UNIVERSITY OF ADELAIDE

A MACHINE LEARNING APPROACH FOR DETECTING SELECTIVE SWEEPS USING ANCIENT DNA

by

Shing Yan Kwong

A thesis submitted in partial fulfillment for the
degree of Master of Philosophy

in the
Faculty of Engineering, Computer and Mathematical Sciences

July 2021

Declaration of Authorship

I, Shing Yan Kwong, declare that this thesis titled, 'A Machine Learning Approach For Detecting Selective Sweeps Using Ancient DNA' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

- This work contains no material which has been accepted for the award of any other degree or diploma in my name, in any university or other tertiary institution and, to the best of my knowledge and belief, contains no material previously published or written by another person, except where due reference has been made in the text. In addition, I certify that no part of this work will, in the future, be used in a submission in my name, for any other degree or diploma in any university or other tertiary institution without the prior approval of the University of Adelaide and where applicable, any partner institution responsible for the joint-award of this degree. I give permission for the digital version of my thesis to be made available on the web, via the University's digital research repository, the Library Search and also through web search engines, unless permission has been granted by the University to restrict access for a period of time. I acknowledge the support I have received for my research through the provision of an Australian Government Research Training Program Scholarship.

Signed: Anthony Kwong

Date: 15th Jan 2021

UNIVERSITY OF ADELAIDE

Abstract

Faculty of Engineering, Computer and Mathematical Sciences

Master of Philosophy

by Shing Yan Kwong

Biological adaptation leads to specific patterns in population genetic data called selective sweeps. Although researchers have applied machine learning to sweep detection, which specific methods are appropriate for any given scenario is not well understood. We conducted a systematic review of a suite of machine learning(ML) classifiers for sweep detection. We found that accurate models can be built using simple, fast classifiers supported by preprocessing. We produced a ML workflow which is applicable for general population genetic problems. Our methods were extended for ancient DNA, showing a sweep signal can be retrieved even at high missing rates.

Acknowledgements

I would like to thank Dr Simon Tuke, Dr Nigel Bean, Dr Joshua Schmidt and Dr Christian Huber for all their help and guidance throughout this project. I also thank my friends from the maths department, Dylan Morris, Curtis Murray, Oliver Lountain and Will Jordan for their ongoing support throughout our studies.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.0.1 DNA

Deoxyribonucleic acid (DNA) is a biological molecule carrying the genetic instructions for the development, growth, maintenance and reproduction of all living things. DNA is a macromolecule (a large molecule) composed of smaller units known as nucleotides. Nucleotides comes in four different forms known as adenine, cytosine, guanine and thymine. We can think of DNA as a biological code consisting of four letters; A,C,T and G. All organisms store their biological information within this 4-letter code. An individual's entire genetic sequence is known as their genomes.

Biologists study the DNA of all kinds of organisms because DNA is such a fundamental molecule for life. Although DNA sequencing used to be slow and expensive, innovations in the 21^{st} century have enabled researchers to sequence entire genomes quickly and economically. As more genetic data is being generated, there is a growing demand for analytical tools to help researchers process and interpret the underlying genetic information. By understanding an organism's genome, researchers can acquire a deeper understanding the function of genomes.

1.0.2 Project Aim

For this project we will design a machine learning approach to scan through a sample of genomes from a given population and identify genetic regions which confer some biological advantage. In genetic parlance, such regions are said to be "under selection" because nature tends to "select" beneficial traits to be transmitted to future generations. (We will cover this in more detail when we discuss evolution in Chapter 2). Whilst selection can take on many forms, we will focus on positive selection which produces

selective sweeps. We will particularly focus on the "hard sweep" pattern because it is the most studied pattern in genetics thus far. Our method will enable researchers to identify regions of interest which they can study further using biochemical and bioinformatic analyses.

This project consists of two main parts.

1. We will apply a suite of machine learning classifiers to the problem of detecting hard sweeps using modern DNA samples which have been extracted from living individuals. After reviewing the performance of each method, we will suggest a standard protocol for using machine learning to detect hard sweeps. This will assist researchers in selecting an appropriate machine learning method and interpreting the results of their models.

2. We will extend our methods to detect hard sweeps in ancient DNA. Ancient DNA refers to genetic samples extracted from archaeological and natural history samples such as fossilised bones and mummified tissues. Ancient DNA is useful because it enables researchers to directly study the genomes of past populations. However, ancient DNA introduces a new set of technical challenges which we will explore and tackle in Chapter 6.

1.0.3 Thesis Road Map

Chapter 2 will introduce the biological background for this project. We will first describe DNA in more detail and then explain the theory of evolution via natural selection. This will lead into a discussion about patterns of selection found in DNA molecules. The chapter will conclude with a discussion about several summary statistics which have been designed to detect selective sweeps.

Chapter 3 will explain the mathematical theory behind machine learning. We will first cover the fundamental concepts of supervised and unsupervised learning, classification and regression problems, the bias variance trade-off and methods of assessing model performance (*e.g.* ROC/AUC). We will then discuss several commonly used methods in machine learning such as principal component analysis, random forests, discriminant analysis and MARS. We will also discuss the emerging area of "interpretable machine learning" which focuses on identifying important variables and unravelling how models make their predictions. We will cover useful tools such as partial dependence plots, independent conditional expectation (ICE) plots and feature importance ranking measure (FIRM) which can enable researchers to understand how their specific trained models are working.

Chapter 4 is a literature review on the current methods that have been developed for detecting hard sweeps. We will discuss the limitations of current approaches and articulate the knowledge gap which this project seeks to address.

In Chapter 5, we will review a suite of machine learning classifiers for detecting hard sweeps. We will compare models based on their accuracy and computational time. We will also investigate variables of importance to see which summary statistics are being used by each model. We will suggest a standard machine learning based workflow for researchers to use for detecting hard sweeps in their population of interest.

Chapter 6 will extend our methods to work with ancient DNA. We will first explain what is ancient DNA, how it is retrieved and why it is useful. We will cover the technical challenges of ancient DNA and suggest several strategies for tackling them. The study will be limited to a simple demographic model. After fitting our models, we will investigate whether we can still retrieve a clear sweep signal after DNA ageing. Tools from "interpretable machine learing" will be used to identify important variables that are useful for detecting hard sweeps in ancient DNA.

Chapter 7 which will review the key findings of this project and explain how our work fits into the broader field of evolutionary biology and population genetics. We will identify future directions for research which have the potential for extending our methods to work in more complicated demographic scenarios.

1.0.4 Accessing The Project

We wrote an R package called "popgen.tools" to process genome data and compute summary statistics. This has been published as an open source package on Github.

`https://github.com/deponent-verb/popgen.tools`

The pipeline for all our analysis is also available on Github.

`https://github.com/deponent-verb/popgen.analysis.pipeline`

Chapter 2

Biological Background

2.1 Genetics

2.1.1 DNA: The Molecular Basis for Inheritance

It is a common observation that offspring often resemble their parents but the biological mechanism for storing genetic information remained largely a mystery until the mid 20th century. In the early 20th century, many biologists considered protein to be a likely candidate for storing genetic information since it is a complex molecule found in all organisms. Although deoxyribonucleic acid (DNA) had already been discovered, it was thought to be too simple to store genetic information. Proteins are long molecules consisting of 20 naturally occurring amino acids, whilst DNA is composed of only 4 nucleotide bases. In 1952 Hershey and Chase conducted a series of experiments with bacteriophages (viruses) to finally confirm DNA to be the molecular basis for inheritance [2].

DNA is a long macro-molecule found in all organisms. DNA is composed of four nucleotide bases (adenine, cytosine, guanine, thymine) and the entire sequence of these bases make up a an individual's genome. Each individual organism has its own unique genome which can be passed down to its offspring. DNA is vital to life, being the chemical blueprint for all the biological processes necessary for the development, survival and reproduction of the organism. The most basic, functional units of DNA are called genes. Genes contain the recipes for making important biological molecules required for the life of the organism. Many genes encode proteins which carry out a vast array of functions such as providing structure for cells, breaking down larger molecules (catabolism), transporting molecules and speeding up biochemical reactions (enzymes). Genes can also encode RNA which can take a number of functions in the cell.

An organism's genome is typically arranged in larger structures known as chromosomes.Each chromosome has its distinct combination of sequences (e.g. genes). A unique combination of sequences on a chromosome is known as a haplotype. Different species vary in the number of chromosomes they possess. In prokaryotes (unicellular organisms without membrane bound organelles), the genome typically exists as a naked, circular chromosome stored in the nucleoid region [3]. The prokaryote class is further divided into bacteria and archae which inhabit a wide range of environments such hot springs, soils and animal intestines.

Eukaryotic organisms possess membrane bound organelles such as the mitochondria (responsible for cellular respiration) and chloroplasts (site for photosynthesis) [4, 5]. Multi-cellular organisms including mammals, fish, plants and fungi are all examples of eukaryotes. Rather than having a single, naked, circular chromosome, the genomic DNA of eukaryotes are wrapped around packaging proteins called histones [6]. This complex of proteins and DNA is called chromatin. Chromatin ensures that DNA strands do not become an unmanageable tangle and controls access to the genome. Organisms may also have multiple versions of the same chromosome in their cells and the number of versions is known as the ploidy. For example, human somatic cells (*i.e.* non-reproductive, body cells such as skin cells) are diploid because they have 2 versions of each chromosome, one paternal and one maternal. Human germ cells (sperm and egg cells) only have one version of each chromosome and thus are haploid.

2.1.2 The Origin of Genetic Variation

Section 2.1.1 introduced DNA, the important biological molecule that contains the heritable information for all organisms. We will now explore how DNA is changed across successive generations. The reproduction of any organism requires the replication of its own genetic material. The replication process must produce high fidelity copies in order to transmit important biological information for the next generation. Although organisms have various molecular mechanisms to ensure accurate DNA replication, errors are still possible. These errors alter the DNA sequence and are known as mutations. For example, human germ cells have an estimated average mutation rate of $\sim 1.2 \times 10^{-8}$ mutations/base pair/generation (bp^{-1} generation^{-1}) [7]. Considering that the human genome consists of over 3 billion base pairs, the expected number of mutations generated during one round of the production of germ cells (meiosis) is ~ 70 [8].

Humans are a case of a sexually reproducing organisms. This means that their offspring are produced by the fusion of two gametes (germ cells); namely a sperm cell from the father and an egg cell from the mother, each carrying its own DNA. Half of the offspring

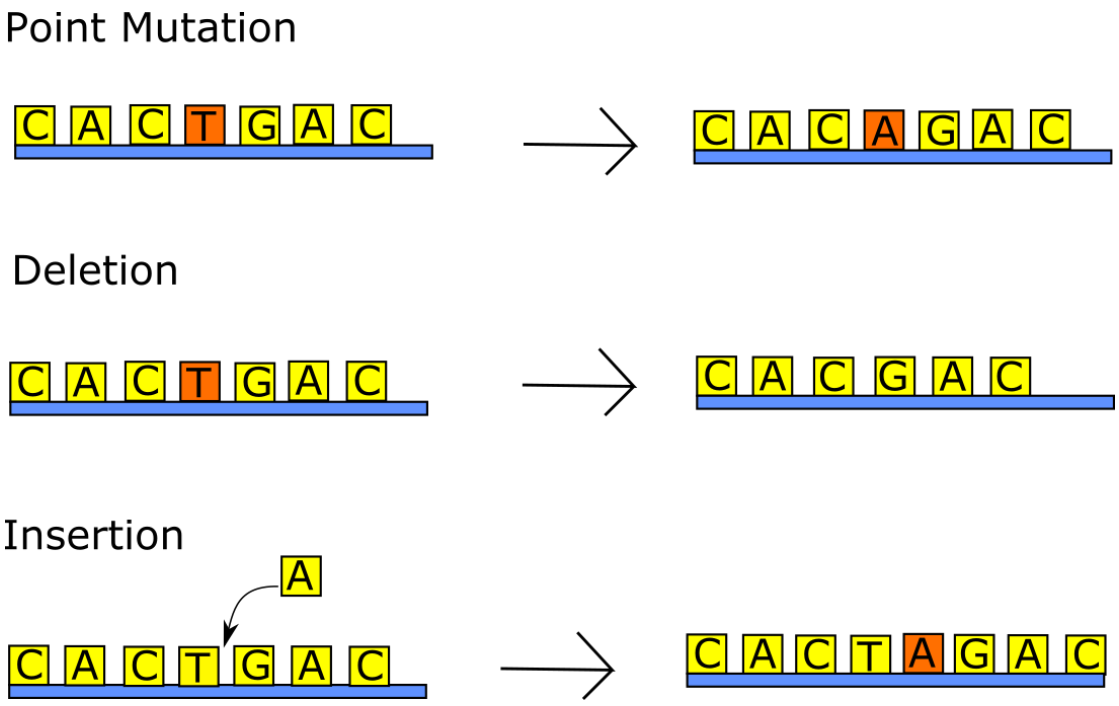

FIGURE 2.1: Diagram of three simple mutation processes.

genome comes from the paternal gamete and the other half from the maternal gamete. In the case of sexual organisms, since the offspring's genome is determined by the two gamete genomes, mutations are only inherited if they occur in the gametes. Mutations which occur in somatic cells (*e.g.* skin cells) may have consequences for the individual but cannot be passed onto offspring. Asexual organisms such as bacteria reproduce via cloning, whereby the offspring is largely a copy of the parent. For asexual organisms, any mutation that occurs during the DNA replication process will be transmitted to the offspring.

Fig. 2.1 is an illustration of three common mutational processes that are found across all organisms. A point mutation occurs when a single nucleotide is substituted by one of the other three. This variable site is known as a single nucleotide polymorphism (SNP). An insertion occurs when a nucleotide is added to the genome, whilst the removal of a nucleotide is called a deletion. Biologists often group insertions and deletions together, denoting them as "indel" mutations. There are more drastic mutations such as translocations, where two different chromosomes exchange genetic material with each other. Another example are inversions where an entire section of a chromosome is swapped back to front. Other types of mutations are known but it would suffice to understand that mutation is an ongoing source for new genetic diversity.

The effect of a mutation depends on its location and how it changed the original sequence. For example, mutations in key genes are likely to be consequential because they alter important gene products. Large scale changes such an inversion are also more likely to

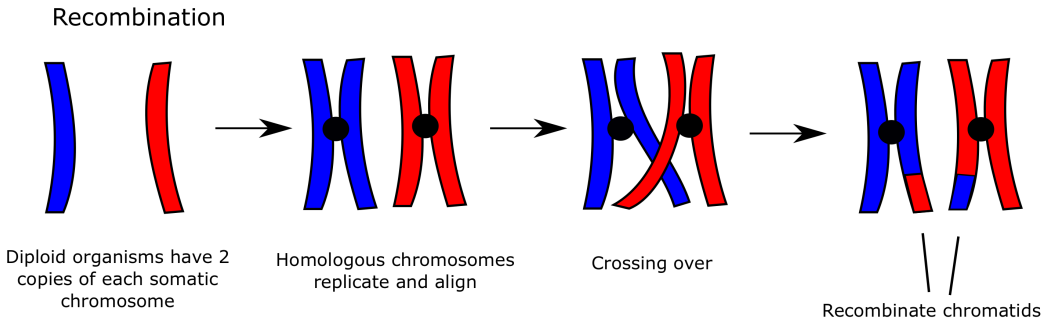

FIGURE 2.2: Illustration showing how chromosomes can recombine during meiosis. The blocks represent a pair of homologous chromosomes. Blue is the paternal chromosome and red is the maternal chromosome.

be consequential than smaller changes such as a single nucleotide substitution. Since mutation is a random, undirected process, mutations typically have either have a neutral or deleterious effect on the individual. The reasoning is that any functioning gene would be reasonable optimized and adapted for its biological task. Whilst there are a virtually infinite number of ways to alter this gene, only a small number of such changes would actually improve its function. Thus any random change to the gene would likely either reduce its function or keep it at the same level of effectiveness.

Nevertheless, the bulk of the genome appears to be non-functional for most species. For example, it is estimated for humans that only $\sim 8 - 15\%$ encode for important gene products (*e.g.* proteins) [9] [10]. Consequently, the genome generally robust to change and most mutations we observe are likely to be neutral and only a small proportion is either deleterious or beneficial.

2.1.3 Recombination

A pair of diploid organisms reproduce sexually by fusing their gametes (sperm + egg). Diploid individuals possess two copies of each chromosome, one paternal and one maternal. Each pair of somatic chromosomes are similar to each other and are said to be homologous. In order to reproduce, an individual must generate its own gametes via meiosis [11]. Figure 2.2 is an illustration of a key stage in meiosis. During meiosis, each chromosome duplicates. The copies of the paternal and maternal chromosomes are called sister chromatids. When homologous chromosomes align, a paternal and a maternal chromatid can make contact and exchange their sequences in a process called recombination. The site of recombination is random and can occur multiple times during

meiosis. Recombination creates novel recombinant chromatids which have a combination of paternal and maternal sequences. The four chromatids later separate into four haploid gametes. A pair of gametes merge during sex to produce a new diploid individual. Although recombination does not produce new sequences, it does shuffle sequences around to produce more genetically diverse offspring.

2.2 Genome sequencing

Recall that DNA is stored in discrete packets known as chromosomes (Section 2.1.1). When analysing modern samples, researchers use restriction enzymes to cut the chromosomes into smaller DNA fragments. The reason is that chromosomes are usually too large to be sequenced as single, intact units. A set of DNA fragments is known as a sequencing library [12]. A sequencing machine sequences the library to form a set of reads [13]. The reads refer to the DNA sequences that have been inferred from each of DNA fragments. Read lengths vary across different sequencing platform used. Popular platforms produced by Illumina and Life Technologies produce $\sim 200 - 300$ bp reads [14]. The Pacific Biosciences platform is well known for producing $10,000 - 15,000$ bp reads, although it tends to be more expensive [15].The technical workings of some popular DNA sequencing platforms can be found here [16]. Once all the reads have been sequenced, they must be merged together to reassemble the original genome [17].

One method is to use assemble the reads using a reference genome as a scaffold [18]. A reference genome is a DNA sequence assembled by scientists as a representative genome for an idealised individual within a species. Since the read lengths are much shorter than the reference genome (*e.g.* human reference genome is 3 billion bp [8]), it is a technical challenge to map millions of these tiny reads onto the much larger reference genome. Mapping algorithms have been developed which attempt to place the reads onto the reference in order to maximise sequence similarity whilst also allowing for small variations such as SNPs, insertions and deletions. Provided there are sufficient high quality reads[1] which span the whole genome, the entire genome of the sample can be deduced with good confidence. The end product is an inferred genome of the sample which is similar but not necessarily identical to the reference genome. Technical details about various DNA mapping algorithms can be found here [17, 19, 20, 21].

[1] A high quality read would be one where the sequence has been inferred with high confidence. Long reads are also beneficial because are generally easier to map to a unique location on the reference genome.

2.3 Evolution

2.3.1 Darwin's Theory of Natural Selection

Evolution is the change of heritable traits in biological populations over time. One of the key breakthroughs in understanding evolution is Darwin's theory of evolution via natural selection. Darwin's theory states that in any given population, there is some variation in traits. These traits include differences in morphology (*e.g.* beak shape), physiology (*e.g.* how a sugar molecule is broken down) and behavior (*e.g.* the migration path of a bird). Traits differ in their impact on an individual's ability to survive and produce viable offspring. Individuals that are well adapted to the environment are more likely to reproduce and transmit their heritable characteristics to the next generation. This selection of adaptive, heritable characteristics is known as natural selection.

2.3.2 Case Study: Peppered Moth Evolution

A classic example for teaching natural selection is the case study of peppered moth evolution in Great Britain [22]. The peppered moth (*Biston betularia*) is an insect found in the forests of Britain. It was originally observed to be whitish grey in color with dark speckles on its wings. During the 19th century industrial revolution, Britain drastically increased coal burning in order to fuel its booming industry. This produced air pollution with the surrounding regions being covered in black soot. Around this time, a new black variant of the peppered moth emerged which were particularly common in areas with high pollution.

Biologists theorized that the moth population was evolving in response to industrial pollution and moth color was the trait under selection. In forests with clean air, the trees were usually covered in lichen and were light in color. Consequently, the whitish grey moths could camouflage whilst resting on the bark, thereby avoiding predators such as birds. In this environment, any black moths would be easily identified and hence they were rare. However, forests adjacent to large industrial centers were smokey and covered in soot. This killed off much of the lichen and darkened the trees. In polluted areas, the black moths were able to camouflage and hence had an evolutionary advantage. Hence, evolution via natural selection could explain why moth populations became black in response to air pollution.

Kettlewell confirmed this theory through a series of mark-release-recapture experiments in the 1950s [23]. To establish that moth color could camouflage against predators, Kettlewell released a combination of black and light moths into an aviary in Cambridge.

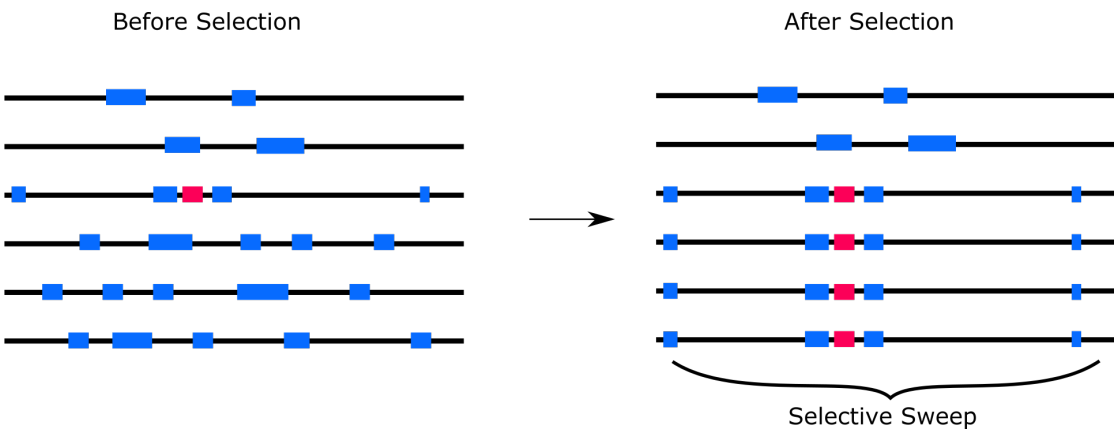

FIGURE 2.3: Illustration of a selective sweep. Each line is a DNA sequence from one sampled chromosome. Black lines are consensus sequences. Blue boxes represent neutral mutations and the red box is a novel beneficial mutations

He observed that the birds would preferentially hunt moths which were poorly camouflaged against the background color. In order to demonstrate the effect of moth color in the wild, Kettlewell captured and marked a group of light and black peppered moths. The experiment involved releasing a batch of moths in an area and recapturing them after several days. By comparing the proportion of light and black moths in the tagged recaptured groups, the survival rate of each moth color could be estimated. Kettlewell found that light moths had a higher survival rate in clean regions, whilst black moths had a higher survival rate in polluted areas. Peppered moth populations have remained a subject of interest in evolutionary biology and the darkening of a species in response to pollution is called "industrial melanism."

2.3.3 Selective Sweeps: Selection on a Molecular Level

Section 2.3.2 provided an example of how evolution via natural selection, could be observed on a phenotype level. Phenotype refers to the observable characteristics of organisms (*e.g.* color in the moth example). This section considers how selection happens on a molecular level. Fig. 2.3 is an illustration of how selection may act on a specific, fixed position on a chromosome. Fixed positions on a chromosome are also called genetic loci. In this figure, each line is a genetic locus on one chromosome which belongs to one individual organism. Since there are six lines, we are examining six chromosomes at a particular region of interest. The black parts refer to consensus sequences where all sampled individuals from the population have identical sequences. The blue boxes represent areas with neutral mutations; mutations which offer neither a selective advantage nor disadvantage. The red box is a novel mutation that confers a beneficial trait. Relating back to 2.3.2, we could consider these to be moth chromosomes from industrial England and the new mutation confers the black color. Each unique version of a genetic

locus is called a haplotype. All six lines in the diagram have some unique mutations so this example has six haplotypes in total.

Since the new beneficial mutation confers an evolutionary advantage, it will increase in frequency over successive generations. In this figure, the red mutation increased from $\frac{1}{6}$ (left panel) to $\frac{4}{6}$ (right panel). Notice that the two neutral mutations on either side of the selected mutation have likewise increased in frequency, despite being neutral. The reason is that these neutral mutations were in close proximity of the beneficial red mutation. As the selected mutation became more prevalent in the population, it also brought along its surrounding neighborhood. This process whereby neutral mutations increase in frequency by being nearby a selected mutation is known as "genetic hitch-hiking" [24].

The closer a locus is to a selected mutation, the more likely genetic hitch-hiking would occur. Section 2.1.3 explained that recombination generates new haplotypes by swapping sequences between different homologous chromosomes. Thus in order to separate two loci on one haplotype, there must be at least one recombination event between the two loci. Assuming that the recombination rate is approximately constant across a chromosome, the closer two loci are, the less likely a recombination event would occur between them and the more likely hitch-hiking would occur. This is shown on the right panel of Figure 2.3 where the two neutral mutations further away from the selected mutation did not increase in frequency.

The curly brackets on the right panel indicate a region of reduced genetic variation. Prior to selection, genetic variation was high with every haplotype looking quite different. This is because the neutral theory provides the expectation of standing levels of neutral genetic variation [25, Chapter 9]. Due to selection, the neutral haplotypes were replaced by the haplotype with the selected mutation. The reduction of genetic variation near a selected mutation is known as a "selective sweep" [26]. Selective sweeps are biologically relevant because they reflect how a population has adapted to its environment over time. Investigating these regions can enable researchers to unravel the underlying mechanisms of evolution and genetics. Consequently, this project focuses on finding selective sweeps by detecting the patterns of genetic variation caused by selection.

2.3.4 Types of Selective Sweeps

Section 2.3.3 established that beneficial mutations tend to spread across populations and this process leaves behind patterns of genetic variation. We will now focus on two key patterns produced by selective sweeps, namely hard sweeps and soft sweeps. Other

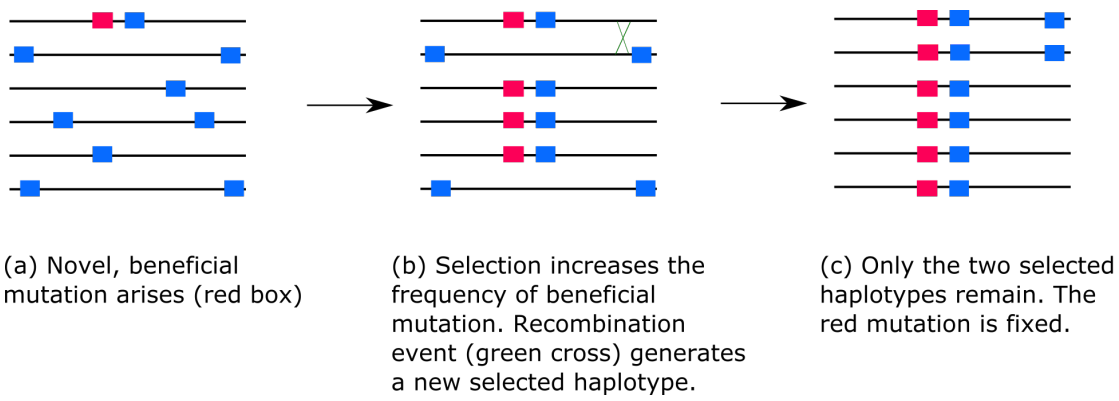

FIGURE 2.4: Illustration of a hard sweep. Each line is a DNA sequence from one sampled chromosome. Black lines are consensus sequences. Colored boxes are mutations. Blue are neutral and red is under selection. The green cross indicates a recombination event.

sweep patterns such as partial sweeps are known but they are beyond the scope of this research. Our focus is to develop methods for detecting hard and soft selective sweeps.

2.3.4.1 Hard Sweeps

Figure 2.4 is an illustration of the pattern produced in a hard selective sweep. This process is expected to occur when beneficial mutations (*i.e.* selected mutations) are rare. When beneficial mutations do occur by chance, they quickly increase in frequency and "sweep" across the population [24]. Figure 2.4 (a) depicts a population of six individuals, each with their own haplotype. A new beneficial mutation (red box) is formed on one haplotype and hence this haplotype is under selection. Figure 2.4 (b) shows that the selected haplotype has increased in frequency over successive generations. Notice that the blue neutral mutation adjacent to the red selected mutation, has likewise increased in frequency due to genetic hitch-hiking (Section 2.3.3). Eventually every individual =possesses the selected mutation and the linked neutral variant and thus both mutations are considered to be "fixed." However, recombination can result in the fixation of linked neutral variants to be incomplete (Section 2.1.3). The green cross in (b) indicates a recombination event which brings another, more distant neutral mutation onto a sequence with the selected mutation. Note that the closer a mutation is to the selected variant, the less likely this will happen. This generates an additional haplotype under selection. Figure 2.4 (c) shows that the population is eventually replaced by individuals who have either one of the two selected haplotypes. Since both selected haplotypes have the same selected mutation, neither haplotype completely dominates the population.

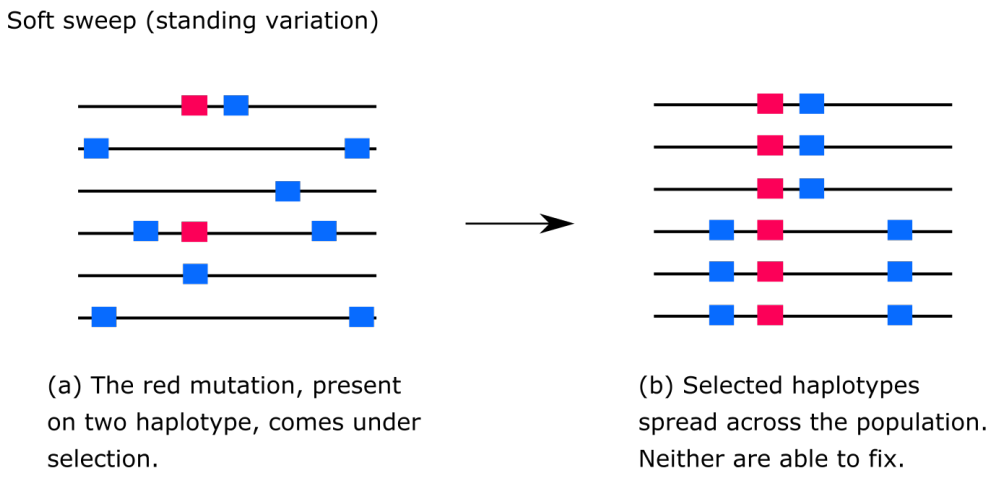

FIGURE 2.5: Illustration of a soft sweep caused by standing variation. Each line is a DNA sequence from one sampled chromosome. Black lines are consensus sequences. Blue boxes are neutral mutations. The red box was previously neutral but late came under selection.

A comparison between Figure 2.4 (a) and (c) shows the reduction of diversity caused by a hard sweep. Recall that the population started with six haplotypes with each pair having multiple differences between them. By the end of the sweep, only two haplotypes remain and there is only one difference between them, namely a neutral mutation on the right end of the genetic sequence. The process of the hard sweep has purged much of the pre-existing variation within the neighborhood of the selected mutation. Methods of detecting hard sweeps involve identifying regions along the genome with this drastic drop in genetic variation.

2.3.4.2 Soft Sweeps

A more subtle pattern of selection is the soft sweep which can be produced in two slightly different scenarios; namely via standing variation and recurrent mutation [27]. Standing variation refers to existing neutral mutations in a genetic locus. Since these mutations are not newly generated, they had the opportunity to recombine with different backgrounds to form multiple haplotypes. Figure 2.5 is an illustration of a soft sweep. A soft sweep can occur when a previously neutral or mildly deleterious variant (*i.e.* segregating site), comes under selection. This is typically due to environmental changes which confer selective advantage to previously neutral traits. The selected mutation increases in frequency and may eventually become fixed. Potentially no single haplotype dominates the locus in a soft sweep because there are multiple haplotypes which possess the selected mutation. However, if the frequencies of the selected variant are low prior

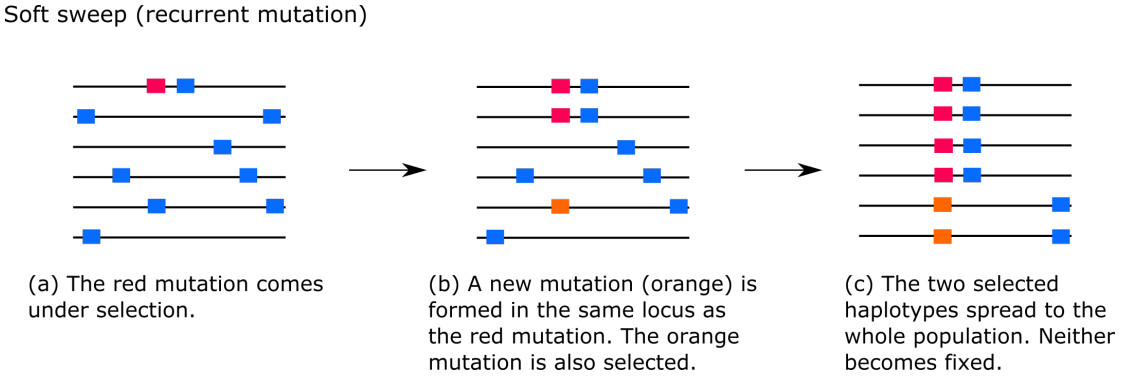

FIGURE 2.6: Illustration of a soft sweep caused by recurrent mutation. Each line is a DNA sequence from one sampled chromosome. Black lines are consensus sequences. Blue boxes are neutral mutations. The red and orange boxes are mutations that come under selection.

to selection, then only a single haplotype may be fixed due to chance alone. This would produce a pattern similar to hard sweeps [28, 29].

Recurrent mutation occurs when multiple, different mutations happen within a single site. Figure 2.6 is an illustration of a soft sweep caused by recurrent mutation. Recurrent mutations produce a soft sweep when they are equally under selection. These mutations increase in frequency simultaneously until their combined frequency in the population reaches one. None of these mutations can fix individually because they have to compete with the other recurrent mutations. In conclusion, both hard and soft sweeps involve selected variants becoming more prominent in the population. The key difference is that soft sweeps leave behind more genetic variation within the locus and thus are generally more difficult to detect [30].

2.4 Representing Population Genetic Data

We have described how evolution can influence the genetic sequence on the molecular level Section 2.3.3. We will now explore ways of representing genome data into suitable forms for data analysis. Recall from Section 2.1.1 that DNA across all organisms is composed of 4 nucleotide bases; namely adenine, thymine, cytosine and guanine. Thus an individual chromosome may be represented as a long character string of 4 letters; A, T, C and G. Since natural selection is a process that occurs within a population, we also need to sample multiple chromosomes from a population for the purposes of detecting selective sweeps. Genome data sequenced from multiple individuals in a population is known as population genetic data. Generally speaking, the more individuals that are present in our population genetic data set, the greater power we have for sweep detection.

Once we have sampled our population of interest, we can align the sequences based on their similarities. This can be represented as a string matrix where each column is a nucleotide base and each row is a sampled chromosome. Recall Section 2.1.1 that genomes are long thereby making the matrix very wide. For example, chromosome Y one of the smaller chromosomes, has approximately 58 million base pairs and represents around 2% of the human genome [31]. High dimensional data is challenging to analyse because it requires immense computational resources. Thus we need a way of simplifying population genetic data without losing the underlying patterns of selection which we seek to detect.

2.4.1 SNP Data

As suggested by Figures 2.3 to 2.6, the majority of population genetic data consists of consensus sequences (*i.e.* sites are non-polymorphic). Thus when comparing the sampled genomes from a population, there are long stretches of DNA have the same base sequence. For example, the 1000 genomes project found that a typical human's genome only differs from the human reference genome at $\sim 4.1 - 5$ million sites [32]. This upper figure roughly corresponds to only $\sim 0.6\%$ of the total number of base pairs. One may reason that since consensus sequences are regions of low diversity, they must indicate selective sweeps. However, this would be naive as there are genetic structures that are common across individuals in a population. For example, retrotransposons are repetitive DNA sequences commonly found across the genome [33]. Retrotransposons are common because they can "copy and paste" themselves into new genomic locations. This mechanism is not related to selection.

