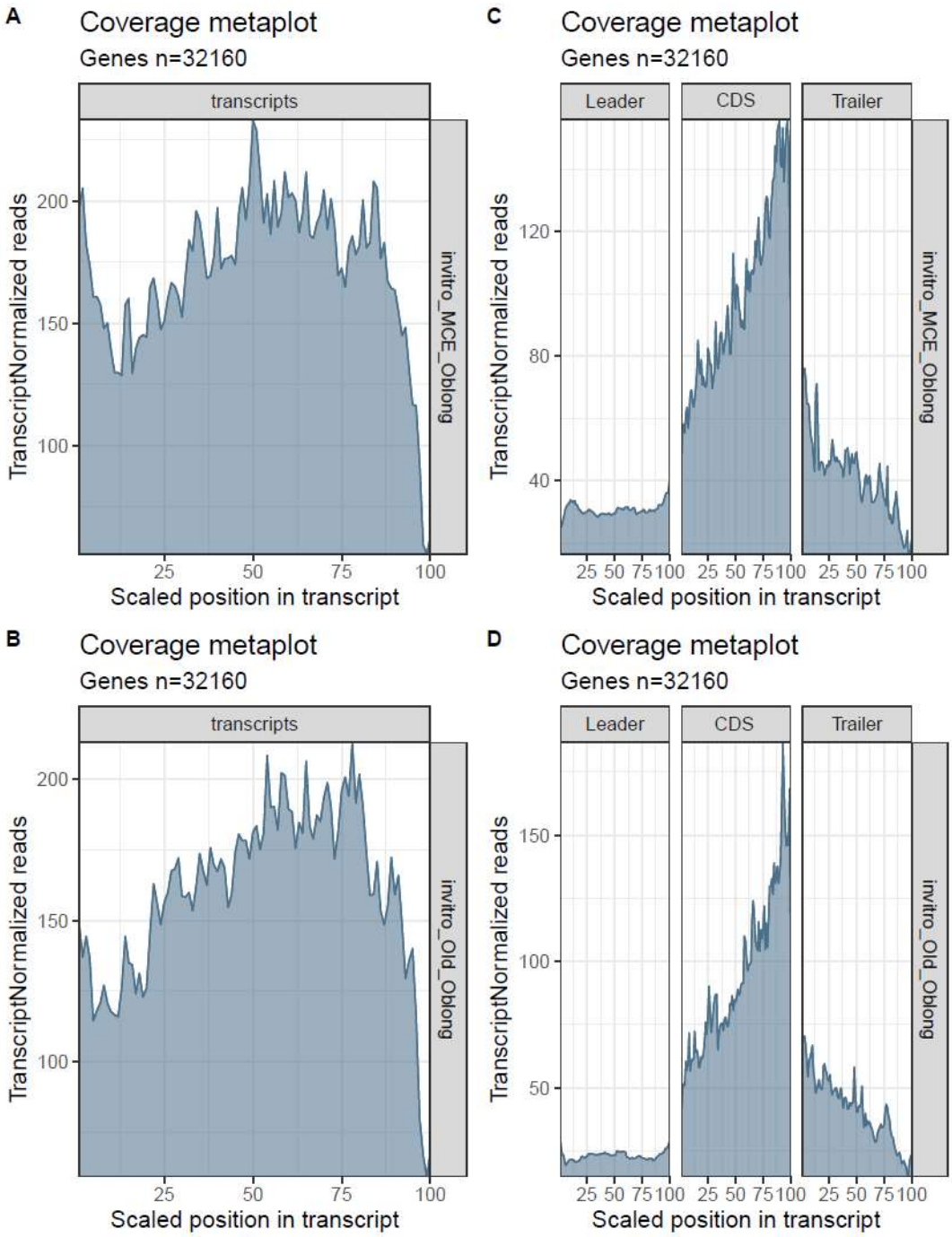

Figure 3-9: A) Coverage meta plot of transcripts in Invitro_MCE_Oblong (replicate one). B) Coverage meta plot of transcripts in Invitro_old_Oblong (replicate two). C) Coverage meta plot of leader, CDS and trailer in Invitro_MCE_Oblong (replicate one). D) Coverage meta plot of leader, CDS and trailer in Invitro_old_Oblong (replicate two)

Figure 3-9 is also in line with the expectation raised from Figure 3-8 where two replicates are similar, and this is an indication of a proper correlation between the two replicates. Also similar to 3.2.4, CDS region of transcript are the least structured region. Again, the detailed outcome related to these observations will be discussed in detail in Comparison and Results section.

3.4 **Processing *in silico* data and results**

The mean and most structured (most negative value of MFE) obtained by MFE of each transcript are consider as score of each transcript that are shown in the following figures and supplementary information including the coverage metaplots and the relevant discussion and comparisons are presented in sections 3.5.3 and 3.5.4.

3.4.1 Density plot of mean and most structured obtained by MFE of each transcript for in silico dataset

In Figure 3-10 A, mean score per gene is plotted against frequency of genes while, in Figure 3-10 B, the Maximum score per gene is plotted versus frequency of genes.

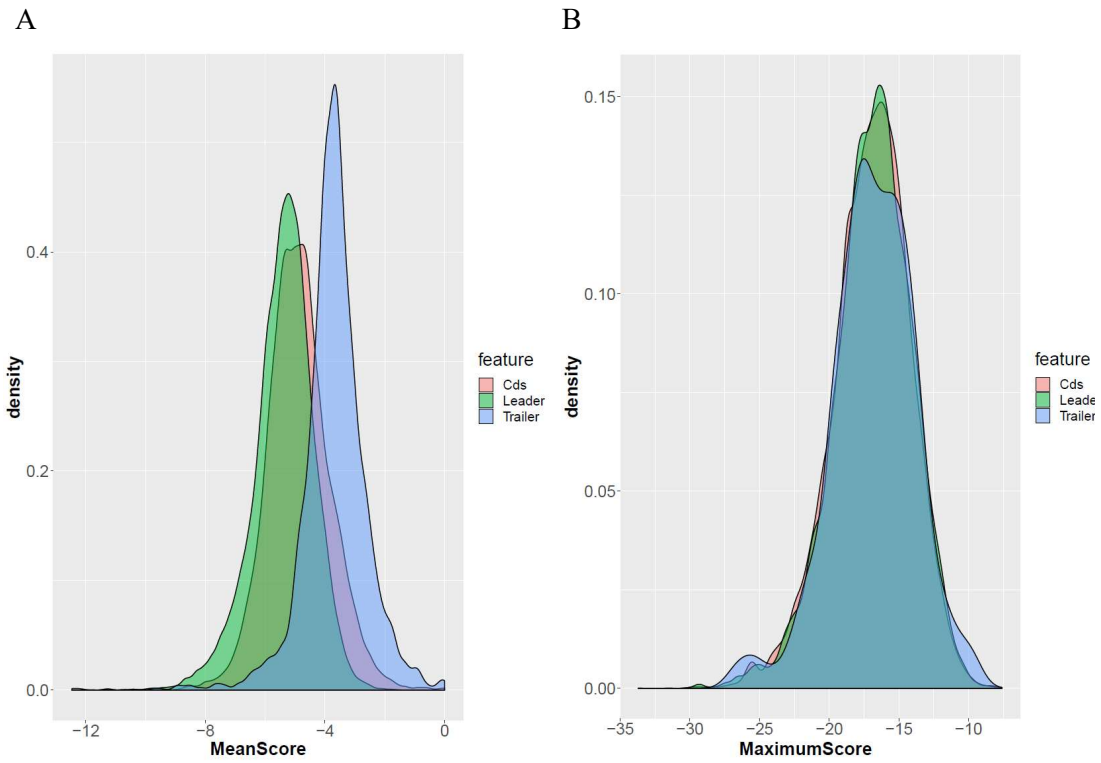

Figure 3-10: A) Density plot of Leader, CDS and Trailer for mean scores of *in silico* data. B) Density plot of Leader, CDS and Trailer for maximum scores of *in silico* data

The observation from Figure 3-10 A is that the leader region is more structure than CDS and CDS itself, is more structure than trailer. Figure 3-10 B shows maximum score divided to leader, CDS and trailer region. It shows how strong the structure is in different transcript regions.

3.5 **Comparison and discussion**

3.5.1 Scatter plot of in vivo and in vitro data

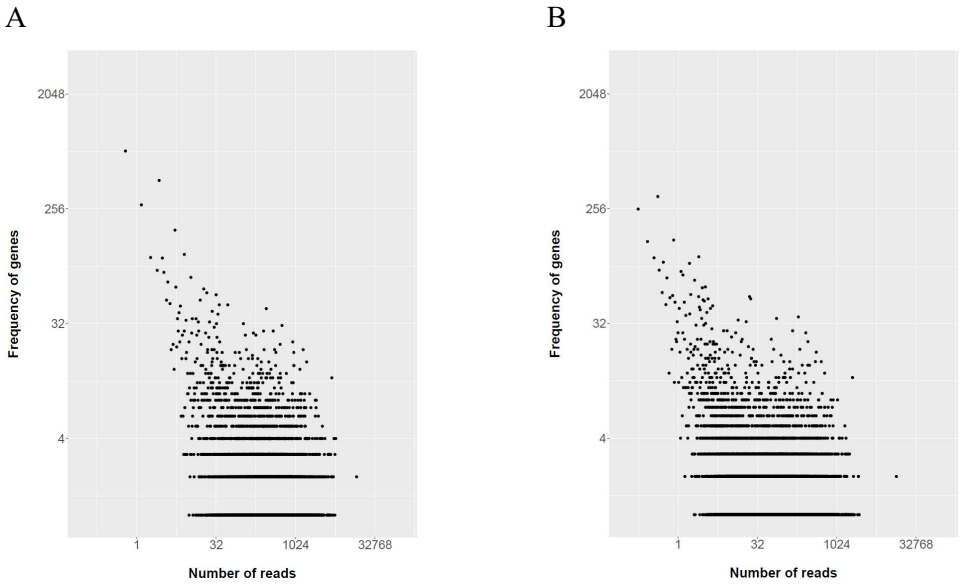

Figure 3-11: A) Scatter plot of *In vitro*. B) Scatter plot of *In vivo.*

Figure 3-11 shows several genes with numerous reads that is a quality control measure to imply high enough number of reads for both *in vivo* and *in vitro* datasets.

3.5.2 Correlation plot of In vivo with In vitro

Pearson's correlation coefficient (R) of number of reads mapped for each gene between *in vivo* and *in vitro* data is shown in this plot. The number of reads that were transformed to logarithmic scale in base 2, are shown as scatter plot here. Each dot represents the number of reads of a certain transcript.

In this following figure, the number of reads per gene for *in vitro* is plotted versus the number of reads per gene for *in vivo* data.

