11-2023

Motif-Cluster: A Spatial Clustering Package for Repetitive Motif Binding Patterns

Mengyuan Zhou
University of Nebraska-Lincoln, mzhou10@huskers.unl.edu

MOTIF-CLUSTER: A SPATIAL CLUSTERING PACKAGE FOR REPETITIVE MOTIF BINDING PATTERNS

by

Mengyuan Zhou

A THESIS

Presented to the Faculty of
The Graduate College at the University of Nebraska
In Partial Fulfillment of Requirements
For the Degree of Master of Science

Major: Computer Science

Under the Supervision of Professor Qiuming Yao

Lincoln, Nebraska
November, 2023

MOTIF-CLUSTER: A SPATIAL CLUSTERING PACKAGE FOR REPETITIVE MOTIF BINDING PATTERNS

Mengyuan Zhou, M.S.

University of Nebraska, 2023

Adviser: Qiuming Yao

Previous efforts in using genome-wide analysis of transcription factor binding sites (TFBSs) have overlooked the importance of ranking potential significant regulatory regions, especially those with repetitive binding within a local region. Identifying these homogenous binding sites is critical because they have the potential to amplify the binding affinity and regulation activity of transcription factors, impacting gene expression and cellular functions. To address this issue, we developed an open-source tool Motif-Cluster that prioritizes and visualizes transcription factor regulatory regions by incorporating the idea of local motif clusters. Motif-Cluster can rank the significant transcription factor regulatory regions without the need for experimental data by applying a density-based clustering approach combined with flexible binding gaps and binding affinities.

Motif-Cluster uses an algorithm which effectively filters out the noise from weak binding sites by balancing region size and binding instances based on binding site gaps and binding affinities. As a result, the algorithm can effectively cluster local binding sites and identify crucial regulatory areas. The tool has

been tested under multiple strategies on local binding sites and has successfully recovered key regulatory regions for ZNF410 discovered previously for its binding clusters in the CHD4 promoter. It provides a useful interface to analyze densely packed binding sites and to visualize prioritized regulatory regions.

Overall, Motif-Cluster provides a more efficient and comprehensive solution to identifying significant transcription factory binding sites in genome-wide analyses than previous solutions. With improved efficiency and visualization capabilities, Motif-Cluster empowers researchers to gain new insights and design novel experiments through a new way of discovery.

DEDICATION

This thesis is a tribute to those who have stood by me during my journey in graduate school. I am profoundly thankful to my advisor, Dr. Qiuming Yao, whose guidance and inspiration have illuminated my academic path, offering invaluable advice and support in both research and personal pursuits. I'm greatly thankful to the other committee members, Dr. Juan Cui, and Dr. Qiang Liu, who offered dependable feedback on my thesis. My heartfelt thanks go to my lab mates, friends and colleagues for their constant life support and friendship. I extend my deepest gratitude towards my parents and family whose guidance has inspired me from the earliest days.

Contents

List of Figures

List of Tables

Chapter 1
Introduction

1.1 Basic Definitions

1.1.1 Introduction of Transcription Factor and Transcription Factor Binding Sites (TFBS)

Transcription factors [1]-[4] are proteins that play a key part in gene regulation by controlling transcription (the process of converting DNA into RNA) of specific genes.

Transcription factors can bind to specific DNA sequences on the genome called Transcription Factor Binding Sites (TFBSs) [5]-[8] for which they correspond to. When this binding takes place, a transcription factor can either activate or repress the expression of the associated gene depending on the context of the binding site and transcription factor itself.

TFBSs typically contain recognizable DNA sequences called motifs. These motifs are short, usually around 6 to 20 nucleotides long, and aid in the identification of binding sites. Identifying these binding sites is crucial for understanding gene regulation and deciphering the complex regulatory networks within cells.

1.1.2 Importance of Transcription Factor Binding Sites

Transcription factors regulate which genes are active in different cells of the body (gene expression) by binding to TFBSs. Recent research shows that transcription factors may search through vast expanses of the DNA sequence to identify these binding sites and that they can recognize a wide variety of DNA sequences with varying affinities [9]. Affinity refers to how strongly a transcription factor binds to its corresponding binding site, which therefore determines how strongly and specifically a transcription factor interacts with DNA to control the expression of genes.

Some local regions of DNA may contain a dense sequence of binding sites characterized by a repetitive nature but diverse affinities. These dense local binding regions play a major role in the process of transcription, which is the first step of transforming DNA sequences into functional proteins. Transcription factors regulate gene expression by controlling how genes are converted into RNA sequences.

1.1.3 Typical Way to Generate Genome-Wide Transcription Factor Binding Sites

There are extensive databases which collect and catalog motifs within the genome. These databases use reliable and reviewed experimentation and sources to document existing and newly discovered motif locations. JASPAR is one such motif database which tracks TFBSs collected with an internal ChIP-seq method as well as from external sources [10]. TRANSFAC is a highly curated collection of motifs and

MATCH Suite as a tool to identify potential binding sites [11]. For more focused applications, HOCOMOCO (Homo sapiens Comprehensive Model Collection) is a motif collection that focuses exclusively on binding sites for humans and mice [12].

In this research, we support data generation using PWM (.pwm) files and the FIMO (Find Individual Motif Occurrences) tool [13], which is part of the MEME Suite [14]. Motif databases typically curate motifs in a format representable with PWMs, and FIMO can take these PWM files to identify genome-wide or chromosome-wide TFBSs to generate FIMO (.fimo) files. Previous methods typically stopped here, but we further rank and analyze these FIMO files which serve as the input to Motif-Cluster.

An important challenge for researchers to note is that predicting TFBSs is a difficult task due to the short length and degeneracy of motifs, as well as the inherent complexity of gene regulation. Experimental validation is a necessary step in prediction and can be used to improve the accuracy of binding site predictions made by tools like Motif-Cluster.

1.2 Motivation for Ranking Transcription Factor Binding Regions

1.2.1 Current Techniques: Individual Motif Ranking

Current techniques for identifying significant regulatory regions in a genome-wide search of TFBSs have underestimated repetitive binding sites in local regions [5]. These

methods typically only consider the individual affinity of motif binding sites rather than the overall affinity of binding regions. There are several existing methods for identifying and predicting TFBSs in this naïve way, which we outline below.

- Positional Weight Matrices: Positional Weight Matrices (PWMs) model the frequencies of each nucleotide at each position in a set of known binding sites in a positional weight matrix. In scanning a genome sequence, scores are assigned to each possible binding site and stored positionally within a PWM. In the end, high-scoring regions may be considered to predict potential binding sites.
- Motif Based Methods: Motif based methods take advantage of the fact that binding sites often contain repeated patterns of DNA sequences called motifs. By searching for similar sequences/motifs with the genome, one can identify TFBSs. Tools like the MEME Suite, FIMO (Find Individual Motif Occurrences) are commonly used for motif-based binding site prediction.
- ChIP-seq: Chromatin Immunoprecipitation followed by sequencing (ChIP-seq) [15] is an experimental technique that can directly identify TFBSs. In this novel technique, antibodies are used to transcription factors bound DNA fragments (i.e., binding sites) and then sequence them to determine their genomic locations.

1.2.2 Improvements to Overall Binding Affinity of Regulatory Regions

These current methods of ranking single binding sites by their individual binding affinities fail to consider the overall binding affinity present in local regions of binding sites. An increased overall binding affinity of these local regions has the potential to indirectly increase the effects of transcription factors on gene expression.

Consequently, current methods of ranking single binding sites by their individual scale may fail to rank regulatory regions accurately and efficiently in genome-wide searches. To solve this essential problem, we designed Motif-Cluster, an open-source tool to rank and visualize the local binding regions of transcription factors [16].

1.3 Challenge Tasks in this Thesis

Current TFBS scanning tools such as FIMO only consider the individual affinity of motif binding sites and fail to incorporate the overall binding affinity of binding regions. Although individual binding affinity is one indicator of significant regulatory regions, including the affinity of TFBS clusters can improve the efficiency and accuracy of ranking regulatory regions.

There are no current methods which can rank clusters or regions given a list of TFBSs that we know of. While naïve methods may find individual TFBSs which have a strong binding affinity by themselves, they may miss TBFS clusters which have many

TBFSs that are weaker but can work together. There are several important factors that indicate a cluster's overall binding affinity, including the sizes of the clusters, the distribution of "gaps" between TFBSs, and the individual binding affinities of the TFBSs in the cluster.

Our challenge for this thesis is to combine all the above factors in ranking and visualizing transcription factors to improve the efficiency of and accuracy of searching for significant regulatory regions.

1.4 Contribution to this Thesis: Design a New Tool to Predict and Rank Transcription Factor Binding Site Clusters

Predicting and ranking TFBSs poses an important but challenging task to researchers. Traditional techniques such as ChIP-seq, while powerful, rely on the presence of high-affinity antibodies in TFBSs and may overlook binding sites with low individual binding affinities. Alternative methods like Cut and Run rely on the presence of experimental data, which is not always available across transcription factors, organisms, tissues, and cell types [17].

To overcome these challenges, we designed Motif-Cluster: a tool which can identify and prioritize significant regulatory regions despite low individual binding affinity. To test and demonstrate the accuracy of our method, we used it on examples like the ZNF410 transcription factor and PHB1 example, where binding clusters play

a crucial role in gene regulation but may be undetected by conventional methods.

1.4.1 Introduction to ZNF410 binding sites

The ZNF410 transcription factor was chosen as a target for cross-validation for its compelling example of binding clusters within the CHD4 promoter region [18]. Characterized as a pentadactyl DNA-binding protein, ZNF410 regulates gamma-globin repressors and demonstrates a unique activation pattern in human erythroid cells by directly targeting the NuRD [19] component CHD4.

This direct targeting and specificity relate to two highly conserved binding clusters (mouse and human) situated near the CHD4 gene. These binding site clusters were independently validated through multiple studies, including Chip-seq, cut-and-run, and further binding motif analysis. ZNF410 [20], being a highly targeted and experimentally validated transcription factor, will serve as the object of evaluation to validate the accuracy of the Motif-Cluster method throughout Chapters 2-5.

1.4.2 Introduction of PHB1

Prohibitin 1 (**PHB1**) [21] is a highly conserved protein with multiple functions. Recent studies have found that FL3, a synthetic derivative called flavaglines, targets PHB1 as a ligand. More information on the new and novel applications of this effect are explored in Chapter 6. In short, because PHB1 binds to the (TGYCC) motif, we can predict the binding sites using the Motif-Cluster method in a novel way.

Chapter 2
Exploration of DBSCAN and Spatial Clustering Approaches

2.1 Introduction to Density-Based Clustering

Density-based clustering [22] is an approach used in data analysis that groups together nearby data points into "clusters" based on how dense they are with respect to each other. Unlike other clustering methods which cluster data points into pre-defined shapes, density-based clustering can group points into arbitrarily shaped and sized clusters. Furthermore, minimal domain knowledge such as the number of clusters is necessary to cluster points based on density. For this reason, this method is particularly effective in identifying clusters of arbitrary shapes in large spatial datasets and is widely used in various domains, from machine learning to genomic analysis.

The density-based clustering techniques we will consider analyze the "neighborhoods" around each data point to group them together: if two data points are found to be within a maximum distance from each other (called epsilon, ε), those points belong in the same "neighborhood." This process can be repeated for each data point to form disjoint sets that represent all the neighborhoods. Finally, clusters

are identified by neighborhoods which have a minimum number of points (minPts) in them, which is given by a parameter to the algorithm. Points belonging to neighborhoods that don't contain this threshold for minimum number of points are said to be outliers or noise.

2.2 Basic DBSCAN Method

2.2.1 Introduction of DBSCAN

Density-based spatial clustering of applications with noise (DBSCAN) is a commonly used algorithm to achieve density-based clustering introduced in 1996 by Martin Ester et al. [23]. Unlike previous clustering algorithms such as GMM and K-Means, DBSCAN is more flexible and practical in use because it requires minimal domain knowledge about the data point space. Specifically, DBSCAN has parameters only for epsilon (ε) and minPts, where other algorithms require parameters such as the number of clusters.

The algorithm classifies points as either core, non-core, or outliers. Core points are those which can reach at least minPts other points directly (i.e., at least minPts are within the epsilon (ε) distance). Non-core points are those which are in the neighborhood of other core points but may not be directly reachable by core points. Outlier points (noise) are those which are not in a cluster.

Consider Figure 2.1 which has 9 points. Circles are drawn around each point

with a radius of epsilon (ε) to illustrate which points belong to the same neighborhood. The minPts argument for this example is 4, so neighborhoods must have at least 4 points to be considered a cluster. Red points denote core points, while yellow points are non-core points (that still belong to the same cluster as the core points). The blue point N is an outlier.

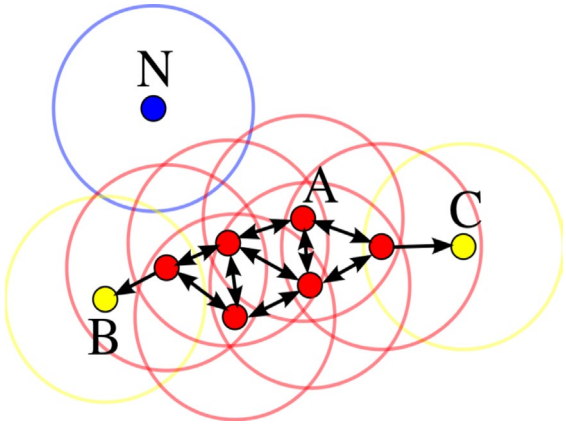


Figure 2.1: Illustration of DBSCAN Method in identifying density-based clusters in data point space as given in [32]

2.2.2 Considering DBSCAN Alternatives

In considering a clustering method to use for Motif-Cluster, we compared DBSCAN to a number of other clustering algorithms. DBSCAN excels at recognizing clusters of high density from those of low density which is crucial in clustering motif matching sites along a linear genomic sequence. However, DBSCAN relies on fixed

parameters epsilon (ε) and minPts which presents problems for motif clustering, since the gaps between binding sites can vary considerably across genomic locations.

