UNIVERSITY OF ZAGREB
FACULTY OF ELECTRICAL ENGINEERING AND COMPUTING

MASTER's THESIS No. 2468

Detection of Modified Nucleotide Clusters in Nanopore Sequenced RNA Reads

Josipa Lipovac

Zagreb, June 2021.

Umjesto ove stranice umetnite izvornik Vašeg rada.

Da bi ste uklonili ovu stranicu obrišite naredbu `\izvornik`.

ZAHVALA

CONTENTS

1. Introduction

RNA or Ribonucleic acid is a single-stranded molecule. It is known that cells contain different forms of RNA - messenger RNA (mRNA), transfer RNA (tRNA), and ribosomal RNA (rRNA). Researches have shown that although RNA is a single-stranded molecule it can form double-stranded structures. Also, single-stranded RNA can form secondary structures so that the RNA molecule overlaps and thus forms hairpin loops. This property can be of great importance, especially in the case of the ability of tRNA to bind to the correct mRNA sequence during translation (Clancy ((2008))).

In order to quantify specific structures of the RNA molecule, different approaches are used. One such approach is the modification of exposed nucleotides. The modification process is done artificially in the laboratory. After the modification process and once the RNA structure is unfolded, it is necessary to observe characteristic patterns of positions of modified nucleotides for each cluster of structures by looking at its primary structure.

Third-generation sequencing technologies provide reads longer than previous technologies. Although this fact facilitates the analysis of RNA structures, there is an additional disadvantage of this approach - the presence of a high error rate in fragments. Modifying nucleotides increases error rate because existing methods for translation of raw input signal to a sequence of nucleotides recognize only canonical nucleotides (Technologies ((2020))).

This thesis presents methods for reducing the error rate in the data and methods for detecting clusters of modified nucleotides. The error rate reduction was performed using the different approaches and procedures that using detailed analysis of data found significant positions on the reads. Detection of clusters of modified nucleotides is performed using various methods such as K-means, RSM algorithm, Non-negative matrix factorization, and Graph spectral clustering. All implementations of these methods can be found at the link.

2. Overview

2.1. Third-generation sequencing

Sequencing is the process of determining the order of individual nucleic base - adenine, cytosine, guanine, timid and uracil - within RNA or DNA chain. Sequencing results are used to determine the genomic itself, the establishment of the related and evolution of species, determining new genes and their association of diseases, determining the potential goals for medicines, and metagenomics - the research of the microbial community directly from the environment (M. Šikić ((2013))).

In recent years, different genomic sequencing technologies have become revolutionary in the field of mutation detection of genetic diseases. Oxford Nanopore Technologies released Nanopore sequencer that performs sequencing in real-time and carries out sequencing by predicting sequences from electric current patterns knows as “squiggle”, which are affected by the bases inside the Nanopore. Nanopore uses flow cell to allow massively parallel sequencing and the flow cell is made of an electrically resistant membrane that has thousands of tiny pores, each with a diameter of one nanometer (hence the name) (A. Di Giorgio ((2019))).

Third-generation sequencing and mapping unlike second-generation technologies, which produce reads a few hundred base-pairs (bp) long, generate over 10 000 bp reads or map over 100 000 bp molecules. This fact has greatly improved the analysis of genome structure (Lee et al. ((2016))).

Even so, third-generation sequencing contains a high rate of errors which may further hamper the detection of groups of modified nucleotides. There are two main reasons why is the error rate in the case of third-generation sequencing high. First, the DNA or RNA passes through the pore very quickly so if the sequence contains homopolymers (more of the same nucleotides) it is hard to detect how many nucleotides are there. Second, modifications make it more difficult. Modifications such as methylations change the size of nucleotides and the electrical current so base callers in these cases have difficulties recognizing nucleotides.

2.2. Related work

Detection of clusters of modified nucleotides is a general problem in the field of bioinformatics. Current work in this area presents different approaches that still do not provide fully satisfactory methods. In this thesis, three different approaches were studied in particular: RSM and ANN, DRACO and DREEM algorithms. Their approaches are presented in the following sections. The work used in the development of these algorithms served as inspiration for the methods used in this thesis described in the Section 5.

2.2.1. RSM and ANN Algorithm

In the paper *Detecting and phasing minor single-nucleotide variants from long-read sequencing data* (Feng et al. ((2021))) the work about detecting and phaseing minor single-nucleotide variants or SNVs is presented. The method used for detecting SNVs is also applicable to the problem of detecting RNA modifications. This method is based on solving the main problem and that is to distinguish between real SNVs and sequencing errors. It is assumed that sequencing errors are independent and it is unlikely that multiple sequencing errors occur together on the same read. On the other hand, it is assumed that the occurrence of SNVs is dependent. For that reason, for detection of SNVs, conditional substitution rate is used.

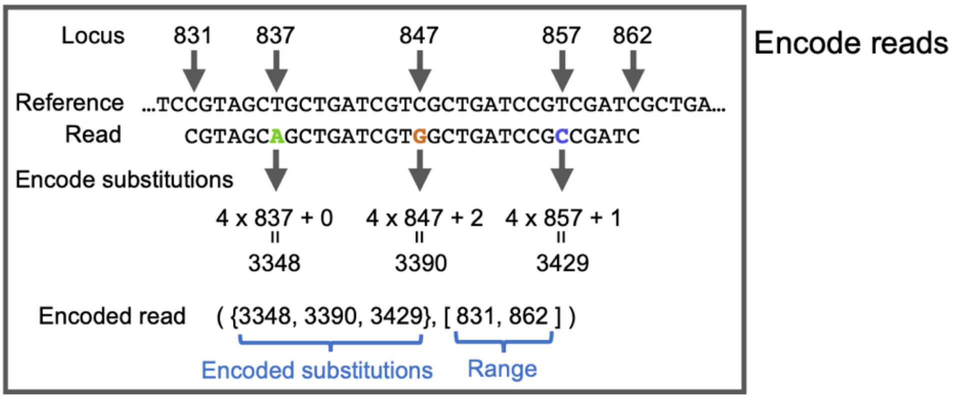


Figure 2.1: Encoding modifications on reads (Feng et al. ((2021)))

In the first step of the whole procedure, it is necessary to encode all modifications on the reads (Figure 2.1). After that, the reads are represented by encoded substitutions and the reference coverage interval of that read.

The algorithm used for detection was named Random Subspace Maximization or RSM algorithm. To detect real SNVs it is necessary to observe p different dependent loci. Locus is the position on the reference where modification occurred. Enumerating all combinations of p loci to get maximal conditional substitution rate is infeasible. So, RSM is used to estimate the maximal conditional substitution rate.

The first step in the RSM algorithm is to measure the similarity between every two reads. The similarity is defined as modified *Jaccard index* (equation 2.1).

$$Jaccard(R_i, R_j) = \frac{|S_i \cap S_j|}{|(S_i \cup S_j) \cap [4max(b_i, b_j), 4min(e_i, e_j) + 3]|} \tag{2.1}$$

In equation 2.1 S_i represents a set of encoded substitutions, b is the start position of the reads coverage interval and e is the end position of it.

After all similarities are calculated, the next step is to calculate w x m subspaces (C_{ij} - equation 2.2). w is user defined and represents number of similar reads, m is the total nuber of reads.

$$C_{ij} = S_i \cap S_j \tag{2.2}$$

The next step is to estimate maximal conditional probability for substitution x_k from C_{ij} using equations 2.3. And after that the detection of SNVs is over.

$$\begin{aligned} H_{C_{ij}}(x_k) &= \max_{\{x_{g1}, x_{g2}, \ldots x_{gp}\} \subset C_{ij}} \{\hat{Pr}(X_k = x_k | X_{g1} = x_{g1}, X_{g2} = x_{g2}, ..., X_{gp} = x_{gp})\} \\ &= \max_{x_{g1}, x_{g2}, \ldots x_{gp}} \{\frac{|\{t | \{x_k, x_{g1}, x_{g2}, ..., x_{gp}\} \subset (S_t \cap C_{ij})\}|}{|\{t | \{x_{g1}, x_{g2}, ..., x_{gp}\} \subset (S_t \cap C_{ij}), k \in [b_t, e_t]\}|}\} \\ \hat{H}(x_k) &= max_{C_{ij}}(\hat{H}_{C_{ij}}(x_k)) \end{aligned} \tag{2.3}$$

Once the SNVs are detected the next step is clustering SNVs using the ANN algorithm. ANN stands for Adaptiv Nearest Neighbours. Firstly, to reduce noise level, only detected modifications for each read were retained as it is shown in equation 2.4 .

$$\tilde{S_i} = S_i \cap \{DetectedSNVs\} \tag{2.4}$$

The main idea of the ANN algorithm is that all loci should be homogeneous by piling up the reads in each cluster. The homogeneous locus is the locus that satisfies the following condition in equation 2.5.

$$\tilde{P}_r(X_k = x_k) = \frac{|\{i|x_k \in \tilde{S}_i\}|}{\sum_{d=0}^{3} |\{i|4k+d \in S_i\}| + |\{i|r_{ik} = t_k\}| \in [0, p_{lim}] \cup [p_{lim}, 1]} \tag{2.5}$$

After the condition is checked the next step is for a $read_i$ to sort q most similar reads according to *Jaccard index* and kept discarding the most dissimilar one until all loci covered by $read_i$ are homogenous or maximal coverage of the loci is smaller than some threshold. Once all loci are homogenous, the consensus sequence is recorded as a draft contig. And then *Jaccard index* of each read with all the draft contigs is calculated, and the read is assigned to the contig with largest *Jaccard index*.

2.2.2. DREEM

DREEM algorithm is presented in paper *Determination of RNA structural diversity and its role in HIV-1 RNA splicing* (Tomezsko et al. ((2020))). DREEM as a name comes from the Detection of the RNA folding Ensembles using Expectation–Maximization. As the name itself says, the main idea of this algorithm is the expectation-maximization algorithm. In this case, RNA was modified with dimethyl sulfate (DMS). The function of DMS is to add methyl groups to the unpaired adenines and cytosines of RNA molecules. During reverse transcription, using TGIRT-III, the presence of a methyl adduct is read. Methyl adduct marks positions by incorporating random mutations in the complementary (c)DNA. Each of the resulting reads is represented as a binary readout of mutations and matches, which is the input for DREEM. The mutations that are observed on a single DNA molecule actually correspond to the DMS-accessible bases on the parent RNA molecule.

There are two main challenges in detecting heterogeneity: (1) DMS modification rate is low (a base has a probability of 2-10% of being modified); (2) the DMS modification rate per base is sensitive to the local chemical environment (not all bases are equally reactive to DMS). The DREEM algorithm groups reads from each structure into distinct clusters. If two individual bases are DMS-reactive but never mutated on an identical read it can be concluded that there are at least 2 groups. DREEM tries to identify patterns of mutations on reads and clusters using the EM algorithm. The modification rate per base for each cluster is determined by iteratively maximizing a log-likelihood that is computed using a multivariate Bernoulli mixture model. As opposed to the algorithm for detecting SNVs (RSM algorithm), in this paper it is assumed that the mutation probabilities are independent of each other.

Concerning the problem of reads not covering the whole reference, the bit representing the base not covered by the read was set as . and the base with low quality was set as ? . If the fraction of non-informative bits ('.', '?' and N) in the bit vector is greater than info_thresh, the bit vector is removed. After the filtering, all the non-informative bits are set to 0 in the remaining bit vectors.

K is the number of structures - clusters present in sample. The model is parametrizied by: μ = { $\boldsymbol{\mu_1}$, ... , $\boldsymbol{\mu_k}$ }. $\boldsymbol{\mu_k}$ = (μ_{k1} , ... , μ_{kD}) are the mutation probabilities of the cluster *k*. $\boldsymbol{\pi}$ = { π_1 , ... , π_K } are the mixing proportions of the *K* clusters in which π_k quantifies the proportion of reads in cluster *k*.

The probability of a base being mutated in cluster *k* is

$$Pr(\boldsymbol{x}_{ni} = 1|\boldsymbol{\mu}_k) = \mu_{ki} \tag{2.6}$$

The probability of observing a read x_n from cluster *k* is:

$$Pr(\boldsymbol{x}_{ni}|\boldsymbol{\mu}_k) = \prod_{i=1}^{D} \mu_{ki}^{x_{ni}}(1-\mu_{ki})^{1-x_{ni}} \tag{2.7}$$

Reads that contain mutations within three bases of each other are very rare - they occur at a frequency close to the sequencing error rate. This is probably due to the reverse transcriptase falling off the template when encountering adjacent methylations. All rare reads containing mutations within three bases of each other are removed and set ***S*** is calculated. This set contains all reads with allowable mutations.

The model parameters $\boldsymbol{\mu}$ and $\boldsymbol{\pi}$ are randomly initialized. After the initialization step EM algorithm is performed. The expectation and the maximization step are executed in loop one after another until convergence of the log-likelihood.

In the expectation step two calculations are performed: (1) the responsibilities of the clusters are computed - the reads are assigned to clusters; (2) the expected log-likelihood of observing data X and latent variables Y={Y_{nk}} is computed.

In the maximization step of the algorithm, the model parameters are re-estimated by maximizing the expected value of the likelihood. After that, the mixing proportion of clusters is updated.

2.2.3. DRACO

DRACO algorithm is presented in paper *Genome-scale deconvolution of RNA structure ensembles* (Morandi et al. ((2021))). DRACO is an abbreviation of Deconvolution of Alternative RNA COnformations. The authors of the algorithm and paper based their work on the previously presented DREEM algorithm. In this case, RNA was modified with dimethyl sulfate (DMS) with the same procedure that is described in the previous section. Although based on the procedures of the DREEM algorithm, the DRACO algorithm is actually quite different from the DREEM algorithm at its core and is based on the use of graph spectral clustering and fuzzy clustering.

In the paper, firstly, the main limitations of the DREEM algorithm are listed: (1) the maximum number of RNA conformations (groups) that can be searched for is user-defined; (2) it can handle only experiments in which the sequencing reads cover the entire length of the target RNA. In the case of DRACO algorithm there are also some limitations: (1) it can also be applied to the analysis of short transcripts that can be covered by a single sequencing read, or of smaller target regions in larger transcripts; (2) the number of clusters is determined by eye inspection of the eigengap plot or by cut-off to the magnitude of eigengaps; (3) sequencing reads are hard clustered by k-means - one read can only be assigned to the one RNA conformation (group). The authors assume that it is likely that certain bases are single stranded in more than one conformation (group).

The whole algorithm has several steps. A graphical representation of these steps is shown in Figure 2.2 .

In the first step of the algorithm, it is necessary to perform side windowing along the transcript. The window size, in this case, is equal to 90% of the median read with 5% increments. The second step is generating a mutation map using only reads that cover the entire window. Mutated bases are represented as 1, and 0 otherwise. By using this mutation map, in the third step, the graph is built. In the graph, each vertex is a base of the transcript and edges are connecting two vertices. These edges are weighted proportionally to the number of reads in which the two connected bases have been observed to co-mutate. In step 4, from the adjacency matrix of an obtained graph, the normalized Laplacian matrix is calculated. It is used for spectral clustering. Then, the null model is derived by repeating the same procedure after shuffling the mutations in the original mutation map. Analysis of the eigengaps allows identifying the number of informative ones - the number of the RNA conformations (groups). After the number of conformations is defined, in the next step, fuzzy clustering is performed by using a custom graph cut approach. In the last step, consecutive windows that show the compatible number of clusters are merged and then reads are re-assigned to the respective clusters.