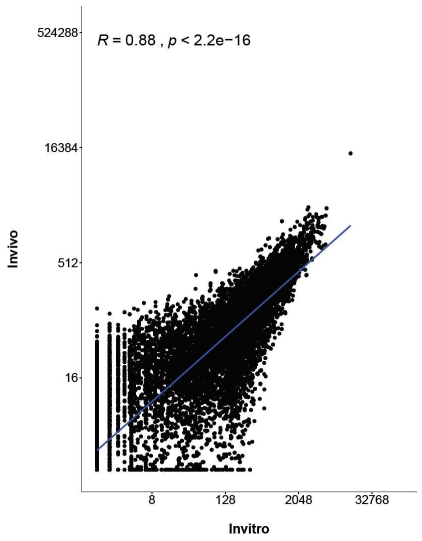

Figure 3-12: Correlation of scatter plot of *in vivo* and *in vitro*

Figure 3-12 demonstrates a high correlation between *in vivo* and *in vitro* data (R=88%). This only means that the number of reads per transcript is similar in both studies.

3.5.3 Plot of percentage difference between in vivo, in vitro and in silico

Here the number of genes in a SHAPE-Seq data is shown in relation to sum of difference percentages of count reads of regions per transcript between *in vivo* and *in vitro*, *in vivo* and *in silico* and *in vitro* and *in silico* data.

In Figure 3-13 A, B and C the X-axis is total percentage difference between *in vivo* and *in vitro, in vitro* and *in silico* and *in vivo* and *in silico* data respectively. The Y-axis is frequency of genes in all graphs.

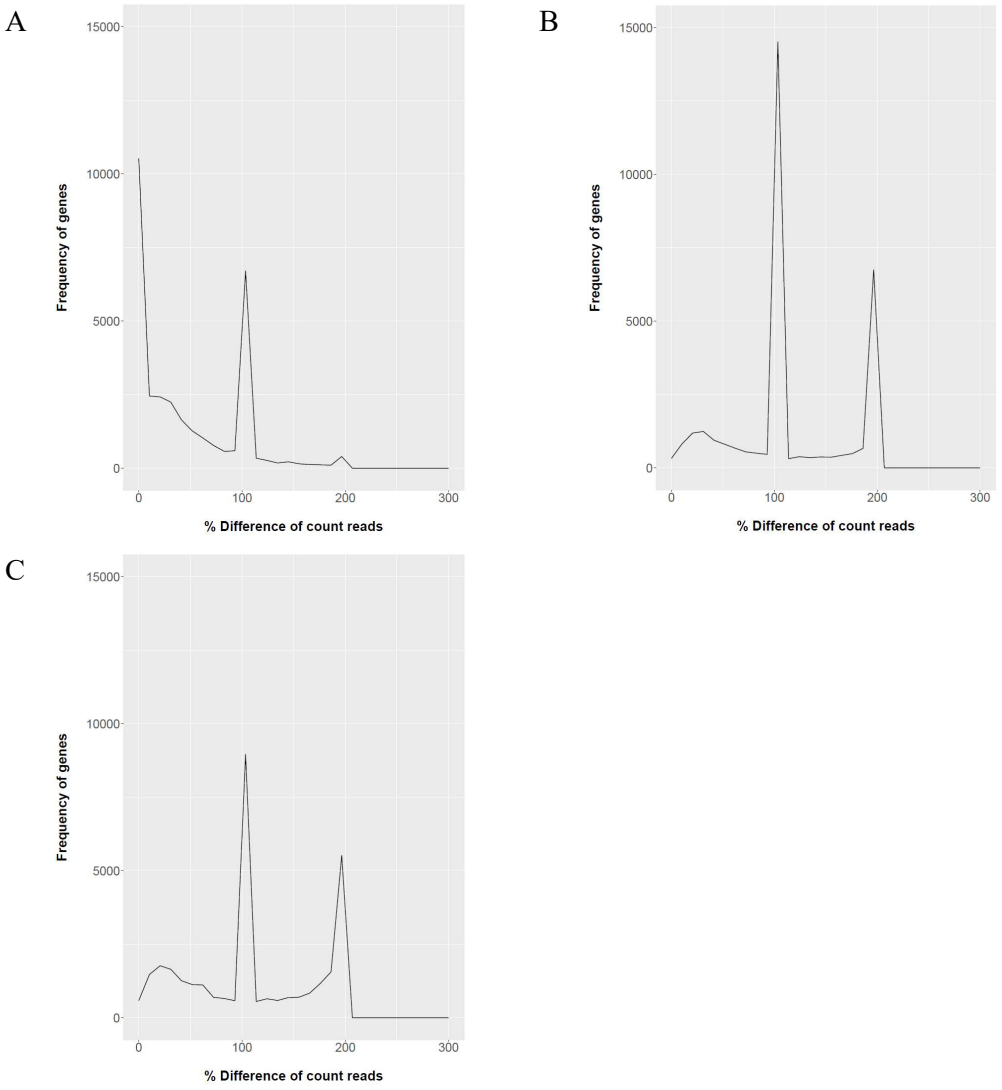

Figure 3-13: A) Plot of percentage difference between *in vivo*, *in vitro. B)* Plot of percentage difference between *in vitro and in silico. C)* Plot of percentage difference between *in vivo and in silico*

Figure 3-13 A shows insignificant difference in the count reads for the transcripts. i.e. the global error is minor here between the corresponding transcripts of *in vivo* and *in vitro.* We can possibly argue here that the cellular environment does not have significant effect on *in vivo* experimental output.

It is also observed in Figure 3-13 B and C that most of the transcripts have 100% global error in both cases. In case of *in vitro* and *in silico* (Figure 3-13 B), the 100% global error peak corresponds to a frequency of gene almost twice as the frequency of gene for the 100% peak in case of *in vivo* and *in silico* (Figure 3-13 C).

3.5.4 Comparison of Coverage metaplot of in vivo, in vitro and in silico.

The number of normalized reads over transcripts is represented in relation to scaled position along transcript and along regions per transcript (leader, CDS and trailer) separately.

The transcripts were filtered based on size restrictions: leader > 100 bases, CDS > 100 bases and trailer > 100 bases. The positions of transcript and each region of transcript is scaled to 100.

As seen in the results of section 3.2.3, the two replicates *in vivo* have a proper correlation. We now pool both replicates and make up one library as *in vivo* data. The next step is to make coverage meta plot of the in vivo data library. Likewise, the same process (pool and plot) was performed for two replicates of *in vitro* data according to the results from section 3.3.3.

Coverage meta plot of transcripts and regions per transcript for *in silico* data was plotted using absolute MFE values of transcripts.

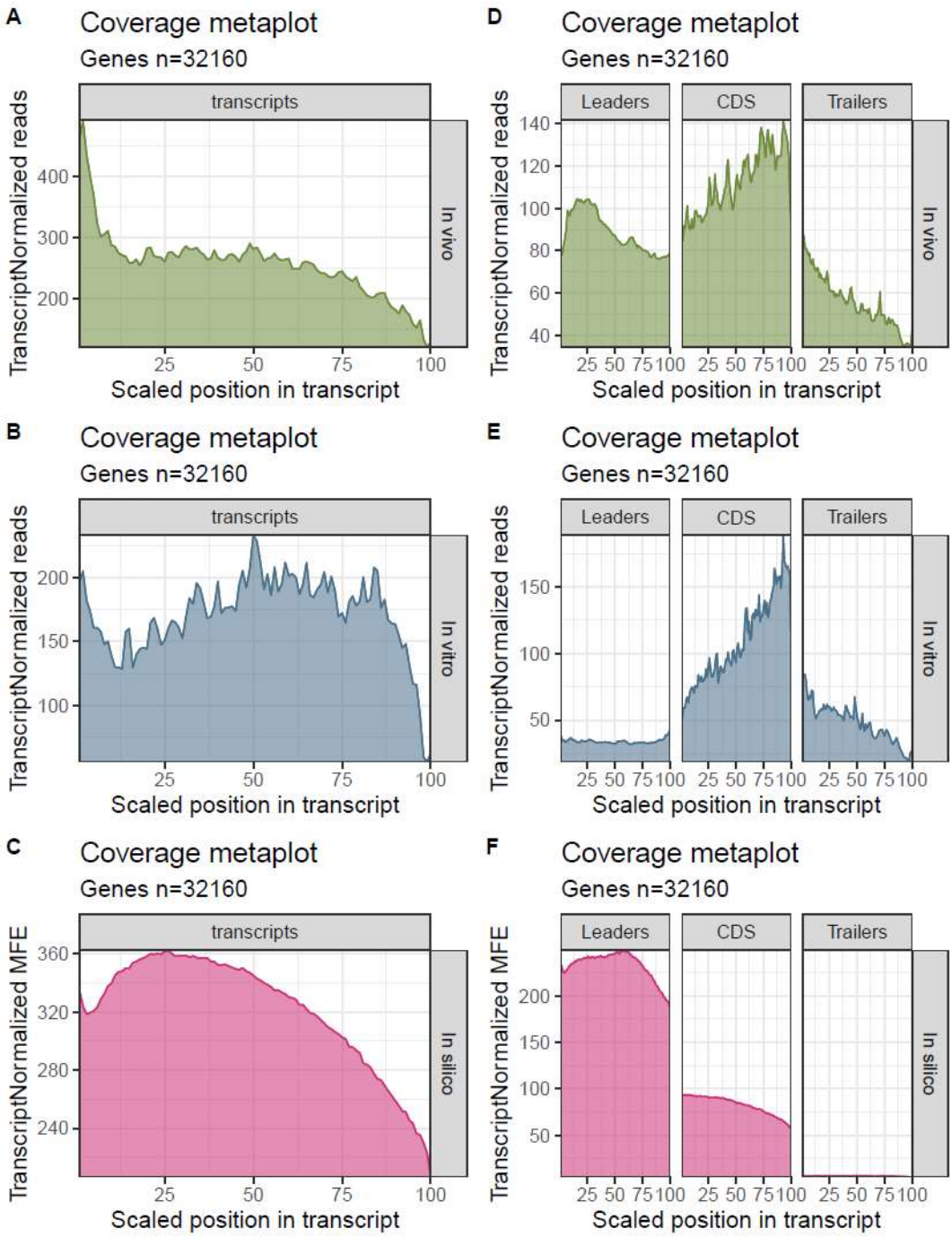

Figure 3-14: A) Coverage meta plot of transcripts *in vivo.* B) Coverage meta plot of transcripts *in vitro.* C) Coverage meta plot of transcripts *in silico.* D) Coverage meta plot of leader, CDS and trailer *in vivo.* E) Coverage meta plot of leader, CDS and trailer *in vitro.* F) Coverage meta plot of leader, CDS and trailer *in silico.*

Figure 3-14 A and B show the coverage meta plot of *in vivo* and *in vitro* transcripts respectively. A feature to note in Figure *3-14* A is the significant peak at early positions which may appear as high number of modifications in these areas. However, this is not the case here. As a matter of fact, many reads in the RT process ended up at these points and the termination of many reads (cDNA) then contributed to this peak.

