Carnegie Mellon University

03713-A4: Bioinformatics Data Integration Practicum

Analysis of eCLIP data to determine RNA-protein interactions in SARS-CoV2

FINAL REPORT

Group 3 Team members:

Arnav Gupta

Tianyi Fei

Ketaki Ghatole

Yuewei Fei

Table of Content

Introduction

The novel coronavirus SARS-CoV-2 led to global outbreak of coronavirus disease 2019 (COVID-19). Understanding the underlying mechanisms of the infection can help in developing new methods for prevention and treatment. RNA binding proteins play an important role in RNA regulation, metabolism and interactions. Predicting the interactions between these viral RNAs and host proteins can facilitate a better understanding of viral infection mechanisms. Thus, we have built the pipeline 'eRNApredict' to identify the RNA binding sites for proteins of interest using eCLIP data from ENCORE. Our pipeline has two main components:

The PART I aims to identify uniquely mapped reads to generate a list of potential binding sites also known as peaks. This pipeline integrates a large number of bioinformatics packages.This pipeline begins with the fastq files from ENCORE as input. The quality of these sequences are improved for enhanced mappability by extraction of their adapter sequences. The reads are then mapped to the human genome. The aligned reads undergo further processing to make them more usable. These reads are used as inputs for peak calling to identify clusters of locally enriched read density. The peaks are then used in PART II.

The PART II is designed to train a classifier for distinguishing between peaks from background sequences. It can predict the probability of a given sequence binding to a particular transcription factor. The trained model can then be used to predict binding sites between protein and RNA. This model is based on convolutional neural networks. The whole model is based on pytorch architecture and can be used to predict binding motifs. Given the viral RNA sequence and a host protein we can predict the corresponding RNA-Protein interactions.

Thus, the results obtained from this pipeline can be used to predict if a host protein will bind to the viral RNA and the peaks will highlight the regions to which they are more likely to bind.

Materials and Methods

Input Data:

1. The eCLIP experiment of HepG2 against RBFOX2 was obtained from ENCODE.

Forward read: ENCFF172GUS to be saved as **RBFOX2.r1.fastq.gz**
Reverse read: ENCFF647KDW to be saved as **RBFOX2.r2.fastq.gz**

2. The STAR alignment tool to generate genome indices that are utilized in mapping with the following input files:

Reference genome (of human) in *FASTA* format from Gencode to be saved as **hg38.fa**
Annotations in *GTF* format from Gencode be renamed to **hg38_refGene.gtf**
The repbase file can be downloaded as **repbase.fasta** from the Github

3. The test RNA sequence data required for the pipeline are freely available on public databases and is provided in the git repository under the fasta folder.

The RNA sequence file of SARS CoV-2 can be downloaded as **COVID.fasta** from the Github

These files should be saved in the $PROJECT on Bridges2 after creating dictionaries **'eclip'**, **'scripts'**, **'DataSet'**, **'STAR'**, **'learn'**,**'hg38'** as represented in the following flowchart.

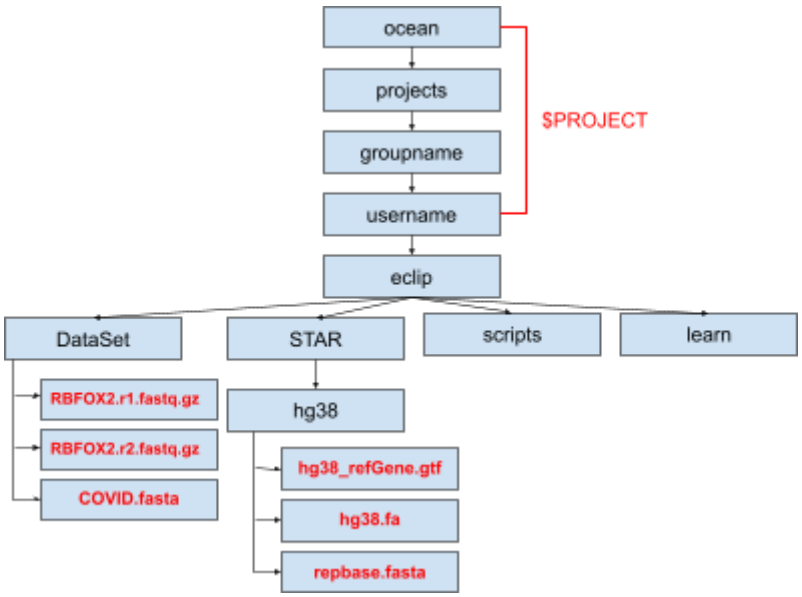


Fig 1: Directory Structure

Scripts

1. The Job scripts are present in the **sbatch** directory on Github and should be stored in the $HOME directory on Bridges2

