國 立 政 治 大 學 資 訊 科 學 系

Department of Computer Science

National Chengchi University

碩 士 學 位 論 文

Master’s Thesis

HiCSeg：針對不同樣本和物種的互動式基因體分割

HiCSeg: an interactive genome segmentation cross samples and species

指導教授：張家銘 博士

研究生：吳映函 撰

中 華 民 國 110 年 7 月

DOI:10.6814/NCCU202101389

謝辭

感謝我的指導教授張家銘老師，雖然我的大學專題跟生物資訊無關，卻還是二話不說地答應我加入實驗室這個大家庭。謝謝老師包容我在大四下學期以及研究所這兩年期間，尤其是在寫作論文上因為能力不足而需要很多指導與建議，跟著老師讓我學到了很多。感謝我的口試委員蘇家玉老師及陳世淯老師願意撥冗給予精闢的建議讓我的論文更加完整。感謝我的大學專題指導教授兼導師廖峻鋒老師，教會我一魚多吃的人生經驗以及給予我寶貴的投稿經驗，也在這個領域給予我很大的幫助。謝謝實驗室學長姐、同學、學弟妹的幫助與陪伴。謝謝芷君學姐在我研究所第一年給予我很大的幫助，謝謝育庭兩年來幫助我度過許多困難的事情，除了正事給予的協助外還一起分享與玩樂。謝謝庭軒總是在心靈上給予我支持與陪伴，想去的地方總是一起去。

最後特別感謝我的爸媽，一路支持我到現在讓我不用為了學費擔憂，雖然只能一個月回家一次仍能感受到滿滿的關心，謝謝你們總是在我身後給予我向前邁進的動力，也總是讓我不用擔心交通費可以想回家就回家。

DOI:10.6814/NCCU202101389

摘要

Hi-C的全基因組染色體接觸可用於研究染色體的更高級別組織，例如隔室或拓撲關聯域。根據哺乳動物Hi-C圖的主成分分析可得到數據中兩個區室A和B。TAD或隔室可被視為基因組的分段。通常我們會使用基因體分割進行數據壓縮，並在不同細胞類型中整理出不同的修飾。我們比較了不同解析度下的PCA結果以找出差異，然後引入ChIP-Seq數據進行進一步分析。我們還引進了其他兩種進行聚類的方法，Louvain和Leiden。它們不僅可以與PCA的結果進行比較，還可以計算出網路的相關性。此外，我們可以基於結合ChIP-Seq和Hi-C的資訊使用兩者相加及網路融合來分割基因組。

關鍵字：基因體分割、Hi-C、ChIP-Seq

DOI:10.6814/NCCU202101389

Abstract

The genome-wide chromosomal contact by Hi-C can be used to investigate the higher-level organization of chromosomes, such as compartments or topologically associating domains (TAD). Hi-C data revealed two compartments, A and B, based on principal component analysis (PCA) of Hi-C maps in mammals. TAD or compartment can be considered as a segmentation of the genome. Generally, we use genome segmentation for data compression and sort out different modifications in different cell types. We compared the PCA results in various resolutions to determine the difference and introduced the ChIP-Seq data for further analysis. We also introduce other methods to do clustering, which are the Louvain and Leiden methods. They can not only compare with the result of PCA but also figure out the correlation of networks. Furthermore, we can segment the genome based on integrated ChIP-Seq and Hi-C information using adding function and network fusion.

Keywords: Genome segmentation, Hi-C, ChIP-Seq

DOI:10.6814/NCCU202101389

Contents

DOI:10.6814/NCCU202101389

DOI:10.6814/NCCU202101389

List of Figures

DOI:10.6814/NCCU202101389

List of Tables

DOI:10.6814/NCCU202101389

1. Introduction

1.1. High-throughput Chromatin Conformation Capture (Hi-C)

Hi-C is a method based on Chromosome conformation capture(3C), and it uses massively parallel sequencing to follow the purification of ligation products. Unlike other methods based on 3C, all possible pairwise interactions genome-wide between fragments are tested (Lieberman-Aiden E et al. 2009). The workflow of Hi-C is in Figure 1.

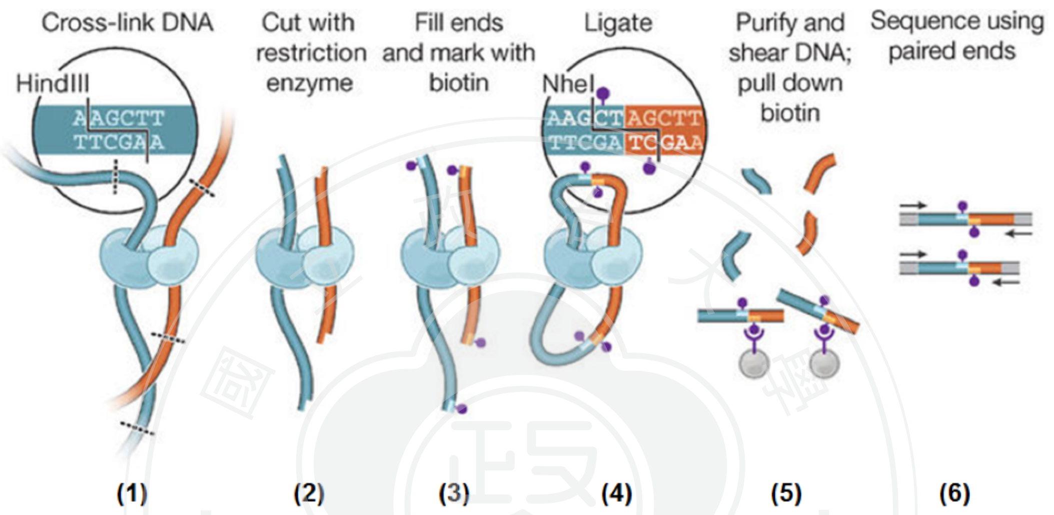

Cross-link DNA
Cut with restriction enzyme
Fill ends and mark with biotin
Ligate
Purify and shear DNA; pull down biotin
Sequence using paired ends
HindIII
AAGCTT
TTCGAA
NheI
AAGCTAGCTT
TTCGATCGAA
(1)
(2)
(3)
(4)
(5)
(6)

Figure 1. Overview of Hi-C technology procedure. (1) Sticking two pieces of DNA together using an enzyme. (2) Using restriction enzymes to cut DNA pieces. (3) Using the biotin to label and mark the DNA segments. (4) Combining DNA segments. (5) DNA is purified and sheared. Biotin junctions are isolated. (6) Using paired-end sequencing to obtain the interactions (Adapted from Van Berkum et al., 2010).

1.2. Chromatin immunoprecipitation sequence (ChIP-Seq)

ChIP-Seq is a method that can analyze genome-wide DNAs' interactions of protein. It combines chromatin immunoprecipitation with sequencing to become a practical method that identifies binding sites of proteins. The workflow of ChIP-Seq is in Figure 2.

DOI:10.6814/NCCU202101389

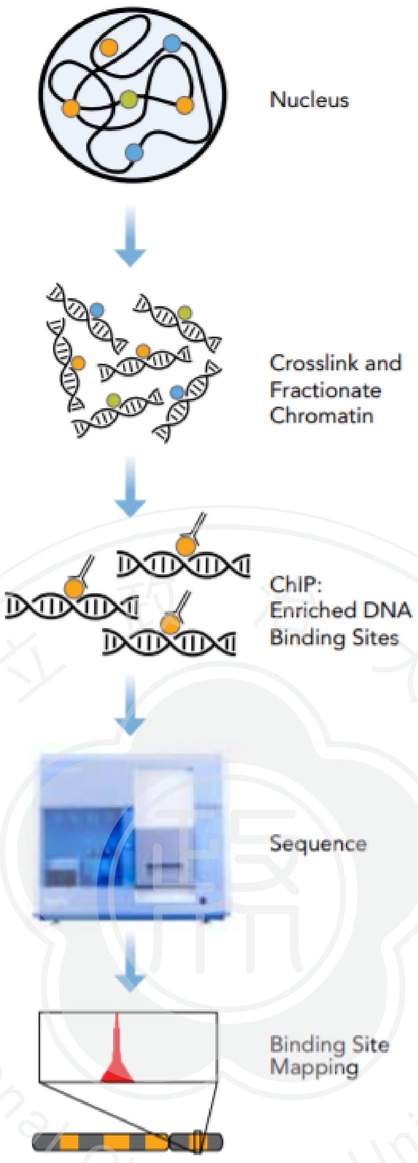

Nucleus
Crosslink and Fractionate Chromatin
ChIP: Enriched DNA Binding Sites
Sequence
Binding Site Mapping

Figure 2. ChIP-Sequencing workflow. The interactions between the protein and chromatin are crosslinked in the first. Next, distinguish the genome-wide sites with a factor or modification with specific DNA fragments co-immunoprecipitated and sequenced (Adapted from Illumina et al., 2007).

1.3. ChromHMM

ChromHMM is a helpful tool that is based on ChIP-Seq data to characterize chromatin states. There are several specific histone modifications within genomic regions. We can get chromatin state segments using ChromHMM. First, we choose the data with the filename extension of '*.bed*' or '*.bam*'. Second, we convert the input file to model learning by typing the command *BinarizeBed* or *BinarizeBam* for BAM

DOI:10.6814/NCCU202101389

or BED formats, respectively. Last, we can learn chromatin state models to generate the segment of chromatin by the command *LearnModel*. After using the ChromHMM command, it will automatically output the file as a website. If the visualization of the output file is needed, use the IGV website to get the visualization figures.

1.4. Similarity Network Fusion (SNF)

Similarity Network Fusion (SNF) is a method that can fuse two similarity networks, making the information from different data types shared (Bo Wang et al., 2014). SNF combines two networks into one network, showing every aspect of the data. Also, SNF provides a way that can integrate different data types. We can use the SNF method in five steps. First, we collect two or more original data from various data types. Second, we need to transform the original data into similarity matrices. Third, the similarity matrices are transferred into similarity networks. Fourth, we can calculate by nonlinear method and do iterations to update the networks. Finally, we can get the fused network after iterations. *SNFtool* (Bo Wang et al., 2014) is applied to combine networks quickly. We compile *SNFtool* from the source of the package.

1.5. Integration of different types of data

In the past, different genome-wide sequencing information, Hi-C, RNA-Seq, and ChIP-Seq, was integrated into several ways.

First, the Hi-C data and RNA-seq data are integrated for observing the variety during mammalian spermatogenesis. The TAD boundaries are also checked by combining the value CTCF of ChIP-Seq data. The ChIP-Seq data provided the information of CTCF, when the information of TAD and CTCF values are checked together, they become the integration of two data. The integration finally implies that TAD maintenance may be related to the stages of mitosis or meiosis (Luo et al., 2020).

Also, the accessibility of the chromatin is related to CTCF binding, using the Hi-C data, ChIP-Seq data and RNA-Seq to check the value individually to get the integration, and the integration of data implies that CTCF bindings are accessible in T-cell but inaccessible in T-ALL (Kloetgen et al., 2020).

The transcription factor binding sites in 3D space are identified by the Hi-C data and ChIP-Seq data, which is decided by whether the gene is nearby or not near the ChIP-Seq peak. When these two data are checked together, the integration of data

DOI:10.6814/NCCU202101389

is produced. The integration method also characterizes the chromatin linkage (Lan et al., 2012).

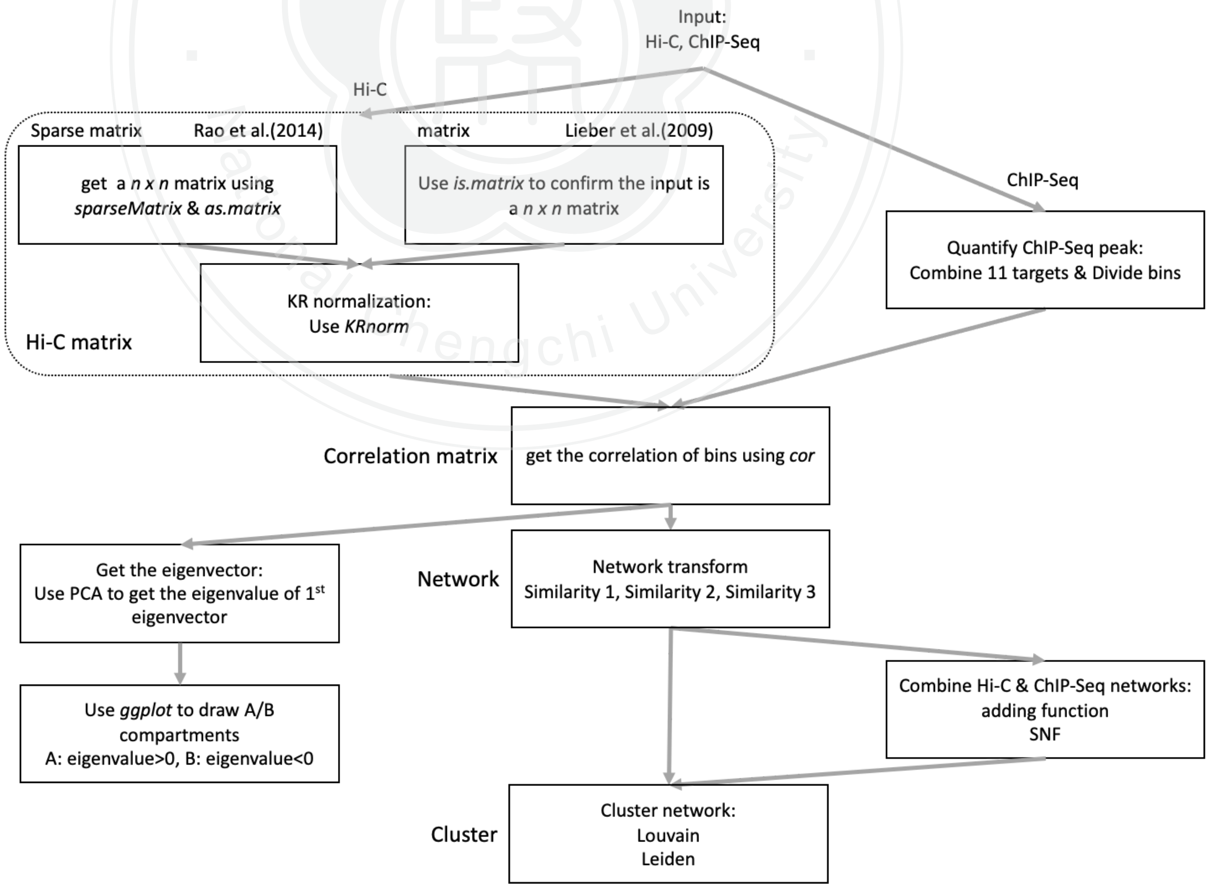
DOI:10.6814/NCCU202101389

2. Methods

2.1. Overview

We use Hi-C datasets in two cell lines: GM06990 (Lieberman-Aiden et al., 2009) and GM12878 (Rao et al., 2014). First, we need an input of a sparse matrix or matrix. Second, we do the KR normalization processing and then get the correlation matrix of the matrix. Then, we perform the PCA method on the Hi-C correlation matrix and use the eigenvector of the first principal component to decide whether a genome region is compartment *A* or *B* based on its value as positive or negative respectively.

In addition, we use ChIP-Seq data and Hi-C data to be networks. We use the Louvain and the Leiden algorithms to cluster Hi-C and ChIP-Seq networks, respectively. Plus, we introduce the adding function and the SNF method to combine Hi-C and ChIP-Seq networks. Finally, we use the Louvain and Leiden methods to cluster the fused network. The workflow of our process is in Figure 3.