It is evident from the comparison of Figure 3-14 A and B that the *in vivo* data have more modifications than that of *in vitro*, i.e. the transcript regions in *in vivo* data are less structured than *in vitro* data. And also, in both figures there is low modifications at the end position of transcripts. However, Figure 3-14 C indicates the scores significantly declined at the end position of transcripts. We know that for RT process to progress, a primer needs to be bound to its complementary sequences on the RNA template and serves as a starting point for synthesis of cDNAs. We then believe that the end positions of transcripts in the Figure 3-14 A and B show that there are not many modifications in this region due to the lack of enough position for initialization of the RT process. In other words, we believe that there were much more modifications there, but RT process could not make them visible.

Figure 3-14 D and E present coverage meta plot of leader, CDS and trailer *in vivo* and *in vitro* respectively. Comparing the leader regions in both figures, it is clear that the *in vivo* data possess higher modifications and consequently less structure compared to *in vitro* data while Figure 3-14 F shows there more structure in the leader region of transcripts as it is observed also *in vitro* leader region. We believe that this structural difference is due to RNA interactions with RNA binding proteins in the cellular environment, that open up the RNA structure.

Looking at CDS regions in both *in vitro* and *in vivo* data, it can be seen that CDS regions are less structured than the other regions, interpreted from the higher reads in the middle of the graphs. And also, the upstream CDS of both *in vivo* and *in vitro* is more structure than the downstream CDS of them. Despite this similarity between the two sets of data, there are some positions in CDS region of *in vitro* that have coverages as high as 180, while the maximum coverage in CDS region of *in vivo* is around 140.

Trailer regions in both *in vitro* and *in vivo* are very similar in the general trends. By looking at Figure 3-14 F, it is obvious that there is almost no structure in the trailer region while the number of modifications in the trailer of *in vivo* and *in vitro* is not high. Here again, there is an observation of almost no structure at the region of trailers. The explanation here is also the same as previously presented, where we believe that the lack of sufficient space to initialize the RT process leads to these modifications not being captured by the assay.

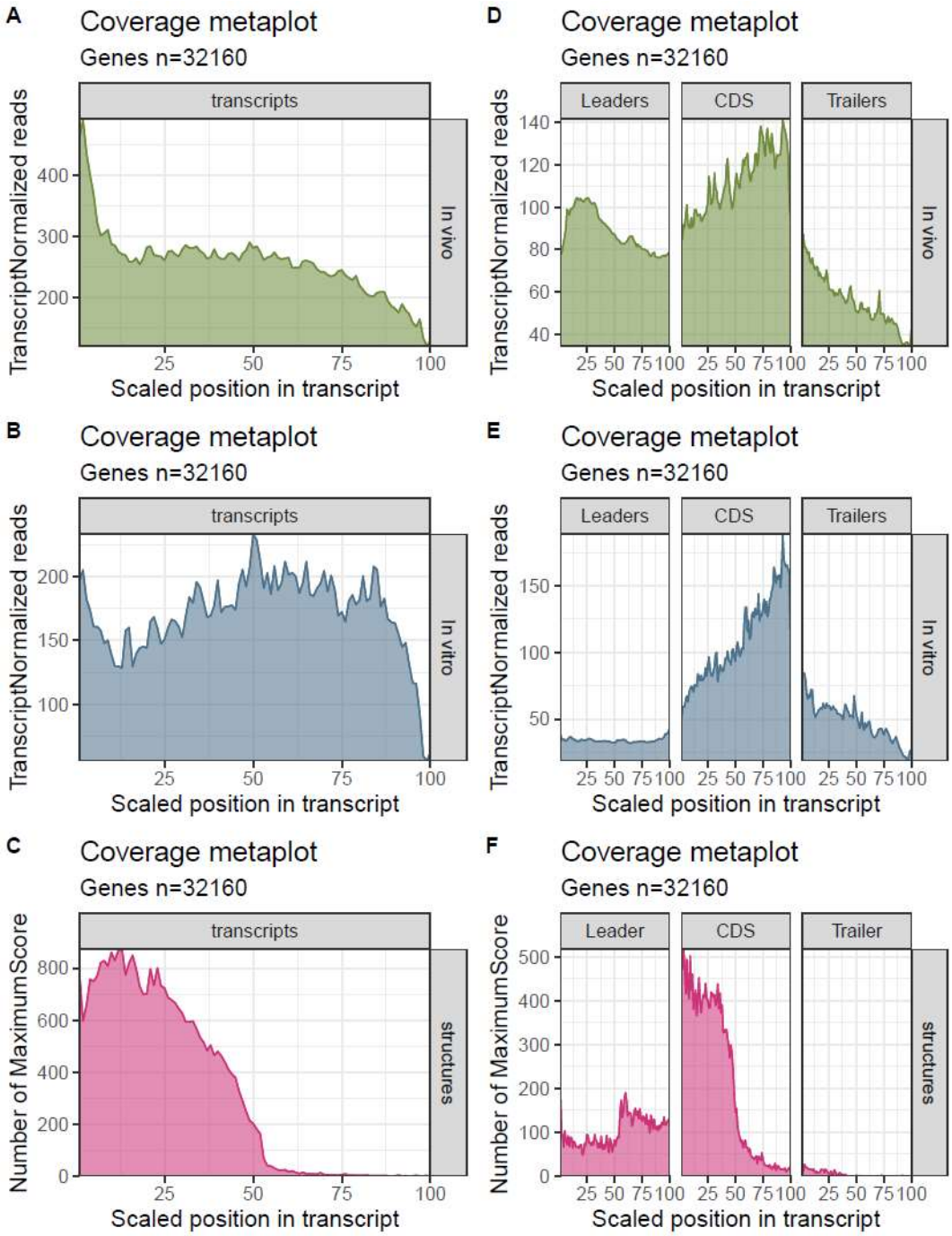

Figure 3-15: A) Coverage meta plot of transcripts *in vivo.* B) Coverage meta plot of transcripts *in vitro.* C) Coverage meta plot of transcripts of the maximum scores from *in silico.* D) Coverage meta plot of leader, CDS and trailer *in vivo.* E) Coverage meta plot of leader, CDS and trailer *in vitro.* F) Coverage meta plot of leader, CDS and trailer of the maximum scores from *in silico.*

Figure 3-15 C presents the coverage meta plot of transcripts for the maximum scores (most negative value of MFE) in the *in silico* dataset. The coverage at each position was calculated by summing the number of the maximum scores for the position in all transcripts. The graph shows that the early positions of the transcripts have more structure while the number of maximum scores significantly declined after the scaled position of ~25 and it almost vanished after the position ~50. It could indicate that the more structure in the beginning of the transcript positions turned to less structure after the second half of the scaled position. This is at odds with the fact that that number of maximum scores cannot become zero at the end position of the transcripts. It could be happened because of picking up the first position of transcript with maximum score when there was multiple position with the maximum score.

Figure 3-15 F shows coverage meta plot transcripts for the maximum scores in the *in silico* data set but this time, divided into leader, CDS and trailer regions. Here again, there is an observation of almost no maximum score at the region of trailers. The explanation here is also the same as previously presented for Figure 3-15 C.

In the area downstream of the TIS from Figure 3-15 F, the highest peaks are seen. It is believed that these peaks are linked to the fact that downstream of the TIS, structures that are supposed to stall the scanning ribosome exist, so that the ribosome can more easily initiate translation [99].

Chapter 4

Conclusion and future work

In this chapter the conclusions of the work are summarized together with recommendations for future work.

4.1 Concluding remarks

This project has been aiming to enhance the understanding of the difference between RNA structure probing approaches; *in vivo*, *in vitro* and *in silico*. To address this, the three sets of data for a zebrafish were compared to each other and similarities and differences were determined.

Quality control of the data were performed prior to analysis and delivered satisfactory results.

It was understood how SHAPE strategy probes RNA structure *in vivo* and *in vitro*

conditions. It was observed that RNA structures are often different in *in vivo* and *in vitro* datasets. This is due to the limitations that exist in both studies. For example, *in vivo* an uncontrolled cellular environment such as the presence of biological components, can lead to changed RNA structure. Also, in *in vitro* experiments, indirect interpretation and linkage to living cells, can provide less-representative results for actual RNA folding and thus negatively affect the resultant structures. Consequently, by comparison of *in vivo* and *in vitro* data, it can be determined how the cellular environment changes the structural pattern of RNA.

An overview of the data comparison for the three sets of data (*in vivo*, *in vitro* and *in silico*) in terms of match or mismatch for the structural information of each transcript region (leader, CDS and trailer) is presented in the following table.

Table 4-1: An overview of the data comparison for each transcript region

Strategies		*In vivo*			*in vitro*			*In silico*		
		Leader	CDS	Trailer	Leader	CDS	Trailer	Leader	CDS	Trailer
In vivo	Leader				✗			✗		
	CDS					✓			✓	
	Trailer						✓			✗
In vitro	Leader	✗						✓		
	CDS		✓						✓	
	Trailer			✓						✗
In silico	Leader	✗			✓					
	CDS		✓			✓				
	Trailer			✗			✗			

Here the ✓ symbol means a match and the ✗ symbol means a mismatch between the data when comparing the structural data of a transcript region for one of the datasets with the corresponding transcript region for another dataset.

Table 4-1 then quickly communicates that the leader region for *in vivo* data mismatch with the *in silico* data leader region while the same region for *in vitro* and *in silico* do match. CDS region data for both *in vitro* and *in vivo* show a proper match with *in silico* data. The trailer transcript region data for both *in vitro* and *in vivo* datasets did not match with *in silico* data of the trailer region.

Comparing *in vivo* and *in vitro* data, CDS and trailer regions match while leader regions in these two datasets do not match.

4.2 **Recommendations for future work**

Probing the RNA folding structures is essential to understand and control the biological structure-dependent characterizes of RNAs. Towards accomplishing this goal, several

researchers have used *in vivo* and *in vitro* studies. Nonetheless both *in vivo* and *in vitro* studies have drawbacks hindering them to fully address this goal. Also in several studies, *in vitro* and *in vivo* datasets for the same structure can have significant mismatch. It appears that the root cause for these discrepancies is that the RNAs do not adopt the same structures *in vivo* as *in vitro*.

Also, the process for probing RNA structural data and relevant handling and analysis of the data today is not perfect since it involves numerous functions in different R packages and various software.

The following list of investigations may be then considered as future work following this master thesis:

- ✓ Proving an ultimate software solution to collect, purify and analyze structural data and export plots and tables of interest for the characterization of RNA structure. Such software will have FASTA and GTF file as the only input and e.g. BAM files, coverage plots of transcripts and matrix of minimum free energies as direct output.
- ✓ Using more efficient reagents in SHAPE_seq experiment capable of distinguishing between structure and superstructures in RNA configurations.
- ✓ Probing RNA structure in an artificial cellular environment as an attempt to provide more representative structural data
- ✓ Developing novel experiments to probe RNA structure using fine-tuned experimental conditions in order to minimize the inconsistencies between dataset

Bibliography

[1] H. Roodashty, “RNA-structure.” [Online]. Available: https://github.com/hanyroze/RNA-structure.git.