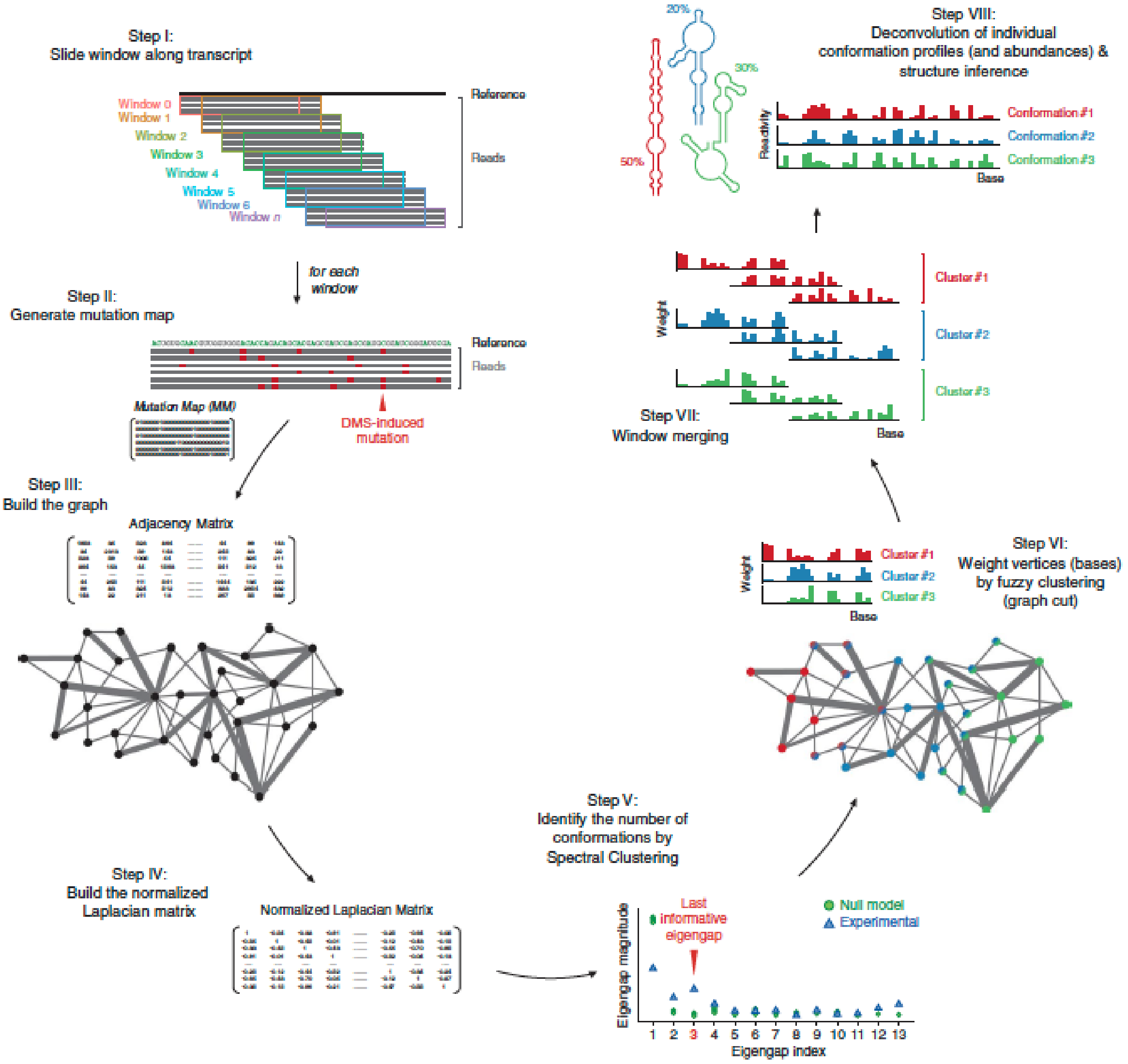


Figure 2.2: Overview of the DRACO algorithm (Morandi et al. ((2021)))

3. Dataset

3.1. Tetrhymena data

The first type of data used in this research is the tetrahymena ribozyme, a group I intron from *Tetrahymena*. Genome Institute of Singapore obtained the data. Sequencing of the data was performed with the Oxford Nanopore MinION third-generation sequencer. Modifying nucleotides was performed by Wan Yue, Ph.D., and Jong Ghut Ashley Aw.

There were three types of the obtained data: unmodified reads, reads acquired by sequencing slightly modified RNA sequences and reads acquired by sequencing quite modified RNA sequences. These reads were the result of base calling given raw signals with albacore v2.3.3. All raw signals are transformed into reads. The number of reads per group is shown in the Table 3.1. The reference was RNA sequence that is 421 base pair long.

unmodified	slightly modified	quite modified
20149	51764	9951

Table 3.1: Number of reads for Tetrahymena data

The whole preprocessing procedure including aligning reads to a reference, polishing and then detection of modifications was done by Ivona Martinović as part of her Bachelor Thesis (Martinović ((2020))).

3.2. Riboswitch data

Riboswitch data was also used in this work. A riboswitch is a regulatory segment of an mRNA molecule. Riboregulation is the use of ribonucleic acid sequences encoded within mRNA that directly affect the expression of genes encoded in the full transcript. Riboswitches bind small molecules with high affinity and specificity, they bind metabolites or metal ions as ligands and regulate mRNA expression by forming alternative structures in response to this ligand binding. Riboswitches are most often located in the 5' untranslated region (5' UTR - a stretch of RNA that is before the translation start site) of the mRNA (Edwards and Batey ((2010))).

These data were also obtained by Wan Yue, Ph.D., and Jong Ghut Ashley Aw. The data contained different riboswitches that are listed in the Table 3.2.

RIBOSWITCH	
GUA	Guanine
FMN	Flavin mononucleotide
SAM	S-Adenosyl methionine
TPP	Thiamine pyrophosphate
MRPS21	Mitochondrial Ribosomal Protein S21

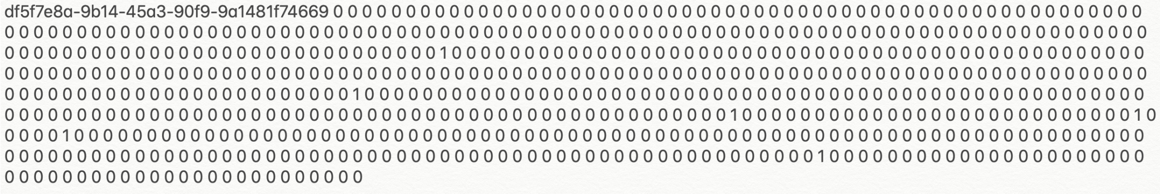
Table 3.2: Types of riboswitches in data

TPP data were used the most in the development, and in some cases, additional verification was performed with MPRS21 and GUA data. There were two sets for every kind of riboswitches. Two sets for TPP data were: TPP riboswitch with ligand and without ligand. As already mentioned, the binding of the TPP molecule to mRNA changes the RNA structure. For this reason, the main assumption was that there were at least two structures in the merged data - one from the sample without ligand and one from the sample with the ligand.

It is important to note that these data are not previously basecalled. To determine whether the base was modified or not, the electrical current from nanopore sequencing was used. Then, the matrix with rows representing the individual strands, and columns representing the positions were generated. The length of all RNA individual strands in TPP data was 635 positions, MPRS21 624 and GUA 642 positions.