2. The codes required to run the pipeline are present in the scripts directory on Github and should be stored under the directory **scripts** on $PROJECT

3. All scripts for model building and training are present in the the **learn** directory from Github should be stored under the directory **learn** on $PROJECT

Packages

UMI-Tools

Unique Molecular Identifiers (UMIs) are small sequences that are sequenced to handle the high rate of duplication in CLIP experiments. Each read has a unique UMI, making it possible to identify the duplicates. **UMI-Tools** can extract the UMIs from reads and remove PCR duplicates.

For more information refer:
https://umi-tools.readthedocs.io/en/latest/QUICK_START.html
https://genome.cshlp.org/content/early/2017/01/18/gr.209601.116.abstract

cutadapt

Adapters play an important role in sequencing and amplification. Reading of these adapter sequences by sequencers can lead to their presence in the data reducing its quality. **Cutadapt** can find and remove adapter sequences, primers, poly-A tails and other types of unwanted sequence from the high-throughput sequencing reads

For more information refer:
https://cutadapt.readthedocs.io/en/stable/
https://journal.embnet.org/index.php/embnetjournal/article/view/200

fastq-tools

This package provides a number of efficient programs to perform common tasks with high throughput sequencing data in the FASTQ format. In our pipeline, we use **fastq-tools** to sort fastq reads to reduce randomness.

For more information refer:
https://github.com/dcjones/fastq-tools

STAR

Spliced Transcripts Alignment to a Reference (STAR) can map sequence data to a reference genome. It creates genome indices using the reference genome after mapping to the organism specific RepBase to remove repetitive elements. It can then map unique reads to the reference genome and outputs alignment bam files.

For more information refer:
https://physiology.med.cornell.edu/faculty/skrabanek/lab/angsd/lecture_notes/STARmanual.pdf
https://www.ncbi.nlm.nih.gov/pubmed/23104886

samtools

Sequence Alignment Map tools (Samtools) works with processing of alignment files. We use Samtools in our pipeline to ensure read pairs are adjacent, to sort our outputs and make bam index for the next steps.

For more information refer:
https://github.com/samtools/samtools
https://www.ncbi.nlm.nih.gov/pubmed/19505943

Barcode collapse

Barcodes are small sequences intentionally sequenced during multiplexing so that they can be de-multiplexed into individual read libraries. **Barcode collapse** can identify barcodes in a bam file and removes duplicates by merging reads with the same barcodes.

For more information refer:
https://github.com/YeoLab/eclip/blob/master/bin/barcodecollapsepe.py

bamCoverage

This tool takes an alignment of reads or fragments as input (BAM file) and generates a coverage track bigWig file as output. Coverage is calculated as the number of reads per bin**.** The bigwig files can be used for visualization of UCSC browser of IGV browser

For more information refer:
https://deeptools.readthedocs.io/en/develop/content/tools/bamCoverage.html

PureClip

PureCLIP is a tool to detect protein-RNA interaction footprints from single-nucleotide CLIP-seq. We use PureClip in our pipeline to visualize peaks from bigwig files.
For more information refer:
https://pureclip.readthedocs.io/en/latest/

BedTools

BedTools offers a wide-range of genomics analysis. We use this tool to generate fasta files from our bam files for further analysis
For more information refer:
https://bedtools.readthedocs.io/en/latest/

Pytorch

Pytorch is an open source machine learning library for deep learning. We use this package to build and train neural networks. For more information refer: https://pytorch.org/docs/stable/index.html

Package Installations

1. Packages pre-installed on PSC

Bridges-2 has a list of commonly used softwares which can be directly used without the need of installation. To run our pipeline, we use the **Anaconda3**, **Cutadapt**, **fastq-tools**, **STAR**, **samtools** packages on PSC

2. Additional packages not available on PSC

Conda environments are created using the installed Anaconda manager for easy installation of these additional packages. The YML files to set up these environments can be downloaded using Github.