The expected number of segregating sites can be computed using the Watterson estimator [34]. Over a given sampled region in a population with biological mutation rates, most sites are expected to be non-polymorphic. In short, consensus sequences are generally uninformative in regards to detecting selection. This leaves us to focus on the differences (*i.e.* mutations) between individual samples within a population. The different variants found at any particular base pair are known as alleles.

Recall from 2.1.2 that a common form of genetic variation are single base pair differences known as SNPs. SNP data condenses population genetic data sets by only including the SNPs identified within a sampled population. SNP data has a number of advantages [35]. SNPs are abundant and distributed widely across the genome. Hence, SNP data can capture interesting selection patterns all over the genome. Due to the technicalities of DNA sequencing methods, SNP data is easy to collect and cheap to produce. The more samples that can be obtained from a population of interest, the more SNPs can

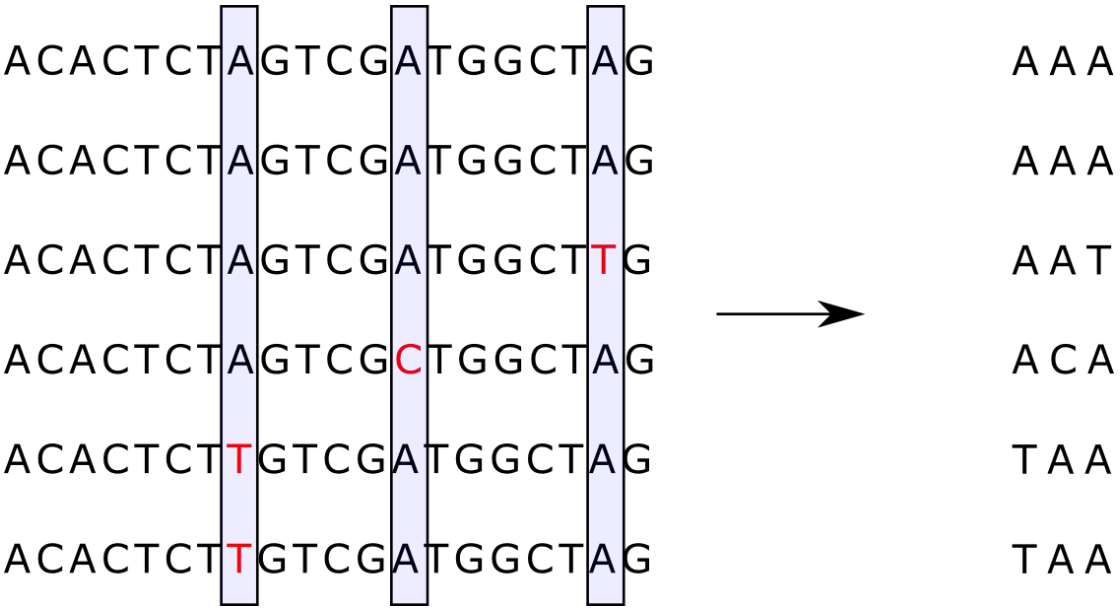

FIGURE 2.7: Illustration of how genome data can be converted into lower dimension SNP data. Here are 6 sampled chromosomes of length 20 bases. There are 3 SNPs in the population. By keeping only the SNP information, we can reduce the dimensions of the dataset by 85%.

be identified, thereby increasing the power for sweep detection. Most importantly, SNP data reduces the dimensions of population genetic data by filtering out uninformative consensus sequences which make up the majority of the genome. Figure 2.7 provides an illustration of this process.

SNP data can be presented as a character matrix G and a numeric vector $\mathbf{p}$. Matrix G contains the genomes, where the columns are the SNPs and each row is a sampled individual. The entry $G_{ij} \in (A, T, C, G)$ would be a character indicating the nucleotide base of the i^{th} sample at the j^{th} SNP position. The vector p would contain the positions of each SNP position along the genome, where p_i represents the base position of the i^{th} column in G. Suppose the third SNP was located on the 500^{th} base on the genome. p_3 will be 500.

2.4.2 Infinite Sites Model

Section 2.4.1 explained how population genetic data can be condensed by only looking at the SNPs identified within the population of interest. Each sample can be represented as a string containing its nucleotide base at every SNP position. We now consider the infinite sites model; a mutational model that is used to represent SNP data. The model devised by Kimura in 1969 [36] makes two key assumptions. (1) For any given population of individuals, there are an infinite number of sites (*i.e.*) where a new mutation could occur. (2) Every new mutation forms at a new site, with no previous segregating

mutations. This means that there are a maximum of two alleles for any given site along the genome. Using this model, Kimura derived a set of useful mathematical results such as average number of heterozygous sites per individual over some k generations and the frequency distribution of mutants in a finite population. These results were later utilized for building population genetic programs to simulate genome data [37].

Assumption 1 is valid as long as the number of sites L is large relative to the mutation rate μ events per base per generation. This means the majority of sites will not be segregating sites (*i.e.* there are no variants. Recall our discussion of the Watterson estimator in Section 2.4.1.). Given that any mutant in a finite population will either fix or go extinct in finite time, segregating sites will also be removed from the population in finite time as well. When $L \gg \mu$, the probability of a multiple mutations occurring at the exact same site approaches 0. These assumptions are reasonable for many problems in population genetics.

Applying the ISM to SNP data can condense population genetic data even further. Recall from 2.4.1 that SNP data can be represented as a character matrix G and a numeric vector of positions p. Using assumption 2, we can modify G into a numeric matrix of 0's and 1's which represent the two alleles available for any segregating site. The convention in population genetics is for 0 to indicate the older allele (ancestral) and 1 to indicate the novel allele (derived). There are a number of ways for determining which allele should be considered ancestral. A naive method would be to assume that the more common allele is the older one. Although this disregards the possibility that the more common allele may be a novel mutation approaching fixation, this assumption may be reasonable for certain population genetic problems. Neutral theory suggests that allele age and frequency are proportional, conditioned on the presence of the allele.

A more sophisticated approach would be to use an outgroup, a population of distantly related organisms to the population of interest [25, Chapter 3]. For example, suppose we are studying a particular locus in a human gene that has alleles C and T. This is known as a genetic polymorphism because there are multiple alleles at this position. To determine the derived allele we can look at the same genetic locus of closely related species such as gorillas, chimpanzees and orangutans. If all 3 species have the C variant, then it is highly likely that the ancestral allele is C. This is because it is more likely for a single T mutation to have occurred in the human lineage than it is for three lineages to develop the same mutation independently. A full exploration of how to assign the ancestral allele is not needed in this section. The key point for this section is that for a sampled population of n individuals with k SNPs, the corresponding genome data can be represented by a numeric matrix G and a numeric vector $\mathbf{p}$ where,

$$G_{n\times k}:\ G_{ij} \in \{0,1\}\ \forall\, i \in \{1,2,\ldots,n\},\ j \in \{1,2,\ldots,k\} \tag{2.1}$$

$$p_{k\times 1} : p_i \in \{0,1\}\ \forall\, i \in \{1,2,\ldots,k\} \tag{2.2}$$

We call G as the genome matrix where G_{ij} gives the allele of sample i at SNP j. Vector $\mathbf{p}$ is vector of SNP positions where p_i is the base position of the i^{th} SNP in G. As in Section 2.4.1, if the third column in G corresponded to the 500^{th} base on the genome then p_3 will be 500.

2.5 Summary Statistics

Section 2.4 explained how sampled genomes from a population could be condensed into a binary genome matrix using the infinite sites model. The rows represent the sampled chromosomes and the columns represent the SNPs found in the sampled population. Given that even a 1Mb region could have thousands of SNPs, genome matrices are still too highly dimensional to be used as the raw input for most analytical tools. To address this technical problem, population geneticists have designed a variety of summary statistics for detecting selective sweeps. A summary statistic is defined as a scalar, real-valued function of the data. In this context, suppose there is a sample of n individuals with k SNPs, represented by a binary genome matrix $G_{n\times k}$. A summary statistic f is a function $f : G_{n\times k} \longrightarrow X,\ X \subseteq \mathbb{R}$. A useful summary statistic is one which captures important and relevant patterns in the data. The summary statistics designed for sweep detection come from three main classes; namely statistics based on the site frequency spectrum (SFS), haplotype frequencies and linkage disequilibrium. We will now explore these three classes, providing examples of commonly used statistics of each class.

2.5.1 SFS Based Statistics

2.5.1.1 Effective Population Size

Before we explore the site frequency spectrum, we must first discuss the concept of an effective population size. The effective population size, typically denoted as N_e in population genetics literature, is defined as the number of individuals in an idealised population that would produce the same population genetic quantities as the actual population of interest [25, Chapter 3]. An idealised population is a theoretical model of a population where a number of simplifications are made for the sake of mathematical

convenience. This would typically be understood to be a Wright-Fisher model (**??**) or another similar model which assumes a constant population size, random mating, a single mating type, neutral evolution and no overlapping generations. For example, suppose we are studying a population of F frogs in a pond and we are interested in the average number of pairwise differences. Suppose Ne individuals in Wright-Fisher population will produce the same average number of pairwise differences as the F frogs. We would then be able to say that the effective population of the frogs is Ne with respect to the average number of pairwise differences. The effective population size Ne can often vary substantially from the census population size F (*e.g.* the actual number of frogs). This is because real populations usually behave differently to the idealised populations used in theoretical population genetics. Factors that affect the effective population size include the census population size, the male:female ratio for sexually reproducing organisms and the variance in offspring produced by each individual. A full theoretical discussion on the factors that impact the effective population size is beyond the scope of this chapter. It would suffice to understand that the effective population size is an important parameter in population genetics which affects the values of various summary statistics.

2.5.1.2 Site Frequency Spectrum

Section 2.4.1 introduced the idea of alleles which are the different genetic variants found at any given locus. The site frequency spectrum (SFS), *a.k.a* allele frequency spectrum in some literature, is defined as the distribution of allele frequencies in a given set of genetic loci. In the context of SNP data, the genetic loci consists of all the SNPs identified in the sampled population and the alleles can either be ancestral or derived. For a sampled population of n individuals, the SFS can be represented by a vector $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ where x_i is the number of derived alleles which appear exactly i times in the sampled population. In this way, the SFS may be considered as a histogram of derived allele frequencies. It can be shown that under a neutral model (*i.e.* no selection) with constant population size, the expected SFS is given by

$$E[x_i] = \frac{\theta}{i}, 1 \leq i \leq n-1, \; \theta = 2Ne \times \mu \tag{2.3}$$

where N_e is the population size and μ is the expected number of mutations per base per generation [38, Chapter 4] . This results means that the bulk of the distribution lies in the low frequency alleles. Mutations are rare and neutral mutations are unlikely to spread across a large population by chance. An intuitive interpretation of (2.3) is that as the mutation rate increases, the more novel variants are being produced thereby

increasing the number of derived alleles. Similarly, the larger the effective population size, the more diversity there is in the population which results in a greater abundance of derived alleles.

Equation (2.3) shows that under a neutral model, the expected site frequency spectrum is a right-skewed. Since selection acts by changing allele frequencies, it also distorts the SFS from the expected neutral distribution [25, Chapter 9]. For example, suppose some novel beneficial mutations are generated within a particular genetic locus. These mutations will be under positive selection and tend to appear at higher frequencies than neutral variants. This will skew the SFS in favor of high frequency variants compared to the expected distribution for neutral alleles. Similarly, a set of deleterious mutations under negative selection will appear at lower frequencies compared to neutral alleles, thereby shifting the SFS in favor of low frequency variants. However, as we discussed in 2.3.3, when selection changes the frequency of one allele, the surrounding alleles will also be affected. This effect is particularly pronounced for alleles that are very close to the selected allele in base pairs. In a selective sweep, the selected allele and those linked to it will increase in frequency, thereby forming an excess of high frequency alleles. The alleles not linked to the selected mutation will decrease in frequency, leading to a excess in low frequency alleles. Thus selective sweeps produce an SFS which has an excess of high and low frequency alleles but far fewer alleles of intermediate frequency. By comparing the shape of observed SFS with that of the expected neutral SFS in (2.3), one could infer whether a selective sweep has occurred in a particular genetic region. SFS based summary statistics are designed to capture the shape of the SFS in a quantitative manner for the purposes of inferring selection. The general idea is that a strong deviation from the expected neutral distribution shown in (2.3), may be considered a evidence of a selective sweep. We will now discuss two of the main SFS based summary statistics for inferring selection; namely Tajima's D and Fay and Wu's H.

2.5.1.3 Tajima's D

Section 2.5.1.2 explained how selection affects the SFS and introduced the idea of detecting selection by comparing the observed site frequency spectrum with the expected distribution under the neutral model. Tajima's D is a test statistic designed to detect deviations in allele frequencies from the neutral model and hence infer selection [25, Chapter 9]. Since Tajima's D is composed of two key components, namely θ_T and θ_w so we will explain how to compute them first. Section 2.4.2 explained how to represent population genetic data with a binary genome matrix and a corresponding position vector Section 2.4.2.

Suppose we are investigating whether a particular genetic locus of B bases in length, has been the site of a recent selective sweep. (A typical length used by researches in 1Mb or 10^6 bases). We sampled n individuals and find k SNPs, giving us a genome matrix $G_{n\times k}$.

θ_T (also called π or Tajima's estimator in some population genetics literature) is defined as,

$$\theta_T \coloneqq \frac{\sum_{i=1}^{j-1} d_{ij}}{n(n-1)/2} \tag{2.4}$$

where d_{ij} is the number of pairwise differences between samples i and j, and n is the number of chromosomes sampled from the population. θ_T is essentially the total number of pairwise differences within the sampled population, divided by the total number of pairs.

θ_w (also called the Waterson's estimator) is defined as,

$$\theta_w \coloneqq \frac{k}{\sum_{i=1}^{n-1} \frac{1}{i}} \tag{2.5}$$

where k is the number of segregating sites and n is the number of samples. Under the infinite sites model, it can be shown that for a neutral model with constant effective population size (standard neutral model),

$$E[\theta_T] = E[\theta_w] = 4Ne \times \mu$$

where Ne is the effective population size and μ is the mutation rate per base per generation [25, Chapter 3]. This result means that if there is no selection in the region and the assumptions of the infinite sites model are valid, then the computed values of θ_T and θ_w are expected to be the same. Of course, even if the neutral model is true for some data, the observed values of θ_T and θ_w could differ by chance alone. However, if the two estimates are substantially different from each other, then this is evidence that the standard neutral is not correct. One of the key potential causes for the deviation is selection which brings us to the Tajima's D test statistic.

$$\text{Tajima's D} \coloneqq \frac{\theta_T - \theta_w}{\sqrt{var(\theta_T - \theta_w)}} \tag{2.6}$$

The denominator is the positive square root of the variance in $\theta_T - \theta w$. The analytical expression for the variance term, along with the full derivation, can be found in [39].

If the region of interest has been evolving under the standard neutral model, its computed value of Tajima's D should be approximately 0. A high negative value of Tajima's D indicates the region has a low number of observed segregating sites relative to the expected number under the standard neutral model. This could be caused by a selective sweep which reduces diversity around a selected mutation. A quick reduction in the population size (population bottleneck) can produce a similar effect by removing individuals from the population and thereby eliminating their variants from the gene pool. A high positive value of Tajima's D indicate an excess of observed segregating sites relative to the expected number. This could be caused by a declining effective population size or balancing selection. In short, balancing selection is a special selective process which promotes multiple alleles being maintained in a population. P-values for Tajima's D are typically hard to compute directly as its underlying distribution is tricky to derive analytically. A standard approach involves simulating population genetic data to give an approximation of the p-value. A general rule is that under no recombination, the standard neutral model can be rejected at 5% significance if |Tajima's D| ≥ 1.8 [25, Chapter 9].

2.5.1.4 Fay and Wu's H

Section 2.5.1.2 explained that under the standard neutral model, we expect very few high frequency derived alleles and selective sweeps tend to make them more common. By comparing the observed SFS with the expected neutral distribution, we can infer whether a region is evolving under the standard neutral model. Fay and Wu's H is another SFS-based statistic which measures the excess of high frequency variants in comparison to intermediate frequency variants [40]. It is particularly useful for regions with low recombination rates and little genetic variation. The definition of Fay and Wu's H is as follows.

Let x_i be the number of segregating sites where the derived allele occurs i times in the sampled population. Let,

$$\theta_H \coloneqq \frac{\sum_{i=1}^{n-1} i^2 x_i}{\binom{n}{2}}$$

θ_H is also an estimate of θ.

$$\text{Fay and Wu's H} \coloneqq \theta_H - \theta_T$$

where θ_T is as defined in Equation (2.4) and n is the number of samples. There is also a standardised version of H which divides by the theoretical standard deviation under the neutral model. An H value that is approximately 0, indicates no evidence of deviation from the standard neutral model. A statistically significant negative value of H indicates an excess of high frequency derived alleles which may be a sign of a selective sweep. Statistically significant positive H values suggest negative selection, whereby new variants are being purged from the population. The i^2 term within the sum means that the largest contributors to the size θ_H come from high frequency derived alleles. This is in contrast to θ_T which gives more weight to derived alleles of intermediate frequency. This makes Fay and Wu's H more sensitive to Tajima's D in regards to detecting the excess of high frequency derived alleles which are often caused by selective sweeps. Using both statistics together can enhance our ability to detect selective sweeps

2.5.2 Haplotype statistics

Section 2.3.3 introduced the concept of a haplotype, unique sequences of DNA found at a particular locus of interest. When a region of DNA is evolving neutrally, we would expect high levels of diversity where multiple haplotypes would be present. No single haplotype dominates the population as each haplotype is equally likely to be transmitted to successive generations. Under a selective sweep, a selected haplotype would rapidly increase in frequency whilst all other haplotypes are purged from the population. Haplotype statistics are designed capture this effect by using the haplotype proportions found in the sampled population. Note that what mathematicians call "proportions" are often called "frequencies" by population geneticists.

For some genetic locus of interest, suppose we sampled n individuals and identified k SNPs. This gives us a genome matrix $G_{n\times k}$. To compute the haplotype statistics from G, we must first identify all the unique rows of G. Recall from Section 2.4.2 that the rows of a genome matrix represent individual genomes, so the unique rows of G correspond to the unique haplotypes found in the sampled population. Let p_i be the proportion of the i^{th} most common row in G. The haplotype statistics h1 is defined as

$$\text{h1} \coloneqq \sum_{i=1}^{c} p_i^2 \tag{2.7}$$

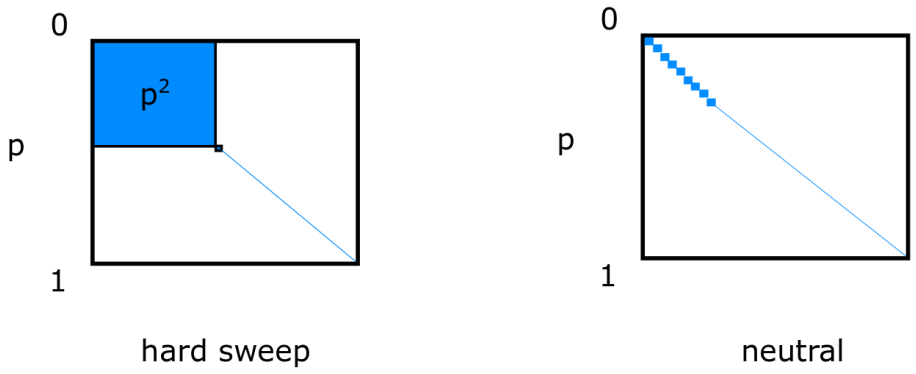

FIGURE 2.8: A visual representation of how the haplotype statistic h1 can be used to distinguish hard sweeps from neutral simulations. The edges of the black squares represent the haplotype proportions between 0 and 1. The blue shaded area represent the value of the h1 statistic.

where c is the number of haplotypes in the sampled population [41]. The h1 statistic is essentially the sum of the squared proportions of all the haplotypes. The h2 statistic is similar to h1 but removes the most common haplotype.

$$\text{h2} := \sum_{i=2}^{c} p_i^2 = h1 - p_1^2 \tag{2.8}$$

The h12 statistic combines the proportion of the two most common haplotypes into one.

$$\text{h12} := (p_1 + p_2)^2 + \sum_{i=3}^{c} p_i^2 = h1 + 2p_1p_2 \tag{2.9}$$

The h123 statistic combines the proportion of the three most common haplotypes into one.

$$\text{h12} := (p_1 + p_2 + p_3)^2 + \sum_{i=4}^{c} p_i^2 = h12 + 2p_1p_3 + 2p_2p_3 \tag{2.10}$$

Figure 2.8 shows how h1 can be used to detect hard sweeps from neutrally evolving regions. Recall from Figure 2.4 that during a hard sweep, a single haplotype with the selected mutation rapidly increases in frequency within the population. Since a single haplotype is highly common in the population, p_1^2 is large, thereby inflating the value of the h1 statistic. However, in a neutral locus, we expect there to be many haplotypes of low frequency, since none are more likely to be inherited than the others.This keeps the proportions for each haplotype low, thereby giving relatively smaller values of h1

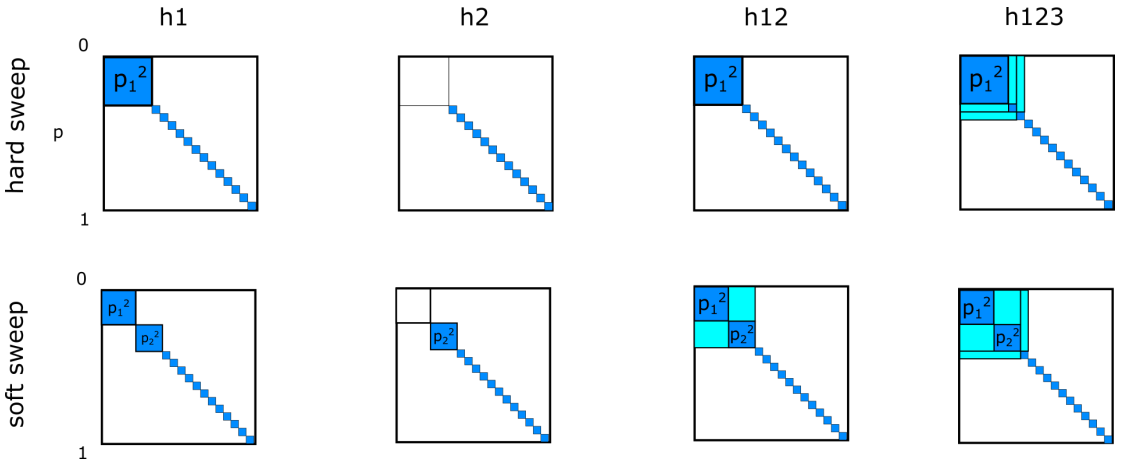

FIGURE 2.9: A visual representation of how the haplotype statistics can distinguish hard and soft sweeps. The edges of the black squares represent the haplotype proportions between 0 and 1. The blue shaded area represent the value of each statistic.

compared to hard sweeps. In this way, large values of h1 in a genetic locus is a sign that it may contain a hard sweep.

Figure 2.9 shows how haplotype statistics are expected to vary between hard and soft selective sweeps. Recall from Figure 2.5 and Figure 2.6 that soft sweeps involve a few selected haplotypes dominating the population. This means that the proportions of the top few haplotypes would collectively be quite high whilst the neutral haplotypes would have relatively low proportions. This is different to a hard sweep where the top haplotype has the highest proportion by far. Generally speaking, neutral simulations should have low values across all four haplotype statistics, whilst sweeps would produce higher values in some of these statistics, depending on whether it is a hard or soft sweep.

Both hard and soft sweeps should have higher values of h1 relative to neutral loci due to the increase proportions of selected haplotypes. This effect is more pronounced in hard sweeps, due to the dominance of the single selected haplotype. Since the h2 statistic excludes the proportion of the most common haplotype, hard sweeps and neutral loci would both have low values of h2. However, soft sweeps would have relatively higher values of h2 since there are several haplotypes which are common in the population. The ratio $\frac{h2}{h1}$ should increase monotonically as the sweep becomes softer. This ratio is useful for distinguishing hard and soft sweeps. The h12 and h123 statistics combine the proportions of the two and three most common haplotypes into a single proportion. Both statistics should have similar power for detecting both hard and soft sweeps.

To conclude, haplotype statistics have been designed to identify selective sweeps using the proportions of haplotypes found in the sampled population. The h1 statistic is particularly useful for detecting hard sweeps where there is a single haplotype which dominates the population. The h2 statistic is useful for detecting soft sweeps since it

excludes the proportion of the most frequent haplotype. The ratio $\frac{h2}{h1}$ is particularly useful for distinguishing hard and soft sweeps. The h12 and h123 statistics are useful for detecting hard and soft sweeps. Using the combination of these statistics enables population geneticists to not only detect selection but to also differentiate between hard and soft sweeps.

2.5.3 Linkage Disequilibrium Statistics

Linkage disequilibrium is a measure of the association between two different loci within a particular population of interest. Since selective sweeps involve a small number of selected haplotypes dominating the population, an excess of linkage disequilibrium is known to be a signature of a selective sweep. Population geneticists have used theoretical proofs and population genetic simulations to show that selective sweeps generate high LD between pairs of loci that are either on the adjacent left or right regions around the selected mutation [42]. However, the LD between one loci on opposite sides of the selected mutation tends to remain low. This pattern on the genome motivates the use of LD based summary statistics for detecting selective sweeps. In order to understand the logic behind the LD statistics, we must first cover the formal definition of linkage disequilibrium.

2.5.3.1 The Linkage Disequilibrium Coefficient

Consider a genome matrix $G_{n\times k}$ where n is the number of samples and k is the number of SNPs. There are c unique rows found in G which correspond to the c haplotypes. Let p_i be the proportion of haplotypes with state 1 (*i.e.* derived state) in column i: $i \in \{1, 2, \ldots, k\}$. Let p_{ij} the proportion of haplotypes that have state 1 in both columns $i, j : i, j \in \{1, 2, \ldots, k\}$, $i \neq j$.

Select two columns A and B from G.Let f_A be the proportion of rows in G which have state 1 in column A and f_a be the proportion with state 0. Similarly, let f_B be the proportion of rows with state 1 in column B and f_b be the proportion with state 0. Denote f_{AB} as the proportion of rows with state 1 in both columns A and B. The LD coefficient between columns A and B, D_{AB} is defined as,

$$D_{AB} \coloneqq f_{AB} - f_A f_B \tag{2.11}$$

If columns A and B are independent of each other then $f_{AB} = p_A \times p_B$. In this case $D_{AB} = 0$ and columns i and j are considered to be in linkage equilibrium [25, Chapter

6]. Should $D_{AB} \neq 0$, then the two columns are said to be in some degree of linkage disequilibrium. We can use this definition to compute the LD coefficient for all $\binom{k}{2}$ column pairs in G.

This brings us to the question of how to consider whether any given LD coefficient sufficiently deviates from 0 which would a substantial amount of LD between a given pair of loci. We can determine the range of possible values of D by writing out D in terms of haplotype proportions [25, Chapter 6]. Using some simple algebra and the law of total probability, the following equations can be derived.

$$f_{AB} = f_A f_B + D \tag{2.12}$$

$$f_{Ab} = f_A f_b - D \tag{2.13}$$

$$f_{aB} = f_a f_b - D \tag{2.14}$$

$$f_{ab} = f_a f_b + D \tag{2.15}$$

where f_k is the proportion of rows in G with allele state k. Note that capital letter represents a derived allele at that particular position whilst a lower case letter represents an ancestral allele. For example f_{Ab} is the proportion of rows with state 1 at SNP A and state 0 at SNP B. f_b is the proportion of rows with state 0 at SNP B.

Consider 2.13 and 2.14. Since proportions are bounded between 0 and 1, we can deduce that in cases where $D > 0$,

$$D \leq min(f_A f_b, f_a f_B)$$

By a similar logic, we can use 2.12 and 2.15 to deduce that when $D > 0$,

$$D \geq max(f_A f_B, f_a f_b) \equiv -D \leq min(f_A f_B, f_a f_b)$$

We can incorporate these results to modify our LD coefficient D in order to account for the range of possible values D may take.

Definition 2.1 (Standardised Linkage Disequilibrium Coefficient)**.**

$$D' = \frac{D}{min(f_A f_b, f_a f_B)} \; if \; D > 0 \tag{2.16}$$

$$= \frac{-D}{min(f_A f_B, f_a f_b)} \; if \; D < 0 \tag{2.17}$$

Since D' is a ratio between D and its maximum value, D' takes values between 0 and 1. For any two columns $i, j \in \{1, 2, \dots k\}, i \neq j$ within a genome matrix $G_{n\times k}$, we define the LD correlation coefficient,

Definition 2.2 (Linkage Disequilibrium Correlation Coefficient: r^2)**.**

$$r_{ij}^2 := \frac{D^2}{p_i(1-p_i)p_j(1-p_j)} \tag{2.18}$$

where p_s is the proportion of rows in G with state 1 in column $s, s \in \{i, j\}$.

Researchers have used LD measurements to infer whether a particular mutation is under selection [43]. We will now discuss a couple of summary statistics who uses the LD coefficient for detecting selective sweeps.

2.5.3.2 Kelly's Z_{nS}

Kelly's Z_{nS} is an LD-based summary statistic which uses the correlation between pairs of polymorphic sites (columns in the genome matrix) within a sampled population [44]. For a genome matrix $G_{n\times k}$ we define,

Definition 2.3 (Kelly's Z_{ns})**.**

$$Z_{ns} := \frac{2}{k(k-1)} \sum_{i=1}^{k-1} \sum_{j=i+1}^{k} r_{ij} \tag{2.19}$$

where r_{ij} is the LD correlation coefficient defined in Theorem 2.2.

Kelly's Z_{nS} is the average of the r^2 values across all possible column pairs in G. The expected value of Z_{nS} under the neutral model can be determined via simulations. Suppose a selected mutation is present at SNP position i. The selected mutation will carry along neighboring SNPs via genetic hitch-hiking. This increases the d_{ij}'s where j represents SNP columns in the neighborhood of i. Thus a selective sweep can inflate Z_{nS} relative to the neutral expectation.

Z_{nS} can be confounded by high recombination rates. When recombination occurs frequently, the selected mutation can quickly move onto new haplotypes. This reduces the effect of genetic hitch-hiking, thereby reducing linkage disequilibrium across the region and deflates Z_{nS}. Hence, tests for selection using Z_{nS} is confounded by recombination.

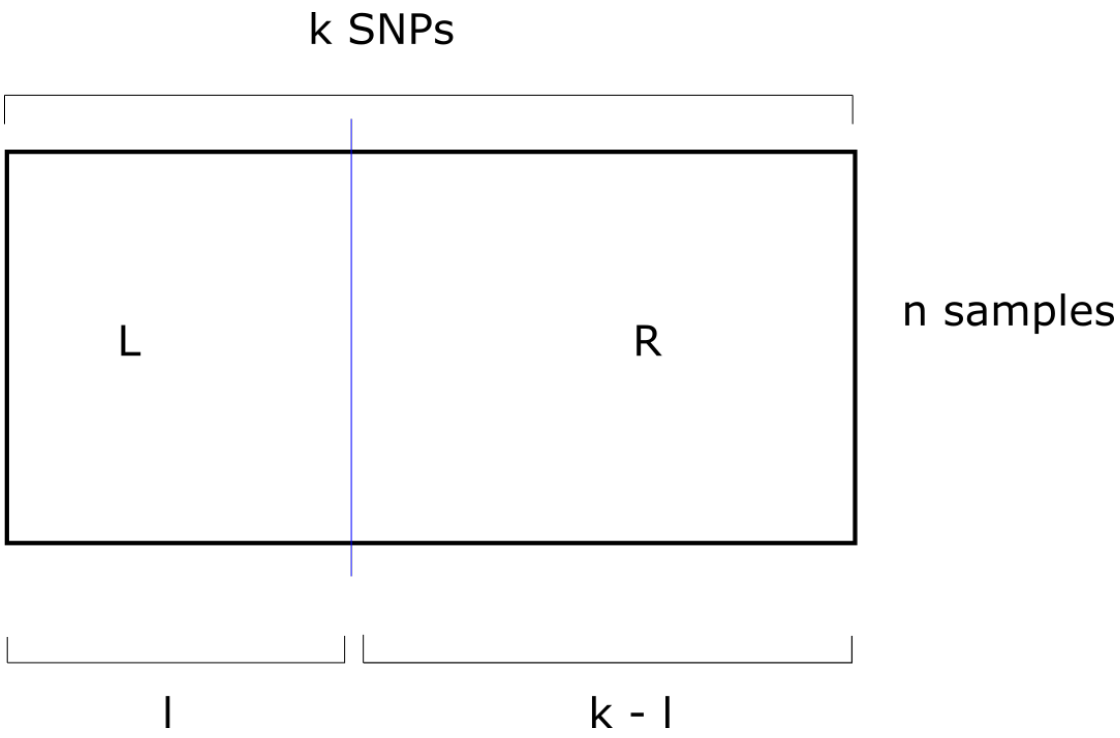

FIGURE 2.10: Illustration of how the w statistic is computed on a genome matrix $G_{n\times k}$. We break the genome matrix into 2 non- overlapping blocks. The left block L consists of the first l columns. The right block R consists of the remaining $k-l$ columns.

2.5.3.3 ω_{max}

Section 2.5.3 said that hard sweeps can produce high LD between pairs of loci within each side of the selected mutation but low LD across sides. Since Kelly's Z_{nS} averages the LD correlation for all loci pairs across a region, it doe not account for this spatial pattern. This motivates the next statistic, ω_{max} which was designed specifically to detect the spatial pattern of LD generated by selective sweeps [42].

Figure 2.10 provides an illustration of how to compute the ω statistic. For a genome matrix $G_{n\times k}$, we select a column l. We partition G into left and right halves (L and R), where L contains the first l SNPs and K contains the remaining $k-l$ SNPs. We define ω_l,

Definition 2.4 (ω_l)**.**

$$\omega_l \coloneqq \frac{[\binom{l}{2} + \binom{k-l}{2}]^{-1}(\sum_{i,j\in L} r_{ij}^2 + \sum_{i,j\in R} r_{ij}^2)}{[l(k-l)]^{-1}\sum_{i,j\in R} r_{ij}^2} \tag{2.20}$$

Theorem 2.4 compares the LD correlation coefficient within the blocks L and R verses the correlation between L and R. This is normalised by the number of loci pairs. Since

a selected mutation can be anywhere within the genome, we can compute ω for every column. ω_{max} is defined

Definition 2.5 (ω_{max})**.**

$$w_{max} \coloneqq \max_{l} w_l \tag{2.21}$$

The logic is that the LD spatial pattern is strongest when ω is computed close to selected mutation. For neutral regions, ω_{max} should be small because the LD within L and R should be similar to that between L and R. A hard sweep produces higher LD within L and R, thereby inflating ω_{max} [45].

Simulation studies have shown that ω_{max} is more robust to recombination compared to statistics which average the level of LD across an entire region [42]. This is because ω_{max} was designed to detect the LD spatial pattern produce by hard sweeps.

2.6 Conclusion

This chapter covered the biological background of this project. DNA is a biological code consisting of four letters which contains the genetic information of all organisms. Darwin's theory of natural selection which describes how populations acquire adaptive traits in response to environmental challenges. These ideas were brought together in our discussion of population genetics which studies how evolution acts on the genome on a population level. Whilst evolution can produce a range molecular patterns, we focused on selective sweeps (particular hard sweeps) because they are the most well studied. We then covered three classes of summary statistics (SFS, haplotype, LD) designed by population geneticists for sweep detection. Having presented the biological problem, the next chapter will cover the statistical background of machine learning.

