

# Contents

<b>1</b>	<b>A Biology Primer</b>	<b>4</b>
1.1	The Central Dogma of Molecular Biology . . . . .	4
1.2	DNA . . . . .	4
1.2.1	Function . . . . .	4
1.2.2	Structure . . . . .	5
1.2.3	Replication . . . . .	5
1.3	Transcription . . . . .	5
1.3.1	mRNA generation . . . . .	5
1.3.2	Post-transcriptional modifications . . . . .	5
<b>2</b>	<b>Hidden Markov Models</b>	<b>7</b>
2.1	Overview . . . . .	7
2.2	Mathematical Definition(s) . . . . .	8
2.3	An Example . . . . .	8
2.4	Computational problems with HMMs . . . . .	9
2.4.1	Decoding problem : . . . . .	10
2.4.2	Likelihood Problem: . . . . .	10
2.4.3	Learning problem: . . . . .	10
2.5	Conclusions . . . . .	11
2.5.1	Pros: . . . . .	11
2.5.2	Cons: . . . . .	11
<b>3</b>	<b>Linear Regression, Chi-Squared Test of Independence, and applications to GWAS.</b>	<b>12</b>
3.1	Brief review of linear models . . . . .	12
3.2	Relation to Hypothesis Testing . . . . .	13
3.3	Relationship to Correlation . . . . .	13
3.4	Multiple Linear Regression . . . . .	14
3.5	Chi-Squared Test of Independence and Relationship to GWAS . . . . .	15
<b>4</b>	<b>Regulatory DNA, Transcription factors, Sequence motifs</b>	<b>16</b>
4.1	Epigenome . . . . .	16
4.2	Epigenomic marks . . . . .	16
4.3	Epigenomics . . . . .	17
4.3.1	Transcription factors . . . . .	17
4.3.2	Sequence motif . . . . .	17
4.3.3	Sequence motif finding . . . . .	17
4.3.4	De novo sequence motif finding . . . . .	19
<b>5</b>	<b>ChIP-seq</b>	<b>21</b>
5.1	ChIP-seq experiment steps: . . . . .	21
5.2	History of the development of ChIP-seq technology . . . . .	21
5.3	ChIP-seq data analysis . . . . .	21

5.4	ChIP-seq: Study design . . . . .	22
5.5	ChIP-seq: Peak calling . . . . .	22
<b>6</b>	<b>Epigenomic domains, Hierarchy and scales of genome structure</b>	<b>23</b>
6.1	Epigenomics and chromatin structure . . . . .	23
6.2	Functional annotation of common histone marks . . . . .	23
6.3	Comparing transcription factors and histone marks . . . . .	24
6.4	Histone modification patterns . . . . .	24
6.5	Chip-seq analysis (continued) . . . . .	24
6.6	SICER . . . . .	25
6.7	Hi-C . . . . .	26
<b>7</b>	<b>Genomic Intervals: formats, data structures and algorithms</b>	<b>27</b>
7.1	Overview . . . . .	27
7.2	WHAT CAN BE REPRESENTED AS AN INTERVAL? . . . . .	27
7.3	FILE FORMATS FOR GENOMIC INTERVALS . . . . .	28
7.3.1	BED . . . . .	28
7.3.2	NARROWPEAK (BED-like) . . . . .	29
7.3.3	BROADPEAK (BED-like) . . . . .	29
7.3.4	GFF . . . . .	29
7.4	RELATED FILE FORMATS . . . . .	30
7.4.1	WIGGLE . . . . .	30
7.4.2	BEDGRAPH . . . . .	31
7.4.3	WHY SO MANY FORMATS? . . . . .	31
7.5	BIOLOGICAL QUESTIONS . . . . .	31
7.6	INTERVAL COMPUTATIONS . . . . .	32
7.6.1	OVERLAP . . . . .	32
7.6.2	COUNTING OVERLAP . . . . .	32
7.6.3	INTERSECTING REGION . . . . .	32
7.6.4	UNION . . . . .	32
7.6.5	INTERVAL SET OVERLAP COUNTING . . . . .	32
7.7	BINARY SEARCH . . . . .	33
7.8	BINARY SEARCH TREES . . . . .	34
7.8.1	Hidden assumption for BST: No containment . . . . .	35
7.9	ADVANCED DATA STRUCTURES . . . . .	35
7.9.1	B trees . . . . .	35
7.9.2	R trees . . . . .	35
7.9.3	Nested containment lists (Nclist) . . . . .	37
7.9.4	Augmented interval list (AIList) . . . . .	37
7.10	INTRO TO BIOCONDUCTOR GENOMIC RANGES . . . . .	37
7.11	REGION SET ENRICHMENT ANALYSIS . . . . .	41
7.11.1	INTERVAL SIMILARITY METRICS . . . . .	41
7.11.2	REGION SET ENRICHMENT ANALYTICAL TOOLS . . . . .	41
7.12	CONCLUSION . . . . .	41
7.13	TERMINOLOGY . . . . .	41
<b>8</b>	<b>ATAC-seq diagnostics and harmonization</b>	<b>42</b>
8.1	ATAC-seq Overview . . . . .	42
8.1.1	What's ATAC-seq? . . . . .	42
8.1.2	Tn5 transposase . . . . .	42
8.1.3	Advantages against other technologies . . . . .	42
8.1.4	Difference between ATAC-seq and ChIP-seq . . . . .	42
8.2	Basic Algorithm for Sequence-Based Genome Data . . . . .	43
8.2.1	Basic Workflow . . . . .	43

8.2.2	ATAC-seq Peak Calling . . . . .	44
8.2.3	QC Metrics . . . . .	44
8.3	Harmonization . . . . .	45
8.3.1	Problem 1: Peak locations differ across samples . . . . .	45
8.3.2	Problem 2: Sequencing depth normalization . . . . .	47
8.3.3	Quantile-quantile plots . . . . .	47
8.4	Batch effect . . . . .	50
8.4.1	Sources of batch effects . . . . .	50
8.4.2	Batch correction approaches . . . . .	50
<b>9</b>	<b>Scalable Computing in Genomics</b>	<b>51</b>
9.1	Lower costs → More data . . . . .	51
9.2	Typical Epigenome Project . . . . .	51
9.3	Approaches for Scalability . . . . .	51
9.4	Scopes of Parallelism . . . . .	51
9.5	Workflows . . . . .	52
9.6	Stop using shellscripts for pipelines! . . . . .	53
9.7	Advantages of workflows . . . . .	53
9.8	Optimization . . . . .	53
9.9	Language choice . . . . .	53
9.10	Databases . . . . .	53
9.11	Problem with Cloud Platforms . . . . .	54

# Chapter 1

## A Biology Primer

### 1.1 The Central Dogma of Molecular Biology

The central dogma of molecular biology explains the flow of genetic information in the cell between information-carrying biopolymers (DNA, RNA and protein). It states that the transfer of information from nucleic acid to nucleic acid, or from nucleic acid to protein may be possible, but transfer from protein to protein, or from protein to nucleic acid is impossible.

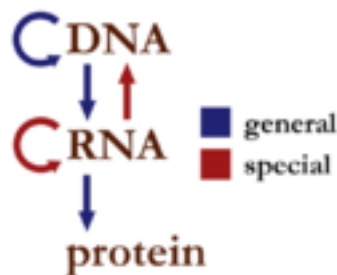


Figure 1.1: Information flows between DNA, RNA and protein. Source: Wikipedia

The genetic code of an organism is stored in DNA, which is converted into portable RNA messages in a process called transcription. These messages travel from the cell nucleus (where the DNA resides) to the ribosomes where they are used as template to make specific proteins in a process called translation. The central dogma states that the pattern of information that occurs most frequently in our cells is:

- From existing DNA to make new DNA (replication)
- From DNA to make new RNA (transcription)
- From RNA to make new proteins (translation).

Besides these, there are some notable possibilities. For instance, retroviruses are able to generate DNA from RNA via reverse-transcription, and some viruses use RNA to make protein. All this is shown in Figure 1.1. The generated proteins carry out most of the cellular functions such as metabolism, DNA regulation, and replication.

### 1.2 DNA

#### 1.2.1 Function

The DNA molecule stores the genetic information of an organism. DNA contains regions called genes, which encode for the proteins that carry out most of the cellular function. Other regions of the DNA contain

regulatory elements, which partially influence the level of expression of each gene.

### 1.2.2 Structure

The DNA molecule consists of two strands that wind around to form a shape known as a double helix. Each strand has a backbone made of alternating sugar (deoxyribose) and phosphate groups. Attached to each sugar is one of the four bases: adenine, cytosine, guanine, and thymine, frequently represented using the letters A, C, G, and T respectively. The two strands are held together by bonds between the bases: A and T are connected by two hydrogen bonds, while C and G are connected by three bonds. This specificity in pairing means that one strand can be used as a template to generate the other strand.

The DNA strands also have directionality, which refers to the positions of the pentose ring where the phosphate backbone connects. This directionality convention comes from the fact that DNA and RNA polymerase synthesize in the 5' to 3' direction. The complementary pairing with directionality means that the DNA strands are anti-parallel. In other words the 5' end of one strand is adjacent to the 3' end of the other strand. As a result, DNA can be read both in the 3' to 5' direction and the 5' to 3' direction, and genes and other functional elements can be found in each direction (on either strand). By convention, DNA is written from 5' to 3'.

Base pairing between nucleotides of DNA constitutes its primary and secondary structure. In addition to DNA's secondary structure, there are several extra levels of structure that allow DNA to be tightly compacted and influence gene expression. The tertiary structure describes the twist in the DNA ladder that forms a helical shape. In the quaternary structure, DNA is tightly wound around small proteins called histones. These DNA-histone complexes are further wound into tighter structures seen in chromatin.

### 1.2.3 Replication

The structure of DNA with its weak hydrogen bonds between the bases in the center allows the strands to easily be separated for the purpose of DNA replication. In the replication of DNA, the two complementary strands are separated, and each of the strands are used as templates for the construction of a new strand. DNA polymerases attach to each of the strands at the origin of replication, reading each existing strand from the 3' to 5' direction and placing complementary bases such that the new strand grows in the 5' to 3' direction. Because the new strand must grow from 5' to 3', one strand (leading strand) can be copied continuously, while the other (lagging strand) grows in fragments that are later pasted together by DNA ligase. The end result is 2 double-stranded pieces of DNA, where each is composed of 1 old strand, and 1 new strand. For this reason, DNA replication is semi-conservative.

## 1.3 Transcription

### 1.3.1 mRNA generation

Transcription is the process to produce RNA using a DNA template. The DNA is partially unwound to form a bubble, and RNA polymerase is recruited to the transcription start site (TSS) by regulatory protein complexes. RNA polymerase reads the DNA from the 3' to 5' direction and placing down complementary bases to form messenger RNA (mRNA). RNA uses the same nucleotides as DNA, except Uracil (U) is used instead of Thymine (T).

### 1.3.2 Post-transcriptional modifications

Messenger RNA (mRNA) in eukaryotes experience post-translational modifications, or processes that edit the mRNA strand further. Most notably, a process called splicing removes introns (intervening regions which don't code for protein), so that only the coding regions (the exons), remain. Different regions of the primary transcript may be spliced out and each can lead to a different protein product. This phenomenon is referred to as alternative splicing. In this way, an large number of protein products can be generated based on different splicing permutations. In addition to splicing, both ends of the mRNA molecule are processed. The

5' end is capped with a modified guanine nucleotide. At the 3' end, roughly 250 adenine residues are added to form a poly(A) tail.

## Chapter 2

# Hidden Markov Models

### 2.1 Overview

A **Hidden Markov Model (HMM)** is a statistical **Markov model** in which the system being modeled is assumed to be a Markov process. That is, it is a “memoryless” system whose trajectory is solely determined by its current state. The HMM is considered “hidden” because we do not (or cannot) know about the states of the variable being observed (say,  $X$ ). Hence, we attempt to learn about  $X$  by observing  $Y$ , some sort of observation/event that occurs due to the hidden states. Like all Markov processes, HMM has an additional requirement that the outcome of  $Y$  at time  $t = t_0$  may be “influenced” **only** by the outcome of  $X$  at  $t = t_0$  and that the outcomes of  $X$  and  $Y$  at  $t < t_0$  must **not** affect the outcome of  $Y$  at  $t = t_0$ . I.e. the states before the current state have no impact on the future except via the current state. It’s as if to predict tomorrow’s weather you could examine today’s weather but you were not allowed to look at yesterday’s weather!

Here is an example of a 3-state **Markov model**:

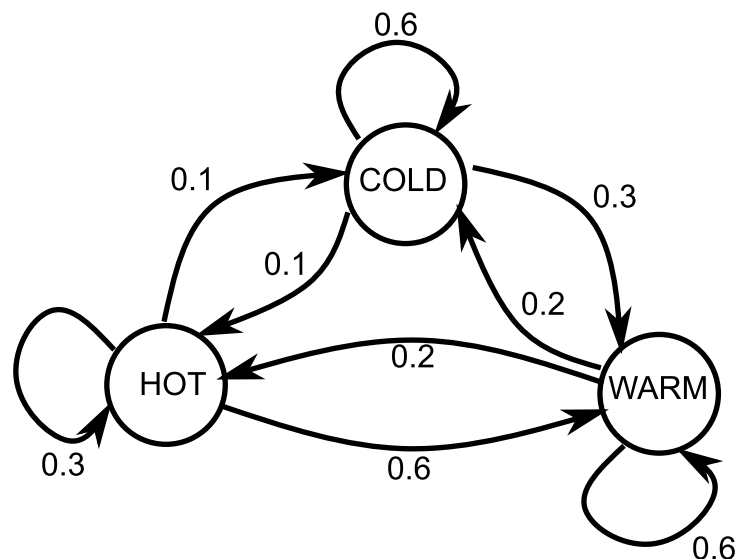


Figure 2.1: Simple Markov Model

As we move from state to state (*node to node* or *circle to circle*), there is a **weight** associated with each edge, indicating the probability that we move from one node to another.

A Markov chain is useful when we need to compute a probability for a sequence of observable events. In many cases, however, the events we are interested in are hidden: we don’t observe them directly. For example, we

don't normally observe part-of-speech tags in a text. Rather, we see words and must infer the tags from the word sequence. We call the tags hidden because they are not observed.

HMMs have applications in all sorts of areas including **thermodynamics, economics, speech, pattern recognition, bioinformatics**, and more. They provide a foundation for probabilistic models of linear sequence 'labeling' problems.

## 2.2 Mathematical Definition(s)

Mathematically, if we consider a sequence of state variables  $q_1, q_2, \dots, q_i$  then the **markov assumption** is as follows:

$$P(q_i = a | q_1, q_2, \dots, q_{i-1}) = P(q_i = a | q_{i-1})$$

The values of weights (or probabilities) associated with each edge coming off of a state (or node) must sum up to 1. A Markov model has a set of states:

$$S = \{s_1, s_2, s_3, \dots, s_n\}$$

The **Markov process** moves from one state to another generating a sequence of states:

$$s_{i1}, s_{i2}, s_{i3}, \dots, s_{ik} \dots$$

The following need to be defined for a **Markov model**: 1. **Transition probabilities**:

$$A = (a_{ij}), a_{ij} = P(s_i, s_j)$$

2. **Initial Probabilities ( $\pi$ )**:

$$\pi = \{P(s_1), P(s_2), \dots, P(s_i)\}$$

A **hidden** Markov model requires one more mathematical definition. We need to know the probability of observing an event **given** a state:

$$B = (b_i(v_m)), b_i(v_m) = P(v_m | s_i)$$

These are known as **emission probabilities**. The probability that given a state, we "emit" to a certain observation.

Given the above, we can alter the graph model above to represent a **hidden** Markov model:

## 2.3 An Example

*The following example problem is pulled from wikipedia:*

Consider two friends, Alice and Bob, who live far apart from each other and who talk together daily over the telephone about what they did that day. Bob is only interested in three activities: walking in the park, shopping, and cleaning his apartment. The choice of what to do is determined exclusively by the weather on a given day. Alice has no definite information about the weather, but she knows general trends. Based on what Bob tells her he did each day, Alice tries to guess what the weather must have been like.

Alice believes that the weather operates as a discrete Markov chain. There are two states, "Rainy" and "Sunny", but she cannot observe them directly, that is, they are *hidden* from her. On each day, there is a certain chance that Bob will perform one of the following activities, depending on the weather: "walk", "shop", or "clean". Since Bob tells Alice about his activities, those are the *observations*. The entire system is that of a hidden Markov model (HMM).

Alice knows the general weather trends in the area, and what Bob likes to do on average. In other words, the parameters of the HMM are known. They can be represented as follows in Python:



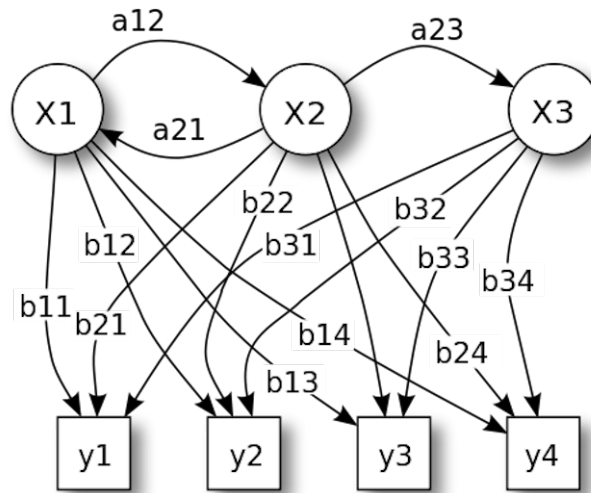


Figure 2.2: Simple hidden Markov model. Source: Wikipedia

```
states = ('Rainy', 'Sunny')

observations = ('walk', 'shop', 'clean')

start_probability = {'Rainy': 0.6, 'Sunny': 0.4}

transition_probability = {
    'Rainy': {'Rainy': 0.7, 'Sunny': 0.3},
    'Sunny': {'Rainy': 0.4, 'Sunny': 0.6},
}

emission_probability = {
    'Rainy': {'walk': 0.1, 'shop': 0.4, 'clean': 0.5},
    'Sunny': {'walk': 0.6, 'shop': 0.3, 'clean': 0.1},
}
```

In this piece of code, `start_probability` represents Alice's belief about which state the HMM is in when Bob first calls her (all she knows is that it tends to be rainy on average). The particular probability distribution used here is not the equilibrium one, which is (given the transition probabilities) approximately `{'Rainy': 0.57, 'Sunny': 0.43}`. The `transition_probability` represents the change of the weather in the underlying Markov chain. In this example, there is only a 30% chance that tomorrow will be sunny if today is rainy. The `emission_probability` represents how likely Bob is to perform a certain activity on each day. If it is rainy, there is a 50% chance that he is cleaning his apartment; if it is sunny, there is a 60% chance that he is outside for a walk.