Three environments are needed to be set up for the pipeline to execute. Steps for this setup are:

1. Download the YML files and place them in the $HOME directory of bridges2
2. Load Anaconda3 using the command:

   ```
   module load anaconda3
   ```

3. Create the conda environments using the commands:

   ```
   conda env create -f environment_eclip.yml
   conda env create -f environment_bc.yml
   conda env create -f pytorch.yml
   ```

Overview of the Pipeline Implementation

The pipeline uses packages like STAR and PureClip that have a high memory requirement. Therefore, we are using the Bridges2 supercomputing systems to implement our pipeline. The execution time of the entire pipeline is estimated to be 60-80 minutes.

We have the forwards and reverse reads that are used as input to our pipeline. Adapters are included during the sequencing and amplification of these reads. These adapters are sometimes read by the sequencers and their presence can reduce the quality of reads. So, the first step in our pipeline is to remove these adapter sequences using **UMI_tools and Cutadapt**. Next, we sort these reads to reduce randomness using **fastq-tools**. The reads are now ready to be mapped to the reference genome. We use Spliced Transcripts Alignment to a Reference - **STAR** for mapping. We perform two runs of STAR, first to remove PCR duplicate elements. The second run is used to map the reads to the human reference genome. After sorting the alignment file using **samtools,** we use **Barcode collapse** to identify barcodes and remove duplicates by merging reads with the same barcodes. After sorting the reads again using **samtools**, we use these sorted reads to obtain a coverage file using **bamCoverage.** Coverage is calculated as the number of reads per bin. The coverage files can be used for visualization of peaks on the UCSC browser or IGV browser. The sorted reads obtained from samtools are also used to get the bed files of the peaks using **PureClip** that are required for building the classifier in PART II of the pipeline. We have also used bedtools to convert the bed files into fasta files to use **MEME suite** for motif discovery for verification purposes for PART1.

In PART II of the pipeline, we use the peak files obtained from PART I to train a classifier to distinguish between peaks from background sequences. This pipeline uses a neural network model that directly takes the sequences as input and outputs the probability of the sequence being a peak. The model consists of an **embedding layer** that encodes every base from the input sequence into a vector of length 8. This embedded representation is then fed into three **1D convolutional layers** and **2 fully connected layers**. The final output is then passed through a **sigmoid function** to output probabilities. The model is trained using identified peak sequences as positive samples and random sequences as negative samples. The final loss is calculated by **BCEwithlogits** loss. The SARS-CoV2 RNA sequence and an additional eClip data from ENCORE is used as an input to test this model. We can analyze these results to predict the binding of this protein to the SARS-CoV2 RNA sequence.