Input:
Hi-C, ChIP-Seq
Hi-C
ChIP-Seq
Sparse matrix
Rao et al.(2014)
get a n x n matrix using sparseMatrix & as.matrix
matrix
Lieber et al.(2009)
Use is.matrix to confirm the input is a n x n matrix
KR normalization: Use KRnorm
Hi-C matrix
Quantify ChIP-Seq peak: Combine 11 targets & Divide bins
Correlation matrix
get the correlation of bins using cor
Get the eigenvector: Use PCA to get the eigenvalue of 1st eigenvector
Network
Network transform Similarity 1, Similarity 2, Similarity 3
Use ggplot to draw A/B compartments A: eigenvalue>0, B: eigenvalue<0
Combine Hi-C & ChIP-Seq networks: adding function SNF
Cluster
Cluster network: Louvain Leiden

DOI:10.6814/NCCU202101389

Figure 3. The workflow of our process. We use the PCA method to divide the correlation matrix of Hi-C datasets into compartments *A* and *B*. We also transform the Hi-C and ChIP-Seq correlation matrix to networks. Then, we use two ways, adding function and SNF method, to combine those two networks into one. Finally, we use Louvain and Leiden methods to cluster Hi-C, ChIP-Seq and fused networks individually.

2.2. Data Sets

2.2.1. Hi-C

We chose two datasets to analyze. The first dataset, whose cell line is GM06990, was produced in 2009, Aiden2009 (Lieberman-Aiden et al., 2009). It is publically available at https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE18199. We use the *GSE18199_binned_heatmaps* and choose 100KB and 1MB resolution.

The second dataset, whose cell line is GM12878, was produced in 2014, Rao2014 (Rao et al., 2014). It is publically available at https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE63. We use the *GSE63525_combined_intrachromosal_contact_matrices* and choose 25KB, 10KB, and 1KB resolution.

2.2.2. ChIP-Seq

We download ENCODE Histone ChIP-Seq of cell line GM12878 with genome version hg19. It is publically available at https://www.encodeproject.org/experiments/ENCSR057BWO/. We analyze 11 targets which are H3K36me3、H3K27me3、H3K9me3、H4K20me1、H2AFZ、H3K27ac、H3K4me1、H3K4me2、H3K4me3、H3K79me2 and H3K9ac.

2.3. Hi-C Contact Matrix preprocessing

There are usually two types of raw Hi-C data, so the initial process converts raw Hi-C data to a *n x n* matrix.

1) Typical contact matrix: The matrix preprocessing varies from data to data. If the input data is a *n x n* list, using the '*matrix*' function to convert it to a matrix (Lieberman-Aiden et al., 2009). When we are sure that it is a matrix format, we finish the matrix preprocessing.
2) Sparse matrix: Input Hi-C map could be a sparse matrix (Rao et al., 2014). We use the function '*sparsematrix*' in the *library*(*matrix*) to get the data and the

DOI:10.6814/NCCU202101389

'*as.matrix*' process to convert the sparse matrix to a typical matrix. The header option should be turned on during parse if the file's first row is a header.

2.3.1. Knight-Ruiz (KR) normalization processing

We usually use Hi-C data by using the *Observed/Expected* file to deal with the Hi-C bias. But, unfortunately, there is no corresponding *Observed/Expected* file in Rao2014 released data (Rao et al., 2014), so we perform the following KR normalization steps according to the procedure (Rao et al., 2014).

The format of the *RAWobserved* file is as following:

$$.\textit{RAWobserved}\text{: loci1 loci2 rawScore} \tag{1}$$

The index of the observed score is defined as dividing the specified resolution (i.e., 25000) and then plus one to the loci1 and loci1 (equation 2).

$$\begin{aligned}\text{index1} &= \text{loci1/resolution+1}\\ \text{index2} &= \text{loci2/resolution+1}\end{aligned} \tag{2}$$

Then, we can extract the KR normalized score KR1 and KR2 from *.KRnorm* file using above index1 and index2, respectively (equation 3).

$$\begin{aligned}\text{KR1} &= \text{entry[index1]}\\ \text{KR2} &= \text{entry[index2]}\end{aligned} \tag{3}$$

We can calculate a $score_{obser}$ as rawScore (the third line) in *RAWobserved* to be divided by the product of KR1 and KR2 (equation 4).

$$score_{obser} = \text{rawScore/(KR1*KR2)} \tag{4}$$

The index of the expected score, $index_{expect}$, is defined as equation (5). Then, we can extract an expected score, $score_{expect}$, using $index_{expect}$ to access the *KRexpected* file (equation 6). Finally, a normalized score is defined as the ratio between $score_{obser}$ and $score_{expect}$ (equation 7).

$$index_{expect} = \text{(loci2 - loci1)/resolution+1} \tag{5}$$

$$score_{expect} = \text{entry}[index_{expect}] \tag{6}$$

$$\text{normalized score} = score_{obser}/score_{expect} \tag{7}$$

2.4. ChIP-Seq binning

We analyzed 11 ChIP-Seq targets, which are H3K36me3、H3K27me3、H3K9me3、H4K20me1、H2AFZ、H3K27ac、H3K4me1、H3K4me2、H3K4me3、H3K79me2 and H3K9ac. We chose the *'.bed'* filename extension and split every

DOI:10.6814/NCCU202101389

target by chromosome. Taking chromosome 14 as an example, we split 11 targets into several bins at 25KB resolution. We define bins to start at 0 and increase by 25000. Since the data could begin with one bin and end at another bin, we deal with this situation using the weighted average (Figure 4). *Bin location* is the bin size we defined. *Overlap length* is the segment in which the bin overlaps the ChIP-Seq peak. The number above represents the overlap length of each element. We calculated the *peak score* of the bin by the product of length and score of every segment and divided the result by 25*k* to get the final score. Taking bin830 (bin location = 20750*k*) as an example, its overlap lengths of ChIP-Seq peak100 and peak101 are 5000 and 10000, respectively. Therefore, its peak score is 80 (= (5000*100+10000*150)/25000). Hence, when we got the score of bins of eleven targets, we could merge them into a matrix where row and column represent bin instance and ChIP-Seq peak score, respectively.

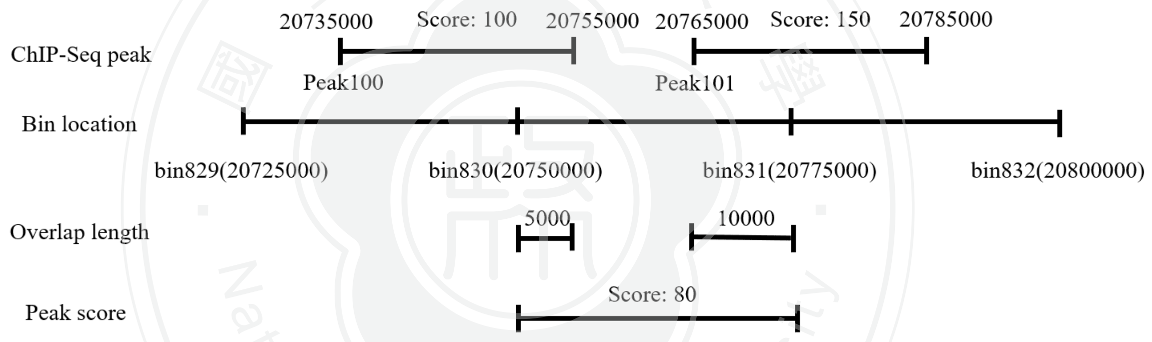


Figure 4. Quantify the ChIP-Seq of the bin using a weighted average. ChIP-Seq peak is the simulation of the *'.bed'* data with start, end, and score.

However, the result of the ChIP-Seq binning has an *NA* value, so we remove *NA*. Also, we find the value of the ChIP-Seq binning removed *NA* has a gap between them, so we use the scale function to normalize the value.

2.5. Correlation Matrix

We calculate the correlation of the normalized data in both Hi-C data and ChIP-Seq data.

In Hi-C data, we obtain more precise information. During this phase, we need to check the variance not equal to 0 by column using *which* and *apply* function in *R* at first, then using the *Pearson correlation* to calculate for all variables. However, there is a problem with the result after operating the steps above. After removing the

DOI:10.6814/NCCU202101389

columns whose variance is equal to 0, the bin index of the raw data would be reset and started by 1. Hence, we need to recover the bin index manually to make sure the bin index and the correlation value of the bins.

In ChIP-Seq data, we have already got the score of every bin we set up in 11 ChIP-Seq targets. We need to merge 11 marks to one matrix and then calculate the correlation matrix by row, representing the correlation of every bin in terms of the ChIP-Seq pattern.

2.6. Principal Component Analysis (PCA)

Principal Component Analysis (PCA) effectively reduces dataset dimensions while keeping spatial characteristics as much as possible. We can use the PCA method to get the eigenvectors of the correlation matrix from the above section. Then, we can specify the *A* or *B* compartment of the genome region based on the sign of its eigenvalue, positive for *A* and negative for *B*. *A* compartment is usually gene-rich and transcriptionally active. In contrast, the *B* compartment is usually gene-poor and transcriptionally silent (Lieberman-Aiden et al., 2009).

In the past, the input correlation matrix was in 100KB resolution, whose file size is around one hundred kilobytes (KB), the speed of running PCA is in a minute (Lieberman-Aiden et al., 2009). However, the input of a 25KB resolution matrix is around three gigabytes (GB), then the speed of running PCA is over hours. As a result, it can get a more precise contact pattern of chromosomes (Rao et al., 2014).

2.7. Network transform

1) Hi-C to a network:

We use data in three resolutions: 100KB and 1MB of Aiden2009, and 25KB of Rao2014. We transform a correlation matrix as a network graph in which a node is a bin of the matrix, and an edge weight is a similarity between two end bins. We calculate similarity in three ways:

a) Hi-C network similarity 1: We use Hi-C normalized contact score to be an edge weight.
b) Hi-C network similarity 2: We use the correlation of two bins as their connected edge weight.
c) Hi-C network similarity 3: The network degree might be negative since the correlation value ranges from -1 to 1. Because of the difficulty of the analysis, we introduce the weighted correlation network analysis to

DOI:10.6814/NCCU202101389

solve the problem. We use signed similarity to transform the correlation value:

$$S_{ij}^{signed} = 0.5 + 0.5cor(x_i, x_j)$$

where x_i and x_j represent nodes in the network. Due to the transformation of the signed similarity, the correlation value will be positive, which corresponds to the network input should be positive of the Louvain method. When using the weighted correlation network analysis, the origin correlation value is between -1 and 1, and the new correlation value is between 0 and 1. If the new correlation value is between 0 and 0.5, then it is negative originally. If the new correlation value is between 0.5 and 1, then it is positive originally. Using the value 0.5, we can shift the correlation value to get all positive values to do the next step.

2) ChIP-Seq to a network:

After getting the signed similarity correlation value matrix, we model every bin to be a node and the correlation value to be edge weight.

2.8. Hi-C and ChIP-Seq Network fusion

Some existing methods can develop genome segmentation by DNA interaction. The Hi-C data are analyzed by the PCA method, and then the eigenvectors of the first principle component are defined as compartment *A* and *B*, where the value of the *A* loci is more significant than 0. Otherwise, it is *B* (Lieberman-Aiden et al., 2009). The ChIP-Seq data are analyzed by the hidden Markov model (HMM) and are captured nucleosome-level information and identified domain-level states (Eugenio Marco et al., 2017). Markov Clustering implements the Hi-C data in human cells to segregate the regions into more detailed clusters inside the affluent areas (Lin Liu et al., 2012). Developing genome segmentation usually uses Hi-C data or ChIP-Seq data. What we want to do is use both of them to get more precise segments. We will try to combine these two different data types by the adding function and SNF.

The adding function would add two matrices, *X* and *Y,* together to be a matrix *Z*, that is, $Z_{i,j} = X_{i,j} + Y_{i,j,}$ which is $Z_{i,j}$, $X_{i,j,}$ and $Y_{i,j}$ are scores in the *i*th row and *j*th column of matrices *X*, *Y* and *Z*.

DOI:10.6814/NCCU202101389

We can use the SNF method to combine ChIP-Seq and Hi-C data (Bo Wang et al., 2014). During the step that changes the matrix, the processed Hi-C and ChIP-Seq data belong to the signed similarity correlation matrix. We take the *1-signed similarity correlation value* to be the distance matrix of two datasets. Also, the parameters of the function *affinityMatrix* we chose are the default, which is followed by empirical rule. After getting the distance matrix, we can use the function *affinityMatrix* to construct a similarity network. Next, when we successfully acquire the similarity networks by Hi-C and ChIP-Seq data, we can use the function SNF to get the fused similarity network representing the combination of Hi-C and ChIP-Seq information.

2.9. Network clustering

After network transform or fusion, we use the Louvain method in *Python* and Leiden in *Java* to cluster networks.

2.9.1. Louvain method

The Louvain method is a powerful method of community detection. It is the optimization of modularity. It decomposes the network into subunits or communities; the more closely connected parts can be regarded as a community and the relatively sparse connection between communities. The Louvain method assumes every node to be a community at first. Then, it calculates the maximum modularity of each node and its neighboring nodes. Then the following step measures the modularity of adding the node to the community of its adjacent nodes. Finally, it chooses the node that maximizes the modularity value and joins its community until no more changes occur (Blondel et al., 2008). We have compared another popular clustering algorithm, smart local moving (SLM) (Waltman et al., 2013), with Louvain. We find that Louvain gives better results and is easier to implement.

We use the *best_partition* function from the Python package *community* for the Louvain algorithm (Python-louvain, 2010). The parameter *resolution* (default = 1) affects the number of the recovered clusters. When the *resolution* is smaller, every group recovers minor data points, getting more clusters. On the contrary, when the *resolution* is larger, every cluster recovers massive data points, resulting in fewer clusters. We adjust resolution when the number of groups is too big. Therefore, we expect the result of the Louvain method is around two, which might correlate with two compartments of the PCA method.

DOI:10.6814/NCCU202101389

2.9.2. Leiden method

The Leiden method is more efficient than the Louvain method for community detection (Traag et al., 2019). It improves the disadvantages of the Louvain method. For example, two nodes in the same cluster may not have an edge between them, and the accuracy of community detection may have limitations. Furthermore, the runtime of the Leiden method is faster than the Louvain method.

We apply this method using the Java package *nl.cwts.networkanalysis* from the Github resource (CWTSLeiden, 2020). After compiling the resource in Github, we can get the *RunNetworkClustering* tool to apply the Leiden method to our data. The *resolution* dominates the fineness of the clustering, which means it controls the communities detected. When the *resolution* is smaller, it results in fewer clusters. On the contrary, when the resolution is larger, it causes more clusters. We usually use the default resolution (default = 1), which can provide better clustering results. We expect the result of the Leiden method is around two, which might correlate with two compartments of the PCA method.