Chapter 3

Statistical Background

3.1 Machine Learning

Machine learning refers to a broad set of statistical tools for understanding data. Data that is put into a machine learning tool is typically called the training data set. Data can be thought of as a set of observations where n predictor measurement(s) could be made. Thus any data point i can be represented by a n dimensional vector $\mathbf{x}_i$ [46, Chapter 2]. In a labelled data set, there is an additional response measurement y_i which is associated with each observation. In statistical learning theory, the measurements $\mathbf{x}_i$ are known as the predictors and the y_i's are called response variables. Some machine learning literature also call $\mathbf{x}_i$, feature vectors [47, Chapter 2].

Machine learning tasks can be generally divided into supervised and unsupervised learning. In unsupervised learning, the training data set consists entirely of predictor variables (*i.e.* unlabelled). The goal of unsupervised learning is to find the underlying relationships between the predictors and the data points. An example of unsupervised learning is when we analyse the purchasing behaviors at a department store where each item has been assigned to single class of which there are fifty in total. Each observation is a vector $\mathbf{x} \in \{0, N\}^{50}$, where the i^{th} element is the number of items purchased belonging to class i. Based on the training data $\tau = \{\mathbf{x_1}, \mathbf{x_2}, \ldots \mathbf{x_n}\}$, an unsupervised method may find distinct groups of customers who purchase certain classes of items together (*e.g.* customers who buy children's clothes also buy toys). It is generally difficult to assess the performance of unsupervised learning because what constitutes "good" result is not clearly specified. Consequently, unsupervised learning is typically used for exploratory data analysis where the researcher is trying to visualize interesting trends in the data. Commonly used methods in unsupervised learning include principal component analysis (PCA) and hierarchical clustering.

In supervised learning, the training data consists of a set of predictors $\mathbf{x}$ and a response variable y. The object of supervised learning is to fit a mathematical model that relates the predictors to the response. Models are useful for predicting the response for future observations (prediction) and to understand the underlying relationship between the predictors and the response (inference) [46, Chapter 2]. Supervised learning can be divided into regression and classification problems. In regression, the response is a continuous numeric variable whilst in classification, the response is a categorical variable. An example of a regression problem would be to use a patient's physical measurements (*e.g.* height, weight) to predict their lung capacity. An example of a classification problem would be to use a person's financial information (*e.g.* income, savings, requested loan amount) to predict whether they will default on a particular loan. Some common regression methods include linear models, lasso regression and ridge regression. Common machine learning classifiers include regularised discriminant analysis and naive Bayes. Note that some methods such as random forests, multivariate adaptive regression splines (MARS) and neural networks can be used for both classification and regression. The aim of this project is to use machine learning classifiers to differentiate selective sweeps from neutrally evolving genetic sequences. This chapter will explain the general, mathematical theory behind machine learning, with a particular focus on classification. This will be followed by a discussion of several machine learning methods, including both supervised and unsupervised techniques, that are commonly used in modern data analysis. Finally, we will explore novel techniques to investigate variable importance which is useful for understanding how our models make their predictions.

3.2 Supervised Classification Paradigm

This section will discuss the mathematical framework behind supervised learning for classification. For any classification task, we have a training data set $\mathbf{T}$ with n observations

$$\mathbf{T} = \{\boldsymbol{\tau}_1, \boldsymbol{\tau}_2, \ldots, \boldsymbol{\tau}_n\} \tag{3.1}$$

$$= \{(y_1, \mathbf{x}_1), (y_2, \mathbf{x}_2), \ldots, (y_n, \mathbf{x}_n)\} \tag{3.2}$$

with predictors $\mathbf{x}_i = (x_{i1}, x_{i2}, \ldots, x_{ip}), i \in \{1, 2, \ldots, n\}$ and response variable $y_i = C_k, k \in \{1, 2, \ldots, K\}$. We assume there is some true, underlying relationship $f : \mathbf{x} \longrightarrow y \in \{C_1, C_2, \ldots, C_K\}$. Although f is not directly observable, we can use the training data as a guide to create a predicting function $\hat{f}$ which is a functional approximation of f [48,

Chapter 2]. The approximating function $\hat{f}$ is also known as the learner in some technical literature. The key purpose of $\hat{f}$ is to take future observations $\mathbf{x}$ and assign them to one of the K classes. We can measure how accurately the predictions of $\hat{f}$ is matching the training data using a loss function, $Loss\Big(y_i, \hat{f}(\mathbf{x}_i)\Big)$. Loss functions essentially compare the learner's prediction with the true response and different loss functions have been developed for classification and regression problems. A simple loss function for classification is to count the number of misclasses, *i.e.* $Loss\Big(y_i, \hat{f}(\mathbf{x}_i)\Big) = I\Big(y_i \neq \hat{f}(\mathbf{x}_i)\Big)$, where I is the indicator function.

Definition 3.1 (Indicator Function)**.**

$$I\Big(y_i \neq \hat{f}(\mathbf{x}_i)\Big) = \begin{cases} 0, & \text{if } y_i = \hat{f}(\mathbf{x}_i) \\ 1, & \text{otherwise} \end{cases} \tag{3.3}$$

Since the loss function quantifies the amount of predictive error in the learner, the optimal learner is the one which minimizes the overall training loss [47, Chapter 2]. The task of finding the optimal learner given some training data set can be formally written as the following optimization problem.

Definition 3.2 (Optimal Prediction Function)**.** Let $\mathcal{F}$ be a function space consisting of a family of approximating functions $f : \mathbf{x} \longrightarrow y$. The optimal prediction function from $\mathcal{F}$ given training data $\mathbf{T}$, $f_{\mathbf{T}}^{\mathcal{F}}$ is defined as

$$f_{\mathbf{T}}^{\mathcal{F}} \coloneqq \min_{f \in \mathcal{F}} \left(\frac{1}{n} \sum_{i=1}^{n} Loss\Big(y_i, \hat{f}(\mathbf{x}_i)\Big) \right) \tag{3.4}$$

The learning component of machine learning is essentially the optimization process whereby we explore $\mathcal{F}$ in order to find a function $f \in \mathcal{F}$ which minimizes the training loss. For example in general linear models, $\mathcal{F}$ is the set of linear functions $f : \mathbf{x} \longrightarrow \beta^T \mathbf{x}$ for all real vectors β of the correct dimensions. The optimal learner $f_{\mathbf{T}}^{\mathcal{F}}$ is a linear function with the unique β which minimizes the training loss. For some simple machine learning methods (*e.g.* linear regression), the optimal learner can be computed using an analytical result. More complex machine learning methods require the application of an optimization algorithm to find the best learner. For example, random forests have a built in algorithm for generating an ensemble of classification and regression trees for prediction. In overparameterized models where the number of model parameters far exceed the number of observations in the training data set (*e.g.* neural networks), the loss function is usually highly complex and multimodal. In these cases, finding the optimal solution is computationally intractable, provided a unique solution even exists.

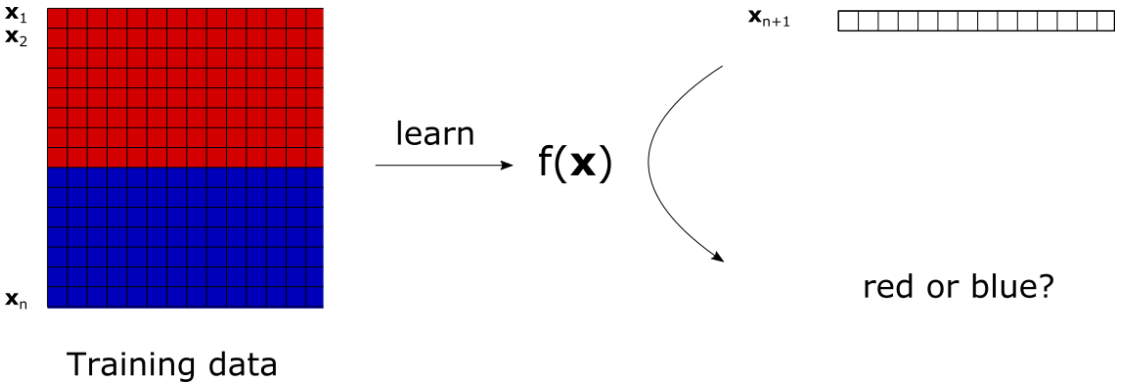

FIGURE 3.1: An illustration of supervised learning with 2 classes. The feature vectors $\mathbf{x}$ are represented by the rows of squares. The 2 classes are denoted by the colours. The objective of machine learning is to find a suitable function f which can accurately assign classes to new observation.

Instead researchers opt for using algorithms such as stochastic gradient descent and backpropagation which can find a local minimum in the loss function. Recall that the key application of machine learning is to make an accurate predictive model [49]. Thus depending on the machine learning method chosen, it is sometimes acceptable to not have the theoretically best model, provided the final model is sufficiently accurate when predicting future observations. In some statistical learning literature, the optimization procedure whereby we find the optimal learner given some training data, is also known as "model fitting" and "model training." After model fitting, the final learner is tested on a new, unseen test data set. Once again, a loss function is used to compare the learner's predictions with the true response. It is the learner's performance on the test data, rather than the training data, that is used as a final evaluation of model's predictive accuracy. The reason for this is the potential of overfitting which will be discussed in the next section.

3.2.1 Bias Variance Trade Off

At the conclusion of Section 3.2, we noted that the final learner is evaluated by using its predictive accuracy on new, unseen testing data. One may wonder why it is not appropriate to use the learner's training accuracy instead. The reason is overfitting, a phenomenon whereby a model fits surprisingly well on the training data but performs poorly when it is used to predict novel data.

Figure 3.2 is a visual representation of how overfitting occurs. Here we have some toy data and the task is to separate the red and blue points based on the predictor variable x. Since there is a linear relationship between colour and x, a linear decision boundary seems sensible. Points above a certain threshold of x will be classed as red, otherwise they will be classed as blue. In order to maximize the training accuracy, we could place

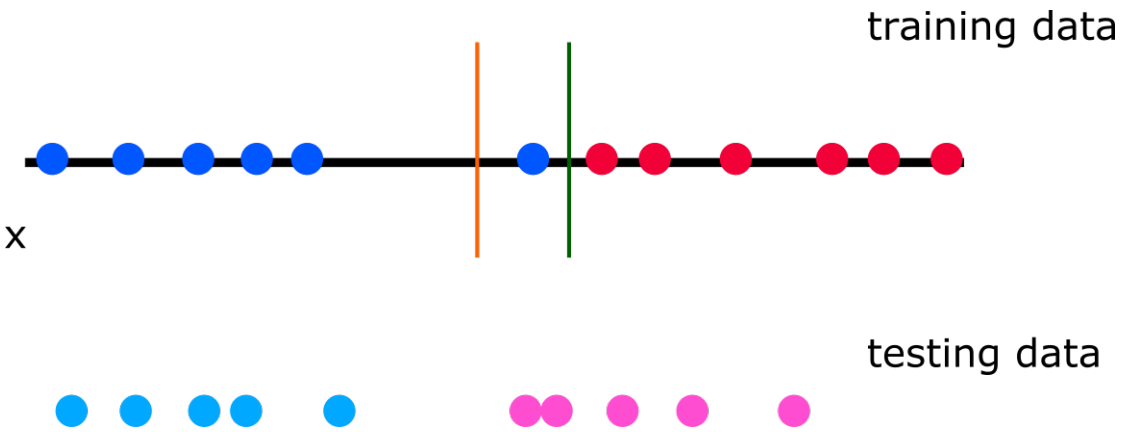

FIGURE 3.2: An illustration of overfitting. Each data point consists of a continuous variable x and a class which is denoted by its colour (red or blue). The darker circles represent the training data and the lighter circles represent the testing data. The green classifier performances perfectly on the training data but performs poorly on the testing data, relative to the orange classifier. The green classifier is overfitted.

the threshold between the closest pair of red and blue points. To prevent any bias towards either side, the threshold may be placed equidistant between the two points. This is known as the maximum margin classifier and is marked by the green line in the figure. Although the green threshold has a perfect training accuracy, it has two misclasses on the testing data. The orange line is closer towards the center of the data points. The logic is that the blue point furthest on the right is an outlier and does not justify moving the threshold that far away. Although the orange threshold has one misclassification in the training data, it has a perfect accuracy on the testing data. The green model is considered to be overfitted because it has become too reliant on following the training data at the expense of having a higher error rate when predicting unseen testing data. This toy example has illustrated two key points. 1. High accuracy in the training data does not necessarily entail a high accuracy on the testing data. A high training accuracy could simply be due to an overfitted model. 2. A model which is less sensitive to small fluctuations in the training data can actually perform better on unseen data, compared to overfitted models which fit the training data perfectly.

This example above is a simple illustration of bias-variance tradeoff in supervised machine learning. When training a predictive model, there are two main sources of error which we seek to minimize. The first is the bias which refers to the error introduced by approximating a complex real-life problem with a much simpler model [46, Chapter 2]. Models which poorly capture the relationship between the response and the predictors are said to be "underfitted." The second source of error is the model variance which refers to how much the model will change if we fitted it with a different set of training data. Models for which small changes in the training data can produce large changes in its predictions, have high variance and are said to be "overfitted." Ideally,

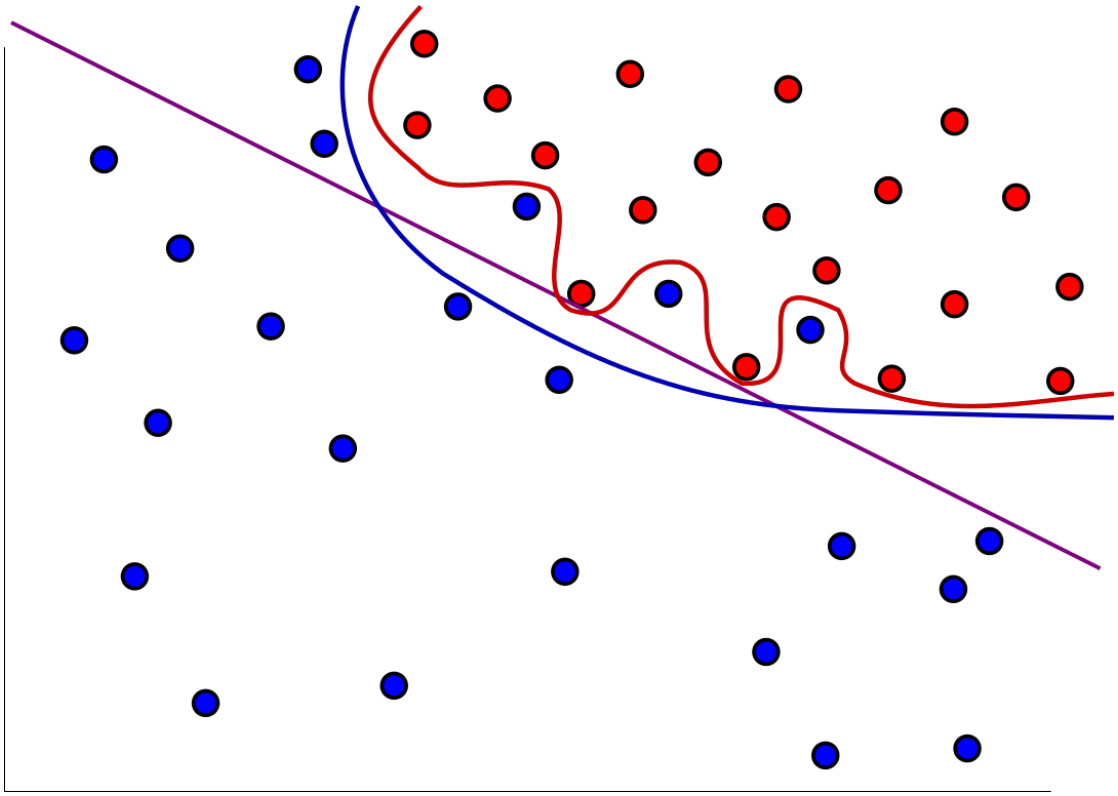

FIGURE 3.3: An illustration of bias-variance tradeoff. Each data point consists of 2 continuous variables (predictors) and a class (response) which is denoted by its colour (red or blue). The purple classifier has high bias but low variance. The red classifier has low bias but high variance. The blue classifier has moderate variance and bias, thus is probably the best model of the three.

we would like a sufficiently flexible model which can adequately capture the underlying relationships between the predictors and the response (low bias). The model should also provide consistent predictions that are robust to small fluctuations in the training data (low variance). In the context of 3.2, we want to ensure $\mathcal{F}$ is sufficiently rich in functions that can model the underlying relationship. However, $\mathcal{F}$ must not contain excessively complex functions as there will typically be some contrived function which fits the training data perfectly, thereby producing an overfitted model.

Consider the classifiers in Figure 3.3. The training data indicates that there is a non-linear relationship between the predictors (represented by the location of the points) and the response (colour). The purple classifier tries to separate the data using a linear decision boundary. This fails the capture the more complex curvature in the data so the model has high bias. However, small changes in the training data will cause little change in the overall model. Thus the model is said to be low variance. This is an example of an underfitted model. We can also think of model variance as its flexibility, *i.e.* how well the model fits the training data. The red classifier is "wiggly" and fits the training data perfectly, making it low bias. However, its "wigglyness" means that small changes in the

data will drastically change the model. Hence the model has high variance. This is an example of an overfitted model. The blue classifier adequately captures the non-linear trend among the data points without being wiggly, thereby having a good balance of bias and variance. This model may be expected to perform better than the others as it fits reasonably well to the training data without overfitting.

Machine learning methods differ in their degree of bias and variance. As a general rule, flexible methods (*e.g.* splines, regression/classification trees) are great at fitting to the training data, even when the underlying relationships between the predictors and the response are complicated [46, Chapter 2]. This is achieved at the cost of producing a higher variance model with a greater risk of overfitting. Simpler and less flexible methods (*e.g.* linear models, logistic regression) have lower risk of overfitting but tend to fit worse to training data, especially when the underlying trends in the data are complex. Mathematically, flexible methods consider a function space $\mathcal{F}$ which is rich in complex approximating functions whilst less flexible methods constrain $\mathcal{F}$ to more simple functions. Overall, there is no single machine learning method that is guaranteed to outperform all others methods across all data sets. The effectiveness of a method will depend on the kind of data available and the underlying relationships we are trying to model. Thus one of the challenges of supervised machine learning is to select a method that is sufficiently flexible so as to capture all the interesting relationships in the training data (low bias), without being too flexible so as to overfit (low variance).

3.2.2 Regularisation and Hyperparameter Tuning

3.2.2.1 Regularisation

Section 3.2.1 introduced the concept of overfitting, a phenomenon whereby a predictive model's high performance on the training data does not generalise to new, unseen data sets. Flexible methods are particularly prone to this problem as there will typically exist some contrived, complicated function which predicts the training data almost exactly. Thus for the purposes of finding a good predictive model, the function which minimizes the training loss may not always be appropriate. Regularisation is a mathematical technique which addresses overfitting by adding a penalty term which punishes model complexity [47, Chapter 2]. This technique is used improve the predictive accuracy of the learner by reducing model variance [46, Chapter 6].

Definition 3.3 (General Form of Regularised Optimization Problem)**.** The optimal regularised learner for function space $\mathcal{F}$, given training data $\mathbf{T}$ and some choice of some

$\lambda > 0$ is defined as

$$f^{\mathcal{F}}_{\mathbf{T},\lambda} \coloneqq \min_{f \in \mathcal{F}} \left(\frac{1}{n} \sum_{i=1}^{n} Loss\Big(y_i, \hat{f}(\mathbf{x}_i)\Big) + \lambda J(f) \right) \tag{3.5}$$

where $J(f)$ is a penalty functional and $\mathcal{F}$ is the function space on which $J(f)$ is defined.

The functional J is used to penalise learners for being complex and flexible. This helps to ensure that a balance is struck between keeping the training loss small whilst also maintaining a simple predictive model. Ideally, we would like the model to be just complex enough to explain the trends in the data and no more. This is known as the principle of parsimony. λ is an example a tuning parameter (a.k.a hyperparameter), a parameter chosen prior to model fitting which can control the learning process. As $\lambda \longrightarrow 0$, the regularised learner approaches the optimal prediction function in Theorem 3.2, which minimizes the training loss. As $\lambda \longrightarrow \infty$, the regularised learner approaches the simplest function $f \in \mathcal{F}$ which has the smallest penalty of $J(f)$. Different machine learning methods each have their own set of tuning parameters to assist researchers in finding a good predictive model.

3.2.2.2 Example: Ridge Regression

A simple example of regularisation is ridge regression which builds upon traditional linear regression [47, Chapter 6]. Consider a training data set $\mathbf{T} = \{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)\}, \mathbf{x} \in \mathbb{R}^p$. In tradition linear models, the problem is to solve

$$\min_{f \in \mathcal{F}} \left\{ \frac{1}{n} \sum_{i=1}^{n} \Big(y_i - f(x_i)\Big)^2 \right\} \tag{3.6}$$

where $\mathcal{F}$ is the space of linear functions $f : \mathbf{x} \longrightarrow \beta^T \mathbf{x} + \beta_0$, $\forall \beta \in \mathbb{R}^p, \beta_0 \in \mathbb{R}$. In other words, the objective is to find the unique straight linear function which minimizes the mean squared error in the training data. Ridge regression modifies the model fitting process by adding a penalty term

$$\min_{f \in \mathcal{F}} \left\{ \frac{1}{n} \sum_{i=1}^{n} \Big(y_i - f(x_i)\Big)^2 + \lambda \left\|\beta\right\|^2 \right\} \tag{3.7}$$

where $\|.\|$ is the L_2 norm applied to the slope coefficients of f (*i.e.* the intercept term β_0 is not penalised). As $\lambda \longrightarrow 0$, the ridge solution approaches the traditional linear model (lower bias, higher variance). As $\lambda \longrightarrow \infty$, the ridge solution approaches a flat

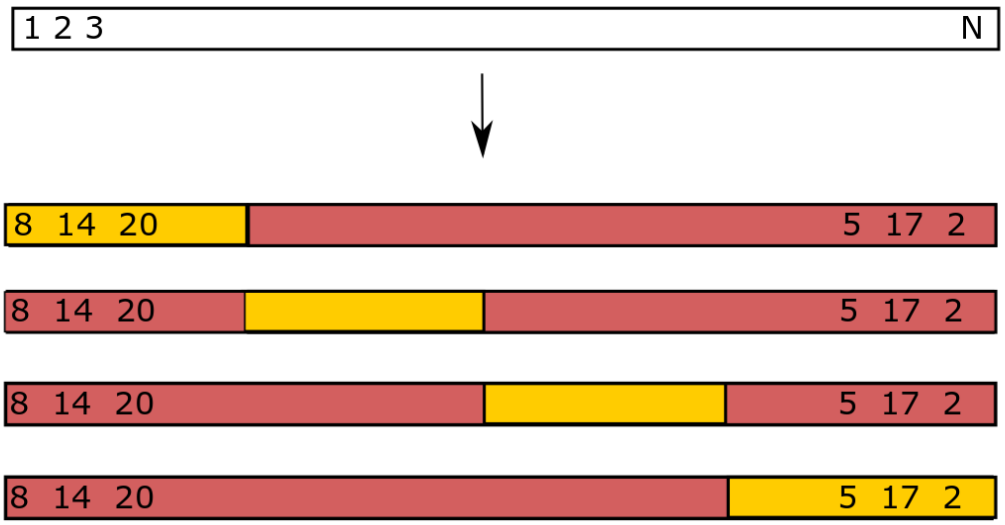

FIGURE 3.4: An illustration of K-fold cross validation with $K = 4$. The numbers represent arbitrary indices attached to each observation in the training data.

line (higher bias, lower variance). Thus the tuning parameter λ is like a dial enabling us to tune our model in order to strike a good balance between bias and variance. The process of selecting an appropriate set of tuning parameters is known as "model tuning" and "hyperparameter tuning."

3.2.2.3 Cross Validation

Section 3.2.2.1 introduced the concept of tuning parameters, a set of parameters chosen prior to model fitting which can influence the optimization process of finding the best learner. An important part of supervised learning is to identify an appropriate set of tuning parameters in order to produce an accurate predictive model. A standard method for model tuning is k-fold cross validation. Figure 3.4 is an illustration of cross validation. K-fold cross validation involves randomly partitioning a training data set $\mathbf{T} = \{(y_i, \mathbf{x}_i), i = 1, 2, \ldots, n\}$ into K approximately equal sized parts $C_1, C_2, \ldots C_K$, where C_i denotes set of observations in part i. For each partition, we fit the model on the other $K - 1$ partitions and use the fitted model to predict for the part removed. Hence, for each partition k we compute the associated loss,

$$Loss_k = \frac{1}{n_k} \sum_{i \in C_k} Loss\Big(y_i, \hat{f}(x_i)\Big) \tag{3.8}$$

where $\hat{f}$ is the fitted model using data with partition k removed and n_k is the number of observations in partition k. We can apply a weighted average over all K partitions to compute an overall K-fold cross validation accuracy.

$$CV_{(K)} = \sum_{k=1}^{K} \frac{n_k}{n} Loss_k \tag{3.9}$$

The cross validation loss provides an overall estimate of a model's predictive accuracy across the whole training set. As a general rule, this estimate is reasonably good at 10 folds [50].

Each machine learning method has its own set of hyperparameters which constrain the complexity of the final model. Some methods such as random forests and MARS, have multiple hyperparameters. For the purposes of hyperparameter tuning, we can construct a regular grid of hyperparameters and compute the cross validation loss over each set [46, Chapter 5]. We then select the set of hyperparameters which give the smallest cross validation loss. Finally, we refit the model on the whole training data set, using our selected set of hyperparameters. This gives us the final predictive model which can be evaluated on the testing data set.

3.2.2.4 Conclusion: Supervised Learning

This section has provided a concise overview of machine learning. Machine learning can be broadly divided into two categories; namely unsupervised and supervised learning. In unsupervised learning, the training data consists entirely of predictors. The objective of unsupervised methods is to find interesting clusters in the training data. Supervised learning uses training data with predictors and a response variable. The objective is to construct a predictive model to predict future observations. This is achieved by selecting a machine learning method and fitting a model onto the training data. Most machine learning methods have hyperparameters which control the learning process and these can be selected via cross validation. Once the final model has been fitted, it is used to predict new, unseen testing data in order to evaluate its predictive accuracy. Having presented

the general framework of machine learning, we will now explore several machine learning methods that are commonly used in modern data analysis.

3.3 Machine Learning Methods

3.3.1 Principal Component Analysis

Principal component analysis (PCA) is an unsupervised learning method used primarily for exploratory data analysis. Suppose we have some training data $\mathbf{T} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}, \mathbf{x} \in \mathbb{R}^p$. To observe the relationships between the predictors, we may look at their pairwise plots. For any p dimensional data, there are $\frac{\binom{p}{2}}{2}$ possible pairwise plots to consider [46, Chapter 10]. Typically, many plots will be uninformative or will only contain a fraction of the total information in the data set. Thus as p gets large, inspecting all the pairwise plots becomes exhaustive and inefficient. PCA addresses this problem by finding a low dimensional representation of the data which captures most of the underlying information. PCA is an example of a dimension reduction method and is particularly useful for visualizing complex data.

Principal components are normalised, orthogonal linear combinations of the predictors. The first principal component (PC1) is the linear combination

$$Z_1 = \phi_{11}x_1 + \phi_{21}x_2 + \cdots + \phi_{p1}x_p \ , \ \sum_{i=1}^{p} \phi_{i1}^2 = 1 \tag{3.10}$$

which has the largest variance. The elements $\phi_{11}, \phi_{21}, \ldots, \phi_{p1}$ are called the principal component loadings and $\boldsymbol{\phi}_1 = (\phi_{11}, \phi_{21}, \ldots, \phi_{p1})^T$ is known as the principal component loading vector. The constraint $\sum_{i=1}^{p} \phi_{i1}^2 = 1$ ensures that the ϕ's do not arbitrarily large which could result in arbitrarily large variance. $\boldsymbol{\phi}_1$ is computed by solving

$$\underset{\boldsymbol{\phi}_1}{\operatorname{argmax}} \left(\boldsymbol{\phi}_1^T \Sigma \boldsymbol{\phi}_1 \right) \ , \ \boldsymbol{\phi}_1^T \boldsymbol{\phi}_1 = 1 \tag{3.11}$$

where Σ is the covariance matrix of the predictors. Using the method of Lagrange multipliers, it can be shown that $\boldsymbol{\phi}_1$ is given by the eigenvector corresponding to the largest eigenvalue of the variance-covariance matrix Σ. The second principal component loading vector $\boldsymbol{\phi}_2$ is computed by solving

$$\underset{\boldsymbol{\phi}_2}{\text{argmax}} \left(\boldsymbol{\phi}_2^T \Sigma \boldsymbol{\phi}_2 \right), \ \boldsymbol{\phi}_2^T \boldsymbol{\phi}_2 = 1, \ \boldsymbol{\phi}_1^T \boldsymbol{\phi}_2 = 0 \tag{3.12}$$

The last constraint $\boldsymbol{\phi}_1^T \boldsymbol{\phi}_2 = 0$, ensures that the principal components are orthogonal. Subsequent principal components can be constructed by repeating this process, ensuring that the principal components are pairwise orthogonal. For p dimensional data, up to p principal components can be computed.

We can plot the data points using the first few principal components (*e.g.* PC1 and PC2) to visualize the spread and clustering within the data. Since the principal components are orthogonal and have been constructed to account for as much variance in the data as possible, many interesting trends in the data would be captured by the first few principal components. Researchers can typically decide on the number of principal components required for visualising the data using a scree plot. The scree plot shows the proportion of variance that is explained by each principal component. Typically the proportion of variance explained falls off after the first few principal components. By observing the scree plot, we can select the smallest number of principal components which explain a sizable amount of variation in the data. In the context of supervised classification, PCA can still be used to visualise the spread between data points of different classes. Although the response variable cannot be used directly in the PCA algorithm, we can colour the data points by class when we plot them by their principal components. This provides a visual indication of how well the predictors may be separating the different classes. Trends in the PCA plot may also inform what machine learning methods to use for classifying the data. For example, if the classes can be separated via a linear decision boundary then a simple method such as logistic regression may be useful. If the classes have complex non-linear relationships with the top few principal components, then a flexible non-linear method such as quadratic discriminant analysis may be useful. We will now explore several classification methods commonly used in supervised machine learning.

3.3.2 K-means Clustering

K-means clustering is an unsupervised technique which partitions N data points into K distinct, non-overlapping clusters [46, Chapter 10]. K must be specified by the user to start the algorithm.

Let $C_1, C_2, \ldots, C_K$ denote the sets containing the indices of the observations in each cluster.

$$C_1 \cup C_2 \cup \cdots \cup C_K = \{1, 2, \ldots, n\} \tag{3.13}$$

This means each observation is assigned to a cluster.

$$C_i \cap C_j = \phi \forall i \neq j \tag{3.14}$$

This ensures observations are assigned to exactly one cluster. A good clustering is one which minimises the within-cluster variation. Hence, we solve

$$\min_{C_1, C_2, \ldots, C_K} \left\{ \sum_{k=1}^{K} W(C_k) \right\} \tag{3.15}$$

where $W(.)$ is a distance function. A common choice is the squared Euclidean distance

$$W(C_k) = \frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2 \tag{3.16}$$

where $|C_k|$ is the number of observations in the k^{th} cluster, p is the number of predictors and x_{ij} is the value of the j^{th} predictor in observation i.

The k-means algorithm starts by randomly assigning each observation to a cluster. The following steps are iterated until there is no change in cluster assignment [48, Chapter 10].

1. Compute the centroid for each cluster. The j^{th} cluster centroid is the vector of p predictor means for all the observations in C_j.
2. Assign each observation to the cluster with the closest centroid. "Closeness" is defined by a predetermined distance function, usually Euclidean square distance.

These steps reduce Equation (3.16) so convergence is assured [48, Chapter 14]. However, the final result may represent a suboptimal local minimum and would vary depending on the starting cluster assignment.

Selecting a suitable K is tricky because this is an unsupervised problem. One method is to use the silhouette value which measures how similar an observation is to its own cluster (cohesion) compared to other clusters (separation) [51].

For data point $i \in C_i$, let

$$a(i) \coloneqq \frac{1}{|C_i| - 1} \sum_{j \in C_i, i \neq j} d(i, j) \tag{3.17}$$

where $d(i, j)$ is the distance between points i and j. The Euclidean square distance can be used here again. $a(i)$ measures how similar point i is compared to all other points in its assigned cluster (cohesion). For the same point $i \in C_i$, define

$$b(i) \coloneqq \min_{k \neq i} \left\{ \frac{1}{|C_k|} \sum_{j \in C_k} d(i, j) \right\} \tag{3.18}$$

$b(i)$ smallest mean distance between i and all the points in any other cluster except C_i. $b(i)$ measures the distance between i and its next closest cluster (separation).

Define the silhouette value of point i

$$s(i) \coloneqq \begin{cases} \frac{b(i) - a(i)}{max\{a(i), b(i)\}}, \text{if } |C_i| > 1 \\ 0, \text{if } |C_i| = 1 \end{cases} \tag{3.19}$$

$s(i) \in [-1, 1]$. The score for clusters with size 1 is 0 to ensure the number of clusters is not inflated. When $s(i) \approx 1$, $a(i) \ll b(i)$ meaning point i is well matched to its assigned cluster and dissimilar from neighboring clusters (high cohesion, high separation). When $s(i) \approx -1$, $a(i) \gg b(i)$ meaning i is dissimilar its assigned cluster and similar to its neighboring cluster (low cohesion, low separation).

Let $K = \{2, 3, \ldots, n\}$. To determine the $k \in K$ which produces the best clustering, solve

$$\underset{k \in K}{\text{argmax}} \left\{ \frac{1}{n} \sum_{i=1}^{n} s(i) \right\} \tag{3.20}$$

I.e. The best clustering is one where most/all the data points are well matched to its own cluster (high cohesion) and poorly matched to its neighboring clusters (high separation).

3.3.3 Regularised Logistic Regression

Logistic regression is a supervised classification method and a specific case of a general linear model. A general linear model consists of three parts; namely 1: a random component with an underlying distribution, 2: a systematic component specifying a linear combination of the predictors, 3: a link function which captures the relationship between 1 and 2. Consider some training data $\mathbf{T} = \{(y_i, \mathbf{x}_i)\},\ i \in \{1, 2, \ldots, N\}, y \in \{0, 1\}, \mathbf{x} \in \mathbb{R}^p$. In the simple case of classification with two classes, the random component would be the response variable y indicating either a success or failure (*i.e.* binomial distribution). The systematic component would be a linear model of the p predictors, $\beta_0 + \beta^T\mathbf{x}$. Ideally, we would like a regression model p which can provide the probability of a success given some feature vector $\mathbf{x}$. Consider the standard linear model

$$p(\mathbf{x}) = \beta_0 + \beta^T\mathbf{x} \tag{3.21}$$

The problem with this approach is that the range of p extends beyond $[0, 1]$ and thus the model potentially outputs invalid probabilities [46, Chapter 4]. To avoid this problem, we need to use a link function $l : \mathbf{x} \longrightarrow [0, 1]\ , \forall \mathbf{x} \in \mathbb{R}^p$. In logistic regression, we use the logistic function

$$p(\mathbf{x}) = \frac{e^{\beta_0 + \beta^T\mathbf{x}}}{1 - e^{\beta_0 + \beta^T\mathbf{x}}} \tag{3.22}$$

where p has range $[0, 1]$. Algebraic manipulation gives

$$log\Big(\frac{p(\mathbf{x})}{1 - p(\mathbf{x})}\Big) = \beta_0 + \beta^T\mathbf{x} \tag{3.23}$$

The quantity $\frac{p(\mathbf{x})}{1-p(\mathbf{x})}$ is known as the odds and always takes values within $[0, \infty]$. Small odds values (*i.e.* close to 0) indicate high probability of failure whilst large odds indicate a high probability of success. The standard method for fitting a logistic regression model

is to solve for the set of β coefficients which maximize the log-likelihood function [48, Chapter 4].

$$\max_{\boldsymbol{\beta}\in\mathbb{R}^{\boldsymbol{p}},\boldsymbol{\beta_0}\in\mathbb{R}} \left\{ \sum_{i=1}^{N} \left(y_i(\beta_0 + \beta^T \mathbf{x}_i) - log(1 + e^{(\beta_0+\beta^T \mathbf{x}_i)}) \right) \right\} \tag{3.24}$$

The log-likelihood function is assumed to be binomially distributed with parameter $p(x)$. $max(f) \equiv min(-f)$ for any function f. Thus if we were to express this using the form in Theorem 3.2, then we would be minimising the negative of the log-likelihood function. The β coefficients can be computed using the Newton-Raphson algorithm for finding roots.

