

Mini project

19336487

Published online 8 October 2018

Nucleic Acids Research, 2018, Vol. 46, No. 22 11743–11758

doi: 10.1093/nar/gky892

Integrative genomic analysis reveals novel regulatory mechanisms of *eyeless* during *Drosophila* eye development

Kelvin Yeung^{1,†}, Feng Wang^{2,†}, Yumei Li^{2,3,†}, Keqing Wang^{2,3}, Graeme Mardon^{1,3,4,*} and Rui Chen^{1,2,3,4,5,*}

Introduction

Transcription factors (TF) are proteins that can regulate gene expression by binding to specific sequences of DNA. They have a key role in cell fate specification, the process of how tissues and organs develop during the embryonic stage. The *eyeless* (*ey*) is one of the fundamental transcription factors in developing *Drosophila* eye and retinal cell fate. The *ey* gene loss of function mutations can lead to suppressing eye development in flies intensively (Lindsley et al 1992). In addition, overexpressing *ey* can grow ectopic eyes on antennae, legs, and wings in flies (Halder et al 1995). However, the exact mechanisms of how it controls other genes and pathways have not been completely understood yet. The critical purpose of this study is to how *ey* regulates the complicated process of eye formation.

This paper is crucial not only for understanding eye formation in flies but also in humans because *ey* has homology to a TF in vertebrates called *Pax6*. Aniridia, an eye disease, is one of the genetic conditions that is related to *Pax6* mutations (Jordan et al 1992). Scientists hope to help reveal insightful knowledge for *Pax6* function by understanding *ey*.

The study designed and performed a ChIP-seq experiment on eye-antenna discs in the third instar larval stage when eye development was initiated. They want to discover ey protein binding sites in the chromatin. The study results show ey binds to promoter regions more frequently. Promoters are responsible for genes turning on. Interestingly, ey was also found to bind to non-promoter sites, such as enhancers. The enhancers are DNA sequences far from the gene location that regulate gene expression. This means the ey role is more complex than was expected. One of the significant findings of this study was that ey directly regulates 311 genes. The ey enabled to formation of a vast genetic network that includes well-known genes related to eye development such as Wg, Dpp, and N pathways that confirmed the past findings about ey function. The other astonishing discovery is non-coding RNA regulatory by ey. Also, other novel pathways are found in this experiment that were unknown before.

To summarize, this study takes a much clearer picture of eye development, pathways, and gene networks. Studying ey in flies can help us understand similar processes in *Pax6*. This paper could lead to more investigations on treatments for eye development mechanisms and disorders.

Materials and methods

ChIP-Sequencing and Data Analysis

Chromatin Immunoprecipitation followed by sequencing (ChIP-Seq) is a robust method based on NGS to investigate regions of the genome that interact with specific proteins and to figure out the protein's binding sites to DNA (Leleu, Lefebvre, and Rougemont, 2010). These sites are crucial for epigenetic and gene expression studies. In this study, Yeung and colleagues employed ChIP-Seq to determine the Eyeless (ey) protein mechanism and gene expression regulation for developing eye and cell fate specification in the third stage of fruit fly larva.

First, four hundred larva eye-antenna discs in the third embryonic stage were dissected. The cross-linking process was done with 1% formaldehyde. This process conserved and stabilized the protein-DNA binding in whole chromatin. After lysing the cells and DNA extraction, they utilized sonication to break down the chromosomes into about 200-400 bp small pieces.

Then, they applied a specific antibody that recognizes ey proteins to isolate the DNA-Ey binding regions. This anti-Ey antibody was provided by Dr.Walldrof which had been confirmed in previous studies (Yeung et al 2018). Controls are a necessary part of quality checks in ChIP-Seq experiments. For control samples, they prepared no-antibody libraries. These help distinguish between true binding positions and background noise. Fowling pulling down the complexes, the samples were washed multiple times to ensure that the isolation was perfect.

Ultimately, the prepared libraries were sequenced and aligned to the *Drosophila* genome. The ChIP-seq data was uploaded to NCBI Gene Expression Omnibus (GEO) as accession number GSE112868. The project accession number is PRJNA449366 including 8 biosamples and SRA. The NGS technique that operated is Illumina single There are two replicates of ey ChIP-seq, two replicates of no-antibody samples as controls, and three ChIP-seq experiments for PolII and histone modifications. I analyzed ey Chip-seq. For this purpose, I used below biosamples:

Samples

Run	# of Spots	# of Bases	Size	Published
SRR6967847	19,093,976	954.7M	714.1Mb	2019-12-21
Run	# of Spots	# of Bases	Size	Published
SRR6967846	26,155,350	1.3G	979.1Mb	2019-12-21

Controls

Run	# of Spots	# of Bases	Size	Published
SRR6967849	26,707,041	1.3G	996.8Mb	2019-12-21
Run	# of Spots	# of Bases	Size	Published
SRR6967848	28,360,860	1.4G	1Gb	2019-12-21

Figure1. Sequencing data of samples (Ey-anti antibody libraries) and controls

(No antibody libraries) .Nih.gov, 2024

Command lines & Script

1. I used **mkdir** command to make a directory for data analyzing Chip-seq and printed the list of directories and files by **ls** command.

```
(base) saeid@DT004E01FD51D8:~$ mkdir eyCHip-seq  
  
(base) saeid@DT004E01FD51D8:~$ ls  
  
eyCHip-seq
```

2. I used **conda** command, a package manager that supports bioinformatics tools to install sratools. This tool provides access to NCBI database for raw data of Sequence Read Archive (SRA) and allows to processing of NGS projects' data analysis. I created a new environment with **create** command and I named it "sratools" by **-n** flag and installed 3rd version of sartoolkit.

```
(base) saeid@DT004E01FD51D8:~/eyCHip-seq$ conda create -n sratools sra-  
tools=3
```

3. I changed the directory to eyCHip-seq by **cd** command and activated sratools environment by **activate** command. I would use this environment to get the raw data of my controls' and samples' reads of the NCBI database.

```
(base) saeid@DT004E01FD51D8:~/eyCHip-seq$ conda activate sratools
```

Getting the data

4. For getting the data, I used **prefetch** command that lets to download raw reads from the NCBI database. I added all of the accession numbers that I needed to download. Adding **-v** option lets to get more detail about the process and it is verbose mood.

```
(sratools) saeid@DT004E01FD51D8:~/eyCHip-seq$ prefetch SRR6967846  
SRR6967847 SRR6967849 SRR6967848  
2024-12-10T18:55:57 prefetch.3.1.1: 1) Resolving 'SRR6967846'..  
2024-12-10T18:55:58 prefetch.3.1.1: Current preference is set to retrieve SRA  
Normalized Format files with full base quality scores  
2024-12-10T18:55:59 prefetch.3.1.1: 1) Downloading 'SRR6967846'..  
2024-12-10T18:55:59 prefetch.3.1.1: SRA Normalized Format file is being retrieved  
...
```

