Contents

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 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.

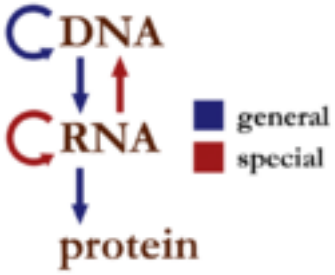


Figure 1.1: Information flows between DNA, RNA and protein. Source: Wikipedia

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

Epigenomics1

From Professor Chongzhi Zang's lecture slides "Regulatory DNA, Transcription factors, Sequence motifs". Scribed by Zhaoxia Ma.

2.1 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

2.2 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, chromation accessibility and 3D genome orgenization, etc.

2.3 Epigenomics

2.3.1 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.

2.3.2 Sequence motif

In general, a motif is a distinctive pattern that occurs repeatedly. In biomelecular 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.

2.3.3 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	**G**uanine
A	A	**A**denine
T	T	**T**hymine
C	C	**C**ytosine
R	G or A	pu**R**ine
Y	T or C	p**Y**rimidine
M	A or C	a**M**ino
K	G or T	**K**eto
S	G or C	**S**trong interaction (3H bonds)
W	A or T	**W**eak 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**N**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:

$$\begin{array}{ccccccccc} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ G & A & G & G & T & A & A & A & C \\ T & C & C & G & T & A & A & G & T \\ C & A & G & G & T & T & G & G & A \\ A & C & A & G & T & C & A & G & T \\ T & A & G & G & T & C & A & T & T \\ T & A & G & G & T & A & C & T & G \\ A & T & G & G & T & A & A & C & T \\ C & A & G & G & T & A & T & A & C \\ T & G & T & G & T & G & A & G & T \\ A & A & G & G & T & A & A & G & T \end{array}$$

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

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

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

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

  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 2.1: 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 $= \log_2 \frac{p_G \times p_A \times p_G \times p_G \times p_T \times p_A \times p_A \times p_A \times p_C}{b_G \times b_A \times b_G \times b_G \times b_T \times b_A \times b_A \times b_A \times b_C}$.

2.3.4 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.

2.3.4.1 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 guatantee to find global optimum, and can find multiple motifs. For disadvanteges, 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.

2.3.4.2 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; θ_0 is non-motif probability (genome backgrouhnd) parameter; θ is motif probability matrix parameter; π 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 π and θ 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 θ_0, seq and θ to estimate π, in which θ_0 and seq are known while θ is given a initial value. In alternative steps, θ is calculated by M-step.

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

seq :

$$
\begin{array}{cccccccccccccc}
T & T & G & A & C & G & A & C & T & G & C & A & C & G & T & \\
T & T & G & A & C & & & & & & & & & & & LR_1 \\
 & T & G & A & C & G & & & & & & & & & & LR_2 \\
 & & G & A & C & G & A & & & & & & & & & LR_3 \\
 & & & A & C & G & A & C & & & & & & & & LR_4 \\
 & & & & C & G & A & C & T & & & & & & & LR_5 \\
 & & & & & G & A & C & T & G & & & & & & LR_6 \\
 & & & & & & A & C & T & G & C & & & & & LR_7 \\
 & & & & & & & C & T & G & C & A & & & & LR_8 \\
 & & & & & & & & . & . & . & & & & &
\end{array}
$$

θ:

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

$$
\begin{aligned}
P(\text{TTGAC}|\theta_0) &= p_{0\text{T}} \times p_{0\text{T}} \times p_{0\text{G}} \times p_{0\text{A}} \times p_{0\text{C}} \\
&= 0.3 \times 0.3 \times 0.2 \times 0.3 \times 0.2 \\
&= 1.08 \times 10^{-3}
\end{aligned}
$$

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

Therefore, the likelihood ratio of the first motif π_1 is $LR_1 = \frac{P(\text{TTGAC}|\theta)}{P(\text{TTGAC}|\theta_0)} = \frac{9\times10^{-5}}{1.08\times10^{-3}}$. Then we can calculate LR_2, LR_3, LR_4, etc.

- M-step: given θ_0, seq, and π to estimate θ, in which θ_0 and seq are known while π with its likelihood ratio LR is calculated by E-step.

  Given an example, seq = TTGACGACTGCACGT, π and its likelihood ratio LR are:

π	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 θ by estimate the probability of A, C, G, T in any of the 5 positions (5 is the length of the motif):

$$P(\text{T in pos1}) = \frac{0.8 + 0.2 + 0.9 + ...}{0.8 + 0.2 + 0.6 + 0.5 + 0.3 + 0.7 + 0.4 + 0.1 + 0.9 + ...}$$

$$P(\text{T in pos2}) = \frac{0.8 + 0.1 + ...}{0.8 + 0.2 + 0.6 + 0.5 + 0.3 + 0.7 + 0.4 + 0.1 + 0.9 + ...}$$

$$P(\text{G in pos2}) = \frac{0.2 + 0.3 + 0.9 + ...}{0.8 + 0.2 + 0.6 + 0.5 + 0.3 + 0.7 + 0.4 + 0.1 + 0.9 + ...}$$

$$P(\text{C in pos5}) = \frac{0.8 + 0.5 + 0.4 + 0.9 + ...}{0.8 + 0.2 + 0.6 + 0.5 + 0.3 + 0.7 + 0.4 + 0.1 + 0.9 + ...}$$

... ...

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

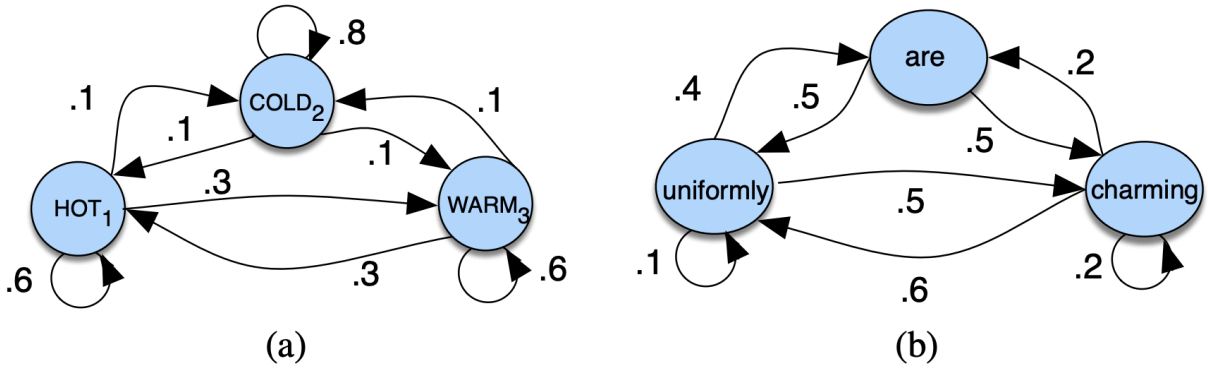
Chapter 3

Hidden Markov Models

3.1 Overview

A **Hidden Markov Model (HMM)** is a type of **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 can not) 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 weren't allowed to look at yesterday's weather!

Here is an example of a 3-state **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.

HMM's 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.

3.2 Mathematical Definition(s)

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

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

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:

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

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

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

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 (π):**

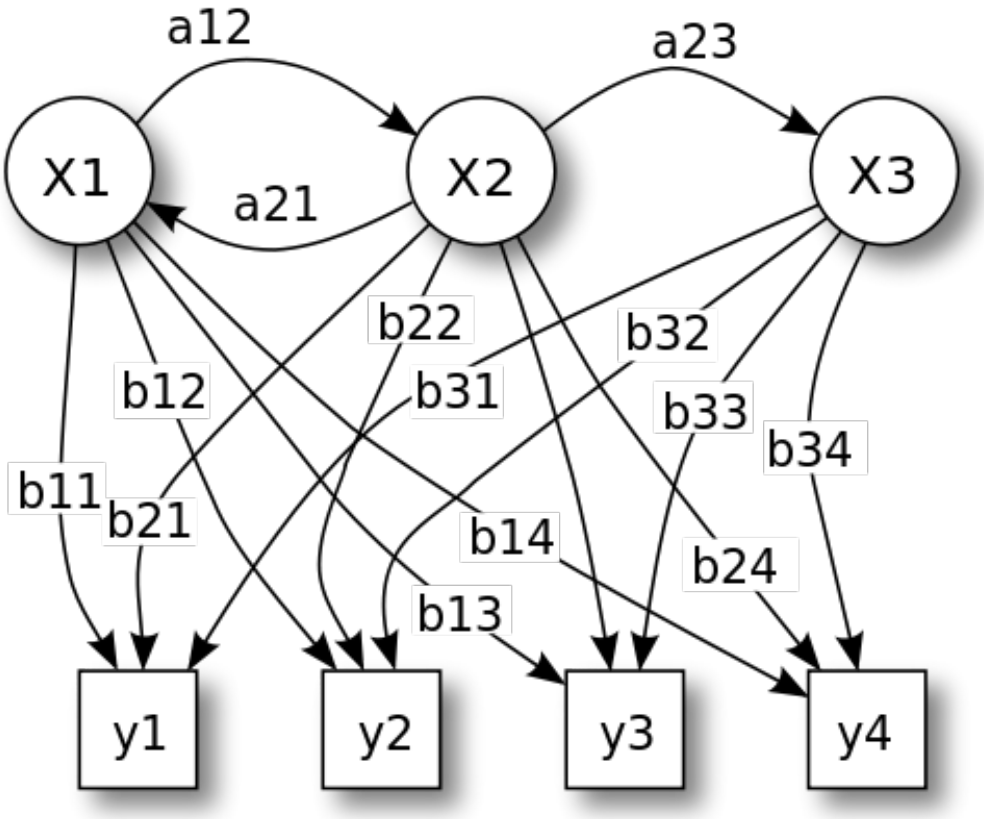
$$\pi = \{P(s_1), P(s_2), \ldots, P(s_i)\}$$

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:



3.3 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:

```
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.

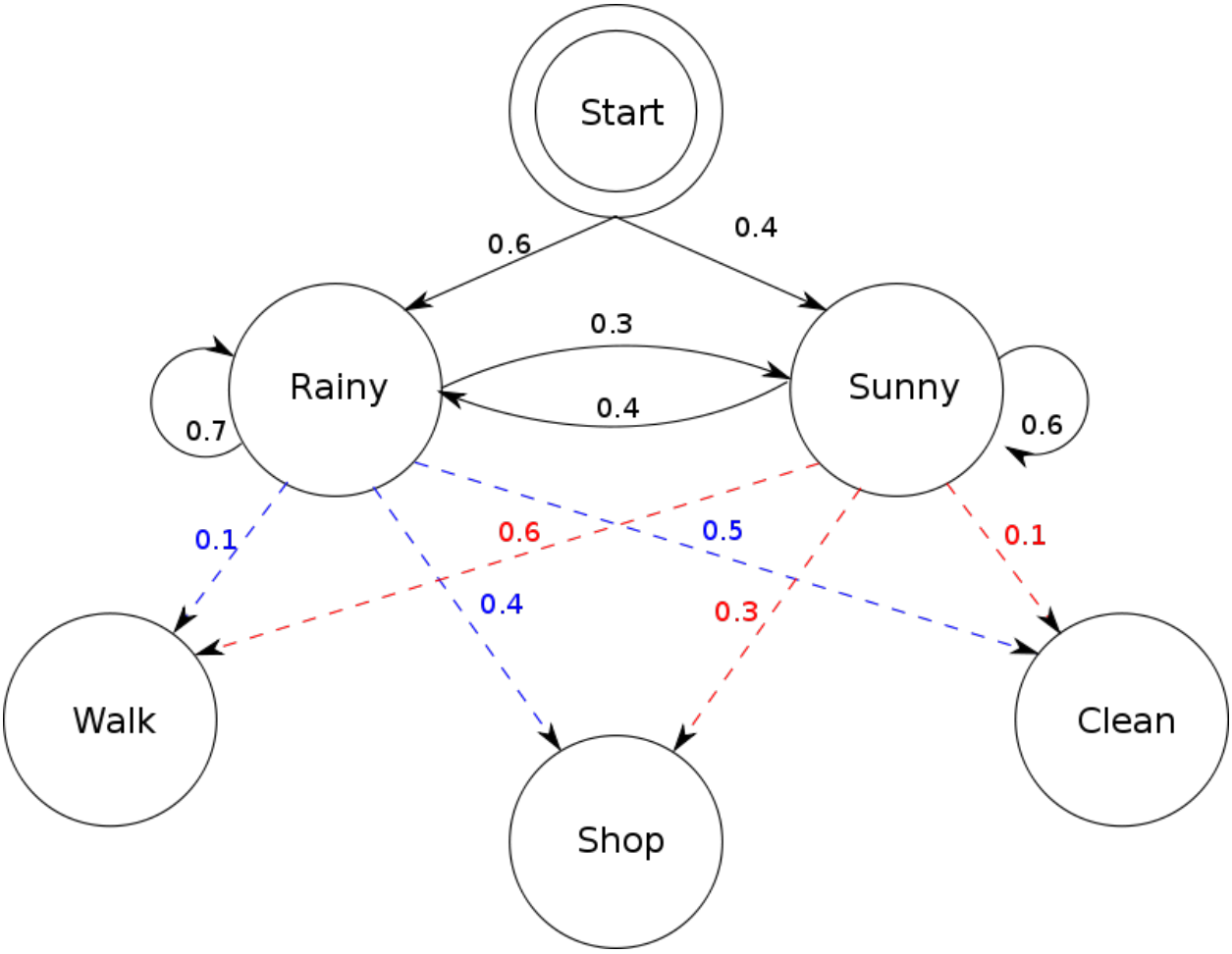


Figure 3.1: Hidden Markov Model to predict weather

3.4 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 **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).

3.4.1 Decoding problem :

Given the HMM M=(A,B,π), and an observation sequence O calculate the most likely sequence of states that produced O. This is commonly solved using the Viterbi Algorithmand 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)

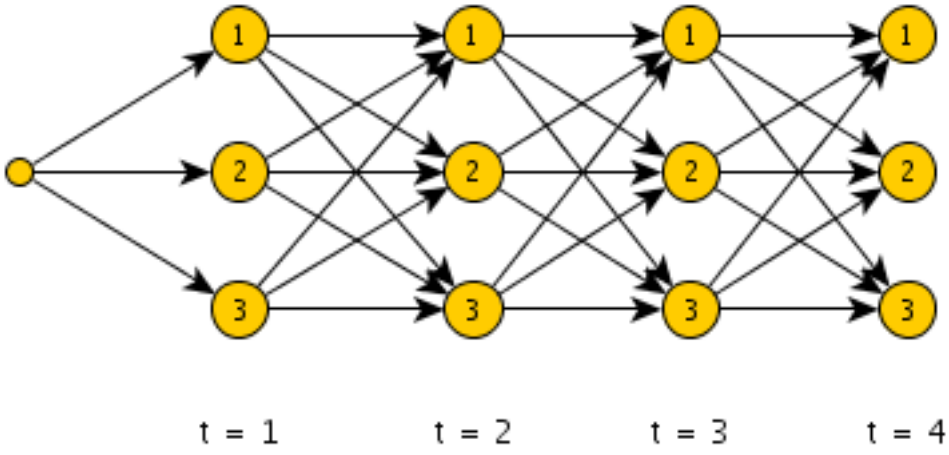


Figure 3.2: Viterbi algorithm

3.4.2 Likelihood Problem:

Similar to the above decoding problem, given the HMM M=(A,B,π), and an observation sequence O,$o_i \in \nu_1, \nu_2, \ldots, \nu_M$ we need to calculate the likelihood P(O|M) using the probabilties 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.

3.4.3 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 π, such that the model can be applied elsewhere.

The most well-known algorithm for this is the Basum-Welch algorithm, it utilizes a stochastic gradient descent algorithm and thus is not garautneed to be a truly optimal solution (local optima). It can also very computationally complex.

3.5 Conclusions

HMM's 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:

3.5.1 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

3.5.2 Cons:

- Bounded by the Markov assumption: The next state is only determined byt 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 4

Genomic Intervals: formats, data structures and algorithms

4.1 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.

4.2 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 drived 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 repersent 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 possitions.

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

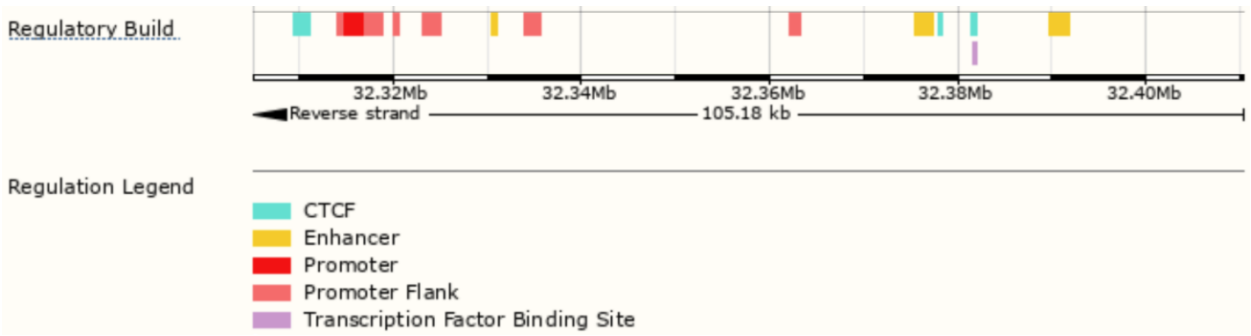


Figure 4.1: nonCodingDNAAnnotation

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 represent as intervals on protein sequences (proteomic interval).

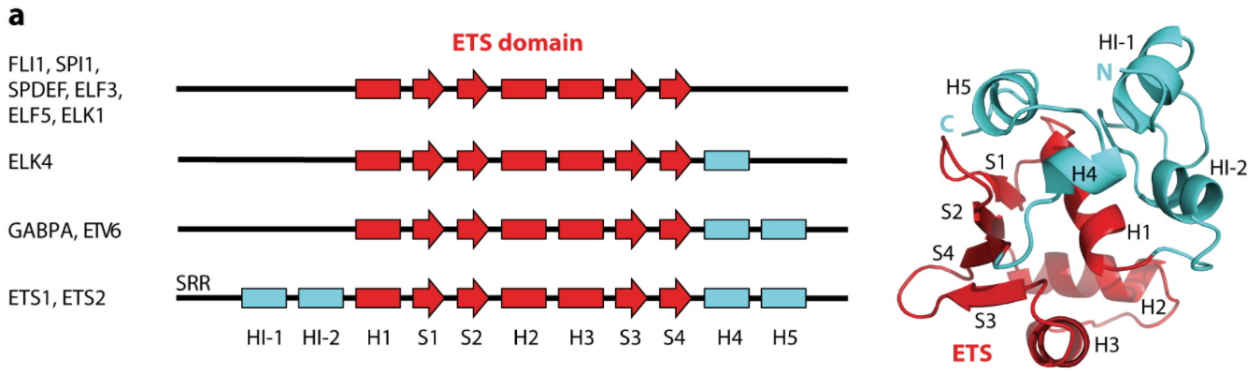


Figure 4.2: proteinDomain

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

4.3 FILE FORMATS FOR GENOMIC INTERVALS

4.3.1 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.

4.3.1.1 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)

4.3.1.2 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 4.3: score

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 RBG 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 initialy used for visilization in UCSC Genome Browser. The visulization related BED fields are: thickStart, thickEnd, itemRgb, blockCount, blockSizes, and blockStarts.

4.3.2 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.

4.3.3 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).

4.3.4 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.

4.4 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.

4.4.1 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 4.4: fixedWiggleFormat

example:

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

Figure 4.5: fixedWiggleExp

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.

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

Figure 4.6: varWiggleFormat

format:

example:

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

Figure 4.7: varWiggleExp

4.4.2 BEDGRAPH

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

example:

```
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 4.8: bedGraph

4.4.3 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)

4.5 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)

4.6 INTERVAL COMPUTATIONS

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

4.6.1 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 $A_{end} > B_{start}$

interval as set

4.6.2 COUNTING OVERLAP

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

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

or

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

4.6.3 INTERSECTING REGION

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

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

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

4.6.4 UNION

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

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

4.6.5 INTERVAL SET OVERLAP COUNTING

By counting the number of intervals in **I** overlapping **q**, where **I** is a set of intervals and **q** is the query interval, we can answer questionas 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 4.9: linearSearch

