

# **RBP Functional Target Predictor using Multiple Peak Callers (RFM)**

## **Table of Contents**

### **I) Introduction**

- i)** About this pipeline
- ii)** Workflow

### **II) Packages/Installation**

- i)** Cutadapt (1.16)
- ii)** Umi\_tools (1.0.1)
- iii)** STAR (2.5.2b)
- iv)** Samtools (1.3)
- v)** FastQC (0.11.3)
- vi)** CLAM (1.2)
- vii)** CLIPper
- viii)** Bedtools (2.25.0)
- ix)** PRAS (1.0)

### **III) Using the Pipeline**

- i)** Running the Pipeline
- ii)** Input Files
- iii)** Output

### **IV) Limitations and Pitfalls**

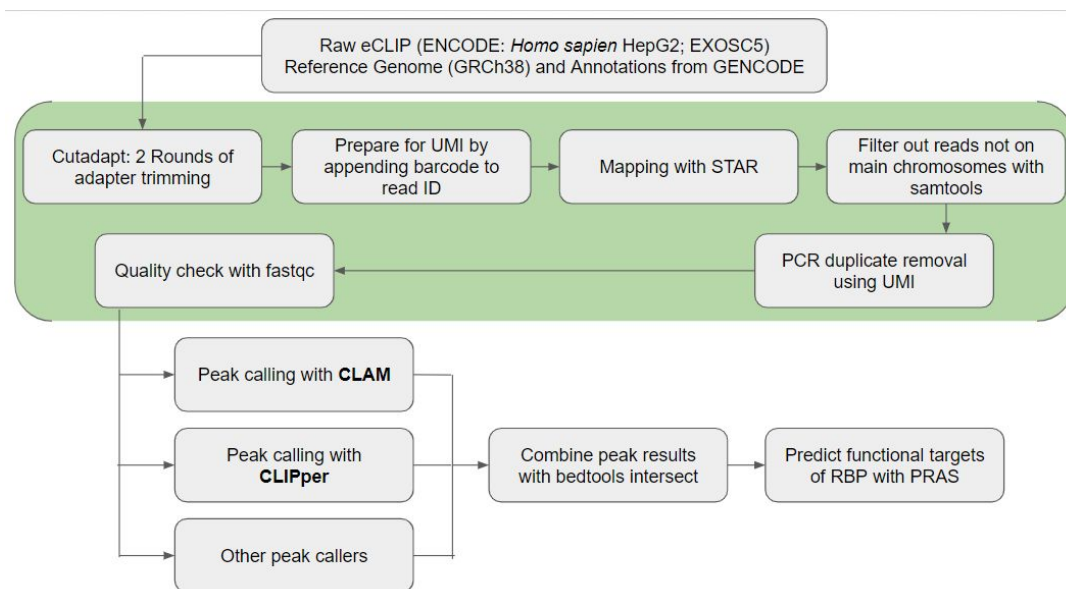
# I. Introduction

## About this Pipeline

RNA binding proteins (RBPs) are essential for the cell, playing a large role in the regulation of transcription, RNA processing, and gene expression (e.g. alternative splicing, mRNA export, translation, turnover, and localization). In order to learn more about these proteins, studies have focused on identifying RBP binding sites and interactions by utilizing methods such as cross-linking immunoprecipitation (CLIP) and other variations. The basic protocol for processing the data includes three main steps: Preprocessing, peak calling, and postprocessing. Preprocessing involves trimming off the adapters from the raw sequence reads and mapping them on a reference genome. Peak calling algorithms attempt to extract authentic binding sites and identify motifs. Postprocessing involves integrating the data and making predictions on potential binding sites. Unfortunately, the peak calling process is not perfectly accurate, containing false negatives and positives. The primary purpose of this pipeline is to utilize and combine peak calls from different software (maximum of three) as a method to improve accuracy and reduce the number of false negatives and positives. This pipeline will generate peak data using *CLAM*, so the user will need to provide one or two more peak data (bed format). In our example, we utilize raw eCLIP data of two replicates (*Homo sapien* HepG2, *EXOSC5*) and *CLIPper* data from *ENCODE*. The *CLIPper* data is then combined with the generated *CLAM* data and used in *PRAS* to make predictions on the functional targets of RBP.

## Workflow

We arbitrarily chose to utilize eCLIP data of RBP EXOSC5 from *Homo sapien* HepG2 cell line.