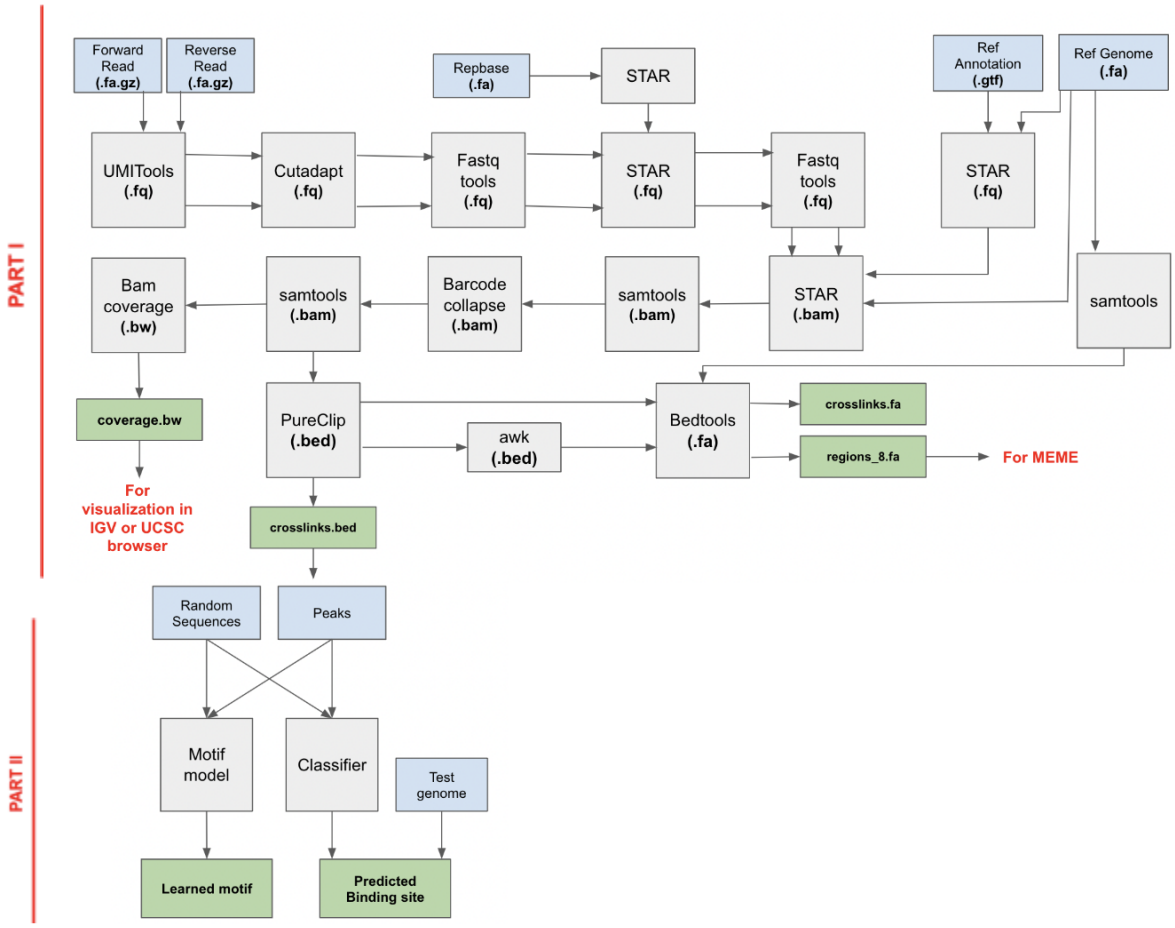


Fig 2: eRNApredict pipeline flow

Running the Pipeline

The instructions for preparing the job files can be found in the user manual of the pipeline. Majorly the whole pipeline can be run in the three commands on bridges2 supercomputer.

1. Running genome_index.job

Run the following command in $HOME

```
sbatch genome_index.job
```

This will create the necessary STAR files for the next sbatch file

2. Running master.job

Run the following command in $HOME

```
sbatch master.job
```

This will create the final output of PART I.

3. Running model.job

Run the following command in $HOME

```
sbatch model.job \
$PROJECT/eclip/output/bed/crosslinks.bed \
$PROJECT/eclip/DataSet/COVID.fasta \
Rbfox2.fa \
./figures
```

Results

PART I

The pipeline makes an additional directory output on $PROJECT to store all the outputs. The coverage.bw output file is in bigWig format. The bigwig file can be viewed on Integrative Genomics Viewer (IGV). The IGV is a high-performance interactive tool for the visual exploration of genomic data. From the coverage we observe that the alignment of eclip reads on the human hg38 reference genome. For best performance of IGV, use the same genome fasta that you used to generate the bigwig files in module.job. Fig 3 shows RBFOX2 binds on the human genome and Fig 3 shows where DDX3X binds on the human genome.