To overcome this barrier, we considered other algorithms including HDBSCAN [24] and OPTICS [25]which can consider multiple or hierarchical epsilon (ε) values. In effect, these dynamic algorithms can adapt to differently sized neighborhoods across genome locations. Unfortunately, HDBSCAN and OPTICS do not consider individual weights of data points in calculating neighborhoods, which makes it unsuitable for consider motif matching sites with different affinities. An alternate algorithm which can incorporate weights or signal strength is imperative, so neither dynamic algorithm is viable.

In the next chapter, we will develop a method of applying DBSCAN such that we can consider dynamic values of epsilon (ε) to enable motif clustering across genomic locations with varying gaps between binding sites.

2.3 Application of Basic DBSCAN Method to Motif Clustering

2.3.1 Clustering Method

A basic method of scoring and ranking motif clusters is to apply the DBSCAN method to cluster motif binding sites based on their density. The first step in this

method	ε (epsilon)	min_sample (total weight)	motif_weight
DBSCAN	Gmean	8	10

Table 2.1: Parameters in DBSCAN method.

process is to prepare the data, which comes from a sorted .bed file which includes sorted chromosome start and end positions, and either their p-value indicating motif matching significance OR an equivalent binding affinity score.

The DBSCAN takes parameters for epsilon (ε) and min_sample. For clustering TFBSs, ε denotes the radius of the neighborhood for individual TFBSs which we wish to "cluster" together (we also refer to this as the "gap"). ε takes on the value of the average distance of all motif gaps such that TFBSs closer together than average will be clustered.

- **min_sample:** denotes the minimum number of samples which must be included in a neighborhood to be considered a cluster, but here we use the other possible value for this parameter which denotes the threshold for the total weight of all items in a neighborhood before being considered a cluster.
- **motif_weight:** represents the binding affinity for a single motif binding site. Both values were experimentally found but are unimportant to this basic application of DBSCAN. Their exact derivation and importance will be explained in later sections which expand upon the basic DBSCAN method.

The parameters are summarized by Table 2.1: Parameters in DBSCAN method epsilon (ε) is the most important parameter, which takes on the value Gmean (the

average distance between motif gaps).

2.3.2 Scoring Method

The basic DBSCAN method does not "score" clusters in the traditional sense, but a score may be derived for each cluster based on the number of motif binding sites it contains.

2.4 Results

We applied the basic DBSCAN method to find and rank motifs in the genome region chr12:*6,716,600-6,724,000* which are the ZNF410 binding clusters on the CHD4 promoter region.

A few example lines from the input .bed file containing information about the human genome chr12 are show below:

Figure 2.2 summarizes the results found by applying the basic DBSCAN method for the ZNF410 transcription factor in the CHD4 promoter region. Each cluster is assigned a different color, which is derived based on proximity/density of binding sites. The x-axis covers the coordinate domain in chr12 (*6,716,600-6,724,000*), and the y-axis denotes the weight of each peak (which is consistently 10 for the basic application of DBSCAN).

chr12	60025	60042	TCCATTCCCTAGAAGGC	-1421	+	MA0752.1	P-value=5.29e-04
chr12	60063	60080	TCCATTCCCTAGAAGGC	-1421	+	MA0752.1	P-value=5.29e-04

Table 2.2: An example of two lines of a sorted bed file.

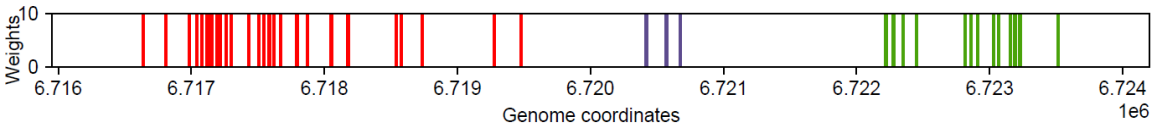


Figure 2.2 : DBSCAN Method performance for ZNF410 binding clusters.

Chapter 3
Method Development with Group-Aware Approach

3.1 Motivation

The first method of applying the basic DBSCAN to cluster motif regions resulted in our region of interest being ranked very low. The uneven distribution and highly variable distance (epsilon (ε)) between the motif binding sites does not pair well with the basic DBSCAN method because it uses fixed parameters.

The basic DBSCAN does not differentiate between gap sizes, which leads to less meaningful clustering in the context of motif binding sites. It is not able to heuristically filter out binding site gaps belonging to different regions. Furthermore, the basic method does not account for the strength of each binding site; DBSCAN applies a uniform weight to everything and does not factor density into the weight of each binding site.

Thus, we are motivated to create a group-aware approach that improves the original DBSCAN method to address these issues. The group-aware method uses the Gaussian Mixture Model (GMM) to identify probable Gaussian components for the binding gaps and to calculate their mean and variance. The number of the

components are based on their observed Akaike Information Criterion (AIC) and Bayesian Information Criterion (BIC) metrics, which result in a more accurate representation of the biological data and distribution of gaps.

3.2 Methodology

3.2.1 Basic Knowledge

A Gaussian mixture model (GMM) is parametric probability density function that is expressed as a weighted sum of Gaussian component densities [26]. These models are frequently used to represent the probability distributions for continuous measurements or features within a biometric system and are a good candidate for motif clustering. The GMM parameters are found through iterative learning algorithms such as expectation-maximization (EM) or maximum a posteriori (MAP) [27],[28]. These algorithms take advantage of well-trained prior data and tweak the parameters to best-fit the data at hand.

The EM algorithm comprises of two distinct steps which are repeated iteratively until the model converges: Expectation and Maximization. The Expectation step (E step) involves calculating the probability of each data point belonging to each distribution then assessing the likelihood function based on the current estimation of the parameters. The Maximization step (M step) updates the running mean, covariance, and weight parameters to maximize the expected likelihood derived in

the E step. By repeating these steps, the expected parameters will converge to a point which maximizes the likelihood function.

The Akaike information criterion (AIC) [29] helps balance the accuracy and complexity of model by quantifying the goodness-of-fit of the model while penalizing the model for its complexity. AIC is calculated using Equation (3.1).

$$\text{AIC} = -2 \times log(L) + 2 \times k \tag{3.1}$$

Bayesian information criterion (BIC) [30] is similar to AIC but uses Bayesian principles to apply a different formula for the penalty for model complexity. BIC is calculated using Equation (3.2):

$$\text{BIC} = -2 \times log(L) + K \times log(n) \tag{3.2}$$

In both Equation (3.1) and Equation (3.2), L is the maximum likelihood estimate given by the likelihood function; K is the number of parameters in the model (which is the number of groups in our application); and n is the sample size which refers to the number of data points used to estimate the model.

3.2.2 Methodology Overview

To apply the group-aware approaches which take advantage of learning models to parameterize the DBSCAN algorithm, the .bed files must be preprocessed first. Binding gap sites exceeding 500 base pairs must be filtered out to enable us to focus

on repetitive binding regions with sufficiently long gaps, while avoiding excessively long gaps that decrease the potential significance of regulatory sites.

The second step is to identify the optimal number of Gaussian mixture components using the iterative BIC or AIC algorithms. These algorithms balance the trade-off between the time cost and score of the likelihood function to find a suitable number of Gaussian mixture models for the GMM algorithm. Using this method, we have experimentally found that 10 is the ideal number of Gaussian components; incorporating more components into the model will not significantly increase the metrics for the BIC and AIC models.

Armed with the ideal number of Gaussian components (10), we then perform the union method and apply DBSCAN across each component using the group's group-wise mean and standard deviation as parameters. This achieves variation in the DBSCAN parameters across the components to identify motif clusters.

To identify the motif clusters and account more accurately for different types of gaps in the genome, the groups need to be combined (union) and subdivided in cases of overlap between groups. This results in the final clusters being more finely tuned to account for different types of gaps.

Once the final clusters are identified, we can finally score and rank them. The final score of each cluster is derived from the posterior probabilities of both peak intensity and gap significance.

3.2.3 Method Details and General Example

- **Clustering Method:**

In the example outlined below, we explore applying the group-aware approach to the whole human genome chr12 data.

We have experimentally found that 10 is the optimal number of groups based on both the AIC and BIC scores, as depicted by Figure 3.1. This step only needs to be completed once, and henceforth 10 is always used as the number of components.

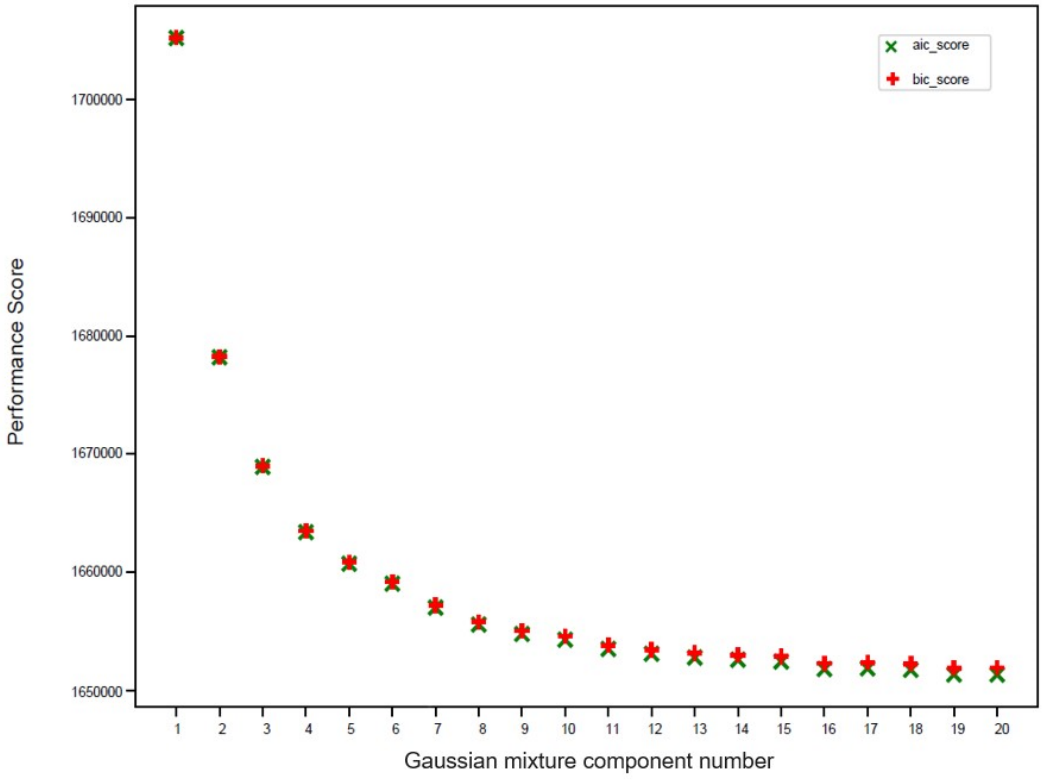


Figure 3.1: Akaike information criterion (AIC) and Bayesian information criterion

(BIC) performance scores vs. Gaussian mixture component number for identifying groups for motif binding gaps.

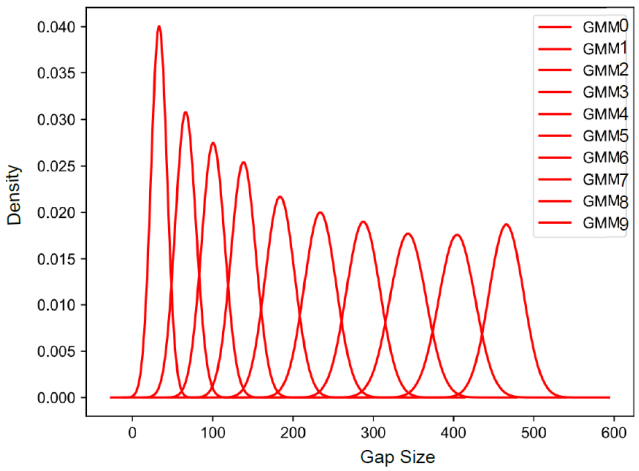


Figure 3.2: Density vs. gap size for ten Gaussian groups distributions.

Figure 3.2 depicts the 10 Gaussian mixture components precisely. Gap size is between 0-500, as gaps greater than 500 were not present in the data and therefore hold no research value. For each component, the corresponding covariance is plotted along the y-axis.

Table 3.1 summarizes the DBSCAN parameters used for each Gaussian component (G_i) based on the group-wise mean ($\mu(G_i)$) and standard deviation ($\sigma(G_i)$) of gap groups. To loosen the neighborhood radius in DBSCAN, we use the group-wise mean plus twice the standard deviation from each gap group as the radiuses (ε). The min_sample and motif_weight have not been modified from Chapter 2.

method	ε (epsilon)	min_sample (total weight)	motif_weight
Group-Aware Approach	$\mu(G_i) + 2\sigma(G_i)$	8	10

Table 3.1: Internal parameters in group-aware DBSCAN.

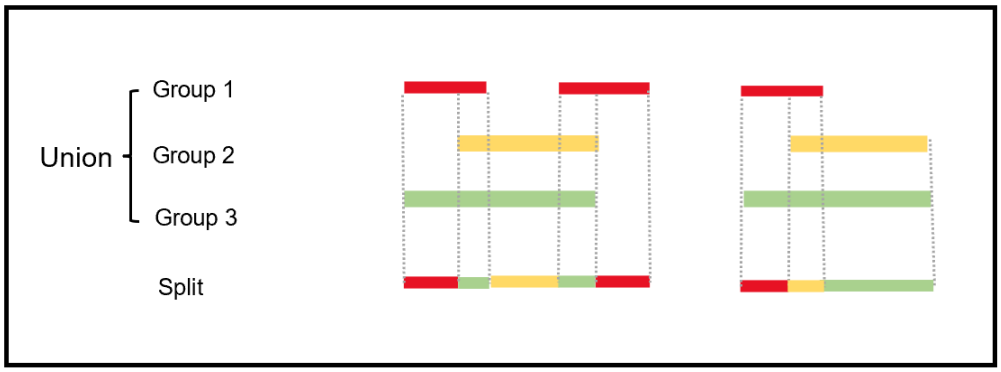


Figure 3.3: Combining three groups of clustering (union) while also subdividing them into smaller clusters in cases of overlap (split).