Regularised versions of logistic regression add a penalty term

$$\max_{\boldsymbol{\beta}\in\mathbb{R}^{\boldsymbol{p}},\boldsymbol{\beta_0}\in\mathbb{R}} \left\{ \sum_{i=1}^{N} \left(y_i(\beta_0 + \beta^T \mathbf{x}_i) - log(1 + e^{(\beta_0+\beta^T \mathbf{x}_i)}) \right) + \lambda J(\beta) \right\} \tag{3.25}$$

where $\lambda \geq 0$ is a tuning parameter and J is a penalty function applied to the slope coefficients β. For example, in lasso regression, J is the L_1 norm and in ridge regression J is the L_2 norm. Mixed penalties combine lasso and ridge regression together and have an addition tuning parameter $p \in [0, 1]$. p controls the proportion of the penalty placed on the L_1 norm as opposed to the L_2 norm [52, 53]. Coordinate descent methods can efficiently solve for these coefficients for a grid of λ values.

The logistic model can be generalised to classify > 2 classes although that technique is no longer in common use [46, Chapter 4]. The main reason is simply because there are other methods such as linear discriminant analysis (LDA) which perform better in multi-class scenarios. (Having said this, some authors comment that LDA and logistic regression typically give similar results, even in multi-class scenarios [48, Chapter 4]) We will explore LDA and other discriminant techniques later in this chapter.

Overall logistic regression is a simple and computationally quick method for supervised classification. Although it can be generalised to classify > 2 classes, researchers typically use it for the supervised learning with 2 classes. In addition to being a good predictive model, it can also be used to make inferential statements about the predictor terms [48, Chapter 4]. This is done by inspecting the p-values of the slope coefficients and checking for statistical significance. The logistic model is also mathematically simple,

enabling researchers to discern the relationship between the predictors and the response variables.

3.3.4 Random Forests

Random Forests is a tree-based supervised learning technique that can be used for both regression and classification. Due to the nature of this project, we will focus on how random forests work in a classification context. Nevertheless, the same theory applies to the regression context with some minor adjustments. Before we explain the random forest algorithm, we must first discuss how classification trees are built.

3.3.4.1 Classification Trees

The general intuition of classification trees is to successively split the predictor space into rectangular regions where each region largely consists of observations belonging to the same class. Consider some training data $\mathbf{T} = \{(y_i, \mathbf{x}_i)\},\ i \in \{1, 2, \ldots, N\},\ y \in \{C_1, C_2, \ldots, C_k\},\ \mathbf{x} \in \mathbb{R}^p$.. Building a classification tree roughly involves two steps [46, Chapter 6].

1. Divide the predictor space $\mathbf{x} \in \mathbb{R}^p$ into J distinct, non-overlapping, rectangular regions $\{R_1, R_2, \ldots, R_J\}$.

2. Every observation in $R_m,\ m \in \{1, 2, \ldots, J\}$, will have predicted class probabilities based on consensus votes of the N_m training data points in R_m. Specifically, let $\hat{p}_{mk}$ be the predicted probability that an observation in R_m is of class k. This will be given by the expression

$$\hat{p}_{mk} = \frac{1}{N_m} \sum_{\mathbf{x_i} \in R_m} I\Big(y_i = k\Big) \tag{3.26}$$

where $I(.)$ is the indicator function found in Theorem 3.1. The final class prediction for any observation $\mathbf{x} \in R_m, k(m)$ is given by

$$k(m) = \underset{k}{\text{argmax}}\{\hat{p}_{mk}\} \tag{3.27}$$

The objective is to find the regions $R_1, R_2, \ldots, R_J$ which minimize

$$\sum_{j=1}^{J} \sum_{i \in R_j} Loss(y_i, k(j)) \tag{3.28}$$

Although the misclassification rate can be used here, it is not sensitive enough to construct good classification trees [46, Chapter 8]. A common loss function to use for constructing trees is the Gini index.

$$G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk}) \tag{3.29}$$

where we measure the total variance across each class k. A useful property of the Gini index is that it only takes on small values when all the p_{mk}'s are close to 0 or 1. In the case of classification trees, this would require each region $\{R_1, R_2, \ldots, R_J\}$ to mostly contain observations of the same class. Since a classification tree can have an arbitrary number of splits, one class may be the majority class across multiple regions.

This brings us to the question of how to best split the predictor space into J so as to minimize the training loss. Unfortunately, for most real supervised learning problems, the predictor space is large and there are many training data points. Thus it is often not computationally feasible to consider all the possible ways to split the data into J rectangular regions. A standard method for building a tree is to use the recursive binary split algorithm. All observations in the training data start off at the top of the tree. We make the split along a predictor j at cutoff s

$$R_1(j, s) = \{\mathbf{x} | x_j < s\} \tag{3.30}$$

$$R_2(j, s) = \{\mathbf{x} | x_j \geq s\} \tag{3.31}$$

minimising

$$\sum_{i: x_i \in R_1(j,s)} Loss(y_i - \hat{y}_{R_1}) + \sum_{i: x_i \in R_2(j,s)} Loss(y_i - \hat{y}_{R_2}) \tag{3.32}$$

where $\hat{y}_{R_j}$ is the most common class in region j. The algorithm recursively splits these regions using the same rule, until a stopping condition is met (*e.g.* sufficient splits have

been made). Although the recursive binary split algorithm is fast, it only ever finds the best split at each step. It does not look ahead into subsequent steps to see if forgoing the best split at one step could enable more accurate splits later on. This is a practical example to show that it is not always feasible to find the optimal learner (Theorem 3.2) due to computational constraints.

A major advantage of classification trees is that they are simple to explain and follow. A picture of a decision tree is easier for non-mathematicians to understand and interpret, compared to many other models in machine learning (*e.g.* regularised logistic regression). Some even argue that decision trees resemble human decision processes and hence are more intuitive [46, Chapter 8]. Unfortunately, classification trees are prone to overfitting and produce highly variable results. Hence, they are inaccurate predictive models compared to other machine learning techniques. This motivates the random forest algorithm which aims to overcome this major limitation in classification trees.

3.3.4.2 Random Forests

As discussed in Section 3.3.4.1, a major limitation of classification trees is they tend to overfit and suffer from high variance. This means that small perturbations in the training data can result in radically different trees, thereby producing inconsistent predictions. Consider a set of n independent random variables $\{X_1, X_2, \ldots, X_n\}$ with variance σ. It can be shown that the average of the values, $\bar{X}$ has variance $\frac{\sigma}{n}$ [46, Chapter 8]. Hence, one way to reduce the variance of a statistical learning method is to take many training data sets, fit a separate model on each set and predict based on a majority vote among all the fitted models.

Due to practical considerations, we typically do not have multiple training data sets. Partitioning the data into smaller sets is also not desirable, as we need to have sufficient data to train a good predictive model. Instead, in a training data set $\mathbf{T}$ with n observations, we construct a "new" training set $\mathbf{T}^*$ by taking random samples from $\mathbf{T}$ with replacement. This technique is known as bootstrapping and the "new" training data produced are called bootstrap data. We can construct B bootstrap data sets and fit a separate tree on each one. The prediction from the final model will be the most common classification among the trees.

$$\hat{f}_{bag}(x) = \underset{x}{\operatorname{argmax}} \, |\{c \in P | x \in c\}| \tag{3.33}$$

where $P = \{\hat{f}^{1*}(x), \hat{f}^{2*}(x), \ldots, \hat{f}^{B*}(x)\}$, $\hat{f}^{i*}$ is the tree fitted on the i^{th} bootstrap data set. This technique is known as bagging and offers a substantial improvement in accuracy by using hundreds/thousands of trees in the final model.

A key problem of bagging is that the ensemble of trees are highly correlated due to the common tree building procedure and the same data points appearing across the bootstrap data sets. Recall that the standard recursive binary split algorithm considers the entire predictor space and makes the split which leads to the greatest reduction in training loss (Section 3.3.4.1). When a strong predictor is present, it will typically be used in the top splits despite minor differences in the bootstrap data set. This causes the trees to look similar and make their predictions are highly correlated. When a strong predictor is present in the training data, bagged trees do not substantial improvement from standard classification trees as the reduction in variance would be small. The random forest algorithm improves on bagging by decorrelating the ensemble of trees. This is achieved by altering the tree building process. Instead of considering the entire predictor space, only a randomly sampled subset of the p predictors may be considered when making each split. On average, $\frac{p-m}{p}$ if the splits would not even be able to consider the strong predictor, thereby making the trees more diverse [46, Chapter 8]. Overall, the random forest method improves on classification trees by reducing the model variance, even when a few strong predictors are present in the training data.

3.3.5 MARS: Multivariate Adaptive Regression Splines

Multivariate adaptive regression splines (MARS) is a supervised technique that was initially designed for regression. MARS is a non-parametric technique because unlike logistic regression, its predictors do not take a predetermined form in the model. MARS has been shown to be well-suited for high dimensional problems where the number of predictors is large [48, Chapter 9]. Consider some training data $\mathbf{T} = \{(y_i, \mathbf{x}_i)\},\ i \in \{1, 2, \ldots, N\},\ y \in \{C_1, C_2, \ldots, C_k\},\ \mathbf{x} \in \mathbb{R}^p$. The MARS model has the form

$$\hat{f}(\mathbf{x}) = \sum_{i=1}^{k} c_i B_i(\mathbf{x}) \tag{3.34}$$

for some constants $c_i \in \mathbb{R}$,$\forall i \in \{1, 2, \ldots, k\}$ and some basis functions B. Each basis function $B_i(x)$ is either a constant, a hinge function or a product of multiple hinge functions. A hinge function has the general form

$$max(0, x_i - t) \text{ or } max(0, t - x_i) \tag{3.35}$$

where x_i is a predictor in training data, $t \in \mathbb{R}$. t is known as the knot in a hinge function. We then form a collection of basis function pairs for each predictor x_j, using their N observed values x_{ij} as knots

$$C = \{max(0, x_i - t) \ , \ max(0, t - x_i)\} \tag{3.36}$$

where $i \in \{1, 2, \ldots, p\}$ and $t \in \{x_{1j}, x_{2j}, \ldots, x_{Nj}\}$.

Assuming that observed values at each predictor are unique, there would be Np pairs of basis functions. A common algorithm for constructing a MARS model is the forward pass algorithm. Forward pass starts with a null model containing only an intercept term, $h_{\mathbf{x}} = 1$. The pairs of basis functions in C are candidate functions which could be added to the model M. At each step, we add to the model M a term of the form

$$\beta_{M+1} h_l(\mathbf{x}) \times max(x_j - t) + \beta_{M+2} h_l(\mathbf{x}) \times max(t - x_j), h_l \in M \tag{3.37}$$

which produces the greatest reduction in training loss. The constant coefficients β_{M+1}, β_{M+2} are estimated via ordinary least squares. This step is repeated until the reduction in training loss is below a designated threshold or the maximum number of model terms has been reached.

Although MARS was originally designed to be a regression technique, it can be generalised for classification problems. One technique is to fit an initial MARS model and extract the basis functions [54]. We can then fit a generalised linear model relating the response to the basis functions. For example, in a classification problem with 2 classes, we can fit a logistic function which constrains the output probability to be within $[0, 1]$. This is similar to the technique explained in Section 3.3.3. There is also a variant of MARS called PolyMARS which uses the multinomial logistic model framework to conduct classification with > 2 classes [46, Chapter 9].

3.3.6 Discriminant Analysis

Discriminant analysis are a class of supervised learning techniques based on the Baye's Classifier [46, Chapter 4]. Under a geometric interpretation, discriminant analysis separates data using decision boundaries. A decision boundary partitions the predictor space into non-overlapping sets, one for each class. Predictions are made according to which partition the input predictor vector is locate within. First we will cover the Bayes' Classifier.

3.3.6.1 Bayes' Classifier

The Bayes's classifier attempts to minimize the probability of misclassification by using Bayes' theorem [46, Chapter 4]. Consider a general classification problem with K classes. Let $f_k(\mathbf{x})$ be the probability that the input feature vector $\mathbf{x}$ is of class k. According to Bayes' theorem,

$$P(Y = k|X = x) = \frac{\pi_k f_k(\mathbf{x})}{\sum_{i=1}^{K} \pi_i f_i(\mathbf{x})} \tag{3.38}$$

where Y is the response, X is the feature space and π_i $i \in 1, 2, \ldots, K$ is the prior probability that a randomly chosen observation of class K. (In Bayesian statistics, the prior probability refers to the researcher's prior belief about the distribution of an uncertain quantity.) The Bayes' classifier assigns a class to the predictor vector $\mathbf{x}$ by finding the class k which maximizes $f_k(\mathbf{x})$. For most real world problems, $f_k(\mathbf{x})$ is unknown and assigning prior probabilities π_k for each class can be tricky. Discriminant analysis methods use estimates of these quantities and plug them back into Bayes' theorem in order to make predictions.

3.3.6.2 Linear Discriminant Analysis

Consider the general case of classification with K classes and a p-dimension predictor space. Denote the response variable as y and the predictors as $\mathbf{x}$. Linear discriminant analysis (LDA) assumes

$$\mathbf{x_k} = (\mathbf{x}|y = k) \sim N(\boldsymbol{\mu_k}, \Sigma) \tag{3.39}$$

where N is the multivariate normal distribution. This means each class is assumed to have the same covariance matrix of the predictors Σ. Let $f_k(\mathbf{x}) \coloneqq P(Y = k|\mathbf{X} = \mathbf{x})$, then

$$f_k(\mathbf{x}) = \frac{1}{(2\pi)^{p/2}\sqrt{|\Sigma|}} exp\Big(-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu_k})^T \Sigma^{-1}(\mathbf{x} - \boldsymbol{\mu_k})\Big) \tag{3.40}$$

We can substitute this result back into Baye's classifier (Equation (3.38)), rearrange terms and take the log to get obtain the discriminant function

$$\delta_k(\mathbf{x}) = \mathbf{x}^T\Sigma^{-1}\boldsymbol{\mu_k} - \frac{1}{2}\boldsymbol{\mu_k}^T\Sigma^{-1}\boldsymbol{\mu_k}^T + log(\pi_k) \tag{3.41}$$

The discriminant function is a linear combination of predictors which can classify data into different groups. In LDA, we use the estimates

$$\hat{\boldsymbol{\mu}}_{\boldsymbol{k}} = \frac{1}{n_k}\sum_{i:y_i=k} \mathbf{x_i} \tag{3.42}$$

$$\hat{\Sigma} = \frac{1}{n-K}\sum_{k=1}^{K}\sum_{i:y_i=k}(\mathbf{x_i} - \hat{\boldsymbol{\mu}}_{\boldsymbol{k}})(\mathbf{x_i} - \hat{\boldsymbol{\mu}}_{\boldsymbol{k}})^T \tag{3.43}$$

$$\hat{\pi_k} = \frac{n_k}{n} \tag{3.44}$$

where n_k is the number of observations in class k and n is the total number of observations in the training data set. Notice that a key assumption in LDA is that the covariance matrix Σ is constant across all the groups. LDA makes predictions by finding the class k which maximizes the value of the discriminant function, *i.e.*

$$\underset{k}{\text{argmax}}\{\delta_k(\mathbf{x})\} \tag{3.45}$$

where $\mathbf{x}$ is the input predictor vector.

Geometrically, the discriminant function corresponds to a decision boundary in the predictor space [55, Chapter 12]. New data is classified using its location relative to the

decision boundaries. As the discriminant function is a linear function of $\mathbf{x}$, the corresponding decision boundaries are also linear. Hence, the name of this technique is linear discriminant analysis.

3.3.6.3 Quadratic Discriminant Analysis

Recall that LDA assumes that all groups share the same covariance matrix for the predictors. Quadratic discriminant analysis (QDA) relaxes this assumption so that each group can have its own covariance matrix.

$$\mathbf{x_k} = (\mathbf{x}|y = k) \sim N(\boldsymbol{\mu_k}, \Sigma_k) \tag{3.46}$$

Using a similar derivation as LDA, we get the QDA discriminant function

$$\delta_k(\mathbf{x}) = \mathbf{x}^T\Sigma_k^{-1}\boldsymbol{\mu_k} - \frac{1}{2}\boldsymbol{\mu_k}^T\Sigma_k^{-1}\boldsymbol{\mu_k}^T - \frac{1}{2}\mathbf{x}^T\Sigma_k^{-1}\mathbf{x} - \frac{1}{2}log|\Sigma_k| + log(\pi_k) \tag{3.47}$$

Σ_k is estimated using the observed covariance matrix for the observations of class k.As before, predictions are made by finding the class which maximises δ for the input vector. The discriminant function is a quadratic function of $\mathbf{x}$, hence the name quadratic discriminant analysis. Geometrically, the discriminant function produces a quadratically curvilinear decision boundary in the predictor space [55, Chapter 13].

Comparing LDA and QDA, QDA is more flexible at the expense of having higher variance. For k class problem with a p-dimension predictor space, LDA estimates $\frac{p(p+1)}{2}$ parameters for Σ whilst QDA estimates $\frac{kp(p+1)}{2}$ parameters for all the Σ's. Having more parameters enables QDA to fit the training data more effectively, whilst having a greater risk of overfitting. Overall, QDA tends to outperform LDA when the different classes of data points cannot be sufficiently separated by a set of linear decision boundaries.

3.3.6.4 Regularised Discriminant Analysis

Recall that discriminant analysis refer to class of techniques which attempt to minimise the misclassification rate using decision boundaries. LDA separates data using linear boundaries (*i.e.* hyperplanes) and the more flexible QDA uses quadratic boundaries. For

some classification problems, a better classifier could be achieved using decision boundaries that are somewhere in between a hyperplane and a quadratic surface. Regularised discriminant analysis (RDA) was designed to be a bridge between LDA and QDA. RDA uses the following covariance matrix

$$\Sigma_l(\lambda) = \lambda\Sigma_l + (1-\lambda)\Sigma \tag{3.48}$$

where $\lambda \in [0,1]$ is a tuning parameter,Σ_l is the covariance matrix for class l and Σ is the pooled covariance matrix across all classes. Σ is then allowed to morph between the observed covariance matrix across all classes and another matrix which assumes that all the predictors are pairwise independent.

$$\Sigma(\gamma) = \gamma\Sigma + (1-\gamma)\sigma^2 I \tag{3.49}$$

where σ^2 is the common variance across all predictors and $\gamma \in [0,1]$ is a tuning parameter. When $\lambda = 0, \gamma = 1$ the model is equivalent to LDA and when $\lambda = 1, \gamma = 1$ the model is equivalent to QDA. When working with high dimension data, it is difficult to visualise what kind of decision boundary would be suitable. Since RDA is computationally fast, the standard practice is to tune over different values of λ and γ, and select the best set of hyperparameters using cross validation.

3.4 Model Assessment

Most machine learning classifiers can produce a soft classification, *i.e.* it assigns a probability that an observation is of any particular class. New data are predicted to be of a particular class based on a chosen cutoff point. Consider a classification problem with 2 classes (success or fail) and some learner g. Since g can provide a soft classification, we can arbitrarily set the cutoff to be 0.5. Thus a new observation $\mathbf{x}$ is predicted to be a success if $g(\mathbf{x}) \geq 0.5$ and a failure otherwise. There are two possible outcomes for each prediction; either the prediction is correct or incorrect. We can lay out the predictive performance of our model using a confusion matrix (see Figure 3.5). A confusion matrix enables researchers to visualise how well a model's predictions matches the true response. Ideally, we want to maximize the diagonal elements (TN,TP) whilst minimizing the off-diagonal elements (FN,FP).

	True condition	
Predicted	Condition positive	Condition negative
Condition positive	True positive (TP)	False positive (FP)
Condition negative	False negative (FN)	True negative (TN)

FIGURE 3.5: A general confusion matrix.

A variety of summary statistics exist for capturing the information within confusion matrices. Two common statistics are the false positive rate (FPR) and the true positive rate (TPR):

$$FPR := \frac{FP}{FP + TN}, \tag{3.50}$$

$$TPR := \frac{TP}{TP + FN}. \tag{3.51}$$

In statistics, the TPR is also called statistical power. The 0.5 cutoff is arbitrary and we could theoretically use any cutoff $\in [0, 1]$. Each cutoff could produce a different confusion matrix and hence different FPRs and TPRs. Generally a good predictive model will have a low FPR and a high TPR. A receiver operator characteristic curve (ROC) is created by plotting the TPR against the FPR at various cutoff points. The specific algorithms used for this can be found in [56]. The ROC curve is a useful visual tool for evaluating the predictive performance of machine learning classifiers.

Figure 3.6 is an illustration of a general ROC curve. The point (0,0) represents the classifier never making a positive prediction (*i.e.* cutoff is 1). When no positive predictions are made, the FPR is 0 at the expense of having a TPR of 0. When the classifier always returns a positive result (*i.e.* cutoff is 0), all positive responses will be detected (TPR = 1). This comes at the cost of misclassifying all negative cases, thereby producing a FPR of 1. This strategy of classifying all cases as positive corresponds to point (1,1) on the ROC curve. The diagonal line represents a classifier that is making entirely random predictions. Accurate models have ROC curve that is close to the top left hand corner because this indicates that the model has a high TPR and a low FPR. We can measure how accurately a model discriminates between two classes using the area under the curve (AUC). A purely random model has an AUC of 0.5 and the maximum AUC for any model is 1. AUC is a useful metric for comparing the predictive accuracy of different machine learning classifiers.

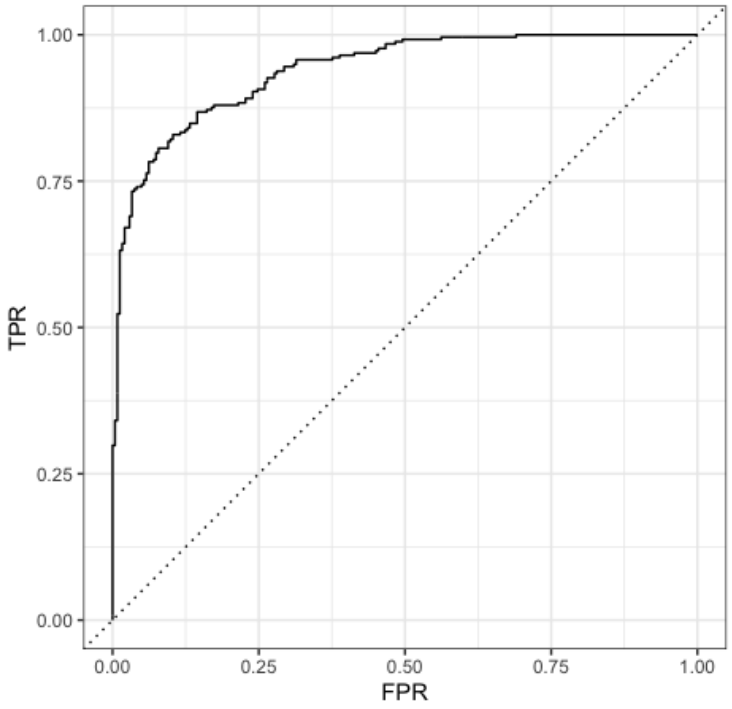

FIGURE 3.6: An ROC curve produced using the `two_class_example` in the yardstick R package.

3.5 Interpretable Machine Learning

Supervised learning uses labelled training data to create a predictive model that can map a feature vector $\mathbf{x}$ to some response variable y. Different machine learning methods have their own way of fitting a model to the training data. The predictive accuracy of the final model can then be assessed using ROC and AUC (Section 3.4). However, a model's ability to accurately predict future observations is not the only concern of statistical modelling. Another important consideration is understanding the cause of a model's decision. This involves understanding why a specific model was fitted and how the predictions are made. This is known as model interpretability. In general, model interpretability is important for the following reasons.

1. Without model interpretability, our models are essentially "black boxes" where there is no clear connection between the input data and the predicted response. Blindly following these models can be dangerous because the model may have learnt inappropriate patterns from the training data which do not generalise to real world scenarios. For example, suppose we were designing computer vision software to classify different breeds of cattle using photographs. We would want to ensure that the classifier is making predictions based on the appearance of the individual beasts rather than unrelated objects in the background (*e.g.* a tractor, the colour of the grass).

2. Model interpretability can enhance our understanding of the research problem and our data. This can help us know when a model may fail [57, Chapter 2].

3. Sometimes the predictions of a model are used for impactful decisions where an explanation is desirable or even required. For example, a bank may use supervised machine learning to predict whether a client will default on a loan. When a loan is declined, the client may wish to know why so that they can improve and be successful in future applications.

Recall that supervised learning tries to estimate a model f which captures the underlying functional relationship between the predictors $\mathbf{x}$ and the response y. Hence, another way of framing model interpretability is understanding the effect that individual predictors have on the predicted response. The more a model relies on a particular predictor to output a response, the more important that predictor is for the model. This concept is known as variable importance. Some machine learning models have their own model-specific metric for variable importance. For example, in general linear models, we can measure a predictor's importance using the absolute value of its t-statistic [58]. In classification trees, we can look at all the internal nodes where a predictor of interest x was chosen for a split. The importance of x is the total reduction of misclassification rate produced by these internal nodes. Although model-specific metrics are useful for interpreting their respective models, there are two key limitations with this approach.

1. Model-specific metrics are not comparable across different machine learning methods.

2. Many machine learning methods lack standard ways of measuring variable importance (*e.g.* k-nearest neighbors) [57, Chapter 4].

This makes it difficult to compare models produced by a range of different machine learning methods. Thus there is a need for a standard method of measuring variable importance which is model agnostic. Variable importance metrics is an emerging area of research in data science and machine learning. It is part of an ongoing effort in developing interpretable machine learning tools. We will now cover several model agnostic methods for determining variable importance which can help us understand our predictive models.

3.5.1 Partial Dependence Plots

A partial dependence plot shows the marginal effect of one or two chosen predictors on the predicted response of a machine learning model [58]. Consider a model $\hat{f} : \mathbf{x} \longrightarrow y$, $\forall \mathbf{x} \in X$, where X is a p dimensional feature space and y is some response variable. Let P be the full set of predictors of X. We select some predictors of interest $S \subseteq P$ which corresponds to a feature vector $\mathbf{x}_s$. In practice, researchers typically pick

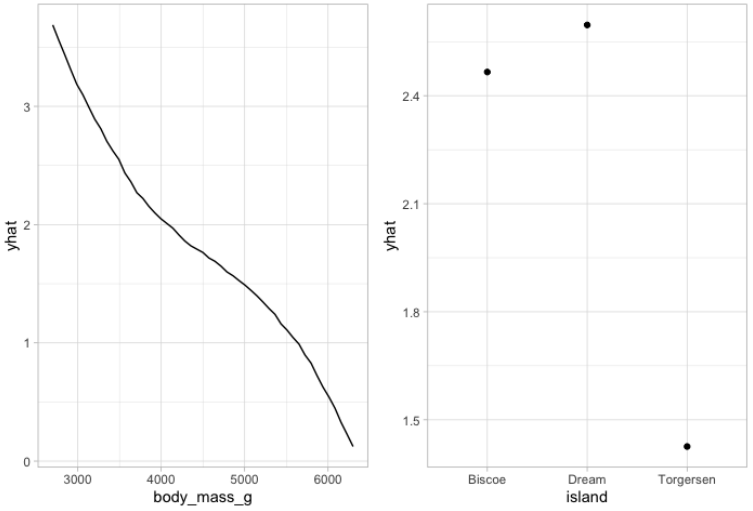

FIGURE 3.7: Partial dependence plots for a numeric and categorical predictor in a logistic regression model using the Palmer penguins data set [1]. The predicted response "yhat" is the predicted log odds of a penguin belonging to the Adelie species.

two predictors at most although choosing only one predictor is not uncommon. The set of remaining predictors is $C \subseteq P \setminus S$ with a corresponding feature vector $\mathbf{x}_c$. The feature vectors $\mathbf{x}_s$ and $\mathbf{x}_c$ are orthogonal and together make up the feature space X (*i.e.* $\mathbf{x}_s + \mathbf{x}_c = \mathbf{x} \in X$, $\forall \mathbf{x}_s, \mathbf{x}_c$).

We define the partial dependence of the response on $\mathbf{x}_s$

$$f_{\mathbf{x}_s}(\mathbf{x}_s) \coloneqq E_{\mathbf{x}_c}[\hat{f}(\mathbf{x}_s, \mathbf{x}_c)] = \int_{\mathbf{x}_c} \hat{f}(\mathbf{x}_s, \mathbf{x}_c) dP(\mathbf{x}_c) \tag{3.52}$$

where $\hat{f}(x_s, x_c)$ is the prediction for feature vector $\mathbf{x} = \mathbf{x}_s + \mathbf{x}_c$ and $P(x_c)$ is the marginal probability of $\mathbf{x}_c$. We estimate $f_{x_s}(x_s)$ using

$$\hat{f}_{\mathbf{x}_s}(\mathbf{x}_s) = \frac{1}{n} \sum_{i=1}^{n} \hat{f}(\mathbf{x}_s, \mathbf{x}_c^i) \tag{3.53}$$

where n is the number of training data points and $\mathbf{x}_c^i$ are the values of the non-selected predictors on the i^{th} data point. This estimates the effect of $\mathbf{x}_s$ on the predicted response by averaging out the effects of $\mathbf{x}_c$. This technique is known as Monte Carlo integration [59]. Pseudo code for generating the partial dependence function can be found in [58].

So how do we use partial dependence plots? Figure 3.7 are examples of partial dependence plots for numeric and categorical predictors respectively. The underlying model

is a logistic regression model predicting the log odds that a penguin is of the Adelie species. Recall that the purpose of a partial dependence plot is to show the marginal effect that a predictor has upon the predicted response. The negative relationship between "yhat" and body mass means that when we average out the effects of all other predictors, the heavier the penguin the less likely that it belongs to the Adelie species. The partial dependence plot for the island variable shows that Adelie penguins are most common on Dream and Biscoe islands and are far less common on Torgensen island. A flat partial dependence plot would indicate that a predictor has little effect on the predicted outcome. This would suggest that the predictor has low importance.

A key advantage of partial dependence plots is that they are an intuitive tool for understanding machine learning models. The partial dependence plot of a predictor x_p shows the average prediction when we force all the training data to take a particular value of x_p [57, Chapter 5].

A key assumption of partial dependence plots is that there is no correlation between the predictor of interest $\mathbf{x}_s$ and the rest of the predictors $\mathbf{x}_c$. Should $\mathbf{x}_s$ and $\mathbf{x}_c$ be highly correlated then the pdp algorithm will consider unrealistic data points when computing the partial dependence function. For example, suppose we have a model to predict the lung capacity of patients using their height and weight. The training data has ranges 130-185cm for height and 40-85kg for weight. To compute the value of the partial dependence function of height at 180cm, the algorithm has to average over the marginal distribution of weight which includes 40kg. However, a 180kg individual that weighs only 40kg is unrealistic so the algorithm is extrapolating at this point. Heterogeneous effects may also be obscured by partial dependence plots. Suppose for half the data, $\mathbf{x}_s$ has a positive relationship with the predicted outcome but has a negative relationship for the remaining half. The output partial dependence plot will be relatively flat because pdp algorithm looks at the average effect that a predictor has on the predicted response. It would be naive to conclude that $\mathbf{x}_s$ is unimportant for the model. Rather $\mathbf{x}_s$ can have a different effect on the prediction depending on the values in $\mathbf{x}_c$. In the next section, we will now explore a technique that builds on the partial dependence plot and is more sensitive to heterogeneous effects. However, the limitation of correlated predictors remain.

3.5.2 Independent Conditional Expectation

Partial dependence plots show the average relationship between selected predictors $\mathbf{x}_s$ and the predicted outcome $\hat{y}$. We will focus on the case where $|S| = 1$ (i.e. only one selected predictor at a time) because that is the standard practice. For any given training

data, the pdp for some feature $\mathbf{x}_s$ is a single line showing the average predicted response as a function of $\mathbf{x}_s$. Independent conditional expectation (ICE) plots show a single line for each observation in the training data [57, Chapter 5]. Each line depicts the predicted response as a function of $\mathbf{x}_s$ conditioned on an observed $\mathbf{x}_c$ [60]. For any fixed value of $\mathbf{x}_s$, there are up to N different values of $\mathbf{x}_c$ across the N observations in the training data. The lines of the ICE plot show the different conditional relationships between $\mathbf{x}_s$ and $\hat{y}$ at different values of $\mathbf{x}_c$. Hence, ICE plots can provide more insight than partial dependence plots when heterogeneous relationships are present. The partial dependence plot of a predictor $\mathbf{x}_s$ is the average of all N lines in the corresponding ICE plot.

The ICE curve for observation i is defined as

$$\hat{ICE}^{(i)}_{\hat{f},s}(\mathbf{x}_s) \coloneqq \hat{f}(\mathbf{x}_s, \mathbf{x}_c^{(i)}) \tag{3.54}$$

where $\mathbf{x}_s$ is a predictor of interest and $\mathbf{x}_c^{(i)}$ are the values of the remaining predictors for observation i [59]. The key difference between Equation (3.54) and Equation (3.52) is that the ice curve is conditioned on a fixed $\mathbf{x}_c$ whilst the pdp averages over all values of $\mathbf{x}_c$. The pseudo code for computing ICE curves can be found in [60]. In summary, the process involves permuting $\mathbf{x}_s$ with a grid of values $\mathbf{g} = (g_1, g_2, \ldots, g_n)$. This generates a new set of n feature vectors $\mathbf{g}_j = (g_j, \mathbf{x}_c^i)$, $j \in \{1, 2, \ldots, n\}$. For each $\mathbf{g}_j$, we use our machine learning model to output a predicted value y_j. The ICE curve for observation i consists of the set of points $\{y_j, g_j\}$, $j \in \{1, 2, \ldots, n\}$. The ICE plot is produced by computing the ICE curve for every observation in the training data.

Let's return to our illustrative example of using a logistic regression model to predict whether a penguin is of the Adelie species in the Palmer penguins data set [1]. The left panel of Figure 3.8 is an ICE plot for bill length. All the ICE curves show that past a certain point, increases in bill length are associated with a higher predicted log odds for a penguin being an Adelie penguin. The precise point at which bill length starts to affect the prediction varies across the different observations in the training data. The red line corresponds to the partial dependence plot (average of all the ICE curves), which shows that on average, longer beak lengths are associated with higher log odds for an Adelie classification. This shows that there is no heterogeneous effect between beak length and the predicted outcome of the logistic model. The right panel of Figure 3.8 is an ICE plot for sex. The faint lines shows that sex does decrease the predicted log odds for some observations in the training data. The two solid black horizontal lines show that for the vast majority of the observations, sex does not change the prediction. Hence, the red line (partial dependence plot) for sex is flat. This makes sense because we wouldn't

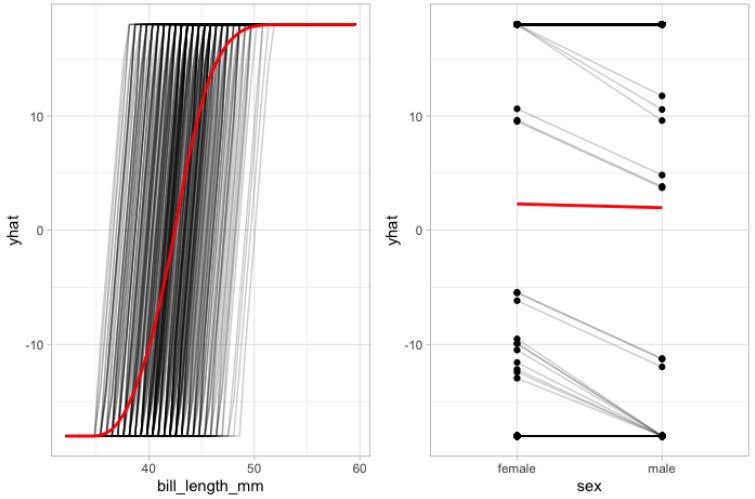

FIGURE 3.8: ICE plots for bill length (mm) and sex for a logistic regression model fitted on the Palmer penguins data set [1]. The predicted response "yhat" is the predicted log odds of a penguin belonging to the Adelie species. Each faint line is the ICE curve for an individual observation. The black lines are produced by multiple overlapping faint lines. The red line is the average of all the ICE curves (i.e. the partial dependence plot).

expect to be able to predict whether a penguin is of the Adelie species based on its gender unless there was a substantial sex bias in the training data.