## 2.4 Computational problems with HMMs

There are many computational problems with HMMs. Below are just a few. In general, they involve the use of dynamic programming and gradient descent while solving for the maximum likelihood of a certain sequence of states given observations. Oftentimes, the probabilities in these algorithms are represented in

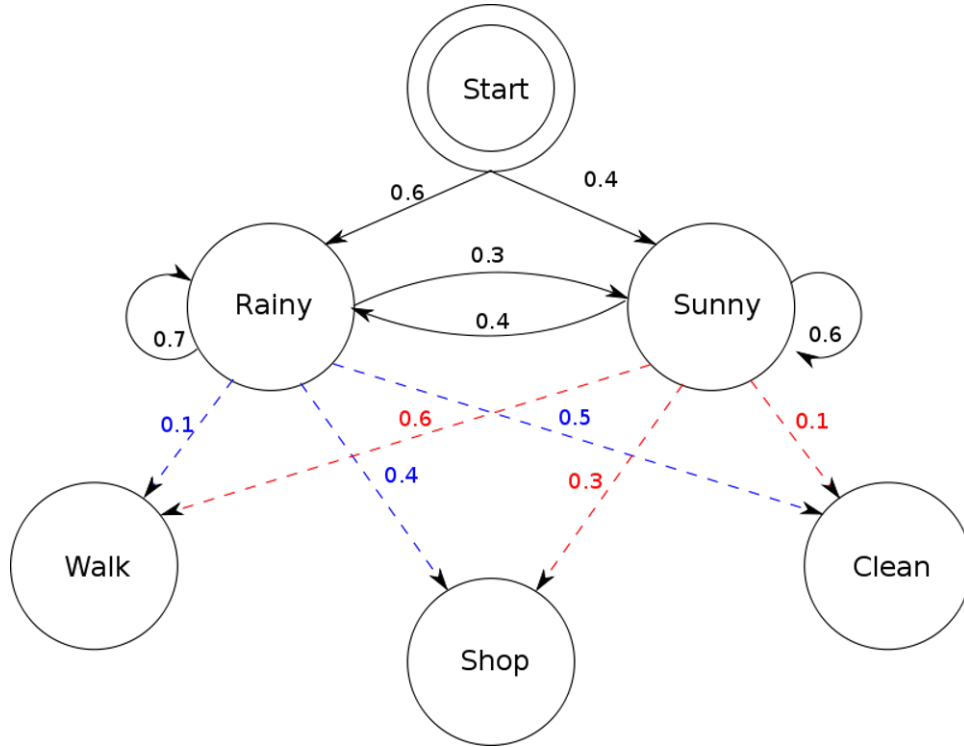


Figure 2.3: Hidden Markov Model to predict weather. Source: Wikipedia

**log space** to make it easier to work with the math while preventing **underflow** errors at the CPU level (numbers way too small for a computer to handle).

### 2.4.1 Decoding problem :

Given the HMM  $M=(A,B,\pi)$ , and an observation sequence  $O$  calculate the most likely sequence of states that produced  $O$ . This is commonly solved using the Viterbi Algorithm and involves the application of dynamic programming to recurse through a state matrix and for obtaining the maximum *a posteriori* probability estimate of the most likely sequence of hidden states—called the Viterbi path—that results in a sequence of observed events, especially in the context of Markov information sources and hidden Markov models (HMM).

### 2.4.2 Likelihood Problem:

Similar to the above decoding problem, given the HMM,  $M=(A,B,\pi)$ , and an observation sequence  $O, o_i \in \nu_1, \nu_2, \dots, \nu_M$  we need to calculate the likelihood  $P(O|M)$  using the probabilities of observations given a set of states:

$$P(O|S) = \prod_{i=1}^T P(o_i|S_i)$$

However, the state sequence is unknown.

### 2.4.3 Learning problem:

Given an observation sequence,  $O$ , and general structure of HMM, determine HMM parameters that best fit the training data. Here we are solving a sort of reverse problem. That is, we **do not know** the specific probabilities of the transition or emission states. All we know is the overall structure of the model, and

using a set of training data, we can fit our model to produce “optimal” values for  $A$ ,  $B$ , and  $\pi$ , such that the model can be applied elsewhere.

The most well-known algorithm for this is the Baum-Welch algorithm, which utilizes a stochastic gradient descent algorithm and is not guaranteed to provide an optimal solution. It can also be very computationally complex.

## 2.5 Conclusions

HMMs offer great prediction and modeling potential in the form of a highly-interpretable and statistically sound model/algorithm. They can be applied to many real-world problems and are often computationally efficient (when making inferences). They still, however, have both pros and cons:

### 2.5.1 Pros:

- HMM models are highly studied, statistically sound, and highly interpretable models.
- Easy to implement and analyze.
- Incorporates prior knowledge into the model architecture.
- Can be initialized close to something believed to be correct
- Widely applicable

### 2.5.2 Cons:

- Bounded by the Markov assumption: The next state is only determined by the current state and not previous ones
- For EM learning problems, the number of parameters to be evaluated is huge. So it needs a large data set for training.
- Training an HMM can often be computationally challenging.

## Chapter 3

# Linear Regression, Chi-Squared Test of Independence, and applications to GWAS.

### 3.1 Brief review of linear models

A linear model is a simple way to demonstrate a relationship between two variables. The most powerful statistical tools to process sequencing data exploit the relationship between linear models and hypothesis testing. When we utilize hypothesis testing with linear models, it allows us to ascribe significance to relationships in the data. This is useful in techniques such as Genome Wide Association Studies (GWAS), where we seek to uncover relationships between genetic patterns and certain phenotypes or diseases. Some might even consider linear models to be a simple form of machine learning! The equation for a linear model is given below:

$$y_i = \beta_0 + \beta_1 x_i + e_i, i = 1, \dots, n$$

Here  $e_i$  are independent random variables with  $E(e_i) = 0$  and  $Var(e_i) = \sigma^2$ . The  $x_i$  are assumed to be fixed. This is referred to as the standard statistical model: value of  $y$  is a linear function of  $x$  plus random noise.  $y$  is called the **dependent** or response variable and  $x$  is called the **independent** or predictor variable.

There are several numerical approaches to finding the parameters of a linear model if they must be determined. One such method is called the “Method of Least Squares”, and it involves optimizing the  $\beta$  parameters such that we minimize the sum of squared residuals. We can write this as the following equation:

$$S(\beta_0, \beta_1) = \sum_{i=1}^n (y_i - \beta_0 - \beta_1 x_i)^2$$

$$\frac{\partial S}{\partial \beta_0} = -2 \sum_{i=1}^n (y_i - \beta_0 - \beta_1 x_i) = 0$$

$$\frac{\partial S}{\partial \beta_1} = -2 \sum_{i=1}^n x_i (y_i - \beta_0 - \beta_1 x_i) = 0$$

Here  $S$  is the sum of squared residuals. Another way of thinking of this is minimizing the squared difference between the actual  $y$  values and the fitted  $y$  values. Here, partial derivatives are taken with respect to each  $\beta$  parameter to minimize the sum of squared residuals. The principle is clearly illustrated in the following image ( $d_1$  and  $d_2$  represent the distances between the actual  $y$  values and their fitted values, which the method of least squares seeks to minimize):

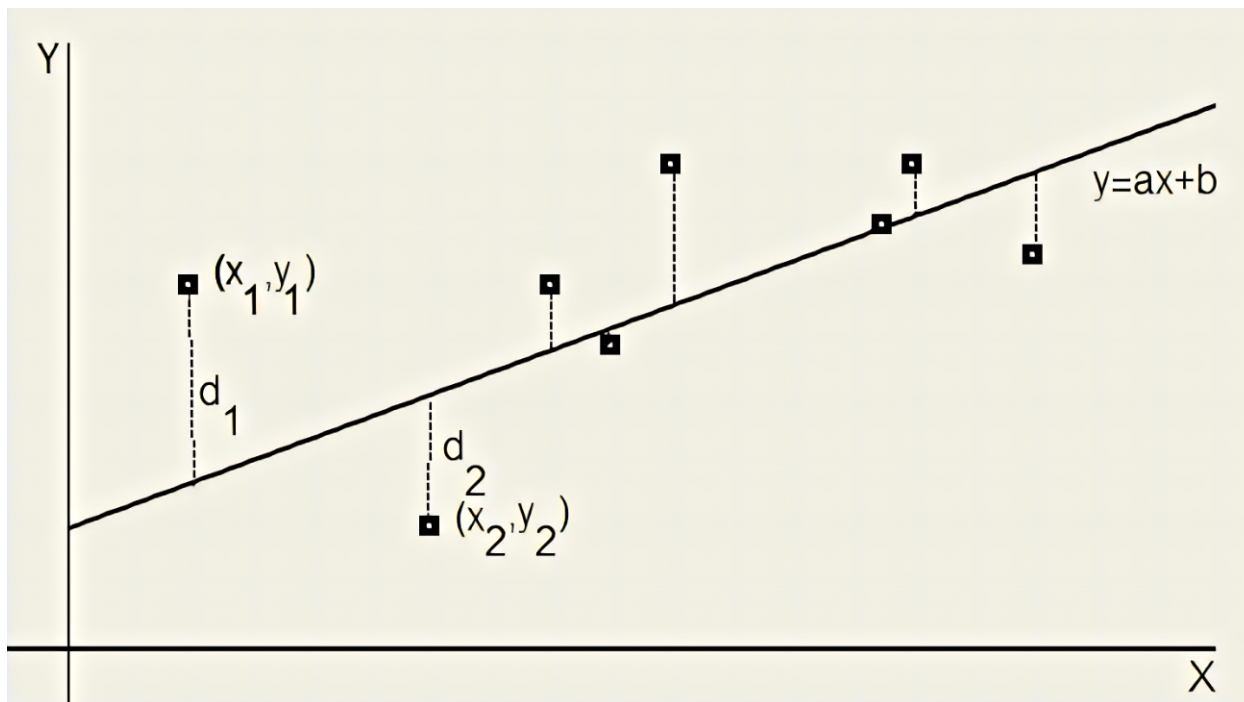


Figure 3.1: Method of least squares seeks to minimize the sum of squared residuals

## 3.2 Relation to Hypothesis Testing

First, we can define the residual sum of squares (RSS) of our data. If the errors are independent normal random variables, then the  $\beta$  parameters are normally distributed with

$$\frac{\hat{\beta}_i - \beta_i}{s_{\hat{\beta}_i}} \sim t_{n-2}$$

Here  $t_{n-2}$  is a t-distribution with  $n-2$  degrees of freedom. Then, we can test the null hypothesis  $H_0 : \beta_1 = 0$ . A rejection of this null hypothesis would indicate that the slope of the regression line is non-zero; therefore, a relationship exists between the dependent and independent variable.

As a reminder, the Student's t-test tests the null hypothesis against a t-distribution. Where our  $\beta$  parameters fall in the t-distribution gives us the probability that  $\beta_1 = 0$ . If the probability  $p < 0.05$  (or another significance threshold), we can reject the null hypothesis.

## 3.3 Relationship to Correlation

Similarly, we can define a correlation coefficient between an independent and dependent variable. The correlation coefficient does not parameterize the relationship between  $x$  and  $y$  as a linear regression can, but it can be used to determine our confidence in the relationship. Let's define the following quantities:

$$s_{xx} = \frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})^2$$

$$s_{yy} = \frac{1}{n} \sum_{i=1}^n (y_i - \bar{y})^2$$

$$s_{xy} = \frac{1}{n} \sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})$$

The correlation coefficient between  $x$ 's and  $y$ 's is  $r = \frac{s_{xy}}{\sqrt{s_{xx}s_{yy}}}$  and  $\hat{\beta}_1 = \frac{s_{xy}}{s_{xx}}$ ; therefore  $r = \hat{\beta}_1 \sqrt{\frac{s_{xx}}{s_{yy}}}$ .

### 3.4 Multiple Linear Regression

Univariate linear models are useful to test the relationship between a single dependent and independent variable. However, in biology, we often have multiple covariates impacting a single dependent variable. Multiple linear regression allows us to examine the contribution of these multiple independent variables to our dependent variable.

The formula for multiple linear regression is similar to a simple linear regression. We model  $y = \beta_0 + \beta_1 x_1 + \dots + \beta_{p-1} x_{p-1}$  to fit data:  $y_i, x_{i1}, x_{i2}, \dots, x_{i,p-1}$  for  $i = 1, \dots, n$

We represent the  $x_{i,j}$  by an  $n \times p$  matrix  $\mathbf{X}$ :  $\mathbf{X} = \begin{bmatrix} 1 & x_{11} & x_{12} & \dots & x_{1,p-1} \\ 1 & x_{21} & x_{22} & \dots & x_{2,p-1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & x_{n1} & x_{n2} & \dots & x_{n,p-1} \end{bmatrix}$

Again, we can apply the method of least squares to this problem to find the solution given multiple unknown  $\beta$  parameters. In matrix notation, the vector containing all members of  $\beta$  is given by the following:

$$\mathbf{X}^T \mathbf{X} \hat{\beta} = \mathbf{X}^T \mathbf{Y}$$

If  $\mathbf{X}^T \mathbf{X}$  is nonsingular, then  $\hat{\beta} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{Y}$

This can be written in terms of the projection matrix,  $P$ , which projects onto the  $p$ -dimensional subspace of  $R^n$  spanned by the columns of  $X$ .  $\hat{Y}$  is a projection of  $Y$  onto the  $p$ -dimensional subspace spanned by the columns of  $X$ . Graphically, we can visualize this projection into a  $p$ -dimensional subspace as the following:

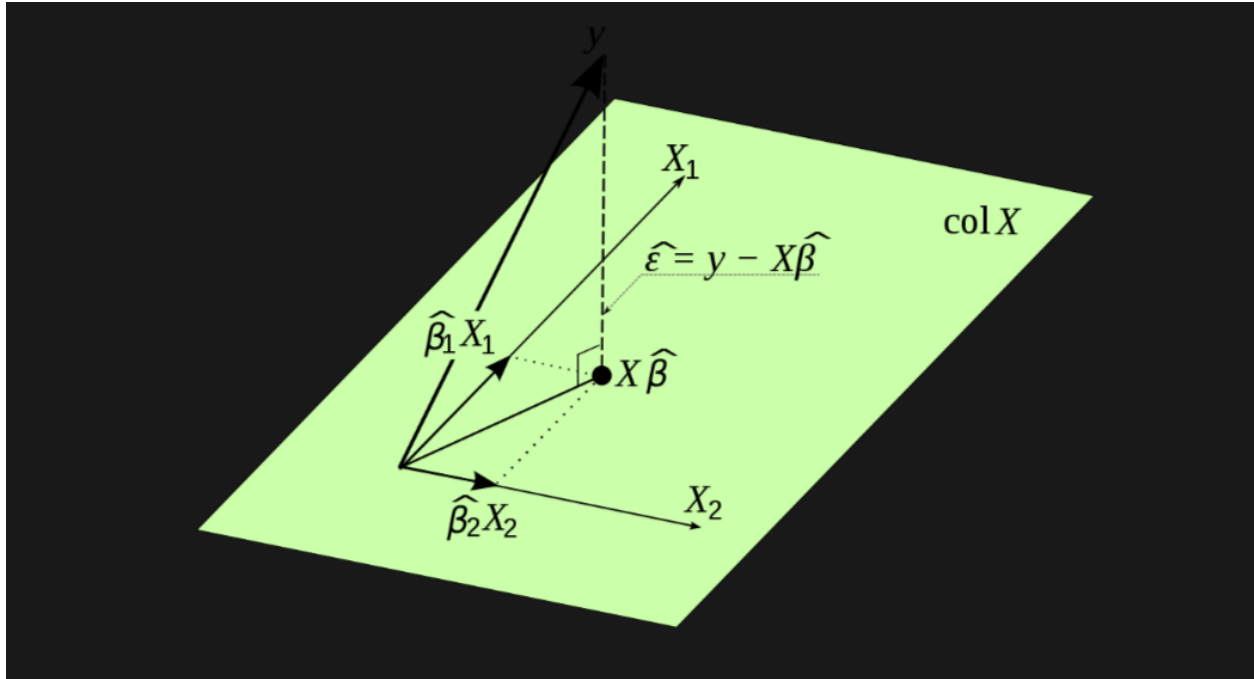


Figure 3.2: OLS estimation can be viewed as a projection onto the linear space spanned by the regressors. Source: Wikipedia

### 3.5 Chi-Squared Test of Independence and Relationship to GWAS

In biology, we can use the chi-squared test to determine if there is a relationship between a disease and a single nucleotide polymorphism (SNP) in the genome. In GWAS, we sequence the genome of a sample population, some with a disease and some without. We can then look for SNPs in the genome and determine the probability that a given SNP is associated with the disease of interest. Chi-squared tests produce the very popular Manhattan plot, where we can compare p-values for individual SNPs across the entire genome. An example Manhattan plot is shown here:

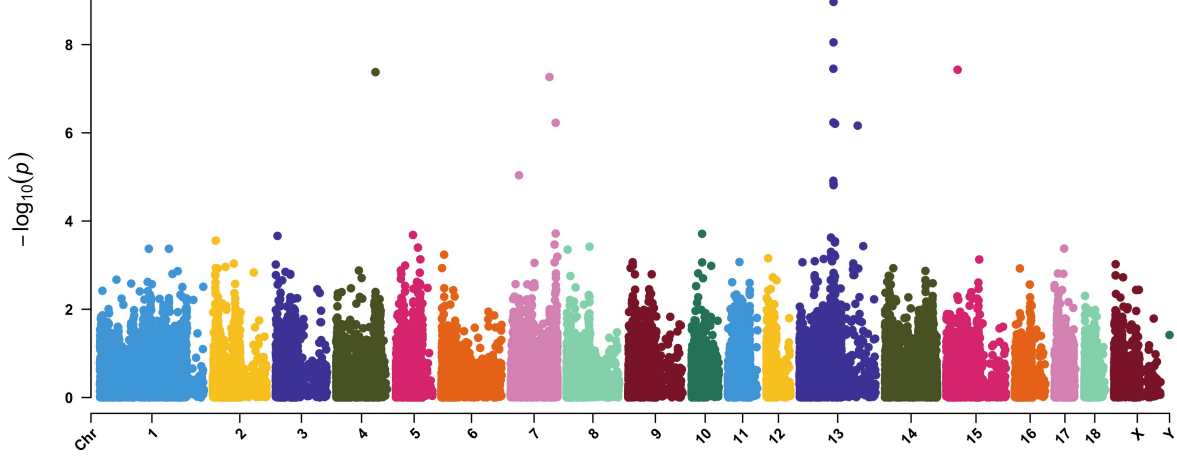


Figure 3.3: An example Manhattan plot

The p-value from the Chi-squared test is given on the y axis and each chromosome is laid out along the x axis. Every point represents a SNP, and the higher the p-value, the more significant its association with the disease of interest.