Gaps are calculated using a bed (.bed) file format, which is commonly used in bioinformatics to represent genomic coordinates and associated features. The format we use comprises the following columns:

```
Chromosome: The name of the chromosome or contig where the feature is located.
Start:      The starting position (0-based) of the feature on the chromosome.
End:        The ending position (1-based) of the feature on the chromosome.
Name:       A unique identifier or name for the feature (optional).
```

We use the '**union_bedgraphs**' function of the '**pybedtools**' which is a Python package to combine multiple BED (.bed) files into a single file which generates coverage comparisons between the files. The function assumes that each file is sorted

by chromosome and start, and that the intervals are non-overlapping.

Usage is shown as below:

```
Usage:   bedtools unionbedg [OPTIONS] -i FILE1 FILE2 .. FILEn
```

Figure 3.3 illustrates how the groups of clustering are combined (union) and subsequently subdivided into smaller clusters in cases of overlap (split). In the real algorithm, we would have 10 groups of clustering instead of 3.

- **Scoring Method:**

The final score for each cluster is derived from the posterior probabilities of both peak intensity and gap significance. Regions with higher-intensity peaks are ranked higher, however regions with lower-intensity peaks can also hold significance if they contain many TFBSs.

The GMM is incorporated into the score to disfavor atypical gaps that deviate from chromosome or genome-wide statistics, such as those that are unusually short or long. To this end, the mean gap distance for the group is used to calculate the best-fitted gap group according to the GMM using Equation (3.3).

$$i = argmax_i P(I_{mean}|C_k, G_i) \tag{3.3}$$

$\mathrm{I_{mean}}$ is the mean binding affinity (peak intensity) for cluster C_k. We assign the cluster to the most probably group G_i by maximum likelihood and build a group-

specific intensity distribution (empirical distribution $f(I|G_i)$) to accommodate the varying contributions of the motif binding intensity given a specific group. As a result, the final score of each cluster is computed as the cumulative sum of the log-likelihoods of the peak intensities and posterior gap probabilities, given by Equation (3.4):

$$Score(C_k) = -\log(P(C_k \mid I_j, G_i)) = -\sum_j \log(P(I_j \mid G_i)) \quad (3.4)$$

C_k refers to the kth motif cluster assigned to the gap group i; $P(I_j|G_i)$ is the empirical probability for all binding site intensity I_j within the cluster C_k. This score is used to rank and prioritize all motif clusters in order to identify the most significant regulatory regions by their top score. In incorporating both peak intensity as well as the region metrics, the score function will work for binding sites with strong affinities as well as clusters with medium or low binding affinity but a large, repetitive cluster size. At the same time, random, low-affinity binding sites that can be considered noise yield relatively low scores.

3.3 Results

Figure 3.4 visualizes the group-aware DBSCAN performance of ZNF410 bindings cluster in the CHD4 promoter region (in the human chr12 between *6717000-6724000*). The genome coordinates are plotted across the x-axis, while the weight of each peak is on the y-axis. The first 10 subfigures belong to the 10 components,

while the last is comprised of the union-split of the groups.

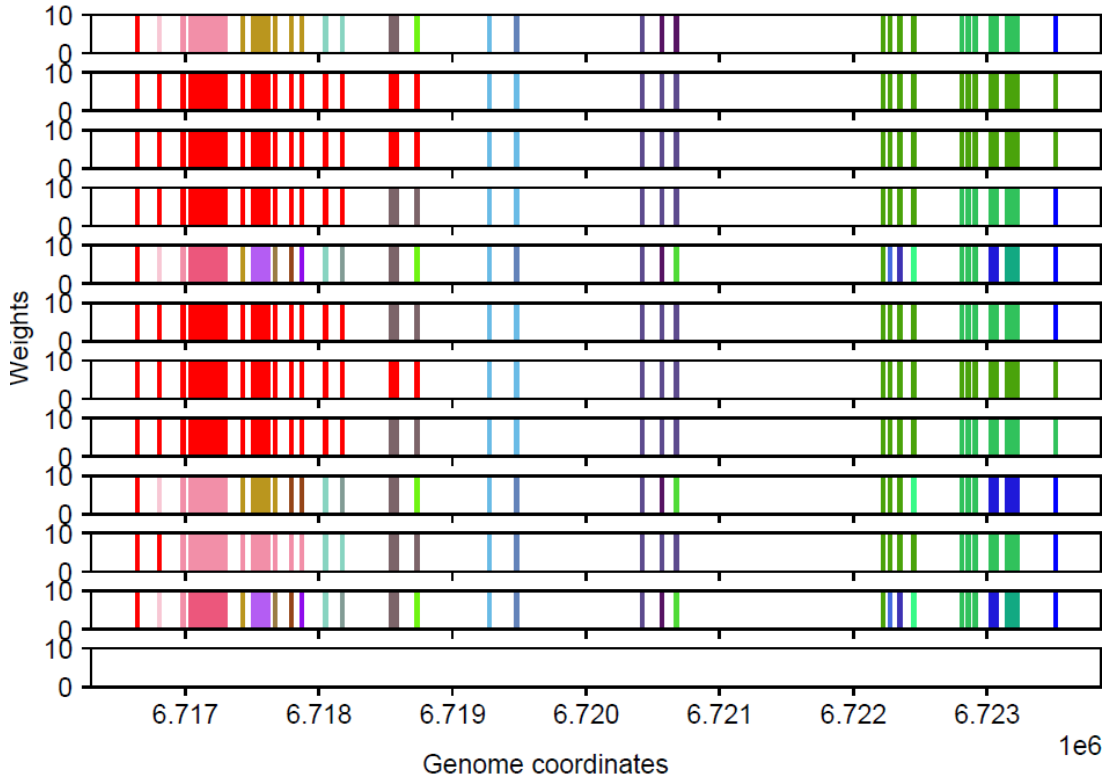


Figure 3.4: Group-aware DBSCAN Methods performance for ZNF410 binding clusters in CHD4 promoter region.

Figure 3.5 compares the original motif clustering using the basic DBSCAN method (as found in Chapter 2) compared to the group-aware DBSCAN clustering. The first row depicting the basic DBSCAN clustering has excessive clustering of certain binding sites that do not exhibit consistent gap ranges. In contrast, the second row depicting the group-aware DBSCAN clustering produces more refined clusters with comparable gap patterns.

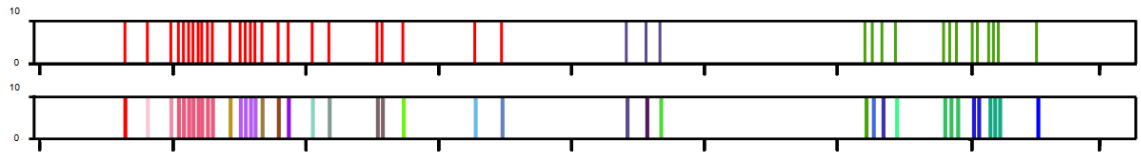


Figure 3.5: Clustering performance comparison between clustering algorithms for the ZNF410 binding clusters in CHD4 promoter region. From top to bottom: (first row) the basic DBSCAN clustering; (second row) group-aware DBSCAN method.

Chapter 4
Method Improvement with Cluster Merging and Signal Weight

4.1 Motivation

In cross-validating the results of the group-aware DBSCAN method with known binding site affinities, we identified two issues which we aimed to rectify: first, the method fails to properly exclude noise from the data; and second, the high-ranking clusters identified were all shown to be low-affinity binding sites which should be ranked lower. To fix these two issues, we further refine the clusters through a process of merging.

4.2 Refinement by Merging

Motif-Cluster incorporates a final step called “merge” to overcome improper ranking of potential clusters and the presence of singletons outliers in clusters. Clusters are merged if the gap between them shares a common, tolerated gap with the clusters of interest. The gap must meet one or two merging criteria:

- “Merge Strategy 1” allows for the fusion of two nearby clusters if the gap

between them can be assigned to the same gap group as one of the nearby clusters according to the Gaussian component.

- "Merge Strategy 2" allows for the fusion of a cluster and one nearby singleton if the gap between them can be assigned to the same gap group as one of the nearby clusters according to the Gaussian component. One exception exists to this criterion: the two can be fused if the gap between them belongs to the second most probable gap group, if the first most probable gap group is no more than 1.5 times the second probability.

The parameters for DBSCAN with merging remain the same as the group-aware approach, summarized by Table 3.1.

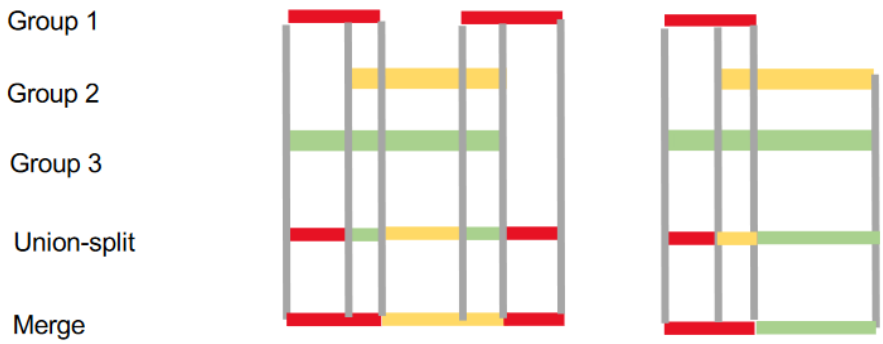


Figure 4.1: the final merge step applied to the results of union-split to combine nearby clusters and outliers with similar gaps.

The final row of Figure 4.1 depicts the merge step as applied to the results of the union-split step of group-aware DBSCAN. Here, nearby clusters and outliers are

fused together if the gap between them meets the criteria described by the two merge strategies.

4.3 Refinement by Adding Signal Weight

4.3.1 Introduction to Signal Weight

In the context of our BED (.bed) files, we will introduce the idea of "p-values" and signal weights to incorporate into our ranking. The p-value refers to the statistical significance of a particular peak, where a low p-value suggests that the peak has high statistical significance. We define the signal weight as -log (p-value) which provides a measure of the signal intensity or level of enrichment for a specific feature across the genome.

We incorporate signal weight into our algorithm in order to properly weigh binding sights with differing affinities. All very low-affinity binding sites will be removed from the high ranking.

Finally, we introduce a non-static motif_weight parameter to DBSCAN according to the signal weight. Complete parameters to DBSCAN are summarized by Table 4.1.

method	**ε (epsilon)**	**min_sample (total weight)**	**motif_weight**
With signal weight	$\mu(G_i) + 2\sigma(G_i)$	8	$-\log_{10}$ (p-value)

Table 4.1: Internal parameters in DBSCAN for signal weight for motif_weight parameter.

4.4 Results

4.4.1 Group-Aware DBSCAN Method with Merge Strategy

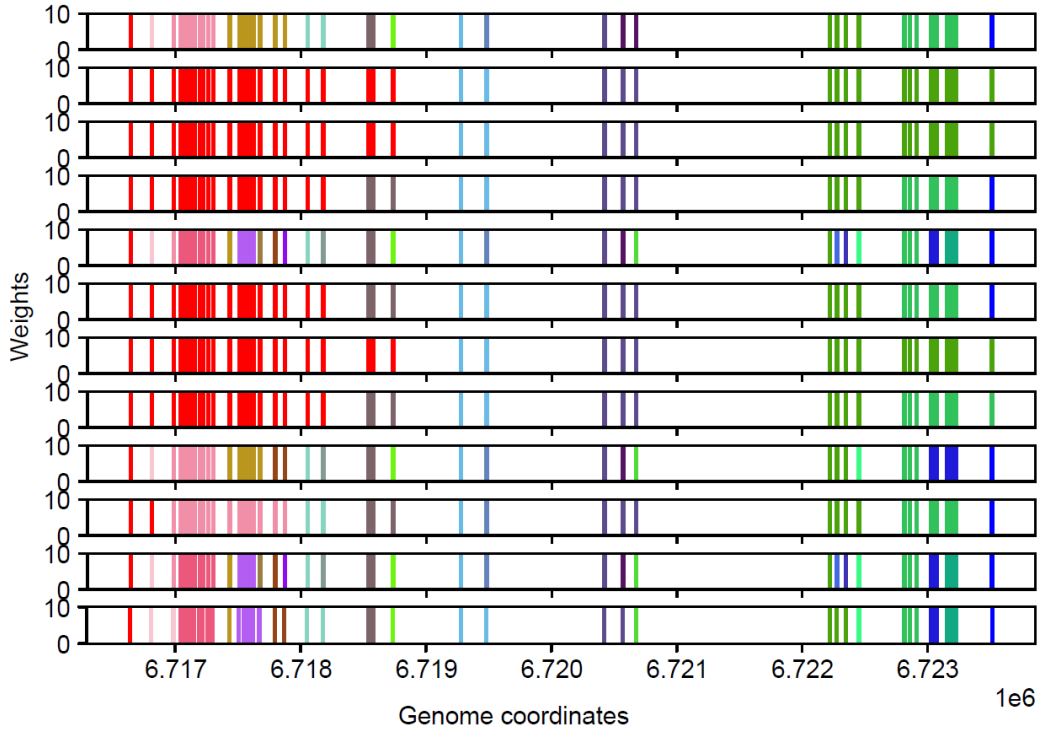


Figure 4.1: Results of group-aware DBSCAN Methods with cluster merging for ZNF410 binding clusters in CHD4 promoter region.

Figure 4.1 depicts the results of a group-aware DBSCAN with merging applied to the human genome region chr12:*6,716,600-6,724,000* which are the ZNF410 binding clusters on the CHD4 promoter region. The first 10 rows depict the individual Gaussian component clustering. The 11th row depicts the results of union-and-split. The 12th row depicts the final merged result of the union-and-split clusters.