The main advantage of ICE is that it can unravel heterogeneous relationships between predictors and the predicted outcome. It is arguably more intuitive than partial dependence plots because each ICE curve depicts how the prediction would change we as vary a selected predictor [57, Chapter 5]. However, ICE curves can become overcrowded if the training data has many observations. In those instances, researchers could down-sample the data for the purposes of computing ICE plots. Similar to partial dependence plots, the ICE method still assumes that the predictor of interest is independent of the remaining predictors. If there is a strong correlation, some points in the ICE curves may be invalid because the ICE algorithm considered invalid data points.

3.5.3 FIRM Method

We have now explored two visual tools (ICE and pdp) for understanding how predictors affect the predicted outcome for any supervised machine learning model (Section 3.5.1, Section 3.5.2). We will now discuss a model agnostic method for quantifying variable importance which can improve model interpretability and assist model comparison. Recall from Section 3.5.1 that a partial dependence function shows the marginal effect of a selected predictor on a model's predicted response. The feature importance ranking measure (FIRM) quantifies the variable importance using the flatness of the partial

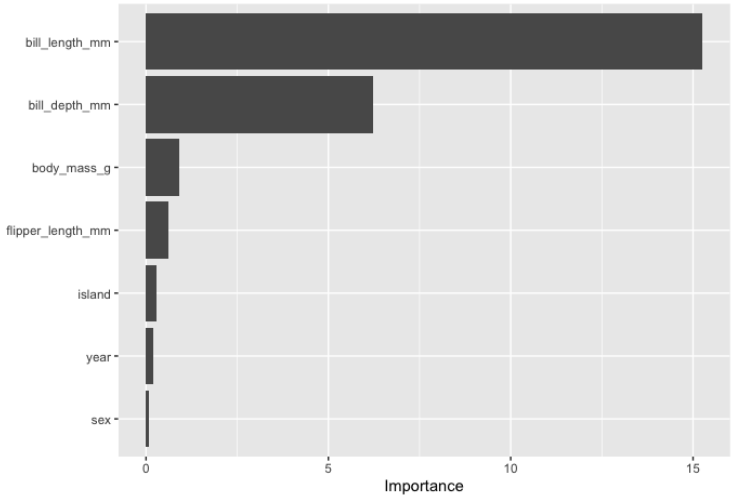

FIGURE 3.9: A bar chart of variable importance (FIRM method) for a logistic regression model fitted on the Palmer penguins data set [1]. The model predicts the log odds of a penguin being of the Adelie species.

dependence function. The logic is that a predictor is important in a model if small perturbations in the predictor produce large changes in the predicted response. Hence, an important predictor is expected to have a variable pdp whilst an unimportant predictor would have a flat pdp.

For continuous variables, the FIRM for a predictor $\mathbf{x}_s$ is defined as

$$i(\mathbf{x_s}) \coloneqq \sqrt{\frac{\mathbf{1}}{\mathbf{k-1}} \sum_{\mathbf{i=1}}^{\mathbf{k}} \left(\mathbf{f_{x_s}(x_{si})} - \frac{\mathbf{1}}{\mathbf{k}} \sum_{\mathbf{i=1}}^{\mathbf{k}} \mathbf{f_{x_s}(x_{si})} \right)^{\mathbf{2}}} \tag{3.55}$$

where f_{x_s} is the partial dependence function of $\mathbf{x}_s$ and x_{si} is the value of $\mathbf{x}_s$ in the i^{th} observation in the training data. Inspecting Equation (3.55), we see that the FIRM importance of $\mathbf{x}_s$ is the standard deviation of the values in its partial dependence plot.

For categorical variables, the FIRM for a predictor $\mathbf{x}_s$ is defined as

$$i(\mathbf{x_s}) \coloneqq \frac{\mathbf{1}}{\mathbf{4}} \left(\mathbf{max_i}\left(\mathbf{f_{x_s}(x_{si})} \right) - \mathbf{min_i}\left(\mathbf{f_{x_s}(x_{si})} \right) \right) \tag{3.56}$$

This corresponds to a standard deviation measurement when sample sizes are small. Continuing with our Palmer penguin example, we compute the FIRM for all the predictors in our logistic regression model. Figure 3.9 is a bar chart of the FIRM scores. Recall from Figure 3.8 that bill length (mm) has a steep partial dependence plot which

has values ranging from -18 to 18. Hence, bill length has a high FIRM score. The partial dependence plot for body mass (Figure 3.7) was less variable (range 0 to 3.5) so it had a lower FIRM score. The partial dependence plot for sex (Figure 3.8) was fairly flat so its FIRM score was approximately 0.

3.6 Conclusion

This chapter explained how to use machine learning to explore data, create predictive models and interpret models. Machine learning can be broken into supervised and unsupervised techniques. Unsupervised learning clusters the training data and is used for exploratory data analysis. Supervised learning uses the training data to fit a predictive model that maps new feature vectors to the response. Since our project is on detecting selective sweeps, we have focused on several supervised techniques for classification. Classification models can be assessed on their predictive accuracy using ROC and AUC. Besides using models for prediction, researchers may also wish to understand why a specific model was fitted and how the predictions are generated. We explored partial dependence plots and ICE plots which provide visualisations of how predictors affect the predicted response. The FIRM score quantifies variable importance thereby revealing which predictors are most impactful for a particular predictive model. Having covered the biological and statistical backgrounds, the next chapter will present current applications of machine learning to sweep detection.

Chapter 4

Current Machine Learning Approaches to Sweep Detection

4.1 Introduction

In this project, we are interested in using machine learning techniques to detect regions of selection using genetic data. Chapter 2 was a biological introduction which covered the nature of DNA, evolution via natural selection and some basic concepts in population genetics. Of particular interest is the concept of selective sweeps which are molecular signatures produced by selected genetic variants. Chapter 3 introduced machine learning, a broad suite of powerful statistical tools used for understanding data. Machine learning has a range of successful applications from sorting out spam emails to computer vision [49, Chapter 5]. We will now apply various machine learning methods to the problem of detecting selective sweeps and assess their effectiveness.

4.2 Why Detect Sweeps?

Recall from Section 2.3 that evolution is the change in the heritable characteristics of a population over time. Heritable characteristics are encoded in the DNA of individuals. Selective sweeps are a particular kind of selection where a new beneficial mutation increases its frequency and spreads throughout the population. We will now outline some reasons why population geneticists want tools for detecting selective sweeps.

1. Genomes are typically very long and it is not clear which regions researchers should focus on (Section 2.1.1). Accurate detection of selective sweeps will enable researchers to

precisely identify adaptive regions in the genome. Further bioinformatic and biochemical analyses could be done on these regions of interest in order to unravel the precise molecular mechanisms which confer selective advantage. For example, a researcher may examine the gene products of a particular locus and investigate how these may impact an organism's phenotype. This can enhance our understanding of how different populations have adapted to past environmental challenges. This information can be used to build evolutionary models to make predictions regarding how populations may respond to future, environmental stress. This has applications in microbiology whereby researchers may study how bacteria colonies evolve under various environmental conditions. This can provide insights for managing antibiotic resistance and using bacteria to produce biofuels and medicines [61]. There are also applications in conservation biology for predicting the effects of introducing new individuals to save inbred populations (genetic rescue) [62].

2. Researchers may want to investigate the location of selective sweeps across the genome. It would be interesting to see if particular areas of the genome tend to generate more sweeps and consider why this may be the case. The patterns we identify can inform our understanding of how evolution acts upon the genome across successive generations.

4.3 Current Machine Learning Approaches

Given the popularity of machine learning in recent years, it should be no surprise that researchers have already applied various machine learning tools to the problem of sweep detection. We now consider several current applications of machine learning for detecting selective sweeps.

4.3.1 Boosting Classifiers

Boosting is an ensemble based machine learning algorithm that has been applied to detecting selective sweeps [63]. The model is trained using 1000 simulated genome regions of 40kb where half were neutral and half were hard sweeps (Section 2.3.4.1). Simulations were produced using coalescent simulation software. Each simulation contained 10 sequences, represented by a binary genome matrix. Each matrix was subdivided into 20 adjacent, non-overlapping segments of 2kb. For predictors, they computed summary statistics on each segment. Hence, for a model with j summary statistics and k subdivided segments, there would be $j \times k$ predictors. Lin *et.al.*, they used a variety of

statistics including Tajima's D, Fay and Wu's H and the Waterson estimator. Multiple statistics were used in order to capture as much of the information in the genome matrices as possible.

The general intuition of boosting is to use an ensemble of weaker learners together to create a single strong learner. Boosting is a general technique that can be applied to a range of machine learning classifiers (*e.g.* classification trees) but in the case of Lin *et.al.*, they used boosting to enhance logistic regression models. Consider a training set with n observations and response y. The boosted model starts with a null model $\hat{f}^{[0]}$

$$\hat{f}^{[0]} = \underset{c}{\operatorname{argmax}} \Big\{ \sum_{i=1}^{n} Loss(Y_i, c) \Big\} \tag{4.1}$$

The null model predicts the most common class in the training set. A logistic regression model $\hat{g}^{[1]}$ is fitted using a randomly selected predictor from the feature vector $\mathbf{x}$. This new model is

$$\hat{f}^{[1]} = \hat{f}^{[0]} + v\hat{g}^{[1]} \tag{4.2}$$

for some learning rate $v \in [0, 1]$ [64]. The hyperparameter v controls how much the model can change between each iteration. The training data points that are misclassified by $\hat{f}^{[1]}$ are given greater weight in the next iteration of fitting another logistic model. The logic is that with each iteration our model learns to correctly classify more and more of the training data. This process is repeated until m steps are made.

$$\hat{f}^{[m]} = \hat{f}^{[m-1]} + v\hat{g}^{[m]} \tag{4.3}$$

For use with empirical data, the genomes must be broken down into regions of 40kb. The sequences are converted into genome matrices and subdivided into 20 segments of 2kb. The same summary statistics can be computed to obtain a feature vector which can be given to the final model $\hat{f}^{[m]}$ for prediction.

The final model was evaluated using test data of different selection strengths and fixation times. Accuracy declined for lower selection strengths and more distant fixation times since these produce weaker selection patterns. This is a problem inherent in the data and

should affect all methods despite the high accuracy observed in this setting. This model achieved an accuracy of $> 98\%$ in most scenarios. Even in the most difficult test of an old and weak sweep ($\alpha = 2Ns = 200, \tau = 2N = 0.2$) [1], the accuracy was 87.6%. The researchers compared their method with other approaches using the testing accuracy. This showed boosting to be more accurate than support vector machines (svm) and traditional population genetic tests.

Population bottlenecks are scenarios where the population size rapidly shrinks for a time and then quickly grows back. The researchers tested their method on population bottlenecks which are known to cause false positives. The final classifier consisted of two boosting models, M1 and M2. M1 was trained using neutral and hard sweeps, M2 trained with bottlenecks and hard sweeps. New data is classed as a sweep if both M1 and M2 classify it as a sweep; otherwise it is neutral. They found the misclassification of bottlenecks to be rare. However, this came at the cost of decreased power in sweep detection.

The researchers used the absolute value of the standardised coefficients in the final model to investigate variables of importance. The statistics were most effective when the segment is close to the selected mutation. They found θ_π[2] to be consistently useful throughout the test sets. θ_w[3] distinguished recent sweeps from bottlenecks. Since SVM's do not have coefficients, they could not compare variable importance between methods.

4.3.2 S/HIC: Soft/Hard Inference through Classification

The S/HIC method designed by Kern and Schrider is a supervised machine learning method for inferring the location of soft and hard selective sweeps within genomes [65]. Similar to random forests (Section 3.3.4.2, this method uses an ensemble of classification trees (Section 3.3.4.1)to mitigate overfitting. S/HIC uses the "Extremely Randomized Trees"(ERT) algorithm which has 3 tuning parameters K,M,N [66]. ERT generates a predictive model with M trees. Each tree starts at a single trunk which consists of all the training data. To generate a split, ERT algorithm randomly selects K non constant predictors without replacement. For each of the K predictors, a randomly split point is generated using a uniform distribution between the maximum and minimum values for that predictor. The algorithm then selects the split which gives the highest value of the score function (a measure of the training accuracy). This is equivalent to minimising a loss function. By splitting the data at random cutoff points for the predictors, the ERT algorithm decorrelates the trees. The algorithm recursively splits the data until

[1] N is effective population size, s is the selection coefficient
[2] A component of Tajima's D. See Section 2.5.1.3
[3] Another component of Tajima's D

either the minimum sample size N for making a split is reached, all the predictors are constant or further iterations will no longer alter the tree. Classification is done using a majority vote on the ensemble of K trees. Similar to random forests, the ERT algorithm reduces the variability of standard classification trees, leading to more accurate predictive models.

Similar to Section 4.3.1, the researchers used a coalescent simulation program to generate genomes under a suite of demographies and selection coefficients. In general, each simulation is a binary genome matrix G which is subdivided into k adjacent, non-overlapping blocks.

Denote the i^{th} subwindow as the matrix W_i. Recall that a summary statistic is a function which maps some data to a real number.

$$f : W_i \to A \subseteq \mathbb{R} \ , \ \forall i \in \{1, 2, \ldots, K\}$$

We can apply the same summary statistic to each subwindow. Thus for any summary statistic f we get the vector,

$$\boldsymbol{f} = \{f(w_1), f(w_2), \ldots, f(w_K)\}$$

The S/HIC method is interested in picking up how various summary statistics change across the genome. This is more informative than computing each summary statistic once across the whole simulated region. Hence, the values of each summary statistic are normalised by the following

$$\boldsymbol{f} = \frac{1}{\sum_{i=1}^{K} f(w_K)} \{f(w_1), f(w_2), \ldots, f(w_K)\}$$

S/HIC used a range of summary statistics designed for detecting selection, including Tajima's D, Fay and Wu's H, ω_{max} and Kelly's Z_{nS}. For a model with p summary statistics, we can construct a feature vector $\boldsymbol{x}$ of $p \times K$ dimensions. The response variable y is a categorical variable with three classes, namely neutral, hard and soft sweeps. The S/HIC model is then fitted onto this training data with the ERT algorithm. To tune the model, different combinations of predictors can be removed from the model to observe the change in the overall training loss. If the loss increase is below some designated threshold, the predictor is kept in the final model. Once the final model is fitted, it can be used to scan real genome data by breaking it into K blocks and computing the relevant summary statistics.

The training data consisted of 2.2Mb[4] regions which were subdivided into eleven 200kb[5] blocks. Using AUC, they showed S/HIC to be more accurate at sweep detection than other methods such as SweepFinder [67] and boosting [63]. By using testing data from various demographies not found in the training data, they showed S/HIC to be more robust to demography compared to other methods. All methods they considered had lower power when considering population bottlenecks, even when bottlenecks were included in the training data. The reason is that population bottlenecks reduce diversity and obscure the impact of selective sweeps.

4.3.3 Neural Networks Method

One of the major challenges of analysing population genomic data is the high dimensional nature of the data. In its original form, each observation is a binary matrix $G_{n\times k}$ where n is the number of samples and k is the number of segregating sites/SNPs. Most machine learning classifiers cannot be trained on entire matrices as the dimensions of the input is too large. The approach used in Section 4.3.1 and Section 4.3.2 is to condense the data into a handful of summary statistics, designed to capture useful patterns in the data. There are two weaknesses with this approach.

1. Summary statistics can be confounded by unrelated demographic factors which produce similar effects on the sampled population. This is especially true when we are dealing with complex demographic models which do not follow convenient population genetic assumptions (e.g. constant population size, constant mutation rate, no interbreeding with external populations). For example, Tajima's D is confounded by population size changes which can distort the site frequency spectrum in a manner which is similar to a selective sweep.

2. Summary statistics do not capture all the information in the population genetic data. Researchers try to mitigate this problem by using a broad suite of summary statistics which are not highly correlated. A major challenge of using supervised machine learning is to construct a method that uses as much information in the input data as possible in order to maximize the predictive accuracy of the final model.

A potential technique for circumventing summary statistics is to use deep learning/neural networks, a supervised machine learning method which can handle high dimensional input [68]. Neural networks can be trained directly on the genome matrices which

[4] $2.2 \times 10^6 bases$
[5] $2 \times 10^5 bases$

should contain more information than a set of summary statistics. Hence, the neural network has the potential to learn genetic patterns associated with selection which are too complex to be captured by summary statistics. Neural networks consists of multiple connected layers of perceptrons ("neurons") which take some input values and transforms them into an output value. A perceptron is a mathematical function with output

$$f(\mathbf{w} \cdot \mathbf{x} + b) \tag{4.4}$$

where $\mathbf{x}$ is a vector of input values, $\mathbf{w}$ is a vector of weights and b is a constant [69]. f is known as an activation function and common choices include logistic and hyperbolic tangent functions.

For a binary genome matrix $G_{N\times K}$, an exchangeable neural network learns the function

$$f : \{0,1\}^{N\times K} \to P_\Theta \tag{4.5}$$

where Θ is the space of all parameters θ and P_Θ is the space of all probability distributions on Θ. For the purposes of detecting selection, P_Θ would be the pdf of having a selective sweep in a given genome matrix. Function f is a composite function of Φ, h, g,

$$f \coloneqq (h \circ g)(\Phi(x_1), ..., \Phi(x_n)) \tag{4.6}$$

where $\Phi : \{0,1\}^d \to \mathbb{R}^{d_1}$ is a convolutional neural network. i.e. Φ takes a row of the genome matrix and turns it into a vector of dimension d_1 for some choice of d_1. A convolutional neural network is a special class of neural network that is commonly used for image analysis.

$g : \mathbb{R}^{n\times d_1} \to \mathbb{R}^{d_1}$ is a symmetric function. Symmetry ensures that the ordering of the individuals in the genome matrix do not matter. This makes f an exchangeable neural network.

$h : \mathbb{R}^{d_2} \to P_\Theta$ is a fully connected neural network, for some choice of d_2. A fully connected neural network means that each perceptron receives some input from every perceptron in the previous layer.

The optimal parameters of f cannot be calculated analytically due to its mathematical complexity. Instead f is fitted using an optimization algorithm such as stochastic gradient descent [49, Chapter 2].

Although Chan *et.al.* originally used exchangeable neural networks to identify recombination hotspots[6], their method can be adapted for sweep detection by changing the response variable [68]. After training their models with data simulated from a human demographic model, they showed exchangeable neural networks to be faster and more accurate than LDhot [70]. Their method had an overall accuracy of 95%, making it superior to traditional neural network architectures. Although exchangeable neural networks have potential to be an effective method for sweep detection, it remains to be tested whether they are more accurate than summary statistic based methods. Nevertheless, neural networks have two main drawbacks.

1. Neural networks are computationally intensive to train, requiring many CPU's or preferably GPU's working in parallel to fit models in a timely manner [49, Chapter 1].

2. Neural networks are hard to interpret (*i.e.* "black box"). It is difficult to discern what patterns the model is using to make its predictions.

4.4 Gap in the Literature

Section 4.3 reviewed several machine learning approaches that have been developed for sweep detection. Although researchers have applied various machine learning methods to this problem, they often use a one classifier in isolation [7]. Researchers often justify their approach using their model's testing accuracy. However, this does not show why their specific technique should be chosen over the available range of classifiers. Besides accuracy, there are other considerations for model selection such as computational speed and model interpretability. Thus there is a knowledge gap regarding which classifier(s) should be chosen for any particular data set. There is a risk that researchers may select a method based on novelty and familiarity.

The first part of this project (Chapter 5) is a systematic comparison of a suite of machine learning classifiers for sweep detection. The goal is to provide a simple protocol for researchers to follow if they are interested in using machine learning for detecting sweeps in population genetic data. Factors that we will consider when comparing different

[6] Areas with high recombination
[7] Except Section 4.3.1 used two, boosting and support vector machines

classifiers include the model's predictive accuracy, how interpretable the model is and the computational time required for model fitting. To enhance model interpretability, we will use tools from Section 3.5 to investigate variable importance. This workflow will assist researchers in comparing different methods in order to find one that is suitable for their research problem and data set. It will also help researchers understand how their models make their predictions which can provide insights into the underlying process of evolution.

Current approaches leverage modern DNA samples from existing individuals in the present. We would like to extend sweep detection methods to incorporate data from ancient DNA. Ancient DNA refers to DNA that has been retrieved from fossils and other archaeological samples. Studies into ancient DNA has emerged in recent years due to innovations in DNA sequencing technologies and analytical techniques. Overall, ancient DNA is useful for evolutionary biology because it enables researchers to directly study the DNA of past populations. Patterns produced by selective sweeps also decay over time due to the emergence of new mutations and recombination. Thus techniques which rely only on modern data are limited to finding recent sweeps. Incorporating ancient DNA into our analysis can enable our methods to detect older sweeps where the beneficial mutation has fixed many generations ago. Identifying these ancient adaptive regions can enhance our understanding of how populations have adapted to past environmental challenges. Chapter 6 will explore how to generalise our method from Chapter 5 to detect sweeps in ancient DNA.

Chapter 5

A Systematic Comparison of ML Classifiers for Sweep Detection

Although various researchers have applied different machine learning methods for sweep detection, there is little understanding regarding which specific methods should be used for any particular research problem (Section 4.4). There is a danger for researchers to deploy particular methods simply because they are new and hyped. In this chapter, we will apply a suite of machine learning classifiers for the purposes of sweep detection. We will compare the different methods based on their predictive accuracy, model interpretability and computational time. The objective is to provide a simple workflow for researchers to follow, should they be interested in using machine learning for sweep detection in modern genomes.

5.1 Methods

5.1.1 Population Genetic Simulations

In supervised learning, we need labelled training data where the true response is known (Section 3.2). For this study, we will use simulated genomes produced by the program discoal [71]. As input, discoal takes in some demographic parameters which describe how a population is evolving (*e.g* mutation rate, effective population size). Discoal's output consists of two parts; namely a binary genome matrix G and a numeric vector of positions p. The columns of G represent SNPs/segregating sites and the rows represent the individual genomes sampled from the population. $G_{ij} = 1$, if the i^{th} individual has a novel mutation at the k^{th} SNP position and 0 otherwise. In this context, the novel mutation is also called a derived allele. $p_j \in [0, 1]$ is the position of the j^{th} SNP in G.

For example, suppose we simulated a region, 1000 bases in length (*i.e.* 1kb). A SNP with position 0.2 is on the 200^{th} nucleotide base along the aligned genomes.

TABLE 5.1: Table of demographic parameters used for the discoal simulations. Note that the mutation and recombination rates are in event per base per generation. The values were based on [72, 73, 74]

Parameter	**Value**
mutation rate	1.5×10^{-8}
recombination rate	1×10^{-8}
effective pop size	10000
genome length	10^6
selection coefficient	$\{0, 0.005, 0.0125, 0.025, 0.0375, 0.05, 0.1\}$

Table 5.1 provides the basic demographic parameters used throughout our simulations. The selection coefficient is a relative measure of the strength of selection. Suppose we were comparing an ancestral and a derived allele with frequencies w_a and w_d respectively. The selection coefficient of the derived allele, relative to the ancestral is defined as [25, Chapter 7]

$$s := 1 - \frac{w_a}{w_d}\ ,\ s \in [0, 1]. \tag{5.1}$$

When $s = 0$, the derived and ancestral alleles are equally likely to be transmitted to the next generation (neutral simulation). Selection occurs when $s > 0$, and for this study we will be simulating these as hard selective sweeps. The reasoning is that hard sweeps are the most well known and many current methods focus on detecting hard sweeps. For every hard sweep simulation, the selected mutation occurs in the middle of the genome (*i.e.* position = 0.5). Our simulations were based on estimates done for modern non-African human populations. The mutation rate, recombination rate and effective population size were based on [72, 73, 74]. Although these estimates typically come with a range, we used fixed values for simplicity. All the sweeps were fixed at the time of sampling. One hundred sequences were generated in each simulation.

Using the parameters in Table 5.1, we simulated genomes under a constant population size scenario and various population bottleneck scenarios. Population bottlenecks were included because they are known to produce false positives in sweep detection. [75]. Many populations (including humans) are known to have a demographic history with bottlenecks [74]. When the population size rapidly shrinks during a bottleneck, this drastically reduces diversity thereby producing genetic patterns which are similar to selective sweeps. Figure 5.1 is a visual representation of a population bottleneck. The

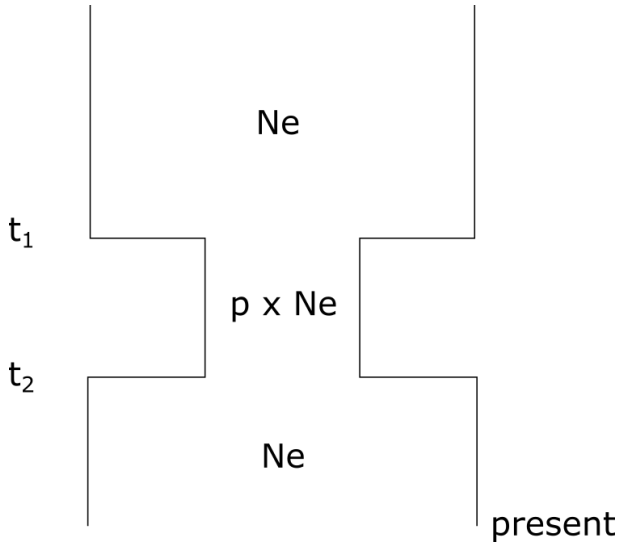

FIGURE 5.1: Diagram of a population bottleneck model

vertical axis represents time in generations which is measure backwards from the present. The present is at the bottom of the diagram and designated as time 0. This is when the samples were taken from the simulated population. Past demographic events are given times according to how many generations ago they happened. The width of the box represents the effective population size of the simulated population. Thus in a bottleneck, the population size shrinks by some factor $p \in [0, 1]$, t_1 generations ago and returns to the original size at t_2. $t_1 - t_2$ is known as the duration of the bottleneck. t_2 is known as the onset or recovery time. The severity of a bottleneck is $(t_1 - t_2) \times p$.

TABLE 5.2: Table of parameters for the bottleneck simulations

Parameter	**Value**
duration	1600, 8000
recovery	80, 800, 8000
bottleneck proportion (p)	0.05, 0.1, 0.5

Table 5.2 is a table of the parameters we used to construct eighteen different bottlenecks taken from [75]. This provides a wide range of severity, duration and age to the bottleneck simulations. Including the constant population size model, we have 19 different demographic scenarios to simulate using the parameters in Table 5.1. For each unique combination of demographic parameters, we conducted 1000 simulations, giving a grand total of 133,000 simulations. The simulations with a selection coefficient of zero are the neutral simulations. The simulations with non-zero selection coefficients are the hard sweeps. 80% of the simulations were randomly allocated into the training data set and the rest went into the testing set.

5.1.2 Discoal Simulations

All our data was simulated using discoal, a coalescent simulation program which generates population samples in a flexible manner [71]. Discoal simulates genetic samples using the coalescent process, a stochastic model which describes the genealogy of a set of alleles from a give population [76]. Samples "coalesce" backwards in time when they share a common ancestor. Coalescent simulations only simulate the requested number of samples, making it much more fast and efficient than "forward-in-time" simulators which simulate an entire population from which samples are drawn [71]. For neutral simulations, discoal simulates realisations of an ancestral recombination graph which models the genealogies of a set of recombining genetic sequences [38, Chapter 7.2]. Details about the algorithm can be found here [37]. Selective sweeps are generated the structured coalescent approach [77]. During the "sweep phase" of the simulation, samples will coalesce at a rate which depends on the frequency of the selected mutation as it spreads throughout the population [71, 78]. A full discussion of these coalescent theory models are beyond the scope of this project. Nevertheless, we have provided some useful papers in this section for interested readers.

5.1.3 Constructing Dataframe

In Section 5.1.1, we simulated population genetic data under various demographic scenarios using discoal. Each simulation consists of a binary genome matrix G and a numeric vector of SNP positions $\mathbf{p}$. As we explained in Section 4.3, this dimensions of this data is too high to be directly used as input for most machine learning classifiers. Drawing on the methods in Section 4.3.1 and Section 4.3.2, we will reduce the dimensions of the data by breaking the genome matrices into smaller blocks and applying a suite of population genetic summary statistics. The reasoning is that hard sweeps and neutral evolution produce different patterns in the summary statistics across a 1Mb genome window.

TABLE 5.3: Table of summary statistics used for the training data

Statistic Class	**Statistic**
Site Frequency Spectrum	Tajima's D, Fay and Wu's H
Haplotype	h1,h2,h12,h123
Linkage Disequilibrium	Kelly's Z_{nS}, w_{max}

Figure 5.2 is an illustration of how we split the genome matrices into smaller non-overlapping blocks of approximately equal size. The whole genome matrix represents a section of the genome and this is subdivided into smaller blocks are called windows.

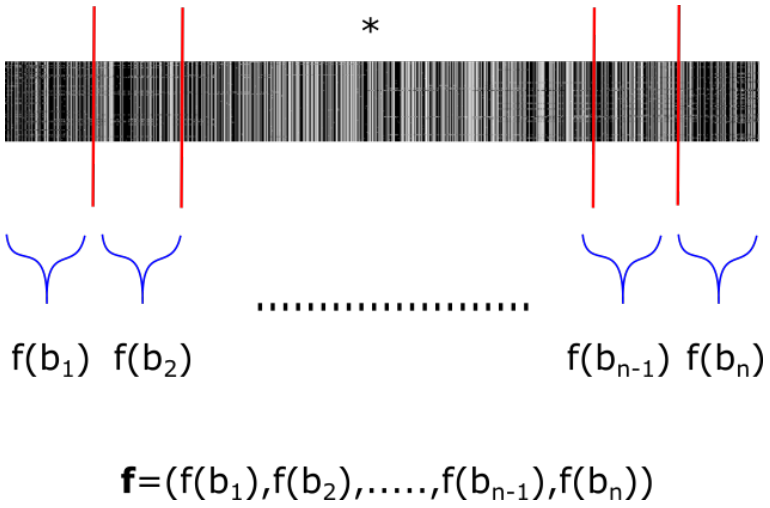

FIGURE 5.2: Illustration of how to split a genome matrix into n smaller blocks to compute summary statistics. Note the asterisk represents the selected mutation at the middle of the genome window, for the hard sweep simulations.

Suppose we have a matrix with K columns which were to be split into N windows. If K is divisible by N, then all the windows will have $\frac{K}{N}$ columns. If K is not divisible by N, all the windows will have $\frac{K}{N}$ columns except for the last window on the right which will contain an additional $K\%N$ columns (% is the modulo operator). The same summary statistic f is then applied across all N windows. We simulated 1Mb regions and subdivided them into 11 windows of 90kb. An odd number of windows ensures there is a central window with the hard sweep. S/HIC (4.3.2) also used 11 windows and we wanted our method to be comparable.

In total, we used eight statistics which cover the three main classes of summary statistics used for sweep detection (Table 5.3). The training data now consists of 88 predictors (11 for each statistic) and a single response variable indicating either a hard sweep or a neutral simulation. It was computationally expensive to compute the LD statistics. Hence, we downsampled 25% of the columns in each window for the purposes of computing Z_{nS} and ω_{max}.

5.1.4 Preprocessing

Before fitting our models, we will apply a set of transformations to the data in order to make it suitable for training. This is known as data preprocessing [79, Chapter 2]. We used the following preprocessing steps.

1. Highly correlated predictors can confound parameter estimation in many methods *e.g.* logistic regression [80]. We applied a correlation filter of 0.8. This removes predictors to ensure that the absolute correlation between any pair of predictors

must be ≤ 0.8. Our cutoff is within the recommended range found in [80]. A potential downside of applying a correlation filter is that some sweep scenarios may produce naturally correlated predictors. For example, θ_w and θ_π should both be low in strong sweeps. Thus in some cases biologically meaningful correlations could be filtered out, thereby reducing predictive power of our methods. We did not investigate different correlation filter cutoffs due to computational time constraints.

2. All predictors (except haplotype statistics[1]) were normalised[2] to ensure that no predictor will have a disproportionate effect on the model due to the range of values it could take. The haplotype statistics were exempt because they are proportions and hence must take values within $[0, 1]$.

After preprocessing the data, we have the final version of the training data which will be used for training our models.

5.1.5 Classifiers For Consideration

Section 5.1.3 explained how we converted high dimensional population genetic simulations into a training data frame with 88 predictors and a single response variable called "sweep" which can be hard or neutral. By condensing the simulations into a relatively small set of predictors, it is now feasible to fit conventional machine learning classifiers onto the training data. In this study, we focus on four machine learning methods; namely regularised logistic regression, random forests, MARS and regularised discriminant analysis (RDA). Regularised logistic regression was chosen because it is a simple classifier which is quick, simple to interpret and models linear patterns. Random forests was chosen because it is a quick and commonly used ensemble[3], tree-based technique. S/HIC successfully used another ensemble tree-based technique so this was worth trying (Section 4.3.2). MARS was chosen because it is a non-parametric technique that can model non-linear relationships. RDA was chosen to include a simple and quick technique based on decision boundaries.

Support Vector Machines (SVM) is another decision boundary method used in supervised classification [48, Chapter 12]. It is effective for modelling complex, non-linear relationships especially when flexible kernels (*e.g.* radial, polynomial) are used. However, given the size of the training data (106,400 observations with 88 predictors), an SVM with a flexible kernel would take excessive computational time to train. A SVM

[1] h1,h2,h12,h123

[2] Subtract the mean, divide by standard deviation

[3] An ensemble technique uses multiple smaller models together. A random forest is an ensemble of classification trees

with a linear kernel is faster but still quite slow compared to the four methods we have selected. Furthermore, a linear kernel SVM would separate data using a linear decision boundary which would be similar to LDA [4]. Hence, instead of SVM's, we opted to use RDA which is a much faster decision boundary method.

Due to the high dimensional nature of genome matrices, it is difficult to capture all their information using summary statistics. Section 4.3.3 explained that we could bypass this problem by using neural networks which can be trained directly on the matrices themselves. However, this approach introduces its own set of problems.

1. Neural networks are highly flexible models with many parameters [49, Chapter 1]. Thus a lot of data is required to train a good predictive model.
2. Training neural networks is computationally intensive and slow [49, Chapter 1].
3. We are interested in interpretable models that allow us to systematically compare how they make their predictions. For summary statistic based methods, we can use tools from Section 3.5 to investigate what kinds of patterns the models are picking up (*e.g.* Negative Tajima's D may increase the probability of a sweep). This cannot be done for neural networks where the predictors are the individual matrix elements of the genome matrix. There are too many low information predictors for any meaningful pattern to be ascertained.

Whilst neural networks could make good predictive models, we did not include them for the reasons above. Overall, we selected a set of models which cover the main classes of machine learning classifiers. If these relatively quick classifiers can function well, then more sophisticated and computationally intensive methods such as neural networks are unnecessary.

5.1.6 Hyperparameter Tuning

A simple grid search was performed to tune each machine learning classifier. This involves constructing a regular grid of hyperparameters and fitting a model on each hyperparameter set. For each classifier, we selected the hyperparameter set which gave the highest 10-fold cross validation(CV) accuracy. The final model is produced by refitting the model on the whole training data set, using the selected hyperparameter set. The aim of the grid search is to explore a broad range of hyperparameters in order to identify a suitable set to use for the final model.

[4] LDA is a special case of RDA when $\lambda = 0, \gamma = 1$

For regularised logistic regression, we used the lasso penalty with 10 regularly spaced $\lambda \in [0, 0.1]$. The lasso penalty was chosen because it shrinks small β coefficients to 0, helping to achieve a parsimonious model. This should be helpful for our data set where the set of predictors is large and some may only have a small effect on the response. For example, Tajima's D on the outer windows should have a minor effect on the response since it is far away from the selected mutation.