5. I printed the list of files and directories in the eyCHip-seq directory. I used **-lh** flag which gives a human-readable and long format for every file's and directory's information such as file permissions for the owner, group, or others (r: read, w: write, x: Execute), user name, size of file or directory and last modified time. In the below example, every directory contains an SRA file.

```
(sratools) saeid@DT004E01FD51D8:~/eyCHip-seq$ ls -lh
total 16K
drwxr-xr-x 2 saeid saeid 4.0K Dec 10 18:56 SRR6967846
drwxr-xr-x 2 saeid saeid 4.0K Dec 10 18:57 SRR6967847
drwxr-xr-x 2 saeid saeid 4.0K Dec 10 18:58 SRR6967848
drwxr-xr-x 2 saeid saeid 4.0K Dec 10 18:58 SRR6967849
(sratools) saeid@DT004E01FD51D8:~/eyCHip-seq$ ls -lh SRR6967846
total 980M
-rw-r--r-- 1 saeid saeid 980M Dec 10 18:56 SRR6967846.sra
```

6. The **vdb-validate** is a tool in the SRA toolkits that confirms the integrity of data downloaded from the NCBI database and checks the data is completed, uncorrupted, and correct format. I utilized this tool to validate my data.

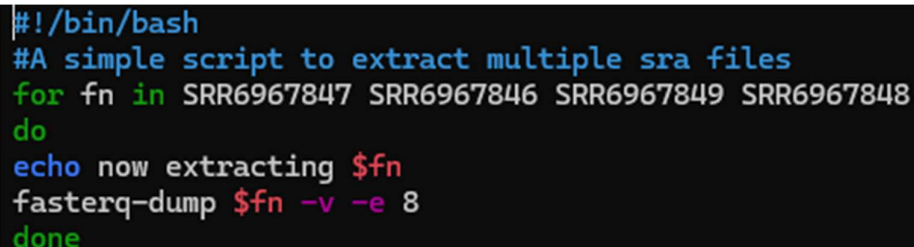
```
(sratools) saeid@DT004E01FD51D8:~/eyCHip-seq$ vdb-validate SRR6967846
SRR6967847 SRR6967849 SRR6967848
2024-12-10T21:12:36 vdb-validate.3.1.1 info: Table 'SRR6967846.sra' metadata: md5
ok
2024-12-10T21:12:36 vdb-validate.3.1.1 info: Column 'ALTREAD': checksums ok
2024-12-10T21:12:36 vdb-validate.3.1.1 info: Column 'QUALITY': checksums ok
2024-12-10T21:12:37 vdb-validate.3.1.1 info: Column 'READ': checksums ok
...
```

7. For extracting the data, I used the nano tool that enabled me to extract my SRAs data with one script that increased the speed of data analysis. The nano tool is a simple, easy-to-use text editor for the command line. I used **nano** command and I put the name of the script that I chose fastq-ey.

```
(sratools) saeid@DT004E01FD51D8:~/eyCHip-seq$ nano fastq-ey
```

8. The **nano** command opened a window that enabled me I write my script. **#!/bin/bash** specifies that the script can be performed by the bash command. The second line is a comment that helps users and is an explanation of the script. Everything after the **#** sign is ignored by the system. The (**for** variable **in** value1 value2 ...) line makes a loop that allows performing a Command for multiple files or directories. The loop starts with **do** and finishes with **done**. The **echo** line that's part of the script will print to the terminal and because of **\$fn** at the end of the line will repeat every time the loop starts. In my case, I used **fastq-dump** which performed for every accession number. This tool is another SRA toolkit command that can convert SRA files to FASTQ format. The **-v** flag is a verbose option that prints more details about files. Also **-e 8** flag which means asks the system to use 8 cores of CPU that increase the speed of the process. By changing the number, the number of usage cores can be changeable.

```
#!/bin/bash
#A simple script to extract multiple sra files
for fn in SRR6967847 SRR6967846 SRR6967849 SRR6967848
do
echo now extracting $fn
fasterq-dump $fn -v -e 8
done
```



```
#!/bin/bash
#A simple script to extract multiple sra files
for fn in SRR6967847 SRR6967846 SRR6967849 SRR6967848
do
echo now extracting $fn
fasterq-dump $fn -v -e 8
done
```

9. I used **bash** command for performing the script

```
(sratools) saeid@DT004E01FD51D8:~/eyCHIP-seq$ bash fastq-ey
now extracting SRR6967847
Preference setting is: Prefer SRA Normalized Format files with full base quality
scores if available.
SRR6967847 is an SRA Normalized Format file with full base quality scores.
spots read   : 19,093,976
reads read   : 19,093,976
reads written : 19,093,976
...
```

10. I performed **ls -lh** command to see new extracting files and the whole data size which was 21 gigabytes.

```
(sratools) saeid@DT004E01FD51D8:~/eyCHIP-seq$ ls -lh
total 21G
drwxr-xr-x 2 saeid saeid 4.0K Dec 10 18:56 SRR6967846
-rw-r--r-- 1 saeid saeid 5.4G Dec 10 21:32 SRR6967846.fastq
drwxr-xr-x 2 saeid saeid 4.0K Dec 10 18:57 SRR6967847
-rw-r--r-- 1 saeid saeid 4.0G Dec 10 21:31 SRR6967847.fastq
drwxr-xr-x 2 saeid saeid 4.0K Dec 10 18:58 SRR6967848
-rw-r--r-- 1 saeid saeid 5.9G Dec 10 21:32 SRR6967848.fastq
drwxr-xr-x 2 saeid saeid 4.0K Dec 10 18:58 SRR6967849
-rw-r--r-- 1 saeid saeid 5.6G Dec 10 21:32 SRR6967849.fastq
-rw-r--r-- 1 saeid saeid 170 Dec 10 21:31 fastq-ey
```