[2] M. Cobb, “60 years ago, Francis Crick changed the logic of biology,” *PLoS Biol.*, vol. 15, no. 9, pp. 1–8, 2017.

[3] H. Lodish, A. Berk, P. Zipursky, S Lawrence Matsudaira, D. Baltimore, and J. Darnell, *Molecular Cell Biology*, 4th ed. New York: W. H. Freeman, 2000.

[4] F. Hubé and C. Francastel, “Coding and non-coding RNAs, the frontier has never been so blurred,” *Front. Genet.*, vol. 9, no. APR, pp. 1–5, 2018.

[5] K. E. Watters, A. M. Yu, E. J. Strobel, A. H. Settle, and J. B. Lucks, “Characterizing RNA structures in vitro and in vivo with selective 2′-hydroxyl acylation analyzed by primer extension sequencing (SHAPE-Seq),” vol. 344, no. 6188, pp. 1173–1178, 2015.

[6] K. A. Leamy, S. M. Assmann, D. H. Mathews, and P. C. Bevilacqua, “ Bridging the gap between in vitro and in vivo RNA folding ,” *Q. Rev. Biophys.*, vol. 49, pp. 1–26, 2016.

[7] I. L. Hofacker and P. F. Stadler, “RNA Secondary Structures,” *Bioinformatics-From Genomes to Ther.*, vol. 1, pp. 439–489, 2008.

[8] R. T. Batey, R. P. Rambo, and J. A. Doudna, “Tertiary Motifs in RNA Structure and Foldin,” p. 18, 1999.

[9] H. Li, D. Zhu, C. Zhang, H. Han, and K. A. Crandall, “Characteristics and prediction of RNA structure,” *Biomed Res. Int.*, vol. 2014, 2014.

[10] W. N. Moss, *Computational prediction of RNA secondary structure*, 1st ed., vol. 530. Elsevier Inc., 2013.

[11] J. A. L. and T. R. Cech, “Defining the inside and outside of a Catalytic RNA Molecule,” vol. 245, pp. 276–282.

[12] P. J. Flor, J. B. Flanegan, and T. R. Cech, “A conserved base pair within helix P4 of the Tetrahymena ribozyme helps to form the tertiary structure required for self-splicing,” vol. 8, no. 1, p. 9, 2009.

[13] Y. K. Kim, L. Furic, M. Parisien, F. Major, L. DesGroseillers, and L. E. Maquat, “Staufen1 regulates diverse classes of mammalian transcripts,” *EMBO J.*, vol. 26, no. 11, pp. 2670–2681, 2007.

[14] T. I. Jr and B. C., “How RNA folds,” vol. 293, no. 2, 1999.

[15] D. M. Layton and R. Bundschuh, “A statistical analysis of rna folding algorithms through thermodynamic parameter perturbation,” *Nucleic Acids Res.*, vol. 33, 2005.

[16] A. M. Mohsen, A. T. Khader, D. Ramachandram, and A. Ghallab, “Predicting the

minimum free energy RNA secondary structures using harmony search algorithm," *World Acad. Sci. Eng. Technol.*, vol. 32, no. 8, pp. 923–929, 2009.

[17] H. G. He L, "MicroRNAs: small RNAs with a big role in gene regulation," *Nat Rev Gene*, 2004.

[18] F. Ahmad *et al.*, "RNA-SSPT: RNA Secondary Structure Prediction Tools," *Bioinformation*, vol. 9, no. 17, pp. 873–878, 2013.

[19] M. DH, "Revolutions in RNA secondary structure prediction," *J. Mol. Biol.*, pp. 526–532, 2006.

[20] E. S. Rivas E, "A dynamic programming algorithm for RNA structure prediction including pseudoknots," *J Mol Biol*, pp. 2053–68, 1999.

[21] S. Engelen and F. Tahi, "Predicting RNA secondary structure by the comparative approach: How to select the homologous sequences," *BMC Bioinformatics*, vol. 8, pp. 1–12, 2007.

[22] P. N. Woese C, "Probing RNA structure, function, and history by comparative analysis," *RNA Worl*, pp. 91–117, 1993.

[23] K. E. Watters, A. M. Yu, E. J. Strobel, A. H. Settle, and J. B. Lucks, "Characterizing RNA structures in vitro and in vivo with selective 2'-hydroxyl acylation analyzed by primer extension sequencing (SHAPE-Seq)," *Methods*, vol. 103, pp. 34–48, 2016.

[24] K. M. Weeks, "Advances in RNA structure analysis by chemical probing," *Curr. Opin. Struct. Biol.*, vol. 20, no. 3, pp. 295–304, 2010.

[25] G. Knapp, "Enzymatic approaches to probing of RNA secondary and tertiary structure," vol. 180, pp. 192–212, 1989.

[26] G. M. Culver and H. F. Noller, "Directed hydroxyl radical probing of 16S ribosomal RNA in ribosomes containing Fe(II) tethered to ribosomal protein S20," *Rna*, vol. 4, no. 12, pp. 1471–1480, 1998.

[27] and H. Y. C. Robert C. Spitale, Ryan A. Flynn, Eduardo A. Torre, Eric T. Kool, "RNA Structural Analysis by Evolving SHAPE Chemistry," 2014.

[28] Y. M. I. Behm-Ansmant, M. Helm, "Use of specific chemical reagents for detection of modified nucleotides in RNA," 2011.

[29] L. JB *et al.*, "Multiplexed RNA structure characterization with selective 2'-hydroxyl acylation analyzed by primer extension sequencing," 2011.

[30] D. Loughrey, K. E. Watters, A. H. Settle, and J. B. Lucks, "SHAPE-Seq 2.0: systematic optimization and extension of high-throughput chemical probing of RNA secondary structure with next generation sequencing," p. e165.

[31] J. Talkish, G. May, Y. Lin, J. L. Woolford, and C. J. McManus, "Mod-seq: High-throughput sequencing for chemical probing of RNA structure," *Rna*, vol. 20, no. 5, pp. 713–720, 2014.

[32] Y. Ding, Y. Tang, C. K. Kwok, Y. Zhang, P. C. Bevilacqua, and S. M. Assmann, "In vivo genome-wide profiling of RNA secondary structure reveals novel regulatory features," *Nature*, vol. 505, no. 7485, pp. 696–700, 2014.

[33] S. Rouskin, M. Zubradt, S. Washietl, M. Kellis, J. S., and Weissman, "Genome-wide probing of RNA structure reveals active unfolding of mRNA structures in vivo," 2014.

[34] M. G. Seetin, W. Kladwang, J. P. Bida, and R. Das, "Massively Parallel RNA Chemical Mapping with a Reduced Bias MAP-Seq Protocol," vol. 1086, pp. 95–117, 2014.

[35] P. Ge1 and S. Zhang, "Computational analysis of RNA structures with chemical probing data," *Univ. Cent. Florida*, 2015.

[36] W. KM, "Advances in RNA structure analysis by chemical probing," *Curr Opin Struct Biol*, pp. 295–304, 2010.

[37] G. W. Peattie DA, "Chemical probes for higher-order structure in RNA," *Proc Natl Acad Sci U S A*, pp. 4679–4682, 1980.

[38] N. H. Culver GM, "Directed hydroxyl radical probing of 16S ribosomal RNA in ribosomes containing Fe(II) tethered to ribosomal protein S20," pp. 1471–1480, 1998.

[39] C. H. Spitale RC, Crisalli P, Flynn RA, Torre EA, Kool ET, "RNA SHAPE analysis in living cells," *Nat Chem Biol*, pp. 18–20, 2013.

[40] W. K. McGinnis JL, Dunkle JA, Cate JHD, "The mechanisms of RNA SHAPE chemistry," *J Am Chem Soc*, pp. 6617–6624, 2012.

[41] W. K. Wilkinson KA, Merino EJ, "Selective 2′-hydroxyl acylation analyzed by primer extension (SHAPE)," *Nat Protoc*, pp. 1610–1616, 2006.

[42] R. C. Spitale, P. Crisalli, R. A. Flynn, E. A. Torre, E. T. Kool, and H. Y. Chang, "RNA SHAPE analysis in living cells," *Bone*, vol. 23, no. 1, pp. 1–7, 2008.

[43] W. KM, "Advances in RNA structure analysis by chemical probing," *Curr Opin Struct Biol*, 2010.

[44] I. Behm-Ansmant, M. Helm, and Y. Motorin, "Use of specific chemical reagents for detection of modified nucleotides in RNA," *J. Nucleic Acids*, vol. 2011, 2011.

[45] S. Aviran, J. B. Lucks, and L. Pachter, "RNA structure characterization from chemical mapping experiments," *2011 49th Annu. Allert. Conf. Commun. Control. Comput. Allert. 2011*, pp. 1743–1750, 2011.

[46] S. Aviran *et al.*, "Modeling and automation of sequencing-based characterization of RNA structure," *Proc. Natl. Acad. Sci. U. S. A.*, vol. 108, no. 27, pp. 11069–11074, 2011.

[47] M. J. P. E. Bindewald, M. Wendeler, M. Legiewicz, M.K.Bona, Y. Wang, "Correlating SHAPE signatures with three-dimensional RNA structures," pp. 1688–1696, 2011.

[48] K. M. W. K.-A. Steen, G.M. Rice, “Fingerprinting noncanonical and tertiary RNA structures by differential SHAPE reactivity,” *J. Am. Chem. Soc*, pp. 13160–13163, 2012.

[49] K. M. W. M.J. Smola, J.M. Calabrese, “Detection of RNA-Protein interactions in living cells with SHAPE,” *Biochemistry*, pp. 6867–6875, 2015.

[50] D. Loughrey, K. E. Watters, A. H. Settle, and J. B. Lucks, “SHAPE-Seq 2.0: systematic optimization and extension of high-throughput chemical probing of RNA secondary structure with next generation sequencing,” *Nucleic Acids Res.*, vol. 42, no. 21, p. e165, 2014.

[51] K. M. W. K.E. Deigan, T.W. Li, D.H. Mathews, “Accurate SHAPE-directed RNA structure determination,” *Proc. Natl. Acad. Sci. U.S.A*, pp. 97–102, 2009.

[52] K. M. W. C.E. Hajdin, S. Bellaousov, W. Huggins, C.W.Leonard, D.H. Mathews, “Accurate SHAPE-directed RNA secondary structure modeling, including pseudoknots,” *Proc. Natl. Acad. Sci. U.S.A*, pp. 5498–5503, 2013.

[53] M. T. W. R. Lorenz, D. Luntzer, I.L. Hofacker, P.F. Stadler, “SHAPE directed RNA folding,” *Bioinformatics*, pp. 145–147, 2015.