3.3. Simulated data

The previous two sections describe the real data representing reads of the modified RNA structures. The downside of this data is that the exact number of structures that



```
df5f7e8a-9b14-45a3-90f9-9a1481f74669 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

Figure 3.1: An example of one individual RNA strand in matrix

need to be detected is not known. For riboswitch data, there was an assumption that there were two structures, but even that was an unreliable fact.

For this reason, it was necessary to generate simulated data on which the implemented methods will be tested. When simulating the data, it has been tried to follow as closely as possible that the simulated data are similar to those that can be obtained in the real world.

Simulated data were created for TPP riboswitch data as behind Tetrahymena data. Real references were used in the simulation. The simulated reads were the length of the real references. Initially, the minimum and the maximum number of mutations per read were initialized. The types of simulated reads are presented in Figure 3.2.

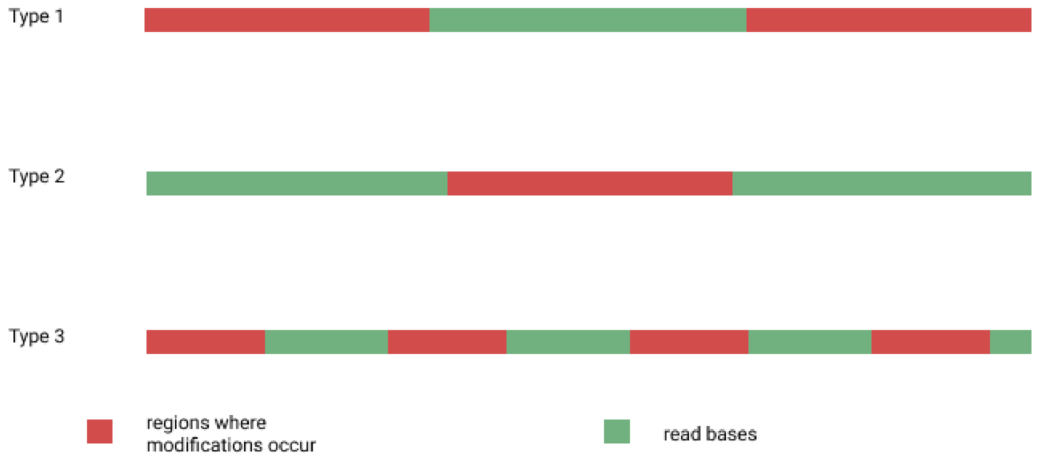


Figure 3.2: Graphical representation of simulated read types

For Tetrahymena data reads that could be divided into three clusters are generated, more precisely, the expected number of structures was three. The minimum number of mutations per read was 80 and the maximum number was 110. Different clusters had mutations at different read positions. The first cluster was defined by mutations that were at the start and end positions of the read. The second cluster contained reads that had mutations only in the middle positions. The third cluster had mutations in every

20 or so positions of the read. About 2000 reads were created for each of the clusters.

Simulated data for the TPP riboswitch reference were created for 2 clusters. Analyzing the TPP data, it was concluded that these data were modified in a rather small percentage. Therefore, both the minimum and the maximum number of mutations for the simulated data were less than in the case of Tetrahymena data. The minimum number was 10, while the maximum number was 80. Three types of such simulated data were generated. In the first type, one cluster had mutations only at starting and ending positions, and the other only at the middle positions. In the second type, one cluster also had modifications in the starting and ending positions, and the other every 20 or so positions. The third type was a combination of clusters with modifications only in the middle positions and with modifications every 20 or so positions.

4. Data analysis

In this thesis, most of the work was done with riboswitch data because there was an assumption of the expected number of clusters. Before performing all the methods, it was necessary to understand the data well.

When analyzing the real data, it was concluded that not all positions on the reads are of equal importance. Certain positions indicate affiliation to one of the clusters. As can be seen on Figure 4.1, there are positions where the number of modifications is significantly higher. Slightly less than 40,000 reads were taken into account. The largest number of modifications is at position 321 wherein almost every case a modification occurs. On the other hand, there is a much larger number of positions where the number of modifications is less than 100.

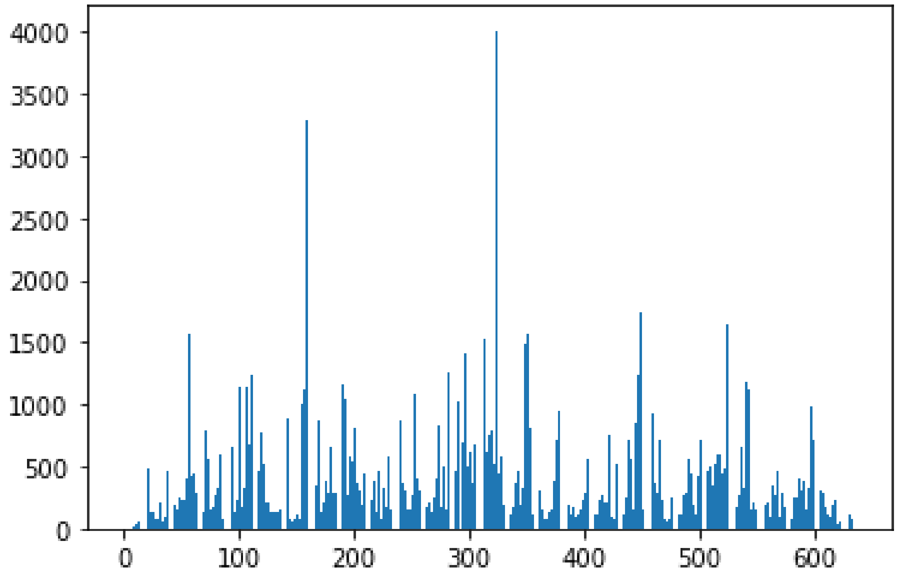


Figure 4.1: Modifications per positions on TPP data

Since it was concluded that TPP riboswitch data are particularly rare, several analyzes were conducted to determine whether there are positions that can be cluster representatives.

4.1. Cumulative sums

Analysis using cumulative sums was performed on TPP riboswitch data - TPP with ligand and TPP without ligand. Firstly, a graph of cumulative sums by positions for data with ligand and without ligand was plotted (Figure 4.2). The first point of the graph represents the total number of modifications of all reads at position 1. The second point represents the sum of the total number of modifications of all reads at position 1 and the total number of modifications of all reads at position 2, and so on.

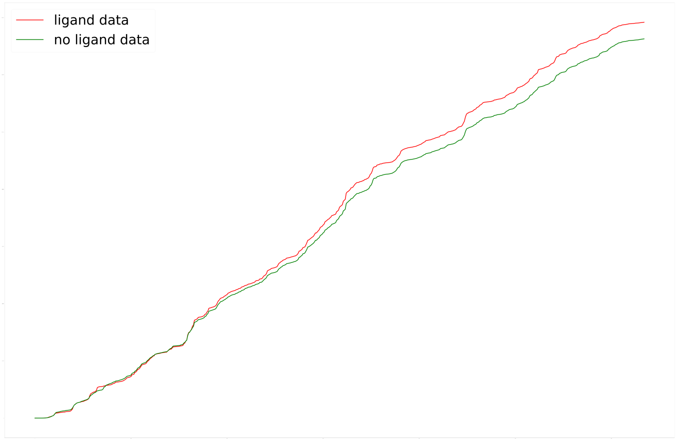


Figure 4.2: Cumulative sums for ligand and non ligand data

Since the plotted graph did not give much information, the read positions were divided into overlapping windows of length 7. The cumulative sums were calculated at the window level and the obtained results are shown in the Figure 4.3.

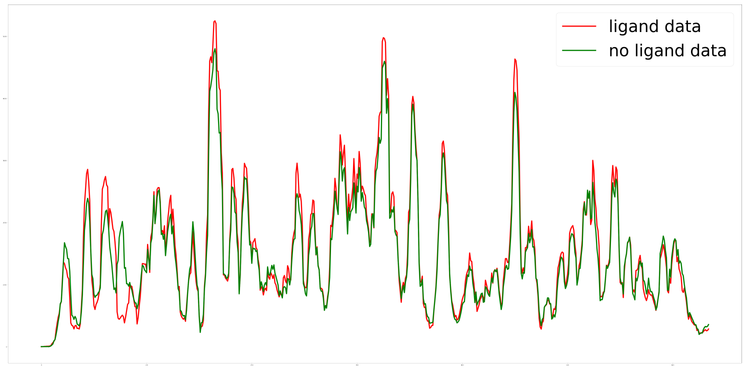


Figure 4.3: Cumulative sums per windows of length 7 for ligand and non ligand data

It was then concluded that a good indicator would be the differences between the ligand and nonligand data for each of the points shown in the previous graph. In addition, a null model was created consisting of reads whose modifications were perturbed and the same procedure was performed with it - the differences of cumulative sums by windows for ligand and nonligand data were subtracted. A comparison of the results obtained for the real and perturbed data is shown in the Figure 4.4.

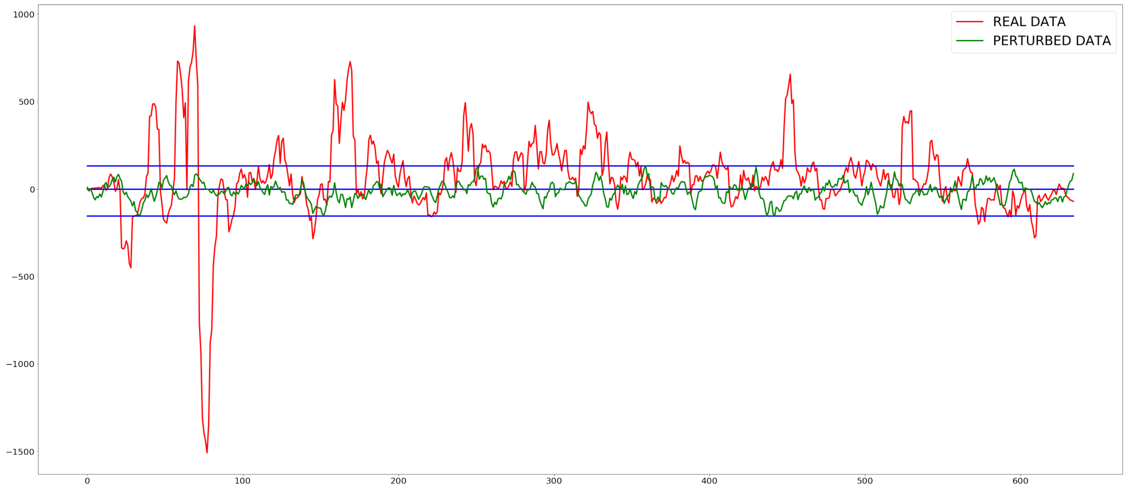


Figure 4.4: Comparison between differences for real data and for perturbed data

On the graph, the red line shows the data for the differences between the ligand and non ligand real data. The green line shows the differences for the null model. The top blue line shows the maximum value for null model results, the middle blue line represents 0, while the bottom represents the minimum value for null model results.

From this graphical representation, it is easy to conclude that this data contains a large amount of noise, which can further complicate the detection of clusters.

4.2. Detection Of Significant Positions

In the previous section, it was shown that the data that need to be clustered contain a large amount of noise. In order to try to reduce the noise level in the data, the implementation of a method that finds only significant positions on reads has been attempted.

Chunks of 100 reads are taken. Read representations (read*) are obtained using windowing. If the window of 10 positions in the original read contains at least one modification, the label in read* is 1, otherwise is 0. For a better understanding, an illustrative example of obtaining a read representation using windowing with a window of size 10 is shown in Figure 4.5

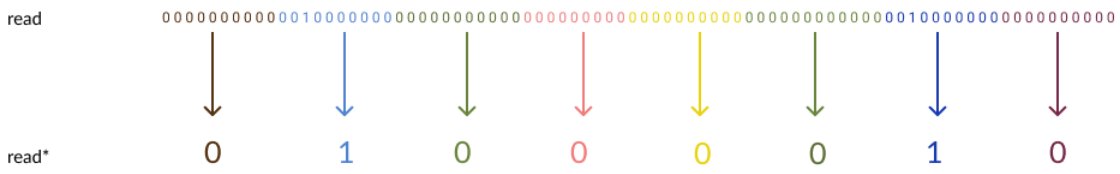


Figure 4.5: Windowing: window length 10

Then, all positions of reads* are summed in one vector. From that obtained vector chunk mean value is calculated. After that, chunk representations are constructed. Every position in chunk representation was calculated as the difference between the sum of that positions of all reads in the chunk and the chunk mean value. The whole procedure is presented graphically in Figure 4.6.

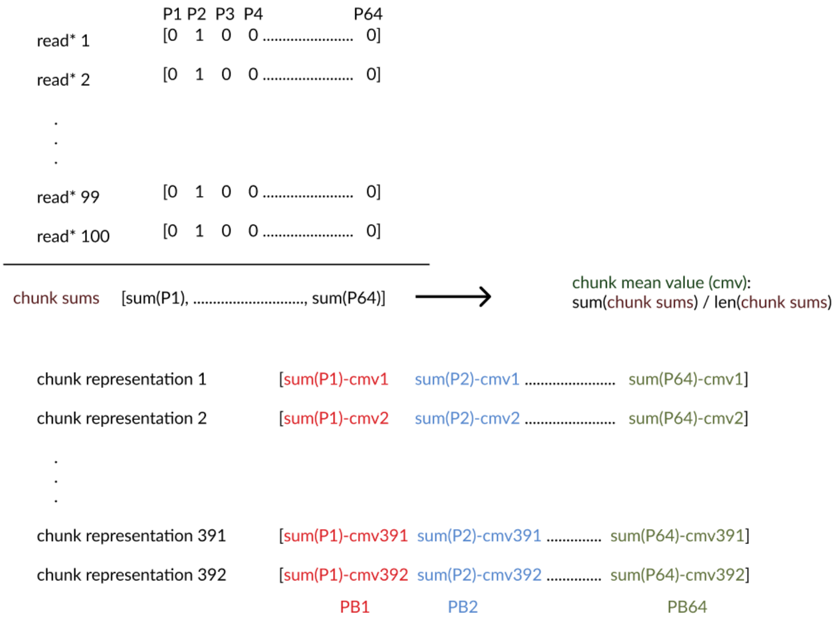

P1 P2 P3 P4 P64
read* 1 [0 1 0 0 .... 0]
read* 2 [0 1 0 0 .... 0]
read* 99 [0 1 0 0 .... 0]
read* 100 [0 1 0 0 .... 0]
chunk sums [sum(P1), ..., sum(P64)]
chunk mean value (cmv):
sum(chunk sums) / len(chunk sums)
chunk representation 1 [sum(P1)-cmv1 sum(P2)-cmv1 .... sum(P64)-cmv1]
chunk representation 2 [sum(P1)-cmv2 sum(P2)-cmv2 .... sum(P64)-cmv2]
chunk representation 391 [sum(P1)-cmv391 sum(P2)-cmv391 .... sum(P64)-cmv391]
chunk representation 392 [sum(P1)-cmv392 sum(P2)-cmv392 .... sum(P64)-cmv392]
PB1
PB2
PB64

Figure 4.6: The method for finding significant positions

In the next step for every column of chunk representations (PB in Figure 4.6) box plot is plotted. Example of box plots obtained for TPP data is shown in Figure 4.7. The same procedure is repeated with simulated data and the box plots are shown in Figure 4.8. As it can be seen in the figures, the distribution of box plots for real and simulated data differs significantly which is an additional indicator of excessive noise in the real data.

