IDENTIFICATION OF M^1A MODIFICATIONS BY MACHINE LEARNING
03-713: Bioinformatics Data Integration Practicum

PC Aiyappa (aparvang), Bernie Zhao (yeiweizha), Gayatri Joshi (gayatrij)

Introduction

RNA modifications abound on non-coding RNA, such as rRNA, tRNA and snRNA, to such a great degree that pseudouridine, one such modification, has been termed the 'fifth nucleoside'. These subtle post-transcriptional alterations have several consequences including effects on the translational fidelity of rRNA[1], tRNA secondary structure stability, spliceosomal assembly and pre-mRNA splicing competency of snRNP[2]. Additionally, RNA modifications such as pseudouridine-55 and m^5C have been shown to act as indirect signals that bridge metabolic states with distinct translational outputs[3]. With these modifications, nature has sought to expand the topological potential of the four nucleotides and codify higher-level information beyond the nucleotide sequence. Thus far, more than 140 such modifications have been discovered and yet more are suspected to exist[4]. Until recently research has neglected the study of mRNA modifications with the understanding that such modifications were static and did not have a major influence on the structure and function of mRNA.

The NGS revolution, equipped with the power of deep sequencing, and the discovery that mRNA modifications are physiologically dynamic and associated with disease(obesity and diabetes) has led to a resurgence in 'epitransriptomics' research[5]. Recent literature has focused on alternate sequencing library preparation protocols for detection of individual mRNA modifications and the latest computational methods have aimed at unraveling the epitranscriptome landscape[6-7]. The nature of artifacts in sequencing data produced by various modifications have been recorded by such epitranscriptome landscape studies and this in turn has spurred the development of tools for prediction of sites of modifications[8]. We have implemented one such tool for the identification of m^1A sites in sequencing data.

N^1-methyladenosine (m^1A) is a post-transcriptional modification prevalent in tRNA and rRNA where it contributes to structural stability and is indispensable for proper biological function. For example in human 28S rRNA, m^1A is methylated at position 1,322 and is required for rRNA biogenesis[7]. The m^1A modification is present at the Watson–Crick interface and interferes with the function of polymerases in 2 ways: truncation of the replication process or read-through with characteristic base misincorporation[3]. These properties of m^1A have previously inspired methods for prediction of m^1A sites from sequencing data; however, these methods demand considerable sequencing depth.

Dominissini et al. developed a technique termed 'm^1A-seq' which combines traditional NGS with immunoprecipitation using antibodies specific to m^1A for enrichment of mRNA containing these modifications. This technique also included a customized library preparation procedure which prevents m^1A to m^6A conversion(Dimroth Rearrangement); which would otherwise have erased the reverse transcription arrest and misincorporation signature of m^1A (as m^6A doesn't manifest itself with such artifacts). The NGS data made available with this paper includes traditional RNA-seq and m^1A-seq experiments on the same biological samples. Thus, we have the traces left by m^1A in normal RNA-seq and also the positional labels from m^1A-seq which are ideal for training machine learning algorithms. Assuming adequate information content in the features, the trained ML model can then be used for predicting m^1A from conventional RNA-seq experiments. Such tools find motivation in the fact that m^1A is physiologically dynamic and is found to preferentially decorate alternate splicing sites; biological inferences can therefore be guided by predicting modified sites using regular RNA-seq, without the additional cost of modifications to existing technologies.

Problem Description

Given RNA-seq data, a reference genome and a list of user-specified regions our objective was to predict m^1A sites on the RNAseq data. To accommodate the form of the training data, the initial goal was modified to predict windows with a high probability of containing m^1A. This entailed extracting distinctive features of regions containing m^1A. In this regard, we were guided by the prior literature on m^1A modifications. Ultimately, the task was to train a machine learning model to learn the coverage artifacts, GC content, minimum free energy(MFE), sequence motifs, misincorporation profile and mismatch rate that characterise regions containing m^1A. The final software builds around the trained model to predict m^1A containing windows(40 nucleotide-wide) within user-specified coordinate bounds on the user's RNA-seq data.