Merging Replicate libraries and Compressing

11. This study has replicated experiments. For declining analysis time, I merged two samples and two controls so got a sample file and a control file. To do that, I used the **cat** command. The **cat** is a tool that has various functions related to context. By **cat >>**, enable to appending a context to an existing file. I applied this command and I added one FASTQ file to another. After merging, I used **ls -lh** command to ensure that I merged it successfully. As you can see, the data size of the files I added increased, which means merging succeeded.

```
(sratools) saeid@DT004E01FD51D8:~/eyCHIP-seq$ cat SRR6967846.fastq >>
SRR6967847.fastq
(sratools) saeid@DT004E01FD51D8:~/eyCHIP-seq$ cat SRR6967848.fastq >>
SRR6967849.fastq
(sratools) saeid@DT004E01FD51D8:~/eyCHIP-seq$ ls -lh
total 32G
drwxr-xr-x 2 saeid saeid 4.0K Dec 10 18:56 SRR6967846
-rw-r--r-- 1 saeid saeid 5.4G Dec 10 21:32 SRR6967846.fastq
drwxr-xr-x 2 saeid saeid 4.0K Dec 10 18:57 SRR6967847
-rw-r--r-- 1 saeid saeid 9.4G Dec 11 00:00 SRR6967847.fastq
drwxr-xr-x 2 saeid saeid 4.0K Dec 10 18:58 SRR6967848
-rw-r--r-- 1 saeid saeid 5.9G Dec 10 21:32 SRR6967848.fastq
drwxr-xr-x 2 saeid saeid 4.0K Dec 10 18:58 SRR6967849
-rw-r--r-- 1 saeid saeid 12G Dec 11 00:02 SRR6967849.fastq
-rw-r--r-- 1 saeid saeid 170 Dec 10 21:31 fastq-ey
```

12. I operated the **rm** command to remove extra files. Again, I used **ls -lh** to check for correct removal.

```
(sratools) saeid@DT004E01FD51D8:~/eyCHip-seq$ rm SRR6967846.fastq
(sratools) saeid@DT004E01FD51D8:~/eyCHip-seq$ rm SRR6967848.fastq
(sratools) saeid@DT004E01FD51D8:~/eyCHip-seq$ ls -lh
total 21G
drwxr-xr-x 2 saeid saeid 4.0K Dec 10 18:56 SRR6967846
drwxr-xr-x 2 saeid saeid 4.0K Dec 10 18:57 SRR6967847
-rw-r--r-- 1 saeid saeid 9.4G Dec 11 00:00 SRR6967847.fastq
drwxr-xr-x 2 saeid saeid 4.0K Dec 10 18:58 SRR6967848
drwxr-xr-x 2 saeid saeid 4.0K Dec 10 18:58 SRR6967849
-rw-r--r-- 1 saeid saeid 12G Dec 11 00:02 SRR6967849.fastq
-rw-r--r-- 1 saeid saeid 170 Dec 10 21:31 fastq-ey
```

13. Because of the enormous size of the files, I turned files into zip files. I utilized a tool called **pigz**. This tool used multiple cores which is faster than **gzip**. I already had this tool but should be installed with the **sudo apt install** command. The purpose of this command is to save space and simplify data transfer. After that, I used **ls -lh** to check the size of the directory.

```
(sratools) saeid@DT004E01FD51D8:~/eyCHip-seq$ pigz *fastq
SRR6967847.fastq to SRR6967847.fastq.gz

SRR6967849.fastq to SRR6967849.fastq.gz
(sratools) saeid@DT004E01FD51D8:~/eyCHip-seq$ ls -lh
total 5.7G
drwxr-xr-x 2 saeid saeid 4.0K Dec 10 18:56 SRR6967846
drwxr-xr-x 2 saeid saeid 4.0K Dec 10 18:57 SRR6967847
-rw-r--r-- 1 saeid saeid 2.6G Dec 11 00:00 SRR6967847.fastq.gz
drwxr-xr-x 2 saeid saeid 4.0K Dec 10 18:58 SRR6967848
drwxr-xr-x 2 saeid saeid 4.0K Dec 10 18:58 SRR6967849
-rw-r--r-- 1 saeid saeid 3.1G Dec 11 00:02 SRR6967849.fastq.gz
-rw-r--r-- 1 saeid saeid 170 Dec 10 21:31 fastq-ey
```


Evaluating read quality and trimming

14. I applied another script for trimming both files at once. I wrote another script in **nano**. I named fastp.ey and performed it by **bash** command. I made a new loop and exploited **fastp** tool for trimming. The **fastp** is a bioinformatics tool that is used for FASTQ file processing. The **fastp** should be set up accurately that get the best result. The **-i** and **-I** flags are used for choosing input files. The **-o** and **-O** flags are used for output files. In the script, I added an F letter to differentiate trimming files from fastq files. The **--qualified_quality_phred** option defines a threshold for every base that is considered qualified. I chose 15 which means phred more than 15 is a high-quality base. The **--unqualified_percent_limit** option deletes reads that have unqualified bases more than setting up percentage. I chose 40% which means reads with more than 40% unqualified base trimmed. The **--length_required** flag cutes short reads that can be aligned with various parts of the reference genome. I set up 36 because my average read length is 50 which is the perfect number for trimming. The **--cut_front** and **--cut_tail** flags allow the tool to cut read from 3'end (tail) and 5'end (front). The **--cut_window_size** Specifies the window size used for quality cutting. The higher number increases the trimming speed but decreases the quality of that. The **--cut_mean_quality** option cuts reads when the average quality of a window is below than arranged number. The **--thread** flag assigns the CPU core numbers that are used for trimming and the **--html** flag generates the trimming report in a html file. The **--json** option produces the trimming report in a json format.

```
(sratools) saeid@DT004E01FD51D8:~/eyCHip-seq$ nano fastp.ey
```

```
#!/bin/bash
#A simple script to trim multiple read files
for fn in SRR6967849 SRR6967847
do
echo now trimming $fn
fastp -i ${fn}.fastq.gz -o f${fn}.fastq.gz \
--qualified_quality_phred 15 \
--unqualified_percent_limit 40 \
--length_required 36 \
--cut_front \
--cut_tail \
--cut_window_size 4 \
--cut_mean_quality 20 \
--thread 8 \
--html ${fn}.html\
--json ${fn}.json
Done
```

```
#!/bin/bash
#A simple script to trim multiple read files
for fn in SRR6967849 SRR6967847
do
echo now trimming $fn
fastp -i ${fn}.fastq.gz -o f${fn}.fastq.gz \
--qualified_quality_phred 15 \
--unqualified_percent_limit 40 \
--length_required 36 \
--cut_front \
--cut_tail \
--cut_window_size 4 \
--cut_mean_quality 20 \
--thread 8 \
--html ${fn}.html \
--json ${fn}.json
done
```

15. I employed **cp** command to copy html files in my documents on the C drive of my PC. I used ***html** to choose every html file in the directory and I addressed the file location at the end.

```
(sratools) saeid@DT004E01FD51D8:~/eyCHip-seq$ cp *.html
/mnt/c/Users/19336487/Documents/
```