Figure 4.2 compares the final clustering result between the basic DBSCAN method (Chapter 2), the group-aware DBSCAN method (Chapter 3), and the group-aware DBSCAN with merging (Chapter 4) as applied to the human genome region chr12:*6,716,600-6,724,000* which are the ZNF410 binding clusters on the CHD4 promoter region. Compared to the group-aware DBSCAN without merging, the merging results in better clustering by merging some single peaks to nearby clusters.

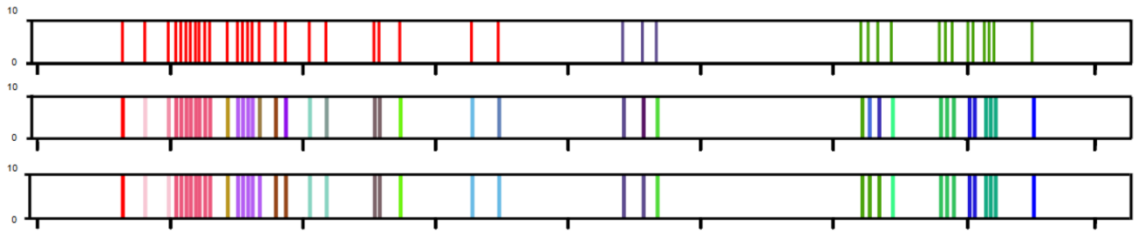


Figure 4.2: Compare performance for ZNF410 binding clusters in CHD4 promoter region. From top row to bottom. (First row) DBSCAN Method, (second row) group-aware DBSCAN Method, (third row) group-aware Method DBSCAN with merging.

4.4.2 Group-aware DBSCAN with Signal Weight

Figure 4.3 depicts the results of a group-aware DBSCAN with added signal weight applied to the human genome region chr12:*6,716,600-6,724,000* which are the ZNF410 binding clusters on the CHD4 promoter region. The first 10 rows depict the individual Gaussian component clustering. The 11th row depicts the results of union-and-split. The y-axis represents the signal weight of each binding site. Outliers are

illustrated as dotted lines.

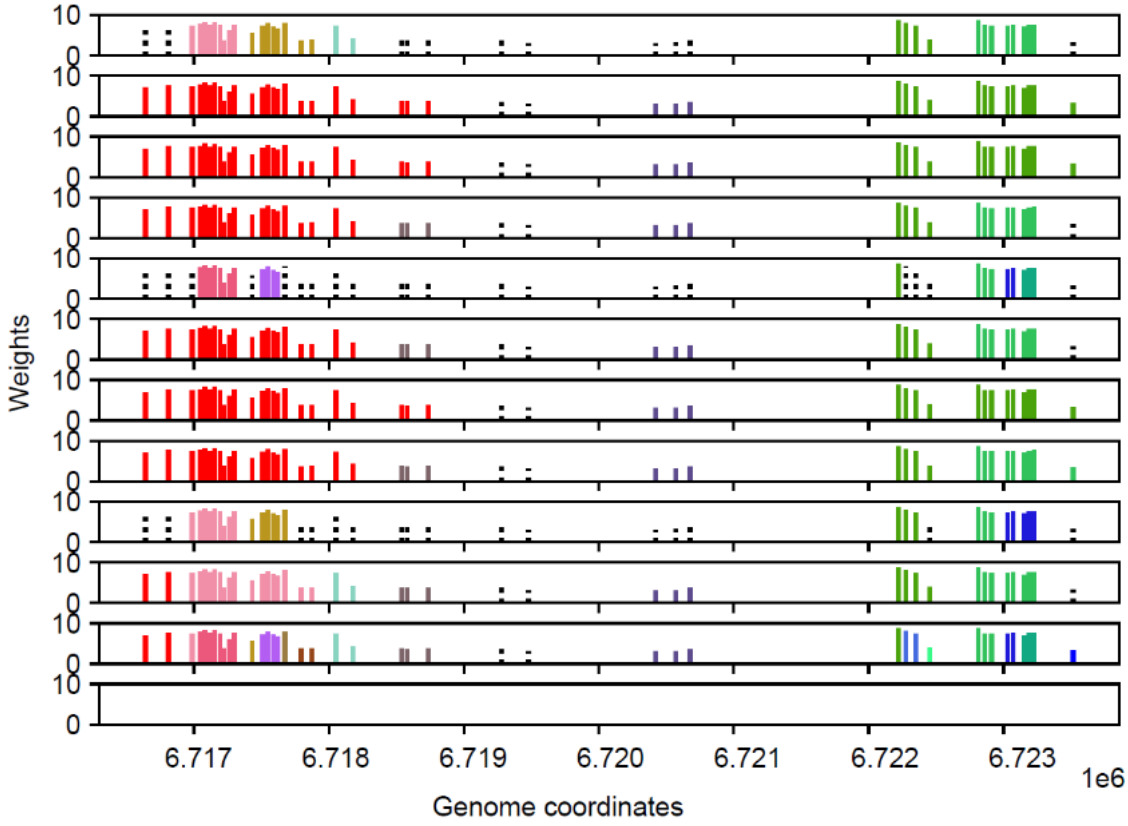


Figure 4.3: Group-aware DBSCAN Methods with added Signal Weight for ZNF410 binding clusters in CHD4 promoter region.

4.4.3 Motif-Cluster Method (with Merging and Added Signal Weight)

- **Clustering Method:**

Finally, we find and compare the results of the Motif-Cluster Method, which we

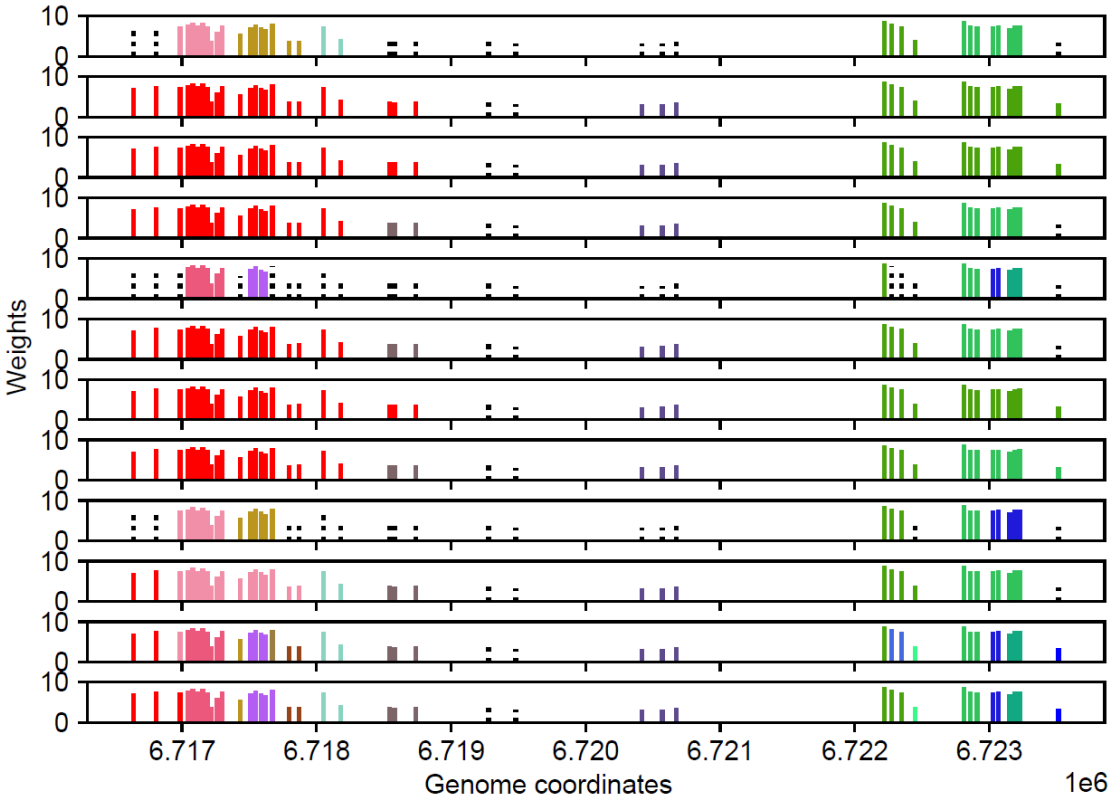


Figure 4.4: Motif-Cluster method results for ZNF410 binding clusters in CHD4 promoter region.

define as the group-aware DBSCAN method with merging and added signal weight. Figure 4.4 depicts the Motif-Cluster method applied to the human genome region chr12:*6,716,600-6,724,000* which are the ZNF410 binding clusters on the CHD4 promoter region. The first 10 rows depict the individual Gaussian component clustering. The 11th row depicts the results of union-and-split. The 12th row depicts the final merged result of the union-and-split clusters. The y-axis represents the signal weight of each binding site. Outliers are illustrated as dotted lines.

- **Scoring Method:**

Scoring method is used the same as the Chapter 3's scoring method.

4.5 Motif-Cluster Method

Here we will examine the Motif-Cluster Method, which we define as the group-aware DBSCAN method with merging and added signal weight.

4.5.1 Motif-Cluster Performance:

- **Anti-Noise Performance:**

Figure 4.5 depicts the rank for ZNF410 for two different p-values, identifying them as high-affinity sites. The ideal p-value threshold for finding reliable binding scores is a complex task that varies for different transcription factors. The list of p-values less than 0.005 are more precise, while the broader set with p-values less than 0.01 include more noise. Figure 4.5 demonstrates that the top 20 significant regions identified in the high-affinity category ($p<0.001$) are preserved in the set with more noise ($p<0.01$). Of 20 regions identified with $p<0.001$, 7 were found in the $p<0.01$ set.

Figure 4.6 shows that of the 20 regions identified with $p<0.001$, 9 were found in the $p<0.005$ set. This suggests that increasing the p-value threshold increases the noise, but this effect is not critically detrimental. This likely demonstrates that the Motif-Cluster method is effective in accounting for both binding affinity and clustering patterns.

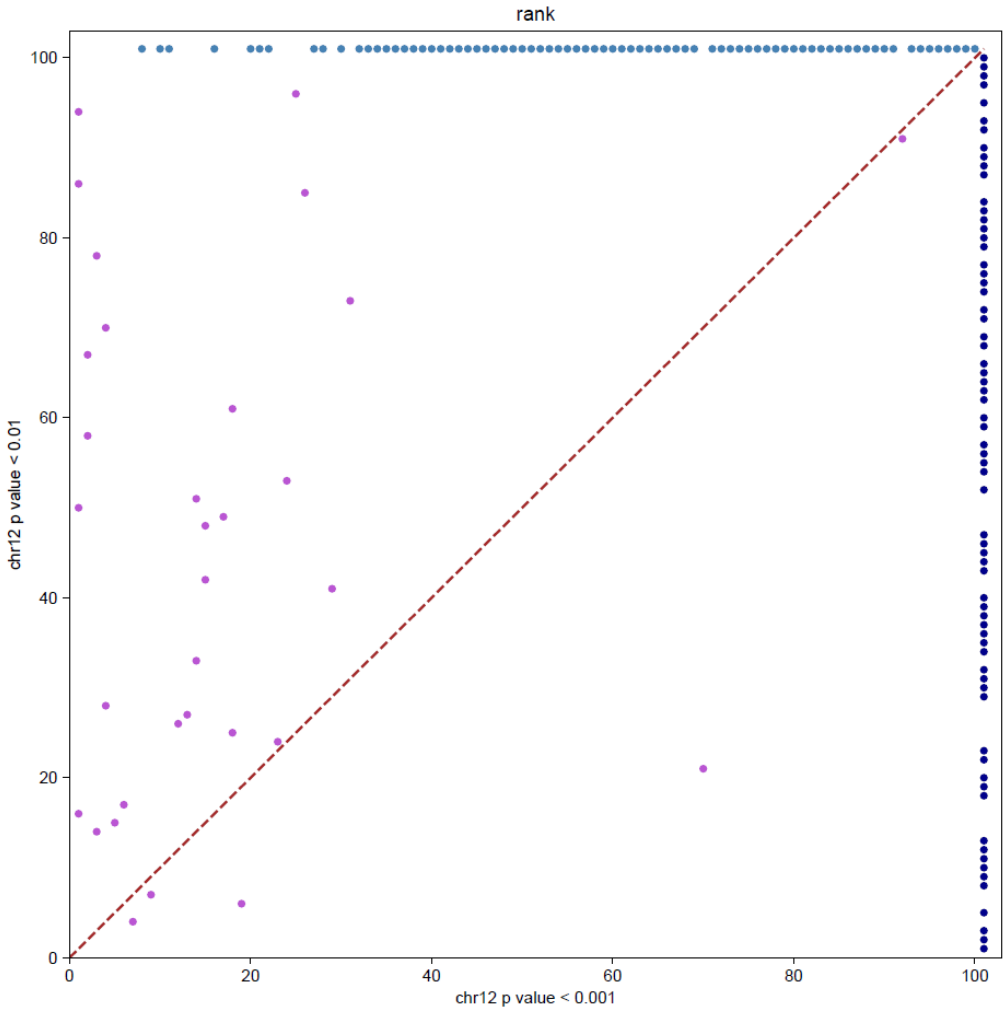


Figure 4.5: Performance with and without noise. Prioritized top 100 regions in both p<0.001 and p<0.01.