The random forests algorithm as implemented by the ranger package in R, has two main hyperparameters; namely "mtry" and "min_n." "mtry" is the number of randomly sampled predictors to consider for each split. "min_n" is the minimum number of training data points required to make a new split. After preprocessing, 52 predictors remained out of the original 88. We fitted models using a 4-level regular grid with values mtry$\in [10, 52]$ and min_n$\in [100, 1000]$. The number of trees was not tuned because this has been shown that once there are enough trees, tuning just models noise [81]. Figure 5.9 suggests our choice of 100 trees was sufficient to produce a good RF model.

The MARS model was fit using the forward pass algorithm (Section 3.3.5). The two hyperparameters to consider is the degree of interaction terms ("prod_degree") and the maximum number of MARS terms ("num_terms") to retain in the final model. We constructed a 5-level regular grid with "num_terms"$\in [1, 52]$. and allowed for interaction terms of up to degree 2.

RDA has two tuning parameters λ and γ. We constructed a 5 - level regular grid with $\lambda, \gamma \in [0, 1]$. This covers the range of possible values for both λ and γ. The grid ensures we fit LDA and QDA models as well as several intermediate models between the two (Section 3.3.6.4).

We considered the viability of applying more sophisticated optimization procedures for hyperparameter tuning (*e.g.* Bayesian optimization). This would be more computationally efficient than a grid search and could potentially yield a more suitable set of hyperparameters. This was deemed not worthwhile because there is already an internal optimization built into the model fitting process. Thus hyperparameter tuning does not need to be precise in order to produce a good predictive model. Rigorous optimization procedures are more suited for problems where there is a fixed objective function (*e.g.* engineering).

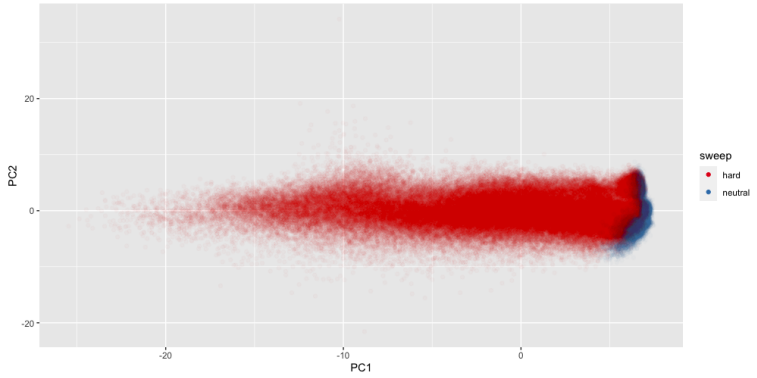

FIGURE 5.3: PCA plot of the whole simulated data using the top 2 principal components. The data points have been colored by sweep. Note that most of the variability in the data points are in PC1.

5.2 Results

5.2.1 Exploratory Data Analysis

We now use some unsupervised learning techniques to investigate patterns between variables in the simulated data.

5.2.1.1 PCA

Figure 5.3 is a PCA plot of the entire simulated data set. The hard sweeps show high variability on PC1 relative to the neutral simulations. There is moderate overlap between hard and neutral simulations for higher values of PC1. We can account for this pattern by considering that as the selection coefficient increases, there is a greater deviation from neutrality. Thus the high variance of hard sweeps along PC1 can be explained by the 7 different selection coefficients used. This can be confirmed by inspecting Figure 5.4 where higher selection coefficients are associated with more negative values on PC1. The linear differences in spread between neutral and hard sweep simulations along PC1, suggests that linear methods would be effective in distinguishing the two, especially when the selection coefficient is large.

5.2.1.2 Parallel Coordinates Plots

Parallel coordinates is a common technique for visualising high dimensional data. In this context, they can be used to visualise how the values of each summary statistic changes across the simulated genomes. Figure 5.5 is a parallel coordinates plot illustrating how Tajima's D varies across the 11 windows. Relative to the neutral simulations,

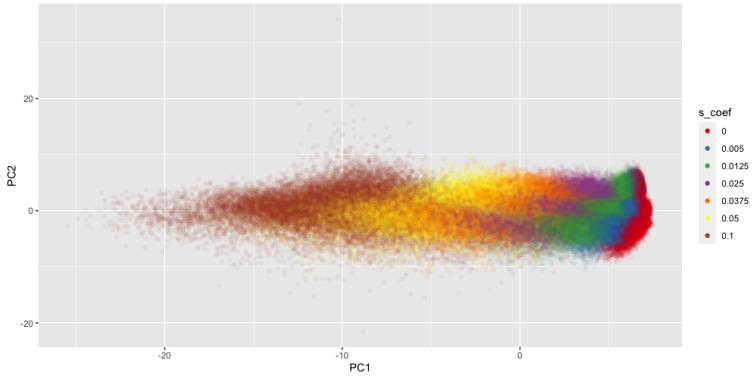

FIGURE 5.4: PCA plot of the whole simulated data using the top 2 principal components. The data points have been colored by the selection coefficient.

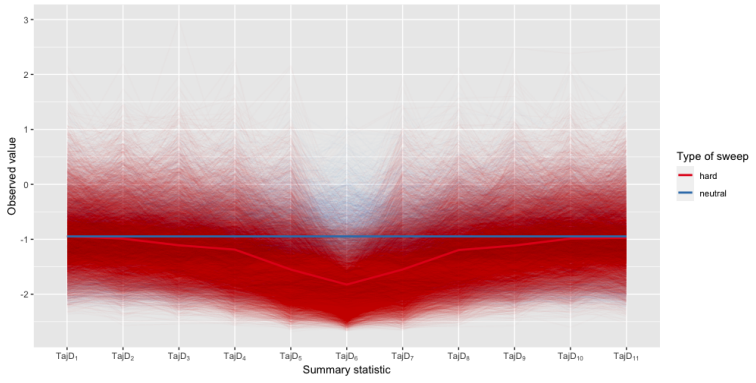

FIGURE 5.5: Parallel coordinates plot for Tajima's D. Each faded line is a simulation. The solid lines represent the mean values for hard and neutral simulations.

the hard sweeps show greater variability in Tajima's D due to the different selection coefficients used. Higher selection coefficients causes more right skew in the site frequency spectrum, thereby producing more negative values of Tajima's D. The mean line for neutral simulations is constant across the windows because each window is under the same evolutionary process. The expected value of Tajima's D for neutral, constant population models is 0. Here the neutral mean line lies around -1 due to the bottleneck simulations. The mean line for hard sweeps decreases as we approach window 6 from either side. The reason is that the selected mutation is situated at the middle of the hard sweep simulations (*i.e.* Window 6). Selection reduces Tajima's D and this effect is more pronounced for windows that are closer to the selected mutation. The differences in Tajima's D between neutral and hard sweeps suggests that the Tajima'D values in the central windows (particularly 5 - 7) may be useful for distinguishing hard sweeps from neutral simulations. Fay and Wu's H showed a similar pattern to Tajima's D, suggesting that they could likewise be useful for sweep detection Figure A.1.

Figure 5.6 shows how h1 across the genome. Unlike the SFS statistics, selection increases the h1 across all the windows. There is a mild shouldering effect where h1 has

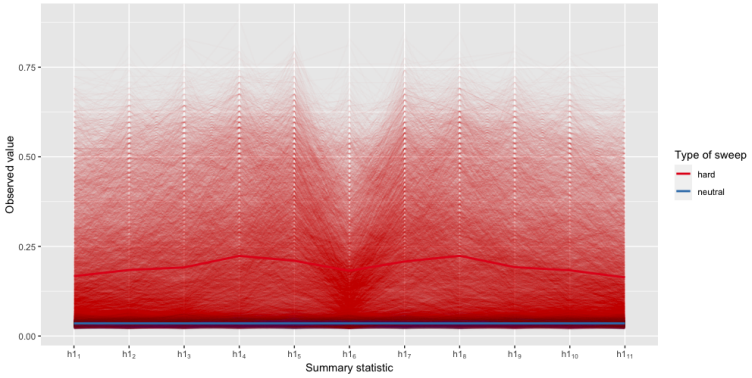

FIGURE 5.6: Parallel coordinates plot for h1. Each faded line is a simulation. The solid lines represent the mean values for hard and neutral simulations.

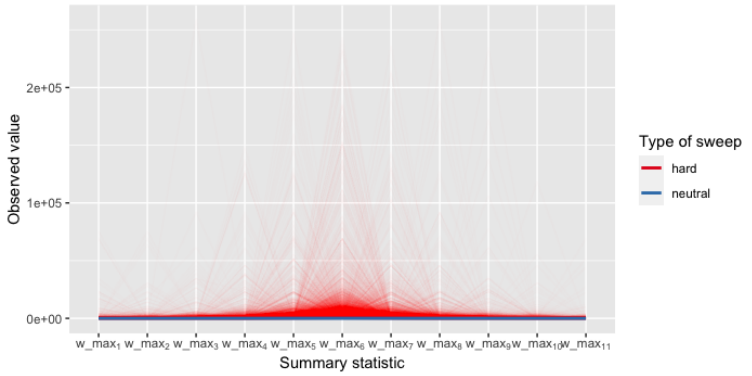

FIGURE 5.7: Parallel coordinates plot for w_{max}. Each faded line is a simulation. Note the log scale used to account for extreme values.

small peaks on Windows 5 and 7. Overall, h1 creates separation between hard sweeps and neutral simulations and hence could also be useful for sweep detection. The h12 and h123 statistics show similar trends to h1 due to the similarities in how they are computed (Figure A.3 Figure A.4). Kelly's Z_{nS} displayed a similar shouldering effect as h1 Figure A.5. The h2 statistic showed poor separation because it is was designed for distinguishing soft sweeps rather than identifying hard sweeps Figure A.2. Relative to neutral simulations, hard sweeps show greater variation of ω_{max} as we get closer to the central window Figure 5.7. Hard sweeps tend to generate larger values and occasionally extreme values. Recall ω_{max} is a ratio of LD values across different SNPs/columns in a genome matrix (Section 2.5.3.3). Hard sweeps reduce the number of SNPs, particularly around the selected mutation. This leads to inaccurate estimates of the numerator and the denominator of ω, thereby producing more extreme values of ω.

This section illustrates how population genetic summary statistics can vary across the genome. For statistics such as h1 and Kelly's ZnS, hard sweeps produce higher values for these statistics across all the windows. Other statistics (*e.g.* Tajima's D, Fay and Wu's H) are sensitive to the hard sweep when computed on a window that is nearby the

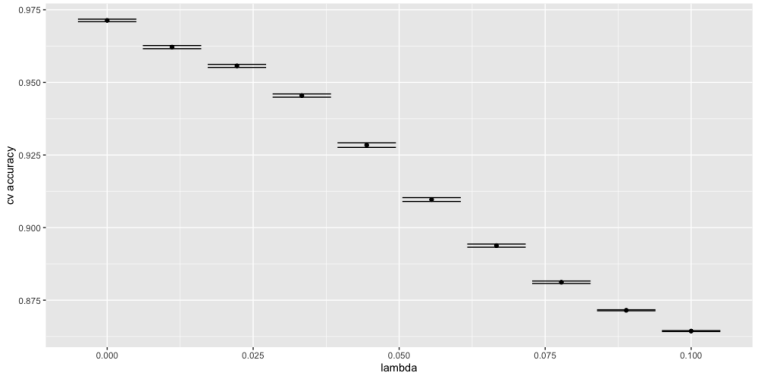

FIGURE 5.8: Plot of 10-fold cross validation accuracy across different values of λ in the regularised logistic regression model

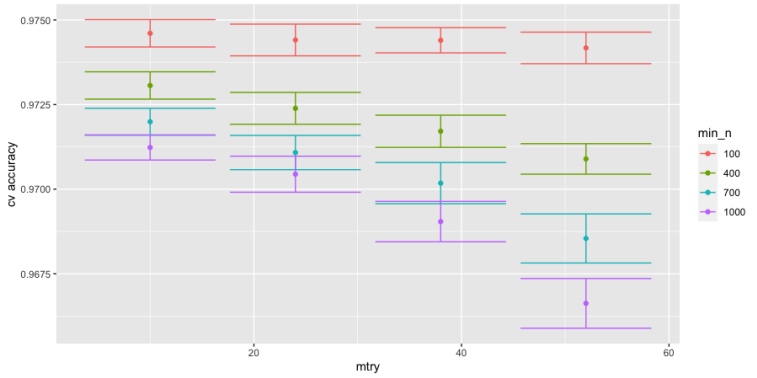

FIGURE 5.9: Plot of 10-fold cross validation accuracy across different hyperparameters in the random forests model

selected mutation. Overall, this shows how summary statistics can be used to distinguish hard sweeps from neutral simulations.

5.2.2 Hyperparameter Tuning

Our models were tuned by performing a grid search and selecting the set of hyperparameters with the highest 10-fold cross validation accuracy. We will now inspect how cross validation accuracy changes across the different sets of hyperparameters to ensure that each machine learning classifier has been adequately tuned to maximize predictive accuracy.

Figure 5.8 shows how the CV accuracy changes across different values of λ in the lasso logistic regression model. The cross validation accuracy has a roughly linear, negative relationship with λ suggesting that the lasso penalty was not needed to mitigate overfitting.

Figure 5.9 shows that the CV accuracy remains fairly stable across different values of "mtry." The more important hyperparameter appears to be "min_n" which underfits the

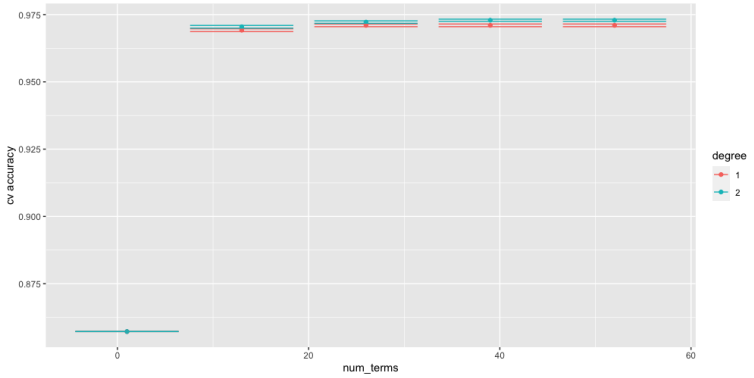

FIGURE 5.10: Plot of 10-fold cross validation accuracy across different hyperparameters in the MARS model. Note that when there is one MARS term, the maximum degree is one.

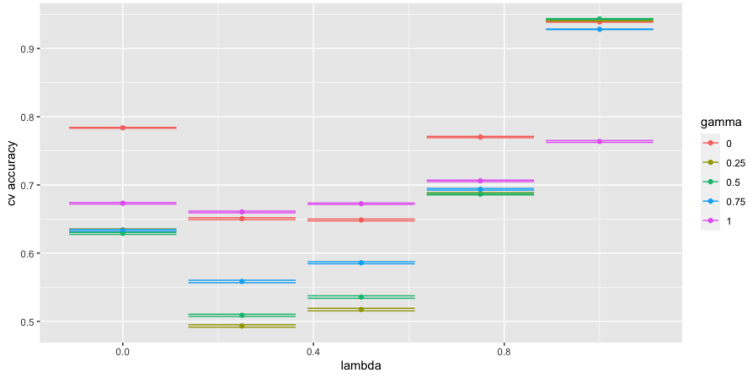

FIGURE 5.11: Plot of 10-fold cross validation accuracy across different hyperparameters in the RDA model.

model when it is too large. The vertical scale shows that the changes in CV accuracy is small overall which suggests there is no need to try more hyperparameters.

The MARS model peaks in cv accuracy when it has around 13 MARS terms. Adding more terms or introducing interaction terms make little difference to model accuracy (Figure 5.10). This is consistent with Figure 5.3 which suggests a linear classifier would be sufficient.

In the RDA model, there is a non-linear relationship between the hyperparameters (λ, γ) and the cross validation accuracy. One of the best models is when $\lambda = 1, \gamma = 0$ which corresponds to the traditional LDA model. This concurs with our exploratory data analysis which suggested that a linear classifier would be suitable. It seems that the more flexible hyperparameters lead to overfitting thereby reducing the CV accuracy.

Notice that the highest CV accuracy across the classifiers is approximately 0.975. The exception is the RDA model which has a smaller accuracy of 0.95. This suggests that given the nature of our data, the highest CV accuracy achievable may be around 0.975.

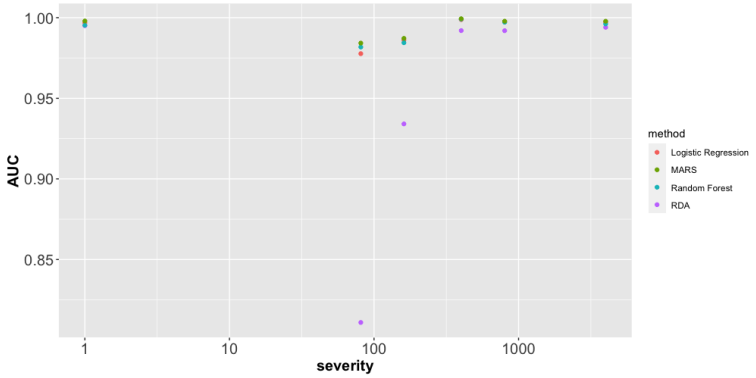

FIGURE 5.12: Plot of the AUC achieved by each classifier, across the bottleneck scenarios. The horizontal axis was offset by one to enable the log scale. Severity is the duration times the proportion of shrinkage in the bottleneck (Section 5.1.1). Some bottlenecks share the same severities.

5.2.3 Model Performance

We assess our suite of machine learning classifiers using the following criteria

1. Predictive accuracy. This can be quantified using the model's AUC for the entire testing set as well as for the different demographic scenarios. See Figure 5.12.

2. Computation time. The time required to fit a model given a set of hyperparameters. We fitted our models in parallel using an iMAC with 2.3 GHz Dual Core CPU and 16 GB 2133 MHz DDR4 RAM. See Table 5.5.

3. Explanatory power. How well does the model explain the relationship between the predictors and the response? This is a theoretical consideration based on how each machine learning method is designed.

TABLE 5.4: Table of the AUC of each machine learning classifier using a combined test set with all the demographic scenarios (constant population and various bottlenecks).

Machine Learning Classifier	**AUC**
MARS	0.995
Regularised Logistic Regression	0.994
Random Forests	0.994
RDA	0.973

TABLE 5.5: Table of the average computational time to fit each single classifier.

Machine Learning Classifier	**Average Time Taken (mins)**
Regularised Logistic Regression	0.386
Random Forests	12
MARS	0.847
RDA	0.635

5.2.4 Predictive Performance and Robustness to Population Bottlenecks

A common method for comparing predictive model performance is to use the AUC. This involves using each model to provide a soft classification (*i.e.* predict a probability of a success) for a testing data set and comparing it with the true response variable. We used a testing data set which contains simulations from a constant population size model as well as the 18 different population bottlenecks discussed in Section 5.1.1. Table 5.4 is a table of the AUC for our four machine learning classifiers when predicting on the entire testing data set. Overall, all four classifiers performed excellent with RDA performing slightly worse than the rest. This is consistent with our observations in Section 5.2.2 which shows that RDA has a lower cross validation accuracy than the other classifiers.

We are particularly interested in how the different classifiers perform when predicting for different population bottleneck scenarios. The reason is that population bottlenecks are known to confound sweep detection methods and produce false positives. Recall that the severity of a population bottleneck is the proportion of shrinkage multiplied by the time duration (Section 5.1.1). We divided the testing data into smaller sets for each severity. The models predicted on each set and the corresponding AUC was computed (Figure 5.12). There is a non-linear relationship between severity and AUC, with model performance declining for intermediate severity bottlenecks.

In severe bottlenecks, most/all of the samples will coalesce (*i.e.* share a common ancestor) within the bottleneck or afterwards. Consequently, the sequence alignments produced will appear like that of an expanding population size model. This does not confound summary statistics for detecting selection (*e.g.* Tajima's D). However, for intermediate severity bottlenecks some samples will also coalesce before the bottleneck. This causes the sweep signal then gets diluted by the bottleneck event which also decreases diversity [63]. Hence, model performance declines for intermediate severity bottlenecks but is higher for high severity bottlenecks and constant population size simulations. The RDA model performs the worst across all scenarios but particularly for intermediate severity bottlenecks. This is consistent with Section 5.2.2 where we found RDA to have the lowest cross validation accuracy among the four classifiers. Regularised logistic regression, MARS and random forests perform almost equally well across the different severities. Figure 5.12 shows mild differences in AUC for the intermediate severity cases, but these differences are too small to be practically relevant. Overall, these results suggest that our chosen machine learning classifiers are robust to population bottlenecks so long as these bottlenecks are provided in the training data. The exception would be RDA which has difficulty predicting for intermediate bottlenecks and has the worst performance overall.

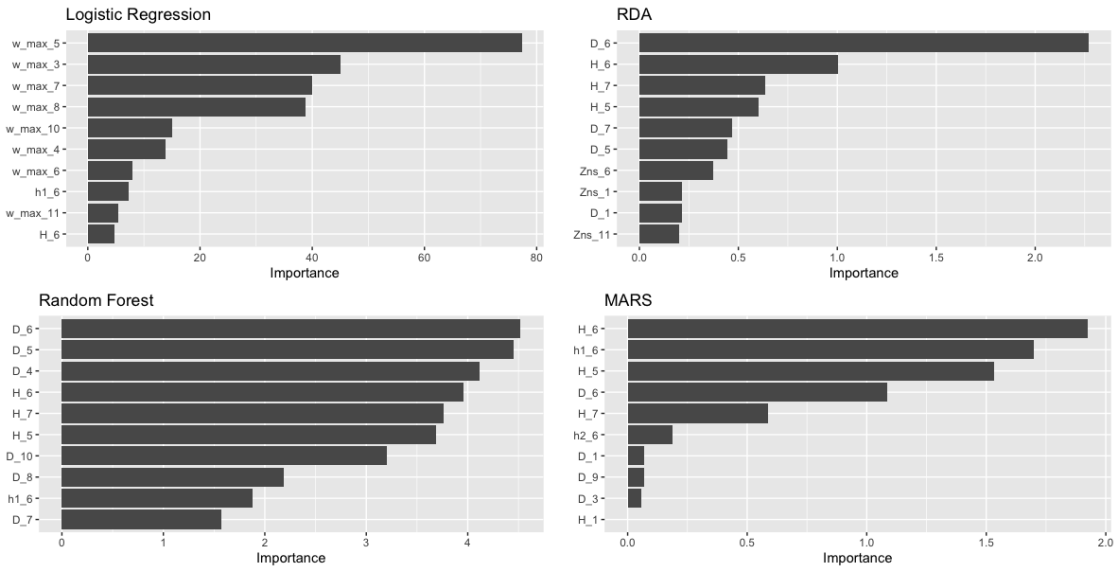

FIGURE 5.13: Barchart of FIRM scores for our four classifiers. The figure shows the top 10 important predictors in each model. The differences in the horizontal scales reflect each model possessing a different functional relationship with the predictors. For the purposes of model comparison, the rank order of the FIRM scores is more important as it reflects how each model weighs up the different predictors.

5.2.5 Variables of Importance

In addition to predictive accuracy, we are also interested in understanding how our models are making their predictions. This will provide insights into how our models work and what patterns they are picking up to detect sweeps. This section will use techniques from Section 3.5 to investigate variable importance.

5.2.5.1 FIRM Scores

FIRM quantifies the importance of a predictor using the variance of its partial dependence function (Section 3.5.3). Chapter 5 shows the top ten FIRM scores for our four classifiers. As explained in the figure legend, we will use the rank order of the FIRM scores to compare our models.

The logistic regression model is unique in that it is mostly using ω_{max}, mostly computed nearby the central window (Window 6). This is consistent with Figure 5.7 which shows greater separation towards the central window. The large scale may reflect the extreme values of ω_{max} shown among hard sweeps. Although they played a smaller role, the model also used h1 and H on Window 6. This is sensible given both statistics showed good separation (Figure 5.6,Figure A.1).

The RDA, random forests and MARS models all relied heavily on the SFS statistics[5] nearby the central window. This concurs with our observations in Section 5.2.1.2 and how the sweep signal is typically stronger around the selected mutation. Random forests and MARS also used h1 which was shown to distinguish hard sweeps (Figure 5.6, Section 2.5.2). h2 was used in MARS but its FIRM score was much lower than those of h1, D and H. This reflects how h2 was not designed for picking up hard sweeps and showed relatively poorer separation (Figure A.2). Instead of using haplotype statistics, RDA used the LD statistic, Kelly's Z_{nS} computed on Windows 1, 6 and 11. This corresponds to the central window as well as the two extreme ends. This is sensible given Z_{nS} shows good separation across the whole simulated regions (Figure A.5).

In conclusion, section 5.2.5.1 shows that each method relies on a different set of summary statistics to reach its classification decision. Nonetheless, all four methods have good predictive accuracy. This challenges a prevailing view that particular summary statistics are characteristic of particular selection scenarios. Instead, it appears that the dynamic between the summary statistics and the methods are most important.

5.2.5.2 Partial Dependence Plots

Section 5.2.5.1 identified the key predictors that each model was using to make its predictions. We now use partial dependence plots (Section 3.5.1) to observe how these predictors affect the predicted response in each model. We won't be discussing the ICE plots because they took too long to generate. We will consider the pdp's of three most important predictors for each classifier according to FIRM. Although three is a somewhat arbitrary cutoff, the top few predictors have the greatest influence over the prediction and hence are most informative about how our models work. As we go down to the low importance predictors, the pdp's will essentially be modelling noise.

Logistic Regression: See Figure 5.14. Recall ω_{max} is an LD statistic which is expected to be larger for hard sweeps. For Windows 3,5,7 the predicted log odds of a hard sweep has a positive, linear relationship with ω_{max}. This is consistent with Figure 5.7 which show hard sweeps to have higher ω_{max} than neutral simulations. Notice that prediction gets really high as ω_{max} gets large. This reflects how hard sweeps occasionally had extreme values of ω_{max} whilst neutral simulations had consistently smaller values. Thus when ω_{max} is large (*e.g.* ≥ 20), the model is almost certain that a hard sweep is present. Thus our logistic regression model detects sweeps by finding large values of ω_{max}, mostly near the central window.

[5] Tajima's D and Fay and Wu's H

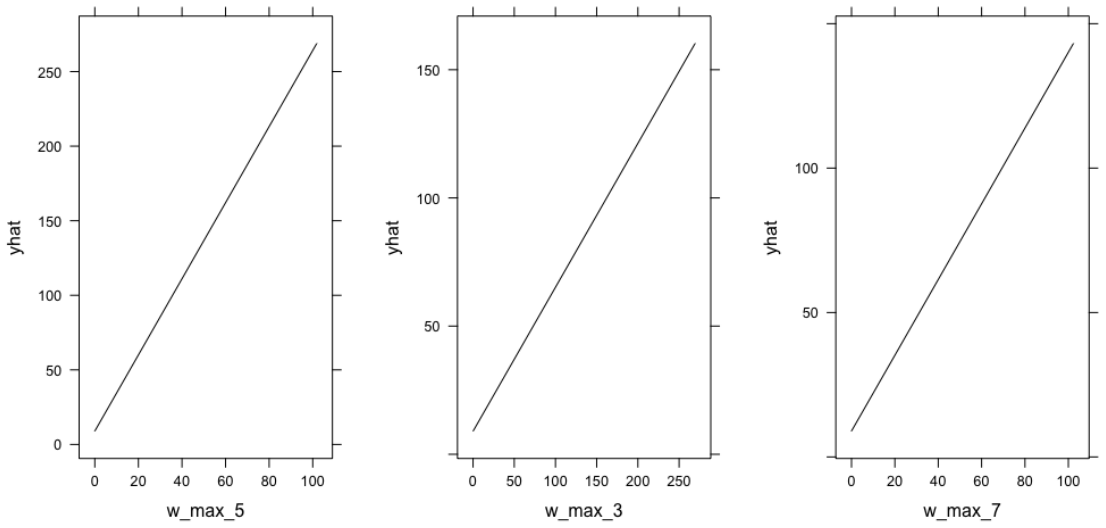

FIGURE 5.14: The partial dependence plots for the top three predictors according to FIRM, in the logistic regression model. "yhat" is the predicted log odds of having a hard sweep.

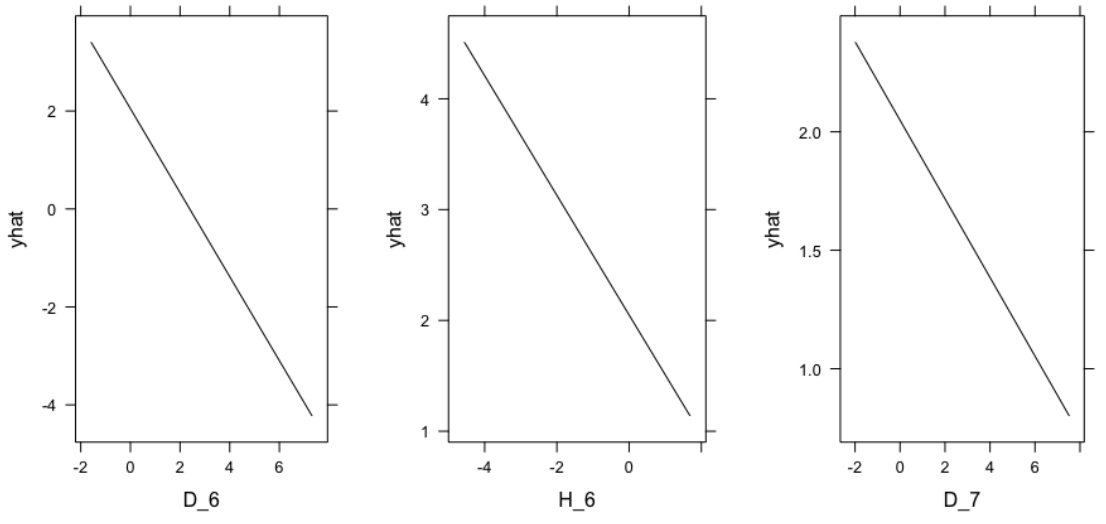

FIGURE 5.15: The partial dependence plots for the top three predictors according to FIRM, in the RDA model. "yhat" is the predicted log odds of having a hard sweep.

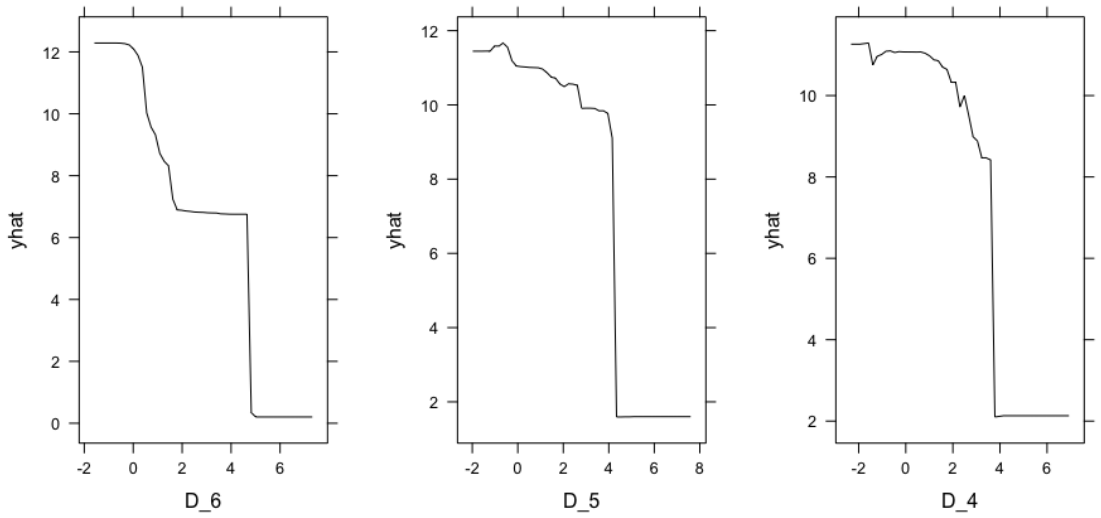

FIGURE 5.16: The partial dependence plots for the top three predictors according to FIRM, in the random forest model. "yhat" is the predicted log odds of having a hard sweep.

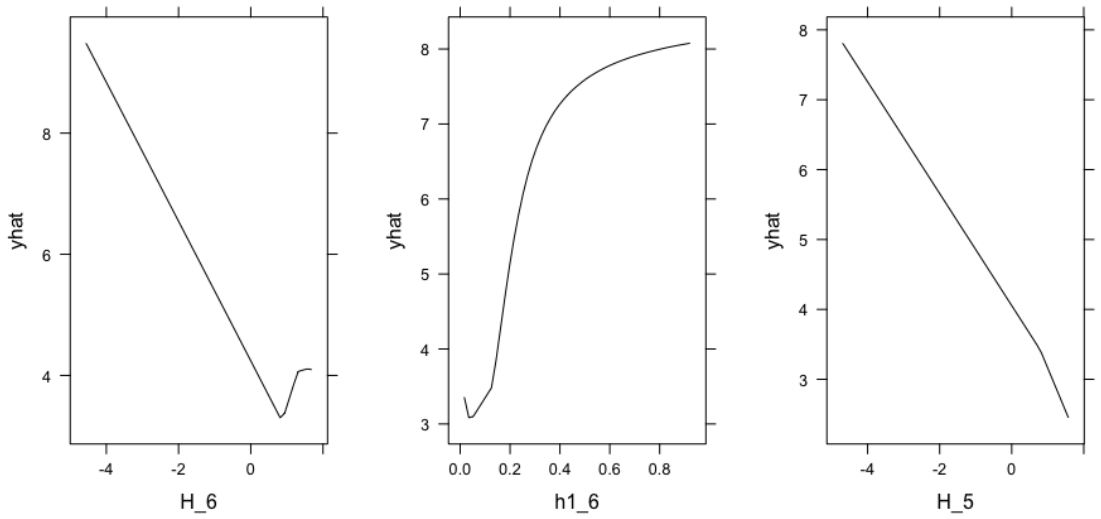

FIGURE 5.17: The partial dependence plots for the top three predictors according to FIRM, in the MARS model. "yhat" is the predicted log odds of having a hard sweep.

RDA: See Figure 5.15. Recall that Tajima's D and Fay and Wu's H are both SFS-based summary statistics which are meant to be smaller for hard sweeps. In the pdp's, we observe that the prediction decreases linearly as D and H goes up nearby the central window. This concurs with Figure 5.5 and Figure A.1 which show hard sweeps to have smaller values of D and H. This means the model detects hard sweeps by looking for smaller values of D and H, particularly around the central window.

Random Forest: See Figure 5.16. Like the previous models, there is a negative relationship between D and the predicted response. The pdp's are wiggly due to the more complex nature of the random forest model, which is an ensemble of classification trees. The pdp's become flat when $D \geq 4$. The reason is that virtually no hard sweeps will have such high values of D around the selected mutation. Thus when $D \geq 4$, the model is very confident that input is a neutral simulation and further increases in D will not change its prediction. Hence, the random forest model detects sweeps using smaller values of D nearby the central window.

MARS: See Figure 5.17. Overall, there is an approximately linear, negative relationship between H and the predicted outcome. There is a small kink upwards for "H_6" at around 0.5 but this is likely due to statistical noise. Recall that h1 is the square proportions of all haplotypes in the population. Hard sweeps are expected to have higher values of h1 since hard sweeps produce a dominant haplotype. Inspecting the pdp, there is a positive, non-linear relationship between "$h1$_6" and the prediction. This concurs with our theoretical expectation as well as Figure 5.6. The small kink around 0.05 is likely due to statistical noise. The non-linear trend reflects the capability of MARS to model non-linear patterns. In summary, the MARS model detects sweeps by looking for smaller values of H and higher values of h1, nearby the central window.