Aligning reads to the genome

16. For aligning read, I needed to download the *Drosophila melanogaster* genome sequences in FASTQ format. I found the proper download link in ensemble.org and by doing **wget** command I got the data in the directory. In the next step, I extract the zip file with the **gunzip** tool.

```
(sratools) saeid@DT004E01FD51D8:~/eyCHip-seq$ wget
https://ftp.ensembl.org/pub/release-
113/fasta/drosophila_melanogaster/dna/Drosophila_melanogaster.BDGP6.46.dna.topl
evel.fa.gz

--2024-12-11 03:49:33-- https://ftp.ensembl.org/pub/release-
113/fasta/drosophila_melanogaster/dna/Drosophila_melanogaster.BDGP6.46.dna.toplevel.
fa.gz
Resolving ftp.ensembl.org (ftp.ensembl.org)... 193.62.193.169
...
(sratools) saeid@DT004E01FD51D8:~/eyCHip-seq$ gunzip
Drosophila_melanogaster.BDGP6.46.dna.toplevel.fa.gz
(sratools) saeid@DT004E01FD51D8:~/eyCHip-seq$ ls -lh
total 12G
-rw-r--r-- 1 saeid saeid 140M Aug 14 22:42
Drosophila_melanogaster.BDGP6.46.dna.toplevel.fa
drwxr-xr-x 2 saeid saeid 4.0K Dec 10 18:56 SRR6967846
drwxr-xr-x 2 saeid saeid 4.0K Dec 10 18:57 SRR6967847
...
```

16. The **HISAT2** is one of the read aligners tool that performs NGS alignment. This tool is widely used in bioinformatics. I already had it but for getting it can be used **conda install -c bioconda hisat2** command line. The **hisat2-build** command provides a genome index which is necessary for mapping sequencing reads. After command, I added the reference genome file and then I chose a name that is a prefix for output index files. The **-p 8** flag is for assigning the number of PC cores for processing.

```
(sratools) saeid@DT004E01FD51D8:~/eyCHip-seq$ hisat2-build
Drosophila_melanogaster.BDGP6.46.dna.toplevel.fa Dm6 -p 8
Settings:
  Output files: "Dm6.*.ht2"
  Line rate: 6 (line is 64 bytes)
  Lines per side: 1 (side is 64 bytes)
  Offset rate: 4 (one in 16)
...
```

17. The **hisat2** command runs the alignment process by the tool. The **-x** flag specifies the reference genome index and **-U** flag defines the trimmed FASTQ reads the file. For getting a summarized report of alignment in text format, enable to use **--summary-file** option. The **-p** is for choosing CPU cores numbers and the **>** sign redirects the output to the sam file. I operate this alignment for both sample and control files.

```
(sratools) saeid@DT004E01FD51D8:~/eyCHip-seq$ hisat2 -x Dm6 -U
fSRR6967849.fastq.gz --summary-file SRR6967849.txt -p 8 >SRR6967849.sam
51544847 reads; of these:
  51544847 (100.00%) were unpaired; of these:
    20733469 (40.22%) aligned 0 times
    27131355 (52.64%) aligned exactly 1 time
    3680023 (7.14%) aligned >1 times
59.78% overall alignment rate
(sratools) saeid@DT004E01FD51D8:~/eyCHip-seq$ hisat2 -x Dm6 -U
fSRR6967847.fastq.gz --summary-file SRR6967847.txt -p 8 >SRR6967847.sam
42454655 reads; of these:
  42454655 (100.00%) were unpaired; of these:
    9381955 (22.10%) aligned 0 times
    30036874 (70.75%) aligned exactly 1 time
    3035826 (7.15%) aligned >1 times
77.90% overall alignment rate
```

Converting to SAM format

18. I converted my **SAM** files (Sequence Alignment/Map) to **BAM** files (Binary Alignment/Map). The BAM files are not human readable but have less size, are easy and fast to process by the system, and support indexing which we need for quick access to specific regions for visualization or downstream analysis. I used **SAMtools** which is a powerful tool for converting **BAM** format to **SAM** and sorting and indexing **BAM** files. I operate the **samtools view** command for converting. In Addition, the **-@** flag sets the number of threads and the **-b** flag indicates that the output is a **BAM** file. Also, **>** option lets to name the output file. Notice that the intended input file is set up before **>** option.

```
(sratools) saeid@DT004E01FD51D8:~/eyCHip-seq$ samtools view -@ 8 -b  
SRR6967847.sam > SRR6967847.bam  
(sratools) saeid@DT004E01FD51D8:~/eyCHip-seq$ samtools view -@ 8 -b  
SRR6967849.sam > SRR6967849.bam
```

19. I needed to see the **BAM** file which I know is sorted or unsorted. The **BAM** files can not be readable by humans. The **samtools view** command can print a **BAM** file in **SAM** format. I operated this command to open my **BAM** file and I added **-H** flag that gives us the header of the file. Also, I used the **|** sign for performing multiple commands in one line. I utilized the **head** command for printing the first ten lines of the file. As you see below, now the file header is readable and contains the version of **SAM** format (1.0) and information about the file whether sorted by genomic coordinates or not (unsorted). Also, there is information on chromosomes' names and lengths.

```
(sratools) saeid@DT004E01FD51D8:~/eyCHip-seq$ samtools view -H  
SRR6967849.bam | head  
@HD VN:1.0 SO:unsorted  
@SQ SN:2L LN:23513712  
@SQ SN:2R LN:25286936  
...
```

20. I applied the **samtools sort** command for sorting my **SAM** files. After the command, it should be the intended **SAM** file, and by **-O** you can choose a name for the output file.

```
(sratools) saeid@DT004E01FD51D8:~/eyCHip-seq$ samtools sort  
SRR6967847.bam -o SRR6967847-sorted.bam -@ 8  
(sratools) saeid@DT004E01FD51D8:~/eyCHip-seq$ samtools sort  
SRR6967849.bam -o SRR6967849-sorted.bam -@ 8
```

21. I used the **samtools index** which organizes and indexes the information to provide quick access to various parts of the genome. This command will generate an output-name.bai.

```
(sratools) saeid@DT004E01FD51D8:~/eyCHip-seq$ samtools index SRR6967849-  
sorted.bam -@ 8  
(sratools) saeid@DT004E01FD51D8:~/eyCHip-seq$ samtools index SRR6967847-  
sorted.bam -@ 8  
(sratools) saeid@DT004E01FD51D8:~/eyCHip-seq$ ls -lh *bai  
-rw-r--r-- 1 saeid saeid 588K Dec 11 17:41 SRR6967847-sorted.bam.bai  
-rw-r--r-- 1 saeid saeid 595K Dec 11 17:41 SRR6967849-sorted.bam.bai
```