## **II. Packages/Installation**

*We suggest installing the tools into a python environment (anaconda3).*

### **Cutadapt**

Cutadapt removes unwanted sequence elements such as adapters, primers, and poly-A tails from raw sequence reads used as inputs. Cleaning of the input data in this way is crucial, as it removes elements which may result in false positives in read data.

<https://cutadapt.readthedocs.io/en/stable/index.html>

### **Umi\_tools**

UMI is responsible for removing PCR duplicates in the data by using mapping position and random barcodes.

<https://umi-tools.readthedocs.io/en/latest/#>

### **STAR**

Spliced Transcripts Alignment to a Reference (STAR) is a program which aligns high-throughput RNA-seq data to a reference file. This tool will allow us to map the CLIP-seq reads.

<https://github.com/alexdobin/STAR>

### **Samtools**

Sequence Alignment/Map tools (Samtools) will be used to filter the reads, leaving only reads that map against the main chromosomes. It will then be used to merge the preprocessed alignments of the replicates.

<http://www.htslib.org/>

### **FastQC**

FastQC is used to assess the quality of the preprocessed data. This pipeline will generate the output file, but the quality assessment is done visually by the user.

<https://www.illumina.com/products/by-type/informatics-products/basespace-sequence-hub/apps/fastqc.html>

<https://www.bioinformatics.babraham.ac.uk/projects/fastqc/>

### **CLAM**

CLIP-seq Analysis of Multi-mapped reads (CLAM) is a tool for re-aligning multi-mapped reads in CLIP-seq or RIP-seq data and for calling peaks.

<https://github.com/Xinglab/CLAM#section1>

## CLIPper (Suggested)

CLIPper is another peak-caller tool for CLIP-seq dataset. It is also the peak-caller in ENCODE's CLIP-seq analysis protocol.

<https://github.com/YeoLab/clipper/wiki/CLIPper-Home>

## Bedtools

We will be using bedtools to merge the peak-caller outputs (.bed format) by the *intersect* function.

<https://bedtools.readthedocs.io/en/latest/>

## PRAS

Protein-RNA Association Strength (PRAS) takes in the peak data and annotation file to predict function targets for the RBP, scoring the genes based on binding intensity and position information.

<https://github.com/ouyang-lab/PRAS>

# III. Using the Pipeline

## Running the Pipeline

Please follow the instructions below or in the README.md at:

[https://github.com/shuangerli/bio\\_practicum](https://github.com/shuangerli/bio_practicum)

- Clone the repo: `git clone https://github.com/shuangerli/bio_practicum.git`
- Make the script executable (if necessary): `chmod 755 run.sh` or `chmod 755 run`
- Run the script with: `run [argument]` or `slurm run [argument]`
- Please use `run -h` or `./run -h` to see instructions
- To run the pipeline with example data (will download the necessary files), use `run -example`. *Note: You do not need to provide your own input files here.*

## Input Files

Human Reference Genome (fasta)

- <https://www.gencodegenes.org/human/>

Annotation (gtf) & ID file (txt)

- <https://genome.ucsc.edu/cgi-bin/hgTables>

- To get the annotation file, set the parameters of the table to be as follows:

**Table Browser**

Use this program to retrieve the data associated with a track in text format, to calculate intersections between tracks, and to retrieve DNA sequence covered by a track. For help in using this application see [Using the Table Browser](#) for a description of the controls in this form, the [User's Guide](#) for general information and sample queries, and the OpenHelix Table Browser [tutorial](#) for a narrated presentation of the software features and usage. For more complex queries, you may want to use [Galaxy](#) or our [public MySQL server](#). To examine the biological function of your set through annotation enrichments, send the data to [GREAT](#). Refer to the [Credits](#) page for the list of contributors and usage restrictions associated with these data. All tables can be downloaded in their entirety from the [Sequence and Annotation Downloads](#) page.

clade:  genome:  assembly:

group:  track:

table:

region: ☒ genome ☐ position

identifiers (names/accessions):

filter:

subtrack merge:

intersection:

correlation:

output format:  Send output to ☐ [Galaxy](#) ☐ [GREAT](#)

output file:  (leave blank to keep output in browser)

file type returned: ☒ plain text ☐ gzip compressed

To reset all user cart settings (including custom tracks), [click here](#).