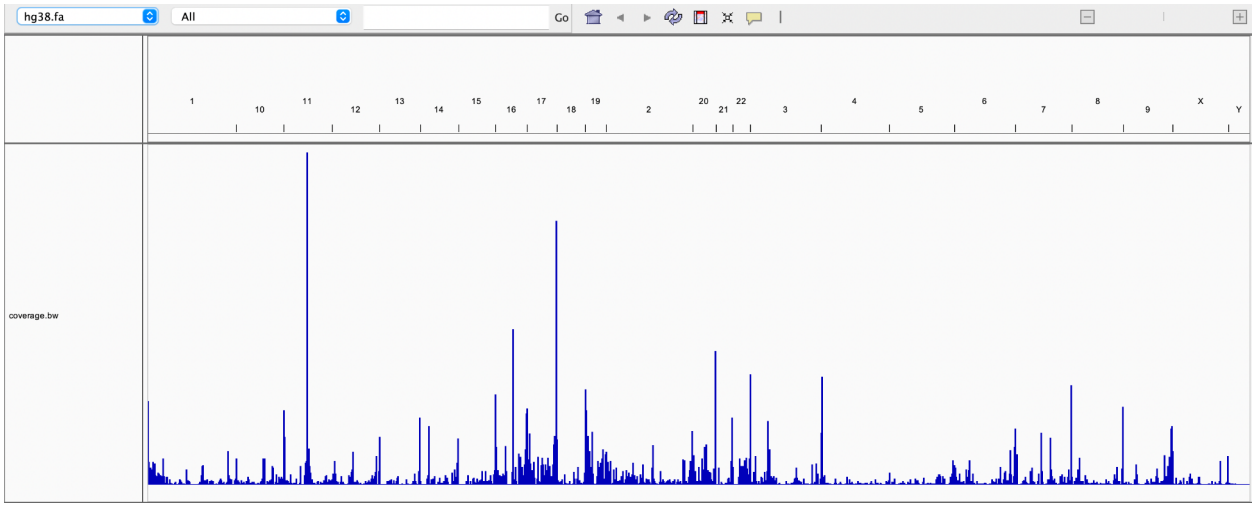


Fig 3: COVERAGE RBFOX2 (IGV)

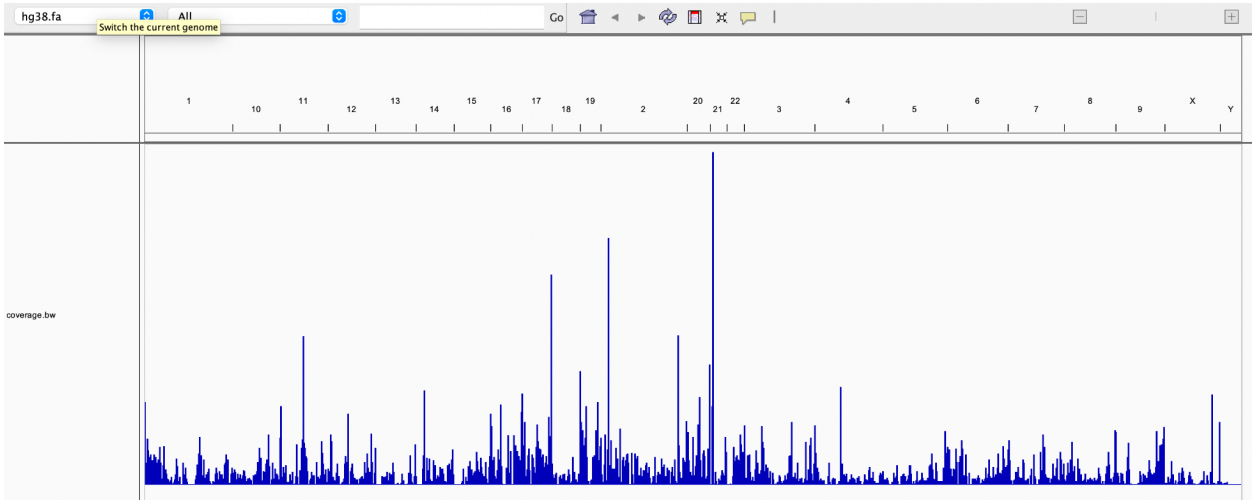
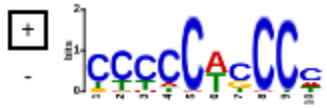
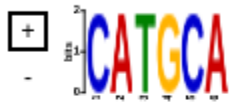
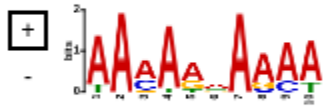


Fig 4: COVERAGE DDX3X (IGV)