Peak Calling

22. The **MACS2** is a widely used software to identify peaks in ChIP-Seq data. The peaks demonstrate the reasons that in interest protein binds to the genome. For installing and activating MACS2, needs to deactivate the current environment and create a new environment for the **MACS2**. I made an environment by **create** command and named it by the **-n** flag. Ultimately, I installed MACS2 on the mac2 environment, and in another line, I activated the environment.

```
(sratools) saeid@DT004E01FD51D8:~/eyCHip-seq$ conda deactivate
(base) saeid@DT004E01FD51D8:~/eyCHip-seq$ conda create -n macs2 macs2
Retrieving notices: ...working... done
Channels:
- bioconda
- conda-forge
- defaults
...
(base) saeid@DT004E01FD51D8:~/eyCHip-seq$ conda activate macs2
(mac2) saeid@DT004E01FD51D8:~/eyCHip-seq$
```

23. For peak calling, I operated the **macs2 callpeak** command and set it up with flags. The **-t** flag indicates the sample or treatment **BAM** file, the **-c** flag assigns the control BAM file, the **-g** flag indicates the genome size of the studied species, *Drosophila melanogaster* (dm), the **-n** flag sets the prefix for output files (ey) and **--nomodel** option disables estimating peaks and building a model by **MACS2**.

```
(macs2) saeid@DT004E01FD51D8:~/eyCHip-seq$ macs2 callpeak -t
SRR6967847.bam -c SRR6967849.bam -g dm -n ey --nomodel
ey --nomodel
INFO @ Wed, 11 Dec 2024 18:13:26:
# Command line: callpeak -t SRR6967847.bam -c SRR6967849.bam -g dm -n ey --
nomodel
# ARGUMENTS LIST:
...
```

Visualization

24. Finally I copied all of the sorted bam and bai files and also my peak file was generated with the **MACS2** in the Documents folder in the C drive. I used one command for copying my bam and bai files which all of the names contain **sorted.bam** .Also for copying callpeak files, I used ey-prefix that I called in the prior command line.

```
(macs2) saeid@DT004E01FD51D8:~/eyCHip-seq$ cp *sorted.bam*  
/mnt/c/Users/19336487/Documents/  
(macs2) saeid@DT004E01FD51D8:~/eyCHip-seq$ cp *ey_*  
/mnt/c/Users/19336487/Documents/
```

25. **IGV** (Integrative Genomics Viewer) is a user-friendly visualization for exploring genomic data. It allows researchers to study various genomic data types including **BAM** files. I installed **IGV**, uploaded BAM files and callpeak files on **IGV**, and set up the tool to explore on intended species (dm6/ *Drosophila melanogaster*).

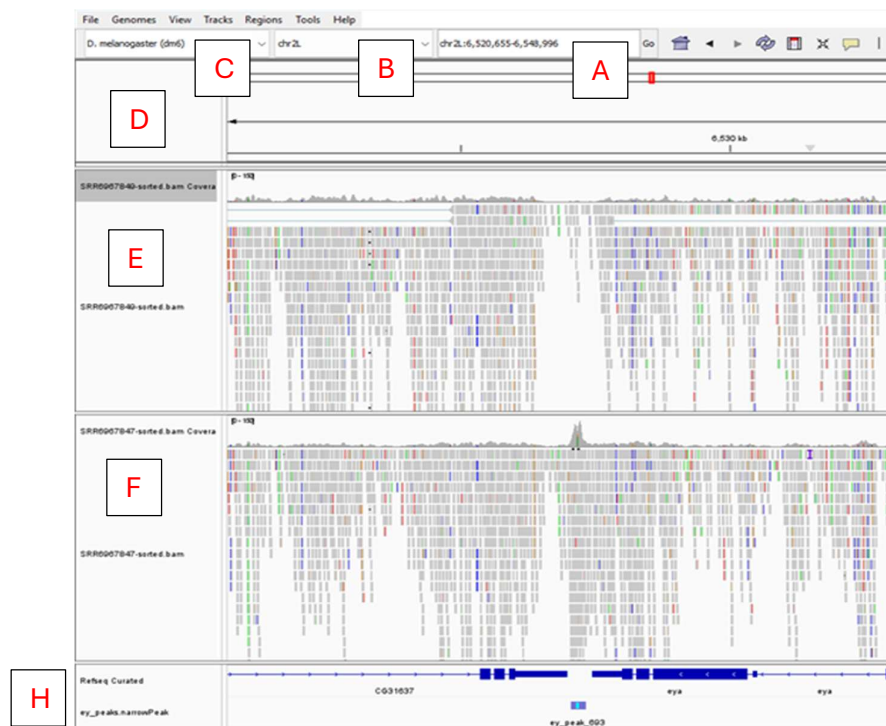


Figure2. "Annotated Overview of Genomic Data Visualization in IGV"

- A. Search bar that allows to find specific regions of the genome
- B. Search and display bar for chromosome
- C. Reference genome track
- E. Control alignment track
- F. Sample alignment track
- H. Gene annotation track which displays genes', exons', and introns' location. In Addition, It displays the peaks generated by MACS2.

Results and Discussion

Libraries' quality and Trimming

The **fastp** tool generated a summary report of trimming in html file format. Based on **fastp** reports I got this information:

Samples

The type of libraries is single-end forward reads with a 50bp average length. I merged two replicated libraries. As a result, the total spots' number is 45.25M. Because of merging, the duplication rates increased (19.75%). After trimming, the mean length of libraries declined to 48bp. The number of cutting-out reads is 2.8M(6.1)% and there are 42.45M(93.8%) reads passed filtering.

Controls

Before trimming, the mean average length of the controls' libraries is 50bp and reduced to 48bp after trimming the same as the samples' reads. The total reads of merged controls' reads is 55.06M 3.5M reads (6.4%) filtered after the process. The type of libraries is single-end sequencing.

The NGS experiments based on trimming information have enough quality for exploration. The results of the controls and samples' libraries are similar together which shows the low probability of fault preparation libraries and lab works. Many reads are filtered because they are too short. There are only 69 and 56 unqualified reads in the controls and samples' libraries respectively. The used setting for discarding bases was a Phred score below 15 and with > 40 % low-quality bases. The quality of the reads is high enough. Increasing the number of qualified qualities Phred and unqualified percent limit was suggested for getting better trimming results. The average read length is 48bp after trimming which is long enough to avoid the wrong alignment. The number of amplification cycles(50 cycles) gave big enough spots to operate a reliable alignment and peaks. Moreover, the trimming process does not affect GC content (Before trimming GC content: 44.4% After trimming GC content 43.5%) which is important for ChIP-Seq alignment. The genes are mainly found in GC-rich regions(Chauhan, 2021).