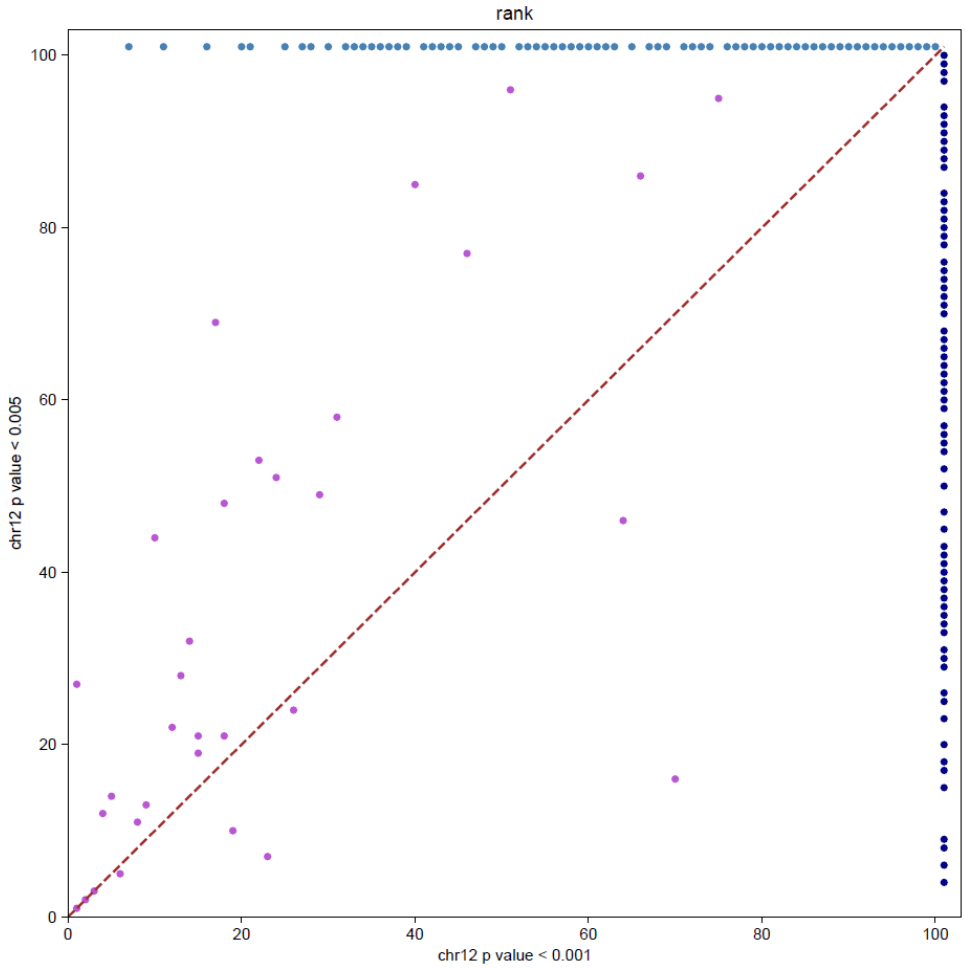


Figure 4.6: Performance with and without noise. Prioritized top 100 regions in both p<0.001 and p<0.005

- **Running-Time performance:**

Figure 4.7 shows after running different number of peaks: 10000, 50000, 100000, 150000, 200000, 227307, the corresponding running time in Step 1 (cluster and merge) and Running time in Step 2 (score and rank).

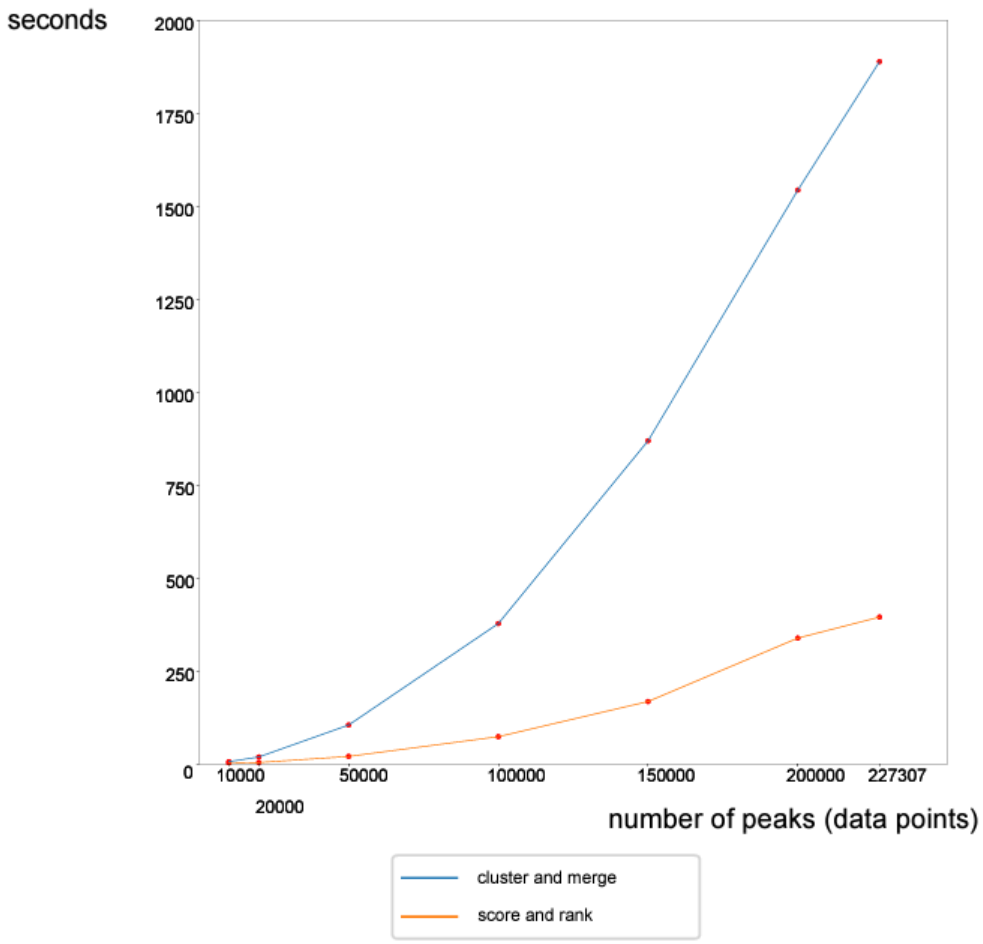


Figure 4.7: Motif-Cluster running time.

4.6 Compare Motif-Cluster with Previous Methods

In some parts of this section, we will introduce a shorthand for referencing the various methods discussed in this paper. You may reference Table 4.2 for a description of each method.

Method	Description	Merging	Signal Weight
a	Basic DBSCAN with neighborhood	No	No
b1	With groups for binding site gap (Group-aware)	No	No
b2	Group-aware with merge	Yes	No
c	Group-aware with signal weight	No	Yes
d	Motif-Cluster Method	Yes	Yes

Table 4.2: Shorthand descriptions of clustrering and ranking methods.

4.6.1 Comparison for Clustering Visualization

Figure 4.8 compares the clustering figures generated by various methods applied to the human genome region chr12:*6,716,600-6,724,000* which are the ZNF410 binding clusters on the CHD4 promoter region. The first row depicts the basic DBSCAN approach which results in excessive clustering of certain binding sites and a lack of uniformity in gap ranges. This lack of uniformity is solved in row two (as well as subsequent rows), which includes group-aware DBSCAN in the clustering. The third and fifth row include the merge step, which results in less isolated singletons or clusters. The fourth and fifth row include the addition of signal weight to effectively exclude low-affinity TFBSs. The final row, then, is the Motif-Cluster method which includes both merging and the addition of signal weight.

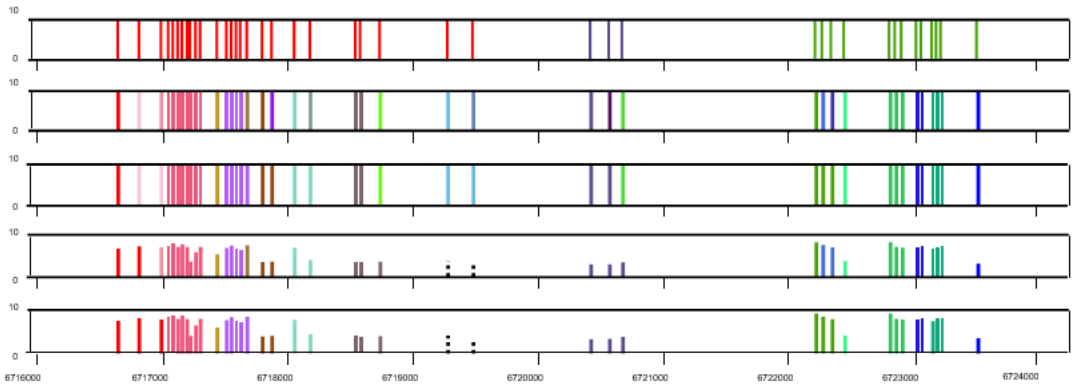


Figure 4.8: Methods performance for ZNF410 binding clusters in CHD4 promoter region. (First row) DBSCAN; (second row) group-aware DBSCAN; (third row) group-aware DBSCAN with merge; (fourth row) group-aware DBSCAN with signal weights; (fifth row) Motif-Cluster.

To sum up, Motif-Cluster method can have clusters with consistent gap range, and at the same time expansion of clusters in some certain ways. And also can remove the low-affinity sites when cluster size does not meet the requirement.

4.6.2 Comparison for Clustering Size

Table 4.4 demonstrates that Motif-Cluster (method d) generates clusters with a size of 3 in 46.83% of cases. Compared to other methods in Table 4.3 and Table 4.4, Motif-Cluster generates isolated data points (singletons) much less which indicates that Motif-Cluster's approach, which combines signal weight and group-aware strategies, is more effective in identifying clusters of binding sites, rather than

isolated TFBSs.

Cluster size	Methods (number of clusters)				
	a	**b1**	**b2**	**c**	**d**
1	104990	195980	60053	95107	12257
2	32763	13559	71870	10432	11858
3	10751	1083	6676	27362	27443
4	3400	116	642	5895	5987
5	1158	23	89	790	807
6	413	5	10	174	177
7	144	5	9	31	32
8	58	7	10	15	17
9	23	4	4	10	10
10	9	3	4	4	5
>10	49	12	12	13	13
total	153758	210797	139379	139833	58606

Table 4.3: Number of clusters for different cluster size using various methods.

Cluster size	Methods (percentage)				
	a	**b1**	**b2**	**c**	**d**
1	68.28%	92.97%	43.09%	68.02%	20.91%
2	21.31%	6.43%	51.56%	7.46%	20.23%
3	6.99%	0.51%	4.79%	19.57%	46.83%
4	2.21%	0.06%	0.46%	4.22%	10.22%
5	0.75%	0.01%	0.06%	0.57%	1.38%
6	0.27%	0.00%	0.01%	0.12%	0.30%
7	0.09%	0.00%	0.01%	0.02%	0.06%
8	0.04%	0.00%	0.01%	0.01%	0.03%
9	0.02%	0.00%	0.00%	0.01%	0.02%
10	0.01%	0.00%	0.00%	0.00%	0.01%
>10	0.03%	0.01%	0.01%	0.01%	0.02%
total	100.00%	100.00%	100.00%	100.00%	100.00%

Table 4.4: Percentage of clusters for different cluster size using various methods.

4.6.3 Comparison for Rank of the CHD4 Promoter Region

Method	Description	Rank of the CHD4 promoter region
a	Basic DBSCAN with neighborhood	14
b1	With groups for binding site gap (Group-aware)	20
b2	Group-aware with merge	21
c	Group-aware with signal weight	8
d	Motif-Cluster Method	2

Table 4.5: Comparison for rank of the CHD4 promoter region for all methods.

From Table 4.5, notably the cross-validated region identified experimentally for CHD4 attains a rank 2 out of a total of 58,606 clusters. Compared to other methods, Motif-Cluster is able to effectively filter out the many putative binding sites and rank CHD4 near the top.

Chapter 5
Motif-Cluster Package and Usage

5.1 Introduction to Motif-Cluster

In order to make a tool which implements the procedures described in the previous chapters and facilitate the user to rank and visualize motif regions, we developed a comprehensive Python package "Motif-Cluster" [16].

5.2 Design Ideas/Procedure

We partition the Motif-Cluster method into two major steps: Cluster and Merge (step 1), and Score and Rank (step 2). By separating these into distinct steps, one can visualize the results of step 1 without having to redo clustering. There are various visualization functions which visualize both the results of step 1 and step 2.

Each command by default utilizes default parameters which define the Motif-Cluster method. For comparison purposes, you may pass in parameters that alter the algorithms used by step 1 and step 2. These arguments and alternative commands are described in section 5.8

5.3 Installation Instructions

5.3.1 Installation

Begin by downloading the code and activating a new Conda environment:

```
git clone https://github.com/yao-laboratory/MotifCluster.git
cd MotifCluster
conda create -n motifcluster
```

Install the necessary packages:

```
conda install python="3.9.10"
pip install -r installation_packages/requirements_pip.txt
conda install --file installation_packages/requirements_conda.txt
```

5.3.2 Command Overview

Run the program without any arguments to view sub-commands. More detailed information may be viewed in the project's README.md.

```
usage: MotifCluster [-h] {…}

positional arguments:
  {…}
                        Sub Commands Help
    cluster_and_merge_simple_dbscan
                        Using direct DBSCAN method to identify
                        local motif clusters

    cluster_and_merge   Identify local motif clusters

    pre_process         Convert fimo.tsv file into the sorted bed
                        file

    calculate_score     conduct scores and give ranks for all
                        clusters

    draw                Draw a region of interest and it shows
                        the distinctively colored clusters view.

{…}
```

5.4 Input and Output files

See Table 5.1 where most of the main commands take in input files as arguments and will output predefined files to the output folder specified by the -output_folder

argument. The order you run the commands is important because the output of some commands must serve as the input files to subsequent commands.

Command	Input file(s)	Output file(s)
pre_process	.tsv	.bed
cluster_and_merge	.bed	results.csv, results_middle.csv, results_draw.csv
calculate_score	.bed, results.csv, results_middle.csv	results_score.csv, results_cluster_weight.csv
draw	.bed, results_draw.csv	draw_figure.pdf
draw_rank	results_score.csv	normal_vs_noise_rank.pdf
draw_score_size	results_score.csv	score_size.pdf
draw_cluster_weight	result_cluster_weight.csv	cluster_weight_draw.pdf
draw_GMM	.npy	GMM_drawing.pdf
cluster_and_merge_ simple_db_scan	.tsv	results_simple_DBSCAN.csv, results_middle_simple_DBSCAN.csv, results_draw_simple_DBSCAN.csv

Table 5.1: Input and output files for the main program commands.