Work Flow :

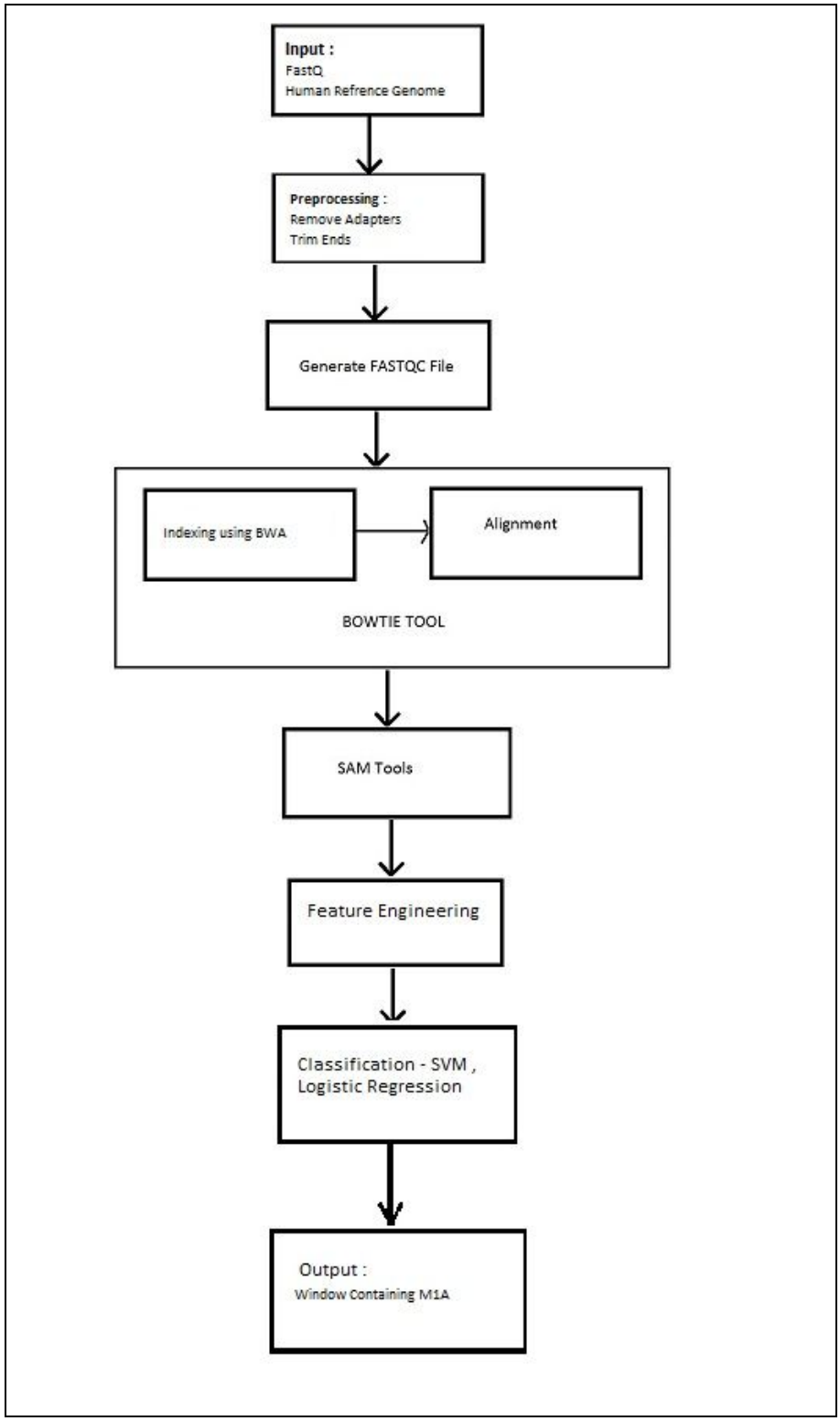

Figure 2: Workflow of Software

Input: RNA-seq reads in Fastq format; human reference genome in FASTA format.