[54] K. M. W. G.M. Rice, C.W. Leonard, “RNA secondary structure modeling at consistent high accuracy using differential SHAPE,” *RNA*, pp. 846–854, 2014.

[55] K. M. Kutchko, “Multiple conformations are a conserved and regulatory feature of the RB1 5′ UTR,” *RNA 21*, pp. 1274–1285, 2015.

[56] C. T. Woods, “Comparative visualization of the RNA suboptimal conformational ensemble in vivo,” pp. 290–301, 2017.

[57] H. Y. Ouyang, Z., Snyder, M. P. & Chang, “SeqFold: genome-scale reconstruction of RNA secondary structure integrating high-throughput sequencing data,” pp. 377–387, 2013.

[58] D. H. M. M.G. Seetin, “RNA structure prediction: an overview of methods,” *Methods Mol. Biol*, pp. 99–122, 2012.

[59] K. M. Smola, M. J., Christy, T. W., Inoue, K., Nicholson, C. O., Friedersdorf, M., Keene, J. D., Lee, D. M., Calabrese, J. M., and Weeks, “SHAPE reveals transcript-wide interactions, complex structural domains, and protein interactions across the Xist lncRNA in living cells.,” pp. 10322– 10327, 2016.

[60] K. M. Rice, G. M., Leonard, C. W., and Weeks, “RNA secondary structure modeling at consistent high accuracy using differential SHAPE,” *RNA20*, pp. 846– 854, 2014.

[61] H. Y. Lee, B., Flynn, R. A., Kadina, A., Guo, J. K., Kool, E. T., and Chang, “Comparison of SHAPE reagents for mapping RNA structures inside living cells,” pp. 169– 174, 2017.

[62] K. M. Siegfried, N. A., Busan, S., Rice, G. M., Nelson, J. A. E., and Weeks, “RNA motif discovery by SHAPE and mutational profiling,” pp. 959– 965, 2104.

[63] K. M. Busan, S. and Weeks, “Accurate detection of chemical modifications in RNA by mutational profiling (MaP) with ShapeMapper 2,” pp. 143– 148, 2018.

[64] J. Poulsen, L. D., Kielpinski, L. J., Salama, S. R., Krogh, A., and Vinther, “SHAPE Selection (SHAPES) enrich for RNA structure signal in SHAPE sequencing-based probing data,” pp. 1042– 1052, 2015.

[65] T. Shiraki *et al.*, “Cap analysis gene expression for high-throughput analysis of transcriptional starting point and identification of promoter usage,” *Proc. Natl. Acad. Sci. U. S. A.*, vol. 100, no. 26, pp. 15776–15781, 2003.

[66] C. P. Takahashi H, Kato S, Murata M, “CAGE (cap analysis of gene expression): a protocol for the detection of promoter and transcriptional networks.,” *Methods Mol Biol*, pp. 181–200, 2012.

[67] A. S. DING Y, TANG Y, KWOK CK, ZHANG Y, BEVILACQUA PC, “In vivo genome-wide profiling of RNA secondary structure reveals novel regulatory features,” pp. 696–700, 2014.

[68] S. E. KERTESZ M, WAN Y, MAZOR E, RINN JL, NUTTER RC, CHANG HY, “Genome-wide measurement of RNA secondary structure in yeast,” pp. 103–107, 2010.

[69] G. B. LI F, ZHENG Q, VANDIVIER LE, WILLMANN MR, CHEN Y, “Regulatory impact of RNA secondary structure across the Arabidopsis transcriptome,” *Plant Cell*, pp. 4346–4359, 2012.

[70] C. H. WAN Y, QU K, ZHANG QC, FLYNN RA, MANOR O, OUYANG Z, ZHANG J, SPITALE RC, SNYDER MP, SEGAL E, “Landscape and variation of RNA secondary structure across the human transcriptome,” pp. 706–709, 2014.

[71] G. B. ZHENG Q, RYVKIN P, LI F, DRAGOMIR I, VALLADARES O, YANG J, CAO K, WANG LS, “Genome-Wide Double-Stranded RNA Sequencing Reveals the Functional Significance of Base-Paired RNAs in Arabidopsis,” 2010.

[72] M. P, “Un modelo de autómata celular para la evolución de los ácidos nucleicos [A cellular automaton model for the evolution of nucleic acids],” 1992.

[73] M. D. SLOMA MF, “Improving RNA secondary structure prediction with structure mapping data,” *Methods Enzymol.*, pp. 91–114, 2015.

[74] S. Moretti, “In silico experiments in scientific papers on molecular biology,” *Sci. Stud. (St. Bonaventure).*, vol. 24, no. 2, pp. 23–42, 2011.

[75] F. Wieber, A. Henri, P. Lhsp, and N. Université, “Theoretical technologies in an ‘ experimental ’ setting : empirical modeling of proteinic objects and simulation of their dynamics within scientific collaborations around a supercomputer .,” pp. 1–10, 2009.

[76] C. E. Hajdin, S. Bellaousov, W. Huggins, C. W. Leonard, D. H. Mathews, and K. M. Weeks, “Accurate SHAPE-directed RNA secondary structure modeling, including pseudoknots,” *Proc. Natl. Acad. Sci. U. S. A.*, vol. 110, no. 14, pp.

5498–5503, 2013.

[77] C. Ton, D. Stamatiou, V. J. Dzau, and C. C. Liew, “Construction of a zebrafish cDNA microarray: Gene expression profiling of the zebrafish during development,” *Biochem. Biophys. Res. Commun.*, vol. 296, no. 5, pp. 1134–1142, 2002.

[78] H. Yang *et al.*, “Deep mRNA Sequencing Analysis to Capture the Transcriptome Landscape of Zebrafish Embryos and Larvae,” *PLoS One*, vol. 8, no. 5, 2013.

[79] S. Saleem and R. R. Kannan, “Zebrafish: an emerging real-time model system to study Alzheimer’s disease and neurospecific drug discovery,” *Cell Death Discov.*, vol. 4, no. 1, 2018.

[80] K. Howe *et al.*, “The zebrafish reference genome sequence and its relationship to the human genome,” *Nature*, vol. 496, no. 7446, pp. 498–503, 2013.

[81] S. Mathavan *et al.*, “Transcriptome analysis of zebrafish embryogenesis using microarrays,” *PLoS Genet.*, vol. 1, no. 2, pp. 0260–0276, 2005.

[82] C. E. Feitsma H, “Zebrafish as a cancer model,” pp. 685–694., 2008.

[83] D. Kim, B. Langmead, and S. L. Salzberg, “HISAT: A fast spliced aligner with low memory requirements,” *Nat. Methods*, vol. 12, no. 4, pp. 357–360, 2015.

[84] H. Li *et al.*, “The Sequence Alignment/Map format and SAMtools,” *Bioinformatics*, vol. 25, no. 16, pp. 2078–2079, 2009.

[85] S. Anders, P. T. Pyl, and W. Huber, “HTSeq-A Python framework to work with high-throughput sequencing data,” *Bioinformatics*, vol. 31, no. 2, pp. 166–169, 2015.

[86] F. Zanini, “HTSeq: Analysing high-throughput sequencing data with Python,” *Stanford University*. .

[87] M. Lawrence *et al.*, “Software for Computing and Annotating Genomic Ranges,” *PLoS Comput. Biol.*, vol. 9, no. 8, 2013.

[88] Y. Liao, G. K. Smyth, and W. Shi, “FeatureCounts: An efficient general purpose program for assigning sequence reads to genomic features,” *Bioinformatics*, vol. 30, no. 7, pp. 923–930, 2014.

[89] N. A. Fonseca, J. Marioni, and A. Brazma, “RNA-Seq gene profiling - A systematic empirical comparison,” *PLoS One*, vol. 9, no. 9, 2014.

[90] M. Zuker and P. Stiegler, “Optimal computer folding of large RNA sequences using thermodynamics and auxiliary information,” *Nucleic Acids Res.*, vol. 9, no. 1, pp. 133–148, 1981.

[91] “Universität Wien, Theoretical Biochemistry Group, Institute for Theoretical Chemistry.” .

[92] C. B. Do, D. A. Woods, and S. Batzoglou, “CONTRAfold: RNA secondary structure prediction without physics-based models,” *Bioinformatics*, vol. 22, no.

14, 2006.

[93] M. Andronescu, R. Aguirre-Hernández, A. Condon, and H. H. Hoos, “RNAsoft: A suite of RNA secondary structure prediction and design software tools,” *Nucleic Acids Res.*, vol. 31, no. 13, pp. 3416–3422, 2003.

[94] J. N. Zadeh *et al.*, “NUPACK: Analysis and design of nucleic acid systems,” *J. Comput. Chem.*, vol. 32, no. 1, pp. 170–173, 2011.

[95] W. Huber, V. J. Carey, L. Long, S. Falcon, and R. Gentleman, “Graphs in molecular biology,” *BMC Bioinformatics*, vol. 8, no. SUPPL. 6, 2007.

[96] S. Urbanek, H.-J. Bibiko, and M. L. Stefano, “R: A language and environment for statistical computing. The R Foundation for Statistical Computing,” 2014.

[97] L. M. N and G. R, “Analyzing biological data using R: methods for graphs and networks,” 2012.

[98] J. Allaire, “RStudio web page.” .

[99] E. M. Mahen, P. Y. Watson, J. W. Cottrell, and M. J. Fedor, “mRNA secondary structures fold sequentially but exchange rapidly in vivo,” *PLoS Biol.*, vol. 8, no. 2, 2010.

