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.

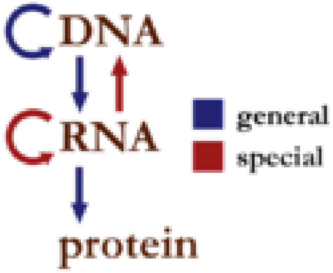

Figure 1.1: Information flows between DNA, RNA and protein. Source: Wikipedia

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

Hidden Markov Models

2.1 Overview

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

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

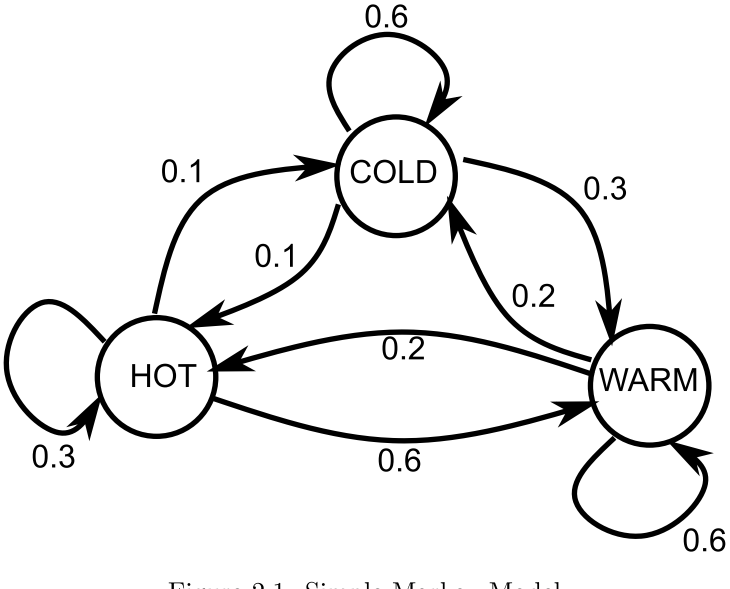

Figure 2.1: Simple Markov Model

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

A Markov chain is useful when we need to compute a probability for a sequence of observable events. In many cases, however, the events we are interested in are hidden: we don't observe them directly. For example, we don't normally observe part-of-speech tags in a text. Rather, we see words and must infer the tags from the word sequence. We call the tags hidden because they are not observed.

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

2.2 Mathematical Definition(s)

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

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

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

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

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

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

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

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

2. **Initial Probabilities (π):**

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

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

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

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

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

2.3 An Example

The following example problem is pulled from wikipedia:

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

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

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

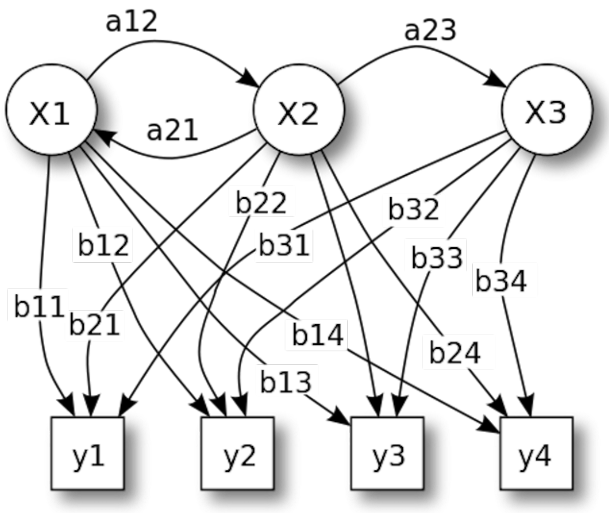

Figure 2.2: Simple hidden Markov model. Source: Wikipedia

```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.

2.4 Computational problems with HMMs

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

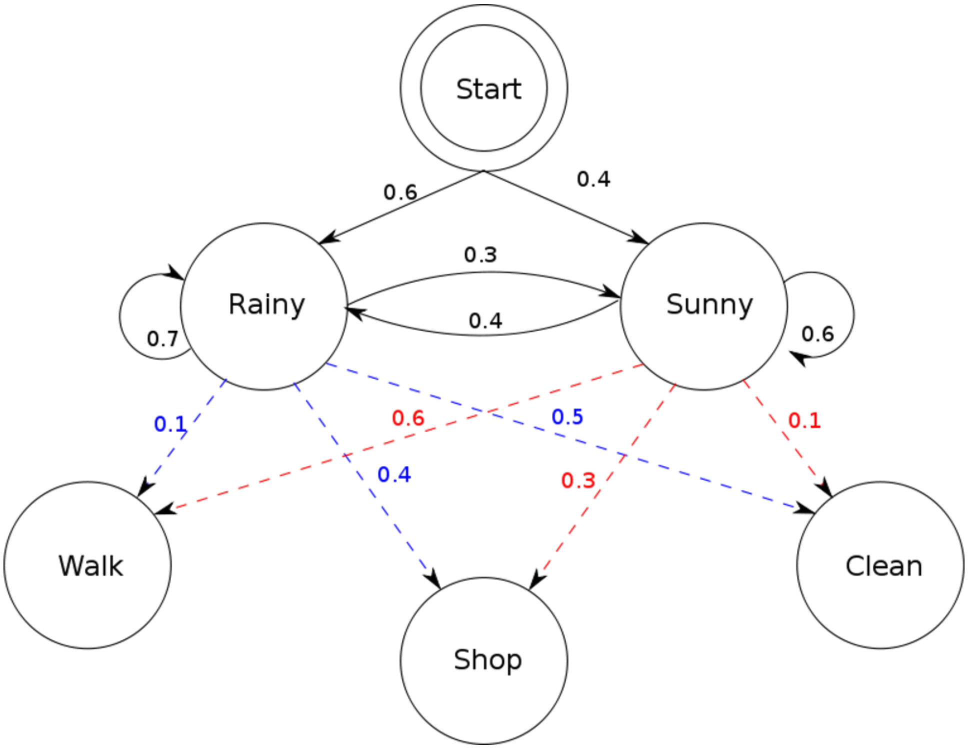

Figure 2.3: Hidden Markov Model to predict weather. Source: Wikipedia

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

2.4.1 Decoding problem :

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

2.4.2 Likelihood Problem:

Similar to the above decoding problem, given the HMM, M=(A,B,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 probabilities of observations given a set of states:

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

However, the state sequence is unknown.

2.4.3 Learning problem:

Given an observation sequence, O, and general structure of HMM, determine HMM parameters that best fit the training data. Here we are solving a sort of reverse problem. That is, we **do not know** the specific probabilities of the transition or emission states. All we know is the overall structure of the model, and using a set of training data, we can fit our model to produce “optimal” values for A, B, and π, such that the model can be applied elsewhere.

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

2.5 Conclusions

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

2.5.1 Pros:

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

2.5.2 Cons:

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

Chapter 3

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

3.1 Brief review of linear models

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

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

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

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