DOI:10.6814/NCCU202101389

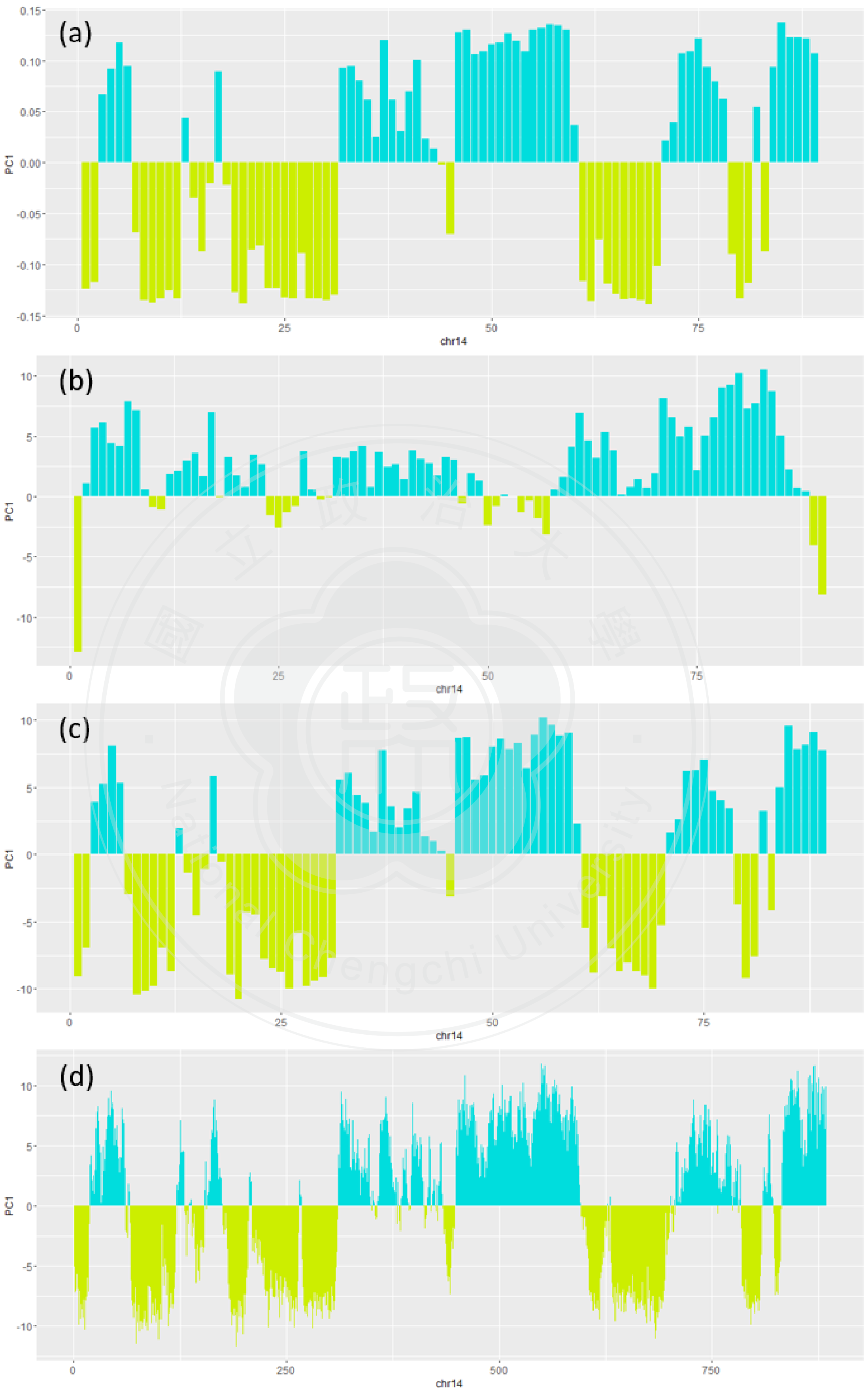
3. Results

3.1. A/B compartment reproducibility

3.1.1. KR normalization effect

Using the data given by the paper (Lieberman-Aiden et al., 2009), we try to figure out whether the result will be different with or without KR normalization. First, we use the eigenvectors given by the paper to draw the reference (Figure 5a). We find that it is sure that the way to mark the diagram will get the same chart (Figure 5c, Lieberman-Aiden et al., 2009). Then we use the contact matrix given by the article to draw two graphs. Figures 5b and 5c show without and with KR normalization, respectively. We conclude that we need to do KR normalization to get the same graph as the paper. We also use the KR normalization and get the A/B compartment in 100KB of Aiden2009 (Figure 5d) and 25KB of Rao2014 (Figure 6), finding that the KR normalization is essential to reproduce a similar segment in different resolutions.

DOI:10.6814/NCCU202101389



DOI:10.6814/NCCU202101389

Figure 5. The result of applying PCA on chromosome 14 of Aiden2009. (a) The eigenvectors are given by the paper (Lieberman-Aiden et al., 2009) in 1MB resolution. (b) We apply PCA on non-normalized data to get the eigenvector in 1MB resolution. (c) We use PCA on KR normalization data in 1MB resolution. (d) We apply PCA on KR normalization data in 100KB resolution.

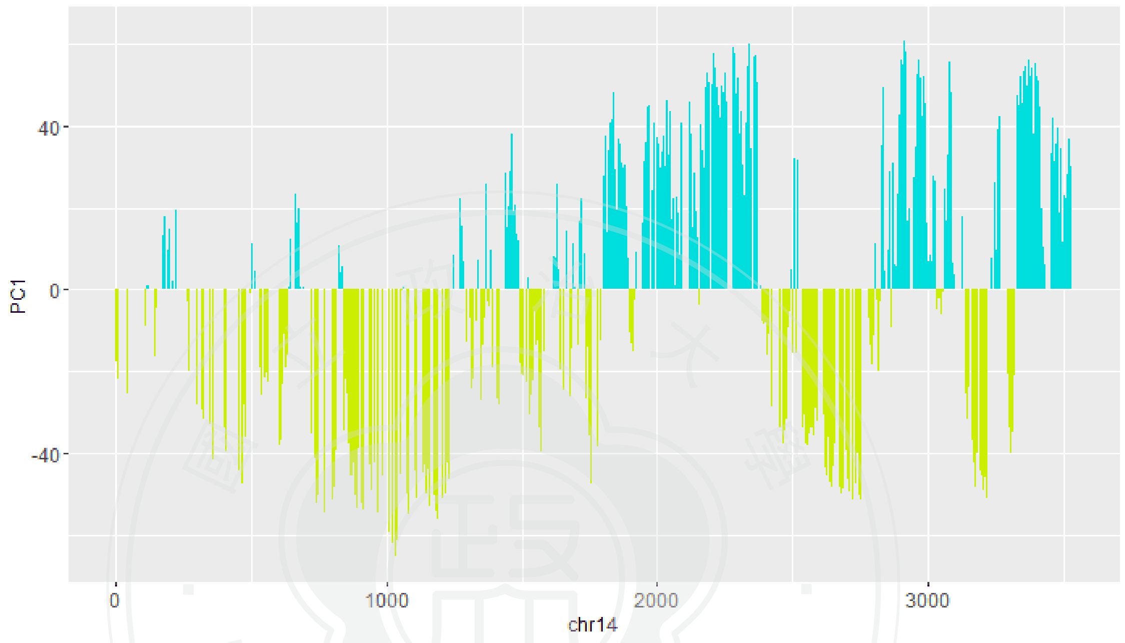


Figure 6. We apply PCA on KR normalization data in the 25KB resolution of Rao2014.

3.1.2. The resolution influence the runtime

After testing two different datasets, we record the file size and runtime (Table 1). At high resolution, which is 1KB, the file size sparse matrix is too big to become a matrix. At low resolution, which is 1MB, the file size of the matrix is relatively small, and its runtime is relatively fast.

DOI:10.6814/NCCU202101389

Table 1. The results of runtime at different resolutions and datasets. The sparse/matrix represents the file size of the sparse matrix and matrix separately.

Data set	Lieber et al.(2009)		Rao et al.(2014)		
Resolution	1MB	100KB	25KB	10KB	1KB
sparse/ matrix	NA/164K	NA/6.18M	133M/107M	317M/418M	1.3G/run out of memory
Runtime(s)	6.44	13.74	150.37	2875.39	NA

3.1.3. The explanation proportion of PCA

We usually use the PCA method to find "an index" to represent the dataset, so we only keep the first principle component. We calculate the variance by the first principle component and then define its proportion as the variance divided by the sum of variance. Thus, we calculate the cumulative ratio of the first *k* principal components, representing the percentage of the whole data explained based on the first *k* components. For $k = 1$, it will help us determine whether only using the first principal component is suitable or not. The variance of the first principle component at 100KB and 1MB resolutions are higher than other principle components of Aiden2009 (Figure 7). We find that the first principal component at 1MB resolution can explain 54% of the whole dataset (Figure 7a). However, the first principal component at 100KB can explain only around 5% of the entire dataset (Figure 7b). Also, the first principal component of Rao2014 at 25KB can explain 44% of the whole dataset (Figure 7c). The accurate number of the cumulative proportion in different resolutions is in Table 2.

Table 2. The cumulative proportion of PC1 in resolutions: 1MB, 100KB of Aiden2009, and 25KB of Rao2014.

Data set	Lieber et al.(2009)		Rao et al.(2014)
Resolution	1MB	100KB	25KB
Cumulative prop.(%)	53.96	4.78	43.76

DOI:10.6814/NCCU202101389

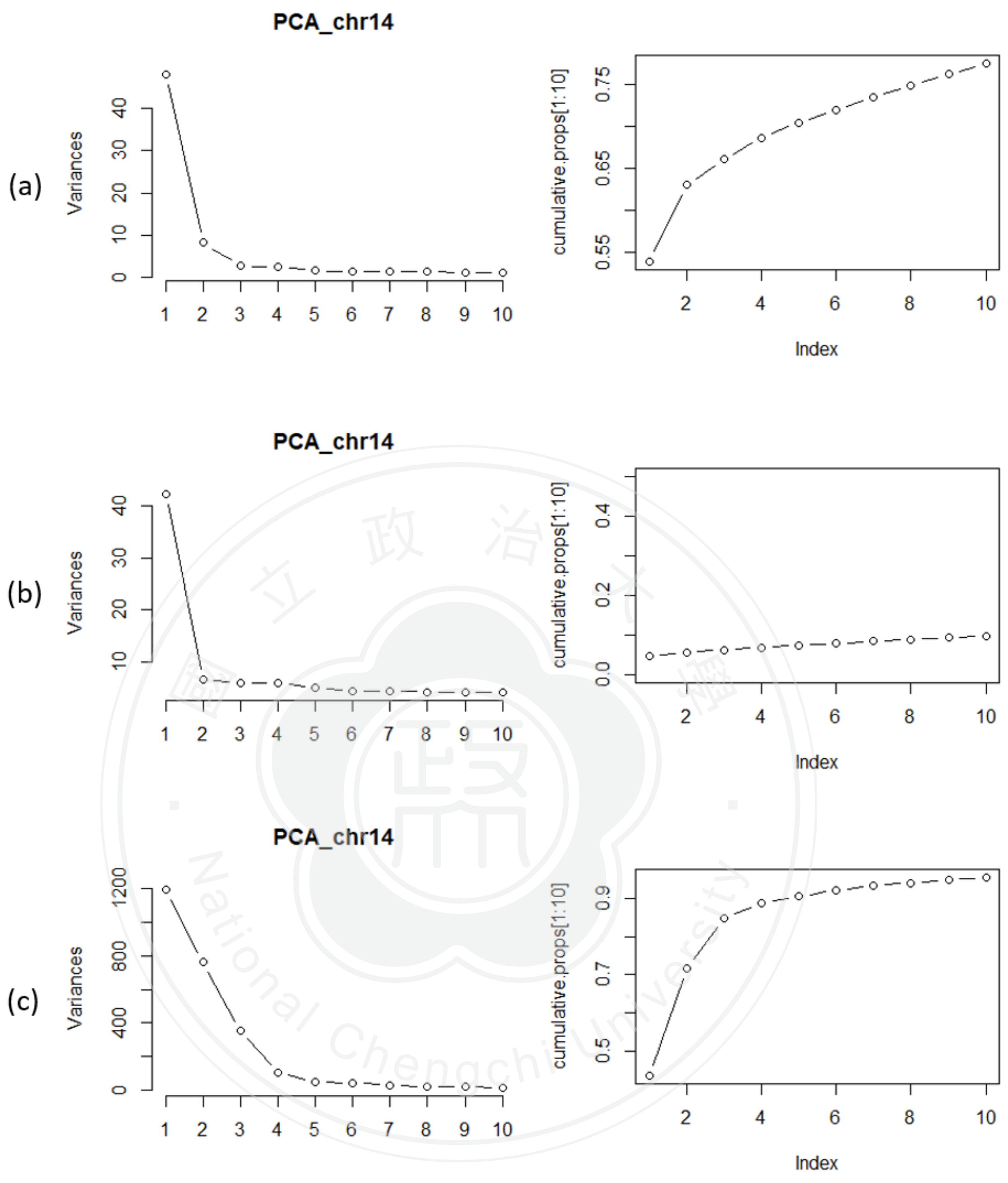


Figure 7. The cumulative proportion of the first principle component at 1MB (a) and 100KB (b) resolution of Aiden2009 and 25KB (c) resolution of Rao2014. We draw the variance and principal component figures on the left and the cumulative proportion and principal component on the right.

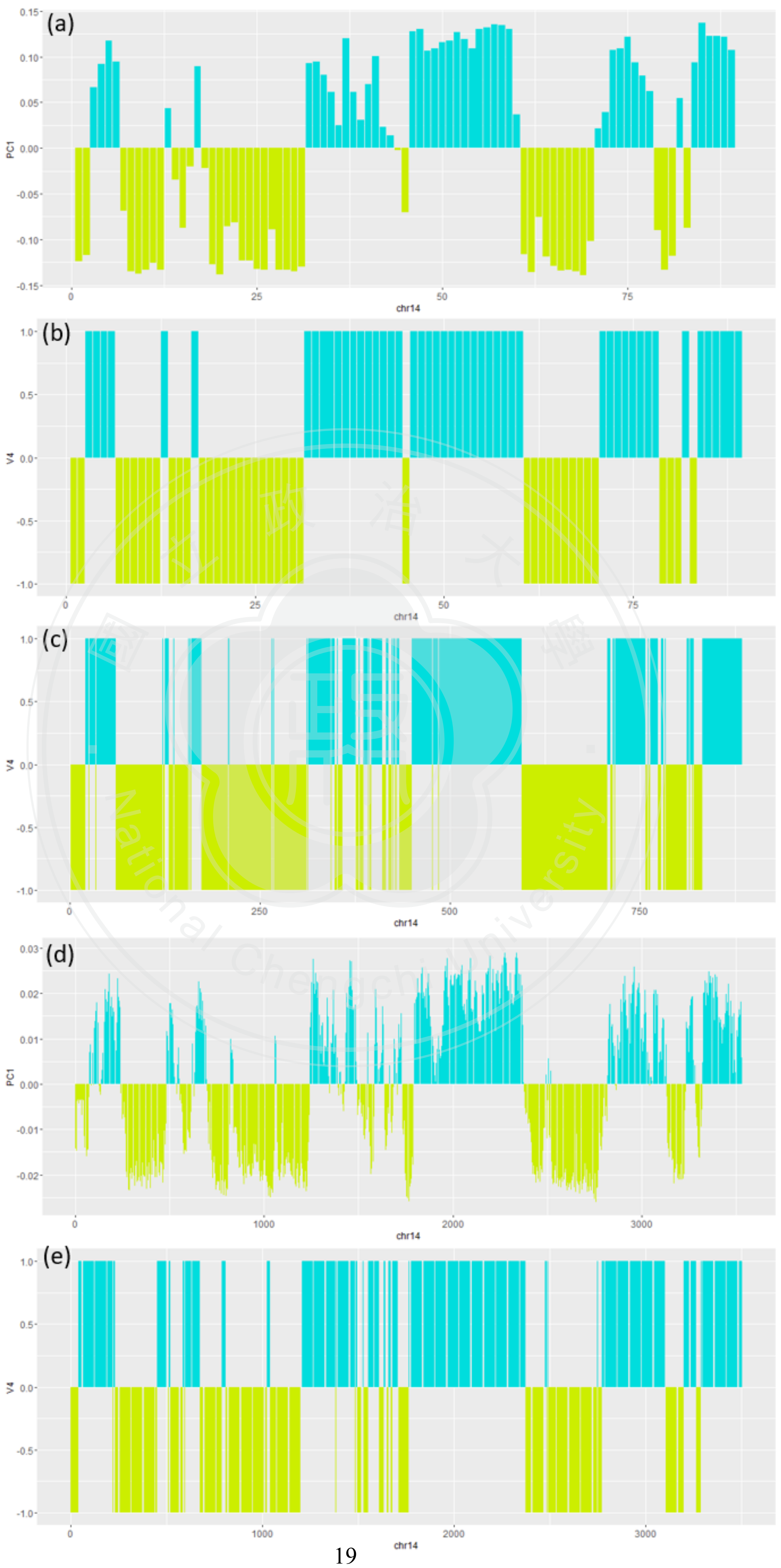
DOI:10.6814/NCCU202101389

3.2. Hi-C network cluster

3.2.1. Louvain cluster of Hi-C network

We find that the best partition is given by the default parameter, which is one. After clustering the datasets, we get two clusters of Aiden2009 (Figure 8b, 8c). Also, we get three groups of 0,1,2 of Rao2014 (Figure 8e). During this phase, the raw dataset of Rao2014 has some *NA* value, so the figure of the result has some blank. We observe the eigenvalue result of PCA, then adjust the clusters. If the result of the Louvain method is equal to two, then we keep the result (Figure 8b, 8c). If the result of the Louvain method is around two but not identical to two, we merge the cluster to compare with the result of the PCA method (Figure 8d). We combine the cluster 0 and 2 to be one cluster and keep the cluster 1 (Figure 8e). We finally get a similar trend of the PCA method (Figure 8).