Python modules: Scikit-Learn, Pysam.

Training dataset: RNA-seq data of 'HEPG2 mRNA untreated input replicate B' (GEO accession number GSE70485).

Tools included in the package: Bowtie2, Samtools, Bedtools, ViennaRNA.

Positive samples: We used the genome coordinates of 20 nucleotide-wide troughs provided in the study [2](Supplementary Table 6). The mean distance of the actual m^1A position from the trough was reported to be ~6-13 nucleotides away from the trough coordinates(Supplementary Note 3); In order to have a higher confidence in the windows actually capturing m^1A, we decided to extend the window sizes by 10 nucleotides both upstream and downstream, to generate 40 nucleotide-wide windows. After filtering samples with missing feature values, 1188 positive samples were used.

Negative samples: We selected 40 nucleotide-wide windows neighbouring(within 40 nucleotides) the positive instances that do not overlap with it. After filtering samples with missing feature values, 1026 negative samples were used.

Preprocessing of Data: Cutadapt was used to trim adapter sequences, remove reads with length shorter than 30 bases and trim low quality bases (PHRED score < 20) from RNA-seq read ends. FastQC was used to generate a read quality report. The FASTQ file was then fed into Bowtie2 for the purpose of aligning it against the Bowtie indexed reference genome(default parameters). Bowtie generates a SAM file which is further processed by Samtools to retain only mapped reads in the BAM file.

Features selection:
As discussed in the study[2], troughs (indentation in the middle of a peak) are observed close to the known m^1A sites in the data obtained from m^1A-seq. We expect a similar distribution of per-base coverage in regular RNA-seq data as well, although the troughs may be less pronounced because the mRNA was not enriched by immunoprecipitation. Therefore, we used the coverage at each position within a window as a feature. We included mismatch rate as a feature because m^1A is known to generate higher mismatches with a characteristic profile during reverse transcription [3]. We used GC content as a feature because the GC content is expected to be higher in regions containing m^1A modification[6]. Finally, we incorporated MFE, a measure of the the secondary structure stability of the RNA molecule, as a feature because m^1A has been reported in literature as being involved in stabilizing RNA conformations[9].

Feature Extraction:
During the training phase we perform feature extraction directly on each fixed window in the positive and negative samples. During the testing phase, we used a sliding window of 40 with a step size of 10 across the region specified by the user for feature extraction. The coverage and base nucleotide were extracted for each position in the window from the BAM file using pysam.The GC content was obtained from the reference genome sequence. MFE was calculated using Vienna RNA, based on the sequence in the window on the reference genome and which strand this sequence maps to (determined by the orientation of the closest RNA-seq read). Windows containing zero coverage at any position and windows devoid of base nucleotide A were discarded. Finally, a file was generated for each feature containing information about the position and the value of the feature.

Features of Input Data :