To further verify the peaks we called, we compared the bed file generated with our pipeline with given reference peaks downloaded from ENCORE. The main difference between our two pipelines is that we used pureclip for peak calling, didn't use multiple replicates and input controls. The original pipeline provided by ENCORE used a different software, clipper. Clipper gives much more peaks. Clipper gives 92550 peaks for DDX3X and 143521 peaks for RBFOX2 while pureclip gives 7563 peaks for DDX3X and 11792 peaks for RBFOX2. Then we compared the overlapping of our peaks with the reference peaks. First, we extend our peaks to a length of 80nts. Then '*bedtools closest'* is used to find all overlapping peaks. For RBFOX2, of all 11792 peaks, 7548 peaks find overlapping peaks in the reference, which is about 64%. For DDX3X, of all 7563 peaks, 5326 peaks find overlapping peaks, which is about 70%. This clearly demonstrates that even though the number of peaks called is significantly less the quality of peaks are still good. Perhaps increasing more replicates and having input controls for filtering out background peaks, will make this match the clipper reference.

Then we use MEME to extract motifs from the peaks we called. We first extend every peak to 80nts and use this fasta file for MEME input. For RBFOX2, we can get the following motifs. Of the 5 motifs, the second motif matches perfectly with the RBFOX2 binding motif GCAUG. This indicates that the peaks we identified are reliable.

	Logo	E-value	Sites	Width	More	Submit/Download
1.		1.9e-120	4991	10		
2.		4.8e-061	2338	6		
3.		2.1e-011	511	10		
4.		1.9e-008	1201	10		
5.		2.2e-001	161	10		

Stopped because requested number of motifs (5) found.

Fig 5: MEME RBFOX2

PART II

Using the peaks from Part I as positive samples, we trained a classification model. Each peak is trimmed to 80 nts as input. We also compared our model with an existing model, RNAprot. Both models can achieve around 90% accuracy during training. Thus, the performance of our model is guaranteed.

When testing on the COVID-19 genome, we use a sliding window with stride 20 to scan through the whole genome sequence. To normalize the results, the predicted probability is then compared with the geometric mean of positive samples. This normalized value across the whole genome is shown in the figure below. We can see clear peaks where the protein is more likely to bind.

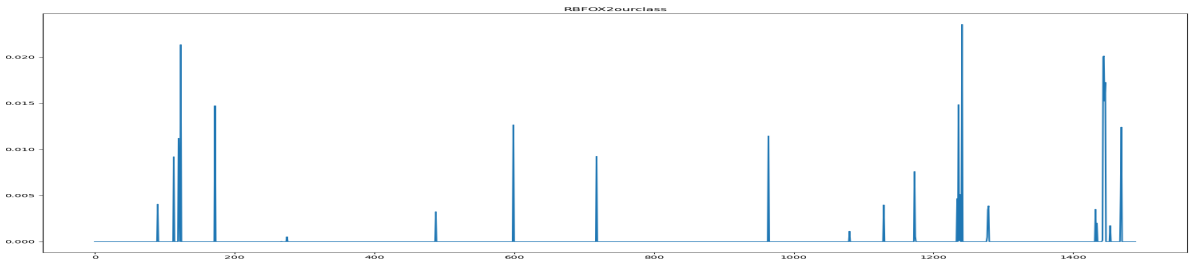


Fig 5: RBFOX2 - SARS CoV-2 RNA binding predictions

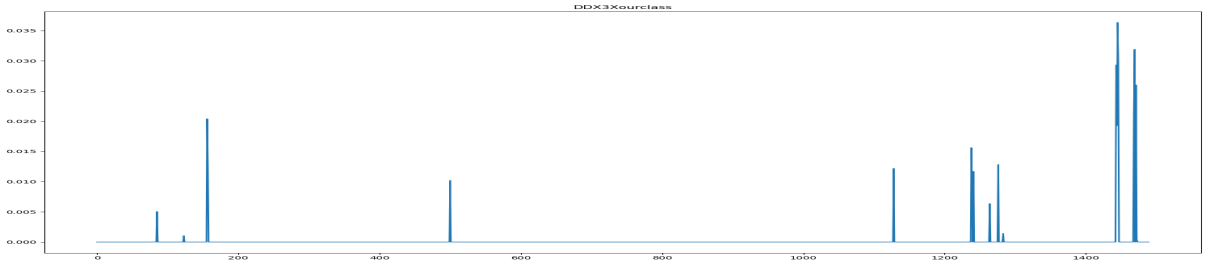


Fig 6: DDX3X - SARS CoV-2 RNA binding predictions

Beside the classification model, we also trained a dedicated motif model. From the training results, we may find motifs similar to the RBFOX2 binding box, which is CATGCA. As for DDX3X, this is a DEAD-box helicase and no specific binding motifs are found. Therefore, we turned to another protein PCBP2. This is a polyC binding protein. Form the results of PCBP2, we can find motifs enriched with C.