DOI:10.6814/NCCU202101389



DOI:10.6814/NCCU202101389

Figure 8. Hi-C network$_{sim1}$ using the Louvain method on chromosome 14 in different resolutions compared with the PCA method result. (a) The paper gives the eigenvectors in 1MB resolution (Lieberman-Aiden et al., 2009). (b) We use the Louvain method in the 1MB resolution of Aiden2009. (c) We use the Louvain method in the 100KB resolution of Aiden2009. (d) The Juicer gives the eigenvectors of Rao2014. (e) We use the Louvain method in the 25KB resolution of Rao2014. We adjust the clusters to be two clusters to compare with the eigenvalues of the eigenvectors.

We are also interested in the matrix of the input. We test three situations using the Louvain method to determine the matrix. The first matrix is the normalized matrix after doing matrix preprocessing (Figure 9a), the second matrix is the correlation matrix of the normalized matrix after doing matrix preprocessing (Figure 9b), and the last matrix is the signed similarity correlation matrix of the normalized matrix after doing matrix preprocessing (Figure 9c). We find that the trends of these third matrices are similar, and all of them are grouped in 3 clusters. After testing the different input matrices, we can choose the signed similarity correlation matrix of the normalized matrix after doing matrix preprocessing to compare with the ChIP-Seq data.

DOI:10.6814/NCCU202101389

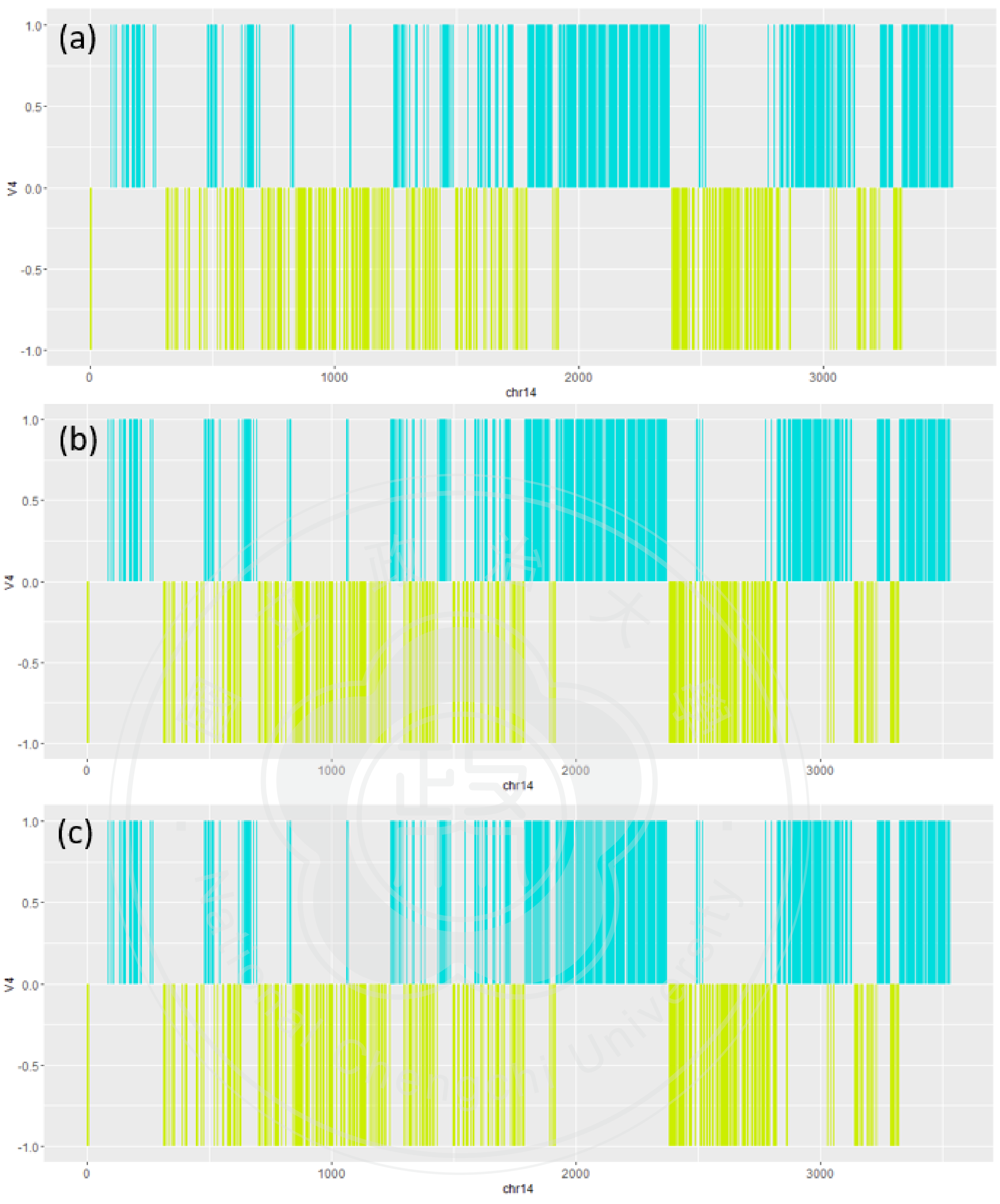


Figure 9. Hi-C network using the Louvain method on chromosome 14 by different input matrices. (a) We use the Louvain method in 25KB resolution of the datasets of normalized input data of Rao2014. (b) We use the Louvain method in 25KB resolution of the datasets of correlation of normalized input data of Rao2014. (c) We use the Louvain method in 25KB resolution of the datasets of signed similarity correlation of normalized input data of Rao2014.

DOI:10.6814/NCCU202101389

Moreover, we test the Louvain method by giving different parameters *resolution* (Table 2 Louvain Hi-C).

Table 3. The numbers of the cluster in different resolutions using the Louvain and Leiden method in the 25KB resolution of Rao2014.

	Resolutions									
methods	1	2	5	10	15	20	25	30	40	50
Louvain										
HiC.networksim1	3	1	1	1	1	1	1	1	1	1
HiC.networksim2	3	1	1	1	1	1	1	1	1	1
HiC.networksim3	3	1	1	1	1	1	1	1	1	1
ChIP-Seq.network	4	1	1	1	1	1	1	1	1	1
ChIP-Seq.network + remNA	4	1	1	1	1	1	1	1	1	1
ChIP-Seq.network + remNA + scale	4	1	1	1	1	1	1	1	1	1
Fusion.network_adding	3	1	1	1	1	1	1	1	1	1
Fusion.network_adding + remNA	3	1	1	1	1	1	1	1	1	1
Fusion.network_SNF	3223	6	1	1	1	1	1	1	1	1
Leiden										
HiC.networksim1	791	1167	2938	4050	4203	4234	4249	4260	4274	4282
HiC.networksim2	4	1135	2441	2621	2678	2694	2702	2706	2712	2713
HiC.networksim3	4	2718	2725	2725	2725	2725	2725	2725	2725	2725
ChIP-Seq.network	4	3914	4292	4292	4292	4292	4292	4292	4292	4292
ChIP-Seq.network + remNA	4	3875	4254	4254	4254	4254	4254	4254	4254	4254
ChIP-Seq.network + remNA + scale	4	3418	3869	4129	4241	4254	4254	4254	4254	4254
Fusion.network_adding	3	4068	4292	4292	4292	4292	4292	4292	4292	4292
Fusion.network_adding + remNA	3	4030	4254	4254	4254	4254	4254	4254	4254	4254
Fusion.network_SNF	3	3001	4158	4283	4291	4292	4292	4292	4292	4292

3.2.2. Cluster Hi-C data by Leiden

We use the Hi-C data network to be input and use the Leiden method in Java. We find that the best partition is given by the parameter *resolution,* which is one. After clustering the datasets, we get four clusters of 0,1,2 and 3 of the datasets in Rao2014 (Figure 10b). First, we observe the eigenvalue result of PCA, then adjust the clusters. Finally, we merge the group to compare with the result of the PCA method (Figure 10). We combine the cluster 1, 2 and 3 to represent positive value, keeping the cluster 0 representing negative value.

DOI:10.6814/NCCU202101389

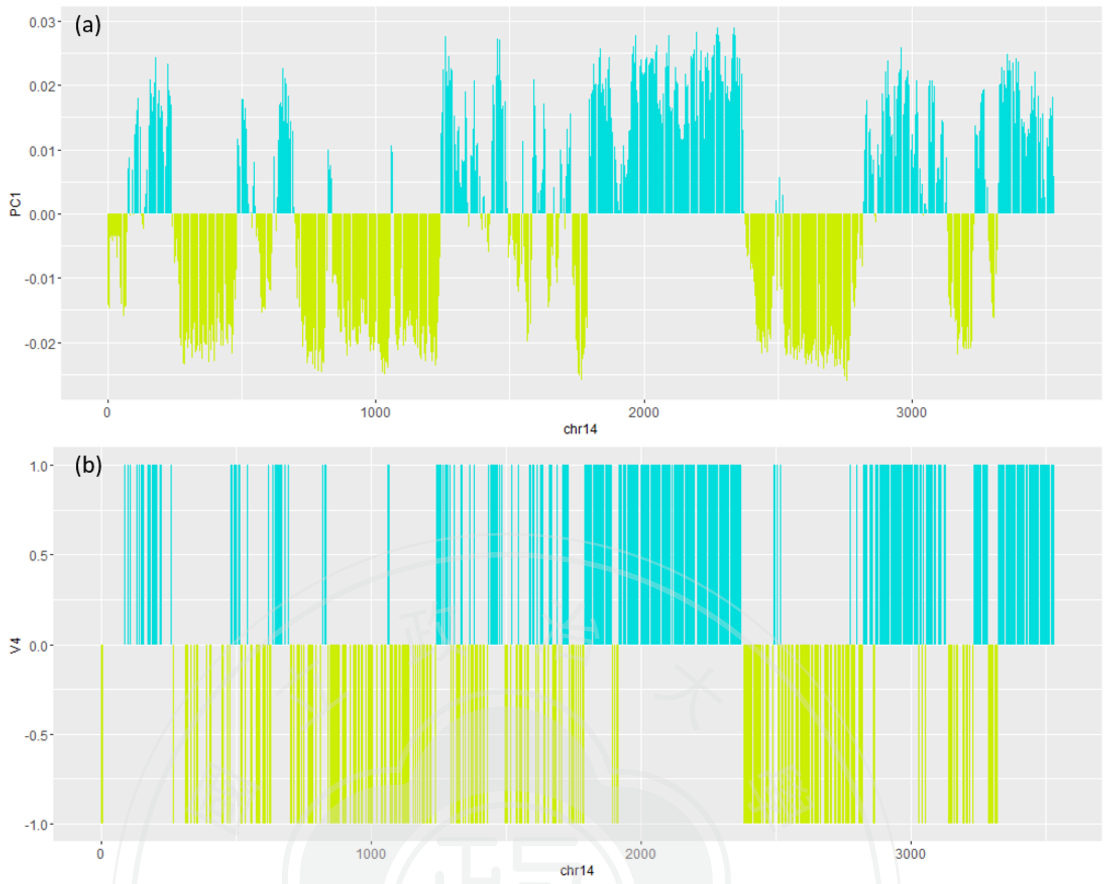


Figure 10. Hi-C datasets using the Leiden method on chromosome 14. (a) The Juicer gives the eigenvectors of the datasets of Rao2014. (b) We use the Leiden method in the 25KB resolution of Rao2014.

Moreover, we test the Leiden method by giving different parameters *resolution* (Table 3 Leiden Hi-C).

Also, we compare the Louvain and Leiden methods with their runtime in different situations (Table 4). We choose the parameter resolution, which results in the number of clusters closed to two, and records two methods' runtime. We find that although the result of the Louvain and Leiden are pretty similar, the runtime of the Leiden method is relatively shorter than the Louvain method.

DOI:10.6814/NCCU202101389

Table 4. The runtime of using the Lovain and Leiden method in second.

methods	**Louvain**	**Leiden**
HiC.networksim1	37	2
HiC.networksim2	25	2
HiC.networksim3	20	1
ChIP-Seq.network	451	6
Chip-seq.network + remNA	362	5
Chip-seq.network + remNA + scale	216	4
Fusion.network_adding	289	4
Fusion.network_adding + remNA	460	4
Fusion.network_SNF	220	4

3.2.3. The consistency between network cluster and PCA decomposition

We use *Pearson Correlation* to check the consistency between PCA decomposition (HiCmatrix, Figure 11~12) and network clusters generated by the Louvain and Leiden method. If the number of groups is larger than two, we combine them into two sets and compare the consistency. First, we compare the result of the Louvain method, and we use three different inputs, which are normalized matrix (HiCnetworksim1, Figure 11), the correlation matrix of the normalized matrix (HiCnetworksim2, Figure 11), and the signed similarity of the correlation matrix of the normalized matrix (HiCnetworksim3, Figure 11). We find that the signed similarity of the correlation matrix of the normalized matrix is slightly less than the normalized matrix, but its correlation is still higher than 0.85.

Next, we test the Leiden method in two network similarities: the correlation matrix of the normalized matrix (HiCnetworksim2, Figure 12) and the signed similarity of the correlation matrix of the normalized matrix (HiCnetworksim3, Figure 12). While network similarity 1 would come out huge clusters (791 clusters), so we do not include it into consistency checking. However, we find that the signed similarity of the correlation matrix of the normalized matrix using the Leiden method is slightly higher than the Louvain method.