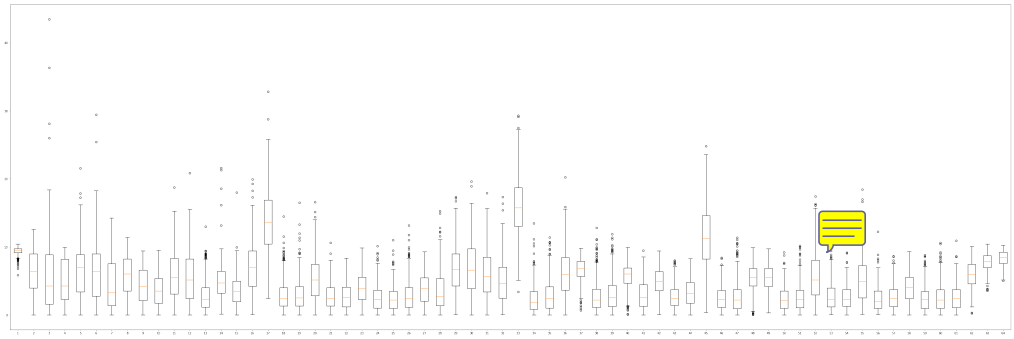


Figure 4.7: Box plots of columns of the chunk representations for the real data

A graphical representation of the box plots for the simulated data was used to detect significant positions. In the Figure 4.9 interesting regions are rounded off. Interesting regions differ from each other in the mean values of the datasets belonging to them. The main idea is to find 10 significant positions. Accordingly, it is necessary to find 10 important regions and take representatives of these regions. The representative was the median value of a particular region. Interesting regions were obtained by first calculating the median difference between the data from every two adjacent positions. The 9 biggest differences were the boundaries of the regions and the regions themselves were constructed concerning them.

With the obtained representatives of the regions, clustering of reads was performed using different methods.

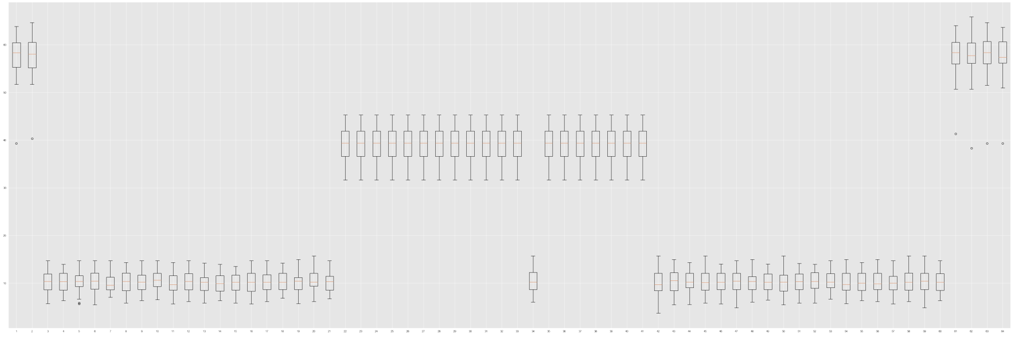


Figure 4.8: Box plots of columns of the chunk representations for one kind of the simulated data

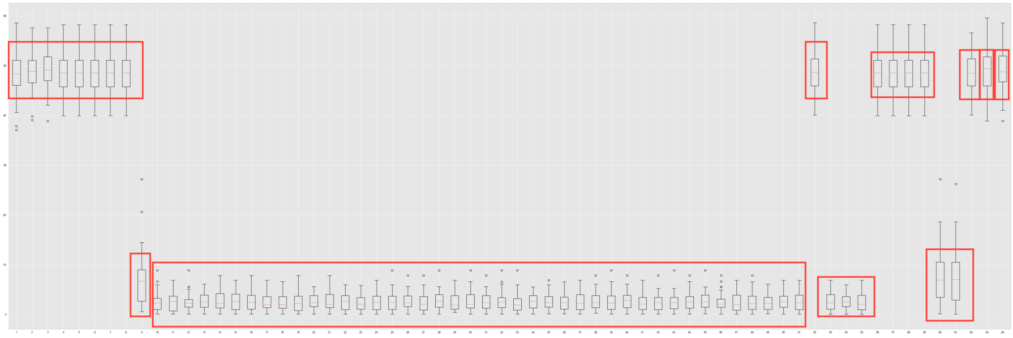


Figure 4.9: Box plots of columns of the chunk representations for one kind of the simulated data with rounded interesting regions

4.3. Windowing analysis

In order to further understand TPP data, additional analysis was conducted. The reads are represented by overlapping windows of length 10 and the following graphs are drawn:

- the average number of modifications in window for all reads (Figure 4.10)
- the maximum number of modifications in window for all reads (Figure 4.11)
- the number of reads with 3 or more modifications in particular window (Figure 4.12)

In this section, graphs are presented only for the case of the window of size 10. The analysis was also performed for both windows of sizes 5 and 15, as well as for the case of non-overlapping windows. Unfortunately, the whole analysis did not provide the significant information needed to fully understand data distribution.

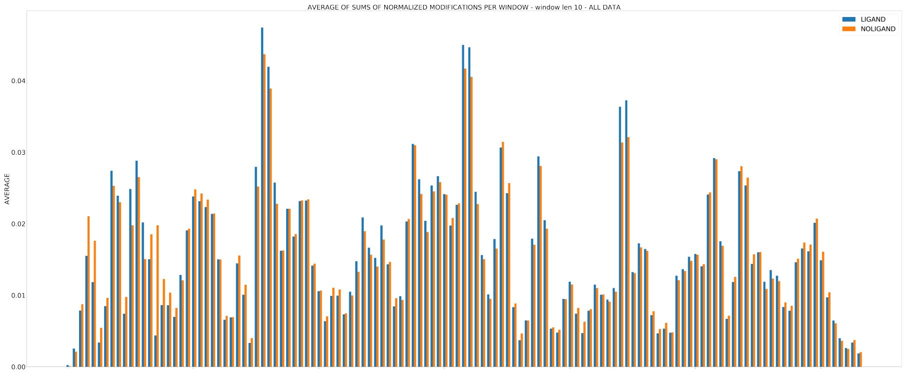


Figure 4.10: The average number of modifications in window for all reads

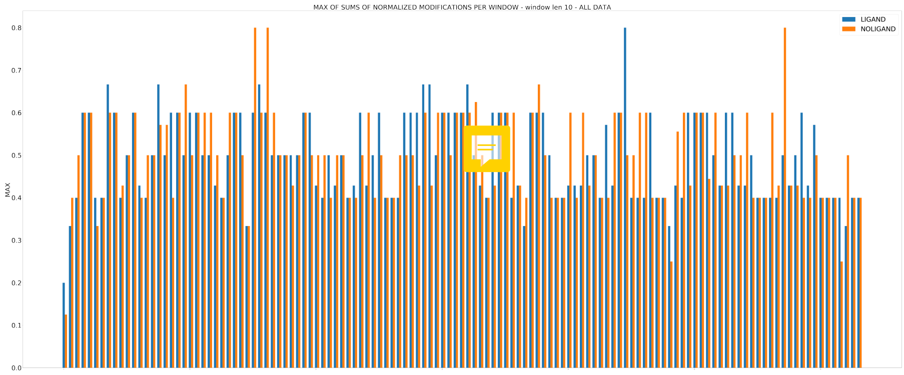


Figure 4.11: The maximum number of modifications in window for all reads

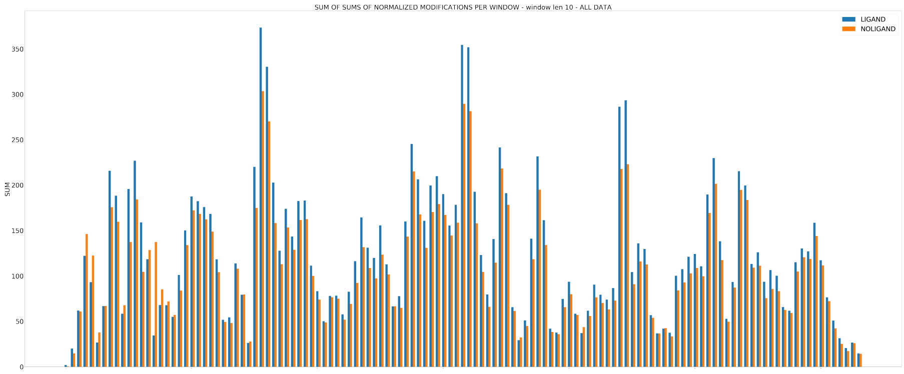


Figure 4.12: The number of reads with 3 or more modifications in particular window

5. Methods

5.1. K-means

K-means algorithm finds groups of data samples with the same variance. This algorithm requires a predefined number of groups - K stands for the number of clusters. K-means is a partition grouping since it results in solid groups - every example belongs only to one group. The main idea of the algorithm is that each of the K groups has its mean value representing the centroid and each sample belongs to the group whose centroid is closest in Euclidean distance. Adding an example to a specific group can cause the centroid to shift and the centroid needs to be recalculated. Therefore, this algorithm is iterative in nature (Šnajder).

In this thesis, the implementation of the python sklearn library for K-means was used (Pedregosa et al. ((2011))). In this implementation, data is shuffled and then K points are randomly used as init centroids. After that, in every iteration, every sample is allocated to the group with the nearest centroid. Then, centroids are recalculated in order to minimize the inertia (a condition in equation 5.1). In equation 5.1 x_i represents data sample and μ_j represents centroid (Pedregosa et al. ((2011))).

$$\sum_{i=0}^{n} \min_{\mu_j \in C}(||x_i - \mu_j||^2) \tag{5.1}$$

The algorithm is executed iteratively until the stop condition is achieved - no centroid change has occurred or the maximum number of iterations has been achieved.

The main disadvantage of the K-means algorithm is that it is necessary to know in advance the number of expected groups. In many cases it is not possible to know this, so various methods are used to determine the expected number of groups. In this case, to detect the expected number of clusters the *Elbow method* was used.

The *Elbow method* consists of several steps. In the first, the result of the K-means algorithm is performed for several different k. For each of k then the total inertia is calculated. The calculated inertia is plotted on a graph. On Figure 5.1 is the example of a plotted graph. The expected number of groups is determined by the apparent number in which the elbow is recognizable. In this case, it is 2.

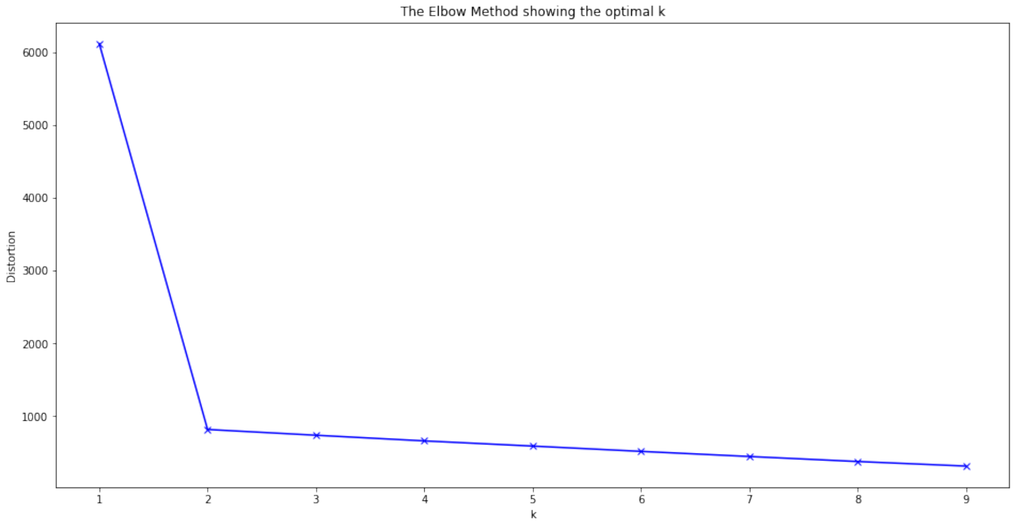


Figure 5.1: Example of Elbow method results

The entire implementation of the K-means approach was previously achieved by Ivona Matrinović as part of her Bachelor thesis. This implementation was additionally improved and then used as a comparative result with the results of other implemented methods (Martinović ((2020))).

5.2. Non-negative Matrix Factorization

Non-negative matrix factorization or NMF is an unsupervised algorithm that projects data into lower dimensional spaces.

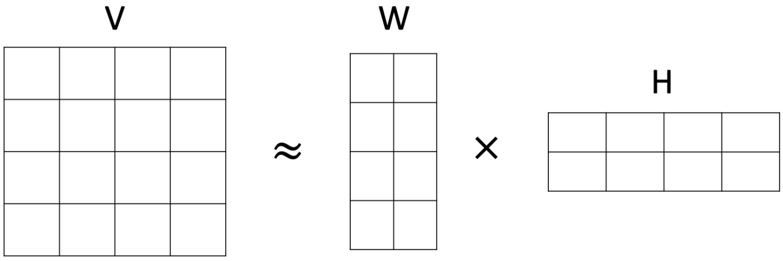


Figure 5.2: Non-negative matrix factorization

In this method matrix, V is factorized into two matrices W and H with the property that all three matrices have no negative elements (Figure 5.2). Non-negativity enables easier inspection of matrices. NMF is an NP-hard problem and it is approximated numerically.

In this thesis, NMF is used for its clustering property (Ding et al. ((2005))). It clusters the columns of V. To get W and H the following steps are performed:

1. Matrix V represents similarity matrix which has n rows and n columns, and n represents the number of reads. The similarity between reads is calculated using Jaccard index and Edlib (Šošić and Šikić ((2017)))

2. Matrices W and H are randomly initialized. The number of columns of the matrix W or the number of rows in matrix H is the same as the expected number of clusters.

3. The values in matrices W and H are iteratively updated using the following equations:

$$\boldsymbol{H}^{n+1}_{[i,j]} \leftarrow \boldsymbol{H}^{n}_{[i,j]} \frac{((\boldsymbol{W}^{n})^{T}\boldsymbol{V})_{[i,j]}}{((\boldsymbol{W^{n}})^{T}\boldsymbol{W}^{n}\boldsymbol{H}^{n})_{[i,j]}} \tag{5.2}$$

$$\boldsymbol{W}^{n+1}_{[i,j]} \leftarrow \boldsymbol{W}^{n}_{[i,j]} \frac{(\boldsymbol{V}(\boldsymbol{H}^{n+1})^{T})_{[i,j]}}{(\boldsymbol{W}^{n}\boldsymbol{H}^{n+1}(\boldsymbol{H}^{n+1})^{T})_{[i,j]}} \tag{5.3}$$

4. Matrices W and H are updated until are stable.

NMF is performed using python sklearn libraries (Pedregosa et al. ((2011))). Once matrices W and H are obtained, the clustering part is performed. Matrix H gives the cluster membership values:

$$\text{if } \boldsymbol{H}_{kj} > \boldsymbol{H}_{ij} \forall i \neq k \tag{5.4}$$

This equation 5.4 suggest that input data v_j belongs to k^{th} cluster. In this way, hard clustering is performed (Figure 5.3).

This can make sense if the maximum value of one column is significantly higher than the other membership values in the column. This was often not the case, so several iterations of NMF are performed. In every iteration, NMF with reads that are not clustered in the previous iteration is performed.

Iterations are performed until there is at least one read which is clustered in the previous iteration (Figure 5.4). After the algorithm stops, hard clustering is performed on the remaining reads. It is determined if the maximum value of one column significantly higher than others approximately by observing values obtained in the matrix

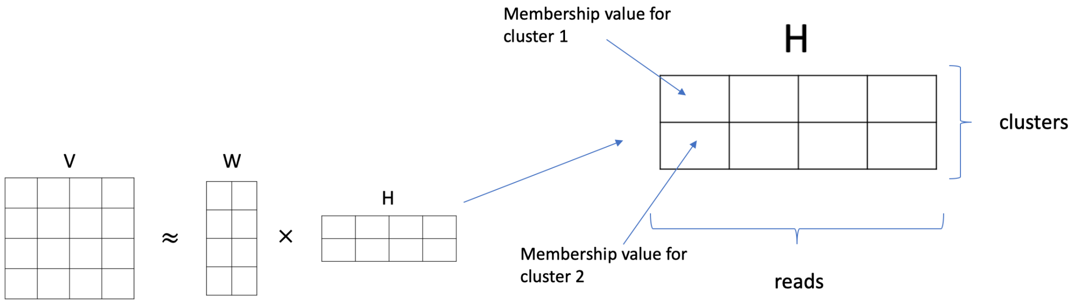

Membership value for cluster 1
H
clusters
V
W
H
≈
×
Membership value for cluster 2
reads