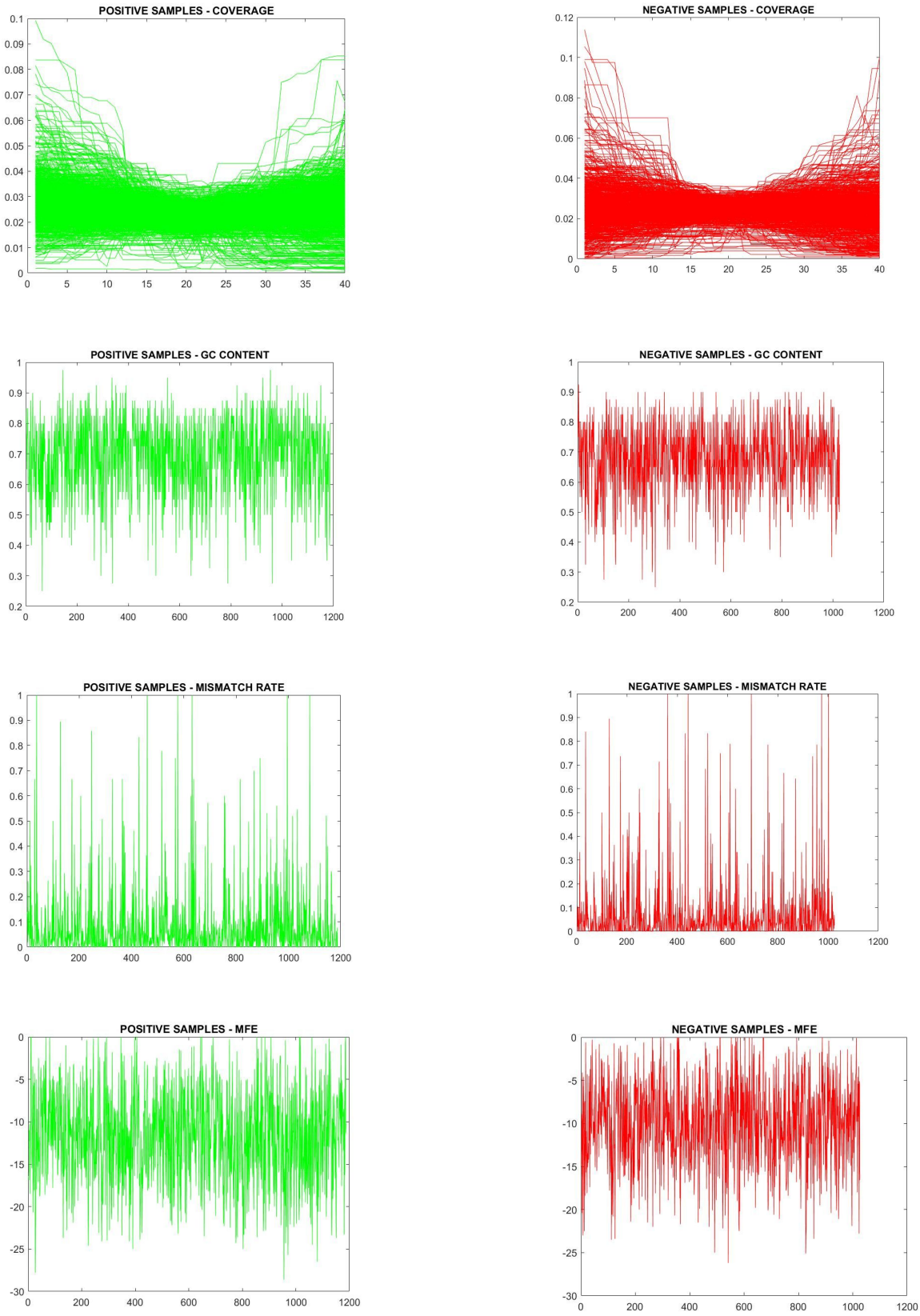

Figure 1**:** Feature Values vs. Sample ID

Classification

Classifier 1 Logistic Regression :

Logistic regression is a binary classification method that utilizes the sigmoid curve and constrains the estimated probabilities to lie between 0 and 1. Logistic Regression in sklearn implements regularized logistic regression taking a dense matrix as an input. This dense matrix represents the features of the data. The logistic regression learns the weights to fit the model given by the following equation.

$$\sigma(t) = \frac{e^t}{e^t + 1} = \frac{1}{1 + e^{-t}}$$

$$t = \beta_0 + \beta_1 x$$

$$F(x) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 x)}}$$

The features are converted into a polynomial features (quadratic fit) before feeding to the classifier.
The parameter Beta is the weight vector in this case and ' x' is the dense matrix consisting of polynomial fit of degree 2 of the following features : coverage, GC content, MFE , Mismatch rate. On training the classifier learns the Beta vector and during the testing phase , F(x) is calculated to classify the window to be containing M^1A or not containing M^1A.