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

```python
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 4.10: binarySearch

4.7 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.

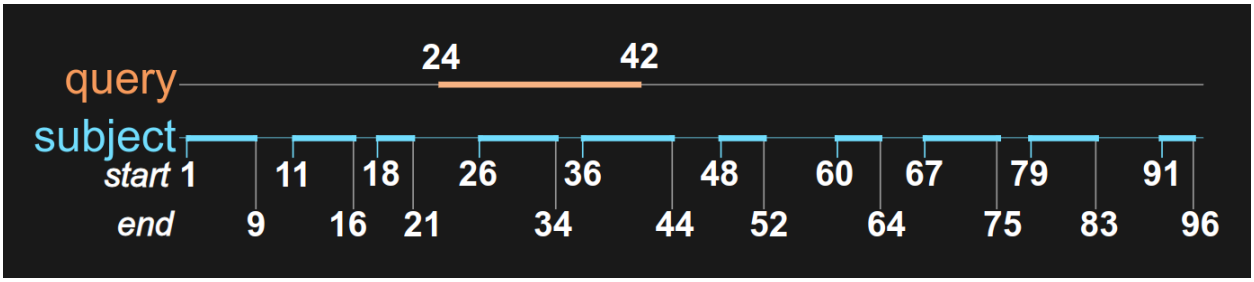


Figure 4.11: binarySearchExp1

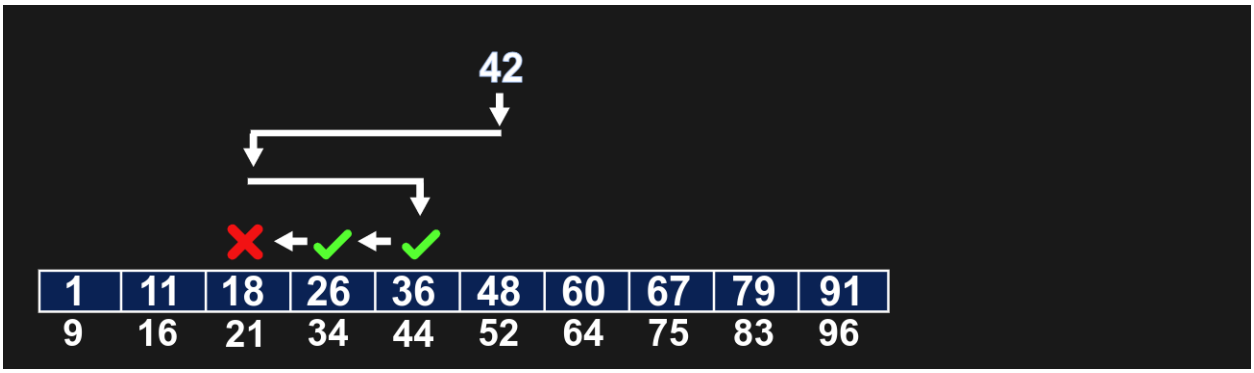


Figure 4.12: binarySearchExp2

However, we can not do binary seach 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.

4.8 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.

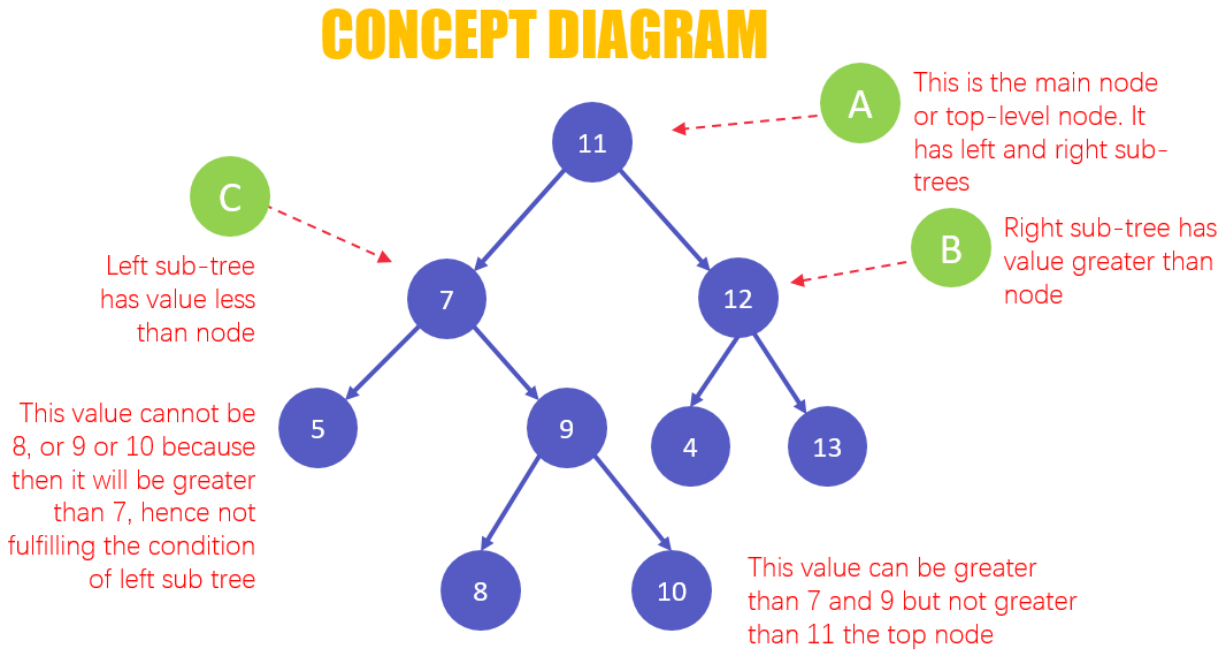
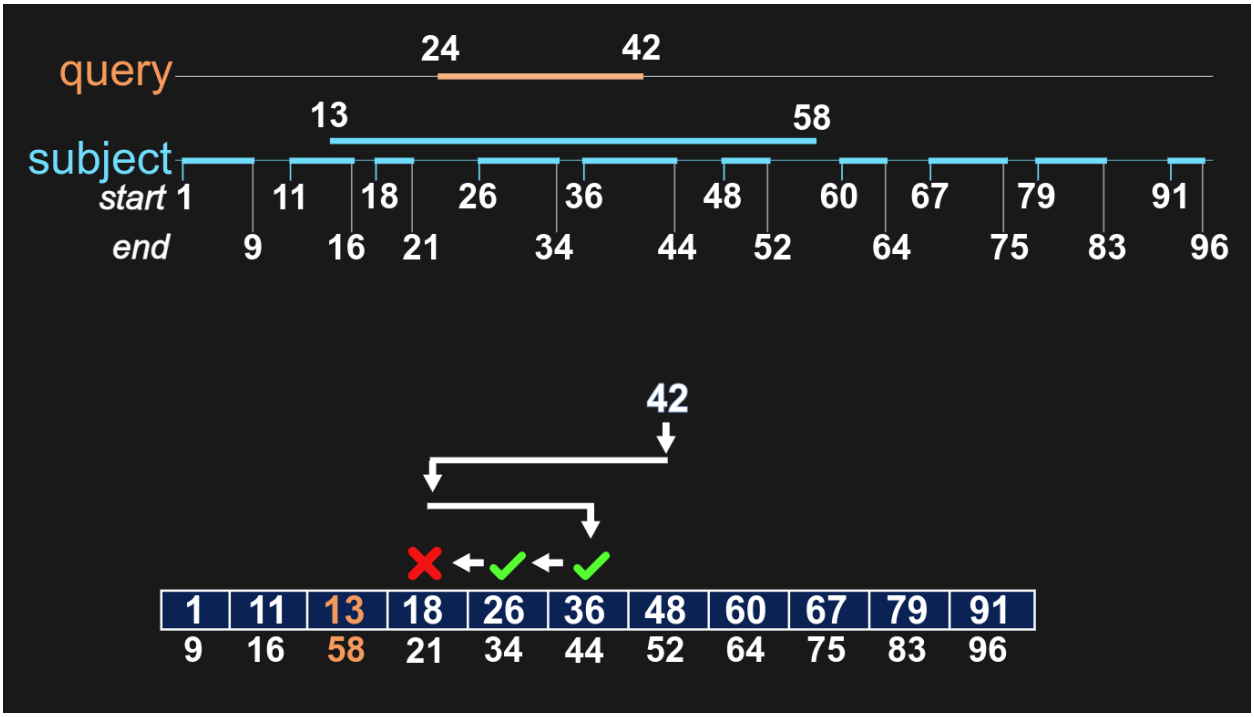


Figure 4.13: BST

4.8.1 Hidden assumption for BST: No containment



There are advanced data structures that addressing the containment problem.

4.9 ADVANCED DATA STRUCTURES

4.9.1 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.