It should be noted that the output files for cluster_and_merge are different for Method b1 and Method c. The output files are documented in their respective sections.

TSV (.tsv) refers to the tab-separated-value FIMO file output from The MEME Suite. We may refer to an instance of this file as ‘fimo.tsv’ , but they may have other names. Each line describes a significant match to a motif, and lines are sorted in decreasing order of statistical significance.

Here are a few sample lines from one such input file shown in Table 5.2:

motif	NC_000016.9	122369	122379	+	12.8	1.53e-07	0.0699 GGCCCCGGCCC
motif	NC_000016.9	122375	122385	+	12.8	1.53e-07	0.0699 GGCCCCGGCCC
motif	NC_000016.9	188276	188286	-	12.8	1.53e-07	0.0699 GGCCCCGGCCCT

Table 5.2: fimo.tsv example file.

chr16	61384	61394	GGCCCCAGCCC	-	P-value=1.83e-06
chr16	62064	62074	GGCTTTGGCCC	+	P-value=1.77e-05
chr16	62660	62670	GGCCTGGGCTC	-	P-value=1e-05

Table 5.3: A sorted .bed file example.

A fimo.tsv file must be preprocessed into a sorted bed (.bed) file before it may be used as an input to the main program commands. Bed files should be stored in the input_files folder, which is where they are output during the preprocess step by default. Here is an example of the sorted bed file sorted_chr16.bed shown in Table 5.3.

Other output files are described in more detail in the section they are generated during. For example, results.csv is described in the section on preprocessing.

5.5 Preprocessing Functions

The preprocessing functions are used to convert .tsv files into a sorted bed file suitable for the main program commands. The general command and program arguments are documented below:

```
usage: python3 MotifCluster/MotifCluster.py pre_process -input_name -
                        output_name -chrome

 required arguments:
 -input_name FILENAME,   FILENAME: your input file name, the file
                         should be put into the input_files folder.
                                  (located: Motif_Cluster/input_files)
 -output_name FILENAME,  FILENAME: your customized output file name,
                         the file automatically
                                   put into the input_files folder.
 -chrome CHROME,         CHROME:   chrome name in the bed file,
                         eg.chr16
```

Here is an example of preprocessing fimo_chr16.tsv to produce the sorted_chr16.bed file:

```
python3 MotifCluster/MotifCluster.py pre_process -input_name
fimo chr16.tsv -output name sorted chr16.bed -chrome chr16
```

5.6 Motif-Cluster Method

5.6.1 Step 1: Cluster and Merge

The cluster and merge command identifies local motif clusters by utilizing the Motif-Cluster Method. The various command parameters are documented below:

```
usage: python3 MotifCluster/MotifCluster.py cluster_and_merge
               -input -merge_switch -weight_switch -output_folder [-start -end]

 required arguments:
 -input        FILENAME,  FILENAME: your input file name(Note: sorted bed file),
                            the file should be put into the input_files folder.
                            (located: Motif_Cluster/input_files)
 -merge_switch STATUS,    STATUS: on or off,
                          on: run the program including merge step,
                          off: run the program without including merge step
 -weight_switch STATUS,   STATUS: on or off,
                          on: run the program including weight information,
                          off: run the program without weight information
 -output_folder FOLDER,   FOLDER: your customized output folder name

 optional arguments:
 -start NUM               NUM: the start coordinate of processing this input bed file
 -end   NUM               NUM: the end coordinate of processing this input bed file
```

Here are a few examples in which we cluster and merge the human_chr12_origin.bed input file. The second command specifies that -start and -end command to only process part of the input bed file:

```
python3 MotifCluster/MotifCluster.py cluster_and_merge -input
            human_chr12_origin.bed -merge_switch on  -weight_switch on
            -output_folder example_output_step1_1
python3 MotifCluster/MotifCluster.py cluster_and_merge -input
            human_chr12_origin.bed -merge_switch on  -weight_switch on
            -output_folder example_output_step1_2 -start 6716000 -end
            6724000
```

There are several output files from the cluster and merge step which will be generated in the output folder specified by -output_folder:

- Temporary processing files
- results.csv
- results_middle.csv
- results_draw.csv

The temporary output files should not be directly used but serve as input to the Score and Rank step. You may ignore the format of these files for the meantime, as they are described in later steps.

See Table 5.4 where the first output file results.csv (previously referenced to as cluster-union.csv in this paper) is formatted such that each line is the nth line (peak) from the original bed file's information. Each column describes some property of the peak, such as the center position (center_pos) or weight. The class_id column

indicates the group the peak belongs to. When consecutive peaks have the same class_id and group, they may be aggregated into a single cluster

id	center_pos	start_pos	end_pos	class_id	weight
1	60033	60025	60042	4	3.2765443279648143
2	60071	60063	60080	4	3.2765443279648143
3	60109	60101	60118	4	3.2765443279648143

Table 5.4: results.csv output from cluster and merge.

See Table 5.5 where the second output file result_middle.csv file is formatted such that each line is the nth cluster from the original bed file. The number of peaks and the group for the cluster in the original bed file are indicated by ‘data_count_new’ and ‘cluster_belong_new’, respectively. ‘data_count_sum’ may be ignored, and is only used internally.

id	data_count_new	cluster_belong_new	data_count_sum
1	3	4	3
2	1	-1	4
3	1	-1	5

Table 5.5: results_middle.csv output from cluster and merge.

See Table 5.6 where the third output file result_draw.csv file is used to store color information for the draw command. Each row is one peak for a different group and contains color information for the subfigures generated by the draw command. Given that the Motif-Cluster Method generates 12 subfigures, there are 12 corresponding columns ranging from ‘draw_input0’ to ‘arr_final_draw’ to represent these 12 subfigures. The weight column is used to visualize the weight in each figure.

id	center_pos	weight	draw_input0	...	arr_final	arr_final_draw
0	60033	3.2765443279648143	0		0	0
1	60071	3.2765443279648143	0		0	0
2	60109	3.2765443279648143	0		0	0

Table 5.6: result_draw.csv output from cluster and merge.

5.6.2 Step 2: Score and Rank

The second step in the Motif-Cluster Method is the Score and Rank command. Score and Rank scores each cluster and ranks them based on their final score. The various command parameters are documented below:

```
usage: python3 MotifCluster/MotifCluster.py  calculate_score
                -input_bed -input_result -input_middle -weight_switch -step1_folder -
output_folder

 required arguments:
 -input_bed     FILENAME,  FILENAME:  your input file name(Note: step 1's sorted bed file)
                                      the file should be put into the input_files folder.
                                      (located: Motif_Cluster/input_files)
 -input_result  FILENAME,  FILENAME:  your input file name(Note: result*.csv),
                                      the file generated from step 1's output folder
                                      (located: example_output_step1_1/result*.csv)
 -input_middle  FILENAME,  FILENAME:  your input file name(Note: result_middle*.csv),
                                      the file generated from step 1's output folder
                                      (located: example_output_step1_1/result_middle*.csv)
 -weight_switch STATUS,    STATUS:    on or off,
                                      on: run the program including weight information,
                                      off: run the program without weight information
 -step1_folder  FOLDER,    FOLDER:    This folder is the output folder name from step 1. It is
                                      necessary to utilize every file within it, as well as
                                      the files in the tmp_output folder.
 -output_folder FOLDER,    FOLDER:    your customized output folder name
```

Score and Rank must be done after Cluster and Merge, as it relies on the input files result.csv and result_middle.csv generated by Cluster and Merge.

Here is an example in which we run the command based on the output of Cluster and Merge. We set the weight switch with the -weight_switch flag to indicate

the program will include weight information:

```
python3 MotifCluster/MotifCluster.py  calculate_score -step1_folder example_output_step1_1 -
input_bed human_chr12_origin.bed -input_result result.csv -input_middle result_middle.csv -
weight switch on -output folder example output step1 1a
```

There are several output files from the score and rank step which will be generated in the output folder specified by -output_folder.

- result_cluster_weight.csv
- result_score.csv

See Table 5.7 where the first output file result_cluster_weight.csv file lists cluster length information for each group. The first row is special in that it contains the total number of peaks for every row. Each subsequent row contains the peaks for a particular group. In this example, columns ‘cluster0’ to ‘cluster9’ list the weights of all peaks in the first group, and columns ‘cluster_length0’ to ‘cluster_length9’ list the weights of all peaks in the first group. Other information in the file is not useful to the user.

	cluster0	...	cluster9	cluster_length0	...	cluster_length9
0	3.0		3.0	18152		20939
1	3.0		3.0	-1		-1
2	3.0		3.0	-1		-1

Table 5.7: result_cluster_weight.csv output file from score and rank.

See Table 5.8 where the second output file "result_score.csv" is formatted such that each row contains information about the nth highest-scoring cluster. Each

column is described below:

- ‘start_pos_head_axis' and 'end_pos_head_axis' denote the start and end coordinates of the cluster, respectively.
- 'cluster_size' column indicates the number of peaks within the cluster.
- 'belong_which_class' specifies the group to which the cluster is assigned.
- 'max_weight' identifies the highest weight of the peaks in the cluster.
- 'average_gap' calculates the average gap of each cluster.
- 'score' represents the final score of the cluster.

rank_id	start_pos	start_pos_head_axis	end_pos	end_pos_tail_axis
1	96048706	96048698	96049258	96049267
2	6717043	6717035	6717299	6717308
3	82498733	82498725	82499113	82499122

cluster_size	belong_which_class	max_weight	average_gap	score
18	4	5.10902	32.470588	80.03299
8	4	8.218245	36.571429	60.059913
15	4	4.416801	27.142857	45.105223

Table 5.8: result_score.csv output file from score and rank.

5.7 Drawing Functions

5.7.1 draw

The draw command can generate an image about a region of interest and show the distinctively colored clusters.

The various command parameters are documented below:

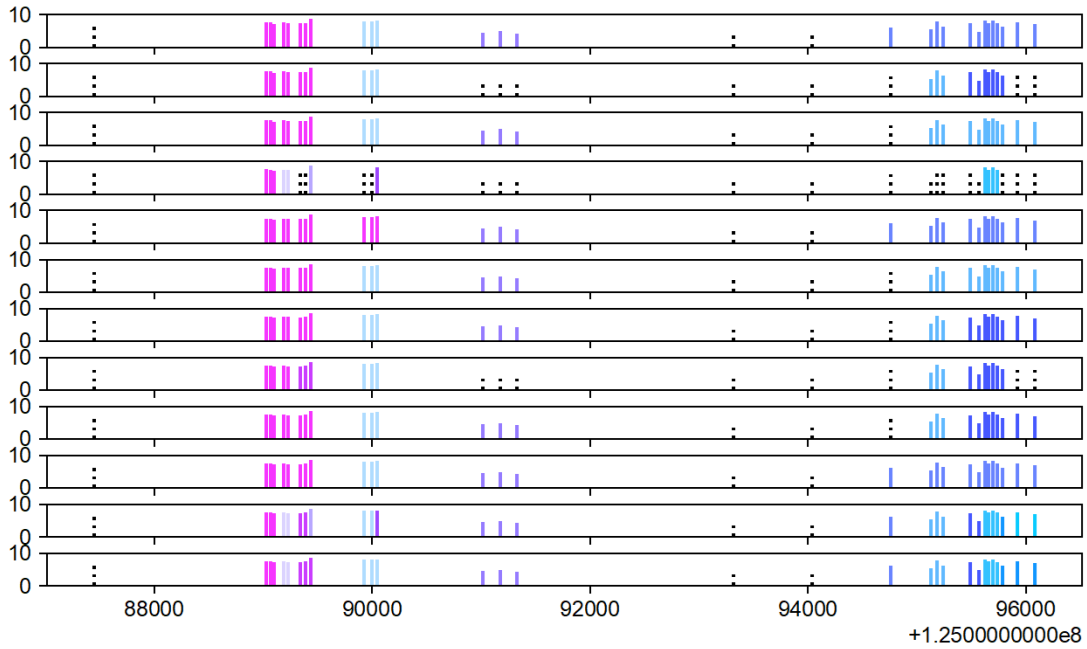


Figure 5.1: ZNF410 binding clusters on the CHD4 promoter region. (Mouse genome chr6: *125,087,000-125,097,000*).

```
usage: python3 MotifCluster/MotifCluster.py draw
                -inputbed -inputcsv -method -step1_folder -output_folder [-start] [-end]

 required arguments:
 -inputbed       FILENAME,   FILENAME:   your input file name(Note: step 1's sorted bed file)
                                         the file should be put into the input_files folder.
                                         (located: Motif_Cluster/input_files)
 -inputcsv       FILENAME,   FILENAME:   your input file name(Note: result_draw*.csv),
                                         the file generated in the output file in step1
                                         (located: example_output_step1_1/result_draw*.csv)
 -method         NUM,        NUM:        The method you use:
                                         NUM = 1: MotifCluster: with peak intensity, with cluster
merge (method d)
                                         NUM = 2: direct DBSCAN without groups (method a)
                                         NUM = 3: No peak intensity, no cluster merge (method b1)
                                         NUM = 4: No peak intensity, with cluster merge (method b2)
                                         NUM = 5: with peak intensity, no cluster merge (method c)
-step1_folder    FOLDER,     FOLDER:     This folder is the output folder name from step 1. It is
                                         necessary to utilize every file within it, as well as
                                         the files in the tmp_output folder.
 -output_folder FOLDER,     FOLDER:     your customized output folder name

 optional arguments:
 -start NUM                 NUM:        the start coordinate of drawing this input bed file
 -end   NUM                 NUM:        the end coordinate of drawing this input bed file
```

Here is an example in which we draw the graphic using the sorted bed file and the output from the cluster and merge step:

```
python3 MotifCluster/MotifCluster.py draw -step1_folder example_output_step1_1 -inputbed
human_chr12_origin.bed -inputcsv result_draw.csv -method 1 -output_folder drawing_f1 -start
6716500 -end 6724000
```

The output graphic draw_figure.pdf shows ZNF410 binding clusters on the CHD4 promoter region shown in Figure 5.1.