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

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

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

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

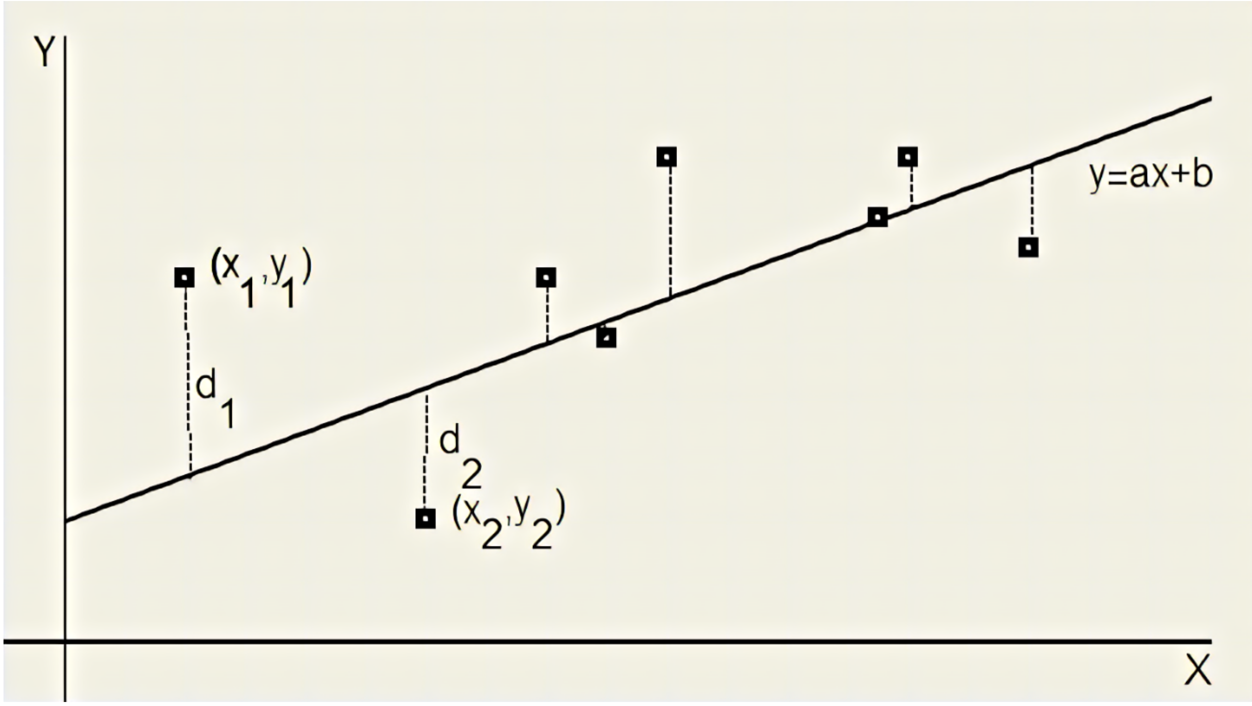

Figure 3.1: Method of least squares seeks to minimize the sum of squared residuals

3.2 Relation to Hypothesis Testing

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

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

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

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

3.3 Relationship to Correlation

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

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

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

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

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

3.4 Multiple Linear Regression

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

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

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

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

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

.

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

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

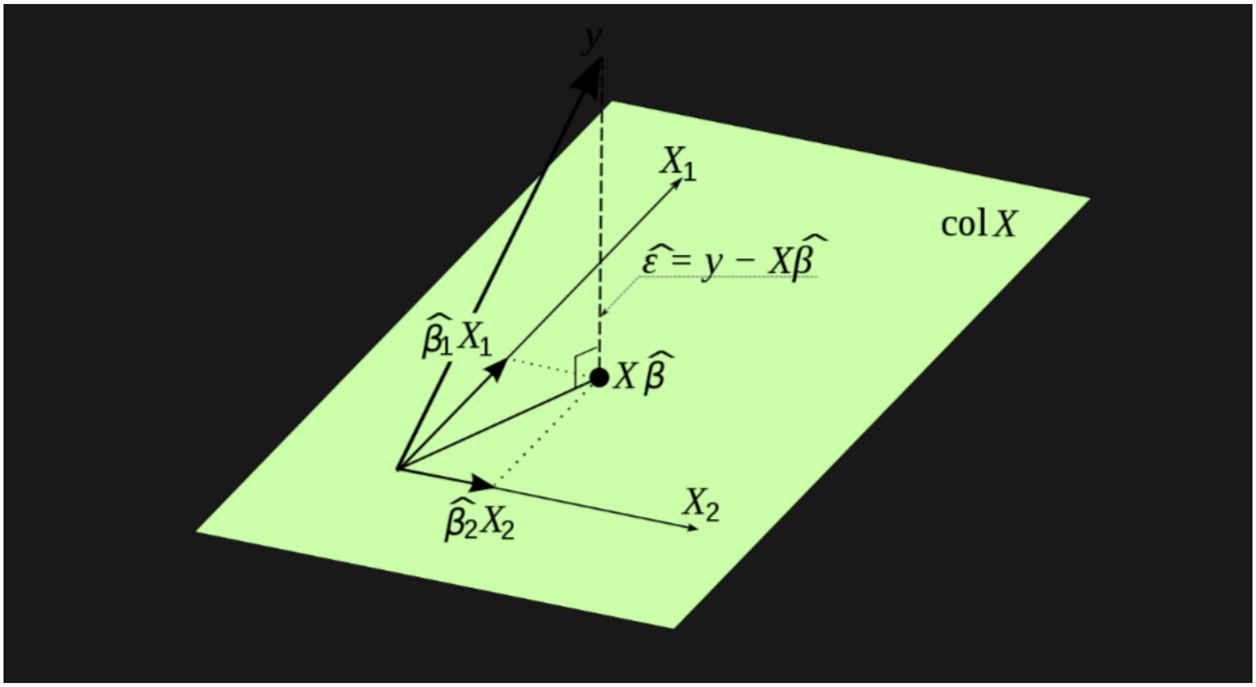

Figure 3.2: OLS estimation can be viewed as a projection onto the linear space spanned by the regressors. Source: Wikipedia

3.5 Chi-Squared Test of Independence and Relationship to GWAS

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

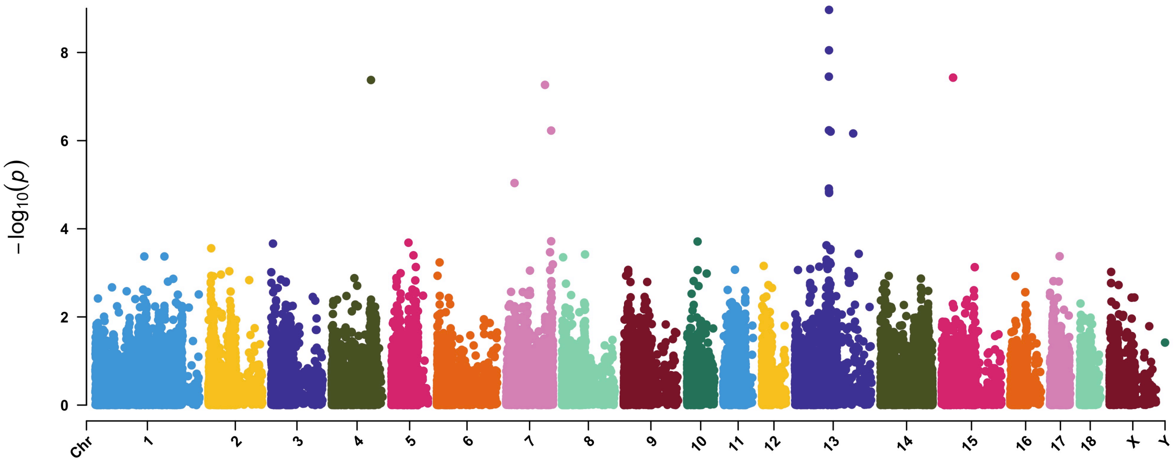

Figure 3.3: An example Manhattan plot

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

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

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

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

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

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

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

Chapter 4

ChIP-seq

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

4.1 ChIP-seq experiment steps:

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

4.2 History of the development of ChIP-seq technology

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

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

4.3 ChIP-seq data analysis

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

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

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

3. Redundancy control: completely identical reads are considered error (for example, induced by PCR)

- Non-redundant rate:The ratio of the number of non-redundant reads to the number of mapped reads
- PBC (PCR Bottleneck Coefficient): The ratio of the number of locations with 1 read mapped to the number of locations with reads mapped

4. DNA fragment size estimation:

- peak model (MACS) for TF
- cross-correlation (SICER) for any ChIP-seq (input): calculate the Correlation between two strings with a displacement; Auto-correlation: Cross-correlation with itself

5. Retrieve DNA fragments

- Full length retrieval (MACS)
- Partial retrieval (sharpen the signal)
- Point retrieval (SICER)

6. Pile up: Signal map generation

4.4 ChIP-seq: Study design

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

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

4.5 ChIP-seq: Peak calling

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

MACS: Model-based Analysis for ChIP-Seq Read distribution along the genome - Poisson distribution (λBG = total tag / genome size) - Negative binomial distribution (MACS2)

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

B-H adjustment to correct for FDR : p-value → q-value

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

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

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