5.2.6 Selecting a Machine Learning Method

When selecting a predictive model, there are three key considerations; namely predictive accuracy, computational time and the model's inherent interpretability. In Section 5.2.4, we found that regularised logistic regression, MARS and random forests have almost equally high predictive accuracy and were robust to different severity bottlenecks.

To measure computational time, a benchmark test was conducted using a Macintosh computer with a 2.3 GHz Dual-Core Intel i5 processor and 16GB of 2133 MHz DDR4 RAM. Table 5.5 shows that average time taken to fit a single model using a single set of hyperparameters. In practice, models are fitted with different hyperparameters simultaneously via parallel computing. Our results show that all our methods took less than one minute, with the exception of random forests which took twelve minutes.

Out of all classifiers available, logistic regression is the easiest to interpret. The effect of each predictor can be quantified by their corresponding β coefficient which shows their effect on the predicted log-odds of a success. Logistic regression is mathematically simple so that the precise mathematical formula can be easily written out. This cannot be said for the other classifiers (*e.g.* MARS) which have a more complex relationship between the predictors and the response. Logistic regression has the added advantage of being simple to explain to a non-technical audience.

Given these considerations, we recommend researchers to use logistic regression for detecting hard selective sweeps in population genetic data where population bottlenecks are present. This recommendation only applies when the underlying demography is relatively simple (*e.g.* no admixture, constant population size). In these simple scenarios the problem is essentially linear. However, in more complex settings where the problem is non-linear, a more sophisticated predictive model may be required. Nevertheless, we recommend researchers to still try logistic regression as a "first pass" to then compare with more sophisticated models.

For logistic regression, a lasso penalty term should be sufficient for mitigating overfitting, although even that may not be needed after some preprocessing. Logistic regression is fast, easy to interpret and has high predictive accuracy. Given its large AUC score of 0.994, it is unlikely for more complicated machine learning classifiers (*e.g.* neural networks) to be much more accurate.

This is reasonable when we consider that logistic regression is a low variance, high bias technique which is much less prone to overfitting compared to many other classifiers. If such an inflexible technique can already have a high predictive accuracy, then more flexible models are unlikely to perform better. Flexible models have a greater danger of overfitting. In a practical context, researchers rely on estimates of the demographic parameters because the true demographic model is unknown. Overfitting could lead to worse performance when the estimated demographic models used for training is different from the true model. Hence, more flexible models are at greater risk of having reduced robustness to model misspecification.

Once again we caution that for the purposes of this review, we only considered simple demographic scenarios with either a known constant population size or a population bottleneck. Other demographic parameters such as the mutation rate and recombination rate were fixed. It is possible that when dealing with more complicated demographic scenarios with different parameters, more flexible machine learning classifiers may be more suitable. Nevertheless, within the small set of scenarios we have considered, regularised logistic regression is suffice.

This recommendation may be surprising because there are many new sophisticated machine learning methods available (*e.g.* neural networks) and logistic regression is a relatively old technique. The general expectation is for new methods to be a substantial improvement from old ones. However, the evidence suggests that this is not always the case. For example, there was a competition in the SemEval2019 conference to classify emotions in text messages [82]. The winner of this competition used a linear support vector machine whilst other competitors used more sophisticated models like neural networks. The authors showed that an accurate model can be produced using an old technique supported by some preprocessing. Our study similarly shows that although new methods may be exciting and fashionable, they are not always the best approach to the problem at hand. In fact, traditional methods coupled with preprocessing may perform surprisingly well. We recommend researchers to first try simple, quick classifiers to set a performance baseline. If the simple models are insufficient, only then should they move to more sophisticated classifiers which require more computational resources.

5.2.7 Suggested Workflow for Researchers

We will now use the results of our study to provide a simple workflow for researchers who want to use machine learning to detect selective sweeps in population genetic data.

1. Determine a set of demographic histories which could adequately describe your population of interest. There are a number of tools available which can use genetic data to infer demographic history. Popular tools include PSMC and MSMC [83] [84].

2. Use discoal to simulate 1Mb genomes under the set of demographic histories determined in Step 1. A balance must be struck between computational time and having a broad set of demographic histories. If the set of demographic histories is too narrow, we risk training our classifier on non-representative data and producing an inaccurate model. If the set is too broad, then many simulations have to be made and more computational resources have to be expended downstream.

3. Convert the simulations into a dataframe using the method explained in Section 5.1.3. Randomly assign $\sim$ 80% of the data for training and the rest for testing.

4. Fit a regularised logistic regression model using the lasso penalty. Tune the model by trying a regular grid of λ values (including $\lambda = 0$) and selecting the one which gives the highest 10-fold cross validation accuracy. Although the lasso penalty was

ultimately not needed in our study, we still recommend using it because fitting logistic models is quick and regularisation reduces the risk of overfitting.

5. Use the final model to predict on the testing data and compute the AUC. This gives an indication of the model's predictive accuracy.

6. Investigate variables of importance using FIRM and partial dependence plots. This will provide insights regarding the kind of patterns the model is detecting as well as which statistics are useful.

7. Write up a wrapper function to convert your empirical data into a data frame using the method described in Section 5.1.3. Use for fitted model to predict each 1Mb genome block.

Although this project focused on detecting hard sweeps, our workflow can be generalised for arbitrary population genetic problems where the appropriate machine learning method is unknown. Here is a more general workflow for any new problem.

1. Consider whether your task is a classification or regression problem (Section 3.1).

2. Use steps 1-3 from Section 5.2.7. These steps may need to be adapted for your specific problem. For example, different summary statistics may be required to capture the response variable.

3. Use a suite of machine learning methods. Try simple and quick methods first. Some suggested methods for classification are regularised logistic regression, RDA, random forests and MARS. For regression, try lasso regression [48, Chapter 6], ridge regression [48, Chapter 6], MARS and random forests. Tune your models by performing a regular grid search and using 10-fold cross validation. Perform benchmark tests to determine the average computational time to fit a model for each technique.

4. Adapt steps 5-6 from Section 5.2.7. Compare your models by considering their predictive accuracy, computational time and interpretability. Use the pdp's to deduce the kind of genetic patterns which are associated with the response. Select a final model and fit that on the whole training data.

5. Step 7 (Section 5.2.7).

5.3 Conclusion

In this chapter, we designed a machine learning workflow for detecting hard sweeps in modern genomes. This involved applying a suite of classifiers and comparing different models using AUC, computational time and considering inherent model interpretability. We found that accurate models can be produced using simple, quick classifiers supported by data preprocessing. Techniques from Section 3.5 were also used to investigate variable importance and to unravel how our models made their predictions. The end product is a simple machine learning workflow which researchers can use for sweep detection in modern data. Although our project focused on detecting hard sweeps, our workflow can also be generalised to arbitrary population genetic problems where the appropriate machine learning method is unknown. Researchers can use our workflow to compare different methods and understand what sort of genetic patterns their models are using for their predictions. The next chapter will generalise our method to work with ancient DNA.

Chapter 6

Detecting Sweeps in Ancient DNA

Section 4.4 articulated two main knowledge gaps in regards to using machine learning to detect selective sweeps. The first gap is that researchers have applied different machine learning methods to sweep detection in isolation. Hence, there is no understanding of what methods are suitable for any particular research problem or data set. The second gap is that current methods are designed for use with modern data. This means current methods cannot be directly applied to the growing set of ancient DNA data that is being produced. Chapter 5 addressed the first gap by applying a suite of machine learning methods to the problem of sweep detection in modern genetic data. We found that hard sweeps and neutral regions can be well separated by using a suite of population genetic summary statistics. By using some basic feature engineering and selection, we were able to produce accurate predictive models using simple, fast and interpretable machine learning methods (e.g. logistic regression). This chapter aims to address the second gap by investigating how we could extend our method to work with ancient DNA.

6.1 Ancient DNA

6.1.1 What is Ancient DNA?

Ancient DNA refers to DNA that has been isolated from ancient specimens. Sources of ancient DNA include fossilised bones, mummified tissues and teeth [85]. One of the earliest ancient DNA studies was conducted in 1984, on a museum specimen of the quagga; a South African plains zebra which became extinct in 1883 [86]. The researchers extracted DNA from dried muscle tissue via a set of biochemical treatments.

They cloned the DNA and sequenced pieces to produce a 229 bp mitochondrial genome of the quagga. They sequenced the much smaller mitochondrial genome instead of the much larger quagga genome because DNA sequencing was still in its infancy in 1984. This meant that DNA sequencing was slow, inefficient and expensive. Nevertheless, this study was important because it demonstrated that DNA sequences can be retrieved from extinct species. The last couple of decades saw major technological advances in ancient DNA and DNA sequencing methods (*e.g.* high throughput sequencing). This enabled ancient DNA researchers to not only sequence mitochondrial genomes but to also generate genome wide data sets from ancient DNA [87]. Ancient DNA is a rapidly emerging field and many ancient genomes are now being sequenced. Consider the field of hominid evolution. Prior to 2010 there were only a few archaic hominid genomes available. As of 2017, over 1100 ancient genomes of archaic hominids and anatomically modern humans have been published. Researchers today are using ancient DNA to examine a broad range of archaeological specimens such as ancient horse bone found preserved in the Canadian Yukon territory and whale tissue decorating an electric lamp in the Australian National Maritime Museum [88, 89].

Ancient DNA is important for evolutionary biology because it provides genetic insights into ancient populations. Comparing ancient genomes with modern data can enhance our understanding of population history and how organisms adapted to past environmental challenges. For example, the first Neanderthal genome was produced using DNA extracted from Neanderthal bones (50k-65k years old) found in the Vindija cave in Croatia [90]. Comparison of the Neanderthal genome with modern human DNA has revealed non-African populations (*e.g.* European, Chinese) to have approximately $1-4\%$ Neanderthal DNA whilst African populations (*e.g.* San, Yoruba) has none [90]. This suggests that the ancestors of non-African humans interbred with Neanderthals when the two populations met in the Middle East [91]. Further studies into the remaining Neanderthal DNA found in non-African populations suggest that they once offered resistance towards certain RNA viruses [92]. Thus researchers have used ancient DNA to unravel the history of human evolution and to understand how the human genome has evolved across time.

In the context of detecting selective sweeps, ancient DNA is also useful for detecting old sweeps. Figure 6.1 is an illustration of how sweep patterns can decay over time. The left panel represents a genetic region where a selected mutation (red box) has recently fixed. The low genetic diversity around the selected mutation means that it can be detected by population genetic summary statistics such as Tajima's D and haploptype statistic h1. As time passes, novel mutations are spontaneously generated thereby increasing the genetic diversity within the region. This adds noise to the sweep signal, making it more difficult to be detected by population genetic statistics. Over time, different populations

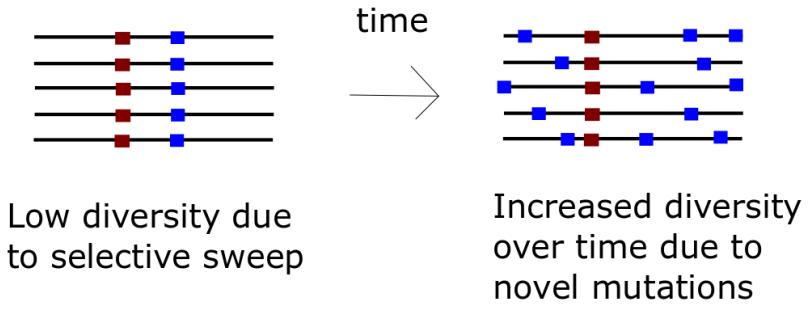

FIGURE 6.1: Illustration of how hard sweep patterns decay over time. Each line represents a chromosome. The blue boxes represent neutral mutations and the red boxes is a selected mutation.

can interbreed, thereby diluting signals over time. Ancient DNA can bypass this problem by enabling researchers work directly with genomes where the old sweep patterns have not yet decayed. This helps researchers to investigate older changes to the genome which cannot be accessed with modern data.

6.1.2 Technical Challenges

Although ancient DNA is useful for evolutionary biology, it introduces its own set of technical challenges. We will now explore the key technical challenges of using ancient DNA which are not found in modern genome data.

Provided the correct protocols are followed, genetic materiel sampled and sequenced from living individuals are typically 100% endogenous [91]. This means that almost all the DNA fragments extracted originated from the sampled individual. This enables researchers to have an accurate genetic record of the individuals. In contrast ancient DNA is scarce and most of the DNA extracted from fossils are exogenous (*i.e.* originating from contaminants). The reason is that after death, an organism's DNA degrades over time. Meanwhile environmental microbes and other organisms can make contact with the remains, leaving behind their own DNA. Inappropriate human handling of the sample can produce additional contamination. Exogenous DNA obscures the data, making it difficult to accurately reconstruct the individual's original genome. Contamination must be adequately addressed for downstream bioinformatic analysis to be valid.

Researchers have developed several strategies for mitigating the effects of contamination in ancient DNA [91]. Standard laboratory protocols have been developed to minimise further contamination from human handling. This involves working within a clean isolated room, using filtered air systems and bleach treatment of surfaces [93] [94]. Prior to sequencing, the extracted DNA fragments are given molecular tags to prevent any additional contaminant DNA from being confused with the sampled ancient DNA [95]. After the sample has been sequenced, there are a suite of bioinformatic tools available which can either filter out contaminant DNA or estimate the overall contamination rate [91].

DNA molecules frequently suffer from chemical insults which can potentially alter the underlying sequences [96]. In living organisms, this process is countered by a set of robust DNA repair mechanisms which maintain the integrity of the genome. These mechanisms shut down upon death, exposing the individual's genome to the set of chemical factors which threaten its stability. For example, intracellular nucleases are special proteins used within DNA repair mechanisms which can cut DNA molecules [97]. Upon death intracellular nucleases are no longer sequestered in the cell, enabling it to access the genome and degrade it. Bacteria growing on decaying tissue can also breakdown the host's DNA. Given sufficient time and the appropriate environmental conditions, all retrievable endogenous DNA can be destroyed. If the sample is frozen or desiccated shortly after death, this can slow down the degradation of its DNA. We will now discuss several key technical problems caused by post-mortem DNA damage.

6.1.2.1 Fragmentation

Section 2.2 was a brief overview of DNA sequencing. The process involves cutting the sampled DNA into smaller fragments, sequencing those fragments into reads and then reassembling those reads to retrieve the original genome. In modern samples, endogenous DNA is plentiful which enables researchers to reassemble the sampled genome with confidence. Ancient DNA is more challenging because the sample DNA already comes in small fragments due to post-mortem DNA damage. A review of ancient DNA extracted from soft tissues that varying in age from 4 - 13 000 years has shown that most ancient DNA has degraded to fragments of 40-500 bp [98]. Another issue is that the majority of the sequenced reads come from contamination rather than the ancient individual [99]. Hence, endogenous reads are short and scarce, making genome assembly challenging. Depending on the quality of the sample, it may not always be possible to reconstruct the whole genome of the individual because there is not enough data from the reads to deduce some areas of the genome [100]. Thus when ancient samples are aligned together for population genetic analysis, there will be missing information since the base pair of

some individuals at some positions are unknown. Analytical tools which seek to leverage ancient DNA must account for potentially high rates of missingness in the data [101].

6.1.2.2 Deamination

Ancient DNA is susceptible to hydrolytic deamination (a chemical reaction) which modifies the DNA so that it is misread by the sequencing machinery [97]. The primary target of deamination is cytosine which is converted uracil. Whilst cytosine binds with guanine, uracil binds with adenine. During DNA sequencing, uracil incorrectly incorporates adenine instead of guanine, thereby creating a $G \longrightarrow A$ substitution on the complimentary strand ($C \longrightarrow T$ mutation on the uracil strand). Although deamination is known to produce other kinds of substitutions (*e.g.* $A \longrightarrow G|T \longrightarrow C$), the $G \longrightarrow A|C \longrightarrow T$ substitution is the most common in ancient DNA [102]. In the context of SNP data, deamination can produce false SNPs by generating variants that did not exist in the original organism. In contrast to modern data, the sequencing of ancient DNA is more error prone due to deamination.

6.1.2.3 Ascertainment Bias

Before explaining ascertainment bias, we need to discuss how ancient samples are sequences. It is challenging to accurately sequence ancient samples because the original, endogenous DNA is mixed with lots of contaminant DNA Section 6.1. One strategy is to use shotgun sequencing which involves randomly cutting up the sampled DNA into smaller fragments using restriction enzymes [103]. The fragments are sequenced into reads and bioinformatic tools are used to filter out contamination. Then an alignment program reassembles the genome using the overlapping ends of the reads with the help of a reference genome. After sequencing several individuals, we can identify all the SNPs within our sample for population genetic analysis. However, shotgun sequencing is expensive and inefficient because endogenous DNA is rare ($\sim 1\%$ or less) [104, 105]. This means resources have to be wasted on sequencing lots of contaminant DNA before there enough endogenous DNA has been sequenced. This method will also sequence many endogenous consensus regions when the population genetic analysis is focused on the SNPs.

An alternative strategy is to use SNP arrays; DNA chips with a collection of microscopic DNA spots [106]. Each spot contains a specific DNA sequences known as oligonucleotide probes. Fluorophore tags are attached to the sampled DNA fragments which are then

added onto the chip. DNA fragments which are complementary to the chip's oligonucleotide probes will pair up with them via hydrogen bonding (hybridisation). By selecting the appropriate probes, researchers can ensure that endogenous DNA rather than contaminant DNA will bind to the probes. This process said to be enriching for endogenous DNA. The chip is washed, removing any DNA that is unattached to the probes. A detection system records the fluorescence signal which is used to sequence the DNA that has remained attached to the probes. The enrichment procedure ensures resources are not wasted on sequencing the contaminants.

This raises the question of which oligonucleotide probes should be used on a DNA chip. Genetic regions which are likely to be polymorphic in the population of interest are desirable because they would be most informative for population genetic analyses. The process of identifying these polymorphic sites is known as SNP ascertainment. Keinan's method identifies SNPs by using two chromosomes within an individual of known ancestry and comes from a closely related population to the population of interest [107]. This individual is known as an ascertainment individual. (For example, we can use an African individual to ascertain the SNPs for some ancient human remains found in Europe). Polymorphic sites within the ascertainment individual are potential SNP sites.The DNA sequences around these sites can be used to make complimentary oligonucleotide probes on the DNA chip. Different probes are made for each variant at each site.

Since the ascertainment individual comes from a closely related population, it is likely to share many SNPs with the population of interest. DNA fragments which bind the the probes would be mostly endogenous because contaminant DNA coming from other species are unlikely to match probes. The ancient individual can then be sequenced at the ascertained SNP positions using high-throughput sequencing methods [108]. Multiple ascertainment individuals can be used to increase the number of potential SNP sites. The DNA chip method is more efficient than the shotgun sequencing approach. Whilst shotgun sequences attempts to reconstruct the sample's entire genome, the chip method only sequences regions which are likely to be informative. It has been shown that the chip method can produce similar quality results as shotgun sequencing, for a fraction of the cost [109]. Hence, researchers commonly use DNA chips for sequencing ancient samples for population genetic analyses.

A limitation of using DNA chips is that we sequence the sample at a subset of SNPs based on a set of ascertainment individuals. In order for a SNP to be identified, both alleles must be found within the chromosomes of at least one ascertainment individual. This method is biased towards discovering SNPs with intermediate frequency alleles and has difficulty identifying SNPs with low frequency alleles. This process is known

as ascertainment bias. Ascertainment bias distorts the site frequency spectrum found in the data and produces deviation of population genetic summary statistics from their theoretical expectations. For the context of detecting selective sweeps, ascertainment bias can potentially confound SFS-based statistics such as Tajima's D and Fay and Wu's H. In summary, a common method of sequencing ancient samples is to use a DNA chip with oligonucleotide probes based on a set of ascertainment individuals. This leads to ascertainment bias because the downstream analysis is conducted on a subset of SNPs which do not represent the overall SNP distribution in the population.

6.1.2.4 Pseudohaplotypes

Since DNA is plentiful in living individuals, modern DNA samples are relatively high quality and many DNA fragments can be retrieved. The abundance of DNA fragments ensures that there is high coverage for the whole genome, meaning that many reads can be found for any region of the genome. This rich data enables alignment algorithms to reconstruct the whole genome with high confidence. When working with diploid organisms which have two copies of each somatic chromosome, bioinformatic tools are available to estimate the haplotypes of each individual [110]. The process of estimating the haplotypes of an individual based on DNA reads is known as phasing.

It is much more challenging to phase ancient DNA because endogenous reads are rare and short. After reassembling the reads (Section 2.2), some regions of the genome may only have a couple or less reads mapped to it. Although variants may be found within two reads, it is difficult to distinguish real genetic variants from false variants produced by deamination. Due to its low information, the underlying haplotypes of ancient DNA cannot be discerned (unphased) [111].

How then does one come up with the sequence of a diploid individual using ancient DNA? A simple technique is to produce a single pseudo-haplotype by randomly sampling from one of the reads at each SNP position. Assuming that homologous chromosomes decay at the same rate, the output pseudo-haplotype will contain approximately half of the variants of each chromosome. However, this method breaks up haplotype patterns which could confound some haplotype based statistics.

6.1.2.5 Conclusion: Ancient DNA

Ancient DNA is useful for evolutionary biology because it enables researchers to directly study past genetic diversity. The utility of ancient DNA comes at the cost of having to deal with its associated technical challenges. Due to post-mortem DNA damage, ancient

DNA can have a high rate of missing data and the sequences we produce are more error prone than in modern data. Since ancient DNA is commonly sequenced using DNA chips, the data suffers from ascertainment bias which distorts the site frequency spectrum. Unlike modern data, ancient DNA is often unphased meaning that the haplotypes of the samples cannot be discerned. Instead researchers resort to using pseudo-haplotypes by randomly sampling from one of the reads at each SNP position. The aim of this chapter is to extend our method from Chapter 5 to account for these technical challenges in ancient DNA. We will focus on using single-timepoint samples from an ancient population. A sweep detection protocol which works with ancient DNA will enable researchers to identify old sweep patterns not found in modern data. This will provide further insights into how populations have adapted to past environmental challenges and how the genome evolves across time.

6.2 Method

6.2.1 Introduction

Section 6.1 covered the utility of ancient DNA in evolutionary biology and the technical challenges that it brings. We now extend our methods from Chapter 5 in order to formulate a machine learning protocol for detecting sweeps in ancient DNA. First we will age our training data by introducing missing information, deamination and pseudo-haplodisation. To account for the wide use of DNA chips in ancient DNA, we will also simulate the ascertainment procedure as described in Section 6.1.2.3. We will then explore some imputation strategies for missing information and preprocessing methods for preparing the data for model fitting. We will use our results to design a machine learning protocol for detecting selective sweeps in ancient DNA.

6.2.2 Simulating DNA Ageing

6.2.2.1 Initial Simulation of Training Data

We used discoal to simulate a simple demographic model with a constant population size of 10,000 with a fixed mutation rate of 1.5×10^{-8} and a fixed recombination rate of 1.0×10^{-8} [71]. All sweeps were fixed at the time of sampling. Unlike Chapter 5, this model contains an outgroup which diverged from the main population $50,000$ years ago. We used a generation time of 25 years [112]. This demographic model coincides with a simple model of human evolutionary history. The two populations represent African and non-African populations of modern humans which diverged $\sim 50,000$ years ago [113].

Each simulation is a 1Mb region. 100 samples are taken from the main population whilst 20 samples were taken from the outgroup. The outgroup sequences will be used for SNP ascertainment to mirror ascertainment bias as described in Section 6.1.2.3. At this stage, the simulated genomes represent the clean, modern DNA sequences that researchers would extract from living individuals. We will now discuss how to age our training data by introducing the technical challenges of ancient DNA described in Section 6.1.2.

6.2.2.2 Aging the Training Data

In order to train an effective model for detecting sweeps in ancient data, we must modify our population genetic simulations to resemble ancient genetic data. We focus on four key technical challenges of ancient DNA namely; missing information, deamination, pseudo-haplodisation and SNP ascertainment bias.

6.2.2.3 Simulating Ascertainment Bias

The first step for converting our training data into ancient DNA is to simulate the ascertainment procedure described in Section 6.1.2.3. The justification is that ascertainment is done using a set of modern ascertainment individuals which should not have any ancient DNA damage [114]. For example, researchers may use modern African genomes (ascertainment set) to ascertain SNPs for some ancient hominid genomes from Europe. Hence we must simulate the ascertainment procedure before adding any ancient DNA damage.

We now discuss how to simulate ascertainment in our training data. We start off with a genome matrix $G_{120\times k}$, where k is the number of SNPs produced in a particular simulation. The rows correspond to the 120 samples; 100 from the main population and 20 from the outgroup. The 20 outgroup samples are the ascertainment individuals which will be used to ascertain SNPs for the main population (*i.e.* sample of interest). The 20 outgroup sequences are grouped into ten pairs of rows. Two sequences from the main population also become a pair, giving 11 pairs of rows in total. Each pair of rows represent the two sequences from a single diploid individual. For each row pair/individual, we identify all the heterozygous sites (*i.e.* any pairwise differences). We then create a new, truncated genome matrix using only the columns of G which were heterozygous in at least one of the ascertainment individuals. The 22 rows corresponding to the ascertainment individuals are removed as their only purpose is to ascertain SNPs. This replicates the real-world process of genotype array construction where the ascertainment individuals are not part of the sample of interest [114].

The downstream analysis is done using the remaining genomes sampled from our population of interest. After simulating ascertainment bias, we have a truncated genome matrix $G^{asc}_{98\times l}$, for some $l \leq k$.

6.2.2.4 Simulating Missing Information

Since endogenous ancient DNA fragments are short and difficult to retrieve, ancient genomes often have missing sequences. When aligning ancient genomes together for population genetic analysis, each individual genome is likely to have missing sequences at different SNP positions. We can represent this data by modifying the binary genome matrices from Chapter 5 to contain some NA elements, representing missing information. Assuming we are using ancient DNA extracted from samples of similar age and condition, each sequence should have roughly the same amount of DNA damage. In this case the sequences are considered to be missing completely at random whereby each element of the genome matrix is equally likely to become missing. We simulated missingness in our genome matrices by using a single Bernoulli trial for each matrix element. The probability of success is determined by a fixed missing rate. For each success the corresponding matrix element is converted to NA, otherwise the element remains unchanged. In order to understand the effects of missingness on predictive accuracy, we applied a range of missing rates onto the training data.

6.2.2.5 Simulating Deamination

Deamination is a form of post-mortem DNA damage which produces $G \longrightarrow A$ and $C \longrightarrow T$ substitutions at various positions across the genome known as transition sites. It has been shown that out of the $\sim$ 1.2 million SNP sites that are commonly targeted in the ancient DNA analysis of humans, $\sim$ 77.6% are transition sites [115]. Hence for each genome matrix, we designate each column to be a transition site with probability 77.6%. For each transition column, we designate either a 0-to-1 or a 1-to-0 transition with equal probability. This reflects how either the 0 or 1 alleles could represent a G/C which can be changed into a A/C via deamination. The relevant entries (either 0 or 1) in that column are then changed into the alternate allele with probability 0.05 [115]. Although it is possible for deamination to generate a third variant at a position which already has a SNP, this was not modelled. Using the ascertainment individuals, we know that two alleles exist per variant site. Thus when a third variant is found, we know that it has been caused by an error and it will be removed.

This simulation of deamination made no decision about which nucleotide base each variant represented. The reasons are

1. Coalescent simulations uses the infinite sites model where alleles are either ancestral or derived (*i.e.* 0 or 1). Each variant is equally likely to be A,C,T or G.

2. The population genetic summary statistics we use are based on the differences between sequences rather than the actual underlying base sequences themselves.

6.2.2.6 Simulating Pseudo-haplodisation

Recall that in a genome matrix, the columns represents the SNPs whilst the rows represent individual sequences. We simulate the process described in Section 6.1.2.4 by breaking up the rows of each genome matrix into pairs. Each pair of rows represents the two homologous chromosomes within a single, diploid individual. For each row pair, we randomly sample one allele for each position to generate a new row. This will only make a difference at heterozygous sites where there are two different alleles (*i.e.*The elements are 0/1 rather than 1/1 or 0/0). The new row represents a pseudo-haplotype where each homologous chromosome has contributed to half of its variants. After pseudo-haplodisation, we have a shorter genome matrix which has half the number of rows as the original input matrix. The number of columns is unchanged. There may occasionally be non-polymorphic columns where all the non-NA elements are either 1's or 0's. All non-polymorphic columns are removed from the genome matrix as they are no longer genetic variants. After ageing our simulations, the end product is a set of genome matrices with fewer rows than those in Chapter 5 (49 vs 100) and generally fewer columns. This is due to ascertainment bias and the removal of non-polymorphic positions.

6.2.3 Imputation

By taking our simulations through the ageing process described in Section 6.2.2, we converted binary genome matrices (containing 0's and 1's) into smaller genome matrices with 0's, 1's and NAs. In statistics, imputation is the process of substituting missing data with reasonable values [116]. Since many traditional population genetic summary statistics are not defined for missing data, we designed a couple of imputation strategies for our simulated aDNA data. It is not necessary to impute every matrix element correctly because the goal is not to retrieve the original matrix. Rather we want to impute sensible values to the missing data so that our summary statistics will separate hard sweeps from neutral simulations.

6.2.3.1 Strategy 1: Zero Impute

For regions that are neutral or under weak selection, ancestral alleles (represented by 0) are much more common than derived alleles (represented by 1). A naive imputation strategy would be to convert all NA's to 0 since this would be the "correct" value for most sites. Whilst this strategy is probably too simplistic, it's worth trying as a "first pass" and will provide a base line for comparison.

6.2.3.2 Strategy 2: Random Impute

This method imputes the NA's of each column using the information present in each column. Suppose some column c in the genome matrix has k NA's and n non-NA elements. For each NA, we randomly sample one element out of the n non-NA elements. The NA element is then imputed to be the sampled non-NA element. This process is repeated across all columns of the genome matrix which contain any NA's.

6.2.4 Window Splitting and Computing Summary Statistics

We split each genome matrix into 5 roughly equal blocks using the method described in Figure 5.2. Although 11 blocks were used in Chapter 5, this time we only used 5 blocks because the DNA ageing process removed many columns from the genome matrices. (have a boxplot comparing the number of SNPs between the two groups for different missing rates) We generated a dataframe by computing population genetic summary statistics across the 5 blocks for each simulation, similar to the process described in Section 5.1.3. The key difference with the aDNA method is that LD based summary statistics were not used. The reason is that computing the LD statistics takes much longer than the other statistics, even when we downsample the columns of our genome matrices. Given the additional time required to simulate DNA ageing, computing the LD statistics would be excessive. Instead our method uses the SFS stats (Tajima's D, Fay and Wu's H) and the haplotype statistics (h1,h2,h12,h123).

6.2.5 Modifying the Haplotype Statistics

Recall from Section 2.5.2, that the haplotype statistics distinguish selective sweeps from neutral simulations using the proportions of the different haplotypes (unique rows) in the data. In the aDNA context, one major problem we encountered was that the haplotype statistics had near-zero variance across the blocks and failed to distinguish hard

sweeps from neutral simulations Figure 6.5. This was because deamination, pseudo-haplodisation and the imputation of missingness have made most/all the rows of the genome matrix different, even when the selection coefficient (*i.e.* strength of selection) is high. Currently, our haplotype statistics treat rows as being separate haplotypes as long as there is at least one pairwise difference between them. It has no inherent measure of how different two rows may be. The result is that the haplotype statistics are unable to detect sweep patterns due to the noise introduced by DNA ageing.

We considered clustering techniques to mitigate this problem. The idea is that each row of a genome matrix is considered as a p-dimensional vector, where p is the number of SNPs for that simulation. A clustering algorithm from machine learning is used to cluster the rows into k clusters, based on the number of pairwise differences between them. The proportion of rows that are within each group are computed and ranked from highest to lowest. (*i.e.* p_k is the proportion of rows that are in the k^{th} biggest cluster). These proportions are then used in place of the haplotype proportions for the purposes of computing the haplotype statistics. For example, consider a genome matrix with 10 rows which are grouped into 3 clusters. 5 rows are in cluster 1, 3 in cluster 2, 2 in cluster 3. The modified h1 statistic for this matrix is $0.5^2 + 0.3^2 + 0.2^2 = 0.38$.

We designed two different clustering strategies.

1. Fixed k method: As a simple first approach, we clustered every genome matrix block into k clusters using the k-means algorithm (Section 3.3.2), where k is a fixed value determined *a priori*. We noted that areas with strong selection can have multiple duplicated rows. For any block which has d duplicated rows where $d < k$, the pipeline will only form d clusters. Considering each genome matrix have 49 rows, we chose $k = 10$ which is 20% of the rows.

2. Silhouette based method: A limitation of method 1 is that the selection of k is somewhat arbitrary. Given the different haplotype patterns produced by different selection coefficients, using the same k across all the simulations may also not be appropriate. For example, a neutral region should have multiple haplotypes which would warrant a large k. A region under strong selection will have fewer haplotypes so a small k would be sensible. One strategy to address this problem is to try different values of k for each block and select the k which provides the "best" clustering. This is similar to the hyperparameter tuning process described in Section 3.2.2.1.

In method 2, we cluster each block for $k = 2, 3, \ldots, 10$, using the k-means algorithm (Section 3.3.2). Each clustering is assessed using the silhouette value which measures how similar the haplotypes are to its own cluster (cohesion) compared to the other clusters (separation). The k which corresponds to the highest silhouette value is chosen

for the final clustering. The cluster-based haplotype statistics is computed using the proportions of rows for each cluster in the final clustering (as per method 1).

Using these clustering techniques, we hope to capture the haplotype patterns produced by hard sweeps despite the noise produced by the DNA ageing process.

6.2.6 Simulation Parameters

TABLE 6.1: Table of demographic parameters for discoal simulations. Note that the recombination and mutation rates are measured in events/base/generation. Values were based on [72, 73, 74, 112, 113]

Parameter	**Value**
mutation rate	$1.5 \times 10^{-8} bp^{-1} gen^{-1}$
recombination rate	$1 \times 10^{-8} bp^{-1} gen^{-1}$
effective population size	10 000
genome length	10^6
sample size	120
selection coefficient	0, 0.005, 0.0125, 0.025, 0.0375, 0.05, 0.1
number of populations	2
population split time	50 000 years
generation time	25 years

Out of the 120 samples generated, 100 are from the population of interest and 20 from the outgroup (Table 6.1). SNPs are ascertained using 2 samples from the former and the 20 samples from the outgroup. We made 1000 simulations for each selection coefficient.

TABLE 6.2: Table of aDNA damage parameters. The values were taken from [115]. The missing rate was modified so we could consider a wider range.