- To get the ID files, use the same parameters, but change the output format to be also specify the fields to only contain:

**output format:**

<input type="checkbox"/>	bin	
<input checked="" type="checkbox"/>	name	Name of gene (usually transcript_id from GTF)
<input type="checkbox"/>	chrom	Reference sequence chromosome or scaffold
<input type="checkbox"/>	strand	+ or - for strand
<input type="checkbox"/>	txStart	Transcription start position (or end position for minus strand item)
<input type="checkbox"/>	txEnd	Transcription end position (or start position for minus strand item)
<input type="checkbox"/>	cdsStart	Coding region start (or end position for minus strand item)
<input type="checkbox"/>	cdsEnd	Coding region end (or start position for minus strand item)
<input type="checkbox"/>	exonCount	Number of exons
<input type="checkbox"/>	exonStarts	Exon start positions (or end positions for minus strand item)
<input type="checkbox"/>	exonEnds	Exon end positions (or start positions for minus strand item)
<input type="checkbox"/>	score	score
<input checked="" type="checkbox"/>	name2	Alternate name (e.g. gene_id from GTF)
<input type="checkbox"/>	cdsStartStat	enum('none','unk','incmpl','cmpl')
<input type="checkbox"/>	cdsEndStat	enum('none','unk','incmpl','cmpl')
<input type="checkbox"/>	exonFrames	Exon frame {0,1,2}, or -1 if no frame for exon

### Raw CLIP-seq Data (fastq)

- This pipeline requires the user to use two replicates, each with read1 and read2 (four total files).

### Peak Data (bed)

- This pipeline requires the user to provide additional peak data (at most 2) in bed format. We have tested and suggest utilizing *CLIPper*. The workflow (with its own preprocessing steps using *GSNAP*) can be found at: <https://github.com/YeoLab/clipper/wiki/CLIPper-Home>.

- Another option is to download the *CLIPper* data directly from ENCODE (narrow\_peak)

## Output

- To manually assess the quality of *CLAM* peak data, we included FastQC output in the /fastqc\_result directory.
- The final output of the pipeline is binding\_PRAS.txt, which will contain the predicted functional targets of the RBP with scores.

## IV. Limitations and Pitfalls

1. In the example run, downloading the files sometimes encounter issues that the downloaded file got broken. If that happens, you will see downloading and cutadapt complete extremely fast (under 1min). In this case, please download the files manually by the following commands (available on the README.txt too):

```

1 mkdir rep1 rep2
2 wget -O rep1/reads.R1.fastq.gz https://www.encodeproject.org/files/ENCFF1730DI/@download/ENCFF1730DI.fastq.gz
3 wget -O rep1/reads.R2.fastq.gz https://www.encodeproject.org/files/ENCFF507EGU/@download/ENCFF507EGU.fastq.gz
4 wget -O rep2/reads.R1.fastq.gz https://www.encodeproject.org/files/ENCFF578XQC/@download/ENCFF578XQC.fastq.gz
5 wget -O rep2/reads.R2.fastq.gz https://www.encodeproject.org/files/ENCFF735QHY/@download/ENCFF735QHY.fastq.gz
6 wget -O annotation.gtf.gz ftp://ftp.ebi.ac.uk/pub/databases/genCODE/GenCODE_human/release_33/genCODE.v33.primary_assembly.annotation.gtf.gz
7 gunzip annotation.gtf.gz
8 wget -O ref.GRCh38.fa.gz ftp://ftp.ebi.ac.uk/pub/databases/genCODE/GenCODE_human/release_33/GRCh38.primary_assembly.genome.fa.gz
9 gunzip ref.GRCh38.fa.gz
10 wget -O clipper.bed.gz https://www.encodeproject.org/files/ENCFF862EKD/@download/ENCFF862EKD.bed.gz
11 wget -O control.bam https://www.encodeproject.org/files/ENCFF608WNY/@download/ENCFF608WNY.bam

```

2. Using peak caller output files in different orders currently yields unique output files. This is to say that inputting *CLIPper* data and then *CLAM* data yields results which are not identical to those seen when first inputting *CLAM* data and then *CLIPper* data. The reason for this believed to be a result of the *intersect* function. For more information, please consult the following link:  
<https://bedtools.readthedocs.io/en/latest/content/tools/intersect.html>
3. A maximum of three peak callers may be utilized in this pipeline.
4. Certain combinations of peak callers will yield unsuccessful *PRAS* data, where the scores of the predicted genes will all be 0. For example, data generated by *PureCLIP*, *CLIPper*, and *CLAM* all run successfully with *PRAS* by themselves. However, all combinations with *PureCLIP* will end with scores of 0, while the combination of *CLAM* and *CLIPper* is successful. We believe this is possibly due to differing file formats, where *PureCLIP* generates BED6, and *CLAM* and *CLIPper* are in BED10 (narrow\_peak) format.