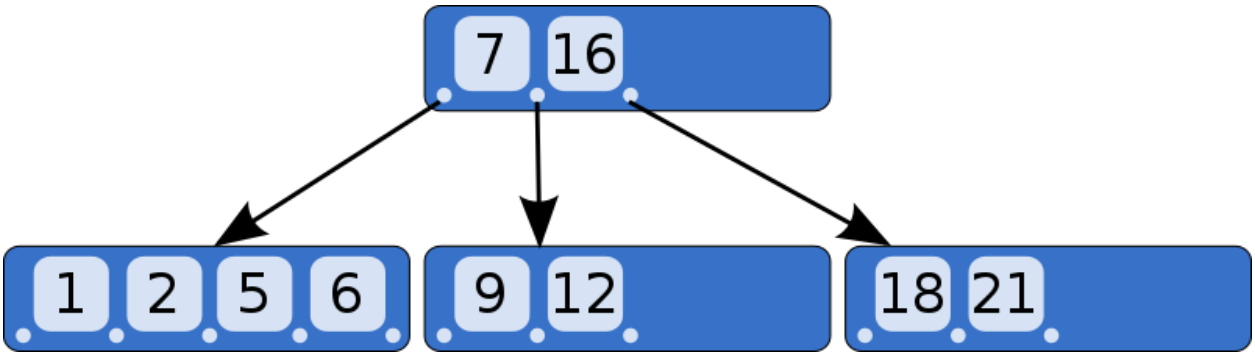


Figure 4.14: BTree

4.9.2 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.

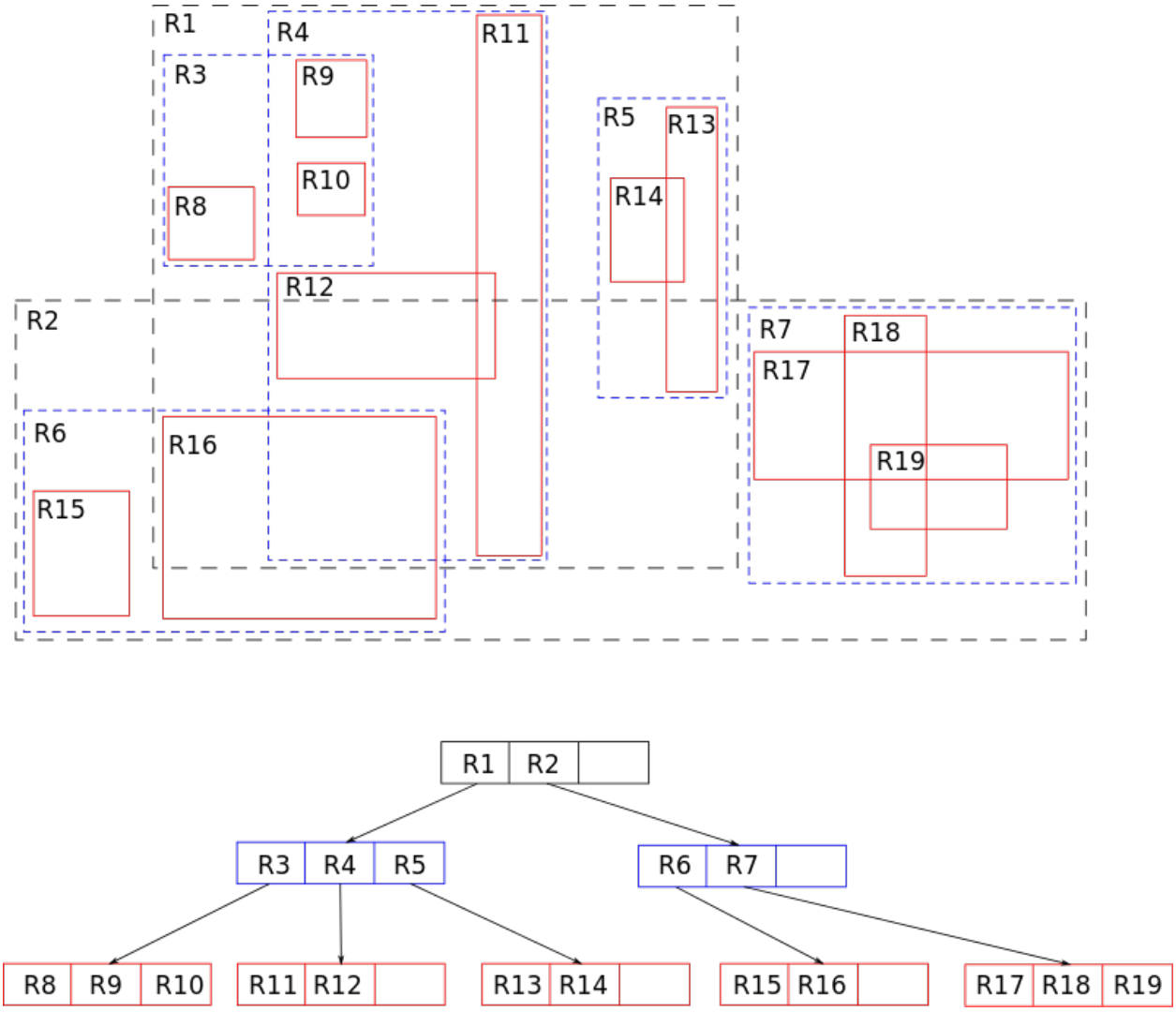


Figure 4.15: RTree

4.9.3 Nested containment lists (NClist)

A NClist 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 NClist 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. NClist is used by Bioconductor GenomicRanges class.

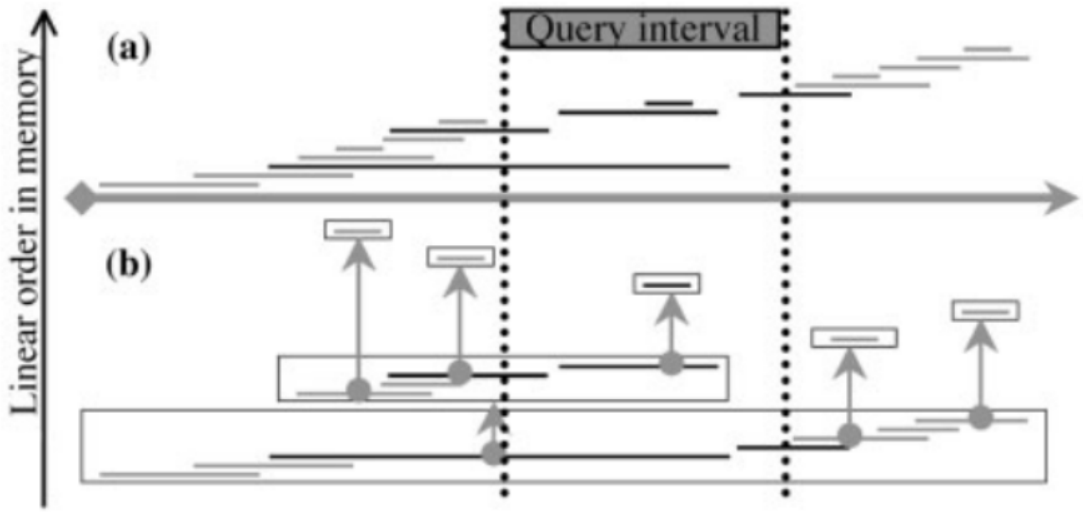


Figure 4.16: NClist

4.9.4 Augmented interval list (AIList)

An AIList 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 AIList is **O(log2N+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.

4.10 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:

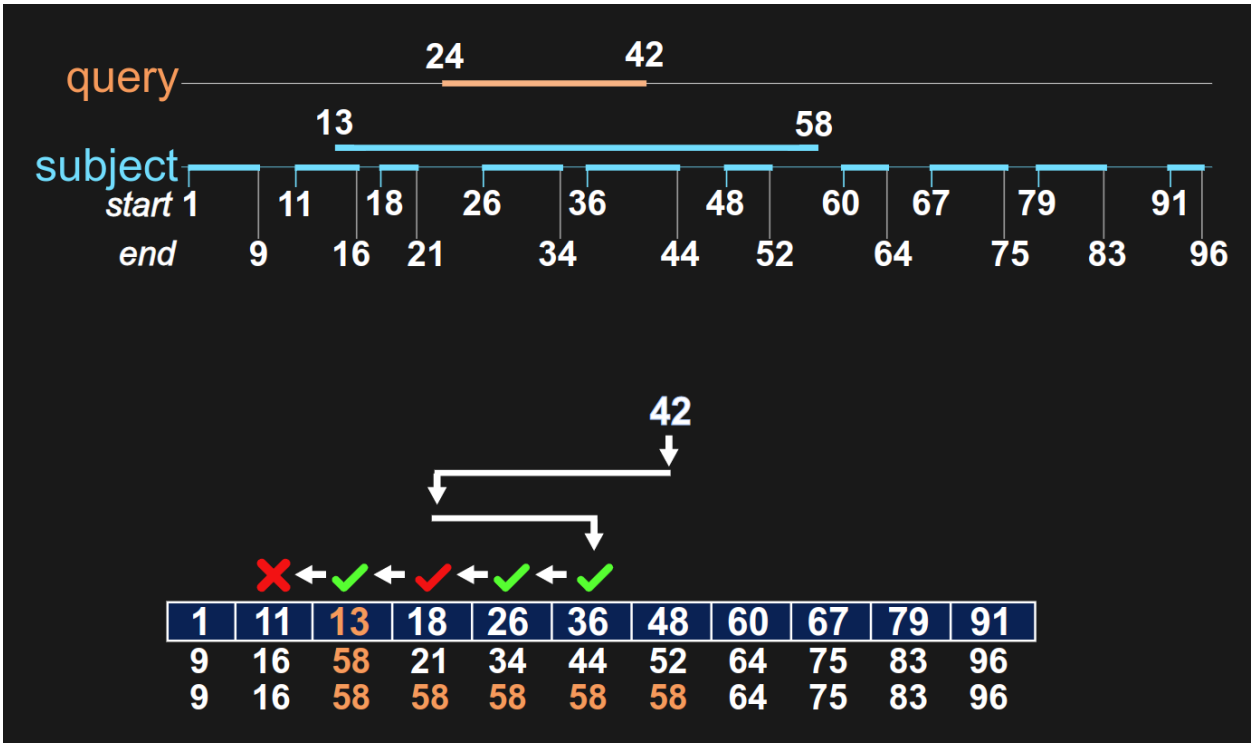


Figure 4.17: AIList