DOI:10.6814/NCCU202101389

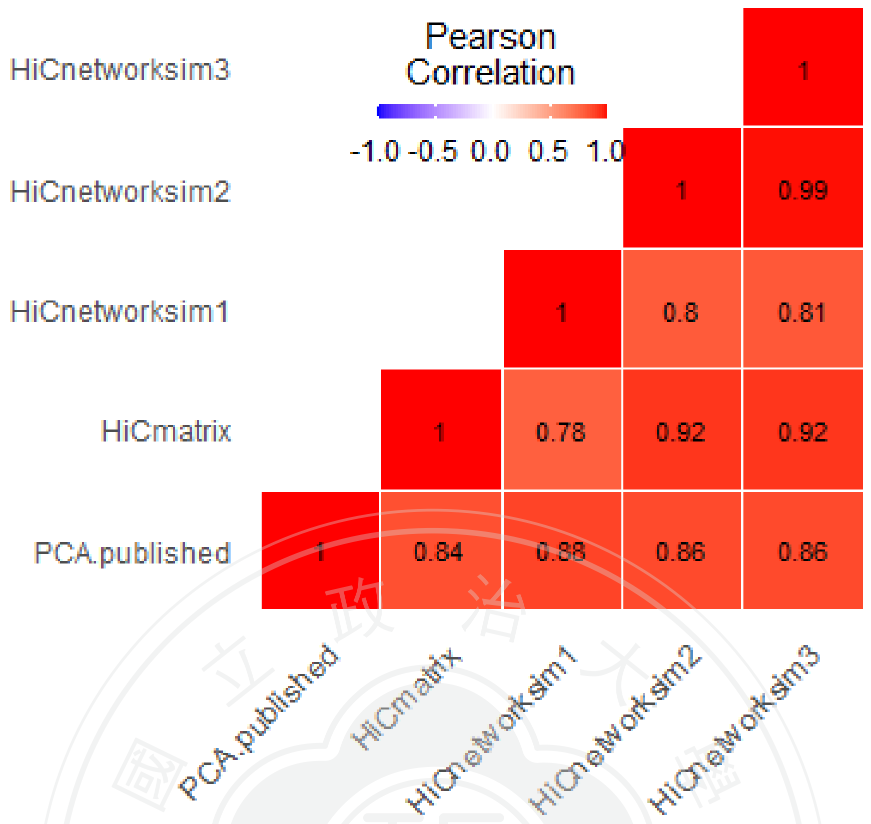


Figure 11. The cluster correlation of different Hi-C networks using the Louvain method.

DOI:10.6814/NCCU202101389

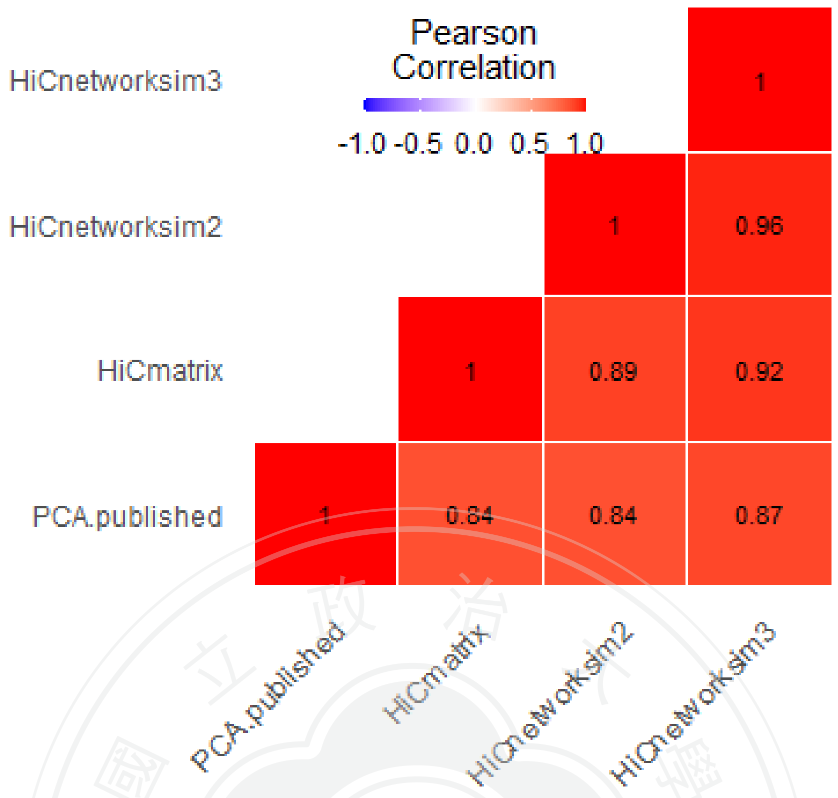


Figure 12. The cluster correlation of different Hi-C networks using the Leiden method.

3.3. ChIP-Seq data

3.3.1. Cluster ChIP-Seq data by Louvain

We use the ChIP-Seq data network to be input and use the Louvain method in the Python package *community*. We find that the best partition is given by the default parameter, which is one. We check whether clusters of Louvain are consistent with ChIP-Seq pattern, that is, activation or repression marks (Table 5) are grouped together or not based on the Louvain clusters. We use *pheatmap* (R library) to analyze the ChIP-Seq network cluster. We find that active marks and the repression targets group together separately (Figure 13a). We find that removing *NA* is similar to one of the original networks, making sense (Figure 13b). Finally, we find that scaling is a bit weird which combines activation targets and repression targets together (Figure 13c).

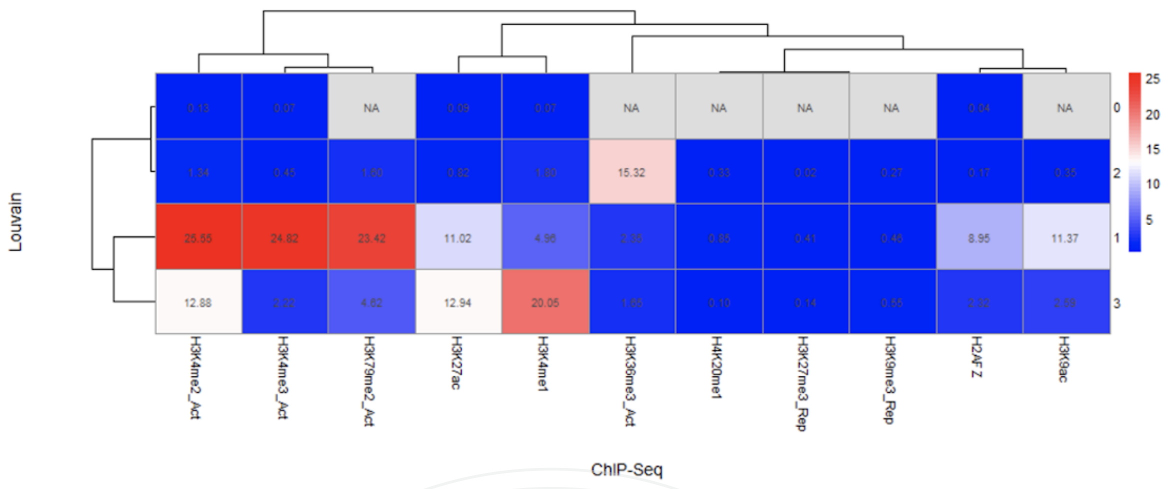
DOI:10.6814/NCCU202101389

Table 5. The role in transcription of the histone modification.

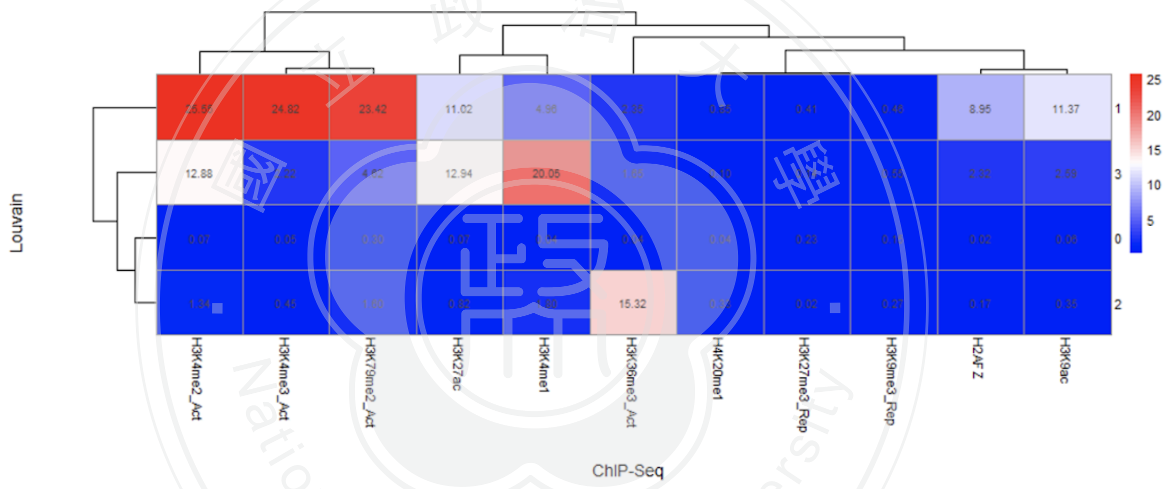
Modification	Role in transcription	Modification site
Acetylation	Activation	H3(K9, K14, K18, K56). H4(K5, K8, K12, K16). H2B(K6, K7, K16, K17). (Strahl and Allis, 2000)
Methylation	Activation	H3(K4me2, K4me3, K36me3, K79me3) (Strahl and Allis, 2000)
Methylation	Repression	H3(K9me3, K27me3) and H4(K20me3) (Balazs, 2014)
Phosphorylation	Activation	H3(S10) (Strahl and Allis, 2000)

DOI:10.6814/NCCU202101389

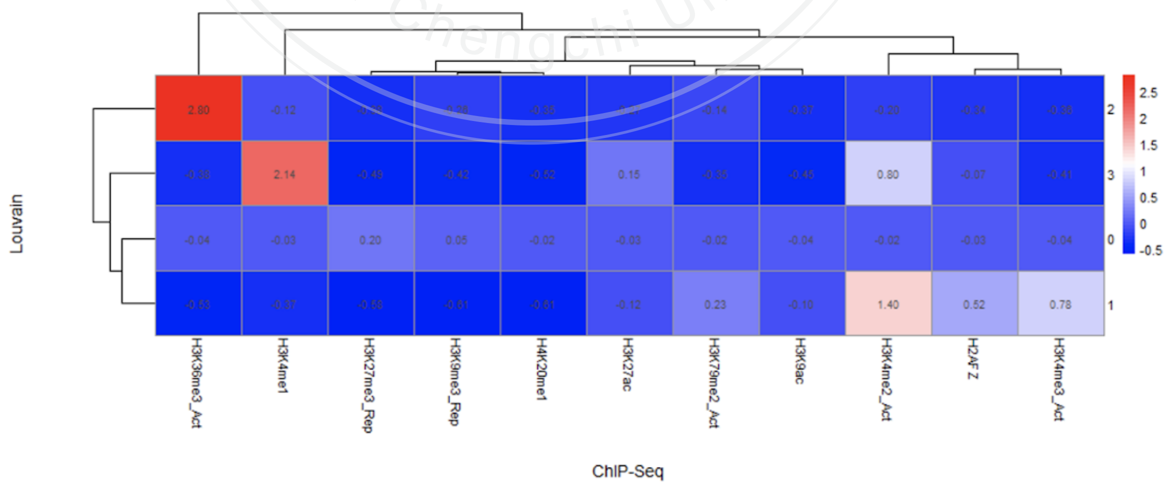
(a) A: 1+3 , B: 0+2



(b) A: 1+3 , B: 0+2



(c) A: 2+3, B: 0+1

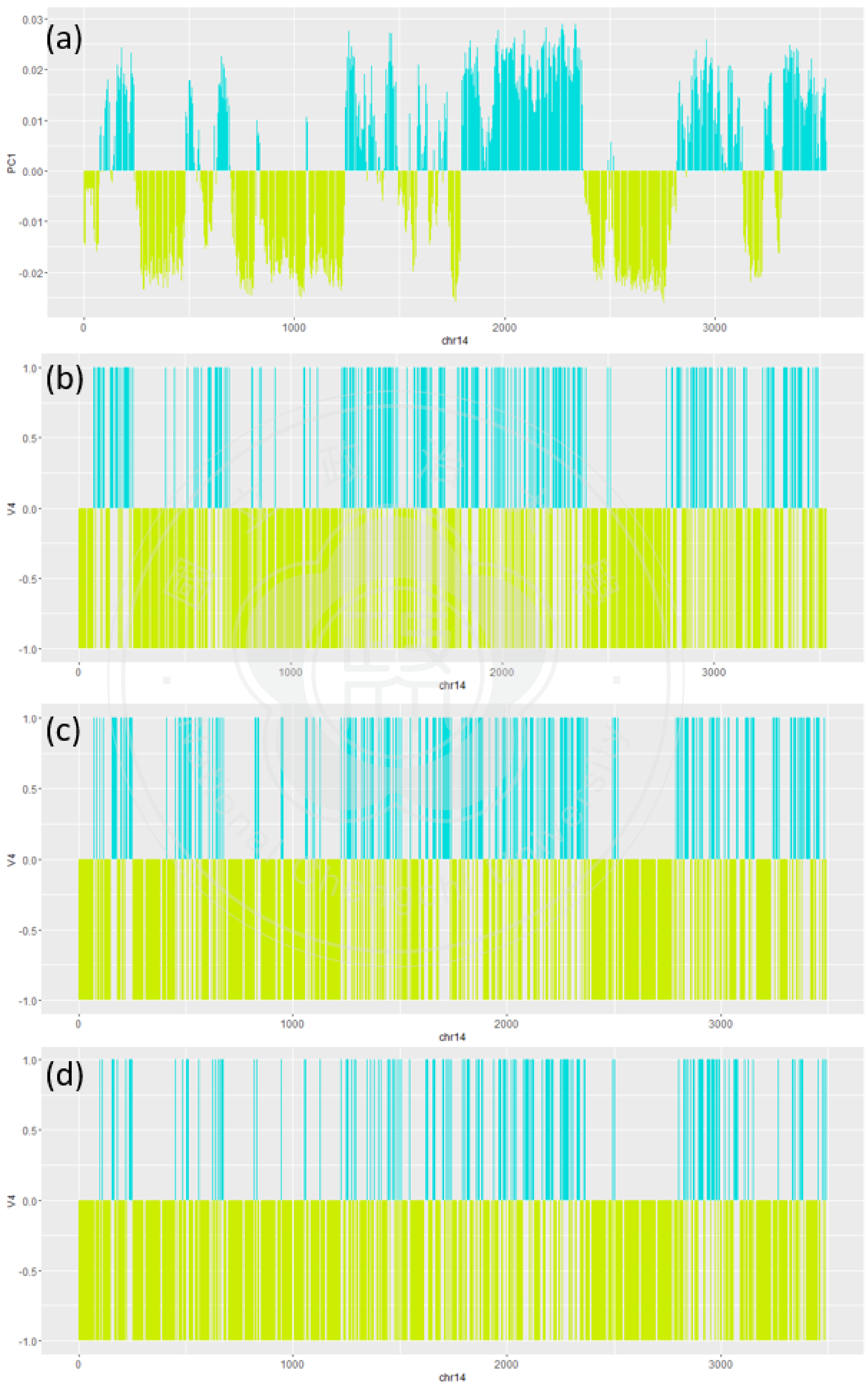


DOI:10.6814/NCCU202101389

Figure 13. The pheatmap results of the ChIP-Seq network using the Louvain method. (a) ChIP-Seq network. (b) ChIP-Seq network removes *NA*. (c) ChIP-Seq network removes *NA* and uses scale, where A and B correspond to A/B compartments.

Since the ChIP-Seq patterns make sense, we combine the clusters into two clusters according to the *pheatmap* result (Figure 13). The inputs ChIP-Seq network and ChIP-Seq removed *NA* combine cluster 1 and 3 to be positive, and cluster 0 and 2 to be negative (Figure 14b~c). The input ChIP-Seq network removed *NA* and used scale to combine cluster 2 and 3 to be positive, and cluster 0 and 1 to be negative (Figure 14d). We find that they have a similar trend to the PCA method. Moreover, we test the Louvain method by giving different parameters *resolution* (Louvain -> ChIP-Seq, Table 3).