General	
fastp version:	0.20.0 (https://github.com/OpenGene/fastp)
sequencing:	single end (50 cycles)
mean length before filtering:	50bp
mean length after filtering:	48bp
duplication rate:	19.754819% (may be overestimated since this is SE data)
Detected readl adapter:	GATCGGAAGAGCACACGTCTGAACTCCAGTCAC
Before filtering	
total reads:	45.249326 M
total bases:	2.262466 G
Q20 bases:	2.093643 G (92.538074%)
Q30 bases:	1.909899 G (84.416699%)
GC content:	44.385629%
After filtering	
total reads:	42.454655 M
total bases:	2.070826 G
Q20 bases:	1.995493 G (96.362172%)
Q30 bases:	1.828277 G (88.287347%)
GC content:	43.506078%
Filtering result	
reads passed filters:	42.454655 M (93.823839%)
reads with low quality:	56 (0.000124%)
reads with too many N:	0 (0.000000%)
reads too short:	2.794615 M (6.176037%)

Figure3. “Summary report of trimming for samples’ reads”

General	
fastp version:	0.20.0 (https://github.com/OpenGene/fastp)
sequencing:	single end (50 cycles)
mean length before filtering:	50bp
mean length after filtering:	48bp
duplication rate:	33.190432% (may be overestimated since this is SE data)
Detected readl adapter:	GATCGGAAGAGCACACGTCTGAACTCCAGTCAC
Before filtering	
total reads:	55.067901 M
total bases:	2.753395 G
Q20 bases:	2.550116 G (92.617132%)
Q30 bases:	2.329093 G (84.589843%)
GC content:	44.532705%
After filtering	
total reads:	51.544847 M
total bases:	2.515929 G
Q20 bases:	2.423615 G (96.330811%)
Q30 bases:	2.223447 G (88.374781%)
GC content:	43.685764%
Filtering result	
reads passed filters:	51.544847 M (93.602346%)
reads with low quality:	69 (0.000125%)
reads with too many N:	0 (0.000000%)
reads too short:	3.522985 M (6.397529%)

Figure4. “Summary report of trimming for controls’ reads”

Description of FASTQ File Format

The below command line opens the first ten lines of the FASTQ file. I chose one read information for more explanation of fields and Phred score.

```
(sratools) saeid@DT004E01FD51D8:~/eyCHip-seq$ head SRR6967847.fastq
```

```
@SRR6967847.1 HWI-ST601:7:1101:1939:2188 length=50
TTTCTAGGCTTTCCTTTGATTAAACATCATCTGATGTATAATCGTATATA
+SRR6967847.1 HWI-ST601:7:1101:1939:2188 length=50
@@@DBDDDDDHAFFGH@BC,<:4?+<A+99<FAF+**C@FE*):9B*?
```

- +SRR6967847.1: indicates the identification number for every read.
- 1 HWI-ST601:7:1101:1939:2188 : includes the information of sequencing runs and technical details and a unique identifier for sequencing instrument (Illumina) which allows tracking read for Quality control.
- length=50: contains reads length which is 50bp
- The second line contains the genetic information of every read nucleotide base, In my case, this line contains 50 nucleotides.
- The third line is repeated information the same as the first line.
- The fourth line includes the ASCII signs for every nucleotide of read that indicates the Phred score or Quality score and estimated probability of an error.

For example, the ASCII sign in the first nucleotide of the read that has a T base is @ sign. The probability of an error of this ASCII sign is 0.00079 and the quality score is 31. The higher quality score is equal with less probability of an error.

Description of SAM File Format

The below command line opens the first 50 lines of the FASTQ file. I chose one read information for more explanation of SAM file fields:

```
(base) saeid@DT004E01FD51D8:~/eyCHip-seq$ cat SRR6967847.sam | head -n 50
```

```
SRR6967847.85618    16    2R    23898027    60    50M    *    0    0
ATCATATTGTTTCGGGATTTCGATTCCCTTCCGAGTGTCTATGTCCTCC
9F;B=CB9<HEFGGFFHFHFCB:*)C8)GFDBGGIGIIHBDFFDD?FF@C@    AS:i:-4 XN:i:0
XM:i:2 XO:i:0 XG:i:0 NM:i:2 MD:Z:23G2A23 YT:Z:UU NH:i:1
```

QNAME	SRR6967847.85618
FLAG	16
RNAME	2R
POS	23898027
MAPQ	60
CIGAR	50M
RNEXT	*
PNEXT	0
TLEN	0

- **QNAME** (Query name): is the identifier number for the read or sequence. The first number is the accession number of the sample, and the second number mentions the unique number of reads.
- **FLAG** (Alignment Flag): is a numeric value that contains information on alignment such as whether it is paired, and properly aligned. This is binary coding which means should turn the number to binary language and check the table of FLAG. In my case, 16 means this read is a reverse strand read.

- **RNAME** (Reference Name): includes the name of the reference sequence (chromosome). 2R means the right arm of the second chromosome of *Drosophila*. This species has 4 paired chromosomes.
- **POS** (position): is the number of the first base of read that starts in the reference sequence.
- **MAPQ** (Mapping Quality): is a number gives a confidence score for alignment. Higher scores demonstrate more quality. The 60 score is high enough quality, and it is reliable confidently.
- **CIGAR** (Alignment Pattern): is a sign or letter to indicate an alignment pattern. For instance, **M** for matches, **I** for insertions, and **D** for deletions. 50M means all 50 nucleotides are matched.
- **RNEXT** (References of next read): is a reference name for the paired read. The * sign means no information is available because this experiment is single-end sequencing.
- **PNEXT** (Position of next read): is a number of paired read's starting positions. The 0 means this is a single-end alignment.
- **TLEN** (template length): is another piece of information for paired-end sequencing that indicates the extent between paired reads.
- **SEQ** (sequence): it is printed in the second line and contains the actual DNA sequence of read.
- **QUAL** (Quality Scores): it is printed in the third line and represents every base quality score in ASCII-encoded format.

Data Analysis Pipelines comparison

For data analysis, I used the same samples and controls that I got from the GEO database.

- The study and my reads libraries are identically the same. I exploited the SRA toolkit for downloading. I validated the row data by vdb-validate. The SRA files were converted to FASTQ format with fastq-dump. The study data preparation is not explicitly described. The analysis can have different methods for preparing data.
- I used the fastp tool for trimming and quality checks which is important. Without a doubt, different trimming tools and setup details affect the outcomes. This point is one of the reasons for the probable differences in analysis results.
- I ran the alignment with HISAT2 tool and my reference genome is dm6. On the other hand, the paper analysis aligned reads with dm3, the older version of the genome, using BWA-MEM. The BWA-MEM is a popular alignment tool that is more accurate and has lower memory usage but is slower than HISAT2.
- The peak identification was done by MACS in the study that is the old version. In comparison, I used MACS2 for calling peaks. The MACS is no longer updated and has less flexibility and accuracy than MACS2.
- For visualization, both analyses utilized IGV. The only difference is that I used a new version of the genome (dm6) for mapping.

Both pipelines assigned close methodology but the analysis that I have done includes some modern tools and genome updates. Also paper has additional experiments and analysis on histone modification that can offer better genomic perspective.