```
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 devided 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
```

4.11 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.

4.11.1 INTERVAL SIMILARITY METRICS

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

4.11.2 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 (Intergrated Genome Database), Feng and Sheffield 2021
  - Uses a linear binning of an array

4.12 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. Dfferent data structures and algorithms enable large-scale genomic interval analysis. And, new methods for genomic intervals are being developed.

4.13 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 chromosme to refere 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 interecting region

Chapter 5

ATAC-seq diagnostics and harmonization

5.1 ATAC-seq Overview

5.1.1 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)

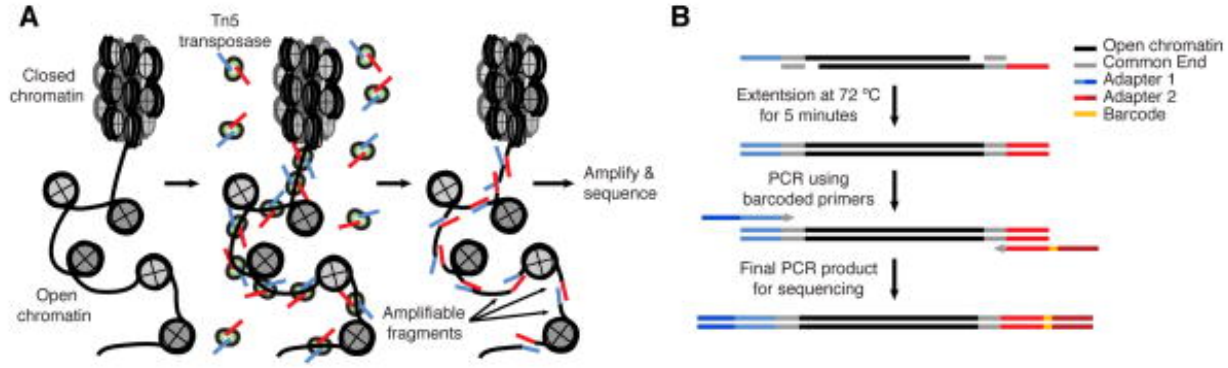


Figure 5.1: ATAC-seq Overview

5.1.2 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.

5.1.3 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.

5.1.4 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.

5.2 Basic Algorithm for Sequence-Based Genome Data

5.2.1 Basic 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.

5.2.2 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.

5.2.3 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

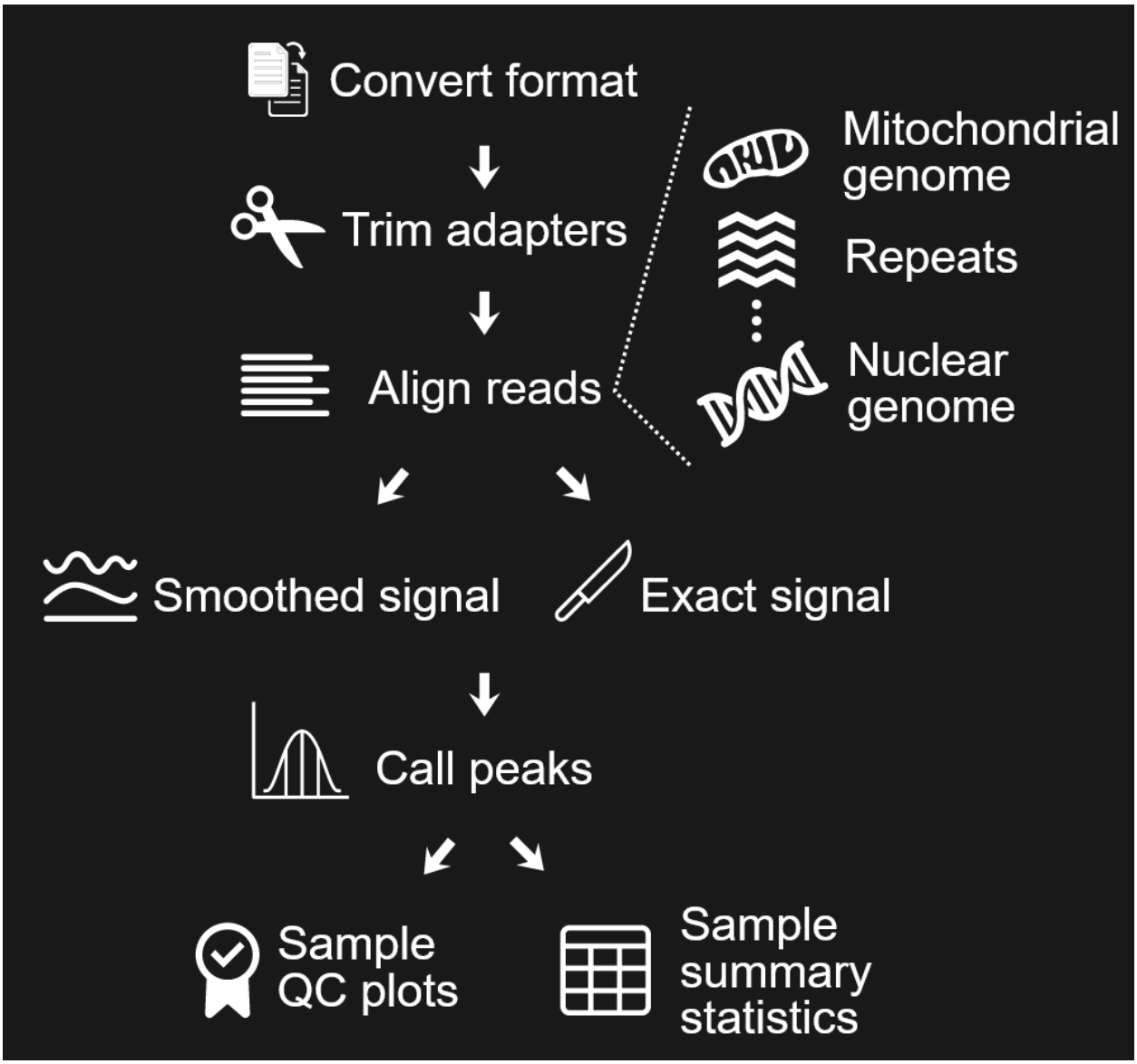


Figure 5.2: ATAC-seq workflow

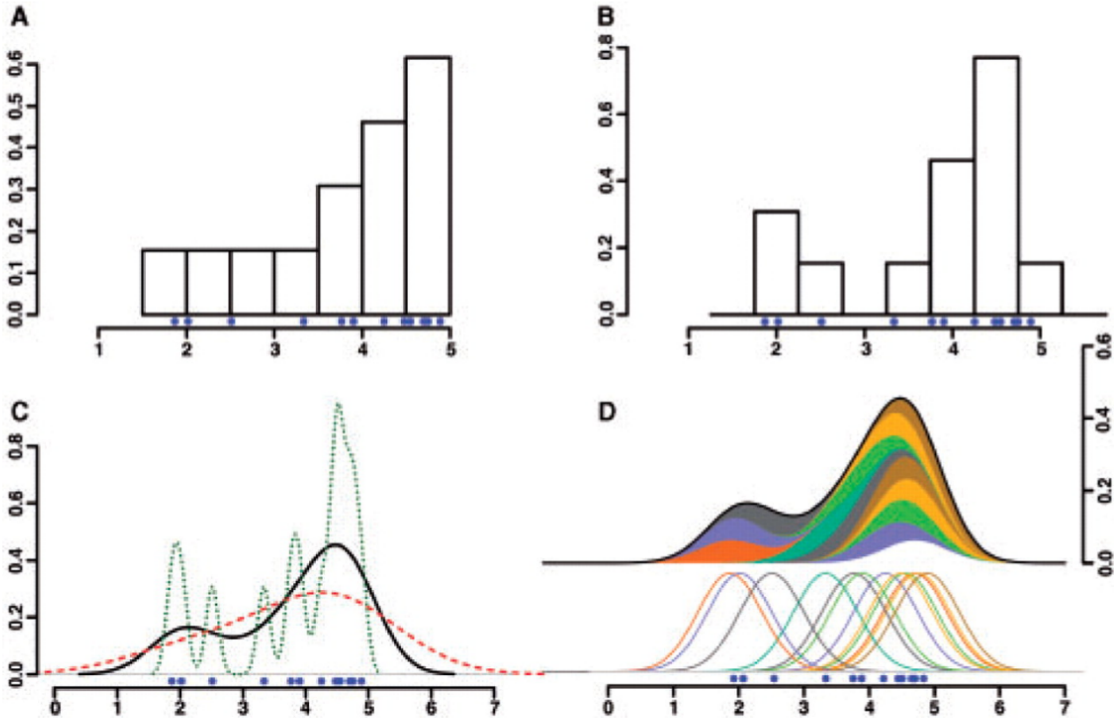


Figure 5.3: Examples of histogram and density estimation properties.

2) Aggregate reads 2000 bp around TSSs
3) Tile into 100bp windows
4) $S_{TSS} = /fracTSSbackground$

- 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)

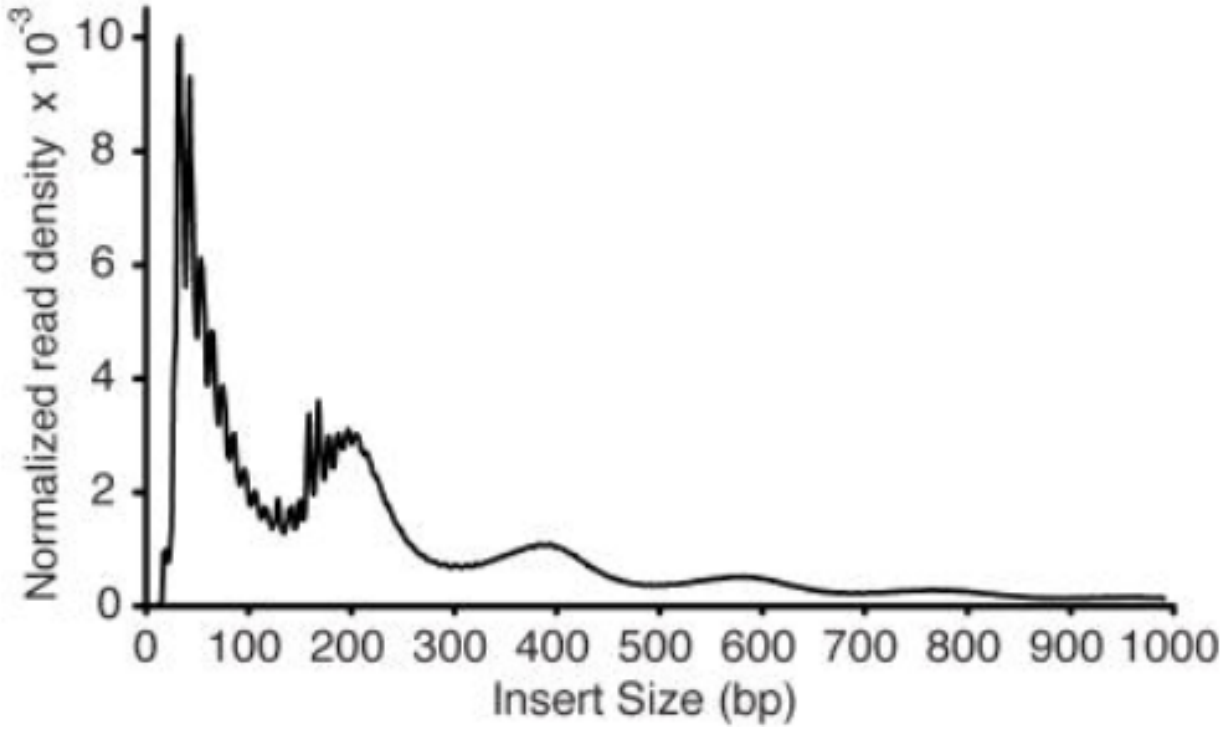


Figure 5.4: ATAC-seq fragment distribution

5.3 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.

5.3.1 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.

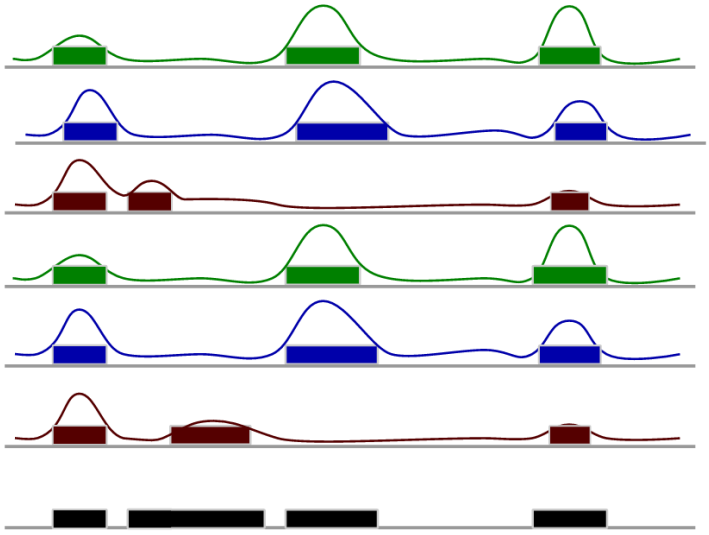


Figure 5.5: Union

- Tile boundaries are artificial and may split meaningful units.

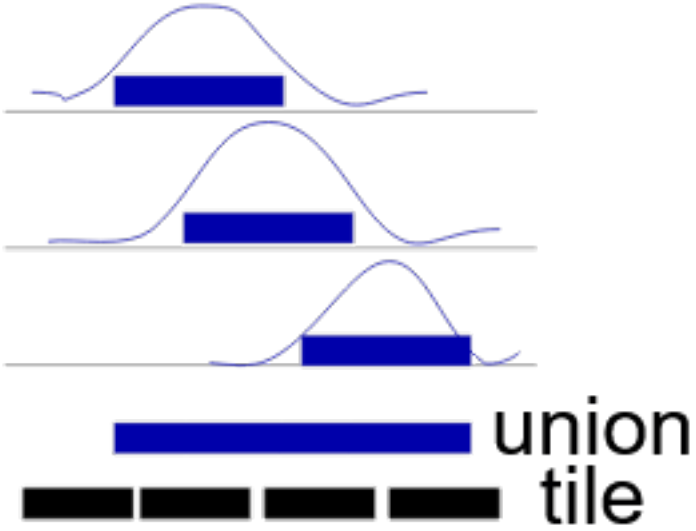


Figure 5.6: Union-tile

- 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:

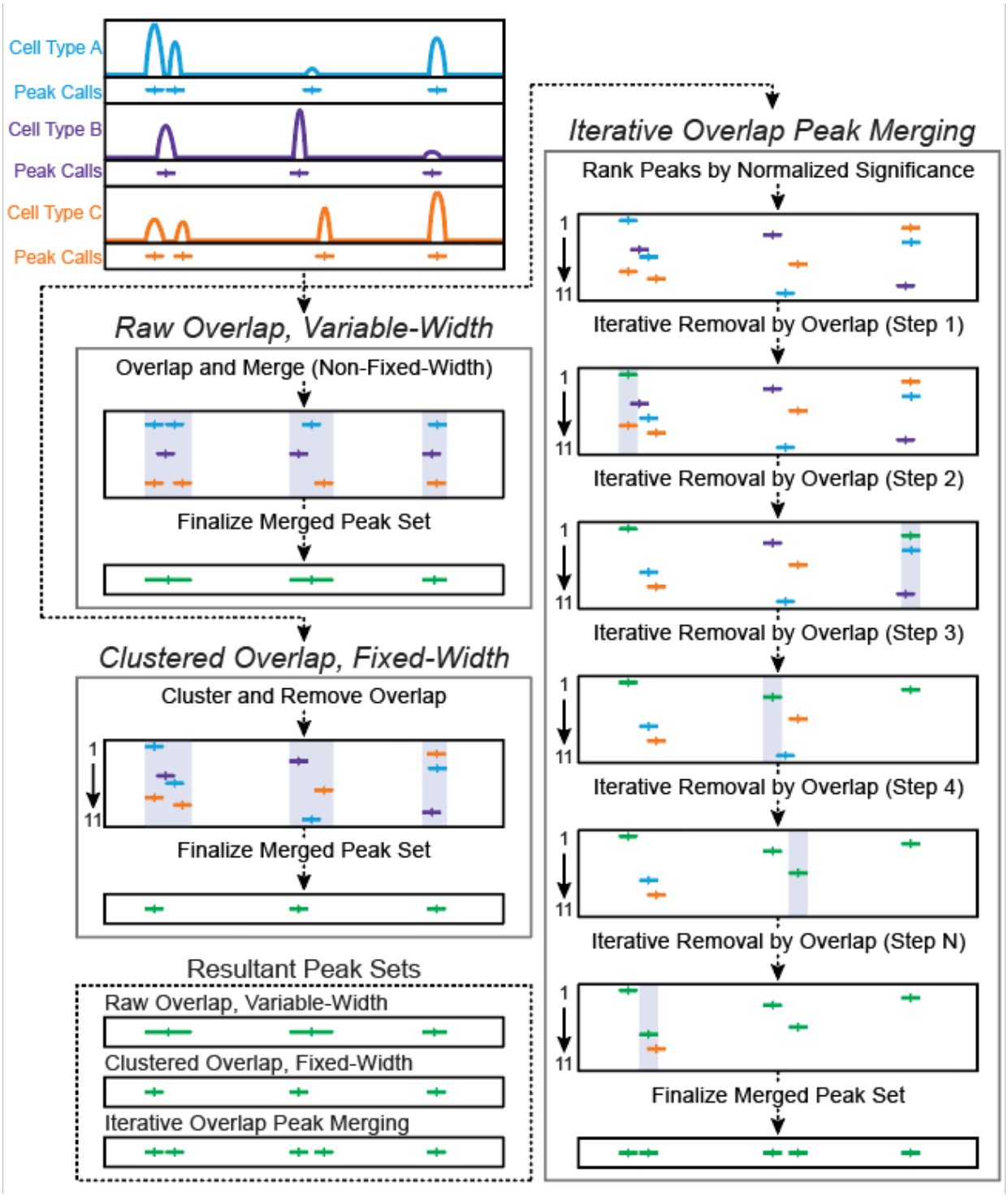


Figure 5.7: Iterative overlap

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

5.3.2 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.*

5.3.3 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:

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

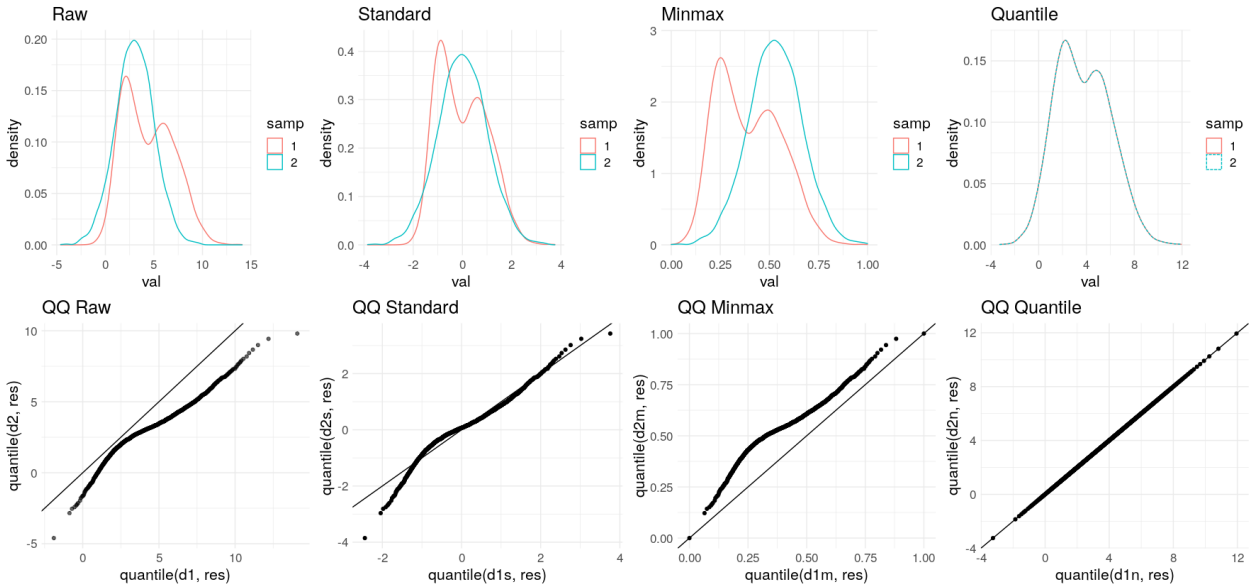


Figure 5.8: Quantile-quantile plot example 1

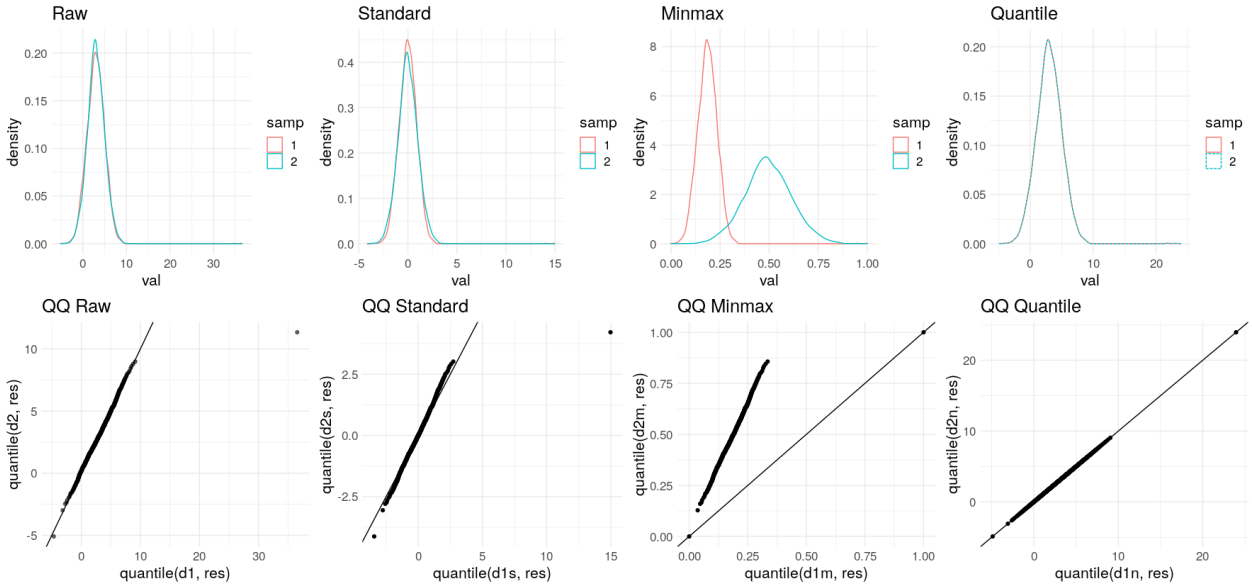


Figure 5.9: Quantile-quantile plot example 2

5.4 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)

5.4.1 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

5.4.2 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 6

Scalable Computing in Genomics

6.1 Lower costs → More data

The reduced cost of sequencing has lead 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.

6.2 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.

6.3 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.

6.4 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.

6.5 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 seuqencer, 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 excitation method, whereas common workflow language is designed to be a universal/standardized language that can be run on different engines/independent of execution engine.

Chapter 7

Stop using shellscipts 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.

7.1 Advantages of workflows

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

7.2 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 algorithms belongs to, we can see how well it scales (see plot in the slides.) Factorial time O 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(logn) (Manhattan phonebook.) Indexing reduces lookups from linear time to constant time. Tabix indexing allows us to randomly access anywhere in the file.

7.3 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.

7.4 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

7.5 Problem with Cloud Platforms

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

Chapter 8

Class 21: Differential Expression Analysis

8.0.1 Computational Genomics Scribe Notes 4/26/22

8.1 The Transcriptome

The transcriptome consists of various RNA species that are transcribed from the genome. The amounts of each species depends on how fast it is transcribed and how stable the RNA is. RNA is degraded in the cytoplasm, so each transcript may have a different half life.

mRNA, or messenger RNA, contains exons of genes, and can be transcribed into proteins. Usually, scientists are most interested in assessing the levels of mRNA species. mRNA length depends on length of gene.

pre-mRNA is the unprocessed version of mRNA. It contains introns, which are spliced out. Further, enzymes add a 5' cap and a 3' poly-A tail to protect the mRNA from degradation in the cytoplasm. Longer than its corresponding mRNA

rRNA, or ribosomal RNA, makes up the ribosomal sub-units and help in protein synthesis

tRNA, or transfer RNA, is able to specifically bind both RNA and amino acids. During protein synthesis, a tRNA will recognize the 3-nt codon sequence, and will transfer the corresponding amino acid to the growing peptide chain.

snRNA, or small nuclear RNA, are components of the spliceosome that contribute to removing introns from pre-mRNA.

snoRNA, or small nucleolar RNA, guide modification of other RNAs. They often do this by methylating or pseudouridylating their target RNAs.

miRNA, or microRNA, are about 22 nt and contribute to gene regulation, mostly by binding complementary mRNAs and silencing gene expression.

siRNA, or small interfering RNA, are about 21 nt double stranded RNA molecule with overhanging 5' and 3' ends. Similar to miRNA, siRNAs bind complementary mRNAs and silence their expression.

lncRNA, or long noncoding RNA, are RNA molecules longer than 200 nt. They can regulate protein abundance at the transcriptional, RNA processing, and translational level.

asRNA, or Antisense RNA, is a single stranded RNA that is complementary to an mRNA molecule. asRNA can bind and inhibit translation of the mRNA.

When performing a sequencing experiment, it is important to think about which RNA species you want to measure. Specific protocols have been developed to look at mRNAs, miRNAs, and even nascent transcripts. Developments in long read sequencing have allowed us to look at differences in isoforms and alternate splicing.

8.2 Splicing

Each gene consists of introns and exons. The spliceosome is a collection of enzymes responsible for recognizing the boundaries of exons and removing introns.

Some genes have more than one functional form due to different exon usage. Isoform A may use exons A, B and C, while isoform B uses exons B and C only. Genetic variation and abundance of cytoplasmic enzymes can effect the abundance of each isoform.

With traditional short read sequencing, we can only identify different isoforms if the read fragments happen to overlap an exon-exon boundary. Long read sequencing allows us to read entire mRNA molecules and determine every exon present in that transcript.

8.3 Gene Expression Quantification

Genome wide quantification of transcription is relatively new technology. The first major development was the microarray chip, which has quickly been replaced by short and long read RNA-sequencing.