We will perform the Pearson's chi-squared test which is asymptotically equivalent to the likelihood ratio test. The chi-squared test allows you determine the difference between observed and expected data. We define Pearson's chi-squared statistic:

$$\chi^2 = \sum_{i=1}^I \sum_{j=1}^J \frac{(O_{ij} - E_{ij})^2}{E_{ij}}$$

Here  $O_{ij} = n_{ij}$  are the observed counts and  $E_{ij} = n\hat{\pi}_{ij} = \frac{n_{i.}n_{.j}}{n}$  are the expected counts under the null hypothesis.

Pearson's chi-squared statistic is then given by:

$$\chi^2 = \sum_{i=1}^I \sum_{j=1}^J \frac{(n_{ij} - n_{i.}n_{.j}/n)^2}{n_{i.}n_{.j}/n}$$

which is  $\chi^2$  distributed with  $k$  degrees of freedom. The degrees of freedom are the number of independent counts minus the number of independent parameters estimated from the data. We can calculate p-values from the chi-squared statistic, which can indicate whether a gene is associated with the phenotype of interest.

## Chapter 4

# Regulatory DNA, Transcription factors, Sequence motifs

### 4.1 Epigenome

Different cells with similar genome sequences have different genes expression. The epigenome can control gene activities to decide which genes are turned on or off.

“The epigenome is a multitude of chemical compounds that can tell the genome what to do. The epigenome is made up of chemical compounds and proteins that can attach to DNA and direct such actions as turning genes on or off, controlling the production of proteins in particular cells.”

—from genome.gov

### 4.2 Epigenomic marks

	Chemical compounds	Proteins	Other molecules
DNA-associated	DNA methylation	Histones; DNA-binding proteins (transcription factors*)	RNA(e.g., R loops)
Chromatin-associated	Histone modifications: methylations, acetylations, ...	Histone variants; Chromatin regulators; Histone modifying enzymes: writer, readers, erasers; Chromatin remodeling complexes	Non-coding RNAs

In addition to DNA-associated and chromatin-associated epigenomic marks, there are some other information served as epigenomic marks, such as nucleosome positioning, chromatin accessibility and 3D genome organization, etc.



## 4.3 Epigenomics

### 4.3.1 Transcription factors

- The transcription factors (TF) is one of the most important group of proteins which can directly interact with DNA. In this case, the DNA region should be open/accessible to proteins.
- The transcription factors should have DNA-binding domains which are used to recognize specific DNA sequences and sites. They also have effector domains which can regulate TF activity, such as ligand binding domains, can mediate protein-protein interactions, such as BTB domain, and can have enzymatic activities, such as SET domain.
- There are some functional studies related to transcription factors, such as studying for the cell-type specific gene expression, binding DNA sequence motif, genome-wide binding sites, target genes, TF co-factors, etc.

### 4.3.2 Sequence motif

In general, a motif is a distinctive pattern that occurs repeatedly. In biomolecular studies, a sequence motif is a pattern common to a set of DNA, RNA, or protein sequences that share a common biological property, such as functioning as binding sites for a particular protein.

### 4.3.3 Sequence motif finding

For motif finding, the input data is a set of DNA sequences and the output is enriched sequence patterns (motifs). - Motif representation

Single-letter and ambiguity codes for nucleotides (Table)

Symbol	Meaning	Origin of designation
G	G	<b>G</b> uanine
A	A	<b>A</b> denine
T	T	<b>T</b> hymine
C	C	<b>C</b> ytosine
R	G or A	pu <b>R</b> ine
Y	T or C	p <b>Y</b> rimidine
M	A or C	a <b>M</b> ino
K	G or T	<b>K</b> eto
S	G or C	<b>S</b> trong interaction (3H bonds)
W	A or T	<b>W</b> weak interaction (2H bonds)
H	A or C or T	not-G, H follows G in the alphabet
B	G or T or C	not-A, B follows A
V	G or C or A	not-T (not-U), V follows U
D	G or A or T	not-C, D follows C
N	G or A or T or C	a <b>N</b> y

It is derived by IUPAC. One limitation is that it can not measure continue relation/difference because it is a binary decision.

- Entropy

Entropy is used to measure the orderliness.

Boltzmann entropy:  $S_B = -k_B \sum_i p_i \ln(p_i)$

Shannon entropy:  $H(X) = -\sum_i P(x_i) \log_2 P(x_i)$

- Position weight matrix

Here are some DNA sequences. The first line is the position:

	1	2	3	4	5	6	7	8	9
G	A	G	G	T	A	A	A	C	
T	C	C	G	T	A	A	G	T	
C	A	G	G	T	T	G	G	A	
A	C	A	G	T	C	A	G	T	
T	A	G	G	T	C	A	T	T	
T	A	G	G	T	A	C	T	G	
A	T	G	G	T	A	A	C	T	
C	A	G	G	T	A	T	A	C	
T	G	T	G	T	G	A	G	T	
A	A	G	G	T	A	A	G	T	

In each position, we count the number of A, C, G, and T, respectively. Then we get the corresponding position frequency matrix (PFM):

$$M = \begin{matrix} & \begin{matrix} A \\ C \\ G \\ T \end{matrix} & \begin{bmatrix} 3 & 6 & 1 & 0 & 0 & 6 & 7 & 2 & 1 \\ 2 & 2 & 1 & 0 & 0 & 2 & 1 & 1 & 2 \\ 1 & 1 & 7 & 10 & 0 & 1 & 1 & 5 & 1 \\ 4 & 1 & 1 & 0 & 10 & 1 & 1 & 2 & 6 \end{bmatrix} \end{matrix}$$

Then we simply do a normalization. Each number is divided by the total number of sequences. The corresponding position probability matrix (PPM) is:

$$M = \begin{matrix} & \begin{matrix} A \\ C \\ G \\ T \end{matrix} & \begin{bmatrix} 0.3 & 0.6 & 0.1 & 0.0 & 0.0 & 0.6 & 0.7 & 0.2 & 0.1 \\ 0.2 & 0.2 & 0.1 & 0.0 & 0.0 & 0.2 & 0.1 & 0.1 & 0.2 \\ 0.1 & 0.1 & 0.7 & 1.0 & 0.0 & 0.1 & 0.1 & 0.5 & 0.1 \\ 0.4 & 0.1 & 0.1 & 0.0 & 1.0 & 0.1 & 0.1 & 0.2 & 0.6 \end{bmatrix} \end{matrix}$$

When we talk about TF binding sites or motifs, we always see some sequence logo. The sequence logo consists of stacks of symbols, one stack is for each position in the sequence. The overall height of the stack indicates the sequence conservation at that position (information content). The height of symbols within the stack indicates the relative frequency of nucleic acid at that position. For example, in this sequence logo, the height of each position is calculated as  $R_i = \log_2(4) - H_i$ , in which  $H_i = -\sum_k M_{k,i} \times \log_2 M_{k,i}$



Figure 4.1: Sequence logo

Given the PPM ( $M$ ) and a background model  $b$ , we can calculate the position weight matrix (PWM). In the PWM ( $M$ ),  $M_{k,i} = \log_2(M_{k,i}/b_k)$ , in which  $b = (b_1, b_2, b_3, b_4) = (p_A, p_C, p_G, p_T)$ . For nucleotides,  $b_k = 0.25$ . In general,  $b_k$  does not have to be equal for each symbol. For example, if the organisms we

studied with a high GC-content, the  $b_k$  for C and G will be higher than that for A and T. Besides, in practice, in order for convenience for calculation, we will give a pseudo count (such as 0.0001) to 0 to avoid the logarithm of 0.

- Motif matching score

Given the PPM and a background model  $b$ , we can also calculate the motif matching score using the likelihood ratio score. For example, the score for GAGGTAAAC =  $\log_2 \frac{p_G \times p_A \times p_G \times p_G \times p_T \times p_A \times p_A \times p_A \times p_C}{b_G \times b_A \times b_G \times b_G \times b_T \times b_A \times b_A \times b_A \times b_C}$ .

#### 4.3.4 De novo sequence motif finding

The goal of de novo sequence motif finding is to look for common sequence patterns enriched in the input data compared to a background genome. There are two kinds of approaches to do de novo sequence motif finding: deterministic approach and probabilistic approach.

##### 4.3.4.1 Deterministic approach

The deterministic approach is regular expression enumeration. The basic idea for this approach is to check over-representation for every  $w$ -mer by comparing observed  $w$  occurrence in data and expected  $w$  occurrence in data. The over-represented  $w$  is potential TF binding motif. The advantages of this approach are that it is exhaustive, can guarantee to find global optimum, and can find multiple motifs. As for disadvantages, one is that it is not as flexible as with base substitutions and long list of similar good motifs, and the other is that it's limited with motif width.

##### 4.3.4.2 Probabilistic approach

Different from deterministic approach which is pattern driven approach, the probabilistic approach is data driven approach. Expectation-Maximization (EM) approach and Gibbs Sampling are two probabilistic approaches. Here we talk about the EM approach.

The objects of this approach are as follows:  $seq$  is sequence data to search for motif;  $\theta_0$  is non-motif probability (genome background) parameter;  $\theta$  is motif probability matrix parameter;  $\pi$  is motif site location. The problem of this approach is to estimate  $P(\theta, \pi | seq, \theta_0)$ . The approach is to alternately estimate one of  $\pi$  and  $\theta$  each time by fixing the other, in which the two steps are called E-step and M-step respectively. Here is an example for this approach:

- E-step: given  $\theta_0$ ,  $seq$  and  $\theta$  to estimate  $\pi$ , in which  $\theta_0$  and  $seq$  are known while  $\theta$  is given a initial value. In alternative steps,  $\theta$  is calculated by M-step.

Given an example,  $\theta_0 : p_{0A} = 0.3, p_{0C} = 0.2, p_{0G} = 0.2, p_{0T} = 0.3$ .

$seq$  :

T	T	G	A	C	G	A	C	T	G	C	A	C	G	T	
T	T	G	A	C											LR <sub>1</sub>
	T	G	A	C	G										LR <sub>2</sub>
		G	A	C	G	A									LR <sub>3</sub>
			A	C	G	A	C								LR <sub>4</sub>
				C	G	A	C	T							LR <sub>5</sub>
					G	A	C	T	G						LR <sub>6</sub>
						A	C	T	G	C					LR <sub>7</sub>
							C	T	G	C	A				LR <sub>8</sub>
								.	.	.					

$\theta$ :

pos	A	C	G	T
1	0.7	0.1	0.01	0.2
2	0.01	0.01	0.8	0.1

pos	A	C	G	T
3	0.32	0.02	0.3	0.18
4	0.03	0.42	0.1	0.47
5	0.2	0.5	0.1	0.2

Then, for  $LR_1$ ,

$$\begin{aligned}
P(\text{TTGAC}|\theta_0) &= p_{0T} \times p_{0T} \times p_{0G} \times p_{0A} \times p_{0C} \\
&= 0.3 \times 0.3 \times 0.2 \times 0.3 \times 0.2 \\
&= 1.08 \times 10^{-3} \\
P(\text{TTGAC}|\theta) &= P(\text{T in pos1}) \times P(\text{T in pos2}) \times P(\text{G in pos3}) \times P(\text{A in pos4}) \times P(\text{C in pos5}) \\
&= 0.2 \times 0.1 \times 0.3 \times 0.03 \times 0.5 \\
&= 9 \times 10^{-5}
\end{aligned}$$

Therefore, the likelihood ratio of the first motif  $\pi_1$  is  $LR_1 = \frac{P(\text{TTGAC}|\theta)}{P(\text{TTGAC}|\theta_0)} = \frac{9 \times 10^{-5}}{1.08 \times 10^{-3}}$ . Then we can calculate  $LR_2, LR_3, LR_4$ , etc.

- M-step: given  $\theta_0$ ,  $seq$ , and  $\pi$  to estimate  $\theta$ , in which  $\theta_0$  and  $seq$  are known while  $\pi$  with its likelihood ratio  $LR$  is calculated by E-step.

Given an example,  $seq = \text{TTGACGACTGCACGT}$ ,  $\pi$  and its likelihood ratio  $LR$  are:

$\pi$	$LR$
TTGAC	0.8
TGACG	0.2
GACGA	0.6
ACGAC	0.5
CGACT	0.3
GACTG	0.7
ACTGC	0.4
CTGCA	0.1
TGCAC	0.9
...	...

Then we can update  $\theta$  by estimate the probability of A, C, G, T in any of the 5 positions (5 is the length of the motif):

$$\begin{aligned}
P(\text{T in pos1}) &= \frac{0.8 + 0.2 + 0.9 + \dots}{0.8 + 0.2 + 0.6 + 0.5 + 0.3 + 0.7 + 0.4 + 0.1 + 0.9 + \dots} \\
P(\text{T in pos2}) &= \frac{0.8 + 0.1 + \dots}{0.8 + 0.2 + 0.6 + 0.5 + 0.3 + 0.7 + 0.4 + 0.1 + 0.9 + \dots} \\
P(\text{G in pos2}) &= \frac{0.2 + 0.3 + 0.9 + \dots}{0.8 + 0.2 + 0.6 + 0.5 + 0.3 + 0.7 + 0.4 + 0.1 + 0.9 + \dots} \\
P(\text{C in pos5}) &= \frac{0.8 + 0.5 + 0.4 + 0.9 + \dots}{0.8 + 0.2 + 0.6 + 0.5 + 0.3 + 0.7 + 0.4 + 0.1 + 0.9 + \dots} \\
&\quad \dots \dots \dots
\end{aligned}$$

After we get the updated  $\theta$  from the M-step, we can re calculate the E-step. Iterate the E-step and M-step until  $\theta$  does not improve. Then we can find the most frequent k-mers by calculate the likelihood ratio of each  $\pi$ .

# Chapter 5

## ChIP-seq

The goal of ChIP-seq is to determine the locations in the genome associating with a protein factor

### 5.1 ChIP-seq experiment steps:

1. Chromatin ImmunoPrecipitation (ChIP)
2. Protein-DNA crosslinking with formaldehyde (for TF)
3. Chop the chromatin using sonication (TF) or micrococcal nuclease (MNase) digestion (histone)
4. Specific factor-targeting antibody
5. Immunoprecipitation
6. DNA purification
7. PCR amplification (~150bp)
8. High-throughput sequencing (Illumina: can only sequencing the end of the DNA fragments)

### 5.2 History of the development of ChIP-seq technology

- UV crosslinking (1984) : the protein-DNA interaction can be captured
- Crosslinking + immunoprecipitation (1993) : use antibody to grab the DNA-protein complex
- ChIP-chip (2000) : **genomewide** microarray method was developed, using a **pre-designed** way
- Unbiased chromosomal coverage by tiling array (2004)
- ChIP-seq (2007)

Today, ChIP-seq has become the predominant method for profiling chromatin epigenomes.

### 5.3 ChIP-seq data analysis

The analysis aims to achieve the following goals: - Where in the genome do these sequence reads come from? This is accomplished using sequence alignment after quality control - What does the enrichment of sequences mean? Accomplished using peak calling - What can we learn from these data? This requires further downstream analysis and integration

Here is a brief outline of steps that are required to achieve those goals:

1. Sequencing quality assessment using fastqc. If the quality scores across bases fail, either re-do the experiment or trim the data.
2. ChIP-seq read mapping: map the fastq file containing the sequence information to the genome; alignment of each sequence read: bowtie, BWA (Burrows-Wheeler Algorithm); usually use the reads can map to a unique/best location in the genome.