Fig 7: RBFOX2 predicted motif

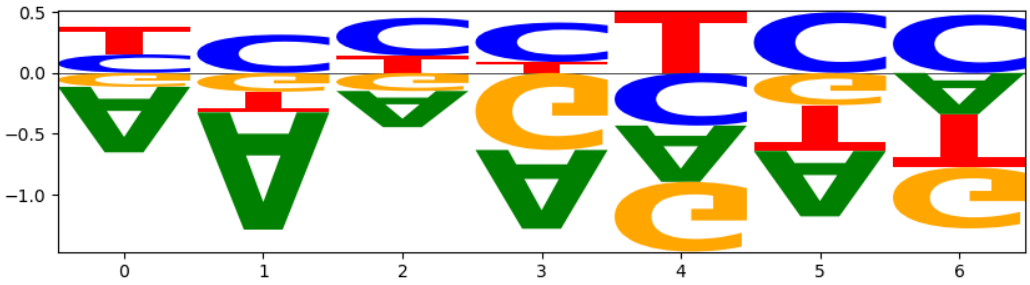


Fig 8: PCPB2 predicted motif

Discussion

In our project, we designed and implemented a pipeline that predicts RNA-protein binding sites. The whole pipeline has two main components: Part I aims to identify peaks from raw sequencing data and part II uses identified peaks to train classifiers. This classifier can then be used to predict binding sites of the given protein on a new genome. We tested the pipeline using different proteins and the results are verified. Now, the whole pipeline is implemented and ready to use on bridges.

For part I, we mainly referenced the ENCORE pipeline. The main difference is that we used pureclip for peak calling instead of clipper. As shown in the results, the called peaks are quite different. This suggests that the algorithm used to call peaks has a great influence on the final results. Therefore, we have to choose softwares carefully so that it matches the tasks and data.

In our original pipeline, we skipped some of the steps and did not use the repbase to filter out repeating sequences in the raw reads. In such cases, the called peaks are of poor quality. Pureclip gives fewer confident peaks and the overlapping between our results and the reference is only about 30%. After making up all these steps, the results improved significantly. The number of confident peaks increased about 2 fold and the overlapping increased to around 70%.

Right now, we have already implemented the whole pipeline, but there's still much we can explore. We have already shown using different peak calling softwares or whether using the repbase can cause a great difference. Actually, there are many more peak calling softwares or read mapping softwares we may try. Besides, some steps such as removing PCR duplicates are applied in every pipeline, we would also like to test the influence of not taking these steps.

For part II, originally we want to train a classifier that may predict whether a protein will bind the genome or not. We specifically picked out proteins that are reported to bind with COVID-19 RNA for experiments. However, different proteins may have quite different properties. It could be hard to directly conclude whether the protein could bind (what threshold should we set). Besides, many other factors may also influence the final result, such as the protein and the RNA can be spatially separated or the predicted binding site actually folds into a 3D structure that prevents the protein from binding. Therefore, in our project, we focus more on predicting the sites where the protein is more likely to bind for a given genome.

Compared with the previous results, binding site prediction is much more precise. This is mainly because pureclip is more strict. We called around 10000 peaks for each protein while the original encore file has about 60000 peaks. With more accurate training samples, the model may do a better job of predicting. Thus, our result showed less noise.

As for the motif finding using deep learning, we may clearly find related motifs in the filters learned by the model. The results match the MEME output. However, we can not get a specific P-value for each motif discovered in this way. Thus, right now, we may not be able to point out which of the 16 motifs identified by the model is the most important. There is already much work on using deep learning for motif discovery. In our project, we just tried this direction and did not go really deep. In future work, we may go deeper in this direction.