Micro-Arrays- The first genome-wide transcriptional assay was the micro-array. These were first produced by Affymetrix. Microarrays consist of probes that hybridize with specific RNA molecules. The probes are grouped into locations on a chip. When the target DNA binds to the probe, fluorescent signal is released. The chips are imaged to quantify the relative amounts of each probe.

A major limitation is that you can only measure known transcripts for which you have a complementary probe. This makes it hard to profile both unknown genes and unknown isoforms of genes.

RNA-seq- RNA-seq is currently the most widely used way to profile the transcriptome. In next generation sequencing, reads are ligated to adapters in a flow cell. Then each read is replicated many times to ensure a measurable signal. Then, fluorescently labeled NTPs are added, the RNA is replicated, and the fluorescent signal is measured after each base pair addition. In this way, RNA-seq can measure many transcript species.

The quality of the RNA-seq data will depend on the sequencing depth and the coverage across the genome, as well as the quality of the input RNA library.

8.4 Gene Expression Analysis

Usually, we want to know which genes are turned on or off in the presence of some treatment, as compared to the control. In this case, we want to quantify the relative amounts of mRNA species and test whether the amounts differ between control and treatment. This often results in a massive gene list, so a final step is often to perform pathway enrichment analysis using published gene sets. In addition, we will perform quality control checks on our data and will normalize across samples.

8.4.1 Typical Workflow:

8.4.2 1. Quality Control

Microarray and RNA-seq data can contain errors and biases due to the technology used to measure the gene expression and the input RNA library. To identify and remove these errors, researchers perform extensive quality control tests on these data. These quality metrics differ for arrays and RNA-seq. Array data must be controlled for multiple chips and for input RNA quality. RNA-seq errors are more complex; they are often due to the amplification of certain reads or types of reads. Much of quality control is identifying duplicated reads, duplicated k-mers, and unexpected GC content that could be indicative of large scale errors. Further,

RNA-seq reads usually begin very high quality, however more and more errors are introduced toward the end of the read. Metrics showing the overall quality and quality per base pair are important in determining where to trim reads and which to remove entirely. For either array or RNA-seq, QC is an important step in the gene expression analysis pipeline.

8.4.2.1 Tools:

FASTQC - A method for identifying and removing low quality reads from raw RNA-seq data. This method performs a variety of quality control steps to identify biases and problems due to both the sequencer and the input library. FASTQC takes in FASTQ input files (or SAM/BAM aligned reads). For each input file, a number of analysis modules are run. Each module tests a specific quality metric, such as quality scores per base and per sequence, GC content, length distribution, and k-mer enrichment. For example, the k-mer enrichment module assumes that any 7-mer should be found equally distributed in any position in a read. This module uses a binomial test to determine if any 7-mer is more enriched in certain positions. This may indicate duplication of reads. For more info on the other modules, check out https://www.bioinformatics.babraham.ac.uk/projects/fastqc/

8.4.3 2. Alignment

Reads must be aligned to genes or transcripts to quantity the amount of gene expression. Alignment was covered in more depth in lectures 4 and 5. Briefly, aligning RNA is slightly more difficult due to the lack of introns. Each exon must be matched separately to its unique location in the genome. Several tools use different methods to address this problem (listed below). Array data does not need to be aligned.

8.4.3.1 Tools:

HISAT - A method for aligning NGS reads to reference genomes. In HISAT, the Burrows-Wheeler Transform (BWT) is used to create a compressed version of the reference genome. Using the FM index, the reference can be searched quickly. HISAT creates one genome-wide index, and then breaks the genome into smaller genomic regions with local indexing. Therefore, when reads span exon junctions, the large part can be globally aligned, then the search space is reduced to the local alignment to find the smaller exon. This method uses less memory intensive than its competitors. More info can be found here: https://www.nature.com/articles/nmeth.3317 and here: https://www.nature.com/articles/s41587-019-0201-4

STAR - This method aligns RNA-seq reads to a reference genome STAR stores the reference genome in an uncompressed suffix tree. STAR then tries to identify and map the longest continuous exon in the read. Then, STAR separately searches for the remaining unmapped portion of the read. STAR then stitches the two parts together into a full transcript. Because the two searches are independent and the suffix tree structure is quickly searchable, this method is very fast, but memory intensive. More info here: https://academic.oup.com/bioinformatics/article/29/1/15/272537

8.4.4 3. Estimation of Expression Levels

Once we have aligned reads to the genome, we want to estimate the amount of mRNA is present for each gene. Essentially, we want to estimate the mean expression value for each gene. This is complicated by the fact that most experiments have 2-4 replicates on average, due to the high cost. This makes estimates of variance very unreliable. To account for this, many tools attempt to pool information across probes and across samples to strengthen the estimations. For microarray data, we perform normalizations before estimating expression.

8.4.4.1 Tools:

dChip - This method estimates expression values from array data by pooling data across samples. dChip tries to estimate the expression of each probe in each array by setting the measured expression minus measured background equal to the relative expression level times the affinity of the RNA for the probe. The measured background is based on the binding of mismatch containing probes compared to perfect matches.

Then, the relative expression and affinity term are iteratively fit. This enables accurate quantification of gene expression across arrays. More info is hard to find unless you have the book, but check out this: https://academic.oup.com/bioinformatics/article/20/4/500/192364

RMA - This method estimates expression values from array data by pooling data across samples. RMA follows a similar framework to dChip. However, they noticed that measured expression is usually log scale. Therefore, they set log(measured expression) equal to log(relative expression) + log(affinity) + a constant background term. Again, they iteratively fit the expression and affinity terms to get a good estimate across arrays. These changes make RMA outperform dChip. For more info: https://pubmed.ncbi.nlm.nih.gov/12925520/

GCRMA - This method estimates expression values from array data by pooling data across samples. GCRMA is also based on the dChip and RMA algorithms. Essentially, the formula is the same as dChip with a non-constant background term. They found that the background signal strongly depends on the probe sequence. Therefore, the background term here is a function of both probe sequence and probe with one mismatch binding. This correction improves its performance over RMA and dChip. More info can be found here: https://www.jstor.org/stable/pdf/27590474.pdf