Figure 5.3: Steps for clustering after NMF

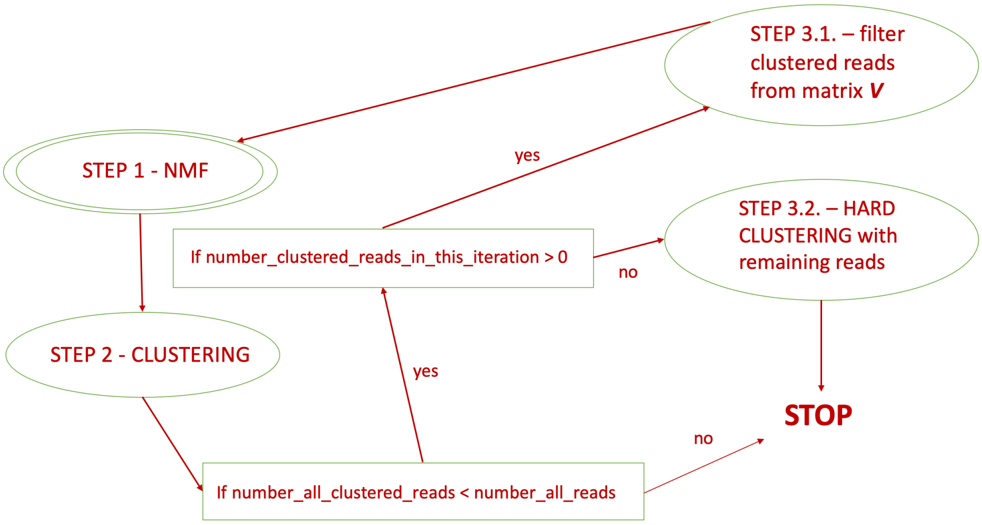

STEP 3.1. – filter clustered reads from matrix *V*
STEP 1 - NMF
yes
STEP 3.2. – HARD CLUSTERING with remaining reads
If number_clustered_reads_in_this_iteration > 0
no
STEP 2 - CLUSTERING
yes
STOP
no
If number_all_clustered_reads < number_all_reads

Figure 5.4: Non-negative matrix factorization in iterations

H. Perhaps it is not the best approach, but it leaves room for improvement of the algorithm. The thing that also can be improved and differently handle is the hard clustering of remaining reads.

5.3. Kalman Filter

In order to reduce noise (sequencing errors) from data *Kalman filter* is used. Kalman filter is an algorithm that provides estimates of some unknown variables given the measurements observed over time and which contain statistical noise and other inaccuracies. The positive side of the Kalman filter is that it has a simple form and requires small computational power.

Kalman filter is the algorithm that iteratively calculates several measurements: Kalman gain (KG), current estimate (EST_t) and error in the estimate (E_{EST}) according to every data input. To calculate KG, E_{EST} and error in data (E_{MEA}) are needed (equation 5.5). If the error in the estimate is smaller than the error in the data, a bigger accent is put on the estimate, otherwise more importance is put on the data. When KG is close to the 1 it can be said that data is accurate and estimations are unstable, and when KG is close to 0 estimations are stable and data is inaccurate (van Biezen ((2015))).

$$KG = \frac{E_{EST}}{E_{EST} + E_{MEA}}, 0 <= KG <= 1 \tag{5.5}$$

EST_t is recalculated as it is shown in equation 5.6. EST_{t-1} is previous estimate and EST_t is new recalculated estimate. MEA represents input data point. The recalculation of the current estimate depends on how much we can trust the estimate and the data.

$$EST_t = EST_{t-1} + KG[MEA - EST_{t-1}] \tag{5.6}$$

In the first iteration, we use the original estimate. The positive side of the Kalman filter is that the original estimate does not matter - for any random value Kalman filter will very quickly zero in on the true value. To calculate the new error in the estimate, the EST_t and the KG are used (equation 5.7).

$$E_{ES_t} = [1 - KG](E_{EST_{t-1}}) \tag{5.7}$$

The data used in this work is not time-related so some preprocessing methods are used (Figure 5.5). Inputs to the Kalman filter were chunks of reads. First, every read was represented as a bit vector. While constructing bit vectors, for every read, every 20 positions as window are observed. If the window contains at least one modification the label in the bit vector is 1, otherwise is 0. After that, chunks are constructed. Every chunk contained 100 reads representations. The input is then created by summing up the read representations per position.

The E_{MEA} used in this case was calculated by making permutations of all reads and separating the permutated reads in chunks of size 100. After that, sum of the reads per position was calculated and additional vectors are obtained. Then the mean value for every vector is calculated. The difference between the minimum and the maximum mean value is represented by E_{MEA}.

For the initial MEA randomly initialized vector of the same length as every other input vector is used. Every position of that vector was a random number between 0 and 100. Initial E_{EST} was set to the random number between 10 and 20.

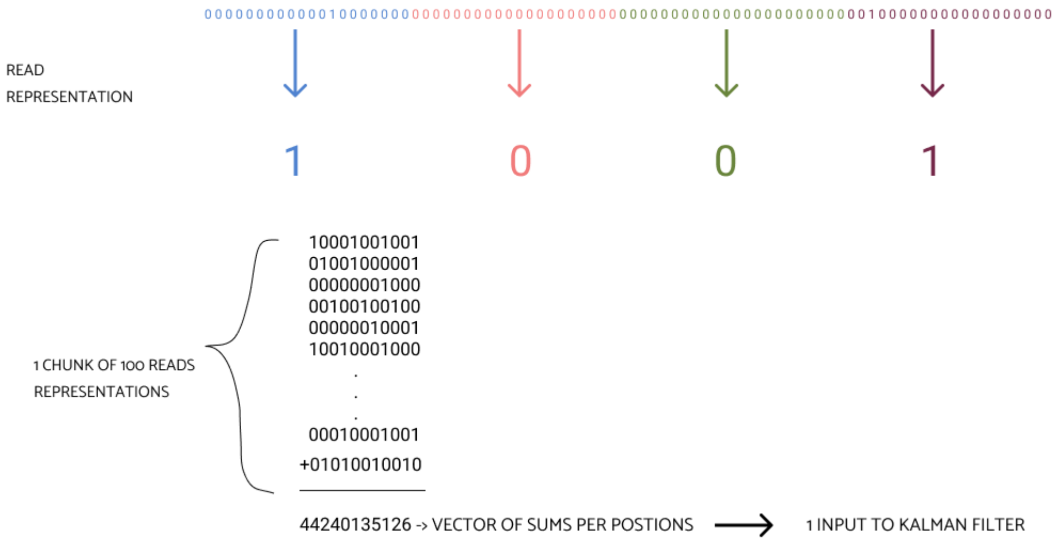


Figure 5.5: Example of construction method for constructing input data for Kalman filter

5.4. Graph Spectral Clustering

The main idea of *Graph Spectral Clustering* is to cluster the spectrum of unorganized data into multiple groups according to data uniqueness. It uses the connectivity approach to clustering, wherein communities of data points - nodes that are connected or very close to each other - are identified in the graph. Those nodes are then mapped to a low-dimensional space. Graph Spectral Clustering uses information from eigenvalues (which represent spectrum) of i.e. Laplacian Matrix derived from the graph or the dataset (Von Luxburg ((2007))).

In order to implement the Graph Spectral Clustering method, the significant positions obtained in the section 2.2.3 (Morandi et al. ((2021))) were used. Each read was represented with 10 obtained significant positions. If any of the reads did not have a modification on at least one of those 10 positions, it is not considered.

The adjacency matrix is calculated using equations from DRACO paper where additional weighting parameter for base coverage is introduced:

$$\mathbf{w} = \{w_i | w_i = \frac{coverage_i}{max(\mathbf{coverage})} \; \forall \; i \; index(\mathbf{bases})\} \tag{5.8}$$

$$\mathbf{A}_{ij} = \frac{\mathbf{Data}[i,:] \cdot \mathbf{Data}[j,:]}{\sqrt{w_i \cdot w_j}} \tag{5.9}$$

After the adjacency matrix is obtained, normalized Laplacian of the matrix is calculated and then the eigenvectors and their associated eigenvalues are obtained. The normalized Laplacian of the matrix was obtained using the following equations:

$$\boldsymbol{D} = diag(\boldsymbol{A}) \tag{5.10}$$

- where $\boldsymbol{A}$ represents adjacency matrix

$$\boldsymbol{L_{sym}} = \boldsymbol{D}^{-\frac{1}{2}}(\boldsymbol{D} - \boldsymbol{A})\boldsymbol{D}^{-\frac{1}{2}} \tag{5.11}$$

$$\boldsymbol{L_{sym}}\boldsymbol{V} = \boldsymbol{\lambda}\boldsymbol{L_{sym}} \tag{5.12}$$

In equation 5.12 $\boldsymbol{\lambda}$ represents eigenvector. In the case of the graph in which all nodes are not connected to each other the number of clusters is equal to the multiplicity of the second eigenvalue. This is not the case with data in this work so the number of clusters is not possible to read from eigenvalues. According to DRACO paper, it is possible to define vector of *eigengaps* ($\boldsymbol{\lambda}^{gap}$). *Eigengap* represents difference between two adjacency eigenvalues (equation 5.13).

$$\boldsymbol{\lambda}^{gap} = \{\lambda_i - \lambda_{i-1} \forall \lambda_i \in \lambda_2, \lambda_3, ..., \lambda_n\} \tag{5.13}$$

In the DRACO paper (Morandi et al. ((2021))) they used this vector to estimate the number of clusters by comparing the eigengapvector obtained from the real data and the eigengapvector obtained from the null model. The whole procedure of obtaining the number of clusters and the clustering itself presented in the DRACO paper is quite complicated and is omitted in the context of work in this thesis.

Further procedures were performed with the assumption that the expected number of clusters for TPP data is 2. Once the adjacency matrix was obtained and its corresponding Laplacian was computed, clustering was performed using python sklearn libraries (Pedregosa et al. ((2011))). Clustering is performed using K-means on eigenvectors whose belonging eigenvalues are nonzero.

For additional analysis, eigenvectors were plotted and eigengaps were calculated. The assumption is that the expected number of clusters is the number of eigenvalues before the largest eigengap. This analysis further verified the existence of at least 2 expected clusters.

6. Results and discussion

Measurements used for results (Šnajder):

1. Accuracy:

$$Acc = \frac{TP + TN}{TP + TN + FP + FN} \tag{6.1}$$

   where TP is the number of true positives, TN is the number of true negatives, FP is the number of false positives and FN of false negatives.

2. F1 score:

$$F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} = \frac{TP}{TP + \frac{1}{2}(FP + FN)} \tag{6.2}$$

   where $precision$ is defined as the number of true positives over the number of true positives plus the number of false positives and $recall$ is defined as the number of true positives over the number of true positives plus the number of false negatives.

3. Rand index:

$$R = \frac{a + b}{\binom{n}{2}} \tag{6.3}$$

   where a stands for number of equally marked pairs in the same groups (true positives), and b stands for number of different pairs marked in different groups (true negatives). In the denominator is the total number of pairs.

6.1. K-means and Non-negative Matrix Factorization clustering

Ordinary K-means (section 5.1) and NMF (section 5.2) were used for clustering *Tetrahymena* data (section 3.1) and also *TPP riboswitch* data (section 3.2). The approximate number of clusters is not known for Tetrahymena data. The expected number of clusters was obtained using the Elbow method. K-means clustering results are presented using the PCA method. Accordingly, it was not even known which read belonged to which cluster. To be able to carry out the evaluation, the results obtained for these two methods were compared. The methods are the first tested on artificially created data (section 3.3).

6.1.1. Results for simulated data

Simulated data for testing these two methods consisted of 3 clusters and each cluster was one of the types shown in the section 3.3. In the Figure 6.1 results of Elbow method for simulated data are presented.

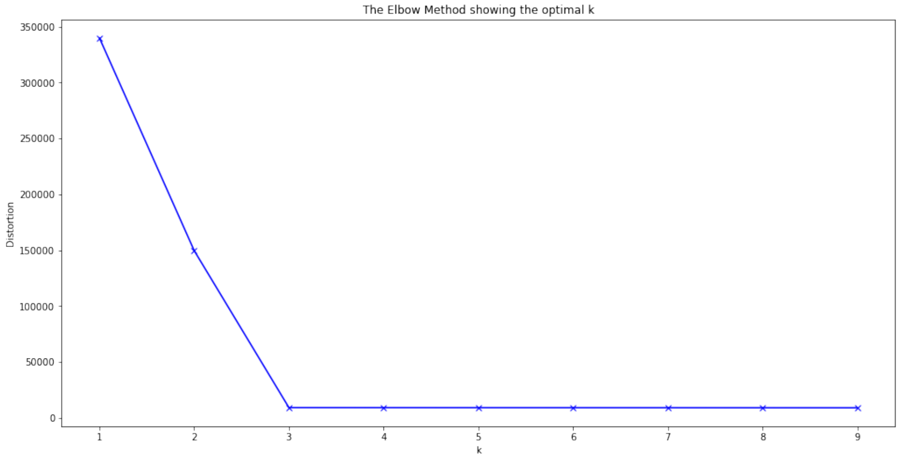


Figure 6.1: Elbow method for the simulated data for testing K-means and NMF

Noticeable elbow is at number 3 which suggests that the expected number of clusters for this data is 3.

In Figure 6.2 K-means clustering is visualized using PCA.

The Table 6.1 shows the obtained values for the measure of Rand index.

Rand index	
K-means	NMF
1.0	0.9519

Table 6.1: Rand indexes for simulated data

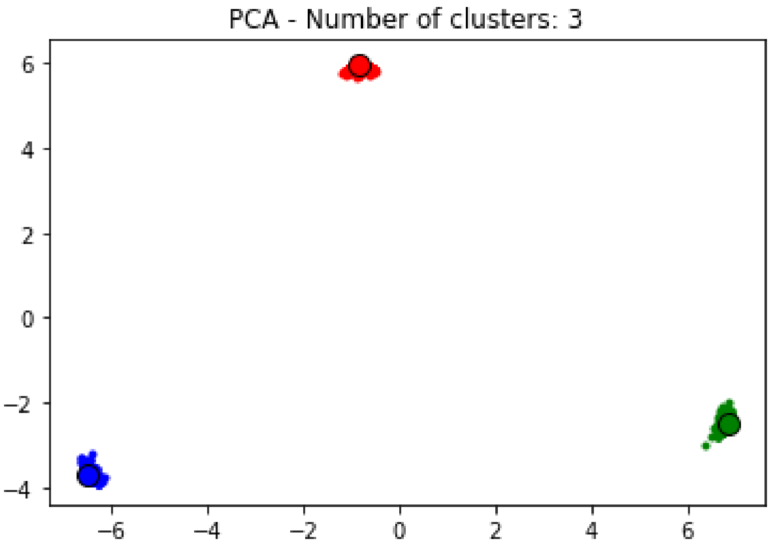


Figure 6.2: K-means clustering on simulated data

6.1.2. Results for Tetrahymena data

For Tetrahymena data Elbow method is performed and results are plotted on Figure 6.3.

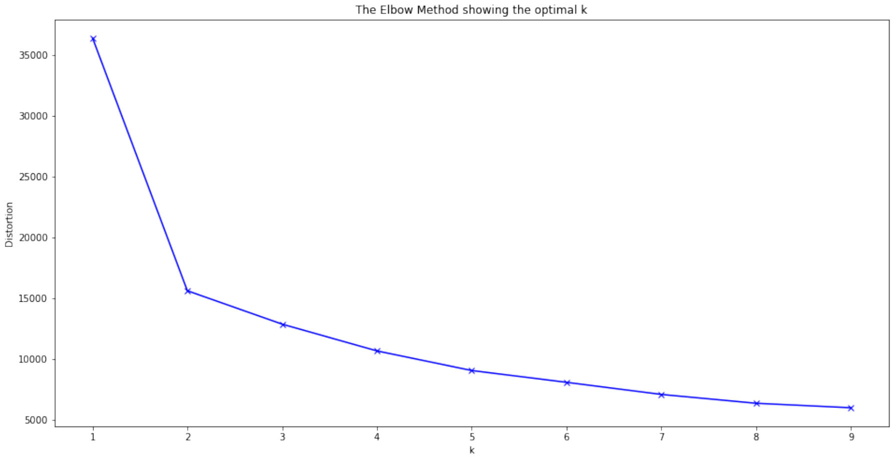


Figure 6.3: Elbow method for the Tetrahymena data

Although the largest elbow can be seen in number 2 due to previous analyzes carried out by Ivona Martinović, it is decided to assume that the data contains 3 clusters.

Visualized K-means clustering is shown in Figure 6.4.

In order to compare the results obtained by the K-means and the NMF method, the Rand index is calculated between the results obtained by the K-means method and the results obtained by the NMF method. In this case, Rand index was 0.7401.

In order to better analyze and compare the results, Venn's diagrams (Figure 6.5) show overlapping clusters obtained by these two methods.

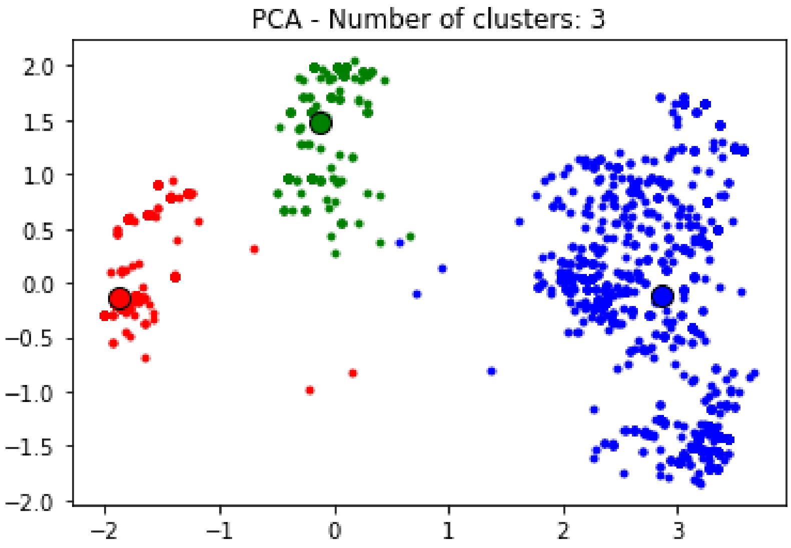


Figure 6.4: K-means clustering on Tetrahymena data

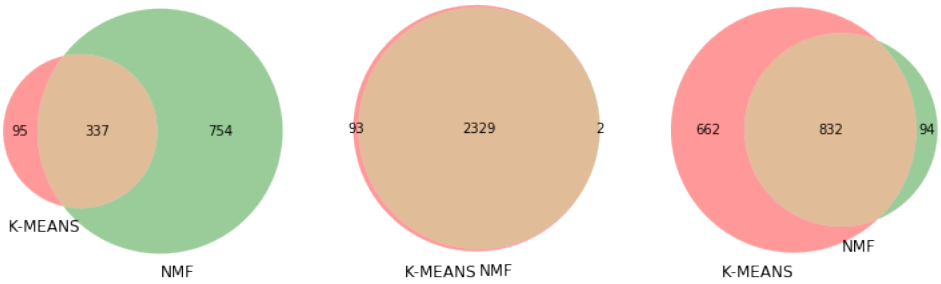


Figure 6.5: Venn's diagrams for 3 clusters obtained using K-means and NMF for Tetrahymena data

6.1.3. Results for TPP riboswitch data

Although TPP data is going with the assumption that the expected cluster number is 2 and that one of them includes ligand reads, and in other nonligand reads, the performed procedure was identical to the one that was performed on Tetrahymena data.

The results of the Elbow method for TPP data are shown in Figure 6.6.

Although the larger elbow can be observed in the area of numbers 5 and 7, the further analysis continued with the assumption that there are 2 clusters in the data. The results obtained with K-means for k=2 are shown in Figure 6.7.

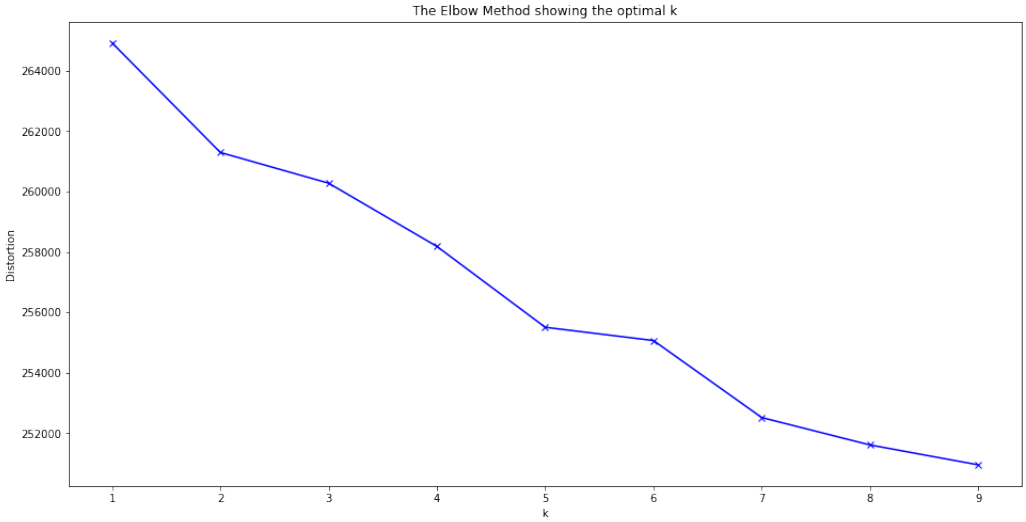


Figure 6.6: Elbow method for the TPP riboswitch data

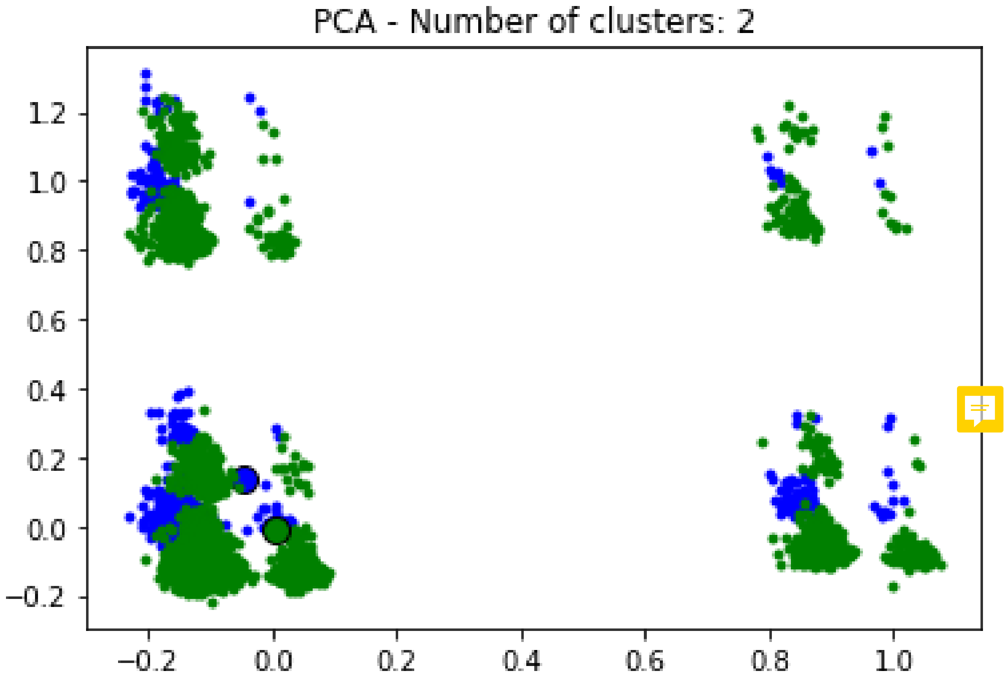


Figure 6.7: K-means clustering on TPP riboswitch data

The Table 6.2 shows the obtained values for the measure of Rand index for TPP riboswitch data.

Rand index	
K-means	NMF
0.5005	**0.7192**

Table 6.2: Rand indexes for TPP riboswitch data

Overlapping clusters obtained by the K-means method and the NMF method are shown by Venn's diagrams on Figure 6.8. The Rand index obtained by comparing the results obtained by the K-means method and the NMF method was 0.5323.

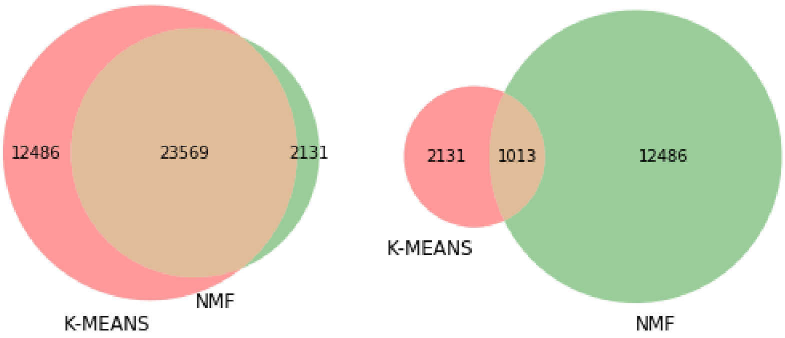


Figure 6.8: Venn's diagrams for 2 clusters obtained using K-means and NMF for TPP riboswitch data

6.1.4. Results for MPRS21 riboswitch data

The same procedure as with TPP data is tested also with MPRS_21 data.

The results of Elbow method for TPP data are shown in Figure 6.9.

The elbow is noticeable at positions 4, 6 and 8, but due to the assumption for riboswitch data, clustering was performed with 2 clusters.

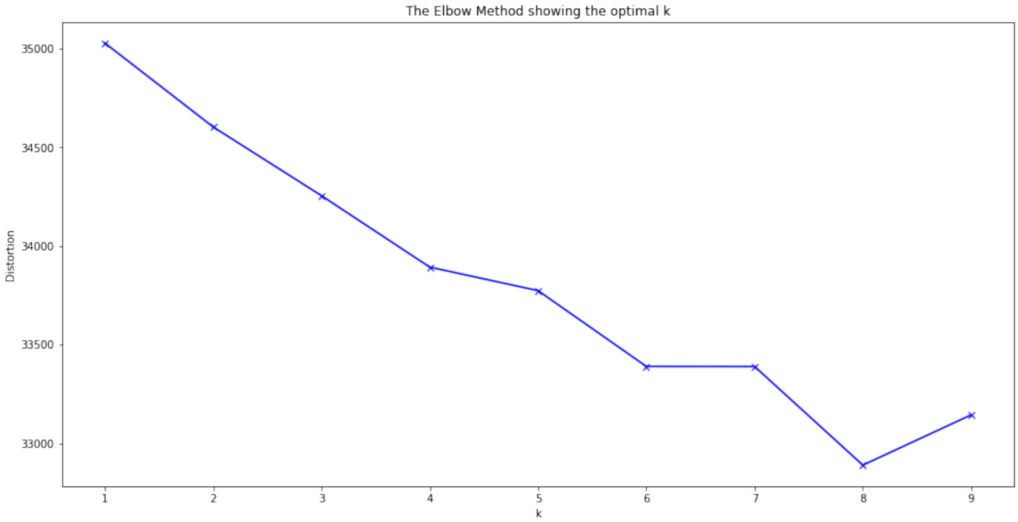


Figure 6.9: Elbow method for the MPRS_21 riboswitch data

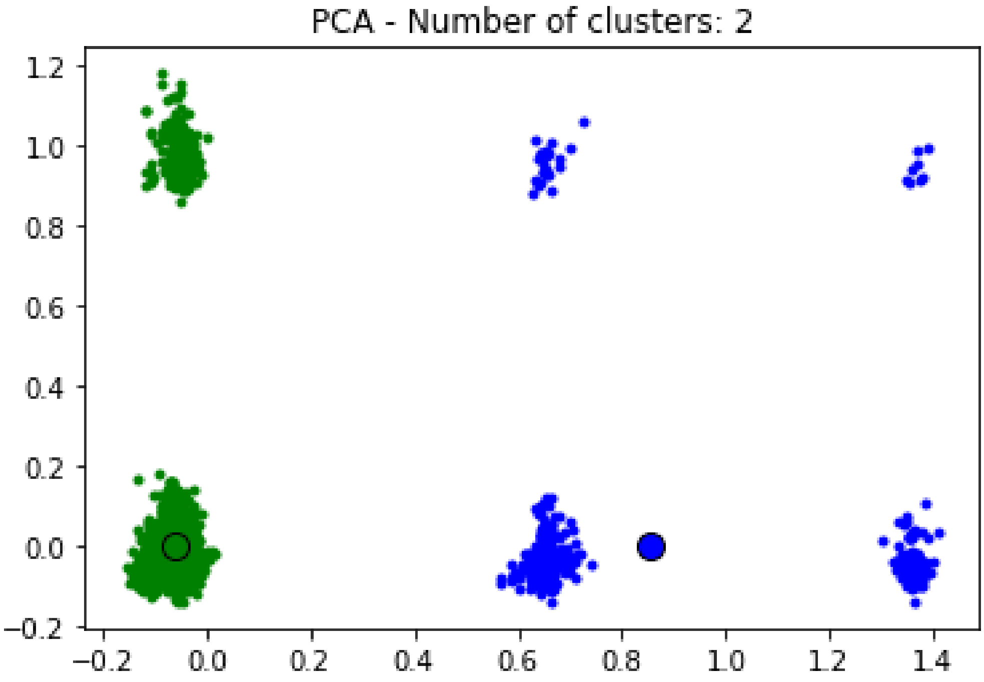


Figure 6.10: K-means clustering on MPRS_21 riboswitch data

The Table 6.3 shows the obtained values for the measure of Rand index for TPP riboswitch data.

Rand index	
K-means	NMF
0.5454	**0.8102**

Table 6.3: Rand indexes for TPP riboswitch data

Overlapping clusters obtained by the K-means method and the NMF method are shown by Venn's diagrams on Figure 6.11. The Rand index obtained by comparing the results obtained by the K-means method and the NMF method was 0.5067.

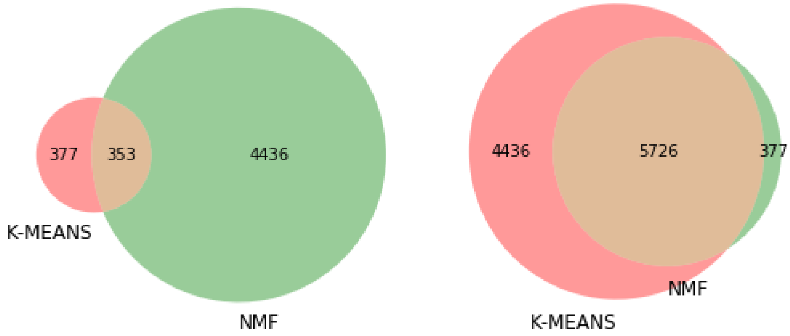


Figure 6.11: Venn's diagrams for 2 clusters obtained using K-means and NMF for MPRS_21 riboswitch data

6.1.5. Discussion

In the case of simulated data by both methods, quite high results were achieved. Data from simulated sets are easily separable. This test shows that on a simple data without a lot of noise these methods work. According to the obtained results, K-means is a bit more successful, but both results are satisfactory.

In the case of Tetrahymena data, the biggest problem is that there is no assumption about the number of clusters or how the operation of these two methods on these data could be verified. In her work, Ivona Martinović, using different methods, came to the conclusion that 3 clusters are expected. Such an assumption was used in this work.

Venn's diagrams (Figure 6.5) show that there is no significant match between the results of the two methods. From this, it can be concluded that data clustering is different for these two methods. As we are not able to validate the methods over this data, we cannot even know which method to trust more and which to consider more successful.

In the case of riboswitch data, the assumption that there are two structures, i.e. two clusters, was used. If the graphs obtained by Elbow methods (Figures 6.7 and 6.10) are looked at it is still not 2 clusters. But if the results in the tables 6.2 and 6.3 are looked at it can be seen that the results for the NMF method are significantly better. The Elbow method used K-means to determine the expected number of clusters, which is probably why the result of clustering for K-means for 2 clusters is much lower, while NMF manages to successfully perform clustering for 2 clusters.

Although the results for clustering using the NMF method are significantly and enviably higher than the others, it is not entirely accurate to say that this is a successful method. Namely, when performing iterations in which one factorization is performed each time, it is assumed that the first row of the matrix H is always cluster 1, and the second row is always cluster 2. Such an assumption is not correct because the centroids of such clusters are deformed. It is assumed that there is a way in which this method could be modified so that the centroids of the cluster still have a spherical shape. In the context of this work, this has not been done and it has not been explained why this method gives good results without it.

6.2. Kalman Filter

Kalman Filter is used in order to reduce noise in *TPP riboswitch* data. Three different datasets are used. One were reads modified with ligad, one without ligand and the third was the set of connected ligand and non ligand reads.

6.2.1. Results

In Table 6.4 the outputs from Kalman filter are shown. Construction of inputs in Kalman filter are explained in Section 5.3.