3. Redundancy control: completely identical reads are considered error (for example, induced by PCR)
  - Non-redundant rate: The ratio of the number of non-redundant reads to the number of mapped reads
  - PBC (PCR Bottleneck Coefficient): The ratio of the number of locations with 1 read mapped to the number of locations with reads mapped
4. DNA fragment size estimation:
  - peak model (MACS) for TF
  - cross-correlation (SICER) for any ChIP-seq (input): calculate the Correlation between two strings with a displacement; Auto-correlation: Cross-correlation with itself
5. Retrieve DNA fragments
  - Full length retrieval (MACS)
  - Partial retrieval (sharpen the signal)
  - Point retrieval (SICER)
6. Pile up: Signal map generation

## 5.4 ChIP-seq: Study design

Background Control: Input or IgG - Input chromatin: sonicated/digested chromatin without immunoprecipitation - IgG: “unspecific” immunoprecipitation

Study Control: - Control exp sample: ChIP + input - Treated exp sample: ChIP + input

## 5.5 ChIP-seq: Peak calling

Goal: Identify regions in the genome enriched for sequence reads: - Compared to genomic background - Compared to input control

MACS: Model-based Analysis for ChIP-Seq Read distribution along the genome - Poisson distribution ( $\lambda BG = \text{total tag} / \text{genome size}$ ) - Negative binomial distribution (MACS2)

ChIP-seq show local biases in the genome - Chromatin and sequencing bias - 200-300 bp control windows have too few tags - But can look further

B-H adjustment to correct for FDR : p-value  $\rightarrow$  q-value

Data Visualization - bedGraph to bigWig - macs2 output data - IGV

Quality Control - FRiP(FractionofReadsinPeaks)score - 1-10% for TF is normal - Numberofpeaks - Number of peaks with high fold-enrichment, e.g, 5, 10, ... - 2000 - Sequenceconservation - Fractionofpeakswithinregulatoryregions - 80%

Biological interpretation: ChIP-seq captures a snapshot of binding patterns from a cell population - TF intrinsic property - Binding activity - Cellular heterogeneity

## Chapter 6

# Epigenomic domains, Hierarchy and scales of genome structure

### 6.1 Epigenomics and chromatin structure

- Interest in epigenomic factors that interact with genetic DNA.
- DNA is wrapped around histone octamers (2 histones of: H2A, H2B, H3, H4) called **nucleosomes** (core particles)
- Each nucleosome is wrapped by 147 bp of DNA.
- Histone tails have residues that can be modified (chemical modification/variants)
- Covalent modifications on histone tails include: methylation (me), acetylation (ac), phosphorylation, etc.
- Histone modifications play an important role in gene transcription since they can define areas of repression/activation of genes.

### 6.2 Functional annotation of common histone marks

Functional Annotation	Histone Marks
Promoters	H3K4me3
Bivalent/Poised Promoter	H3K4me3/H3K27me3
Transcribed gene body	H3K36me3
Enhancer (active and poised)	H3K4me1
Active Enhancer	H3K4me1/H3K27ac
Polycomb Repressed Regions	H3K27me3
Heterochromatin	H3K9me3

Extra notes:

- Bivalent domains are mainly seen in embryonic stem cells. These are primed with H3K4me3 an activating mark for genes. This allows them to differentiate into their specific cell types. H3K27me3 is also present to repress transcription as needed.
- H3K36me3 is mainly found in exons which suggests that splicing functions are also coded in the epigenome

## 6.3 Comparing transcription factors and histone marks

### Cell Type Specificity:

- Transcription Factors: Cell type specific. They also have their own profile where they can bind in DNA.
- Histone marks: They only have a profile. They're not cell type specific. Most of them can be found in all cell types. (with exceptions)

### Signal width

- Transcription Factors: Narrow (signals are in 'peaks')
- Histone marks: Marks can be narrow (e.g. H3K4me3) or broad (e.g. H3K9me3)

### Chromatin accessibility

- Transcription Factors: They need chromatin to be open to be able to bind to their specific sequence motif in DNA
- Histone marks: Can be found in both open/closed chromatin

### DNA sequence motif

- Transcription Factors: Have a specific sequence motif
- Histone Marks: They don't have sequence specificity they look for their histone tail residues

### Resolution

- Transcription Factors: 1-10 bp (we can accurately determine their position in the genome)
- Histone marks: 200 bp. This is based on the nucleosome that is composed of 147 bp and linker DNA that is ~50 bp

## 6.4 Histone modification patterns

- These patterns tend to be diffuse, noisy, hard to see, enriched in spread-out regions and lack saturation
- These patterns tend to be broad because once a nucleosome is modified it's a chain reaction to proximal nucleosomes. If a complex is attracted to one nucleosome (nucleation) it will start binding the adjacent nucleosomes creating chromatin regions of open/closed marks (propagation). (e.g. HP1 and H3K9me3, PRC1/PRC2 and H3K27me3)

## 6.5 Chip-seq analysis (continued)

### Chip-seq pipeline

#### Data preprocessing comparison

	MACS	SICER
DNA fragment size estimation	Peak model (optimal for TFs)	Cross-correlation (optimal for histone modifications)
DNA fragment retrieval	Full length (extend d) (fragment length)	Point position (shift d/2) Actual center location of fragment
Signal profile generation	Fragment pile up (it's easy because TF are very enriched)	Read count in bins (Bin the entire genome with no overlaps. Each bin is 200 bp=nucleosome. Reads are counted per bin)

### Detecting the signal



	MACS	SICER
Initial model	Poisson	Poisson
Signal scan	Sliding windows with bandwidth	Non-overlapping bin read count
Peak region identification	Merge windows	Merge windows allowing gaps
Peak scoring	Pile-up signal amount (#reads*fragment size)	Aggregate score on islands
Significance modeling	Poisson with dynamic $\lambda$	Asymptotic estimation of island score statistics model, then compare with control
Additional information	Read count, Pile-up height, Summit position	Read count, peak score, E-value

#### Extra Notes:

- Poisson distribution is selected to separate the real signal from the random background. It takes into account the randomness of looking at 10's millions of bins in the genome and determine the signal from background. *Null hypothesis*: All reads are randomly distributed in the genome. From this distribution a cut-off is determined to differentiate between noise and statistically significant signal.

## 6.6 SICER

### ISLANDS

- To determine the islands a Poisson distribution model is selected with very lenient parameters. This is a first pre-selection step which doesn't need to be as stringent. It'll only give some statistical significance to start differentiating signal from noise.
- There are eligible and ineligible windows.
- Eligible windows are separated by *gaps* of ineligible windows
- Islands are defined as the clusters of eligible windows that are separated by gaps of size at most  $g$  windows.

### Scoring Islands

- This is based on the probability of finding the observed tag count in a random background
- Having  $m$  reads in a window the probability of finding  $m$  reads in Poisson ( $m, \lambda$ ) is the average number of reads in each window
- Eligible windows are scored with the function:  $S = -\ln P(m, \lambda)$
- This score will aggregate all the eligible windows in the island and corresponds to the background probability of finding the observed pattern

*Scales of histone mark islands and chromatin domains:*

- Narrow: a few nucleosomes, 0.5kb~5kb, e.g. H3Kme4
- Broad: 5kb~100kb, Gene Loci, chromatin domains, super-enhancers, e.g. H3K4me1, H3K27ac, H3K36me3, H3K27me3, etc.
- Very Broad: > 100kb, Large chromatin domains, chromatin compartments, e.g. H3K9me3, H3K27me3

### SICER Statistics

- The statistics of SICER heavily rely on recursion methods.
- To find an island there has to be a gap on both sides of the extremes.
- It follows an asymptotic distribution in the background, modeled by an exponential distribution.
- Advantage: Since it's not looking for peaks but islands then it's able to capture broad areas that are enriched. Since various nucleosomes that are close together are usually enriched with the same histone

marks then these islands can be 5-100kb in size. (Gene loci, chromatin domain, super-enhancers). This makes this method **SCALE FREE**.

### Other approaches for chromatin domains

- ChromHMM: Hidden Markov Models (Ernst & Kellis) -> It relies on a chromatin region state and the transition between states. Limitation: not scale free which doesn't account for different sized regions/islands
- Recognizer: Coarse-graining (Zang, et al. 2020) -> Includes scale free analysis. It does block transformation under a majority rule, this is recursive and then there's a trace back to identify candidate enriched regions.
- Scale free chromatin domains follow the power law (This can be seen when plotting the frequency of islands to width (bp) of islands)

## 6.7 Hi-C

Experiment Pipeline:

Processing Pipeline:

Analysis:

- Using a PCA Analysis the first 2 components are selected to separate chromatin into two big compartments: open and closed chromatin.
- Topologically Associating Domains (TADs): These are domains with high probability of interactions.
- 2-D map depicting the density or number of reads between each part of the chromatin

Genome wide contact maps are created to visualize the results. It is expected to have high enrichment in the diagonal because there's interactions between close nucleosomes.

Some interpretations of Hi-C results:

### Hi-C Follows the power-law property

- From 500kb to a few megabites contact probability observed follows a linear model with  $m=-1$ . The theoretical calculation gives an  $m=-1.5$ . This suggests that intrachromosomal contact probability follows a fractal structure.
- It is proposed that chromatin is organized like spaghetti. The only difference is that chromatin does not have knots. This is because all areas have to have the same probability of interacting with its surrounding. Also, when chromosomes are going to split in mitosis chromatin has to be able to separate and not be knotted.
- The fractal structure compared to the equilibrium allows for clear organization of chromatin domains and going down at the sub scale this way of organization is also maintained.
- A better proposed model is that chromatin is like an instant noodle:

## Chapter 7

# Genomic Intervals: formats, data structures and algorithms

### 7.1 Overview

Technological advances and lower sequencing costs have led to rapidly increasing epigenomic data generation, such as ChIP-seq and ATAC-seq experiments. This data can be used to dissect regulatory networks, annotate genetic variation, understand cellular differentiation, and more. Data from epigenome experiments is often presented as genomic intervals, which are then used for many types of downstream analysis. A **genomic interval** is a consecutive stretch on a genomic sequence defined by a start and end.

### 7.2 WHAT CAN BE REPRESENTED AS AN INTERVAL?

Because of the linear nature of DNA and RNA, many biological entities can be conceptualized as genomic intervals.

**ChIP-seq or ATAC-seq peaks:** The results of peak calling are intervals that were derived from the ChIP-seq or ATAC-seq experiments. Peak calling is a computational method used to identify areas in a genome that have been enriched with aligned reads as a consequence of ChIP-seq or ATAC-seq experiment.

**Single-Nucleotide Polymorphisms (SNPs):** SNPs can be represented as a interval of length of 1.

**Genes and gene components:** Genes and other genomic components such as TSS, exons, introns, and transcripts can be represent as intervals defined by the their start and end positions.

**Non-coding DNA annotation:** Intervals can be used for non-coding DNA annotation.

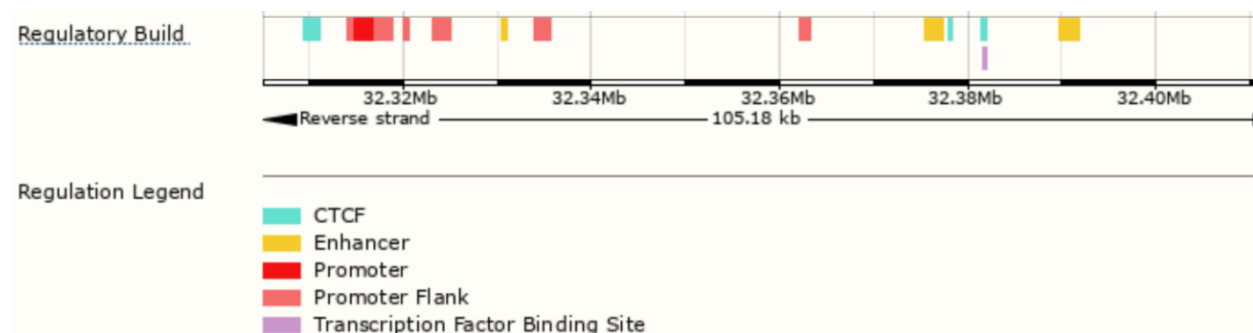


Figure 7.1: Non Coding DNA annotations

**Chromosomes:** Chromosomes can be represent as intervals define by the chromosome lengths (e.g. chromosome sizes file)

**Aligned sequence reads:** The locations of aligned reads resulted from any experiments can be represent as intervals. The interval representation can compress sequence data with sequence coordinates.

**Protein domains:** Protein domains can be represented as intervals on protein sequences (proteomic interval).

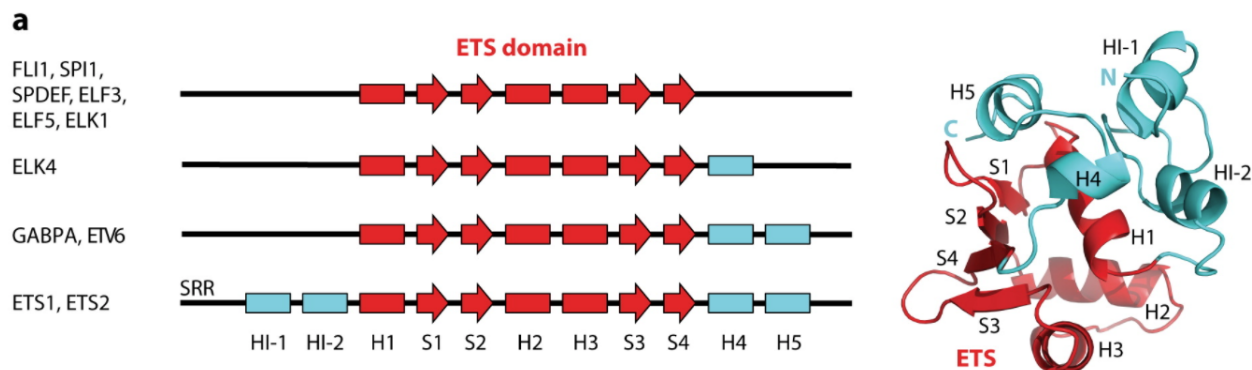


Figure 7.2: Protein domains

Genomic intervals are often a simplified abstraction of genomic sequence. And, interval operations are fundamental in genomics.

## 7.3 FILE FORMATS FOR GENOMIC INTERVALS

### 7.3.1 BED

BED (Browser Extensible Data) format provides a flexible way to define the data lines that are displayed in an annotation track. BED lines have three required fields and nine additional optional fields.

#### 7.3.1.1 The three required BED fields are:

**chrom** - The name of the chromosome (e.g. chr3, chrY, chr2\_random) or scaffold (e.g. scaffold10671).

**chromStart** - The starting position of the feature in the chromosome or scaffold. (0-based vs 1-based coordinate system)

**chromEnd** - The ending position of the feature in the chromosome or scaffold. (open vs closed interval)

#### 7.3.1.2 The 9 optional BED fields are:

**name** - Defines the name of the BED line. This label is displayed to the left of the BED line in the Genome Browser window when the track is open to full display mode or directly to the left of the item in pack mode.

**score** - A score between 0 and 1000. If the track line useScore attribute is set to 1 for this annotation data set, the score value will determine the level of gray in which this feature is displayed (higher numbers = darker gray). This table shows the Genome Browser's translation of BED score values into shades of gray:

shade									
score in range	≤ 166	167-277	278-388	389-499	500-611	612-722	723-833	834-944	≥ 945

Figure 7.3: Genome Browser translates BED score values into shades of gray

**strand** - Defines the strand. Either “.” (no strand) or “+” or “-”.

**thickStart** - The starting position at which the feature is drawn thickly (for example, the start codon in gene displays). When there is no thick part, thickStart and thickEnd are usually set to the chromStart position.

**thickEnd** - The ending position at which the feature is drawn thickly (for example the stop codon in gene displays).

**itemRgb** - An RGB value of the form R,G,B (e.g. 255,0,0). If the track line itemRgb attribute is set to “On”, this RGB value will determine the display color of the data contained in this BED line. NOTE: It is recommended that a simple color scheme (eight colors or less) be used with this attribute to avoid overwhelming the color resources of the Genome Browser and your Internet browser.

**blockCount** - The number of blocks (exons) in the BED line.

**blockSizes** - A comma-separated list of the block sizes. The number of items in this list should correspond to blockCount.

**blockStarts** - A comma-separated list of block starts. All of the blockStart positions should be calculated relative to chromStart. The number of items in this list should correspond to blockCount.

The number of fields per line must be consistent throughout any single set of data in an annotation track. The order of the optional fields is binding: lower-numbered fields must always be populated if higher-numbered fields are used. BED3 format only have the 3 required fields; BED6 have the additional name, score, and strand field. The BED format was initially used for visualization in UCSC Genome Browser. The visualization related BED fields are: thickStart, thickEnd, itemRgb, blockCount, blockSizes, and blockStarts.

### 7.3.2 NARROWPEAK (BED-like)

This format is used to provide called peaks of signal enrichment based on pooled, normalized (interpreted) data from experiments such as ChIP-seq and ATAC-seq. The ENCODE narrowPeak is a BED6+4 format. The 4 fields in addition to the BED6 format are:

**signalValue** - Measurement of overall (usually, average) enrichment for the region.

**pValue** - Measurement of statistical significance (-log10). Use -1 if no pValue is assigned.

**qValue** - Measurement of statistical significance using false discovery rate (-log10). Use -1 if no qValue is assigned.

**peak** - Point-source called for this peak; 0-based offset from chromStart. Use -1 if no point-source called.

### 7.3.3 BROADPEAK (BED-like)

This format is used to provide called regions of signal enrichment based on pooled, normalized (interpreted) data. The ENCODE broadPeak is a a BED 6+3 format (without the **peak** filed in the narrowPeak format).

### 7.3.4 GFF

All GFF (General feature format) formats (GFF2, GFF3 and GTF) are tab delimited with 9 fields per line. They all share the same structure for the first 7 fields, while differing in the content and format of the ninth field. Some field names have been changed in GFF3 to avoid confusion. For example, the “seqid” field was formerly referred to as “sequence”, which may be confused with a nucleotide or amino acid chain. The general structure is as follows:

**seqname/seqid** - The name of the sequence. Must be a chromosome or scaffold.

**source** - The program that generated this feature.

**feature** - The name of this type of feature. Some examples of standard feature types are “CDS” “start\_codon” “stop\_codon” and “exon”li>

**start** - The starting position of the feature in the sequence.

**end** - The ending position of the feature.

**score** - A score between 0 and 1000. If the track line useScore attribute is set to 1 for this annotation data set, the score value will determine the level of gray in which this feature is displayed (higher numbers = darker gray). If there is no score value, enter “.”.