DOI:10.6814/NCCU202101389



DOI:10.6814/NCCU202101389

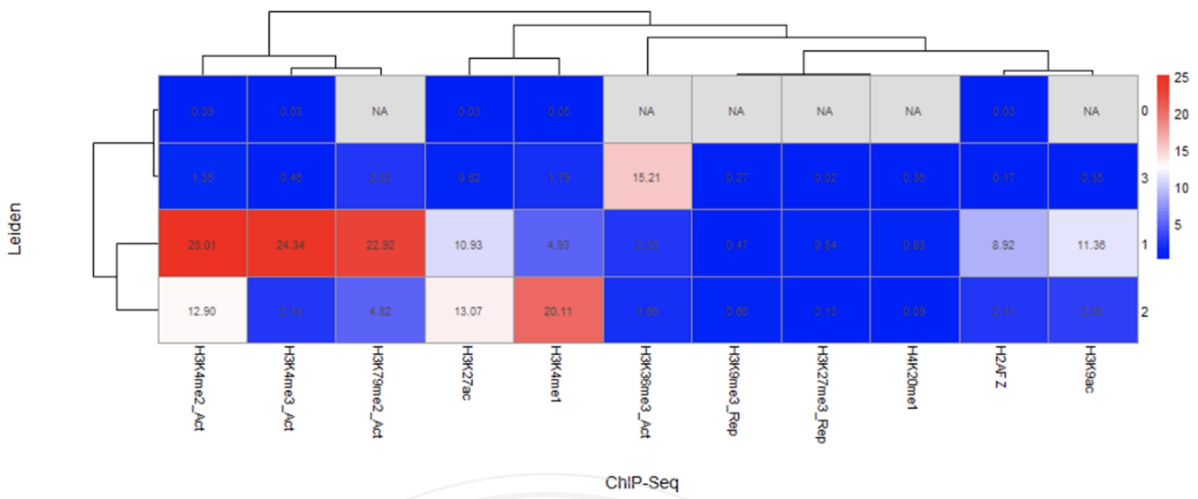
Figure 14. ChIP-Seq data using the Louvain method on chromosome 14 in 25KB resolution. (a) The eigenvectors are given by the Juicer (Eigenvector, 2017). (b) The combined clusters of the ChIP-Seq network. (c) The combined clusters of the ChIP-Seq network removed *NA*. (d) The combined clusters of the ChIP-Seq network removed *NA* and used scale.

3.3.2. Cluster ChIP-Seq network by Leiden

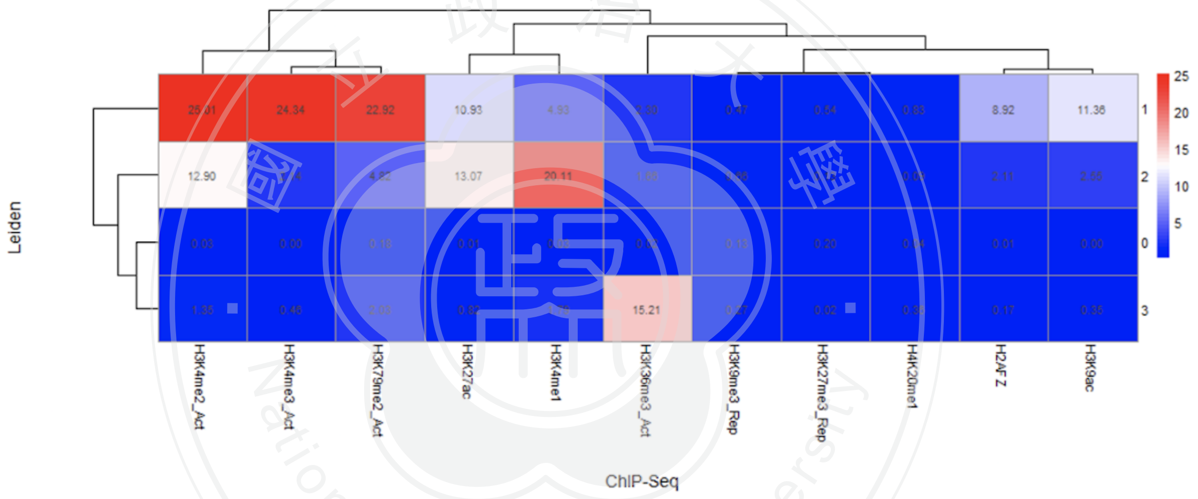
We use the ChIP-Seq data network to be input and use the Leiden method in Java. We find that the best partition is given by the parameter *resolution,* which is one. We also use the *pheatmap* function to figure out whether the result of clusters using the Leiden method makes sense or not. We also tested three inputs: ChIP-Seq network, ChIP-Seq network removed NA, and ChIP-Seq network removed NA and used scale (Figure 15). We find that the active targets are together, and the marks in repression are also together in Figure 15a. Finally, the result is weird and combines activation targets and repression targets together (Figure 15c).

DOI:10.6814/NCCU202101389

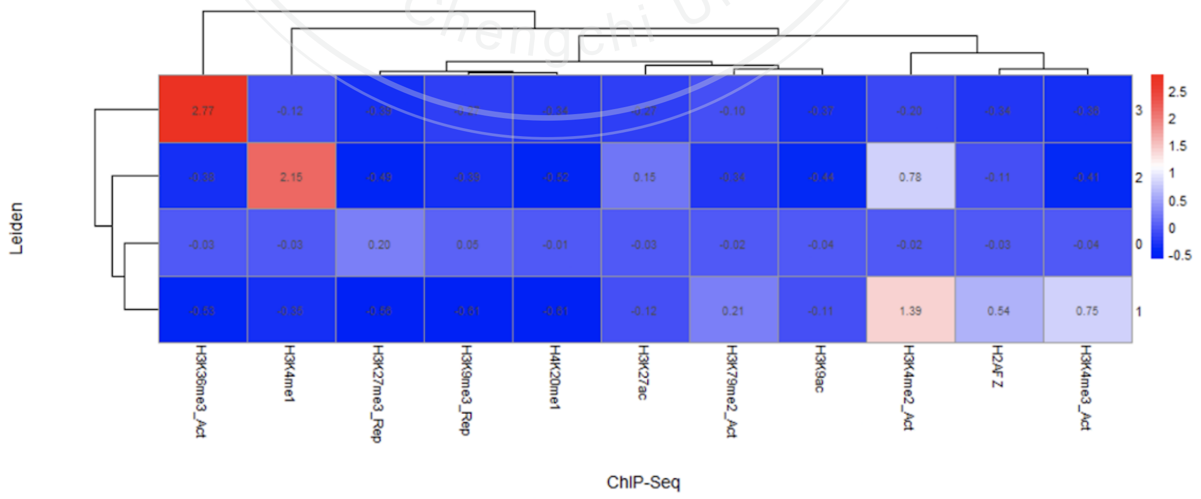
(a) A: 1+2, B: 0+3



(b) A: 1+2, B: 0+3



(c) A: 2+3, B: 0+1

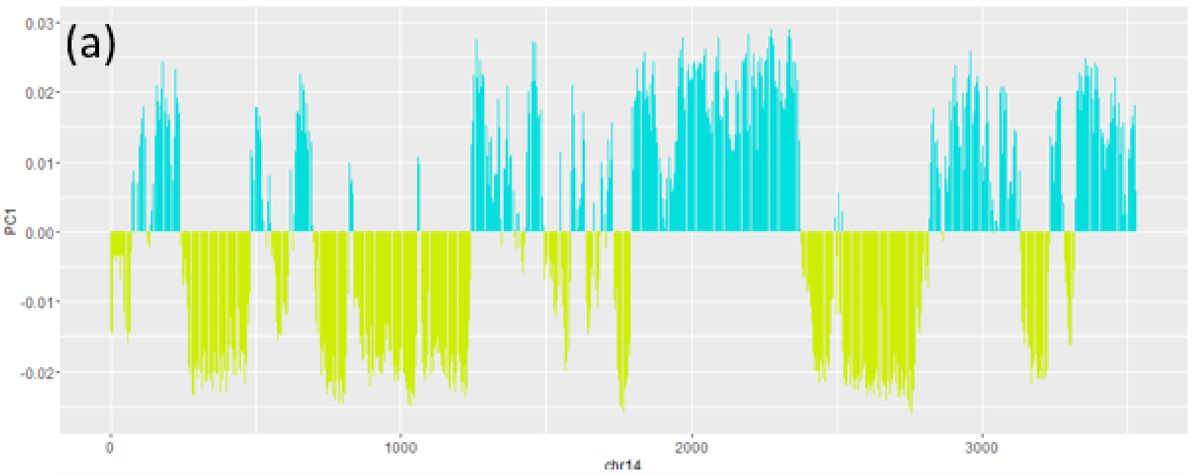


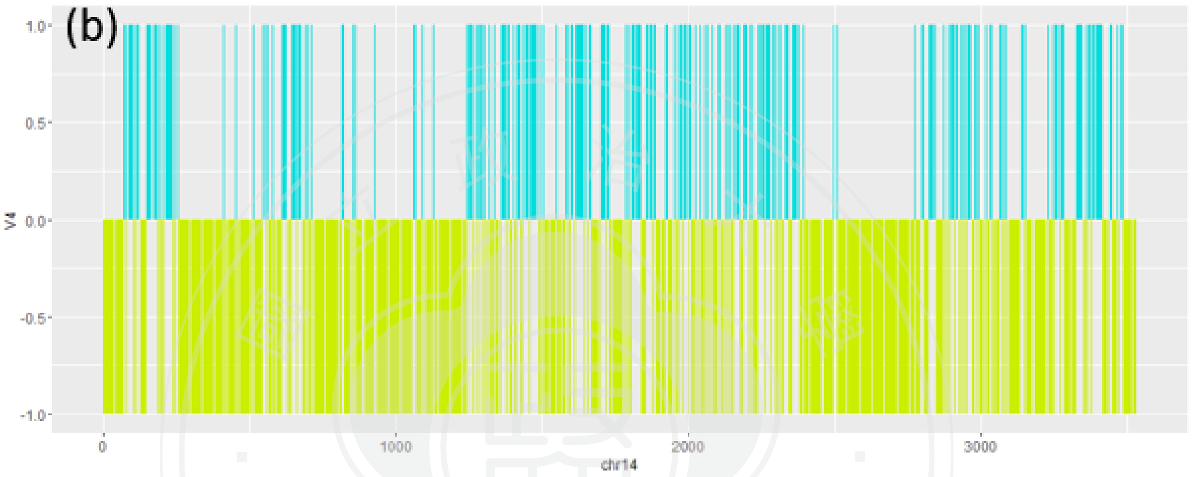
DOI:10.6814/NCCU202101389

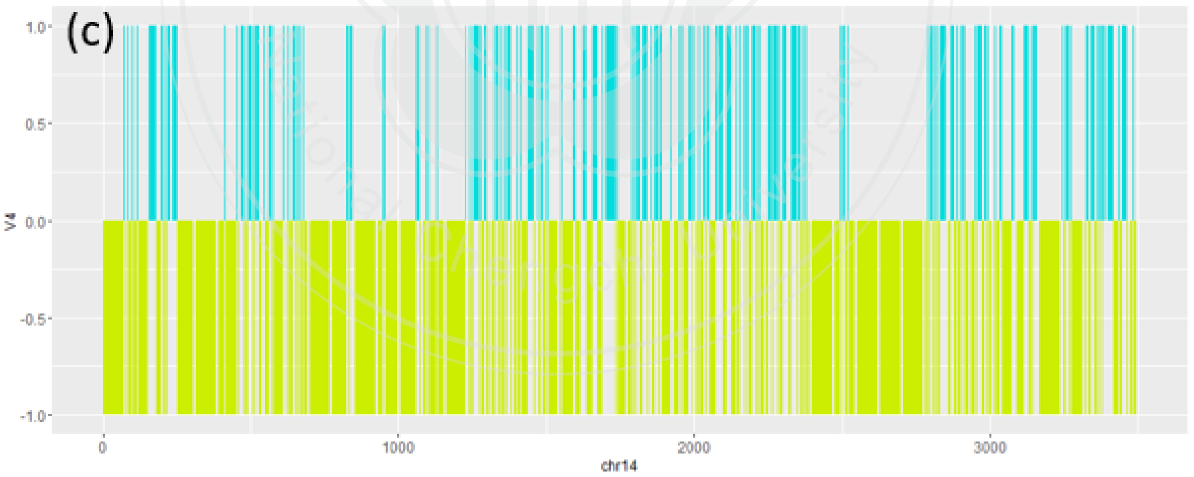
Figure 15. The pheatmap results of the ChIP-Seq network using the Leiden method. (a) ChIP-Seq network. (b) ChIP-Seq network removes *NA*. (c) ChIP-Seq network removes *NA* and uses scale, where A and B correspond to A/B compartments.

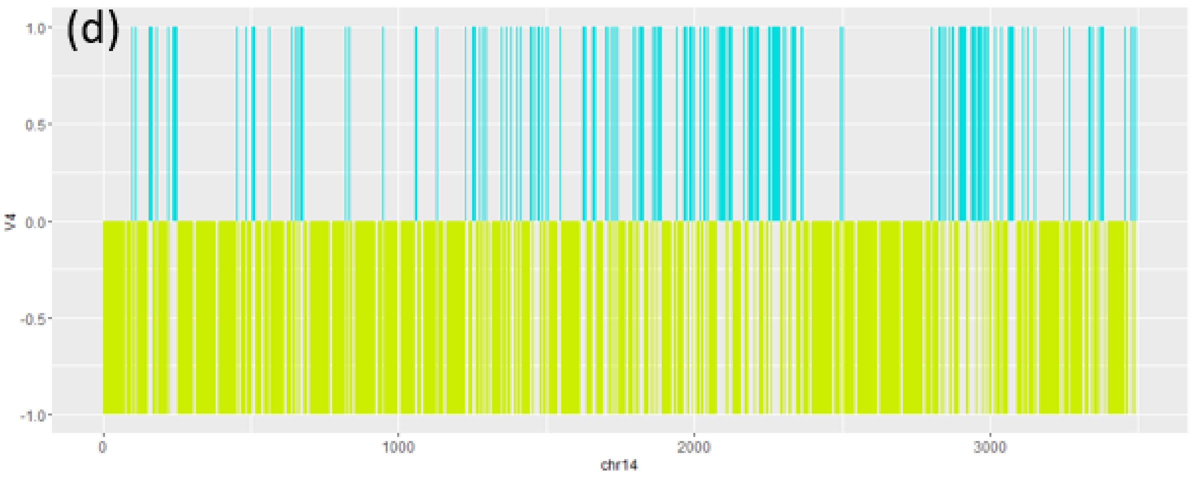
Since the ChIP-Seq patterns make sense, we combine the clusters into two clusters according to the *pheatmap* result (Figure 15). The inputs ChIP-Seq network and ChIP-Seq removed *NA* combine cluster 1 and 2 to be positive, and cluster 0 and 3 to be negative (Figure 16b~c). The input ChIP-Seq network removed *NA* and used scale to combine cluster 2 and 3 to be positive, and cluster 0 and 1 to be negative (Figure 16d). We find that they have a similar trend to the PCA method.

DOI:10.6814/NCCU202101389









DOI:10.6814/NCCU202101389

Figure 16. ChIP-Seq datasets using the Leiden method on chromosome 14. (a) The eigenvectors are given by the Juicer (Eigenvector, 2017). (b) The combined clusters of the ChIP-Seq network. (c) The combined clusters of the ChIP-Seq network removed *NA*. (d) The combined clusters of the ChIP-Seq network removed *NA* and used scale.

We also compare the cluster correlation of ChIP-Seq data using the Louvain and Leiden method. We find that the PCA and ChIP-Seq data with around 0.45 (Figure 17) is lower than the correlation of the PCA and Hi-C data, whose values are about 0.85 (Figure 11~12). Since the correlation is lower, it means that ChIP-Seq data capture some different information from Hi-C data.