Visualization Comparison

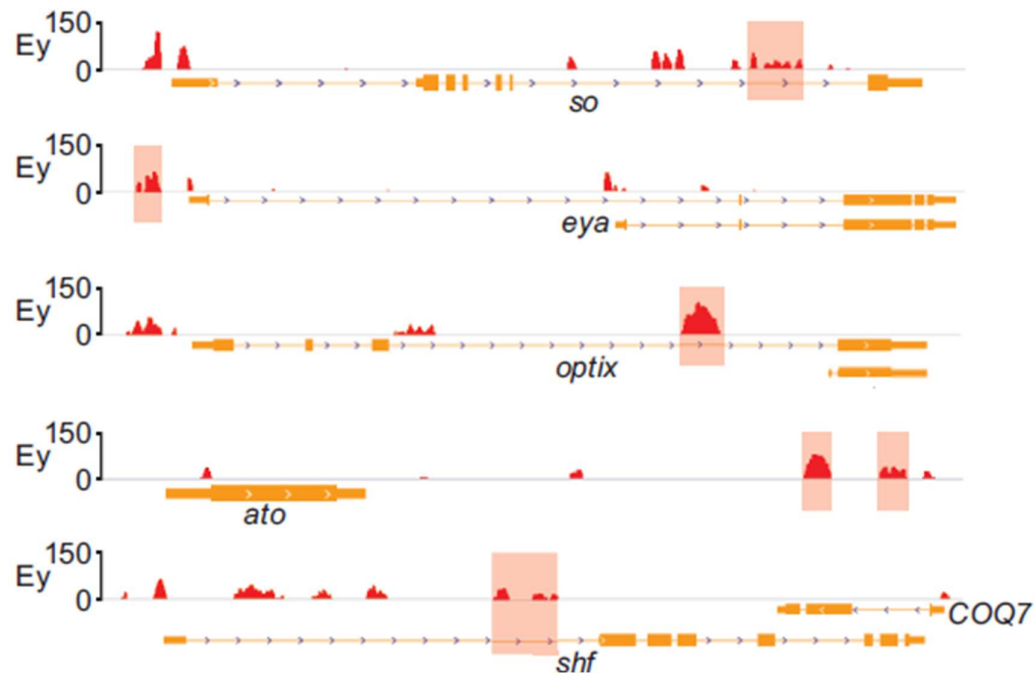


Figure 5. *Ey* Chip-seq peaks (Yeung and colleagues' data analysis) demonstrate *Ey* protein binds to prior known *ey* binding sites that show the data analysis accuracy. The red-marked regions are known enhancers that *ey* bind to it. Yeung et al, 2018

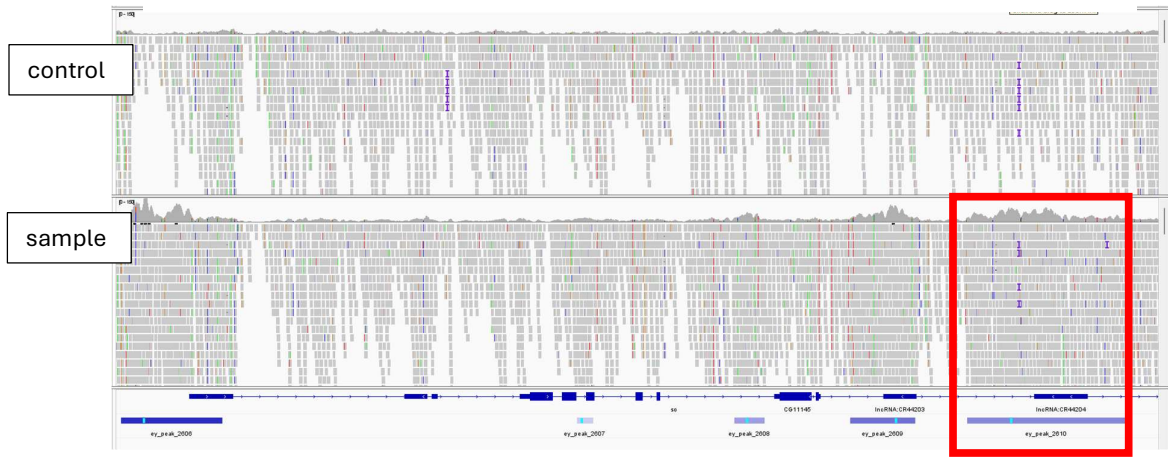


Figure6. *Ey* Chip visualization that shows *Ey* binding known enhancer located in a part *so* gene. The enhancer is highlighted with red color.

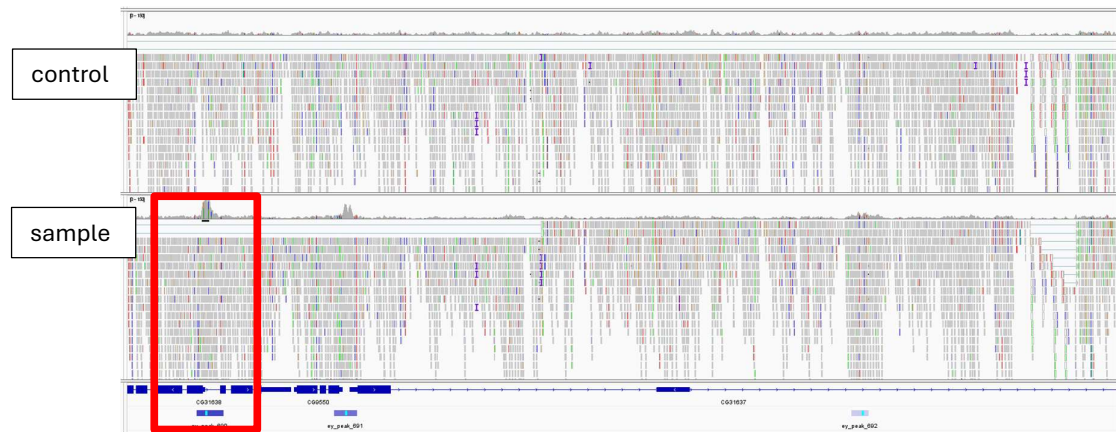


Figure7.Ey Chip visualization that shows Ey binding known enhancer located near to **eya** and **CG31637** gene. The enhancer is highlighted with red color.