**strand** - Valid entries include “+”, “-”, or “.”

**frame** - If the feature is a coding exon, frame should be a number between 0-2 that represents the reading frame of the first base. If the feature is not a coding exon, the value should be “.”.

**group** - All lines with the same group are linked together into a single item.

## 7.4 RELATED FILE FORMATS

**Wiggle** files and its **bedgraph** variant allow you to plot quantitative data as either shades of color (dense mode) or bars of varying height (full and pack mode) on the genome.

### 7.4.1 WIGGLE

Wiggle format is line-oriented. For wiggle custom tracks, the first line must be a track definition line (i.e., track type=wiggle\_0), which designates the track as a wiggle track and adds a number of options for controlling the default display. Wiggle format is for continuous-valued data with constant size.

**Fixed-step:** This format is used for data with regular intervals between new data values and is the more compact wiggle format. After the wiggle track definition line, fixed-step begins with a declaration line and is followed by a single column of data values. The declaration line starts with the word fixedStep and includes specifications for chromosome, start coordinate, and step size. The optional span parameter (default: span=1) allows data composed of contiguous runs of bases with the same data value to be specified more succinctly. The span begins at each chromosome position specified and indicates the number of bases that data value should cover.

*format:*

```
fixedStep chrom=chrN start=position step=stepInterval [span=windowSize]
dataValue1
dataValue2
...
```

Figure 7.4: fixedWiggle format

*example:*

```
fixedStep chrom=chr3 start=400601 step=100
11
22
33
```

Figure 7.5: fixedWiggle example

**Variable-step:** This format is used for data with irregular intervals between new data points. After the wiggle track definition line, variable-step begins with a declaration line and is followed by two columns containing chromosome positions and data value. The declaration line starts with the word variable-step

and is followed by a specification for a chromosome. The span specification has the same meaning as in fixed-step format.

*format:*

```
variableStep chrom=chrN [span=windowSize]
chromStartA  dataValueA
chromStartB  dataValueB
... etc ...  ... etc ...
```

Figure 7.6: varWiggle Format

*example:*

```
variableStep chrom=chr2
300701 12.5
300702 12.5
300703 12.5
300704 12.5
300705 12.5
```

Figure 7.7: varWiggle example

## 7.4.2 BEDGRAPH

The bedGraph format allows display of continuous-valued data in track format. This display type is useful for probability scores and transcriptome data.

```
track type=bedGraph
chr19 49302000 49302300 -1.0
chr19 49302300 49302600 -0.75
chr19 49302600 49302900 -0.50
chr19 49302900 49303200 -0.25
chr19 49303200 49303500 0.0
chr19 49303500 49303800 0.25
```

Figure 7.8: bedGraph example

## 7.4.3 WHY SO MANY FORMATS?

Different format can store the same data (wide vs tall structure). But based on the sparsity of the data, different format can store the data with different efficiency. Different format also store data with different processing level (raw signal vs. summary signal vs. regions)

## 7.5 BIOLOGICAL QUESTIONS

Analysing genomic interval data can answer biological questions such as:

- Is SNP  $s$  located in a promoter element?
- What is the nearest gene downstream of SNP  $s$ ?
- What is the average distance to genes for SNP set  $S$ ?
- Which TF binding sites overlap region  $r$ ?
- Are binding sites of TF  $y$  enriched in enhancers?
- How many sites of TF  $y$  are within 10 kb of gene  $z$ ?
- What is the shape of signal across interval set  $I$ ?
- What is the frequency of annotations for interval set  $I$ ?

- How many aligned reads mapped to gene  $g$ ? (RNA-seq expression quantification)

## 7.6 INTERVAL COMPUTATIONS

An **interval comparison** assumes a homology statement. A shared **coordinate system** is a prerequisite for interval comparison. Intervals with different reference genome assemblies are not comparable.

### 7.6.1 OVERLAP

Assuming the intervals are well-formed, which the position of start is less than the end ( $X_{start} < X_{end}$ ), and the intervals are ordered by start ( $A_{start} < B_{start}$ ), interval  $A$  overlapping with  $B$  if  $A_{end} > B_{start}$

### 7.6.2 COUNTING OVERLAP

To compute the degree of overlap, assuming there is no containment between interval  $A$  and  $B$ :

$$overlap = A_{end} - B_{start}$$

or

$$overlap = (A_{end} - B_{start}) - \min(0, (A_{end} - B_{end}))$$

### 7.6.3 INTERSECTING REGION

The intersecting region of interval  $A$  and  $B$ ,  $R = A \cap B$ , where:

$$R_{start} = \max(A_{start}, B_{start}) \quad R_{end} = \min(A_{end}, B_{end})$$

If  $\max(A_{start}, B_{start}) > \min(A_{end}, B_{end})$ , then interval  $A$  and  $B$  are not overlapping.

### 7.6.4 UNION

To identify the union of two intervals  $A$  and  $B$ , first, we need to confirm they intersect,  $A \cap B$ . Then,  $R = A \cup B$ , where:

$$R_{start} = \min(A_{start}, B_{start}) \quad R_{end} = \max(A_{end}, B_{end})$$

### 7.6.5 INTERVAL SET OVERLAP COUNTING

By counting the number of intervals in  $\mathbf{I}$  overlapping  $\mathbf{q}$ , where  $\mathbf{I}$  is a set of intervals and  $\mathbf{q}$  is the query interval, we can answer questions such as the RNA-seq expression level of gene  $g$  a set of intervals. We can do that by looping through the list of intervals:

```
1 def overlap(query, i):
2     return query.start < i.end && query.end > i.start
3
4 def count_overlaps(query, intervals_set):
5     n_overlaps = 0
6     for i in interval_set:
7         if overlap(query, i):
8             n_overlaps = n_overlaps + 1
9     return n_overlaps
```

Figure 7.9: Linear search

However, we may have billions of intervals to loop through, which make  $\mathbf{O(N)}$  comparisons where  $N$  is the number of intervals. We can improve the performance with **binary search**:



```
def binary_search(array, query, low, high):
    pivot = int((low + high)/2)
    if (low >= high):
        return False
    if (array[pivot] == query):
        return pivot
    if (array[pivot] > query):
        return binary_search(array, query, low, pivot-1)
    else:
        return binary_search(array, query, pivot+1, high)
```

Figure 7.10: Binary search

## 7.7 BINARY SEARCH

Binary search is a search algorithm that finds the position of a target value within a **sorted** array (ordered intervals). Binary search compares the target value to the middle element of the array (pivot). If they are not equal, the half in which the target cannot lie is eliminated and the search continues on the remaining half, again taking the middle element to compare to the target value, and repeating this until the target value is found. If the search ends with the remaining half being empty, the target is not in the array. Binary search runs in logarithmic time in the worst case, making  $O(\log N)$  comparisons, where  $N$  is the number of elements in the array.

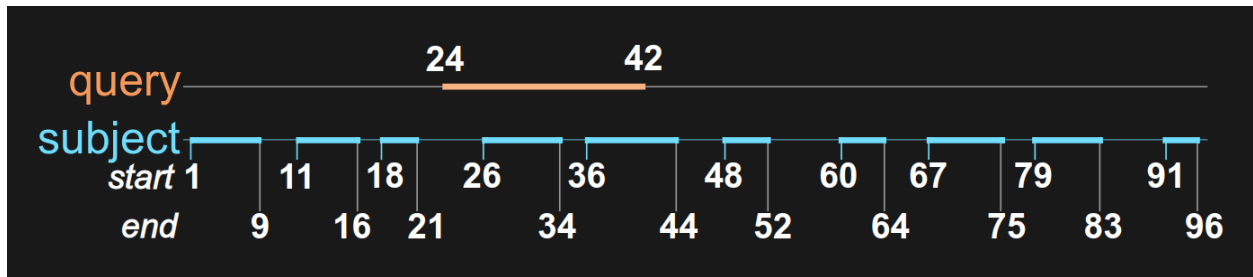


Figure 7.11: binarySearchExp1

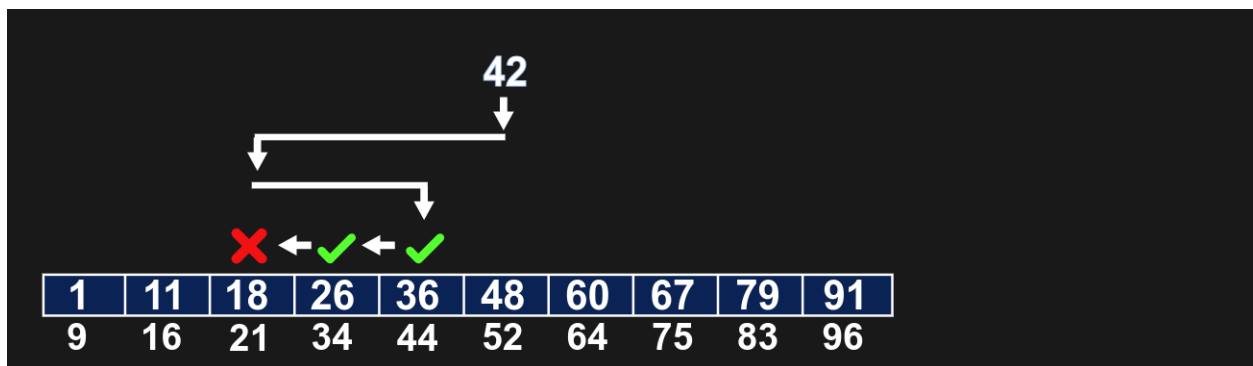


Figure 7.12: binarySearchExp2

However, we can not do binary search on a BED file with sorted regions because binary search also requires random access. It requires the ability to access any element directly. Indexing would provide random access to the file.

## 7.8 BINARY SEARCH TREES

Binary search tree (BST) is a rooted binary tree data structure whose internal nodes each store a key greater than all the keys in the node's left subtree and less than those in its right subtree. Insertion in a sorted array is  $O(N)$ . BST have  $O(\log N)$  search and insertion, where  $N$  is the number of nodes. BST do not require random access for binary search.

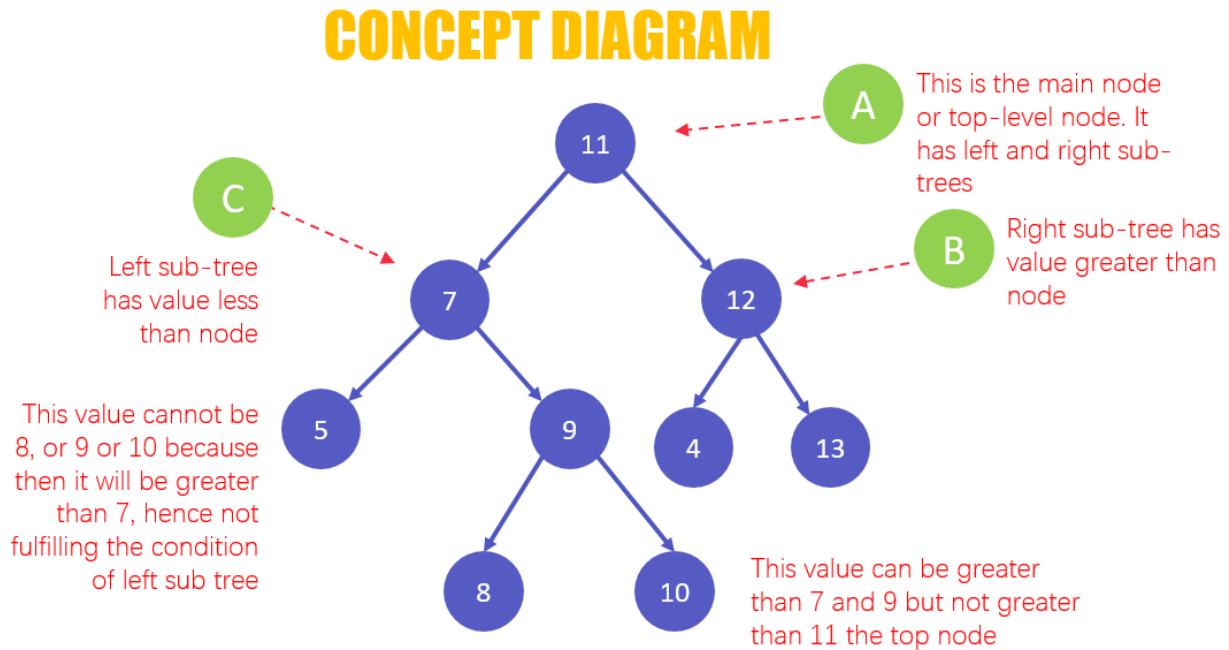
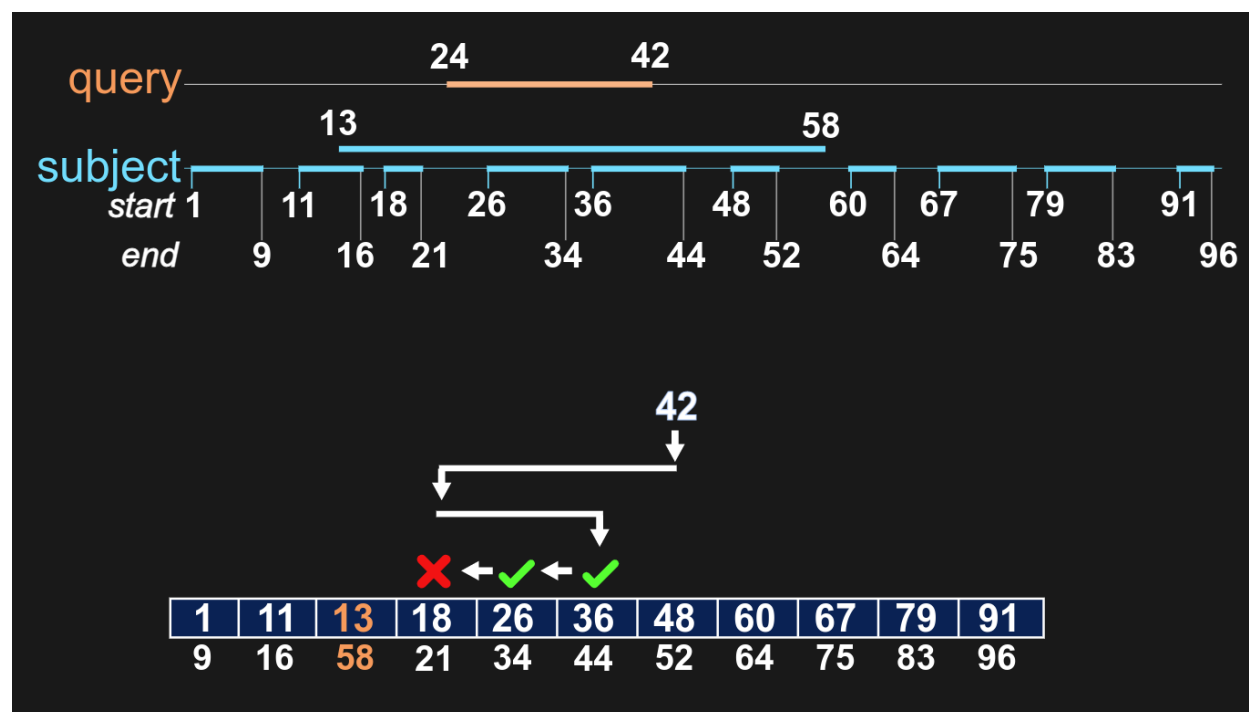


Figure 7.13: BST

### 7.8.1 Hidden assumption for BST: No containment



There are advanced data structures that addressing the containment problem.

## 7.9 ADVANCED DATA STRUCTURES

### 7.9.1 B trees

The B-tree generalizes the binary search tree, allowing for nodes with more than two children. The node values of a B-tree are boundaries for subtrees, which provide overlap constraints.

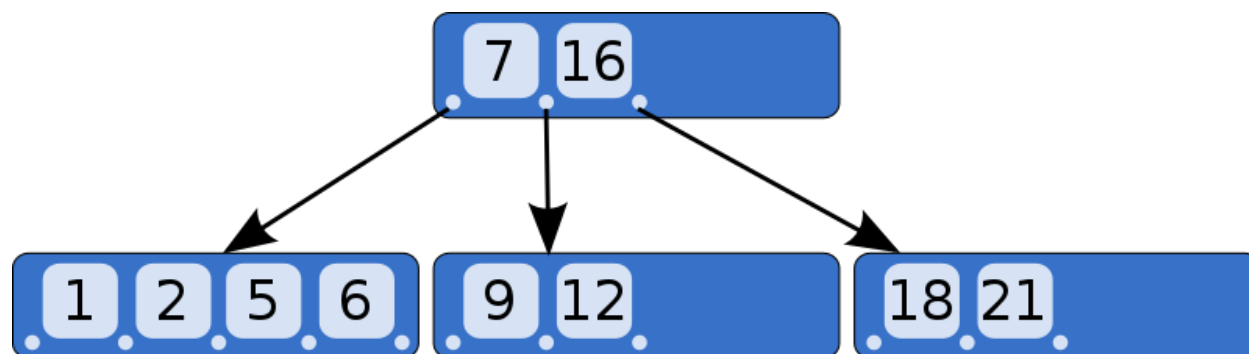


Figure 7.14: BTree

### 7.9.2 R trees

The R-tree was proposed by Antonin Guttman in 1984. R-tree annotates tree nodes with a minimum bounding rectangle of elements. A query that does not intersect the bounding rectangle will not intersect any child element. R-tree is used by Bedtools for detecting overlaps.