5.7.2 draw_rank

The draw_rank command can generate an image about the ranks (performance) of the top 100 clusters. It compares the clusters with and without noise data. The various command parameters are documented below:

```
usage: python3 MotifCluster/MotifCluster.py draw_rank
               -input1 -input2 -output_folder

 required arguments:
 -input1       FILENAME,  FILENAME:  your input file name(result_score.csv without noise)
                                     the file should be put into the input_files folder.
                                     (located: Motif_Cluster/input_files)
 -input2       FILENAME,  FILENAME:  your input file name(result_score.csv with noise),
                                     the file generated in the output file in step1
                                     (located: example_output_step1_1)
 -output_folder FOLDER,   FOLDER:    your customized output folder name
```

Here are two examples of running the command with half noise and whole noise results scores.

```
python3 MotifCluster/MotifCluster.py draw_rank -input1 result_score_chr12.csv -input2
     result_score_chr12_half_noise-0.005.csv -output_folder drawing_f2
python3 MotifCluster/MotifCluster.py draw_rank -input1 result_score_chr12.csv -input2
     result_score_chr12_whole_noise-0.01.csv -output_folder drawing_f2
```

The output file is normal_vs_noise_rank.pdf in the output folder you defined and will look similar to Figure 4.5 and Figure 4.6.

5.7.3 draw_score_size

The draw_score_size command can generate an image about corresponding cluster scores and sizes for the top 100 clusters in specific genomes. The various command parameters are documented below:

```
usage:  python3 MotifCluster/MotifCluster.py  draw_score_size
                -input -output_folder

 required arguments:
 -input       FILENAME,  FILENAME: your input file name(result_score.csv)
                                    the file generated in the output file in step2
                                    (located: example_output_step1_1)
 -output_folder FOLDER,   FOLDER:   your customized output folder name
```

In this example, we use the command on the results_score.csv file as input and output to the drawing_f3 folder:

```
python3 MotifCluster/MotifCluster.py  draw_score_size -input
    example_output_step1_1/result_score.csv -output_folder drawing_f3
```

The resulting graph image has the rank id as the x-axis, the corresponding score as the left y-axis, and the corresponding cluster size as the right y-axis shown in Figure 5.2.

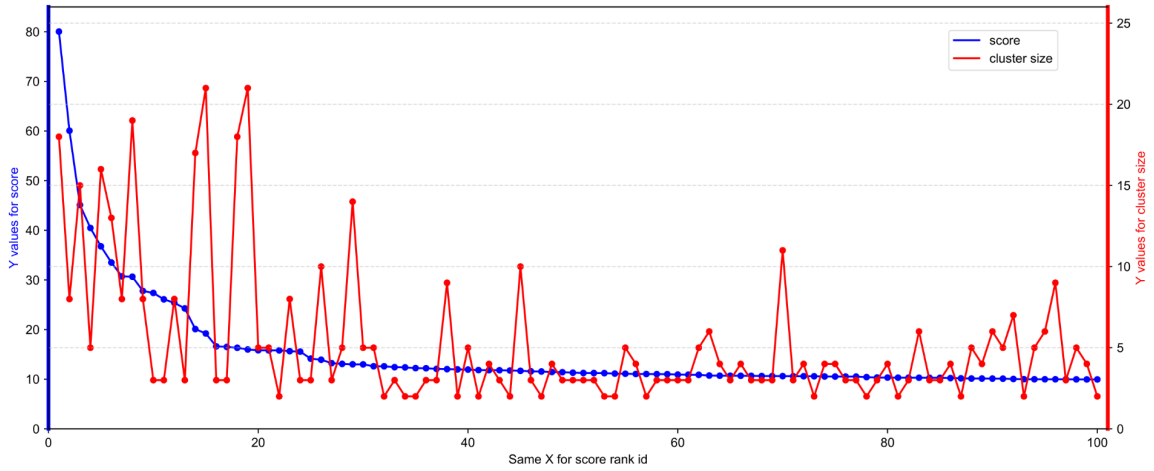


Figure 5.2: Cluster size and score for top 100 regions in chr6 on Human genome. Each color in cluster graph represents one cluster. On the curve plot, blue curve means the score and red curve means the cluster size.

5.7.4 draw_cluster_weight

The draw_cluster_weight command is used to visualize every Gaussian component whose peaks are best fitted according to how far each peak lies from its neighbor. The resulting peaks are graphed for each Gaussian component they belong to and distributed from 0 to 9 with an interval of 1. The various command parameters are documented below:

```
usage:  python3 MotifCluster/MotifCluster.py  draw_cluster_weight
                -input -output_folder

 required arguments:
 -input       FILENAME,  FILENAME: your input file folder and name(result_cluster_weight.csv)
                                   the file generated in the output file in step2
                                   (located: example_output_step1_1)
 -output_folder FOLDER,   FOLDER:   your customized output folder name
```

In this example, we use result_cluster_weight.csv as the input and output to the drawing_f4 folder:

```
python3 MotifCluster/MotifCluster.py  draw_cluster_weight -input
    example_output_step1_1/result_cluster_weight.csv -output_folder drawing_f4
```

See Figure 5.3 which consists of 10 subfigures, each corresponding to a Gaussian component. For each subfigure, the weight value is on the x-axis and the cluster count is on the y-axis.

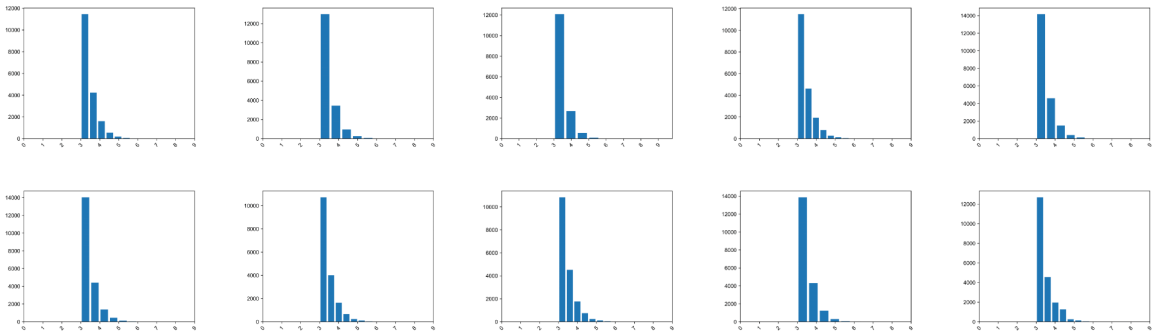


Figure 5.3: Weight distribution of peaks that best fit the corresponding 10 Gaussian components.

Here is a closer view of one of the subfigures depicting a single Gaussian component shown in Figure 5.4:

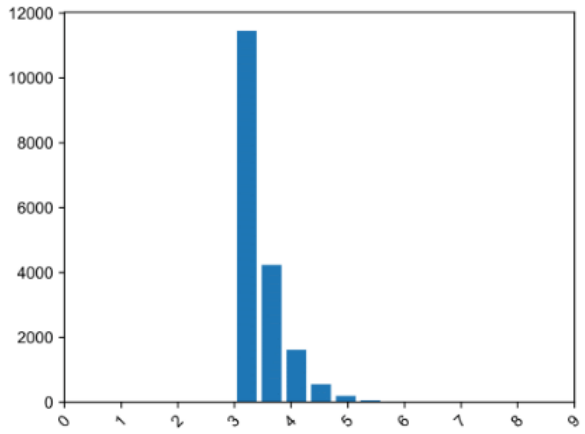


Figure 5.4: Weight distribution of peaks that best fit the corresponding nth Gaussian component.

5.7.5 draw_GMM

The draw_GMM command is used to visualize the GMM distributions of all Gaussian components. The various command parameters are documented below:

```
usage: python3 MotifCluster/MotifCluster.py draw_GMM
               -input -step1_folder -output_folder

 required arguments:
 -step1_folder  FOLDER,    FOLDER:    This folder is the output folder name from step 1. It is
                                      necessary to utilize every file within it, as well as
                                      the files in the tmp_output folder.
 -output_folder FOLDER,    FOLDER:    your customized output folder name
```

The command must be run after the calculate_score command, as it relies on temporary files generated by the score and rank step. Specifically, it expects three .npy files to be in the tmp_output folder, which is done automatically by calculate_score: GMM_covariances.npy, GMM_means.npy, GMM_weights.npy.

You may also use a different input folder instead of tmp_output. In this following example, we use the example_output_step1_1 folder.

```
python3 MotifCluster/MotifCluster.py draw_GMM  -step1_folder example_output_step1_1 -
    output_folder drawing_f5
```

Figure 3.2 is one example of the output from draw_GMM.

5.8 Methods Comparison by Motif-Cluster

Motif-Cluster can be used to apply variations of the method for comparison purposes. By default, Motif-Cluster recommends the best practice by using gap and weight information.

5.8.1 Method a: Direct DBSCAN without Groups

The cluster_and_merge_simple_dbscan command uses the direct DBSCAN method and displays the results.

The various command parameters are documented below:

```
usage:  python3 MotifCluster/MotifCluster.py cluster_and_merge_simple_dbscan
                -input -output_folder [-start] [-end]

 required arguments:
 -input        FILENAME,  FILENAME: your input file name(Note: sorted bed file),
                                    the file should be put into the input_files folder.
                                    (located: Motif_Cluster/input_files)
 -output_folder FOLDER,   FOLDER: your customized output folder name

 optional arguments:
 -start NUM               NUM: the start coordinate  of processing this input bed file
 -end   NUM               NUM: the end coordinate  of  processing this input bed file
```

In these command examples, the first command processes the whole .bed file, while the second command only processes part of the .bed file.

```
python3 MotifCluster/MotifCluster.py cluster_and_merge_simple_dbscan -input
     human_chr12_origin.bed  -output_folder other_method1
python3 MotifCluster/MotifCluster.py cluster_and_merge_simple_dbscan -input
     human_chr12_origin.bed  -output_folder other_method1  -start 6716000 -end 6724000
```

The rest of the process is similar to the Motif Cluster method, and only the command arguments differ. For the calculate_score command, you must pass the "-weight_switch off" argument:

```
python3 MotifCluster/MotifCluster.py calculate_score -step1_folder other_method1 -input_bed
     human_chr12_origin.bed -input_result result_simple_DBSCAN.csv -input_middle
     result_middle_simple_DBSCAN.csv -output_folder other_method1 -weight_switch off
```

For the draw command, you must pass the "-method 2" argument:

```
python3 MotifCluster/MotifCluster.py draw -step1_folder other_method1 -inputbed
    human_chr12_origin.bed -inputcsv result_draw_simple_DBSCAN.csv -method 2 -output_folder
    drawing_m2 -start 6716500 -end 6724000
```

5.8.2 Method Group-based, Group-based with Merge, & Group-based with Weight

The Motif-Cluster Method may be modified to use weight and cluster information, which we describe as Method b1, b2, and c shown in Table 5.9. The process and commands are the same but vary in the values for the -merge_switch and -weight_switch program arguments. The output files are of the same format as the base command, but may differ in the file name.

Method	-merge_switch	-weight_switch	Output files
b1: Group-based	off	off	result_union.csv, results_draw_union.csv, results_middle_union.csv
b2: Group-based with merge	on	off	results.csv, results_middle.csv, results_draw.csv
c: Group-based with weight	off	on	results_union.csv, results_draw_union.csv, results_middle_union.csv

Table 5.9: Overview of alternate methods according to -merge_switch and -weight_switch arguments.

Here is the general format for the commands utilized by Method b1, b2, and c:

```
python3 MotifCluster/MotifCluster.py cluster_and_merge
   -input -merge_switch [on|off] -weight_switch [on|off] -output_folder [-start -end]

python3 MotifCluster/MotifCluster.py  calculate_score
   -input_bed -input_result -input_middle -weight_switch [on|off] -step1_folder -output_folder

python3 MotifCluster/MotifCluster.py draw
   -inputbed -inputcsv -method [3|4|5] -step1_folder -output_folder [-start] [-end]
```

Method b1 uses a union-split only, without merge or weight information. To visualize the output files, you can pass the "-method 3" argument to the draw command as following:

```
python3 MotifCluster/MotifCluster.py draw -step1_folder other_method2 -inputbed
   human_chr12_origin.bed -inputcsv result_draw_union.csv -method 3 -output_folder drawing_m2 -
   start 6716500 -end 6724000
```

Figure 3.4 depicts an example of a figure generated by Method b1.

Method b2 uses a union-split with merging clusters, without weight information. To visualize the output files, you can pass the "-method 4" argument to

the draw command:

```
python3 MotifCluster/MotifCluster.py draw -step1_folder other_method3 -inputbed
    human_chr12_origin.bed -inputcsv result_draw.csv -method 4 -output_folder drawing_m3 -start
    6716500 -end 6724000
```

Figure 4.1 depicts an example of a figure generated by Method b2.

Method c uses a union-split with weight information, but without merge clusters. To visualize the output files, you can pass the "-method 5" argument to the draw command:

```
python3 MotifCluster/MotifCluster.py draw -step1_folder other_method4 -inputbed
   human_chr12_origin.bed -inputcsv result_draw_union.csv -method 5 -output_folder drawing_m4 -
   start 6716500 -end 6724000
```

Figure 4.3 depicts an example of a figure generated by Method c.