Appendix: R code

Analyzing and plotting HTSeq_count data

```
#Specify a working directory

getwd()
setwd("/Users/haniehroodashty/bin")
list.files()

#load HTseq_count data (two replicates for in vitro)

htseq_invivo_rep1 <- read.delim("result_HTSeq_count_invivo1.txt",header = FALSE)
htseq_invivo_rep2 <- read.delim("result_HTSeq_count_invivo2.txt",header = FALSE)

htseq_invivo = htseq_invivo_rep1
names(htseq_invivo) = c("gene_Id","reads")
m <- dim(htseq_invivo)[1]

#Remove summery text from the 5 last rows of HtSeq_count data
htseq_invivo<- htseq_invivo[1:(m-5),]


#Consider only genes which have reads
high_count <- subset(htseq_invivo, reads > 0)

#Find how many genes have zero,1,2, ... reads(frequency of genes)
read_freq_table = table(htseq_invivo$reads)

read_freq = as.data.frame(read_freq_table)
names(read_freq)[1] = 'reads'

#Find how many genes have 1,2,3,.. (frequency of genes)
high_count_freq_table= table(high_count$reads)
high_count_freq= as.data.frame(high_count_freq_table)

names(high_count_freq)[1]= 'Reads'

high_count_freq$Reads<-as.numeric(as.character(high_count_freq$Reads))

#Save data in corresponding data

```

```
39  saveRDS(high_count_freq, 'high_count_freq_Invivo_rep1.rds')
40
41  library(ggpubr)
42  library(ggplot2)
43  library(scales)
44
45  #ScatterPlot of Inviro replicate one or two
46  #Use high_count-freq_InvitroRep1 or high_count-freq_InvitroRep2 or
    high_count_freq_InvitroData as input data
47  high_count_freq = high_count_freq_Invivo_rep1
48  sp <- ggplot(high_count_freq,x = "Reads",y = "Freq" , aes(x = Reads, y = Freq)) +
    geom_point()
49
50  printplot <- sp +scale_y_continuous( name = "Frequency of genes\n" ,trans =
    'log2',limits = c(1,3000)) + scale_x_continuous(name = "\nNumber of
    reads",trans='log2',limits = c(0.1,100000)) + theme(
51    axis.text = element_text(size = 18), axis.title  = element_text(size = 20 , face = "bold")
52  )
53  print(printplot)
54
55  #Compaire invivo replicate one and two (correlation of two replicates)
56  #Use htseq_invivo_rep1 anad htseq_invivo_rep2 as input data (data are after removing
    summery text)
57  htseq_invivoData1 <- readRDS('high_count_freq_Invitro_rep1.rds')
58  htseq_invivoData2 <- readRDS('high_count_freq_Invitro_rep2.rds')
59
60  #Change the second column of data for merging two data
61  names(htseq_invivoData1)[2]= 'Invivo1'
62  names(htseq_invivoData2)[2]= 'Invivo2'
63
64  mergeInvivo1Invivo2 <- merge(x = htseq_invivoData1, y = htseq_invivoData2, by =
    "gene_Id", all.x = TRUE)
65  mergeNew<- mergeInvivo1Invivo2[,2:3]+1
66
67  sp <- ggscatter(mergeNew ,x = "Invivo1",y = "Invivo2" , add = "reg.line" , add.params =
    list(color = "blue", fill = "lightgray"),
68                conf.int = TRUE ,cor.coef = TRUE, cor.coef.size = 8 ,cor.method =
    "pearson")
69  print(sp +scale_y_continuous( name = "Invivo2\n" ,trans = 'log2',limits = c(1,600000)) +
    scale_x_continuous(name = "\nInvivo1",trans='log2',limits = c(1,100000))+ theme(
70    axis.text = element_text(size = 18), axis.title  = element_text(size = 20 , face = "bold")
71  ) )
72
73  cor.test(mergeNew$Invivo1,mergeNew$Invivo2 , method = "pearson")
```

```
74
75  # Pool two in vitro(in vivo) replicates if replicates have a good correlation
76  # First we need normalization CPM for both replicates
77  # Second add the number of reads rep1 to rep2
78
79  htseq_invivo_rep1$reads <- (( htseq_invivo_rep1$reads ) / sum
    (htseq_invivo_rep1$reads) ) * 10 ^ 6
80  htseq_invivo_rep1$reads <- round (htseq_invivo_rep1$reads ,3)
81
82  # Save data in corresponding data for both replicates
83
84  saveRDS(htseq_invivo_rep1 ,'htseq_invivo_NormRep1.rds')
85  htseq_invivo_NormRep1 <- readRDS('htseq_invivo_NormRep1.rds')
86
87  saveRDS(htseq_invivo_rep2, 'htseq_invivo_NormRep2.rds')
88  htseq_invivo_NormRep2 <- readRDS('htseq_invivo_NormRep2.rds')
89
90  # DO the same as befor for making scatterPlot of invivo data
91  htseq_invivo$reads <- htseq_invivo_NormRep1$reads +
    htseq_invivo_NormRep2$reads
92
93  saveRDS(htseq_invivo, 'htseq_invivoData.rds')
94  htseq_invivoData <- readRDS('htseq_invivoData.rds')
```

Making coverage meta plot of transcripts

```
1   #Specify a working directory
2
3   getwd()
4   setwd("/Users/haniehroodashty/bin")
5
6   #index fasta file
7   indexFa("danRer11.fa")
8   fa <- FaFile("danRer11.fa")
9
10  library(data.table)
11  library(ORFik)
12  library(GenomicAlignments)
13
14  #Load the aligned reads of in vivo(in vitro)
15
16  invivo1 <- GRanges(readGAlignments("NAI-N3_Oblong_S21_L008.bam"))
```

```
invivo2 <- GRanges(readGAlignments("NAI-N3_old_Oblong_S24_L008.bam"))

typeSample = invivo2

seqlevelsStyle(typeSample) <- "NCBI"

# Load the annotation file
library(GenomicFeatures)
txdb <- makeTxDbFromGFF (file = "annotation_danRer11_edited.gtf" )
seqlevelsStyle(txdb) <- "NCBI"

#Filter transcripts
txNames <- filterTranscripts(txdb,100,100,100)

# Get leaders
leaders <- fiveUTRsByTranscript(txdb,use.names=TRUE)[txNames]

# Get the CDS grouped by transcript
cds <- cdsBy(txdb,"tx", use.names = TRUE)[txNames]

# Get the trailers
trailers <- threeUTRsByTranscript(txdb,use.names=TRUE)[txNames]

# Get the exons grouped by transcript
tx <- exonsBy(txdb, by = "tx", use.names = TRUE)[txNames]

#Make coverage plot of one transcript by ggplot
tx_cvg <- coverageByTranscript(typeSample,tx,ignore.strand=TRUE)

a = IRanges::IntegerList(tx_cvg)
l <- lengths(tx_cvg,use.names = FALSE)
l <- rep.int(seq.int(length(l)), l)
res <- data.table(count= unlist(a) , genes= l)

#Give one gene (first or second ,..)
gene <- 1
test_gene<- res[genes==gene,]
test_gene[, position := cumsum(genes)]

#Plot one gene (count across position)
library(ggplot2)
plot <- ggplot(test_gene, aes(x = position, y = count)) + geom_bar(stat = "identity") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1,
                                   vjust = 0.5)) + labs(title = paste("gene:", names(tx)[gene])) +
```

```
    ylab("Averaged counts")

print(plot)

# Get windowCoverage in scale=100 for all transcripts (all width of transcripts are divided to 100 (so under 100 width is ignored))

seqlevelsStyle(tx) <- "NCBI"
coverage <- scaledWindowPositions(tx,typeSample)
coverageTran <- windowCoveragePlot(coverage = coverage,scoring ="transcriptNormalized")
print(coverageTran)

# Get windowCoverage in scale=100 for regions per transcripts
coverageLeader <- scaledWindowPositions(leaders,typeSample)
coverageLeader[, `:=` (fraction="In vivo replicate one",feature="transcripts")]

coverageCds <- scaledWindowPositions(cds,typeSample)
coverageCds[, `:=` (fraction="In vivo replicate one",feature="transcripts")]

coverageTrailers <- scaledWindowPositions(trailers,typeSample)
coverageTrailers[, `:=` (fraction="In vivo replicate one",feature="transcripts")]

coverage <- rbindlist(list(coverageLeader,coverageCds,coverageTrailers))
coverageRegions <- windowCoveragePlot(coverage = coverage ,scoring ="transcriptNormalized,")
print(coverageRegions)

# Multiple plot in one image

A<- ggarrange(coverageTran , coverageRegions,
          ncol = 1, nrow = 2,labels = c ("A","B"),
          heights = c(1,1))

# Normalization CPM  for both replicates in vivo(in vitro), htseq_invitro_rep1 is HTSeq_count for replicate one
names(txNames)=c ('gene_Id')
txNames$gene_Id <- sub("\\.\\d+$", "", txNames$gene_Id)# have the same  gene_Id for Invivo and txNames Data
htseq_invivo_rep1 <- merge(htseq_invivo_rep1, txNames, by='gene_Id')

coverageTran$score <- (( coverageTran$score ) / sum (htseq_invivo_rep1$reads) ) * 10 ^ 6
coverageTran$score <- round(coverageTran$score,3)
```

```
98  saveRDS(coverageTran, 'coverageTran_invivo_NormRep1.rds')
99
100 coverageLeader$score <- (( coverageLeader$score ) / sum (htseq_invivo_rep1$reads) )
    * 10 ^ 6
101 coverageLeader$score <- round(coverageLeader$score,3)
102 saveRDS(coverageLeader, 'coverageLeader_invivo_NormRep1.rds')
103
104 coverageCds$score <- (( coverageCds$score ) / sum (htseq_invivo_rep1$reads) ) * 10
    ^ 6
105 coverageCds$score <- round(coverageCds$score,3)
106 saveRDS(coverageCds, 'coverageCds_invivo_NormRep1.rds')
107
108 coverageTrailers$score <- (( coverageTrailers$score ) / sum (htseq_invivo_rep1$reads)
    ) * 10 ^ 6
109 coverageTrailers$score <- round(coverageTrailers$score,3)
110 saveRDS(coverageTrailers, 'coverageTrailer_invivo_NormRep1.rds')
111
112 # Do the Normalization CPM for replicate two as before
113 # Add two replicates, make coverage plot for regions per transcripts
114 # Make coverage plot for invivo data as before
115 coverageLeader_invivo$score <- coverageLeader_invivo_NormRep1$score +
    coverageLeader_invivo_NormRep2$score
116 saveRDS(coverageLeader_invivo, 'coverageLeader_invivoData.rds')
117
118 coverageCds_invivo$score <- coverageCds_invivo_NormRep1$score +
    coverageLeader_invivo_NormRep2$score
119 saveRDS(coverageCds_invivo, 'coverageCds_invivoData.rds')
120
121 coverageTrailer_invivo$score <- coverageTrailers_invivo_NormRep1$score +
    coverageTrailers_invivo_NormRep2$score
122 saveRDS(coverageTrailer_invivo, 'coverageTrailer_invivoData.rds')
123
124
125 ##Number of reads for each gene from plot_htseq_count.r file
126
127 htseq_invivo_rep1 <- read.delim("result_HTSeq_count_NAI-
    N3_Oblong_S21_L008_yes.txt",header = FALSE)
128 htseq_invivo_rep <-
    read.delim("result_HTSeq_count_invitro_MCE_Oblong_yes.txt",header = FALSE)
129
130 htseqTypesample = htseq_invitro_rep1
131
132 names(htseqTypesample) = c("gene_Id","reads")
133 n <- dim(htseqTypesample)[1]
```

```
htseqTypesample <- htseqTypesample[1:(n-5),]


# Single gene plot for comparing  with vienna
htseqTypesample <- htseqTypesample [order(htseqTypesample$reads,decreasing =
TRUE),]
pickThisone <- txNames[1]
# Get transcriptName from geneName by function geneToTxnames from
functionMaster.R
sortedTranscripts <- geneToTxnames(txdb,htseqTypesample$gene_Id)

# Pick first transcript from sorted list of transcripts
pickThisone <- sortedTranscripts[1]

library(ORFik)

plotGene <- coveragePerTiling(tx[pickThisone],typeSample,as.data.table = TRUE)
singleGeneplot <- windowCoveragePlot(plotGene, scoring = "sum", title = pickThisone )
print(singleGeneplot)

# Vienna print top gene by function outputFasta from functionMaster.R
outputFasta(fa,tx,gene = pickThisone)
```