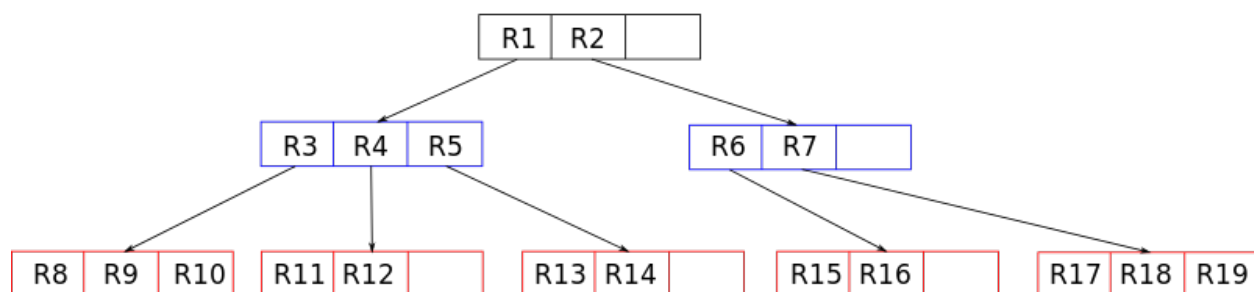
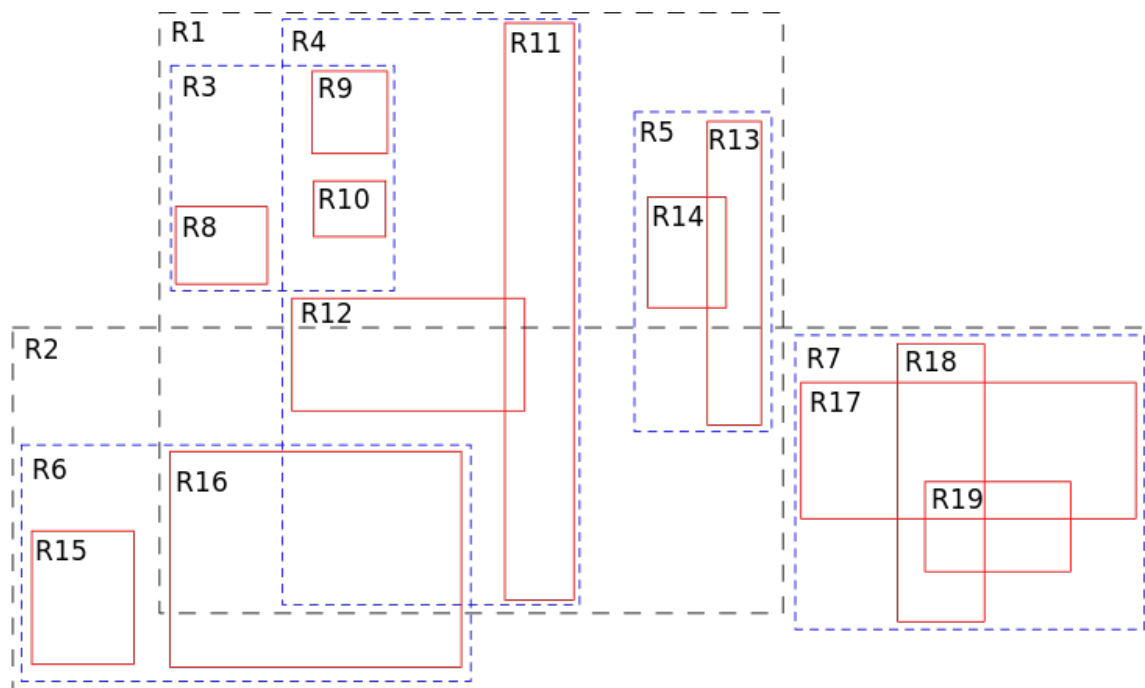


Figure 7.15: RTree

### 7.9.3 Nested containment lists (NList)

A NList is a datastructure that can be queried for elements overlapping intervals. It was invented and published by Alexander V and Alekseyenko Christopher J. Lee in Bioinformatics in 2007. The NList internals rely on the observation that when a set of intervals, where all are non-contained (based on their interval bounds) in any of the other intervals in the set, are sorted on their start coordinate are also sorted on their end coordinate. If this requirement is fulfilled the items overlapping an interval can be found by a binary search on the query start and returning items until the query end coordinate has been passed, giving a complexity of  $O(\log(N) + M)$  where  $N$  is the size of the set and  $M$  is the number of overlaps. NList is used by Bioconductor GenomicRanges class.

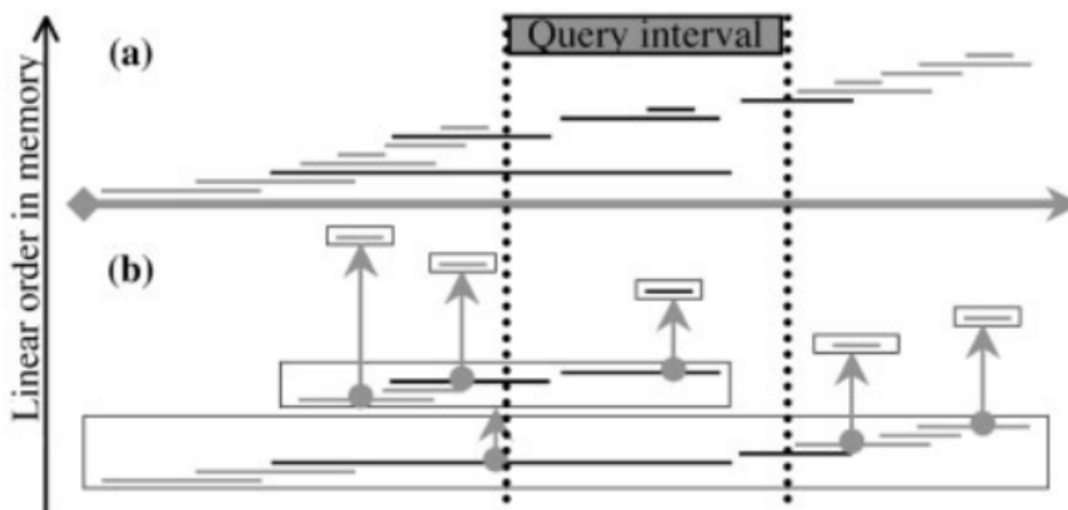


Figure 7.16: NList

### 7.9.4 Augmented interval list (AList)

An AList is constructed by first sorting the interval set,  $R$ , as a list by the start coordinate, then decomposing it into a few approximately flattened components (sublists), and then augmenting each sublist with the running maximum interval end. The query time for AList is  $O(\log 2N + n + m)$ , where  $n$  is the number of overlaps between  $R$  and the query interval,  $q$ ,  $N$  is the number of intervals in the set  $R$  and  $m$  is the average number of extra comparisons required to find the  $n$  overlaps.

## 7.10 INTRO TO BIOCONDUCTOR GENOMIC RANGES

The GenomicRanges is a R/Bioconductor package that defines general purpose containers for storing and manipulating genomic intervals and variables defined along a genome.

To install GenomicRanges:

```
if (!require("BiocManager"))  
  install.packages("BiocManager")  
BiocManager::install("GenomicRanges")
```

To define a GRange object:

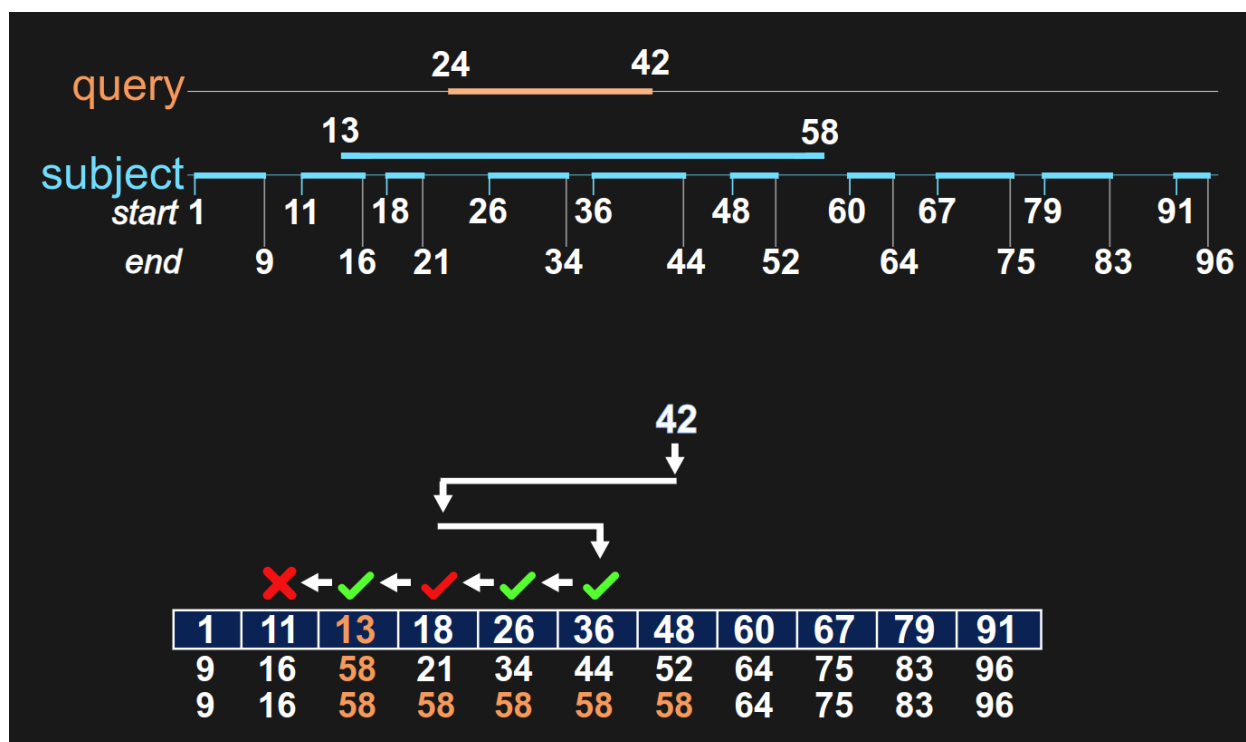


Figure 7.17: AIList

```
gr <- GRanges(
  seqnames = Rle(c("chr1", "chr2", "chr1", "chr3"), c(1, 3, 2, 4)),
  ranges = IRanges(101:110, end = 111:120, names = head(letters, 10)),
  strand = Rle(strand(c("-", "+", "*", "+", "-")), c(1, 2, 2, 3, 2)),
  score = 1:10,
  GC = seq(1, 0, length=10))
```

The components of the genomic coordinates within a GRanges object can be extracted using the `seqnames`, `ranges`, and `strand` accessor functions:

```
seqnames(gr)
```

```
## factor-Rle of length 10 with 4 runs
## Lengths: 1 3 2 4
## Values : chr1 chr2 chr1 chr3
## Levels(3): chr1 chr2 chr3
```

```
ranges(gr)
```

```
## IRanges object with 10 ranges and 0 metadata columns:
## start end width
## <integer> <integer> <integer>
## a 101 111 11
## b 102 112 11
## c 103 113 11
## d 104 114 11
## e 105 115 11
## f 106 116 11
```

```
##   g      107      117      11
##   h      108      118      11
##   i      109      119      11
##   j      110      120      11
```

```
strand(gr)
```

```
## factor-Rle of length 10 with 5 runs
##   Lengths: 1 2 2 3 2
##   Values  : - + * + -
## Levels(3): + - *
```

GRanges objects can be divided into groups using the `split` method:

```
sp <- split(gr, rep(1:2, each=5))
sp
```

```
## GRangesList object of length 2:
## $`1`
## GRanges object with 5 ranges and 2 metadata columns:
##      seqnames      ranges strand |      score      GC
##      <Rle> <IRanges> <Rle> | <integer> <numeric>
##   a      chr1    101-111      - |          1 1.000000
##   b      chr2    102-112      + |          2 0.888889
##   c      chr2    103-113      + |          3 0.777778
##   d      chr2    104-114      * |          4 0.666667
##   e      chr1    105-115      * |          5 0.555556
##   -----
##   seqinfo: 3 sequences from an unspecified genome
##
## $`2`
## GRanges object with 5 ranges and 2 metadata columns:
##      seqnames      ranges strand |      score      GC
##      <Rle> <IRanges> <Rle> | <integer> <numeric>
##   f      chr1    106-116      + |          6 0.444444
##   g      chr3    107-117      + |          7 0.333333
##   h      chr3    108-118      + |          8 0.222222
##   i      chr3    109-119      - |          9 0.111111
##   j      chr3    110-120      - |         10 0.000000
##   -----
##   seqinfo: 3 sequences from an unspecified genome
```

The `flank` method can be used to recover regions flanking the set of ranges represented by the `GRanges` object. So to get a `GRanges` object containing the ranges that include the 10 bases upstream of the ranges:

```
flank(g, 10)
```

```
## GRanges object with 4 ranges and 2 metadata columns:
##      seqnames      ranges strand |      score      GC
##      <Rle> <IRanges> <Rle> | <integer> <numeric>
##   a      chr1    112-121      - |          1 1.000000
##   b      chr2     92-101      + |          2 0.888889
##   c      chr2     93-102      + |          3 0.777778
##   j      chr3    121-130      - |         10 0.000000
##   -----
##   seqinfo: 3 sequences from an unspecified genome
```

The `shift` method will move the ranges by a specific number of base pairs, and the `resize` method will extend the ranges by a specified width.

```
shift(g, 5)
```

```
## GRanges object with 4 ranges and 2 metadata columns:
##      seqnames      ranges strand |      score      GC
##      <Rle> <IRanges> <Rle> | <integer> <numeric>
##  a      chr1    106-116      - |         1  1.000000
##  b      chr2    107-117      + |         2  0.888889
##  c      chr2    108-118      + |         3  0.777778
##  j      chr3    115-125      - |        10  0.000000
##  -----
##  seqinfo: 3 sequences from an unspecified genome
```

```
resize(g, 30)
```

```
## GRanges object with 4 ranges and 2 metadata columns:
##      seqnames      ranges strand |      score      GC
##      <Rle> <IRanges> <Rle> | <integer> <numeric>
##  a      chr1     82-111      - |         1  1.000000
##  b      chr2    102-131      + |         2  0.888889
##  c      chr2    103-132      + |         3  0.777778
##  j      chr3     91-120      - |        10  0.000000
##  -----
##  seqinfo: 3 sequences from an unspecified genome
```

Between-range methods, such as `union` and `intersect`, calculate relationships between different `GRanges` objects.

```
g2 <- head(gr, n=2)
union(g, g2)
```

```
## GRanges object with 3 ranges and 0 metadata columns:
##      seqnames      ranges strand
##      <Rle> <IRanges> <Rle>
##  [1]      chr1    101-111      -
##  [2]      chr2    102-113      +
##  [3]      chr3    110-120      -
##  -----
##  seqinfo: 3 sequences from an unspecified genome
```

```
intersect(g, g2)
```

```
## GRanges object with 2 ranges and 0 metadata columns:
##      seqnames      ranges strand
##      <Rle> <IRanges> <Rle>
##  [1]      chr1    101-111      -
##  [2]      chr2    102-112      +
##  -----
##  seqinfo: 3 sequences from an unspecified genome
```



## 7.11 REGION SET ENRICHMENT ANALYSIS

Given a query region set (e.g. result from a ChIP-seq experiment), region set enrichment analysis looks for the most similar region sets in previously published data.

### 7.11.1 INTERVAL SIMILARITY METRICS

Measuring similarity between intervals: - Overlap count - Jaccard index (similarity of two sets)  $\frac{A \cap B}{A \cup B}$  - Fisher's Exact Test (used in LOLA, Sheffield and Bock 2016)

### 7.11.2 REGION SET ENRICHMENT ANALYTICAL TOOLS

- LOLA, Sheffield and Bock 2016
  - Uses contingency table, and Fisher's Exact Test
- GIGGLE, Layer 2018
  - Uses B+ tree, a B tree with nodes hold keys pointing to a linked list of values.
- IGD (Integrated Genome Database), Feng and Sheffield 2021
  - Uses a linear binning of an array

## 7.12 CONCLUSION

Epigenomic data generation has rapidly increased, leading to large volumes of genomic intervals data. Genomic intervals are an extremely useful abstraction. Diverse biological questions can be asked using intervals. Different data structures and algorithms enable large-scale genomic interval analysis. And, new methods for genomic intervals are being developed.

## 7.13 TERMINOLOGY

- **interval**: a location on a sequence defined by a start and end
- **region**: often used as a synonym of interval (biological context, regions of genome)
- **chromosome**: In the context of genomic intervals, chromosome is often used to indicate the name of the sequence on which an interval is defined, but this is really a fudging, use chromosome to refer the interval = sequence name
- **interval set (region set)**: a collection of intervals/regions, result of ChIP-seq exp is a interval set
- **coordinate system**: a set of sequence names and lengths
- **intersection**: intersection of two interval A and B is the region which is common to both A and B
- **union**: union of two interval A and B consists of all regions belonging to either A or B
- **overlap**: Two regions that are overlapping if they have intersecting region

## Chapter 8

# ATAC-seq diagnostics and harmonization

### 8.1 ATAC-seq Overview

#### 8.1.1 What's ATAC-seq?

The assay of transposase accessible chromatin (ATAC-seq) uses hyperactive Tn5 transposase to simultaneously cut and ligate adapters for high-throughput sequencing at regions of increased accessibility. Genome-wide mapping of insertion ends by high-throughput sequencing allows for multidimensional assays of the regulatory landscape of chromatin with a relatively simple protocol that can be carried out in hours for a standard sample size of 50,000 cells. (Buenrostro et al. 2016)

#### 8.1.2 Tn5 transposase

The ATAC-seq methodology relies on library construction using the hyperactive transposase Tn5. Tn5 is a prokaryotic transposase, which endogenously functions through the “cut and paste” mechanism, requiring sequence-specific excision of a locus containing 19 base-pair inverted repeats. Tn5 transposase used in ATAC-seq are loaded with sequencing adapters which create an active dimeric transposome complex. Engineered Tn5 transposase harbors specific point mutants to the Tn5 backbone which significantly increases activity. Such transposase preferentially inserts sequencing adapters into unprotected regions of DNA, therefore acting as a probe for measuring chromatin accessibility genome-wide.

#### 8.1.3 Advantages against other technologies

The biggest advantage of ATAC-seq is that it requires **less material** and requires **less preparation time**.