Chapter 6
More Applications: Putative Effects of PHB1 Binding

6.1 Introduction

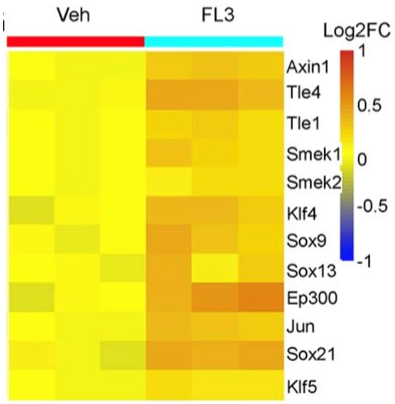
One other application of Motif-Cluster is based on the work of Jackson et al. [21], which demonstrates a model for the invivoanti-cancer mechanism of FL3 involving PHB1-induced Axin1 expression and β-catenin degradation.

FL3 is a synthetic derivative called flavaglines which is being tested in intestinal cells in pre-clinical models for the first time. Consequently, it is also the first time to verify that flavaglines target Prohibitin 1 (PHB1) as a ligand in the intestines, where PHB1 is a highly conserved protein with multiple functions.

PHB1 is as a novel factor able to bind to the (TGYCC) motif which is like the underlined portion of the p53 binding site (RRRCWWGYYY; R = A or G, W = A or T, Y = C or T) which repeats this motif with a 0-21 base pair spacer in-between.

The figure on the left is the one in the original paper which is a heat map of genes implicated in regulating Wnt/β-catenin signaling identified by RNAseq analysis using total RNA from tumoroids treated with FL3. Figure 6.1 on the right displays the chromosomes that each gene regulating Wnt/β-catenin signaling in the

left figure belongs to. It shows among the genes identified by RNAseq, the expression of Axin1 was most significantly altered by FL3 and Axin1 is in chr16.



Gene	Chromosome
Axin1	chr16
Tle1 and Tle4	chr9
Smek1 (PPP4R3A)	chr14
Smek2 (PPP4R3B)	chr2
Klf4	chr9
Sox9	chr17
Sox13	chr1
Ep300	chr22
Jun	chr1
Sox21	chr13
Klf5	chr13

Figure 6.1: The significant shown Chromosome.

ChIP assays demonstrated that PHB1 associates with a sequence: 5′GGCCTGGGCTTCGGCGCTCTGGCTCGGGCTCTGGCTC-3′ located -5668 to -5631 from the transcriptional start site (TSS) in the 5′UTR which is contained in Axin1 promoter. Our Motif-Cluster package can predict the motif in that location is most or relatively essential by ranking all the meaningful motifs in Axin1 area in

chr16.

By using the following method, we found that 5′GGCCTGGGCTTCGGCGCTCTGGCTCGGGCTCTGGCTC-3′ is truly the second place in the Axin1 area of chr16. It means our package has a good prediction in essential motif finding.

6.2 Method

The paper reports the potential RGYYY motif. R=(A,G) Y=(C,T) W=(A,T). Specifically, we finished the experiments using three steps:

- Step 1: We search for double motif RGYYYNRGYYY sites.
- Step 2: We rank the sites by the repetitive numbers and potential binding affinity of the RGYYYNRGYYY by applying the Motif-Cluster package.
- Step 3: We filtered the regions overlapping the repetitive features to get nontrivial regions.

6.3 Results

The results are summarized by Table 6.1 which ranks all predicted TFBS in the Axin1 area. Chromosome 16 is shown with parameter total-weight = 4.3, the local

rank in Axin1 area(chr16:*336440-403676*) and global rank in chr16. Then in the gene Axin1 area is in *337440-402676*. The second place of motif in the Axin1 area on the ranking list is located in *402683-402694* which is among 5′GGCCTGGGCTTCGGCGCTCTGGCTCGGGCTCTGGCTC-3′.

Local Rank in Axin1 area	Global rank in chr16	Start position	End position
1	111	Start:337736	End:337747
2	895	Start:402683	End:402694
3	1013	Start:343281	End:343297
4	1920	Start:390518	End:390529
5	2132	Start:347844	End:347854
6	2133	Start:358492	End:358502
7	2134	Start:375705	End:375715
8	2135	Start:387726	End:387736
9	2136	Start:395014	End:395024
10	3318	Start:337381	End:337391
11	3319	Start:340666	End:340676
12	3320	Start:346356	End:346366
13	3321	Start:354131	End:354141
14	3322	Start:356863	End:356873
15	3323	Start:396465	End:396475
16	5320	Start:359622	End:359633
17	5321	Start:402657	End:402673
18	5446	Start:362776	End:362786

Table 6.1: The local and global ranking of Motif-Cluster predicted clusters.

6.4 Limitations

RGYYYNRGYYY is a very general motif that occurs universally. It would be better to obtain a more specific motif from ChIP-seq. Also, we would like to link with high

throughput experimental data such as RNA-seq to test and improve our scoring function for motif occurrence.

Chapter 7
Conclusions and Future Work

7.1 Conclusions

In this thesis, the fusion of density-based clustering with flexible binding gap parameters and affinities offers a robust technique for the effective ranking of transcription factor regulatory regions, even without relying on experimental data. Motif-Cluster emerges as an advanced and adaptable tool, effortlessly utilizing this algorithm to analyze genome-wide binding site data, thereby identifying key regulatory areas for various transcription factors.

By arming researchers with this essential tool, Motif-Cluster not only promotes groundbreaking discoveries but also inspires the design of cutting-edge experiments, enhancing our comprehension of transcriptional regulation in diverse biological scenarios. It enables scientists to explore the complexities of gene expression control more thoroughly, uncovering new insights and potentially revealing regulatory patterns that might go unnoticed with traditional methods. Motif-Cluster, with its extensive and flexible structure, paves the way for new research opportunities, advancing the field of transcription factor analysis and marking a new phase of precision and depth in the study of regulatory genomics.

7.2 Shortcomings

The work have fewer cases as validation data. There are not many experimentally discovered important motif clusters.

We haven't integrated other biological data, such as ChIP-seq to improve the ranking strategy. Current work is purely computational.

We only consider one motif at a time. Currently we don't consider multiple motifs on a genome-wide analysis.

The cluster and merge step of Motif-Cluster is a bottleneck when compared to the score and rank step. Future efforts should be made to optimize the efficiency of core processes within cluster and merge through using BEDTools or pybedtools in a parallelized (multi-process) way. We can parallelize these operations on different segments of the genome by local dependency of the binding sites to improve overall performance and responsiveness of the tool.

Motif-Cluster requires slightly more computational time for the cluster and merge step compared to the score and rank phase. Future efforts should focus on optimizing the efficiency of these bottleneck functions within the clustering and merging processes by using BEDTools or pybedtools in a multi-process way [31]. Optimizing these operations by local dependency of the binding sites and conduct a genomic segmentwise parallel computing would further enhance the tool's overall

performance and expeditiousness.

7.3 Future Work (Integration with Other Tools)

Our TFBS cluster prediction and ranking tool can be used in any biological analysis. We can rank more TF regulation networks, and the network can be used in tools such as NETBID2.

This work can be improved cluster ranking scores by integrating biological experimental data.

Bibliography

[1] Suter, D.M.: Transcription Factors and DNA Play Hide and Seek, (2020).

[2] Latchman D S. Transcription factors: an overview[J]. International journal of experimental pathology, 1993, 74(5): 417.

[3] Lambert S A, Jolma A, Campitelli L F, et al. The human transcription factors[J]. Cell, 2018, 172(4): 650-665.

[4] Papavassiliou A G. Transcription factors[J]. New England Journal of Medicine, 1995, 332(1): 45-47.

[5] Maurano, M.T., Humbert, R., Rynes, E., Thurman, R.E., Haugen, E., Wang, H., Reynolds, A.P., Sandstrom, R., Qu, H., Brody, J., Shafer, A., Neri, F., Lee, K., Kutyavin, T., Stehling-Sun, S., Johnson, A.K., Canfield, T.K., Giste, E., Diegel, M., Bates, D., Hansen, R.S., Neph, S., Sabo, P.J., Heimfeld, S., Raubitschek, A., Ziegler, S., Cotsapas, C., Sotoodehnia, N., Glass, I., Sunyaev, S.R., Kaul, R., Stamatoyannopoulos, J.A.: Systematic localization of common disease-associated variation in regulatory DNA. Science (1979). 337, 1190–1195 (2012).

[6] Cartharius K, Frech K, Grote K, et al. MatInspector and beyond: promoter analysis based on transcription factor binding sites[J]. Bioinformatics, 2005, 21(13): 2933-2942.

[7] Tompa M, Li N, Bailey T L, et al. Assessing computational tools for the discovery of transcription factor binding sites[J]. Nature biotechnology, 2005, 23(1): 137-144.

[8] Bulyk M L. Computational prediction of transcription-factor binding site locations[J]. Genome biology, 2003, 5(1): 1-11.

[9] Kribelbauer, J.F., Rastogi, C., Bussemaker, H.J., Mann, R.S.: Low-affinity binding sites and the transcription factor specificity paradox in eukaryotes, (2019).

[10] Kidder, B.L., Hu, G., Zhao, K.: ChIP-Seq: Technical considerations for obtaining high-quality data, (2011).

[11] Matys, V., Fricke, E., Geffers, R., Gößling, E., Haubrock, M., Hehl, R., Hornischer, K., Karas, D., Kel, A.E., Kel-Margoulis, O.V. and Kloos, D.U., 2003. TRANSFAC®: transcriptional regulation, from patterns to profiles. Nucleic acids research, 31(1), pp.374-378.

[12] Kulakovskiy, I.V., Vorontsov, I.E., Yevshin, I.S., Sharipov, R.N., Fedorova, A.D., Rumynskiy, E.I., Medvedeva, Y.A., Magana-Mora, A., Bajic, V.B., Papatsenko, D.A. and Kolpakov, F.A., 2018. HOCOMOCO: towards a complete collection of transcription factor binding models for human and mouse via large-scale ChIP-Seq analysis. Nucleic acids research, 46(D1), pp.D252-D259.

[13] Grant, C.E., Bailey, T.L. and Noble, W.S., 2011. FIMO: scanning for occurrences of a given motif. Bioinformatics, 27(7), pp.1017-1018.

[14] Bailey, T.L., Johnson, J., Grant, C.E. and Noble, W.S., 2015. The MEME suite. Nucleic acids research, 43(W1), pp.W39-W49.

[15] Jiang, S., Mortazavi, A.: Integrating ChIP-seq with other functional genomics data. Brief Funct Genomics. 17, (2018)

[16] yao-laboratory/MotifCluster (github.com)

[17] Can CUT&RUN and CUT&Tag Replace ChIP-Seq? Genetic Engineering and

Biotechnology News. 40, (2020).

[18] Vinjamur, D.S., Yao, Q., Cole, M.A., McGuckin, C., Ren, C., Zeng, J., Hossain, M., Pinello, L. and Bauer, D.E., 2020. ZNF410 represses fetal globin by devoted control of CHD4/NuRD. Blood, 136, p.1.

[19] Sher, F., Hossain, M., Seruggia, D., Schoonenberg, V.A.C., Yao, Q., Cifani, P., Dassama, L.M.K., Cole, M.A., Ren, C., Vinjamur, D.S., Macias-Trevino, C., Luk, K., McGuckin, C., Schupp, P.G., Canver, M.C., Kurita, R., Nakamura, Y., Fujiwara, Y., Wolfe, S.A., Pinello, L., Maeda, T., Kentsis, A., Orkin, S.H., Bauer, D.E.: Rational targeting of a NuRD subcomplex guided by comprehensive in situ mutagenesis. Nat Genet. 51, (2019).

[20] Lan, X., Ren, R., Feng, R., Ly, L.C., Lan, Y., Zhang, Z., Aboreden, N., Qin, K., Horton, J.R., Grevet, J.D. and Mayuranathan, T., 2021. ZNF410 uniquely activates the NuRD component CHD4 to silence fetal hemoglobin expression. Molecular cell, 81(2), pp.239-254.

[21] Jackson, Dakota N., Kibrom M. Alula, Yaritza Delgado-Deida, Redouane Tabti, Kevin Turner, Xuan Wang, K. Venuprasad, Rhonda F. Souza, Laurent Désaubry, and Arianne L. Theiss. "The synthetic small molecule FL3 combats intestinal tumorigenesis via axin1-mediated inhibition of Wnt/β-catenin signaling." Cancer research 80, no. 17 (2020): 3519-3529.

[22] Kriegel, Hans-Peter, et al. "Density-based clustering." Wiley interdisciplinary reviews: data mining and knowledge discovery 1.3 (2011): 231-240.

[23] Khan, Kamran, et al. "DBSCAN: Past, present and future." The fifth international conference on the applications of digital information and web technologies (ICADIWT 2014). IEEE, 2014.

[24] McInnes, Leland, John Healy, and Steve Astels. "hdbscan: Hierarchical

density based clustering." J. Open Source Softw. 2.11 (2017): 205.

[25] Ankerst, M., Breunig, M.M., Kriegel, H.P. and Sander, J., 1999. OPTICS: Ordering points to identify the clustering structure. ACM Sigmod record, 28(2), pp.49-60.

[26] Reynolds, Douglas A. "Gaussian mixture models." Encyclopedia of biometrics 741.659-663 (2009).

[27] Do C B, Batzoglou S. What is the expectation maximization algorithm?[J]. Nature biotechnology, 2008, 26(8): 897-899.

[28] Yang J H, Hung J. A preliminary study of emotion recognition employing adaptive Gaussian mixture models with the maximum a posteriori principle[C]//2014 International Conference on Information Science, Electronics and Electrical Engineering. IEEE, 2014, 3: 1576-1579.

[29] Akaike, Hirotugu. "Akaike's information criterion." International encyclopedia of statistical science (2011): 25-25.

[30] Vrieze, Scott I. "Model selection and psychological theory: a discussion of the differences between the Akaike information criterion (AIC) and the Bayesian information criterion (BIC)." Psychological methods 17.2 (2012): 228.

[31] Quinlan, A.R., Hall, I.M.: BEDTools: A flexible suite of utilities for comparing genomic features. Bioinformatics. 26, (2010).

[32] https://en.wikipedia.org/wiki/DBSCAN