Table 6.4: Kalman filtering - initial values and results

Initial value	Result
10	0.1001
12	3.5642
8	6.8096
13	5.0636
8	11.6298
13	11.8077
10	9.4393
5	6.8829
10	5.3408
10	5.9359
11	12.7406
13	12.6774
13	8.7665
7	5.0928
8	6.2929
10	15.6743
11	22.6296
8	10.4172
15	11.3746
9	14.5442
8	7.9481
8	7.8744
10	5.7619
5	10.2363
8	9.3358
7	11.0714
9	5.8175
8	12.393
9	16.2656
8	16.4245
13	15.429
11	14.1602
5	25.4211
9	10.1102
15	11.0548
9	15.6403
11	2.8509
6	10.8884
4	11.3318
8	3.6274
10	7.0433
11	4.5781
13	7.7919
7	6.6345
11	20.7732
11	8.8335
5	10.8751
12	4.0543
12	4.0617
8	8.8441
13	9.6924
4	14.9539
9	10.2213
9	7.9093
16	14.4633
11	9.5799
12	8.0328
11	5.7045
13	8.2812
15	8.7909
13	8.1776
11	3.5099
9	1.6169
7	1.1818

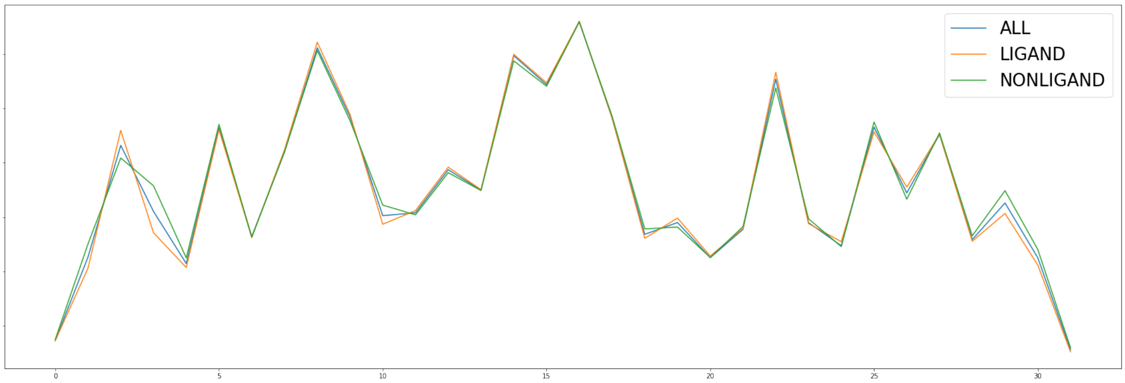


Figure 6.12: Output signals from Kalman filter for TPP data

6.2.2. Discussion

It was expected that Kalman filter will give output signal that was cleared of noise. Although the results in Table 6.4 suggest that the change of the signal really exist, on the Figure 6.12 can be seen that output signals for three different datasets (data with ligand, data without ligand and connected data) are the same. This fact shows that the Kalman filter has indeed made some changes, but it may be too simple method to reduce such noise that is present in this data. The assumption is that the noise in this data does not have any distribution.

6.3. Detection Of Significant Positions

The detection of significant positions is performed for four different simulated datasets (presented in Table 6.5) and for three different *riboswitch* datasets (TPP, MPRS21 and GUA). The method for obtaining significant positions is presented in Section 4.2.

	Number of modifs per read	Modifs C1	Modifs C2
Dataset 1	10 - 80	beginning and end	every 20 bases
Dataset 2	10 - 80	beginning and end	middle
Dataset 3	10 - 40	beginning and end	every 20 bases
Dataset 4	10 - 40	beginning and end	middle

Table 6.5: Types of simulated datasets for detecting significant positions on reads

6.3.1. Results for simulated datasets

Obtained positions:

1. Dataset 1 - Figure 6.13

   Positions: [4, 9, 30, 52, 54, 57, 60, 62, 63, 64]

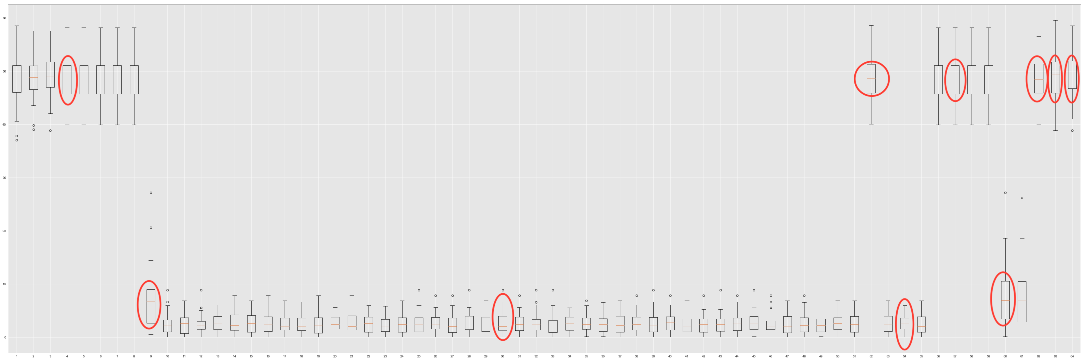


Figure 6.13: Simulated dataset 1 - significant positions

2. Dataset 2 - Figure 6.14

   Positions: [2, 4, 15, 28, 36, 42, 51, 60, 62, 64]

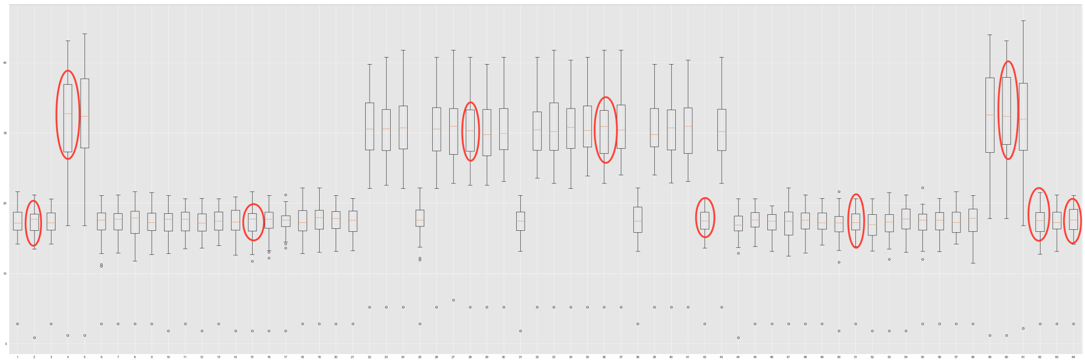


Figure 6.14: Simulated dataset 2 - significant positions

3. Dataset 3 - Figure 6.15

   Positions: [1, 3, 4, 20, 41, 53, 60, 61, 62, 64]

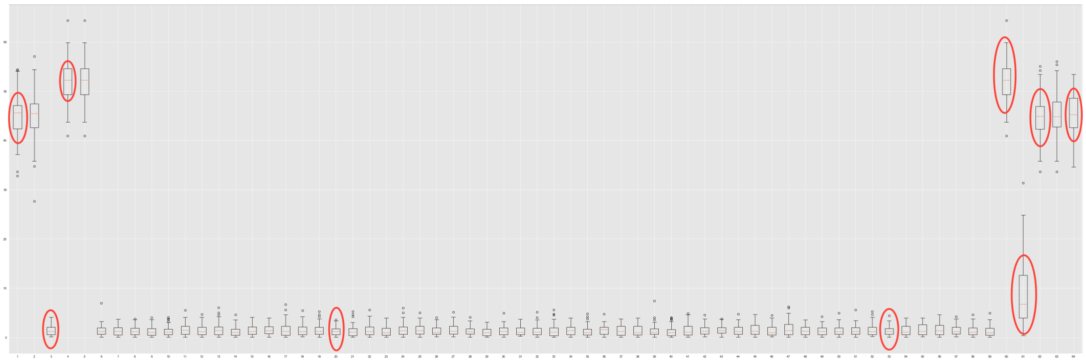


Figure 6.15: Simulated dataset 3 - significant positions

4. Dataset 4 - Figure 6.16

   Positions: [11, 23, 28, 32, 34, 36, 47, 60, 62, 64]

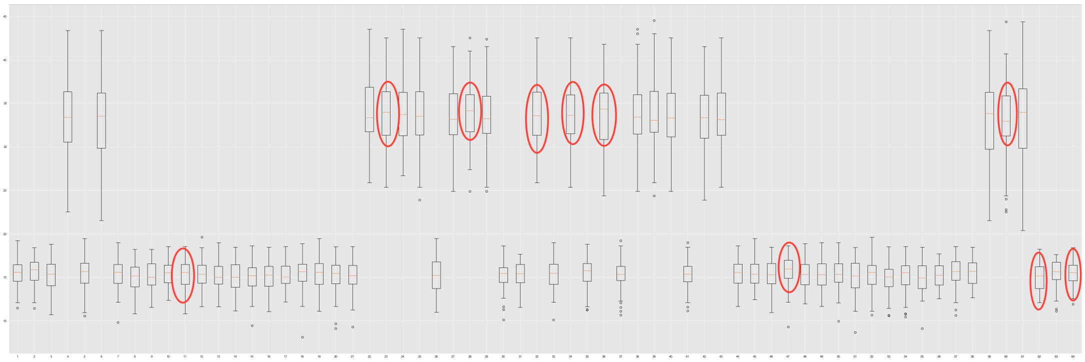


Figure 6.16: Simulated dataset 4 - significant positions

6.3.2. Results for riboswitch datasets

Obtained positions:

1. TPP data - Figure 6.17

   Positions: [8, 17, 23, 30, 33, 35, 41, 45, 54, 64]

2. MPRS_21 data - Figure 6.18

   Positions: [1, 7, 14, 17, 19, 23, 30, 34, 49, 63]

3. GUA data - Figure 6.19

   Positions: [2, 5, 9, 14, 29, 44, 50, 56, 60, 65]

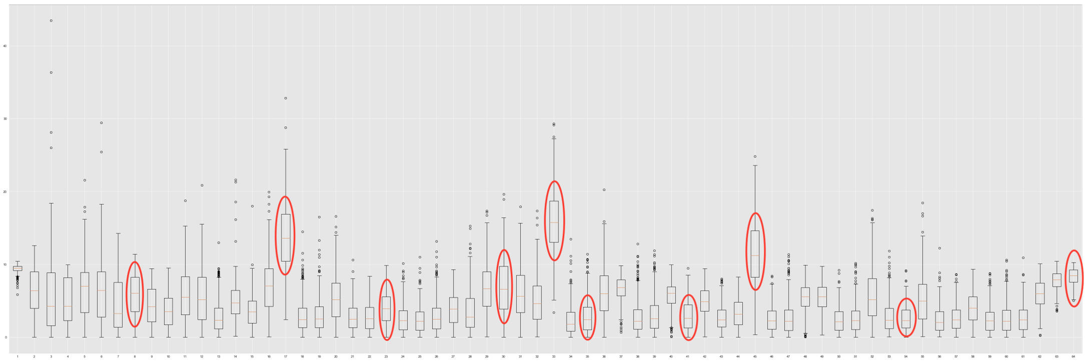


Figure 6.17: TPP dataset - significant positions

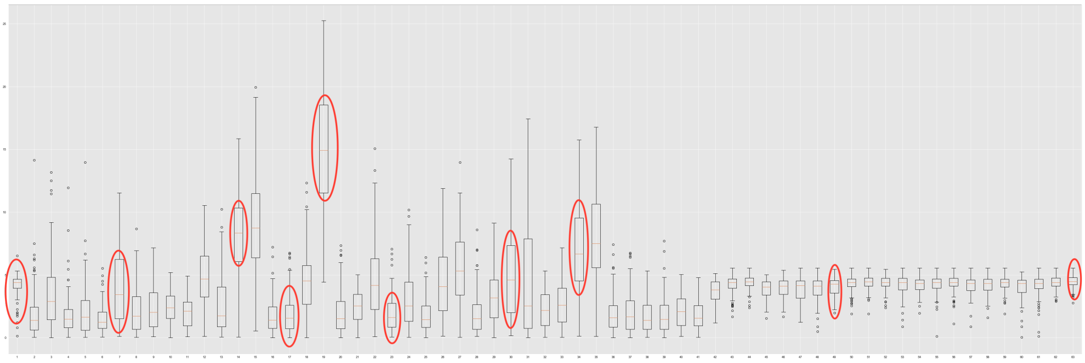


Figure 6.18: MPRS_21 dataset - significant positions

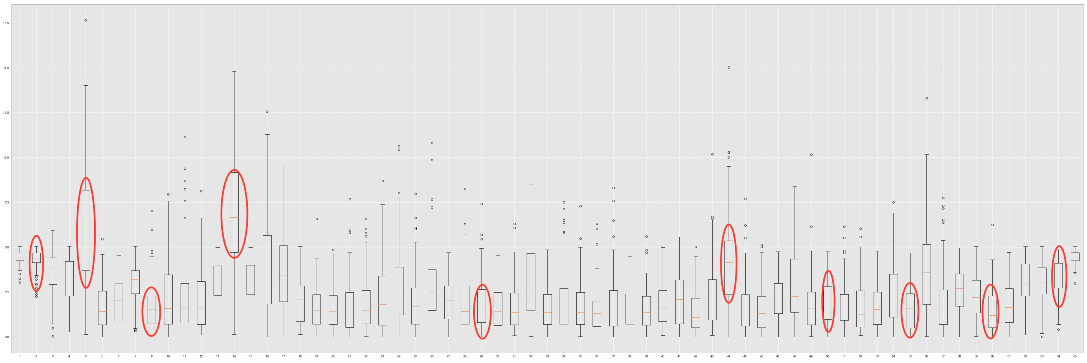


Figure 6.19: GUA dataset - significant positions

6.3.3. Discussion

The main conclusion after obtaining significant positions for simulated and real data is that data distribution between these two kinds of datasets is significantly different. Differences between positions mean values are higher for simulated data and groups of positions are easier to identify from their positions box plots.

There is still place for improvement for the method of obtaining significant positions. Taking only one value for groups that have many elements can cause the omission of a position that may be important for clustering. Improving this step is left for future work.

6.4. Graph Spectral Clustering

Graph Spectral Clustering was performed for data for which significant positions were obtained in the previous chapter. Each read of each of these datasets is represented only by these positions. Explanations for the simulated datasets can be found in the Table 6.5.

6.4.1. Results for simulated data

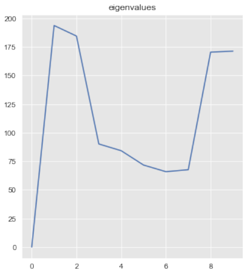


Figure 6.20: Eigenvalues for Dataset 1

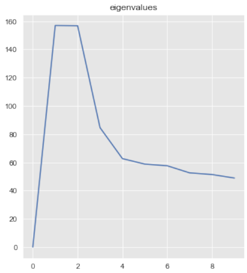


Figure 6.21: Eigenvalues for Dataset 2

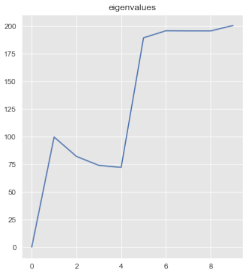


Figure 6.22: Eigenvalues for Dataset 3

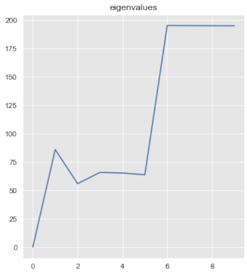


Figure 6.23: Eigenvalues for Dataset 4

	Accuracy	F1 score	Rand index
Dataset 1	0.96	0.97	0.93
Dataset 2	**0.99**	**0.99**	**0.99**
Dataset 3	0.96	0.96	0.93
Dataset 4	**0.99**	**0.99**	0.97

Table 6.6: Graph spectral clustering performance on simulated data

6.4.2. Results for riboswitch data

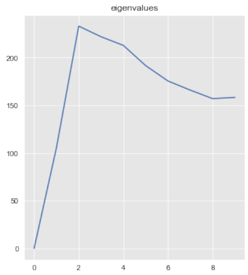


Figure 6.24: Eigenvalues for TPP

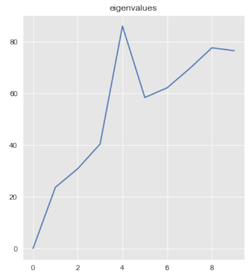


Figure 6.25: Eigenvalues for MPRS_21

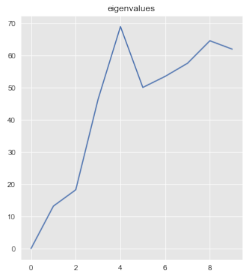


Figure 6.26: Eigenvalues for GUA

	Accuracy	F1 score	Rand index
TPP	0.54	**0.62**	0.5
MPRS_21	**0.57**	0.26	**0.51**
GUA	0.51	0.14	0.5

Table 6.7: Graph spectral clustering performance on simulated data

6.4.3. Discussion

In the graphs (Figures 6.20, 6.21, 6.22 and 6.23) showing the eigenvalues for the simulated data, it can be seen that for Dataset 1 and Dataset 2 the largest eigengaps are between the first and second eigenvalue. In the case of Dataset 3 and Dataset 4, the largest eigengaps are between the third and fourth eigenvalue. The assumption of the existence of 2 clusters is confirmed for Dataset 1 and Dataset 2, while for other sets it is not. Despite that, clustering with 2 clusters gave good results (Table 6.7).

If the graphs for riboswitch data are looked at (Figures 6.24, 6.25 and 6.26), in TPP data the largest eigengap is between the first and second eigenvalue which confirms the existence of 2 clusters. For MPRS_21 data, the largest eigengap is between the third and fourth, and for GUA between the second and third eigenvalue. This does not confirm the assumption of the existence of only 2 clusters. Also, if the results in Table 6.7 are looked at, it can be concluded that clustering with 2 clusters is not successful.

This implementation of this method does not give good results. There is a possibility that the method itself needs to be modified, but it is also assumed that a large amount of noise in riboswitch data plays a major role in the failure.

6.5. K-means With Significant Positions

The reads represented by significant positions were also clustered using K-means. The same datasets as in the previous section were used.

6.5.1. Results for simulated data

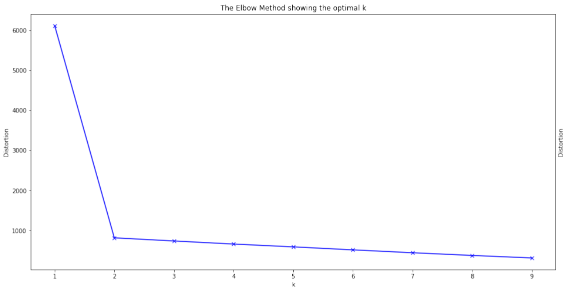


Figure 6.27: Elbow method for Dataset 1

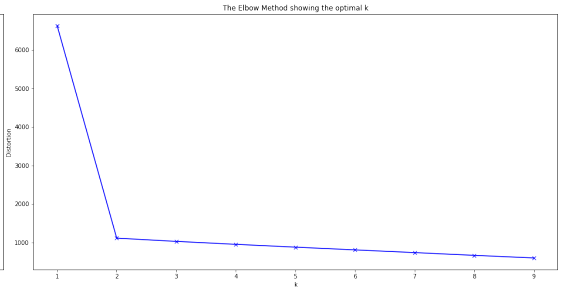


Figure 6.28: Elbow method for Dataset 2

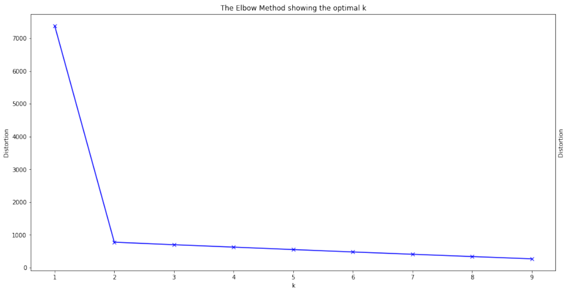


Figure 6.29: Elbow method for Dataset 3

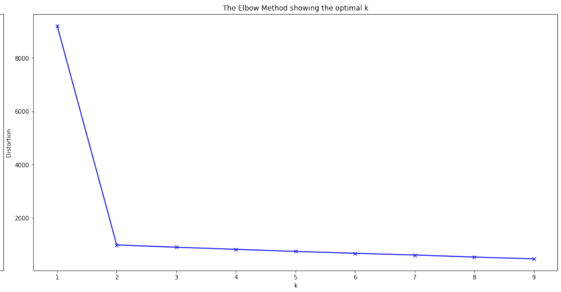


Figure 6.30: Elbow method for Dataset 4

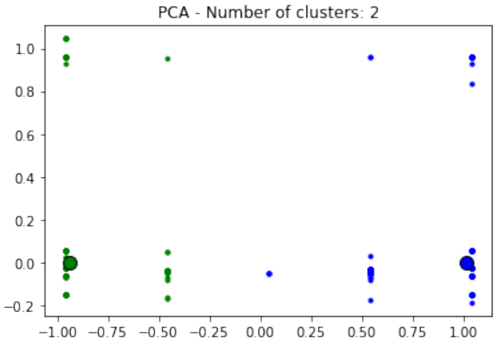


Figure 6.31: PCA for Dataset 1

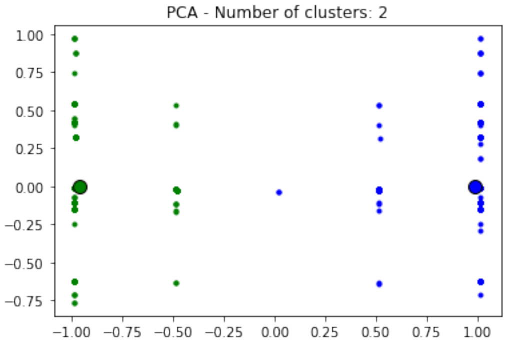


Figure 6.32: PCA for Dataset 2

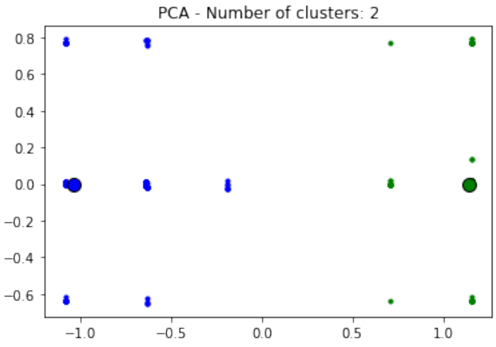


Figure 6.33: PCA for Dataset 3

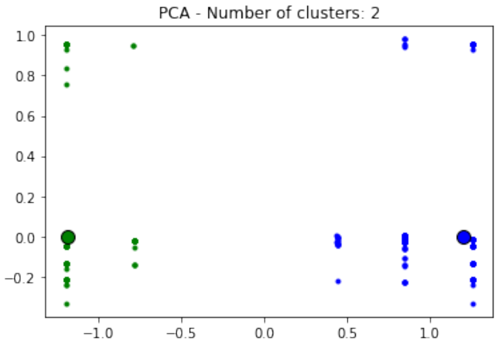


Figure 6.34: PCA for Dataset 4

	Accuracy	F1 score	Rand index
Dataset 1	0.99	0.99	0.99
Dataset 2	0.99	0.99	0.99
Dataset 3	**1.0**	**1.0**	**1.0**
Dataset 4	**1.0**	**1.0**	**1.0**

Table 6.8: K-means performance on simulated data represented with significant positions

6.5.2. Results for riboswitch data

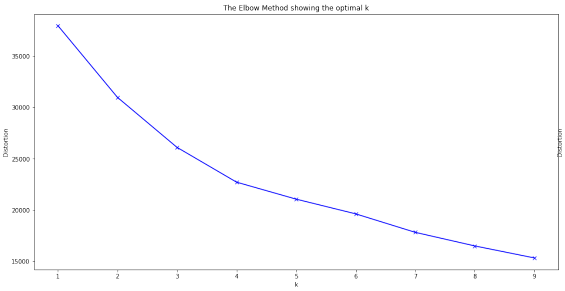


Figure 6.35: Elbow method for TPP