Other methods for assaying chromatin structure and composition such as MNase-seq, ChIP-seq and DNase-seq often require tens to hundreds of millions of cells as input material, averaging out heterogeneity in cellular populations. In many cases, rare and important cellular sub-types cannot be acquired in amounts sufficient for genome-wide chromatin analyses.

#### 8.1.4 Difference between ATAC-seq and ChIP-seq

ChIP-seq

- uses an antibody
- selects a specific factor
- identifies fragments

ATAC-seq

- no antibody
- all factors
- identifies nucleotides

ATAC-seq identifies nucleotides instead of fragments thus loses specificity compared with ChIP-seq.

## 8.2 Basic Algorithm for Sequence-Based Genome Data

### 8.2.1 Basic Workflow

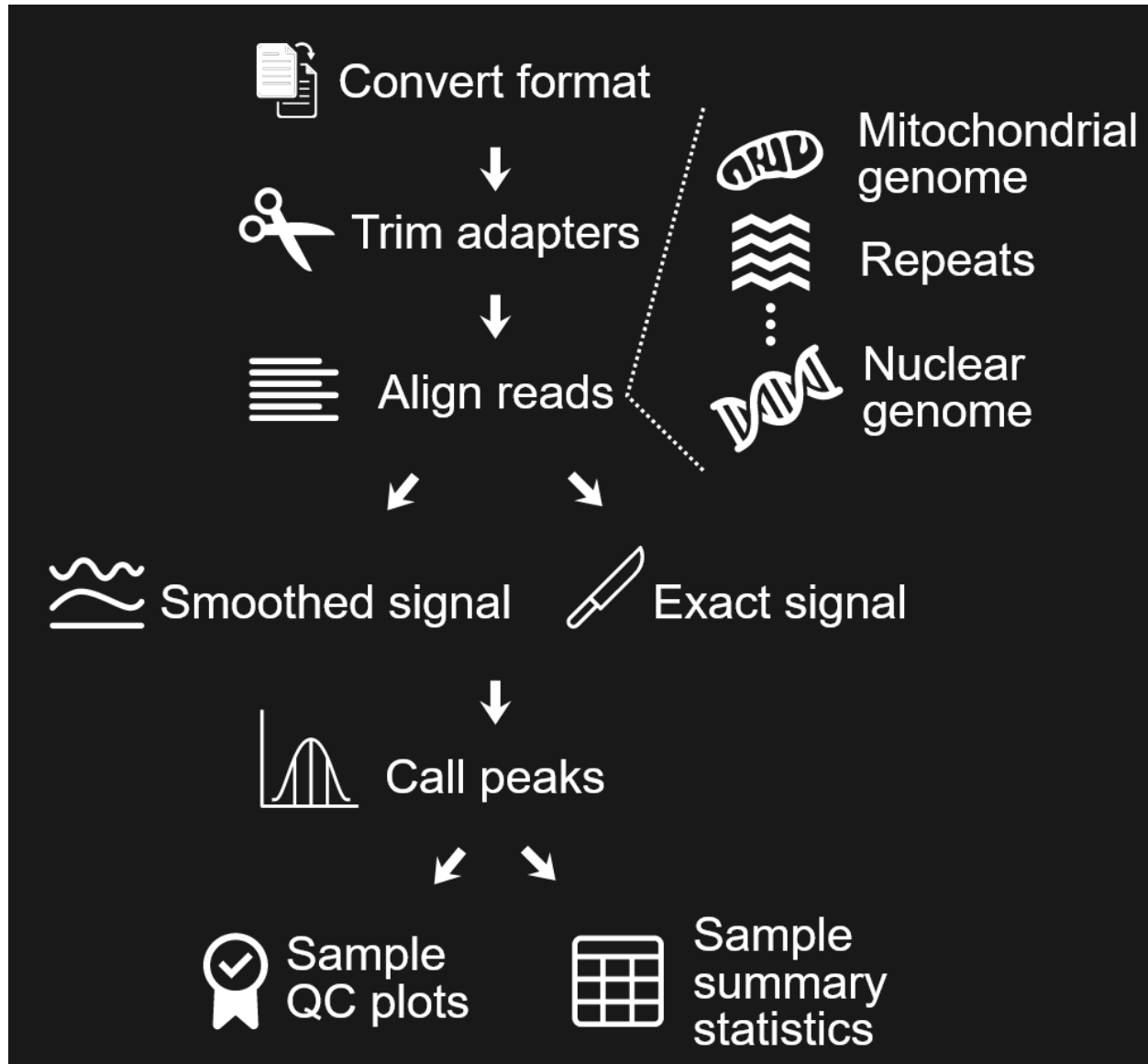


Figure 8.1: ATAC-seq workflow

- Reads can be aligned to mitochondrial genome and nuclear genome separately for filtering.
- DNA is cut off by two transposases. Repair of sticky ends cut by Tn5 transposases induces 9bp duplication on the sides. Trim 4bp from 5' end and 5bp from 3' end to correct this.

### 8.2.2 ATAC-seq Peak Calling

- Macs2 (<https://pypi.org/project/MACS2/>)
- SICER (<https://github.com/zanglab/SICER2>)
- F-seq (<https://fureylab.web.unc.edu/software/fseq/>)

F-seq uses univariate kernel density estimation (kde) to infer the pdf and the smoothness of the density estimates can be controlled by setting the bandwidth.

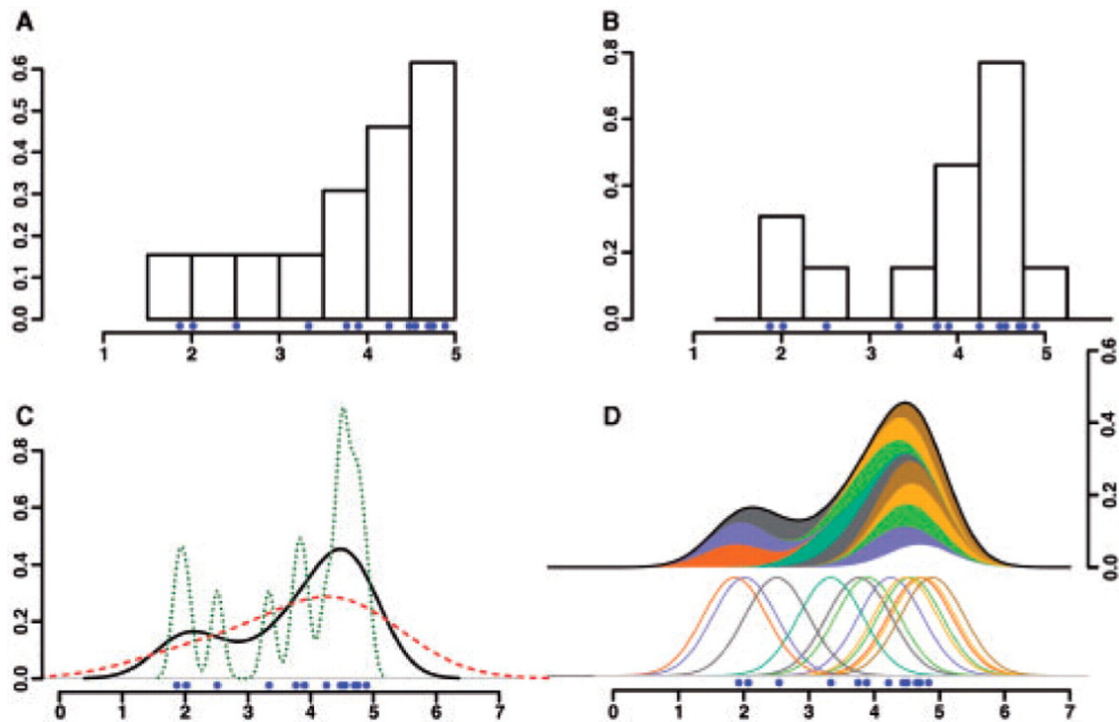


Figure 8.2: Examples of histogram and density estimation properties.

### 8.2.3 QC Metrics

Pre-alignment

- Fastqc

Post-alignment

- TSS Enrichment Score

The idea behind the TSS enrichment score metric is that ATAC-seq data is universally enriched at gene TSS regions compared to other genomic regions, due to large protein complexes that bind to promoters. It was also found that TSS enrichment to be representative across the majority of cell types tested in both bulk ATAC-seq and scATAC-seq (ArchR).

1. Collect a reference set of TSSs
2. Aggregate reads 2000 bp around TSSs
3. Tile into 100bp windows

4.  $S_{TSS} = \frac{TSS}{background}$

- Fragment distribution

Typically, a successful ATAC-seq experiment should generate a fragment size distribution plot with decreasing and periodical peaks corresponding to the nucleosome-free regions (NFR) (< 100 bp) and mono-, di-, and tri-nucleosomes (~200, 400, 600 bp, respectively) (Yan et al. 2020)

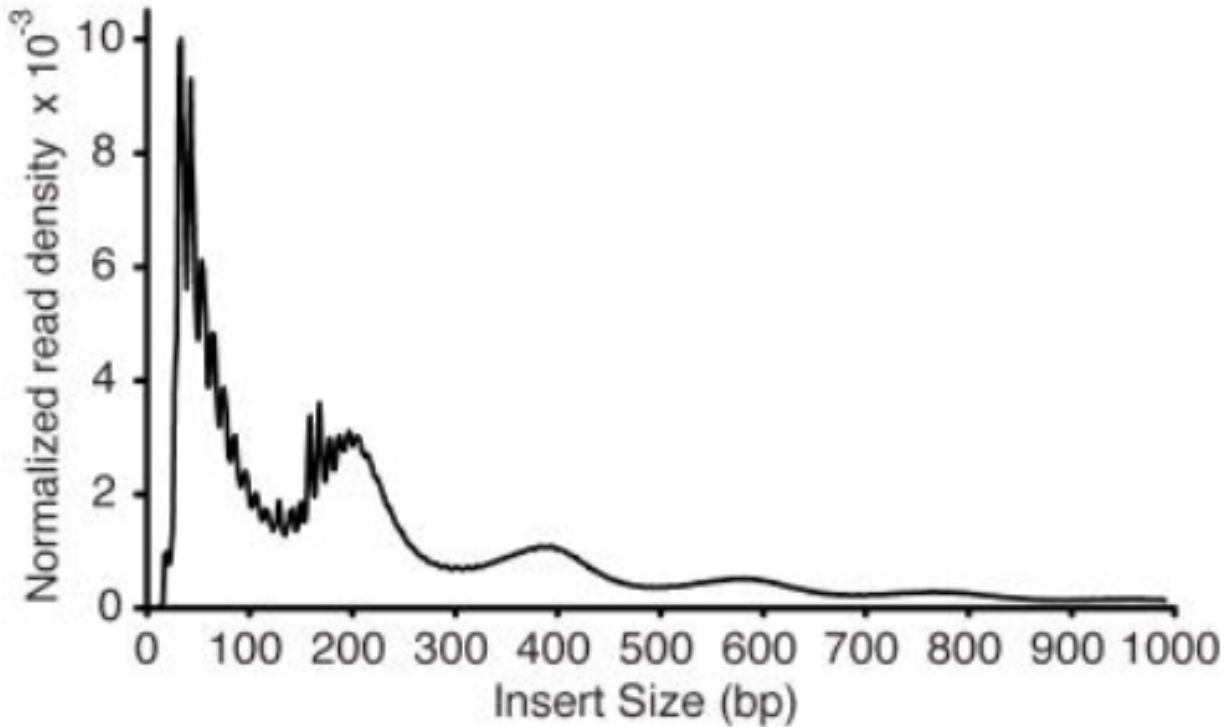


Figure 8.3: ATAC-seq fragment distribution

## 8.3 Harmonization

To answer biological questions for chromatin analysis we need to compare samples. The harmonization steps transform data so as to make them more comparable. Two or more datasets can be normalized to each other or to an independent reference, it depends on the data.

### 8.3.1 Problem 1: Peak locations differ across samples

Possible solution: Consensus peaks

Approaches to get consensus peaks:

- Union: simply merge all the peaks into one
  - Union doesn't work for many samples, especially for heterogeneous samples.
- Union-tile: Divide the whole genome into tiles evenly and project peaks from different samples to the tile array.
  - Union tile increases resolution, but also increases compute time.
  - Tile boundaries are artificial and may split meaningful units.

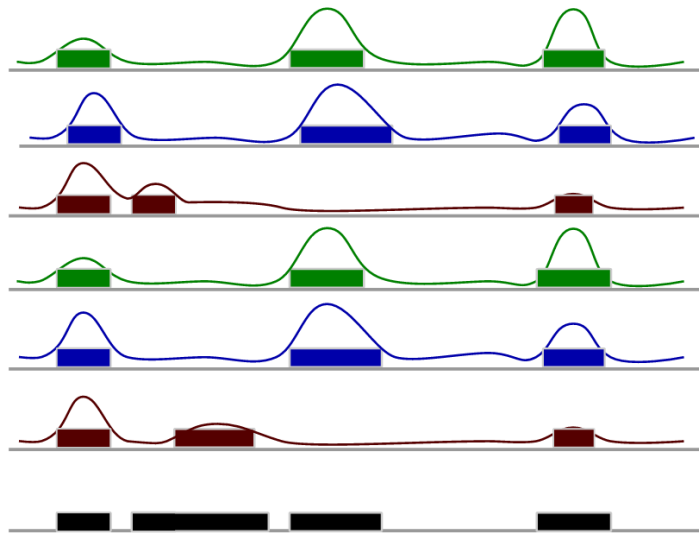


Figure 8.4: Union

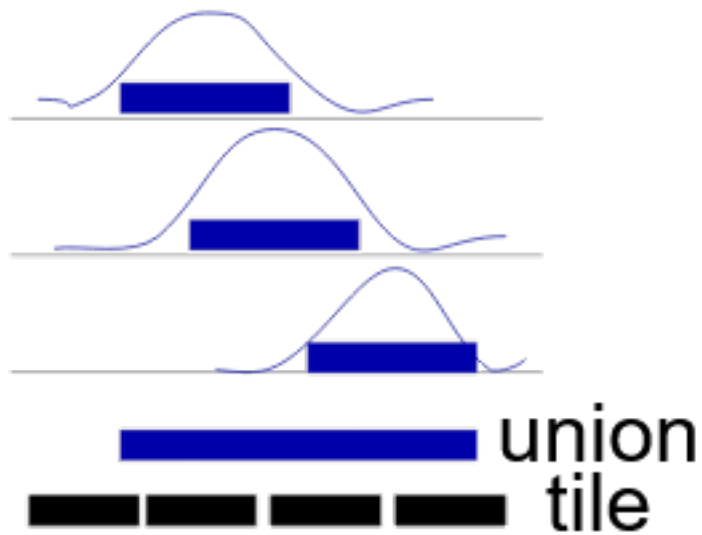


Figure 8.5: Union-tile

- Iterative overlap
  - Peaks are first ranked by their significance. The most significant peak is retained and any peak that directly overlaps with the most significant peak is removed from further analysis. Then, of the remaining peaks, this process is repeated until no more peaks exist. This avoids daisy-chaining and still allows for use of fixed-width peaks.
  - This algorithm avoids many issues of the first two methods.

With consensus peaks, peak accessibility matrix could be filled by extracting score from signal track (.wig) or aligned reads (.bam).

Matrix values could be:

- Number of reads aligned
- Maximum height of signal curve
- Area under the curve (sum of signal curve)

### 8.3.2 Problem 2: Sequencing depth normalization

Reads per Kilobase per Million (RPKM) is usually used to normalize RNA-seq data. For ATAC-seq, we need to consider both the region width (fixed vs variable) and data type (raw read counts vs. model signal track). It also needs to be noted that sample-specific peaks are usually **low-accessible**.

**ATAC-seq normalization approaches:**

- Standard normalization
  - $x' = \frac{x-\mu}{\sigma}$
  - Centers: puts data around the mean
  - Scales: puts data on the same range
  - Result is standard deviations away from the mean
  - Doesn't change the shape of a distribution
- MinMax scaling: Sets all data on a scale from 0-1
  - $x' = \frac{x-x_{min}}{x_{max}-x_{min}}$
  - Makes data interpretable: 1= highly open; 0= closed
  - Accessibility matrix can be normalized on both axes, to compare samples or regions.
- Quantile normalization
  - Assumes sample distributions should be the same. Forces them to look identical.
  - Given reference distribution  $R$ , first rank sample data points  $a_i \in A$ , then set the value of  $A_r$  to  $R_r$ , for each rank  $r$ . So, the highest  $a$  assumes the highest value in  $R$ , the 2nd highest  $a$  assumes the 2nd highest in  $R$ , etc.

### 8.3.3 Quantile-quantile plots

The quantile-quantile (q-q) plot is a graphical technique for determining if two data sets come from populations with a common distribution.

A q-q plot is a plot of the quantiles of the first data set against the quantiles of the second data set. By a quantile, we mean the fraction (or percent) of points below the given value. That is, the 0.3 (or 30%) quantile is the point at which 30% percent of the data fall below and 70% fall above that value.

A 45-degree reference line is also plotted. If the two sets come from a population with the same distribution, the points should fall approximately along this reference line. The greater the departure from this reference line, the greater the evidence for the conclusion that the two data sets have come from populations with different distributions.