Classifier 2 SVM :

Support Vector Machine constructs a hyperplane in a N-dimensional space to classify the data into two classes. 'N' is equal to the number of features describing the input samples. It may to be always necessary that the data is perfectly linearly separable and the hyperplane needs to follow a polynomial curve to classify the data.

$$w \cdot x_i + b$$

Here w is the weight vector that the classifier learns such that every positive sample yields a +1 and negative would yields a -1.We have fitted a SVM model of order 3 using Scikit -learn on the data.

Features are of all the samples are converted into a dense matrix for feeding to the classifier. The coverage for each position is normalized for each window.For mismatch content of each window, the highest mismatch rate given that the corresponding base nucleotide is A are calculated. The trained models are saved to be later loaded a spickle files during testing.

Final labels:

Since the accuracy of both the classifiers is comparatively low, we combine the results of the Logistic Regression and SVM to label the final data.
The windows classified as positive by both the Logistic Regression as well as the SVM are labelled as positives in the final result. This helps to improve the the accuracy of the classifier.
The output file consists of a list of windows containing M^1A.

Evaluation

Logistic Regression (Leave 1 out 10 fold Cross validation) :

	Precision	Recall	F1- Score	No.of Samples
Negative	0.65	0.69	0.67	1026
Positive	0.72	0.68	0.70	1188
Total	0.68	0.68	0.68	2214

Table 2 : Evaluation matrix for Logistic Regression

Logistic Regression Accuracy: **68.291%**

SVM Order 3 (Leave 1 out 10 fold Cross validation) :

	Precision	Recall	F1- Score	No.of Samples
Negative	0.52	0.64	0.58	1026
Positive	0.61	0.49	0.55	1188
Total	0.57	0.56	0.56	2214

Table 3 : Evaluation matrix for SVM

SVM Accuracy : **56.187 %**

Final Accuracy :

True Positives	393	False Positives	130
True Negatives	433	False Negatives	218
Total Correct Labels	826	Total Incorrect Labels	448

Table 4 : Evaluation matrix for Final Labels

Final accuracy= **70.35 %**

Other Classifiers Tested:

Classifier	Accuracy
K-Means CLustering	0.334782520077
Decision Trees	0.636856519209

Other Teams' Softwares

Team A:

Team A's software predicts pseudouridines in user-provided nucleotide sequences. It utilizes a pre-trained SVM model for this task. According to our understanding, initially a multiple sequence alignment is performed and then a consensus sub-sequence(of length unknown) neighbouring the base of interest is used as a feature to train the model. Additionally, a metric of the secondary structure stability(perhaps MFE) is also included; although, since no tool is included in the software package for such a purpose, it remains unclear how the metric is calculated for the user-provided test data. The softwares performs fast for the first sample command and generates as output fasta sequences with the character 'Y' representing predicted sites of pseudoUridylation. However, when the '-p' or '-t' option is specified, the program terminates with an error directed at the sklearn library:

'sklearn.exceptions.NotFittedError: predict_proba is not available when fitted with probability=False.'

The readme file is well documented and is sufficient to get the software running on the basic functionality. One major drawback of the software is that the predictions do not take into account any features from the RNA-seq experiment and therefore are not applicable to the task of detecting dynamic pseudouridylation. One possible suggestion is test Hidden Markov Model as the prediction method, considering their suitability and proven performance for such tasks on biological sequences.

Team B:

Like our software, Team B's pipeline also focuses on the detection of m^1A region. The following paragraph summarizes the salient distinctions between the two pipelines.