Rsubread - This tool performs alignment to a reference and estimates counts for RNA-seq reads. First, Rsubread creates an index of the reference genome by creating a hash table. Next, they use a two pass strategy to align reads. First, Rsubread selects 16-mers from the read and maps these to the reference via the hash table. By selecting multiple 16-mers, they identify the largest location of mapping. Then, local alignment is used to map the rest of the read. This accounts for introns and indels. Reads are then counted if they overlap features. Features can be genes, exons, promoters, etc. Features and degree of overlap are user defined. More info can be found here: https://academic.oup.com/nar/article/47/8/e47/5345150

StringTie - This method aligns RNA-seq reads and annotated transcripts and calculated their expression levels simultaneously. StringTie first maps reads to a reference genome using a spliced-based aligner. An optional step is to assemble "super-reads", or multiple unique transcripts that map together that can be treated as a single read. Then, StringTie groups all reads mapped to the same gene. From these reads, a splice graph is constructed. This graph shows "paths" of possible exon usage in the gene, and "weights" based on how many transcripts follow that path. In this way, a relative expression value can calculated for each isoform in the gene. More info here: https://www.nature.com/articles/nbt.3122#Sec2

RSEM - A method for aligning RNA-seq reads to reference transcripts and quantifying gene expression. RSEM does not require a reference genome, and is able to work with reads aligned de novo. It only requires reference transcripts, which can be input be the user from a de novo alignment. RSEM creates a graphical model with latent and observed variables. The observed variables are read lengths, quality scores, and sequences and the latent variables are the transcript from which the read was derived and positional information. The primary variable is the probability that a read comes from a transcript. Using this model, RSEM uses Expectation-Maximization to estimate this probability and the probabilities of the unobserved variables until the values converge. The output is a wiggle file. More info can be found here: https://bmcbioinformatics.biomedcentral.com/articles/10.1186/1471-2105-12-323

Sailfish - This method estimates transcript abundance from annotated RNA isoforms. Sailfish is one of the first tools to implement k-mer hash tables for pseudo-alignment. Here, they divide a reference genome into its k-mers and store k-mer- transcript pairs in a hash table. Then, each read is divided into k-mers and hashed against the index to determine which transcripts it could be derived from. Then, using expectation-maximization, transcript abundances are estimated from average k-mer coverage and read k-mers are reassigned to transcripts. More info here: https://www.nature.com/articles/nbt.2862

kallisto - This method estimates transcript abundance from RNA-seq reads, both bulk and single-cell. kallisto uses two newer methods, de Bruijn graphs and k-mer hash tables, to drastically improve the speed and memory of expression quantification. Further, kallisto does not require aligned reads, instead uses k-mers. First, de Bruijn graphs are constructed using k-mers present in the reads. Paths through the de Bruijn graphs represent transcripts observed in the data. Each k-mer and its transcript compatibility, or equivalence class, are stored in a hash table. For each read, we can hash each k-mer to see which transcripts it could have pos-

sibly come from. Then, expectation-maximization is used to estimate the probability that a read came from a transcript, similar to other tools. More info can be found here: https://www.nature.com/articles/nbt.3519

Salmon - A method for estimating the expression of transcripts from RNA-seq data. Salmon improves on kallisto and Sailfish by controlling for sequence specific biases, GC content, and positional biases. Salmon can input aligned or unaligned reads. First, salmon estimate initial abundances for each transcript and computes the possible transcript compatibility, or equivalence class for each read. In the second step, expectation-maximization is used to estimate the probability that a read came from a transcript, similar to other tools. Salmon adds GC content and sequence information to the model to improve estimation and lower false positives. More info here: https://www.nature.com/articles/nmeth.4197

8.4.5 4. Normalization Across Samples

When estimating gene expression, it is important to have comparable replicates and samples. In micro arrays, batch effects are introduced when using more than one Chip, while in sequencing, the sequencing depth and read length strongly influence RNA-seq quantification. An increase in sequencing depth or gene length would lead to more reads per gene by chance, and this should be accounted for. Further, RNA-seq is skewed by strong differentially expressed genes, so accounting for RNA species is important as well. Often, we present RNA-seq reads as transcripts per kilobase per million reads (TPM), or reads per kilobase exon per million reads (RPKM) which accounts for length and total sequencing depth.

8.4.5.1 Tools:

DESeq2 Normalization - DESeq2 normalization creates a metric similar to TPM that accounts for sequencing depth and RNA composition. For each gene, DESeq2 takes the geometric mean the expression across samples to create a pseudo-reference. Then they divide each sample by the reference and finds the median value per sample. Then, each value in the sample is divided by this median value. This takes all genes into account when normalizing. More info can be found here: https://genomebiology.biomedcentral.com/articles/10.1186/s13059-014-0550-8

EdgeR Normalization - EdgeR normalizes data by computing scaling factors for the library size used in the TPM calculation for each sample. The scaling factors are chosen because they minimize fold change between samples. EdgeR uses weighted trimmed mean of M-values (TMM) to calculate the differences. In this procedure, the upper and lower parts of the data are trimmed, then a mean is taken. More info here: https://genomebiology.biomedcentral.com/articles/10.1186/gb-2010-11-3-r25

Quantile Normalization - covered in more depth in lecture 16. This is a method that forces two data sets to share the same mean and distribution. In brief, each data set is rank ordered, the mean of each rank is found, then the rank means are substituted in the original data sets in place of the value found at that rank. This tool is very good at removing batch effects in array data. More here: https://en.wikipedia.org/wiki/Quantile_normalization#:~:text=In%20statistics%2C%20quantile%20normalization%20is,an

8.4.6 5. Differential Expression Anlaysis

Once we have determined that are data is high quality, our reads have been aligned to the genome/ transcriptome if needed, and data has been normalized between samples, we often want to identify genes that change drastically between two experimental conditions. These changes in gene expression often lead to the observable phenotype that we want to study. Therefore, identifying genes that change in response to treatment with a small number of false positives is usually the goal of the researcher.