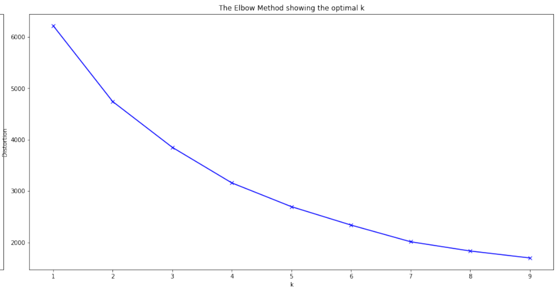


Figure 6.36: Elbow method for MPRS_21

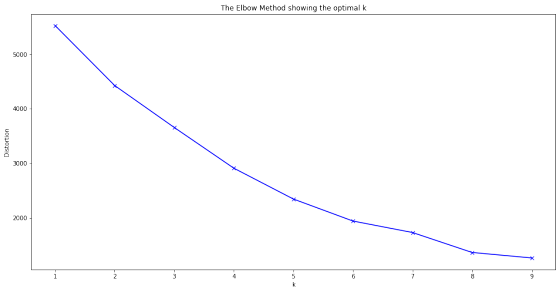


Figure 6.37: Elbow method for GUA

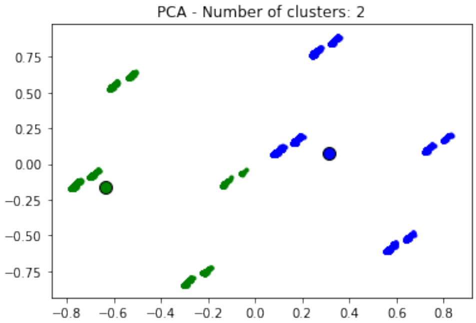


Figure 6.38: PCA for TPP

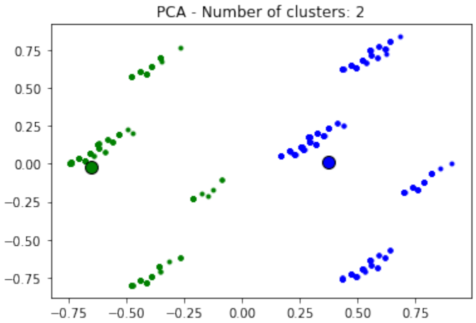


Figure 6.39: PCA for MPRS_21

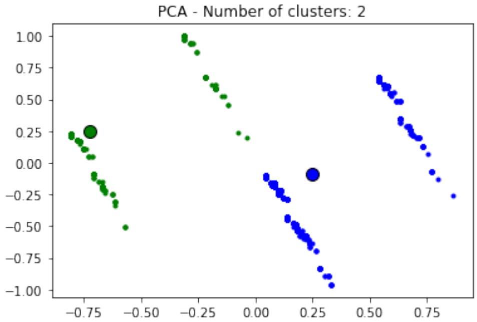


Figure 6.40: PCA for GUA

	Accuracy	F1 score	Rand index
TPP	0.5	0.57	0.5
MPRS_21	**0.56**	**0.67**	**0.51**
GUA	0.51	0.61	0.5

Table 6.9: K-means performance on riboswitch data represented with significant positions

6.5.3. Discussion

Representations of reads with the significant positions were supposed to remove noise from real data and thus improve clustering. As expected, this method gives very good results on simulated datasets (Table 6.9). On the other hand, if we compare Table 6.9 and Tables 6.2 and 6.3 it can be concluded that the success of clustering with K-means is not improved by introducing such representations.

The results of clustering in both cases are similar, which suggests that the approach with such representations can replace the basic one and thus speed up the clustering process, but it can also be concluded that this approach did not remove enough noise to make it easier for K-means to divide clusters.

7. Conclusion

The main goal of this thesis was to cluster modified nucleotides. Numerous methods have been tested and existing methods for solving this problem has been investigated.

In order to validate those methods, it is necessary to have good and realistic data. The main problem with this task is the data. Sequencing RNA structures with third-generation sequencers still generates a large amount of error that introduces noise into the data. This poses an additional challenge, and that is the distinction between the actual modification and the sequencing error. In this thesis, the focus was on clustering modifications, assuming that modifications were successfully detected and separated from errors. Additional analysis of these data showed that noise in the data was still present. This is why clustering methods should be robust enough so that noise cannot affect the final outcome or the data must be in some way further cleared of noise.

The methods implemented in this thesis have proven successful on simulated datasets. From this, it can be concluded that for easily separable data that do not contain a large amount of noise, such approaches work. On the other hand, the results for real datasets were not so good. The best result (rand index 0.72 and 0.81) was achieved by a method that uses non-negative matrix factorization in iterations, but the approach used by this method is not completely correct and requires additional improvement. Therefore, it cannot be said with certainty that these results are really good results. Other methods, even after attempts to remove noise in the data, gave poor results. Whether this is really bad data or the approaches are too simple or in some way wrong, it is difficult to say with certainty.

Improving this type of clustering requires additional data analysis and a full understanding of modification detection methods. The detection step can play a key role in the clustering itself, and such an approach has not been used in this work. Also, some kind of robust and complex method could further identify the noise in the data, which methods used in this work have failed.

In conclusion, one should be aware that each clustering method has its own peculiarities and it is difficult to achieve global clustering for all types of modifications.

Bibliography

M. D. A. Di Giorgio. Synthetic small-molecule rna ligands: futureprospects as therapeutic agents, 2019. URL `https://pubs.rsc.org/en/content/articlepdf/2019/md/c9md00195f`.

S. Clancy. Chemical structure of rna, 2008. URL `https://www.nature.com/scitable/topicpage/chemical-structure-of-rna-348/`.

C. Ding, X. He, and H. D. Simon. On the equivalence of nonnegative matrix factorization and spectral clustering. In *Proceedings of the 2005 SIAM international conference on data mining*, pages 606–610. SIAM, 2005.

A. L. Edwards and R. T. Batey. Riboswitches: a common rna regulatory element. *Nature Education*, 3(9):9, 2010.

Z. Feng, J. C. Clemente, B. Wong, and E. E. Schadt. Detecting and phasing minor single-nucleotide variants from long-read sequencing data. *Nature communications*, 12(1):1–13, 2021.

H. Lee, J. Gurtowski, S. Yoo, M. Nattestad, S. Marcus, S. Goodwin, W. R. McCombie, and M. C. Schatz. Third-generation sequencing and the future of genomics. *BioRxiv*, page 048603, 2016.

M. D.-L. M. Šikić. Bioinformatika, 2013. URL `https://www.fer.unizg.hr/_download/repository/bioinformatika_skripta_v1.2[1].pdf`.

I. Martinović. *Pipeline for Detection Clusters of Modified Nucleotides in Nanopore Sequenced RNA Reads*. PhD thesis, 2020.

E. Morandi, I. Manfredonia, L. M. Simon, F. Anselmi, M. J. van Hemert, S. Oliviero, and D. Incarnato. Genome-scale deconvolution of rna structure ensembles. *Nature Methods*, 18(3):249–252, 2021.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

O. N. Technologies. How does nanopore dna/rna sequencing work?, 2020. URL `https://nanoporetech.com/how-it-works`.

P. J. Tomezsko, V. D. Corbin, P. Gupta, H. Swaminathan, M. Glasgow, S. Persad, M. D. Edwards, L. Mcintosh, A. T. Papenfuss, A. Emery, et al. Determination of rna structural diversity and its role in hiv-1 rna splicing. *Nature*, 582(7812):438–442, 2020.

M. van Biezen. Special topics 1 - the kalman filter. `https://www.youtube.com/watch?v=CaCcOwJPytQ&list=PLX2gX-ftPVXU3oUFNATxGXY90AULiqnWT&index=1`, 2015.

U. Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17(4): 395–416, 2007.

J. Šnajder. Lectures notes in machine learning course.

M. Šošić and M. Šikić. Edlib: a C/C++ library for fast, exact sequence alignment using edit distance. *Bioinformatics*, 33(9):1394–1395, 01 2017. ISSN 1367-4803. doi: 10.1093/bioinformatics/btw753. URL `https://doi.org/10.1093/bioinformatics/btw753`.

Detection of Modified Nucleotide Clusters in Nanopore Sequenced RNA Reads

Abstract

The main goal of this thesis was to detect groups of modified nucleotides in RNA reads obtained by third generation sequencers. Reads obtained by third-generation sequencers often contain a large amount of noise. In this work, detailed analyzes of the data were performed in order to try to remove the noise and so that the data could be better understood. The related work achieved in the area of this problem is shown. Clustering was performed by various methods such as K-means, Graph Spectral Clustering and Non-negative matrix factorization. Clustering methods were tested on simulated datasets as well as on real datasets. The results are presented in tables and graphs. The source code of the implemented methods can be found at:

Keywords: RNA modifications, clustering methods, data analysis, nanopore reads

Detekcija grupa modificiranih nukleotida u očitanjima RNA dobivenih metodom nanopora

Sažetak

Glavni cilj ovog diplomskog rada bio je detektiranje grupa modificiranih nukleotida u RNA očitanjima dobivenim sekvencerima treće generacije. Očitanja dobivena sekvencerima treće generacije često sadržavaju veliku količinu šuma. U ovom radu provedene su detaljne analize podataka kako bi se šum pokušao ukloniti te kako bi se podatci mogli bolje razumjeti. Prikazan je trenutni postignuti rad u području ovog problema. Provedeno je grupiranje raznim metodama kao što su K-means, Graph Spectral Clustering te Non-negative matrix factorization. Metode grupiranja testirane su na simuliranim skupovima podataka kao i na stvarnim skupovima podataka. Rezultati su prikazani tablično i grafički. Izvorni kod implementiranih metoda nalazi se na:

Ključne riječi: RNA modifikacije, metode grupiranja, analiza podataka, nanopor očitanja