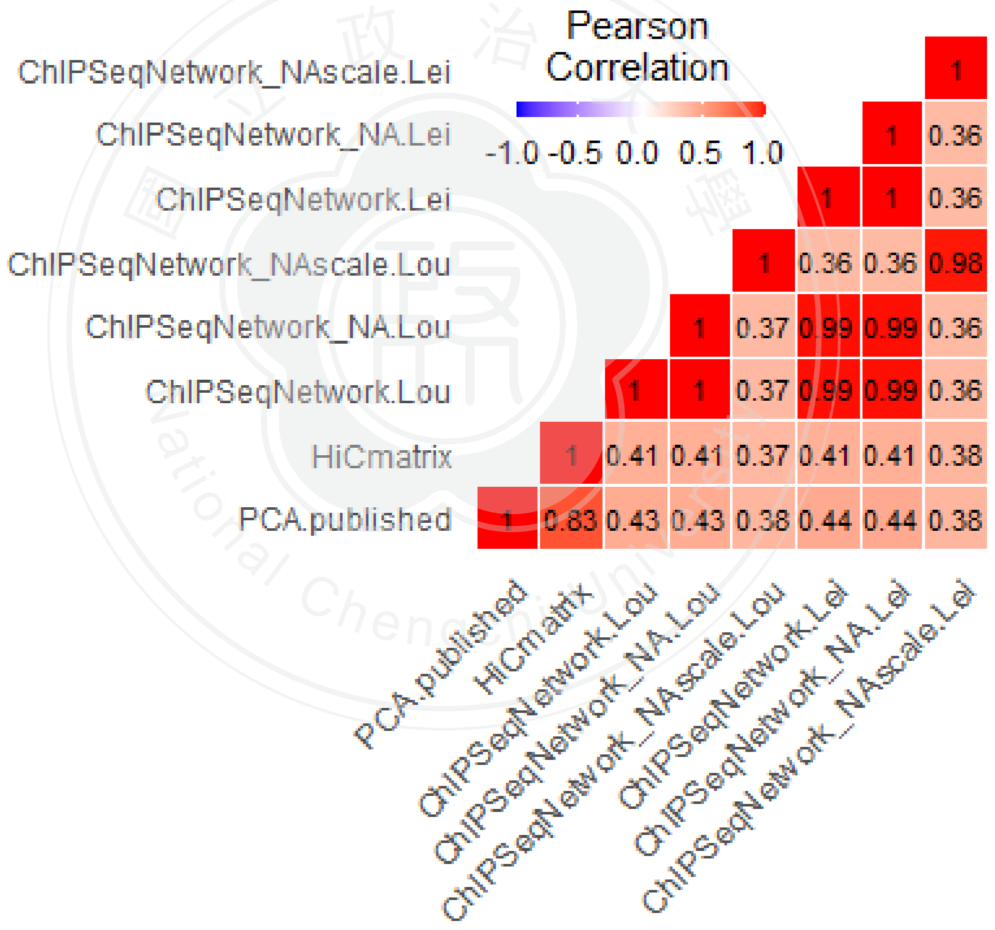


Figure 17. The correlation of the ChIP-Seq cluster results after doing the Louvain and Leiden method.

Moreover, we test the Leiden method by giving different parameters *resolution* (Table 3 Leiden ChIP-Seq).

DOI:10.6814/NCCU202101389

3.3.3. Further analysis of the A/B compartment

Although we can reproduce the A/B compartment by combining the result of the Louvain or Leiden method to two clusters (Figure 11~12), we are also interested in whether the clusters could be segmented better. The A/B compartment can be further segmented into more subcompartments (Rao et al., 2014). Although the runtime of the Leiden method is relatively shorter than the Louvain method (Table 4), we find that the result using the Louvain method of *hclust* to H3K27me3 and H3K9me3 is more suitable (Figure 18). As a result, We use our *pheatmap* result of the ChIP-Seq network removed *NA* using the Louvain method (Figure 13b) to compare with the subcompartments (Figure 19). The ChIP-Seq data patterns show that our clusters 0, 1, 2, 3 can correspond to subcompartment B1, A1, B4, A2 separately. As a result, we offer consequences where using the Louvain method to cluster can be more accurate than only segmenting to the A/B compartment.

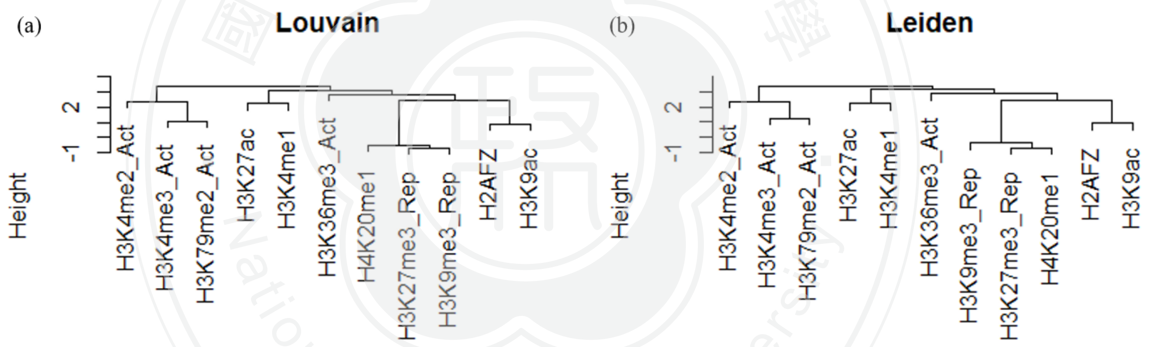
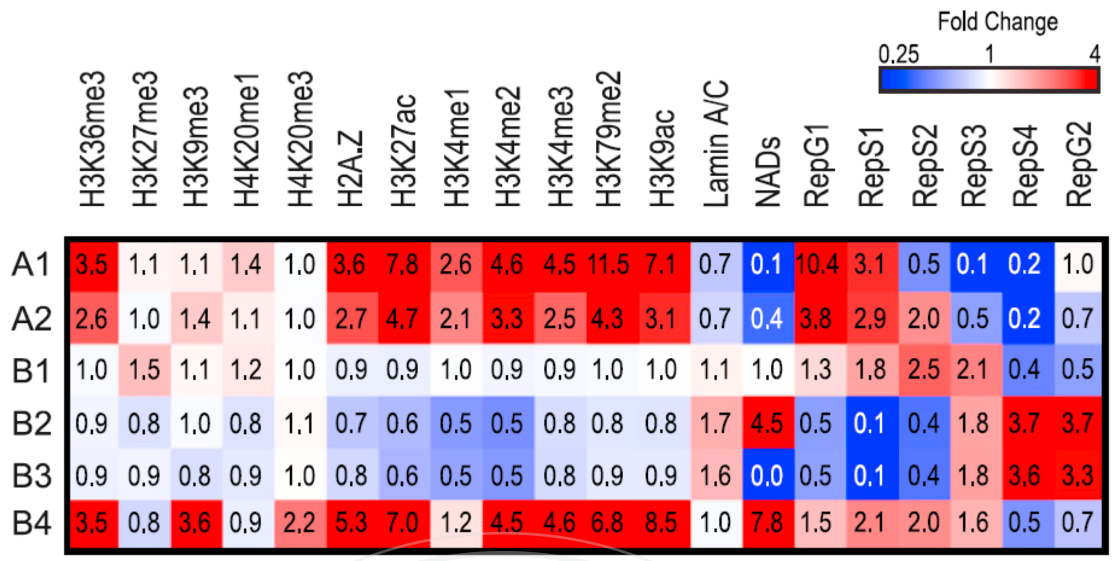


Figure 18. The *hclust* of the ChIP-Seq network removes *NA*. We use *log* distance to show the clusters of the Louvain (a) and Leiden (b) clearer.

DOI:10.6814/NCCU202101389



	H3K36me3	H3K27me3	H3K9me3	H4K20me1	H4K20me3	H2A.Z	H3K27ac	H3K4me1	H3K4me2	H3K4me3	H3K79me2	H3K9ac	Lamin A/C	NADs	RepG1	RepS1	RepS2	RepS3	RepS4	RepG2
A1	3.5	1.1	1.1	1.4	1.0	3.6	7.8	2.6	4.6	4.5	11.5	7.1	0.7	0.1	10.4	3.1	0.5	0.1	0.2	1.0
A2	2.6	1.0	1.4	1.1	1.0	2.7	4.7	2.1	3.3	2.5	4.3	3.1	0.7	0.4	3.8	2.9	2.0	0.5	0.2	0.7
B1	1.0	1.5	1.1	1.2	1.0	0.9	0.9	1.0	0.9	0.9	1.0	1.0	1.1	1.0	1.3	1.8	2.5	2.1	0.4	0.5
B2	0.9	0.8	1.0	0.8	1.1	0.7	0.6	0.5	0.5	0.8	0.8	0.8	1.7	4.5	0.5	0.1	0.4	1.8	3.7	3.7
B3	0.9	0.9	0.8	0.9	1.0	0.8	0.6	0.5	0.5	0.8	0.9	0.9	1.6	0.0	0.5	0.1	0.4	1.8	3.6	3.3
B4	3.5	0.8	3.6	0.9	2.2	5.3	7.0	1.2	4.5	4.6	6.8	8.5	1.0	7.8	1.5	2.1	2.0	1.6	0.5	0.7

Figure 19. Six contact patterns of ChIP-Seq data (Adapted from Rao et al., 2014).

3.4. Correlation of Hi-C clusters and ChIP-Seq clusters

We get clusters individually when we finish doing the Louvain method on the Hi-C network and ChIP-Seq network. We can use the Pearson correlation value to figure out the similarity of our results. The results of the Pearson correlations of Hi-C clusters using the Louvain method by three different input methods are higher than 0.86 (Figure 20HiC). Hence, we are sure that we can reproduce the A and B compartments. On the other hand, the Pearson correlations of ChIP-Seq sets using the Louvain method by three different input methods results are relatively low (Figure 20ChIPSeq). Although the value of ChIP-Seq is not that high, it represents ChIP-Seq data and Hi-C datasets provide different information. Therefore, we can try to fuse two datasets which are Hi-C and ChIP-Seq.

DOI:10.6814/NCCU202101389

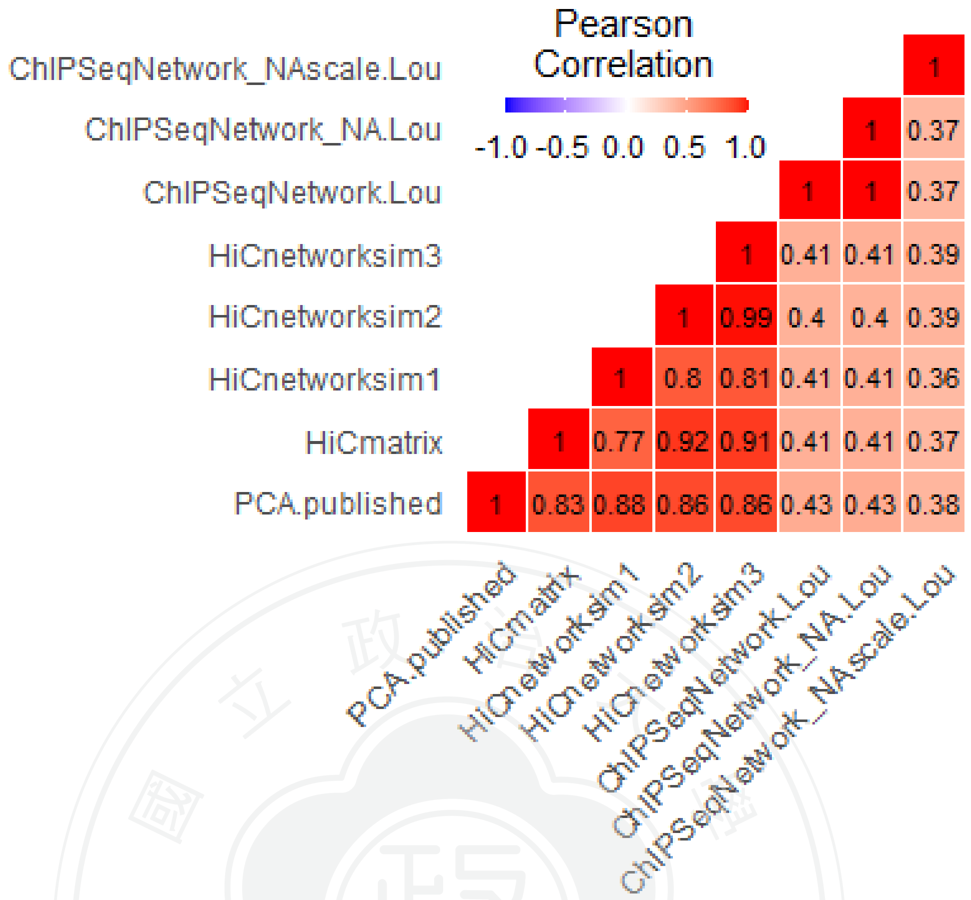


Figure 20. The correlation of cluster results after doing the Louvain method. We compare the eigenvector given by Juicer and Hi-C data using PCA method in 25KB resolution to the result of Hi-C data using three different input methods and ChIP-Seq data using three other input methods.

We also use the Pearson correlation value to determine the similarity of the results using the Leiden method. As a result, the numerical distribution of the Pearson correlation value is similar to the result of the Louvain method (Figure 21).

DOI:10.6814/NCCU202101389

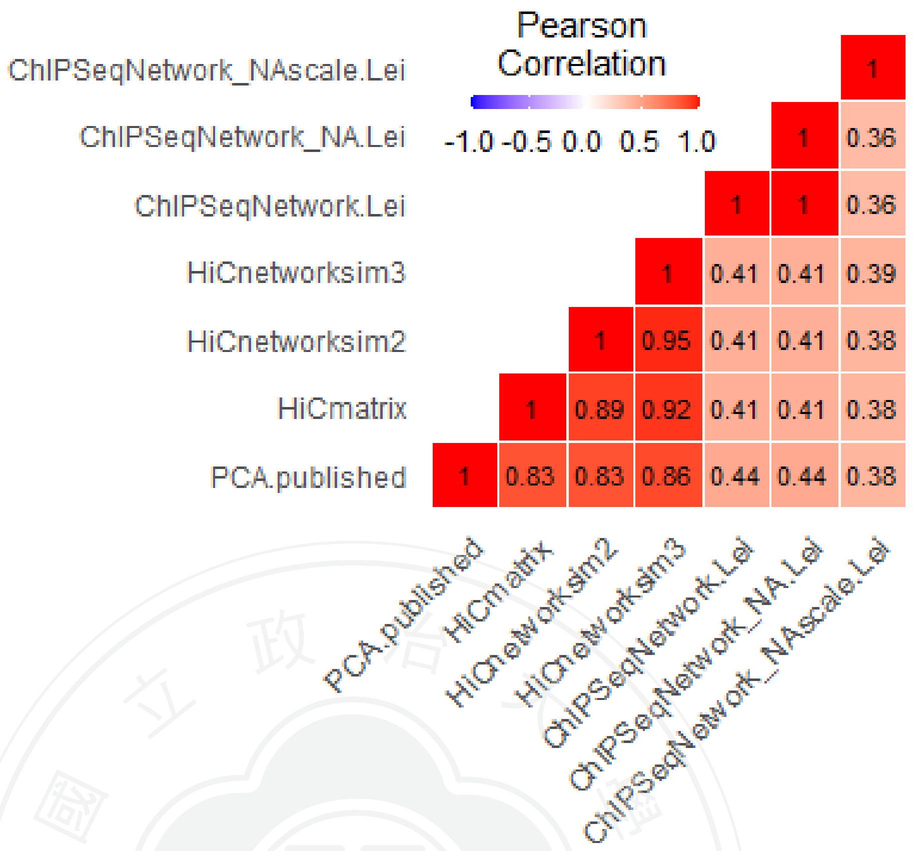


Figure 21. The correlation of cluster results after doing the Leiden method.