Examples:

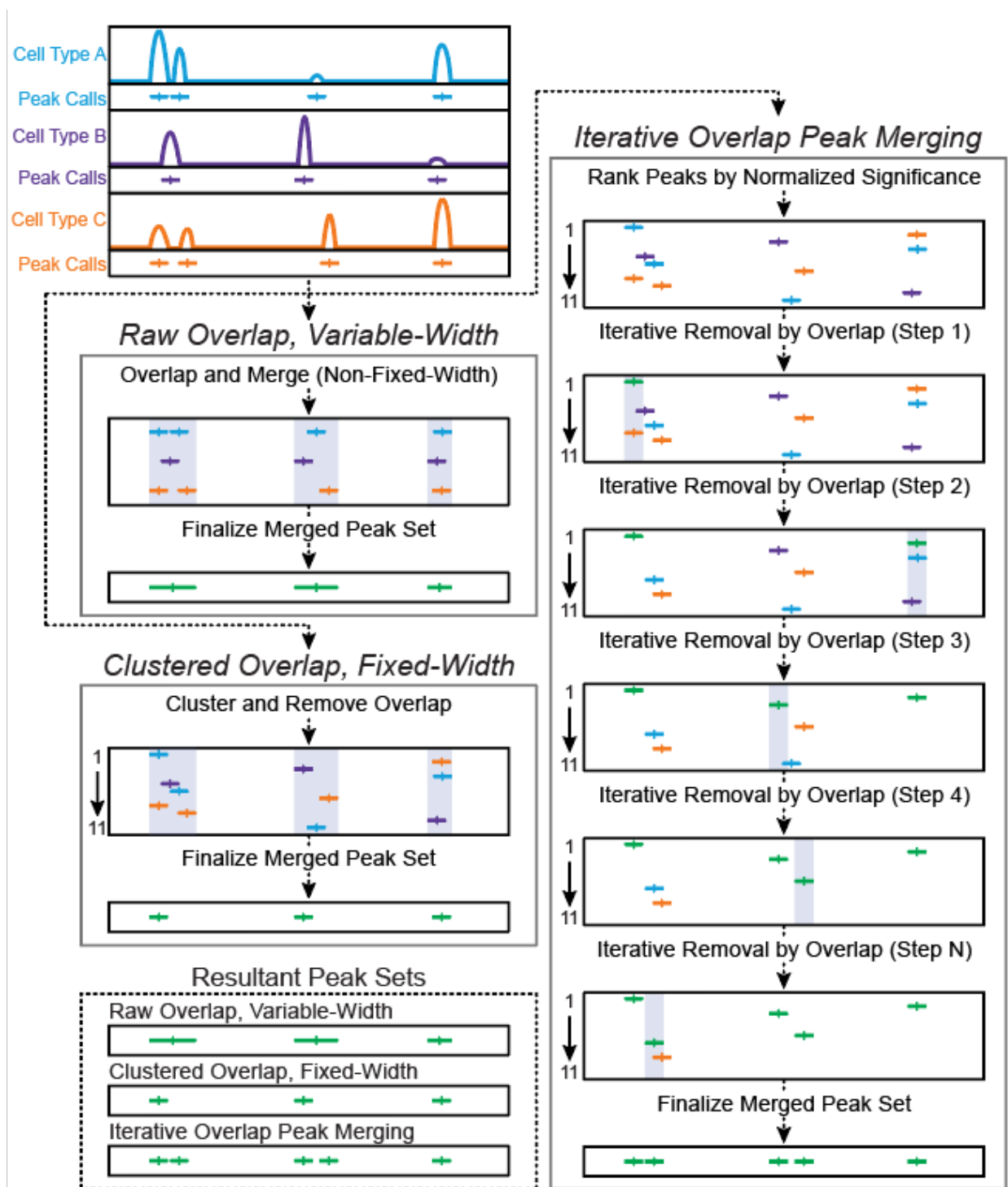


Figure 8.6: Iterative overlap



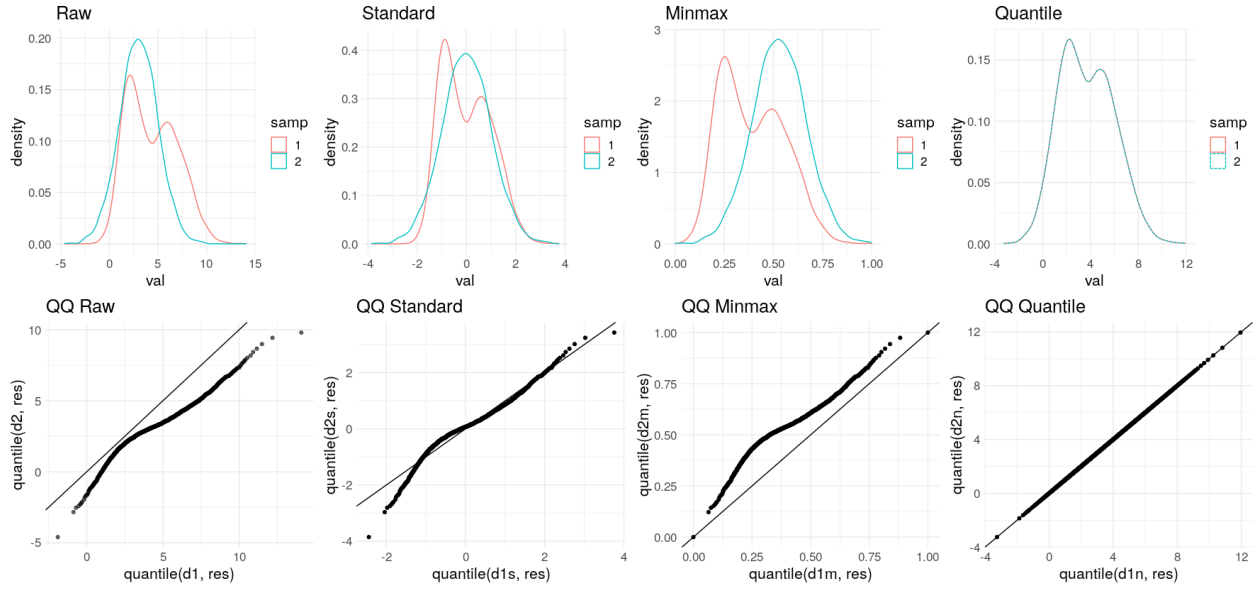


Figure 8.7: Quantile-quantile plot example 1

- On q-q plot, a curve formed by points indicating different distributions.
- Standard and MinMax normalizations tend to keep the shapes of raw distributions.
- Quantile normalization always returns an average distribution, so points are on the diagonal.

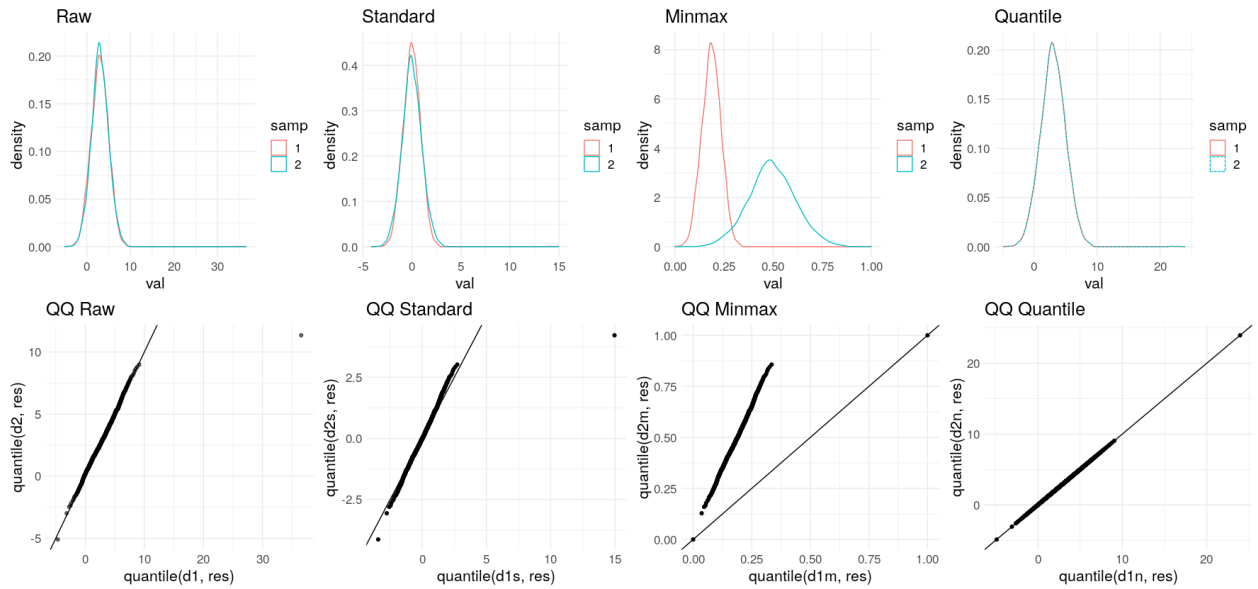


Figure 8.8: Quantile-quantile plot example 2

- MinMax scaling is highly influenced by outliers.
- Trim extreme data is a way to handle outliers ( clip function:  $x = \min(x, \text{quantile}(x, 0.99))$  )

## 8.4 Batch effect

In molecular biology, a **batch effect** occurs when non-biological factors in an experiment cause changes in the data produced by the experiment. Such effects can lead to inaccurate conclusions when their causes are correlated with one or more outcomes of interest in an experiment. They are common in many types of high-throughput sequencing experiments, including those using microarrays, mass spectrometers, and single-cell RNA-sequencing data. They are most commonly discussed in the context of genomics and high-throughput sequencing research, but they exist in other fields of science as well. (wikipedia)

It must be noted that **normalization doesn't equal batch correction**.

Normalization adjusts global properties of measurements for individual samples so that they can be more appropriately compared. Normalization does not remove batch effects, which affect specific subsets of genes and may affect different genes in different ways. (Leek et al. 2010)

### 8.4.1 Sources of batch effects

Sources of batch effects include:

- experimental conditions
- different protocols
- different labs
- different technicians
- sequencing equipment
- lane in a sequencer
- day of the week
- reagent lots

### 8.4.2 Batch correction approaches

- Known batch effects can be revealed using PCA, clustering and individual feature plots
- If batches are known, tools such as ComBat could be applied to remove the effects.
- If batches are unknown, we should remove everything else than biological signal.
- Surrogate variable analysis is an example of unknown batch correction. The idea is:
  1. Remove signal of variable of interest from data.
  2. Decompose the residual matrix to identify vectors with more variation than expected by chance (permutation test).
  3. Identify the subset of genes driving each vector.
  4. For each subset of genes, build a surrogate variable based on the original expression data.
  5. Include surrogate variables as covariates in subsequent analyses.

Confounding here is when technical batches coincide with biological interest. So avoid this situation when doing the experiment.

## Chapter 9

# Scalable Computing in Genomics

### 9.1 Lower costs $\rightarrow$ More data

The reduced cost of sequencing has led to a huge growth in sequencing data. The more samples the more we care about scalability. The less refined the data the more we care about scalability. There are two components of scalability, time and space. Time as in how long it takes to process the data; space as in how we store the data, how we can compress it such that we can store and retrieve.

### 9.2 Typical Epigenome Project

A typical epigenome project consists of many independent samples such as patients, cell lines, conditions, etc.,. We define our analysis in two stages: ‘pipeline’ is where we process raw data, i.e. same procedures done on every sample; then we get to a stage where we perform ‘analysis’, which is taking our processed data, looking across samples and analyze them.

### 9.3 Approaches for Scalability

There are four approaches for scalability. Parallelization and optimization are more relevant to the time question, i.e. how to execute more efficiently computationally, whereas compression and databases are more relevant to the space complexity question. Parallelization, where we split a compute task, and then compute each split simultaneously. One of the framework to perform this is ‘split-apply-combine.’ This is useful because many problems call for a similar type of computing architecture. Here we have our data analogous to a puzzle, where we split them into chunks. For each chunk we apply some function, then take the results from chunks and combine them as one. Existing frameworks are MapReduce and Hadoop. These approaches are applicable at different levels, sample level or within sample level. If we think of these puzzle pieces as different samples, then we can view this as splitting across different samples, and then applying our function to each sample. We can also apply this to one sample, say if we have an aligned reads file, and now we can split by chromosomes, then we apply our function to each chromosomes. So our combination is now summarizing those results across chromosomes.

### 9.4 Scopes of Parallelism

We have a sample and a sequence of steps we will apply to that sample. We can run our data naively in a serial way, or we can parallelize it. There are three different ways to parallelize. The first way is to parallelize by process, where we use more than one CPU on that process. Another way to parallelize is by sample, where we run many samples simultaneously, but at each single step we only use one CPU. The third way is to parallelize by dependence, the idea is to identify the independence and dependence

between steps and parallelize the independent parts. There are advantages, disadvantages to each of these way of parallelization. Parallelize by process is straightforward using tools. Its disadvantage is that it is a node-threaded parallelism, meaning we cannot parallelize across computers/nodes in a cluster. We are also limited by tool capacity. Sometime the aligner does not have the capacity to parallelize. Aside, in the cluster hardware, we have nodes, and each node is a computer. Each node has 16 cores times 4 different units, so if we are running a process on this cluster, we cannot use more than 64 cores to parallelize by process. Now we look at parallelize by sample/job. We have no shared memory so we cannot parallelize the same process across nodes, but we can parallelize by sample across nodes independently. HPC clusters are intended for parallelize by sample, where the job is restricted by the size of HPC instead of the size of the node. Another advantage is we no longer depend the tool's capacity to parallelize. The third one is parallel by dependency, where we take the different tasks in the pipeline, and run different tasks simultaneously. This is not necessarily node-threaded because we can run different tasks on different nodes. The tasks should be independent hence requires no shared memory. This is the benefit over the process level. The cons are from the dependency. 1) we may have shared file-system requirements 2) requires a dependency graph of workflow steps 3) requires a layer of task management (such as task synchronization) above typical HPC usage 4) limited to independent workflow elements 5) requires (partial) independence of jobs In summary, parallelizing by process is very easy. Parallelizing by sample is not as easy as parallelizing by process, since we need to submit different jobs, but these jobs are independent of pipeline. Parallelizing by dependency is not independent of the pipeline. For this, we have to design a dependency graph for each task. (See graphs for time benefit versus sample resources ratio in the slides) There are two ways to think about parallelization. One way is to parallelize jobs (BatchJobs and snow), and the other is parallelizing within one interactive environment (parallel package in base R.) The mclapply is node parallelism. In python, there are two way of parallelizing. The first one is using the subprocess module, which spawns a new process through a shell-like system (see code in the slides) from within python. The other way to do it is multiprocessing (package), which creates a new pool of processors and provides a map function that operates similar to lapply functions in R.

## 9.5 Workflows

Workflows are important in epigenomics and lots of different areas in genomics. A workflow is a repeatable sequence of tasks that process a piece of data. For example, we get raw read *s* from a sequencer, we align them, then we take those alignments and run MACS or SCICER to call peaks on them, followed by some analysis on these peaks. This is a workflow spectrum, we can approach this in different ways. From interactive analysis to pipeline, say we have a single sample, we type a sequence of commands such as trimmomatic, bowtie2 and the interactive means we are interacting within the shell. This is not efficient enough when we have to enter the same commands for many samples, so we write shell scripts to run these different tasks. One step forward is to wrap the shell scripts into a pipeline framework. Frameworks, or workflow frameworks are basically development toolkits to make it easier to build workflows.

There are many workflows out there, most of them are domain specific. Here we introduce three mainstream workflows, snakemake, nextflow, and common workflow language. Make is mostly used for compiling software, containing recipes for how to compile source code into the binary output executable. What makes it useful in our case is that we can define output or target, where make automatically computes the dependency (the order of which commands to execute as well as executing the dependent parts of the commands to compute the input.) So instead of writing a sequence of tasks, we actually write rules for the workflow to figure out the dependency for us. Nextflow is similar to snakemake. In common workflow language, we have a separate tool file, making it even more modular so that we can use the same tool file across different workflows. The other difference is that both snakemake and nextflow have their specific execution method, whereas common workflow language is designed to be a universal/standardized language that can be run on different engines/independent of execution engine.

## 9.6 Stop using shellscripts for pipelines!

They are difficult to write, difficult to read. From stack overflow:

The shell makes common and simple actions really simple, at the expense of making more complex things much more complex. Typically, a small shell script will be shorter and simpler than the corresponding python program, but the python program will tend to gracefully accept modifications, whereas the shell script will tend to get less and less maintainable as code is added. This has the consequence that for optimal day-to-day productivity you need shell-scripting, but you should use it mostly for throwaway scripts, and use python everywhere else. -Anonymous

Still, shell scripts are super useful for small tasks. When the task gets more complex, it is better to move on to something more maintainable.

## 9.7 Advantages of workflows

- Reproducibility
- Restartability
- Reusability
- Logging
- Provenance (keep track of updates)
- Scaling up compute resources
- Dependency management

## 9.8 Optimization

The big O notation is a way of simplifying the complexity of an algorithm (by looking at the highest order terms.) By looking at which class an algorithm belongs to, we can see how well it scales (see plot in the slides.) Factorial time  $O(n!)$  arises in problems involving permutations such as the Traveling Salesman and the shortest common superstring in a brute force way. Exponential time is worse than polynomial time. The exponential time comes from nested subproblems. The polynomial time comes from nested loops. Find overlaps, sequential search can be completed in linear time, binary search can be completed in  $O(\log n)$  (Manhattan phonebook.) Indexing reduces lookups from linear time to constant time. Tabix indexing allows us to randomly access anywhere in the file.

## 9.9 Language choice

Existing implementations are often faster than yours; Compiled languages are faster than scripting languages; Loops in R are slow, vectorized way are faster. Programs run faster in machine code. We can also link C code into R or Python. Often in genomics, disk is the bottleneck, i.e. read/write takes more time. We can prevent read/write by loading into memory. Memory lookups/random access is much faster, but we may not have enough memory for a big file. Extract ATAC in consensus, method 2 is faster and uses more memory because we loaded the file into memory thus avoiding reading from disks. We cannot switch peaks and reads because reads are much larger than peaks, and we may not have enough memory to load all the reads. Interconnect is fast between nodes on a cluster.

## 9.10 Databases

- Database:Space as Parallelization:Compute
- They offload storage requirements
- They are critical for simultaneous multi-user
- Reduce storage requirements by centralizing
- Data has gravity, it brings compute to it

## 9.11 Problem with Cloud Platforms

- Platform lock-in
- Difficulty integrating across platforms
- Duplicated effort for both users and developers