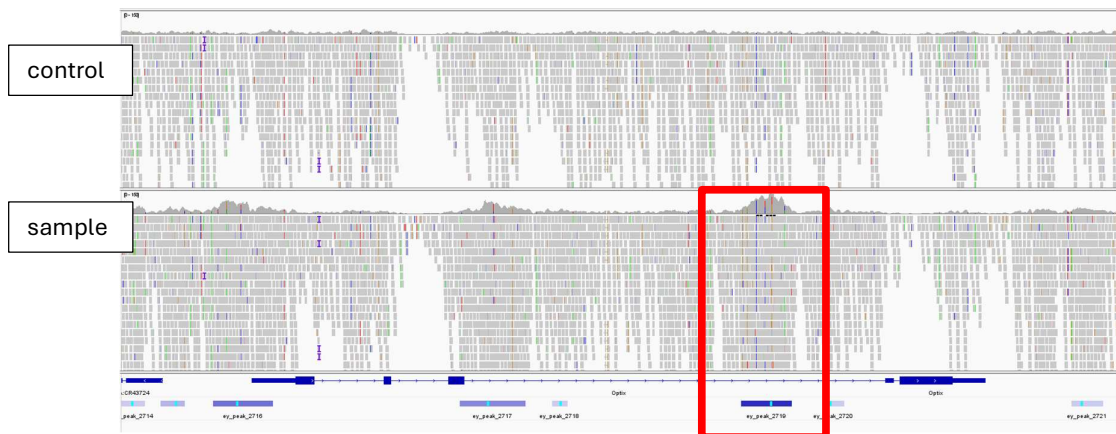


Figure8.Ey Chip visualization that shows Ey binding known enhancer located in **optix** gene. The enhancer is highlighted with red color.

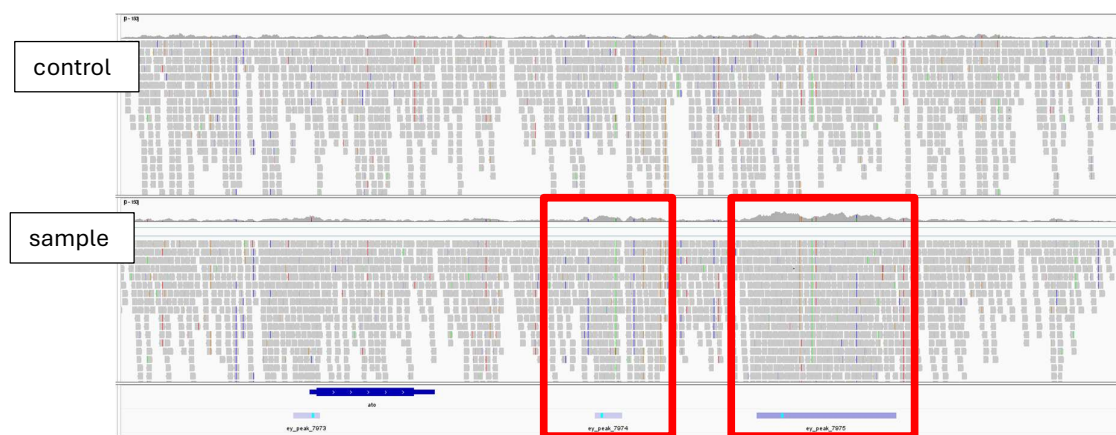


Figure9.Ey Chip visualization that shows Ey binding known enhancer located in **ato** gene. The enhancer is highlighted with red color.

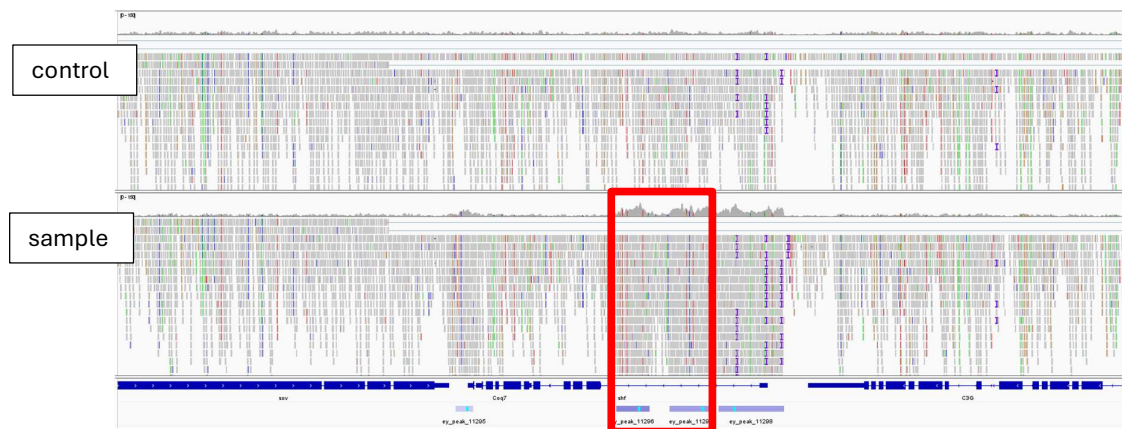


Figure10.Ey ChIP visualization that shows Ey binding known enhancer located in *shf* gene. The enhancer is highlighted with red color.

Figure 5 is a published visualization in the study that I chose for comparison with my analysis. In this image, the data indicates enhancers' binding sites that interact with the Ey transcription factor. The figure confirms previously found enhancer regions that are regulated by Ey for eye development. Despite the differences in the methodology of analysis, the generated visuals in the five gene locations that I examined, show the same peaks and binding sites. I used a 150 scale number the same as paper to have a better comparison.

References

- Chauhan, D.T. (2021). *What is the importance of GC Content?* –. [online] Geneticeducation.co.in. Available at: https://geneticeducation.co.in/what-is-the-importance-of-gc-content/?utm_content=cmp-true.
- Halder, G., Callaerts, P. and Gehring, W. (1995). Induction of ectopic eyes by targeted expression of the eyeless gene in *Drosophila*. *Science*, 267(5205), pp.1788–1792. doi:<https://doi.org/10.1126/science.7892602>.
- Jordan, T., Hanson, I., Zaletayev, D., Hodgson, S., Prosser, J., Seawright, A., Hastie, N. and van Heyningen, V. (1992). The human PAX6 gene is mutated in two patients with aniridia. *Nature Genetics*, 1(5), pp.328–332. doi:<https://doi.org/10.1038/ng0892-328>.
- Leleu, M., Lefebvre, G. and Rougemont, J. (2010). Processing and analyzing ChIP-seq data: from short reads to regulatory interactions. *Briefings in Functional Genomics*, 9(5-6), pp.466–476. doi:<https://doi.org/10.1093/bfgp/elq022>.
- Lindsley, D.L. and Zimm, G.G. (2012). *The Genome of Drosophila Melanogaster*. Academic Press.
- Yeung, K., Wang, F., Li, Y., Wang, K., Mardon, G. and Chen, R. (2018). Integrative genomic analysis reveals novel regulatory mechanisms of *eyeless* during *Drosophila* eye development. *Nucleic Acids Research*, [online] 46(22), pp.11743–11758. doi:<https://doi.org/10.1093/nar/gky892>