We perform differential expression analysis to determine which genes change expression levels between groups. At its most basic, we can simply test whether the mean expression is significantly higher in one group vs another, perhaps using a t-test. Usually, we are interested in genes that are both significantly differently expressed, and genes that change expression by a large amount. We use volcano plots to show log-fold-change and significance levels, and usually select the most drastic in both categories.

There are a few problems with this basic method, first, their are thousands of genes in the genome. Correcting for testing each gene is an important step. Secondly, usually we have very few samples for each treatment/ phenotype. The cost of sequencing is decreasing, but experiments usually will have 2-4 replicates only. This makes estimating variances challenging and unreliable. Further, because the n is small, outliers will have a large effect and must be dealt with. To solve these problems, many tools will pool information across genes to improve these estimations of variance. Some of these tools are described below.

8.4.6.1 Tools:

limma - This tool performs differential gene expression analysis on array and RNA-seq data. limma uses linear modeling to obtain estimates of gene expression between samples. limma organizes the data into a matrix of genes and samples, where each sample belongs to a treatment/ phenotype group. Then, for each gene, a linear model is made that sets the expectation (mean value) of each gene equal to a coefficient matrix times a design matrix that describes the experimental groups. From this linear model, coefficients can be computed. Then, limma creates a modified t-statistic. Using empirical Bayes methods, they adjust the variance of each specific gene toward the pooled variance across all genes, and they increase the degrees of freedom. In this way, limma identifies genes that distinguish each experimental group best. More info can be found here: https://academic.oup.com/nar/article/43/7/e47/2414268?login=true

edgeR - edgeR is a tool for analyzing differential gene expression. edgeR starts with a matrix of expression counts across genes and samples. They create a model where the observed counts data is equal to a negative binomial model that depends on the total number of reads, the abundance/mean value for the gene in each treatment group, and a modified variance. The variance is again modified with a dispersion coefficient that describes the differences between samples. The dispersion per gene and the abundance per group are first estimated using maximum likelihood, then refined using empirical Bayes. Finally, edgeR uses a test similar to the Fisher's Exact test that has been adapted for over-dispersed data. This test identifies differentially expressed genes. More info can be found here: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2796818/pdf/btp616.pdf

DESeq2 - This tool estimates differential expression from RNA-seq data. DESeq2 starts with a matrix of expression counts across genes and samples. Like edgeR, they create a model that sets the observed counts data equal to a negative binomial model with a mean and dispersion term. The variance is modeled as a mean term and a dispersion term, which allows researchers to model both biological and technical variation. The mean is modeled as the concentration of reads in a gene times a normalization factor that accounts for differences in sequencing depth. Genes that are not differentially expressed have the same normalization factor between samples, while DE genes do not. First, DESeq2 estimates dispersion for each gene using maximum likelihood. They then use empirical Bayes to shrink these dispersion values towards the global trend. Finally, they provide the maximum a posteriori estimate for each dispersion and use this dispersion value to shrink the fold changes of genes with high dispersion (variance). Then, the same procedure is used to estimate log fold change. They first use a linear model to estimate the maximum likelihood for each fold change. Then they fit a transcriptome wide distribution and use empirical Bayes to modify the fold changes toward the common distribution. Finally, the give a maximum a posteriori estimate for each log-fold change. More info here: https://genomebiology.biomedcentral.com/articles/10.1186/s13059-014-0550-8

8.4.7 6. Gene Set Enrichment

Sets of differentially expressed (DE) genes, or even sets of highly expressed genes can be long lists, often hundreds or thousands of genes. This would be time consuming to manually search. Further, genes are activated in programs or pathways that function together to achieve some cellular function.

We can use databases of gene sets, pathways, interaction networks, etc to make sense of our large gene sets. Many organizations exist to curate and organize what is known about genes, pathways, disease, mechanisms, etc, into distinct gene sets. Some of these databases maintain non-overlapping gene sets, but most adhere to a hierarchical structure of overlapping gene sets that span broad to specific categories. Other databases not listed below include Kegg, Gene Ontology, Reactome, and Wikipathways.

To determine whether our gene set is similar to any published gene sets, we often look for enrichment of the published gene set in our genes compared to background. For example, if our set of 50 DE genes contains 20 Wnt signaling genes (40%), but our background universe of genes is only 3% Wnt signaling genes. We can use the Fisher's Exact Test, a hypergeometric test, or a leading edge test to determine if there are significantly more Wnt related genes in our DE set.

8.4.7.1 Tools and Databases:

MSigDB - The molecular signatures database is a database of almost 40,000 annotated gene sets. The database contains gene sets such as "Hallmark Apoptosis" or "Adipogenesis at 8HR". Each gene added to the set has been published and verified to be involved with the gene set term. Using MSigDB, researchers can learn about a particular gene, identify genes involved with certain processes, or test an input gene set for enrichment for any of the MSigDB gene sets. GSEA can be performed on the MSigDB gene sets. More info here, but requires login: https://www.gsea-msigdb.org/gsea/msigdb/index.jsp

GSEA - Gene set Enrichment Analysis (GSEA) is a tool for interpreting gene expression data. First, samples are divided into control vs treatment, or any two phenotypes. Then, the genes are ranked by their correlation to the treatment. Then, using this ranked gene list, GSEA determines whether genes from a curated gene set are found at the top/bottom of the ranked list or randomly distributed. This is called the leading edge test. They compute an enrichment score by walking through the data and increasing the running sum when genes from the curated set are encountered. Using a permutation test, they determine how often the enrichment score happens randomly. Finally, they correct for multiple tests. More info can be found here: https://www.pnas.org/doi/10.1073/pnas.0506580102

String - String is a database of protein-protein interactions. They curate both functional and direct protein interactions from published data, functional studies, genomic prediction, and co-expression. You can query individual proteins to see their interaction network. Further, you can input a set of genes/ proteins and test for enrichment of published gene sets and String interaction modules. They use a hypergeometric test to determine over-representation of the String modules in the input gene set (expected vs observed). More info here: https://academic.oup.com/nar/article/47/D1/D607/5198476?login=true and here: https://string-db.org/cgi/about?footer_active_subpage=content