Analyzing and plotting minimum free energy

```
# Get vienna structure
# 1 get fasta file of gene we are looking at
# 2 send that to viennaRNA
# 3 look at structure

#Specify a working directory
getwd()
setwd("/Users/haniehroodashty/bin")
list.files()

# Load the annotation file
library(GenomicFeatures)
txdb <- makeTxDbFromGFF (file = "annotation_danRer11_edited.gtf" )

#index fasta file
library(Rsamtools)
```

```
17  library(ORFik)
18  indexFa("danRer11.fa")
19  fa <- FaFile("danRer11.fa")
20
21  seqlevelsStyle(txdb)  <- seqlevelsStyle(fa)
22  txNames <- filterTranscripts(txdb, 100, 100, 100)
23
24  # Get leaders
25  leaders <- fiveUTRsByTranscript(txdb,use.names=TRUE)[txNames]
26
27  # Get the CDS grouped by transcript
28  cds <- cdsBy(txdb,"tx", use.names = TRUE)[txNames]
29
30  # Get the trailers
31  trailers <- threeUTRsByTranscript(txdb,use.names=TRUE)[txNames]
32
33
34  #tx filtered by txNames
35  tx <- exonsBy(txdb, use.names = TRUE)[txNames]
36  tx <- tx[widthPerGroup(tx) > 300]
37  a <- extractTranscriptSeqs(fa, transcripts = tx)
38  writeXStringSet(a, filepath = "tx.fasta")
39
40  # Now do perl scripts to get minimum free energy
41  library(pander)
42  system(p("/Adams_vienna_explain.sh -f tx.fasta -o ", getwd()))
43  # Now we have csvs folder which contains am.csv file (MFE of each position of
    transcripts)
44  # Now do analysis
45  setwd("/Users/haniehroodashty/bin/vienna/csvs")
46
47  library(data.table)
48  hits<- fread("am.csv" ,nrow = length(tx), header = FALSE , fill=TRUE )
49
50
51  # Check that all was made
52  if(nrow(hits) != length(tx)) stop("did not create all")
53
54
55  h <- hits[,-1]
56  m <- setDT(melt(t(h)))
57
58  best <- m[, .(which.min(value)), by = Var2]
59  bestValue<- m[, .(score = min(value, na.rm = TRUE)), by = Var2]
```

```
60
61  saveRDS(best, 'best_filter.rds')
62  best <- readRDS('best.rds')
63
64  positions <- IRanges(best$V1, width = 1)
65
66  # Now create window coverage plot
67  # grl is the minimum free energy position per gene
68  # pmapfromtranscript means tx coord -> genomic coord
69
70  library(GenomicFeatures)
71
72  grl <- ORFik:::pmapFromTranscriptF(positions, tx[hits_filter$V1], removeEmpty =
    TRUE)# check this is correct
73
74
75  #In silico scatterplot
76  meansOfAllGenes <- data.table ( gene_Id = hits_filter[,1], Silico =
    rowMeans(hits_filter[,c(-1)], na.rm = T))
77
78  names(meansOfAllGenes)[1] = 'gene_Id'
79
80  meansOfAllGenes$gene_Id <- seq.int(nrow(meansOfAllGenes))
81
82
83  # Make window coveragePlot of for all transcripts
84
85  txCoverage <- scaledWindowPositions(tx, grl)
86  txCoverage[, `:=` (fraction = "structures", feature = "transcripts")]
87
88  outName <- paste("structure_sum", ".pdf", sep="")
89  windowCoveragePlot(txCoverage, scoring = "sum")
90  outName <- paste("structure_zscore", ".pdf", sep="")
91  windowCoveragePlot(txCoverage, output = outName ,scoring = "zscore")
92  outName <- paste("structure_transcriptNorm", ".pdf", sep="")
93  windowCoveragePlot(txCoverage ,scoring = "transcriptNormalized")
94
95  # Make coverage plot for regions per transcript
96
97  coverageLeader <- scaledWindowPositions(leaders,grl)
98  coverageLeader[, `:=` (fraction="In silico",feature="transcripts")]
99
100 coverageCds <- scaledWindowPositions(cds,grl)
101 coverageCds[, `:=` (fraction="In silico",feature="transcripts")]
```

```
102
103 coverageTrailers <- scaledWindowPositions(trailers,grl)
104 coverageTrailers[, `:=` (fraction="In silico",feature="transcripts")]
105
106 coverage <- rbindlist(list(coverageLeader,coverageCds,coverageTrailers))
107 coverageRegions <- windowCoveragePlot(coverage = coverage ,scoring
    ="transcriptNormalized,")
108
109 # Coverage plot of all transcripts for Minimum free energy
110
111 cov <- m[,-1]
112
113 colnames(cov) <- c("genes", "count")
114 cov <- cov[!is.na(cov$count),]
115 perGroup <- cov[,.N, by = genes]$N
116 if ( length(tx) != length(perGroup)) stop('you messed up the filtering of tx, they are not
    the same!')
117 cov[, ones := rep.int(1L, length(genes))]
118 cov[, position := cumsum(ones), by = genes]
119
120 cov$ones <- NULL
121
122 scaleTo = 100
123 scoring = "meanPos"
124
125 cov[, scalingFactor := ((scaleTo)/perGroup[genes])]
126 cov[, position := ceiling(scalingFactor * position)]
127
128 cov[position > scaleTo]$position <- scaleTo
129 groupFPF <- quote(list(genes, position))
130 res <- cov[, .(score = mean(count, na.rm = TRUE)), by = eval(groupFPF)]
131 res$feature <- 'transcripts'
132 res$fraction <- 'In silico'
133
134 outName <- paste("MFE_distribution", ".pdf", sep="")
135 coverageTx_plot_insilico<-windowCoveragePlot(res, scoring = "transcriptNormalized",
    colors = c("violetred3 ","deeppink4"))+ theme_bw(base_size = 15)
    +ylab("TranscriptNormalized MFE") + theme(legend.position = "none")
136 print(coverageTx_plot_insilico)
137
138
139 # Make leader,cds and trailer metacoverage with all free energy positions
140 # Per gene, find which position cds start, trailers start in transcript coordinate for tx
141 names(cds) <- sub(x = names(cds), pattern = '_', replacement = '.')
```

```
142 names(tx) <- sub(x = names(tx), pattern = '_', replacement = '.')
143 names(trailers) <- sub(x = names(trailers), pattern = '_', replacement = '.')
144
145 cdsTrans <- ORFik:::asTX(cds,tx)
146 cdsStarts <- ORFik:::startSites(cdsTrans)
147 cdsStartsLong <- cdsStarts[cov$genes]
148
149 trailerTrans <- ORFik:::asTX(trailers,tx)
150 trailerStarts <- ORFik:::startSites(trailerTrans)
151 trailerStartsLong <- trailerStarts[cov$genes]
152
153 # Split hits per gene, into leader, cds, trailer
154
155 leaderhits <- cov[position < cdsStartsLong, ]
156
157 cdshits <- cov[position >= cdsStartsLong & position < trailerStartsLong, ]
158 cdshits$position <- cdshits$position - cdsStarts[cdshits$genes] + 1
159
160 # The position of cov here is not sacled here
161 trailerhits <- cov[position >= trailerStartsLong, ]
162 trailerhits$position <- trailerhits$position - trailerStarts[trailerhits$genes] + 1
163
164 perGroupLead <- leaderhits[,.N,by = genes]
165 perGroupCds <- cdshits[,.N,by = genes]
166 perGroupTrail <- trailerhits[,.N,by = genes]
167
168 scaleTo = 100
169 scoring = "meanPos"
170
171 leaderhits <- merge(x = leaderhits, y = perGroupLead, by = "genes")
172 leaderhits$scalingFactor <- scaleTo/leaderhits$N
173 leaderhits[, position := ceiling(scalingFactor * position)]
174 leaderhits$feature <- 'Leaders'
175
176 cdshits <- merge(x = cdshits, y = perGroupCds, by = "genes")
177 cdshits$scalingFactor <- scaleTo/cdshits$N
178 cdshits[, position := ceiling(scalingFactor * position)]
179 cdshits$feature <- 'Cds'
180
181
182 trailerhits <- merge(x = trailerhits, y = perGroupTrail, by = "genes")
183 trailerhits$scalingFactor <- scaleTo/trailerhits$N
184 trailerhits[, position := ceiling(scalingFactor * position)]
185 trailerhits$feature <- 'Trailers'
```

```
186
187 fractionhits <- rbindlist(list(leaderhits,cdshits,trailerhits))
188 fractionhits[position > scaleTo]$position <- scaleTo
189 fractionhits$N<- NULL
190 fractionhits$scalingFactor <- NULL
191
192
193 groupFPF <- quote(list(genes, position, feature))
194
195 resPerTx <- fractionhits[, .(score = mean(count, na.rm = TRUE)), by = eval(groupFPF)]
196 resPerTx$fraction <- 'In silico'
197
198 # Make coverage plot for leader,cds and trailers of minimum free energy
199 outName <- paste("MFE_distributionLeaderCdsTrailer", ".pdf", sep="")
200 coverageRegionTx_plot_insilico<- windowCoveragePlot(resPerTx ,scoring =
    'transcriptNormalized', colors = c("violetred3 ","deeppink4"))+ theme_bw(base_size =
    15) +ylab("TranscriptNormalized MFE") + theme(legend.position = "none")
201 print(coverageRegionTx_plot_insilico)
202
203
204 # Density plot of insilico data for best(most negative MFE) and mean score
205 # For best
206 BestScore_Position <- merge(x = best, y = bestValue, by = "Var2")
207
208 LeaderBestPosition <- BestScore_Position[V1 < cdsStarts, ]
209 LeaderBestPosition[, `:=` (feature="Leader")]
210
211 CdsBestPosition <- BestScore_Position[V1 >= cdsStarts & V1 < trailerStarts, ]
212 CdsBestPosition[, `:=` (feature="Cds")]
213
214 TrailerBestPosition<- BestScore_Position[V1 >= trailerStarts, ]
215 TrailerBestPosition[, `:=` (feature="Trailer")]
216
217 BestScoreFeature <-
    rbindlist(list(LeaderBestPosition,CdsBestPosition,TrailerBestPosition))
218 ggplot(BestScoreFeature,aes(x=score, fill = feature)) + geom_density(alpha=0.5)+
    theme(
219   axis.text = element_text(size = 18), axis.title  = element_text(size = 20 , face =
    "bold"),legend.title = element_text(size=20),legend.text= element_text(size=15))
220 saveRDS(BestScoreFeature, 'BestScoreFeature.rds')
221
222 # For mean score
223
```

```
LeaderMeanPerGene <- leaderhits[, .(MeanScore = mean(count, na.rm = TRUE)), by = genes]
LeaderMeanPerGene[, `:=` (feature="Leader")]

CdsMeanPerGene <- cdshits[, .(MeanScore = mean(count, na.rm = TRUE)), by = genes]
CdsMeanPerGene[, `:=` (feature="Cds")]

TrailerMeanPerGene <- trailerhits[, .(MeanScore = mean(count, na.rm = TRUE)), by = genes]
TrailerMeanPerGene[, `:=` (feature="Trailer")]

MeanScoreFeature <- rbindlist(list(LeaderMeanPerGene,CdsMeanPerGene,TrailerMeanPerGene))
saveRDS(MeanScoreFeature, 'MeanScoreFeature.rds')
ggplot(MeanScoreFeature ,aes(x=MeanScore, fill = feature)) + geom_density(alpha=0.5) + theme(
  axis.text = element_text(size = 18), axis.title  = element_text(size = 20 , face = "bold"),legend.title = element_text(size=20),legend.text= element_text(size=15))
```