3.5. Combination of ChIP-Seq and Hi-C

After analyzing the Hi-C datasets and ChIP-Seq data of Rao2014, we wonder whether the information combined by these two datasets are better segments or not. Hence, we first add these two datasets together, and then use the Louvain and Leiden method to cluster. Next, we use the SNF method and still use the Louvain and Leiden method to compare with the way we only add together.

We use the Louvain method to compare the figures, which are only ChIP-Seq data and only Hi-C data (Figure 22). We use Juicebox software to visualize our result of the Louvain method. The first track (PCA.published) is the eigenvectors produced by Juicer. The second track (ChIP-Seq.network) results from the Louvain clustering method of ChIP-Seq data. The third track (HiC.network$_{sim3}$) results from the Louvain method of Rao2014. The fourth track (Fusion_SNF) results from the Louvain method of SNF of ChIP-Seq and Hi-C, grouping into six clusters. Finally, the last track (Fusion_adding) results from the Louvain method of ChIP-Seq and Hi-C data added together, set into three groups.

DOI:10.6814/NCCU202101389

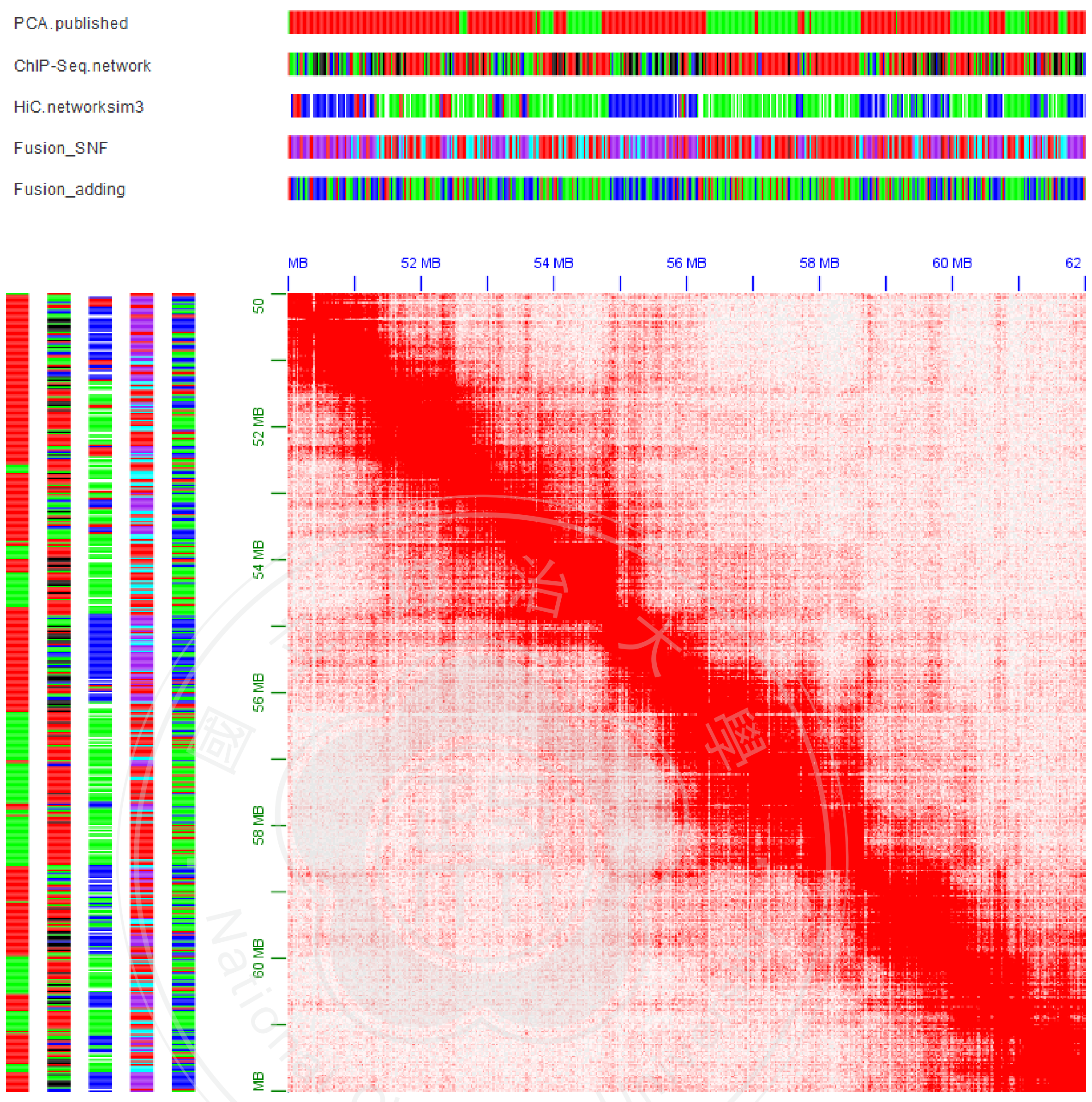


Figure 22. Genome segmentation based on PCA of HiC matrix, ChIP-Seq, Hi-C and fusion networks using the Louvain method on chromosome 14:50MB~62MB in 25KB resolution of Rao2014 where white indicates missing data (NA) in HiC.network.

We also use the Leiden method to compare the figures, only ChIP-Seq data and Hi-C data (Figure 23). The second track (ChIP-Seq.network) results from the Leiden clustering method of ChIP-Seq data. The third track (HiC.networksim3) results from the Leiden method of Rao2014. The fourth track (Fusion_SNF) results from the Leiden method of SNF of ChIP-Seq and Hi-C, grouping into three clusters.

DOI:10.6814/NCCU202101389

Finally, the last track (Fusion_adding) results from the Leiden method of ChIP-Seq and Hi-C data were added together, set into three groups.

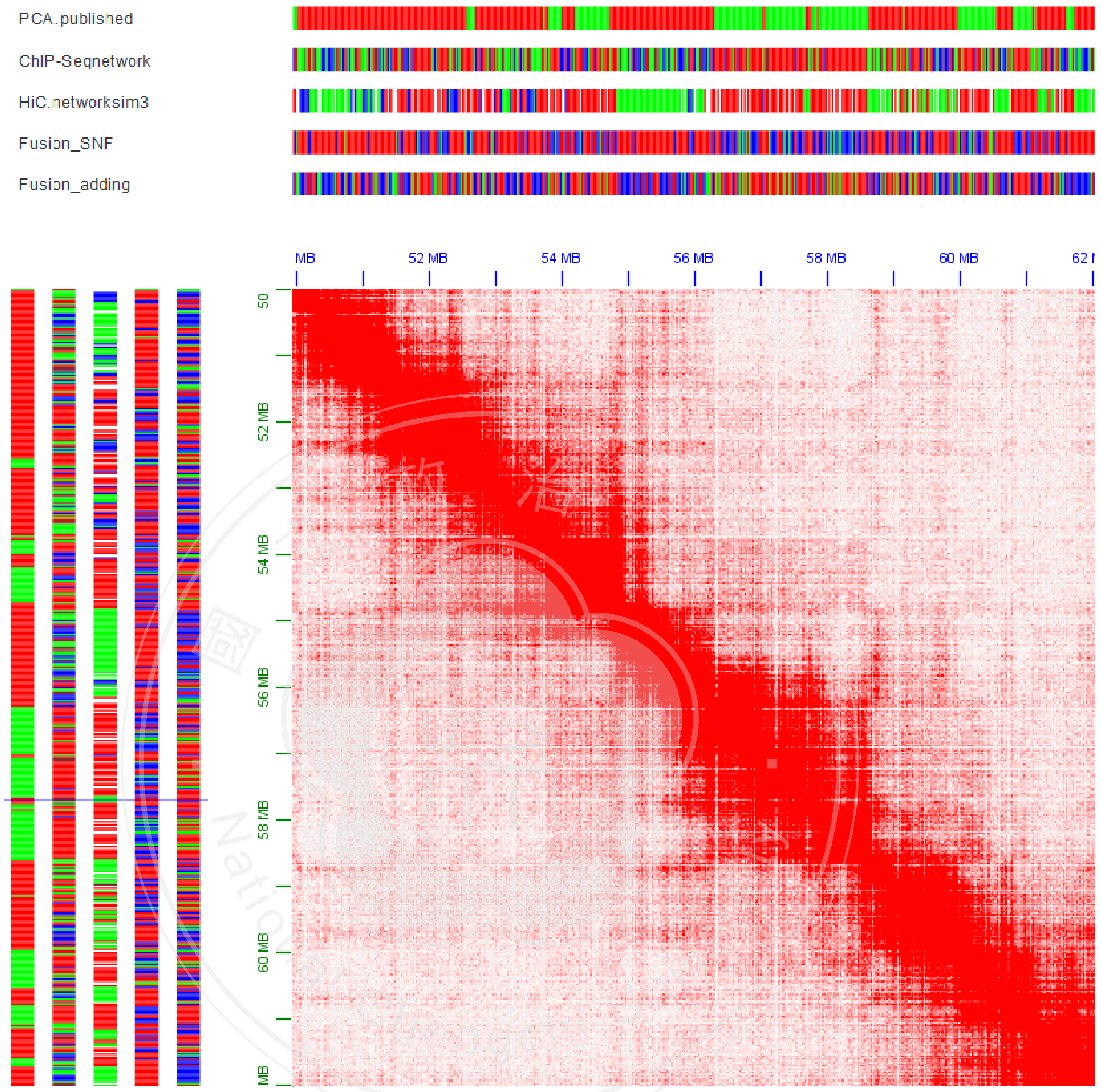


Figure 23. Genome segmentation based on PCA of HiC matrix, ChIP-Seq, Hi-C and fusion networks using the Leiden method on chromosome 14:50MB~62MB in 25KB resolution of Rao2014 where white indicates missing data (NA) in HiC.network.

After acquiring the result of the Louvain and Leiden method, we find that the the eigenvectors produced by Juicer only produce the information of A/B compartments, and the ChIP-Seq network produce too fragmented, and the Hi-C network is similar to eigenvectors but not that precise. Our result of adding function performs the segments which are almost affected by the ChIP-Seq data, and the result of SNF performs the balanced result which contains the precision of Hi-C data and

DOI:10.6814/NCCU202101389

also more information from ChIP-Seq data. Hence, using our SNF method can get a compromise of the Hi-C data and ChIP-Seq data.

We also test the Louvain and Leiden methods by giving different parameters *resolution* for two fusion networks (Fusion.network, Table 3).

DOI:10.6814/NCCU202101389

4. Discussion and Conclusion

We provide a platform that clusters Hi-C and ChIP-Seq individually. In this platform, we can use Hi-C datasets to reproduce the result of compartments A and B. Furthermore, we can use ChIP-Seq data to classify the role in transcription, which belongs to activation or repression. On top of that, we merge two different data types to get more information to receive more references to segment the genome better.

We try to figure out the clusters produced by the SNF method corresponding to which targets in ChIP-Seq, which means the combined groups are classified as activation or repression. Hence, the result of the SNF method needs more analysis.

DOI:10.6814/NCCU202101389

References

1. Balazs, R. (2014). Epigenetic mechanisms in Alzheimer's disease. Degenerative neurological and neuromuscular disease, 4, 85.
2. Blondel, V. D., Guillaume, J. L., Lambiotte, R., & Lefebvre, E. (2008). Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment*, 2008 (10), P10008.
3. Bo Wang, Aziz M Mezlini, Feyyaz Demir, Marc Fiume, Zhuowen Tu, Michael Brudno, Benjamin Haibe-Kains & Anna Goldenberg (2014). Similarity network fusion for aggregating data types on a genomic scale. *Nature Methods* volume 11, 333–337.
4. ChromHMM: Chromatin state discovery and characterization. http://compbio.mit.edu/ChromHMM/
5. Community detection for NetworkX's documentation (2010). https://Python-louvain.readthedocs.io/en/latest/
6. Dekker,J. et al. (2002) Capturing chromosome conformation. *Science*, 295, 1306–11.
7. Eigenvector, Juicer (2017). https://github.com/aidenlab/juicer/wiki/Eigenvector
8. ENCODE: Encyclopedia of DNA Elements. https://www.encodeproject.org/
9. Eugenio Marco1, Wouter Meuleman, Jialiang Huang, Kimberly Glass, Luca Pinello, Jianrong Wang,Manolis Kellis & Guo-Cheng Yuan (2017). Multi-scale chromatin state annotation using a hierarchical hidden Markov model. *Nature communications*. DOI: 10.1038/ncomms15011
10. Illumina et al. (2007) *Pub*. No. 770-2007-007 Current as of 26 November 2007. Whole-Genome Chromatin IP Sequencing (ChIP-Seq).
11. Introduction of dataset preprocessing (2014). File: GSE63525_GM12878_combined_README.rtf. https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE63525
12. Kloetgen, A., Thandapani, P., Ntziachristos, P., Ghebrechristos, Y., Nomikou, S., Lazaris, C., ... & Tsirigos, A. (2020). Three-dimensional chromatin landscapes in T cell acute lymphoblastic leukemia. *Nature genetics*, 52(4), 388-400.
13. Lan, X., Witt, H., Katsumura, K., Ye, Z., Wang, Q., Bresnick, E. H., ... & Jin, V. X. (2012). Integration of Hi-C and ChIP-seq data reveals distinct types of chromatin linkages. *Nucleic acids research*, 40(16), 7690-7704.

DOI:10.6814/NCCU202101389

14. Lieberman-Aiden E, Van Berkum N L, Williams L, et al. Comprehensive Mapping of Long-Range Interactions Reveals Folding Principles of the Human Genome. *Science* 326, 289–293 (2009).
15. Lin Liu, Yiqian Zhang, Jianxing Feng, Ning Zheng, Junfeng Yin, Yong Zhang (2012). GeSICA: genome segmentation from intra-chromosomal associations. *BMC Genomics*. 2012 May 4;13:164. doi: 10.1186/1471-2164-13-164.
16. Luo, Z., Wang, X., Jiang, H., Wang, R., Chen, J., Chen, Y., ... & Song, X. (2020). Reorganized 3D genome structures support transcriptional regulation in mouse spermatogenesis. *iScience*, 23(4), 101034.
17. Network fusion. https://nbisweden.github.io/workshop_omics_integration/session_nmf/SNF_main.html
18. Rao, S.S.P., Huntley, M.H., Durand, N.C., Stamenova, E.K., Bochkov, I.D., Robinson, J.T., Sanborn, A.L., Machol, I., Omer, A.D., Lander, E.S., et al. (2014). A 3D map of the human genome at kilobase resolution reveals principles of chromatin looping. *Cell* 159, 1665–1680.
19. SIMILARITY NETWORK FUSION(SNF). http://compbio.cs.toronto.edu/SNF/SNF/Software.html
20. Strahl, B. D., & Allis, C. D. (2000). The language of covalent histone modifications. *Nature*, 403(6765), 41-45.
21. Traag, V.A., Waltman, L. & van Eck, N.J. From Louvain to Leiden: guaranteeing well-connected communities. *Sci Rep* 9, 5233 (2019). https://doi.org/10.1038/s41598-019-41695-z
22. Van Berkum, Nynke L et al. (2010) Hi-C: a method to study the three-dimensional architecture of genomes. Journal of visualized experiments : *JoVE* ,39, 1869.
23. Visualization tool: Juicebox. https://www.aidenlab.org/juicebox/
24. Waltman, L., & Van Eck, N. J. (2013). A smart local moving algorithm for large-scale modularity-based community detection. *The European physical journal B*, 86(11), 1-14.
25. Weighted correlation network analysis. https://en.wikipedia.org/wiki/Weighted_correlation_network_analysis
26. networkanalysis, CWTSLeiden (2020). https://github.com/CWTSLeiden/networkanalysis

DOI:10.6814/NCCU202101389