A smaller window size (20 nucleotides) was used compared to ours (40 nucleotides). There is a distinct possibility of biasing the learned discriminative model because the training data consists of 20% positive samples and 80% negative; in contrast, our team's training data consists of roughly equal numbers of positive samples and negative samples. One advantage of the Team B's pipeline is that it allows users to train the classifiers , whereas ours only uses pre-trained models. This is important as, the model performance is sure to improve with every additional training sample. On the other hand, Team B's pipeline doesn't include data pre-processing (generate BAM file from RNA-seq reads) in the pipeline and the user has to download and execute these requisite tools themselves. One of the most important distinguishing features between the two models is that Team B used fewer only two features (coverage and GC content) whereas our model uses for features. Finally, during testing, Team B's pipeline requires the user to obtain the coverage separately.

We found that there were certain key ideas that our software should adopt. Of these, an option in the pipeline to allow the users to train the classifiers is the most important. Team B's user manual contains detailed instructions for installation; we definitely need to improve on this account and render our user-manual more clear. Last, we report that taking inspiration from the high accuracy of Team B's model, we too tried random forest model on our dataset(accuracy reported above).

We would like to suggest to theam B that they make the usage of rf.py easily accessible since it is buried in the user manual. We could not find the usage of *getPositiveCoverage.py* and *getNegtiveCoverage.py* in the user manual. It is unclear how the user can get coverage during testing. In general we would like to suggest them to try to automate the process of getting coverage.

Future Directions

We intend to make the pipeline compatible with different versions of Python. We plan on including the sequence motif as a feature and train the data on different classifiers trying to get a better accuracy. We would train the model using different organisms (eg. mouse, yeast) and incorporate into the pipeline.We plan on making the pipeline user friendly by adding interactive dialogue boxes and allowing the user to choose what stage he wishes to implement.

Reference

1. Baxter-Roshek JL, Petrov AN, Dinman JD (2007) "Optimization of Ribosome Structure and Function by rRNA Base Modification". *PLOS ONE* 2(1): e174.
https://doi.org/10.1371/journal.pone.0000174

2. Karijolich J, Yu Y-T. Spliceosomal snRNA modifications and their function. *RNA biology*. 2010;7(2):192-204.

3. Hauenschild, Ralf et al. "The Reverse Transcription Signature of *N*-1-Methyladenosine in RNA-Seq Is Sequence Dependent." *Nucleic Acids Research* 43.20 (2015): 9950–9964.

4. Machnicka MA, Milanowska K, Osman Oglou O, et al. MODOMICS: a database of RNA modification pathways—2013 update. *Nucleic Acids Research*. 2013;41 (Database issue): D262-D267. doi:10.1093/nar/gks1007.

5. Klungland, A., & Dahl, J. A. (2014). Dynamic RNA modifications in disease. *Current Opinion in Genetics & Development, 26*, 47-52. doi:10.1016/j.gde.2014.05.006.

6. Dominissini, Dan et al. "The Dynamic N^1-Methyladenosine Methylome in Eukaryotic Messenger RNA." *Nature* 530.7591 (2016): 441–446.

7. Li, X. et al. Transcriptome-wide mapping reveals reversible and dynamic N(1)-methyladenosine methylome. Nat. Chem. Biol. 12, 311–316 (2016).

8. Ryvkin P, Leung YY, Silverman IM, et al. HAMR: high-throughput annotation of modified ribonucleotides. *RNA*. 2013;19(12):1684-1692. doi:10.1261/rna.036806.112.

9. Li, X., Xiong, X., & Yi, C. (2016). Epitranscriptome sequencing technologies: decoding RNA modifications. *Nature Methods, 14*(1), 23-31. doi:10.1038/nmeth.4110

10. Helm, Mark, Juan D. Alfonzo, Posttranscriptional RNA Modifications: Playing Metabolic Games in a Cell's Chemical Legoland, Chemistry & Biology, Volume 21, Issue 2, 20 February 2014, Pages 174-185, ISSN 1074-5521, https://doi.org/10.1016/j.chembiol.2013.10.015.