Calculating correlation between different datasets

```
#Correlation of replicate one and replicate two in vivo (in vitro ) by computing difference percentage of count reads
#Correlation of in vivo and in vitro data
#Correlation of in silico and in vivo as well as in vitro  by computing difference percentage of hits

#Load the aligned reads of in vivo (in vitro)

invivo1 <- GRanges(readGAlignments("NAI-N3_Oblong_S21_L008.bam"))
invivo2 <- GRanges(readGAlignments("NAI-N3_old_Oblong_S24_L008.bam"))

seqlevelsStyle(invivo1) <- "NCBI"
seqlevelsStyle(tx) <- "NCBI"

# Replicate one
invivo1_new <- ORFik:::convertToOneBasedRanges(invivo1)

txHits <- countOverlaps(tx, invivo1_new)
leaderHits <- countOverlaps(leaders, invivo1_new)/txHits
```

```
cdsHits <- countOverlaps(cds, invivo1_new) /txHits
trailerHits <- countOverlaps(trailers, invivo1_new) /txHits
Hits_Invivorep1 <- data.frame(leaderHits, cdsHits, trailerHits)

# Percentage of hits per leader, cds and trailer in each transcript
Hitst_Invivorep1<- saveRDS(Hits_Invivorep1,'Hits_Invivorep1.rds')

# Replicate two
seqlevelsStyle(invivo2) <- "NCBI"

invivo2_new <- ORFik:::convertToOneBasedRanges(invivo2)

txHits <- countOverlaps(tx,invivo2_new)
leaderHits <- countOverlaps(leaders, invivo2_new) / txHits
cdsHits <- countOverlaps(cds, invivo2_new) / txHits
trailerHits <- countOverlaps(trailers, invivo2_new)/ txHits
Hits_Invivorep2 <- data.frame(leaderHits, cdsHits, trailerHits)

# Percentage of hits per leader, cds and trailer
saveRDS(Hits_Invivorep2, 'Hits_Invivorep2.rds')
Hits_Invivorep2 <- readRDS('Hits_Invivorep2.rds')
# Difference percentage between two replicates
# Replace na value with zero
Hits_Invivorep1[is.na(Hits_Invivorep1)] <- 0
Hits_Invivorep2[is.na(Hits_Invivorep2)] <- 0

# Find total difference percentage for each transcripts between invivo replicate one and
replicate two
totalScore <- abs(Hits_Invivorep1$leaderHits - Hits_Invivorep2$leaderHits) +
abs(Hits_Invivorep1$cdsHits - Hits_Invivorep2$cdsHits) +
  abs(Hits_Invivorep1$trailerHits - Hits_Invivorep2$trailerHits)
df <- data.frame(totalScore)
df$totalScore <- round (df$totalScore ,2)
df$totalScore <- (df$totalScore) *100

# Plot of percentage difference between in vivo replicates for each transcript
sp <- ggplot(df, aes(x = totalScore)) + geom_freqpoly()
sp +scale_x_continuous(name = "\n% Difference of count reads", limits = c(0,300)) +
scale_y_continuous(name = "Frequency of genes\n")+theme(
  axis.text = element_text(size = 18), axis.title  = element_text(size = 20 , face = "bold")
)

# Pool two replicates in vivo (in vitro)
txHitsInvivo1 <- countOverlaps(tx, invivo1_new)
```

```
59  txHitsInvivo2 <- countOverlaps(tx, invivo2_new)
60  txHitsInvivo <- (txHitsInvivo1 / sum (htseq_invivo_rep1$reads) + txHitsInvivo2  / sum
    (htseq_invivo_rep2$reads))* 10 ^ 6
61
62  leaderHitsInvivo1 <- countOverlaps(leaders, invivo1_new)
63  leaderHitsInvivo2 <- countOverlaps(leaders, invivo2_new)
64  LeaderHitsInvivo <- (leaderHitsInvivo1 / sum (htseq_invivo_rep1$reads) +
    leaderHitsInvivo2  / sum (htseq_invivo_rep2$reads))* 10 ^ 6
65  LeaderHitsInvivo <- LeaderHitsInvivo /txHitsInvivo
66
67  cdsHitsInvivo1 <- countOverlaps(cds, invivo1_new)
68  cdsHitsInvivo2 <- countOverlaps(cds, invivo2_new)
69  cdsHitsInvivo <- (cdsHitsInvivo1 / sum (htseq_invivo_rep1$reads) + cdsHitsInvivo2 /
    sum (htseq_invivo_rep2$reads))* 10 ^ 6
70  cdsHitsInvivo <- cdsHitsInvivo / txHitsInvivo
71
72  trailerHitsInvivo1 <- countOverlaps(trailers, invivo1_new)
73  trailerHitsInvivo2 <- countOverlaps(trailers, invivo2_new)
74  trailerHitsInvivo <- (trailerHitsInvivo1 / sum (htseq_invivo_rep1$reads) +
    trailerHitsInvivo2 / sum (htseq_invivo_rep2$reads))* 10 ^ 6
75  trailerHitsInvivo <- trailerHitsInvivo / txHitsInvivo
76
77  # percentage of count reads per leader, cds and trailer in each transcript
78  Hits_Invivo <- data.frame(LeaderHitsInvivo, cdsHitsInvivo, trailerHitsInvivo)
79  saveRDS(Hits_Invivo, 'Hits_Invivo.rds')
80  Hitst_Invivo <- readRDS('Hits_Invivo.rds')
81
82  # Do the same script above for in vitro replicates and get in vitro data hits
83  # Difference percentage between Invitro and Invivo for each transcript
84  # Replacing na value with zero
85
86  Hits_Invivo[is.na(Hits_Invitro)] <- 0
87  Hits_Invivo[is.na(Hits_Invivo)] <- 0
88
89  # Find total difference percentage for each transcripts between in vivo and in vitro data
90  totalScore <- abs(Hits_Invivo$LeaderHitsInvivo - Hits_Invitro$leaderHitsInvitro) +
    abs(Hits_Invivo$cdsHitsInvivo - Hits_Invitro$cdsHitsInvitro) +
91    abs(Hits_Invivo$trailerHitsInvivo - Hits_Invitro$trailerHitsInvitro)
92  df <- data.frame(totalScore)
93  df$totalScore <- round (df$totalScore ,2)
94
95  # Plot of percentage difference between in vivo and in vitro data
96  sp <- ggplot(df, aes(x = totalScore)) + geom_freqpoly()
```

```
97  sp +scale_x_continuous(name = "Score difference") + scale_y_continuous(name =
    "Frequency of genes")
98
99  # Hits In silico
100 # Using perGroupLead,perGroupCds and perGroupTrail from ViennaScript.R
101
102 perGroupLead <- leaderhits[,.N,by = genes]
103 perGroupCds <- cdshits[,.N,by = genes]
104 perGroupTrail <- trailerhits[,.N,by = genes]
105 # Using cov data from ViennaScript.R
106 pergroupTx <- cov[, .N,by = genes]
107
108 # Find percentage of hits for regions per transcript
109
110 Txlead<- merge(x = pergroupTx, y= perGroupLead, by ="genes", all=TRUE)
111 TxleadCds <- merge ( x = Txlead , y= perGroupCds , by ="genes", all = TRUE)
112 names(TxleadCds) <- c("genes","Tx","leader","Cds")
113
114 TxleadCdsTrai <- merge (x = TxleadCds , y= perGroupTrail ,by ="genes", all = TRUE)
115 colnames(TxleadCdsTrai)[colnames(TxleadCdsTrai)=="N"] <- "Trailer"
116
117 TxleadCdsTrai[is.na(TxleadCdsTrai)] <- 0
118
119 TxleadCdsTrai$leader <-TxleadCdsTrai$leader/TxleadCdsTrai$Tx
120 TxleadCdsTrai$Cds <-TxleadCdsTrai$Cds/TxleadCdsTrai$Tx
121 TxleadCdsTrai$Trailer <-TxleadCdsTrai$Trailer/TxleadCdsTrai$Tx
122 Hits_AllInsilico <- TxleadCdsTrai
123 saveRDS(Hits_AllInsilico, 'Hits_AllInsilico.rds')
124 Hitst_AllInsilico <- readRDS('Hits_AllInsilico.rds')
125
126 # Difference percentage between Insilico and Invitro or Invivo and Insilico
127 # Replace na value with zero
128
129 totalScore <- abs(Hits_Invivo$LeaderHitsInvivo - Hits_AllInsilico$leader) +
    abs(Hits_Invivo$cdsHitsInvivo - Hits_AllInsilico$Cds) +
130   abs(Hits_Invivo$trailerHitsInvivo - Hits_AllInsilico$Trailer)
131 df <- data.frame(totalScore)
132 df$totalScore <- round (df$totalScore ,2)
133
134 sp <- ggplot(df, aes(x = totalScore)) + geom_freqpoly()
135 sp +scale_x_continuous(name = "Score difference") + scale_y_continuous(name =
    "Frequency of genes")
```

Description of functions used in the R scripts

```
# Two function are using in Coverage_plot.R
# Get transcriptName from geneName
geneToTxnames <- function(txdb,geneName){
  names <-transcriptLengths(txdb)
  res <- names$gene_id[names$gene_id %in% geneName]
  res <- chmatch(as.character(geneName),as.character(names$gene_id))
  return(names$tx_name[res])
}


# Vienna print top gene
outputFasta <- function(fa,refrence,gene=NULL){
  seqlevelsStyle(refrence) <- seqlevelsStyle(fa)
  if (is.null(gene) ){
    seq <- extractTranscriptSeqs(fa,refrence)
  } else {
    seq <- extractTranscriptSeqs(fa, refrence[gene])
  }

  writeXStringSet(seq, filepath = "geneNP_059333.1.fasta")
  return()
}
```