Parameter	**Value**
missing rate	0,0.2,0.4,0.6,0.8,0.95
proportion of transition sites	0.776
deamination rate	0.05

6.2.7 Preprocessing

After ageing our simulations and computing summary statistics, we obtain a dataframe with response variable "sweep" and 30 predictors. Each row corresponds to one aged discoal simulation which has been processed with the imputation and clustering techniques described in Section 6.2.3 and Section 6.2.5. 30 predictors come from the 6 population genetic summary statistics (Tajima's D, Fay and Wu's H, h1, h2, h12, h123) computed over the 5 blocks. There is a non-modelling, factor variable "processing technique"

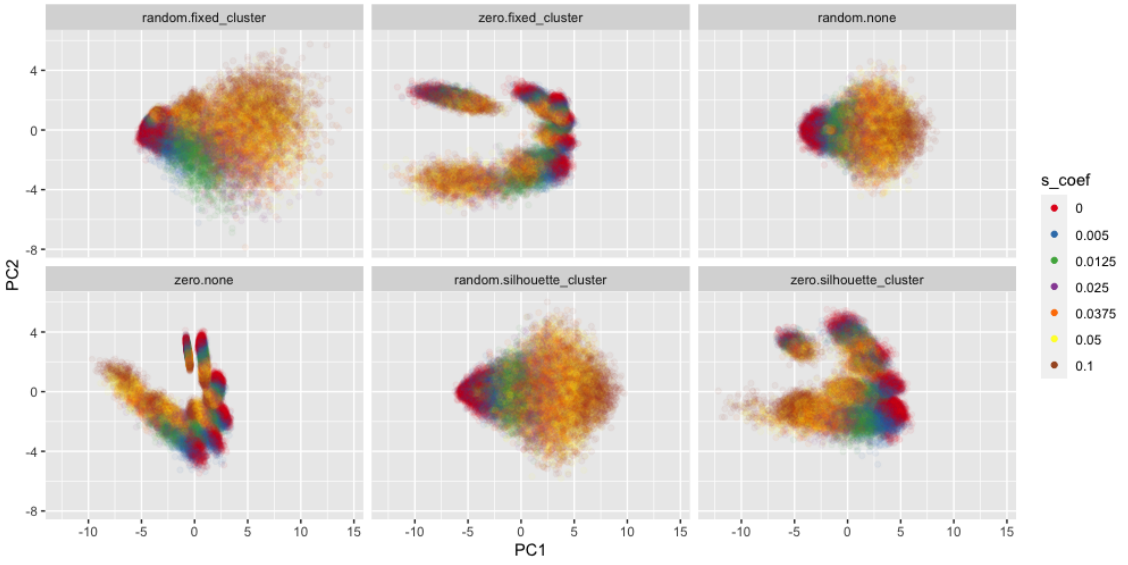

FIGURE 6.2: PCA plots for the training data processed with 2 imputation techniques (zero, random) and 3 clustering techniques (none, fixed clustering, silhouette clustering). The colors represent the different selection coefficients.

which indicates the imputation and clustering method used to generate each data point. There are 6 techniques in total because we have 2 imputation methods (zero impute, random impute) and the 3 clustering methods (no clustering, fixed clustering, silhouette clustering). The simulated data was partitioned into 6 separate dataframes, one for each processing technique. Each dataframe was split 80:20 into training and testing sets. For preprocessing, we applied a 0.8 correlation filter and standardised [1] the SFS-based statistics. As with Section 5.1.4, there was no need to standardise the haplotype statistics because they are bounded by 0 and 1. This ensures that all the predictors are roughly on the same scale.

6.2.8 Principal Component Analysis

Before deciding on which machine learning classifier to fit, we used PCA to visualise the spread of the data. Figure 6.2 is a PCA plot of our ancient DNA data set processed with the 6 different techniques (2 imputation strategies by 3 clustering strategies). The general trend for all techniques is that the hard sweeps tend to move further apart from neutral data points as the selection coefficient increases. Neutral data points also have less variability than hard sweeps.

For zero impute, the neutral simulations form several small clusters spread out across each plot. The complex spread of the neutral data suggests that a flexible classifier which

[1] subtract mean and divide by standard deviation

can model non-linear trends would be preferable. For random impute, the neutral simulations are concentrated on a single cluster. There is some overlap with weaker selection coefficients (0.005, 0.0125) but separation is good overall, particularly for higher selection coefficients (≥ 0.025). The spread of the data suggests a linear boundary on PC1 can separate hard sweeps from neutral data reasonably well. The cleaner separation of the random impute data suggests that it may be the better imputation technique. We will now fit a machine learning classifier to separate hard sweeps from neutral simulations.

6.2.9 Model Fitting

After preprocessing we have 6 total training dataframes, one for each processing technique. For each dataframe, we fitted a MARS model using the forward pass algorithm Section 3.3.5. Unlike Chapter 5, we have far fewer predictors since we used 6 summary statistics (8 before) computed over 5 blocks (11 before). To ensure a parsimonious model, higher order interaction terms were not included. For hyperparameter tuning, we considered a 5-level regular grid for "num_terms" $\in [4, 12]$. We determined the upper bound based on the number of summary statistics we used, noting the strong correlations when the same statistic is computed on adjacent blocks. Although more sophisticated classifiers are available, we selected MARS for this chapter because of its good performance overall in Chapter 5. The PCA plots suggest that a non-linear classifier may be useful, especially for zero imputation. For this context, MARS seems suitable because it can model non-linear trends whilst maintaining computational speed. As we found in Chapter 5, simple methods combined with preprocessing can produce effective, predictive models.

6.3 Results

6.3.1 DNA Ageing Effect On Summary Statistics

We will now examine some boxplots to see how well our summary statistics can distinguish hard sweeps from neutral simulations. We will focus on the central window (Window 3). This is the location of the selected mutation where the sweep signature tends to be the strongest.

6.3.1.1 Tajima's D

Figure 6.3 shows how Tajima's D changes with missing rate. Section 2.5.1.3 explained that hard sweeps have lower values of Tajima's D compared to neutral regions. For both

imputation strategies, the separation between hard and neutral simulations is strongest when the missing rate is low. The interquartile ranges (IQR) decrease as missing rate rate goes up. The decline is stronger for hard sweeps due to the variation in selection coefficients used. Zero impute has a non-linear relationship between missing rate and Tajima's D. The hard sweeps start off having lower values than neutral simulations with the two IQR's converging around missing rate 0.4. For higher missing rates, the hard sweeps have higher D values. This means zero impute shows poor separation for intermediate missing rates ($\sim 0.4 - 0.5$). Random impute consistently shows the hard sweeps to have lower D values across all missing rates. There is better separation relative to zero impute, since there is no overlap in the IQR's across the different missing rates. For both imputation techniques, there are some outlier points from the hard sweeps with values within the neutral range. This corresponds to hard sweep simulations with low selection coefficients (*e.g.* 0.005). In regards to Tajima's D, random impute seems to be the better imputation technique because it shows good separation across all the missing rates. The linear relationship between D and missing rate means that it can be captured by simple, linear models like logistic regression.

6.3.1.2 Fay and Wu's H

Hard sweeps are tend to have lower values of H relative to neutral simulations (Section 2.5.1.4). See Figure 6.4. For random impute, H has a negative, linear relationship with missing rate whilst the hard sweeps have a positive, non-linear relationship. For zero impute, H has a non-linear relationship with missing rate for both neutral and hard sweeps. Similar to Tajima's D, the IQRs for both groups and both imputation techniques decline as missing rate increases. The decline is stronger for hard sweeps because of the different selection coefficients. Random impute shows greater separation as the IQRs do not overlap until missing rate 0.9 (zero impute shows overlap for 0.8). The spread of IQRs for the two groups suggest that a linear classifier would perform better for random impute compared to zero impute. As random impute also shows better separation at high missing rates (0.7-0.9), random impute may be the better imputation strategy.

6.3.1.3 Haplotype Statistics

Recall that our modified h1 statistic is the sum of the proportions of haplotypes assigned to each cluster. Hard sweeps are supposed to have higher h1 values because there is a dominant, selected haplotype. See Figure 6.5. When the haplotypes are not clustered, the hard sweeps and neutral simulations have the same fixed values. The imputation makes no difference. As explained in Section 6.2.5, this is because the ageing process

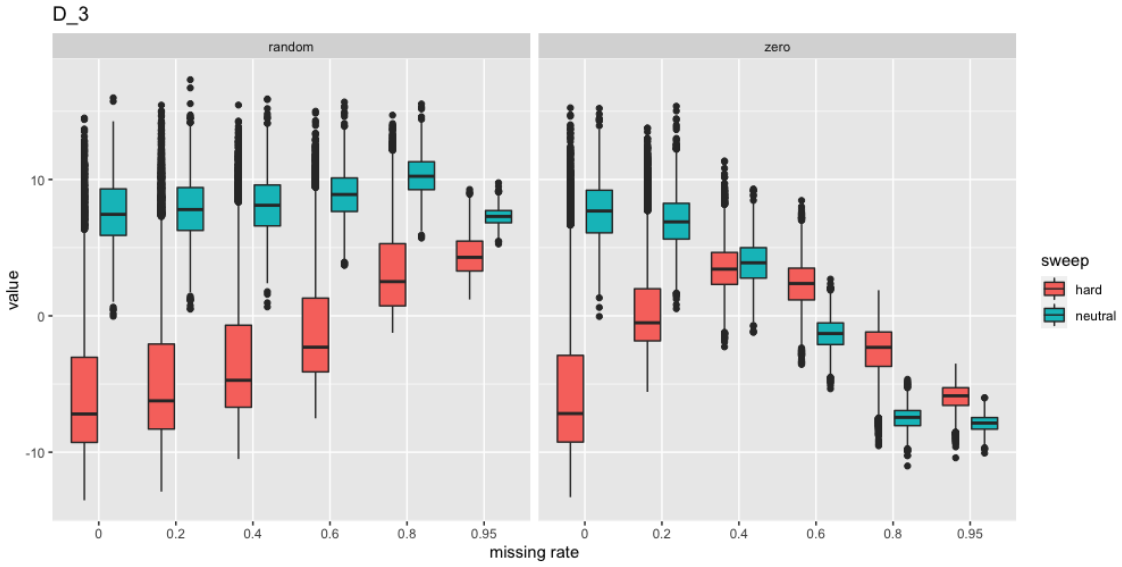

FIGURE 6.3: Boxplot showing how Tajima's D on the central window changes with different missing rates for the two imputation methods

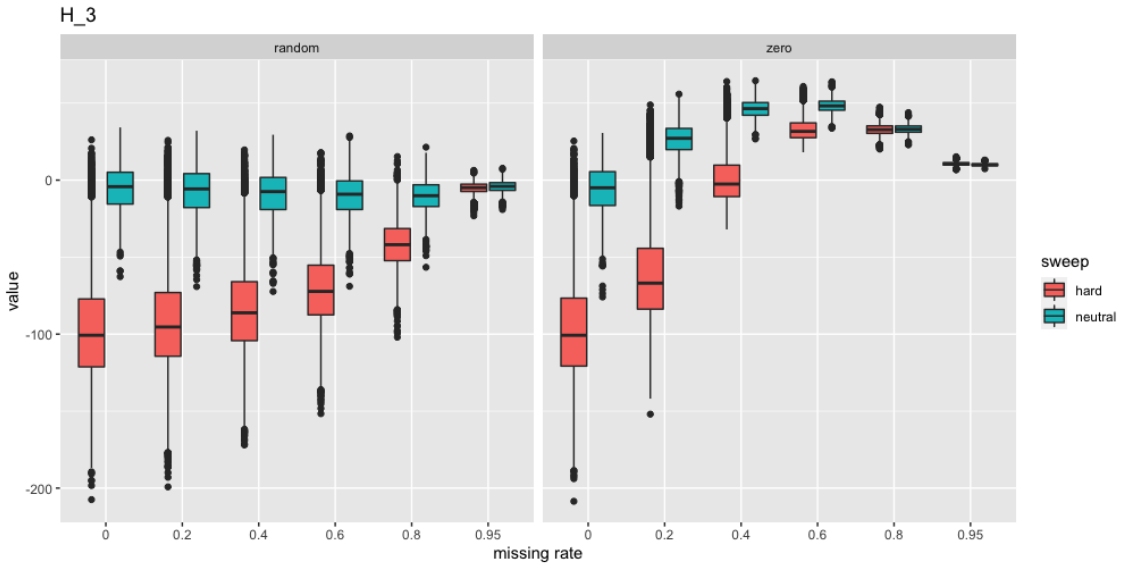

FIGURE 6.4: Boxplot showing how Fay and Wu's H on the central window changes with different missing rates for the two imputation methods

renders all the haplotypes different, whether there was a sweep or not. When every haplotype is different, h1 becomes close to zero.

Zero impute with silhouette clustering increases the variability of h1 across both groups. This is probably caused by variation in the number of clusters used for each data point. Separation is good up to a missing rate of 0.2, past which the IQRs of the hard sweeps and neutral simulations become overlapping. Random impute works better with silhouette clustering, showing good separation for missing rates up to 0.8.

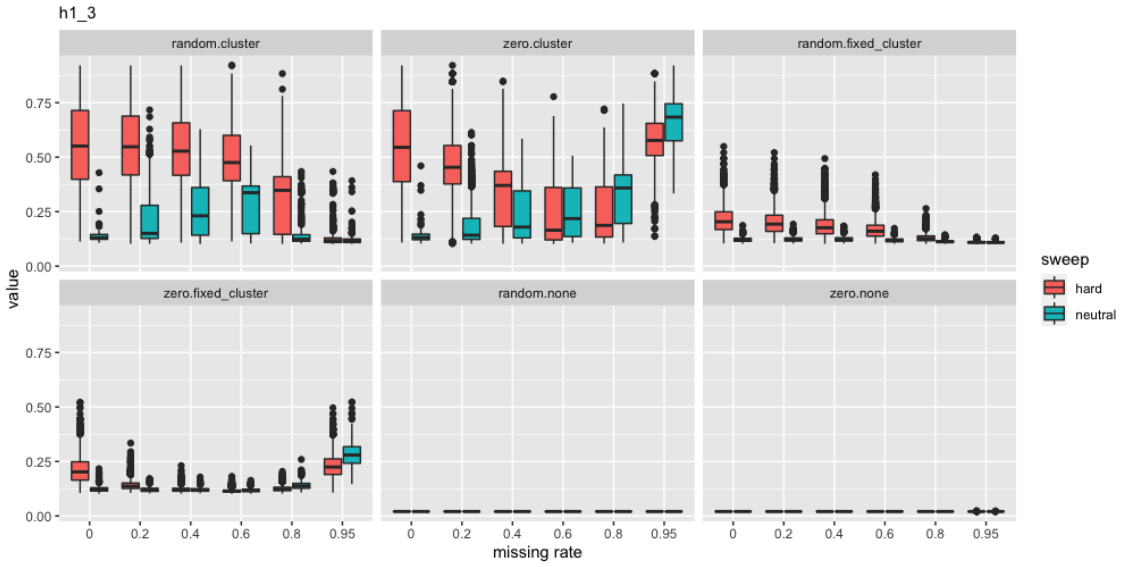

FIGURE 6.5: Boxplot showing how h1 on the central window changes with different missing rates for all 6 processing techniques (imputation and clustering). The imputation methods are random and zero impute. The clustering methods are fixed clustering with k = 10 and silhouette clustering. "none" indicates no clustering was done.

For zero impute/fixed cluster, h1 is higher for extreme missing rates ([0,0.2], [0.8,0.95]) relative to intermediate missing rates. At extreme missing rates, setting all NAs to zero produces similar looking rows thereby increasing h1. Although hard sweeps tend to have higher h1 values than neutral simulations at low missing rates (≤ 0.2), the two groups start to converge around missing rate 0.4. For missing rate > 0.8, the neutral simulations have higher h1. The reason is that the genome matrices of neutral simulations have mostly 0's due to the lack of a selected mutation. When missing rate is high, many of the 1's become NAs and get imputed to be 0's. This produces many similar rows (lots of 0's) which then get clustered together, resulting in a larger h1. Hard sweeps have more derives alleles so that more 1's will remain after adding missingness. This creates more pairwise differences between the haplotypes, thereby giving a lower h1 than neutral simulations. The non-linear spread of the two groups for zero impute(both clustering) and random impute, silhouette clustering, suggest that a linear classifier would not be adequate for this method.

The boxplots for the remaining haplotype statistics can be found in Appendix B. h2 showed poor separation and was uninformative (Figure B.1). This reflects how h2 was designed primarily for picking up soft sweeps. The plots for h12 were similar to those of h1, although the separation is poorer (Figure B.2). This reflects how the

Random impute/fixed clustering consistently shows neutral simulations to have lower values with small IQR across the missing rates. The IQR of hard sweeps decreases with increasing missing rate. There is good separation between the two groups until missing

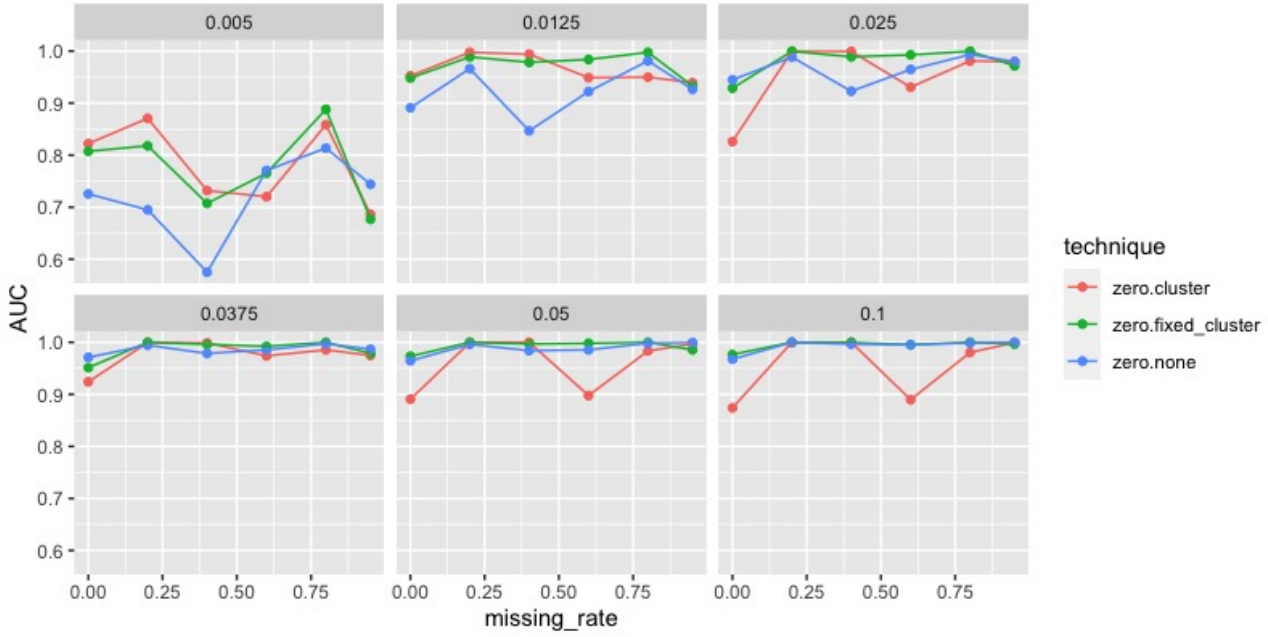

FIGURE 6.6: AUC plots for the classifiers trained on the zero impute data. The plots show the AUC values across the different missing rates. The numbers above each plot indicate the selection coefficient.

rate 0.9, when the two IQRs overlap. The spread of the data suggests that a linear classifier would separate the two groups up to a missing rate of 0.8. This suggests that random impute and fixed clustering is the best strategy out of the ones considered.

6.3.2 Model Accuracy

TABLE 6.3: Table of AUC values for the MARS fitted on data processed with each of the imputation and clustering techniques

Technique	**AUC**
random impute, fixed clustering	0.9664358
zero impute, fixed clustering	0.9504123
zero impute, silhouette clustering	0.9321226
zero impute, no clustering	0.9292167
random impute, no clustering	0.9085888
random impute, silhouette clustering	0.8956103

Table 6.3 shows the accuracy of the 6 models using a test data set containing equal numbers of simulations from the 6 missing rates and selection coefficients (7 if we include neutral). The fixed clustering method produced the most accurate for both imputation techniques. When fixed clustering is used, random impute has 0.01 more AUC which suggests that it may be the better technique overall.

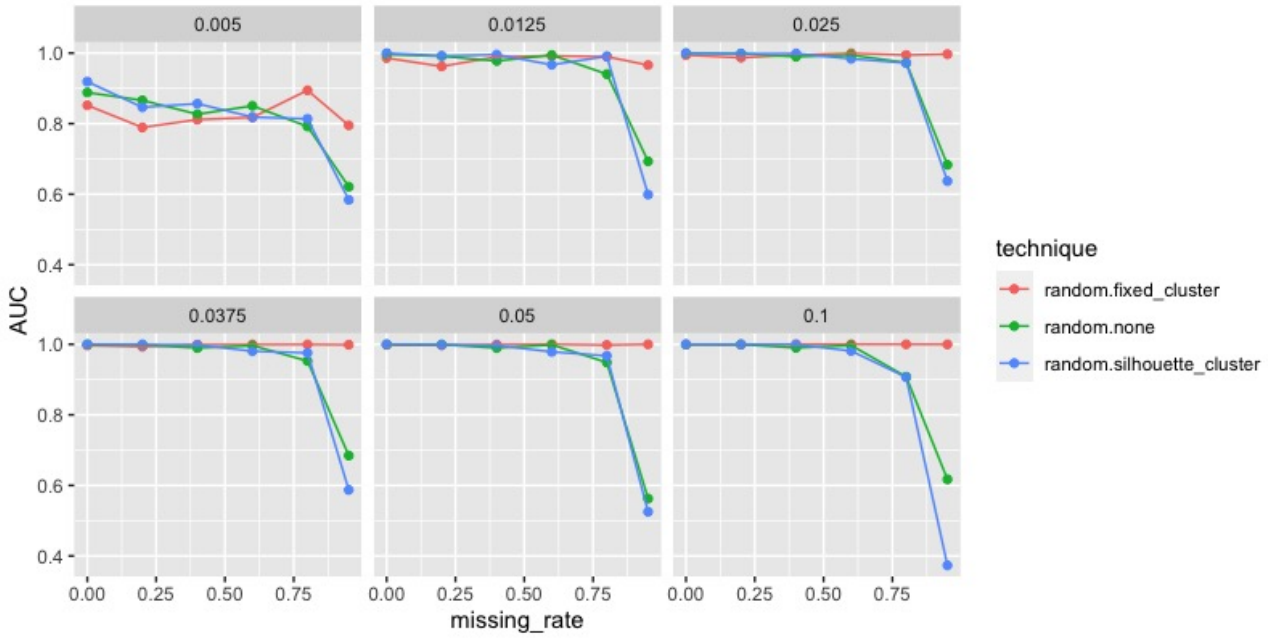

FIGURE 6.7: AUC plots for the classifiers trained on the random impute data. The plots show the AUC values across the different missing rates. The numbers above each plot indicate the selection coefficient.

6.3.2.1 Assessing Zero Imputation

Figure 6.6 shows the breakdown of the AUC for the zero impute models at the six different missing rates and selection coefficients. At the smallest selection coefficient (0.005), most of the AUC values fluctuate between $\sim 0.7 - 0.9$. The model produced with no clustering performs the worst with the AUC going below 0.6 at missing rate 0.4. No trend can be discerned between missing rate and AUC. All three models are poor at distinguishing hard sweeps and neutral simulations when the selection coefficient is 0.005, regardless the missing rate. This suggests that when selection is very weak, the hard sweep signal is too faint to be distinguished by our summary statistics.

The AUC tends to increase and stabilise as the selection coefficient goes up. However, there is substantial fluctuation of AUC at selection coefficients 0.0125 and 0.025, particularly when there is no clustering or silhouette clustering. Silhouette clustering appears to be the worst technique, with the AUC dropping to ~ 0.9 for some missing rates even at strong selection strengths (0.05, 0.1). Both no clustering and fixed clustering have consistently high AUC when the selection coefficient is at least 0.0375. Considering the results across all six selection strengths, zero imputation goes best with fixed clustering because it more accurate at weaker selection coefficients (0.0125, 0.025).

6.3.2.2 Assessing Random Imputation

Similar to zero imputation, the random impute models were inaccurate at detecting sweeps at the lowest selection coefficient. There is less fluctuation of AUC across the missing rates. Most AUC values are within 0.8 - 0.9, although it does dip at missing rate 0.95. Considering the results across all the missing rates, random impute models are more accurate than zero impute models at selection coefficient 0.005. Compared to the similar case in modern data, both aDNA models performed substantially worse. Figure A.6 shows the AUC of last chapter's MARS model when tested on data with different selection coefficients. Observe that the previous MARS model achieved an AUC of $\sim$ 0.978, even when it had to deal with population bottlenecks. This shows that our model struggles to retrieve a strong sweep signal after DNA ageing when the selection is weak.

All three models improve substantially for the higher selection coefficients (0.0125 - 0.1), achieving AUC values close to 1 for at least 5 of the 6 missing rates. There is some small drops in AUC for lower missing rates at selection coefficient 0.0125 but this is improved at higher selection strengths. Models with no clustering and silhouette clustering showed poorer performance at high rates (0.8-0.95). Particularly, they show a consistent, large drop in AUC for missing rate 0.95. The fixed clustering model was more consistent, maintaining a high AUC across all the missing rates, including 0.95. This suggests that when random imputation is used, the fixed clustering method is useful for maintaining model accuracy at high missing rates (0.8-0.95). Note that silhouette clustering performed worse than fixed clustering, particularly at higher missing rates (≥ 0.6). It seems that by tuning the number of clusters for each observation, silhouette clustering caused more variability in h1, particularly among the neutral simulations (Figure 6.5). This produced more overlap between the two groups in h1 at the higher missing rates, making the data more difficult to separate. In contrast, fixed clustering increased h1 for hard sweeps, whilst keeping it small for neutral simulations. This reduced the overlap between the two groups at h1, making separation easier.

6.3.2.3 Preferred Method

Although both imputation methods struggled to detect hard sweeps at selection coefficient 0.005, random impute performed better.

The preferred imputation method is random impute because its three models showed consistently high AUC across the missing rates. The caveat is that the AUC dropped substantially for silhouette clustering and no clustering at missing rate 0.95. Random

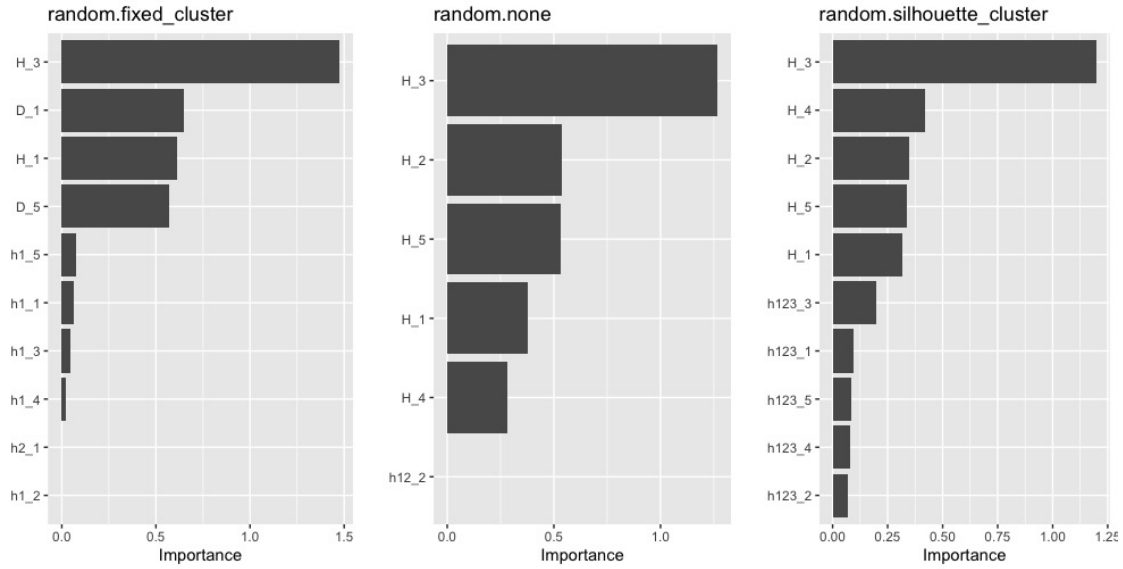

FIGURE 6.8: Boxplot showing the FIRM scores for the random impute three models. The numbers after the underscore indicate the window where the summary statistic was computed. There were 5 windows in total. Not all predictors are included in the final models because they can be removed via the correlation filter or excluded by the MARS algorithm.

impute seems to go best with fixed clustering because it maintains high AUC even for missing rate 0.95. This method is also better than zero imputation with fixed clustering which shows drops in AUC, particularly for selection coefficients 0.0125 and 0.025. Thus given our data set, random imputation with fixed clustering is our preferred method for detecting hard sweeps in ancient DNA. Its high accuracy can be explained by the good separation of Tajima's D, Fay and Wu's H and h1 shown by the box plots in Figures 6.3 to 6.5. The next section will use variable importance tools to investigate this model further.

6.3.3 Variables of Importance

6.3.3.1 FIRM Scores of Random Impute Models

We now explore the variables of importance used by the random impute models. Figure 6.8 shows the FIRM scores of the most important predictors for each of the three models. A predictor with zero importance means that it survived the correlation filter but was not selected to be a term by the MARS algorithm. Fay and Wu's H plays a prominent role, with "H_3" (H computed at Window 3/central window) having the highest FIRM score for all three models. We observed in Section 5.2.1.2 that SFS-based statistics (*e.g.* Tajima's D, Fay and Wu's H) best separate hard sweeps at the window with the selected mutation. Since the selected mutation is on Window 3, "H_3" was an

important predictor for all 3 models. This is consistent with Figure 6.4 which showed good separation at "H_3" when using random imputation.

When no clustering is used (random.none), the important predictors are the H values across the 5 windows. None of the D's appear in the model as they were removed by the correlation filter. The small differences in FIRM between the outer windows (1,2,4,5) are likely due to statistic noise. The haplotype statistics played no role because they are zero variance predictors with poor separation (Figure 6.5). Given random.none's performance in Figure 6.7, the FIRM scores suggest that H alone does a good job of distinguishing hard sweeps when the selection coefficient is ≥ 0.0125 and the missing rate ≤ 0.8.

When silhouette clustering is introduced, only H, h2 and h123 survived the correlation filter. The 5 values of H and h123 made it to the final model. However, the h123 FIRM scores were much lower than that of H. Recall from Figure 6.7 that the silhouette clustering model performed similarly to the no clustering model for most missing rates. At missing rate 0.95, silhouette clustering made the model more inaccurate. This can be explained by the silhouette model using h123 which is a statistic designed for distinguishing soft sweeps from hard sweeps, rather than detecting hard sweeps from neutral simulations. By latching onto noise from an inappropriate statistic, the silhouette model performed worse than the no clustering model.

After applying fixed clustering, some D and H values survived the correlation filter in addition to all five values of h1 and h2. The final model selected a combination of Tajima's D, H and h1 as predictors. H on the central window ("H_3") had the highest FIRM, followed by D and H on the outer windows. Although "D_3" by itself should be more informative than D's on the outer windows (for the same reason as H), it was removed by the correlation filter. This means that "D_3" was not needed when "H_3" is already present. Four values of h1 were included in the model, although their FIRM scores were low relative to those of D and H. If any of the four haplotype statistics were to be included, it makes sense for h1 to be chosen because it was specifically designed for picking up hard sweeps. Given the h1 FIRM scores are so small, the model is making predictions mostly based on H and D. Recall from Figure 6.7 that random imputation with fixed clustering produced the most accurate model overall. The FIRM scores suggests the MARS model can distinguish sweeps with high accuracy, even at missing rate 0.95 using mostly H and D. Fixed clustering appears to retrieve some signal from h1 after the noise introduced via the DNA ageing process. However, h1 only plays a minor role in the overall prediction.

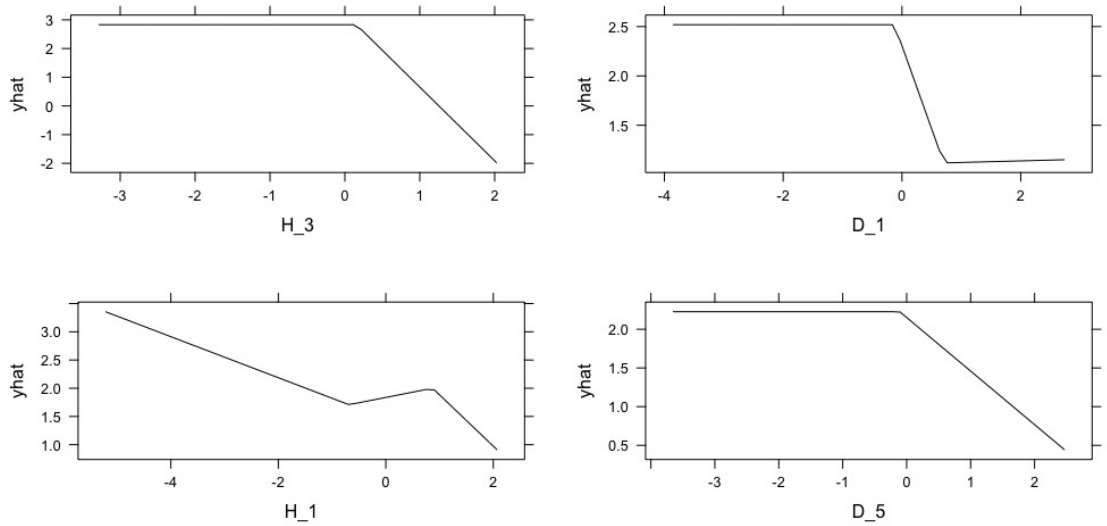

FIGURE 6.9: Partial dependence plots for the four most important predictors in the preferred model (random impute/fixed clustering). "yhat" is the predicted log odds of being a hard sweep. Note the values of D and H were normalised during preprocessing.

6.3.3.2 Investigating Random Impute/Fixed Clustering Model

We now investigate how our preferred model (random impute/fixed clustering) is making its predictions. Figure 6.9 shows the partial dependence plots for the predictors with the top 4 FIRM scores.

The pdp for the most important statistic "H_3", has the highest range (-2,3). This reflects how "H_3" has the highest FIRM score since FIRM measures the variability of the pdp. The predicted log odds of a hard sweep is high when "H_3" is negative. The prediction decreases linearly as "H_3" increases beyond 0. This fits with Figure 6.4 which showed hard sweeps to have very negative values of "H_3" whilst neutral simulation had values close to 0. "D_5" shows the same trend as "H_3" but except the pdp for "D_5" has a smaller range. This is due to Window 1 being further away from the center (Window 3) and hence its D value is less informative. The pdp for "D_1" drops more quickly past 0. Theoretically, the spread of D values at Windows 1 and 5 should be approximately the same since the region is symmetric (although Window 5 can have up to 4 more columns). The difference in pdp's may be caused by stratification in the data. This means D splits the data so well such that multiple hinge functions can fit well to the data (insert boxplot for D1 when I get home). Since the general trend for "D_1" is the same as "D_5" and "H_3", these pdp's make sense from a population genetics perspective.

The pdp for "H_1" is different because the predicted log odds decreases with increasing "H_1" but is never flat. Figure 6.10 is a boxplot showing the distribution of "H_1"

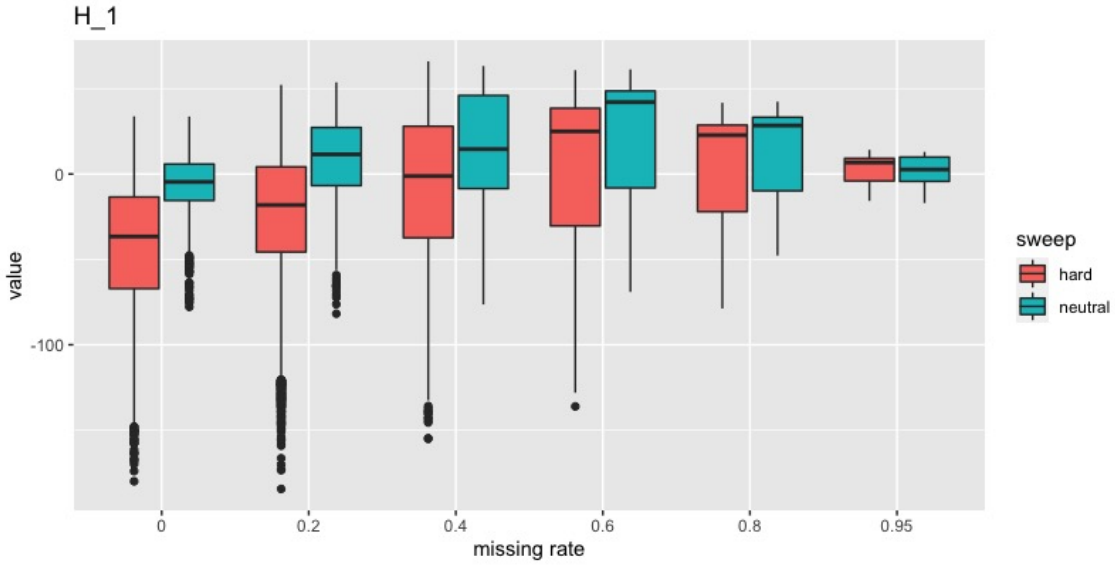

FIGURE 6.10: Boxplot for "H_1" in the random impute data.

across the missing rates. There is worse separation here relative to Figure 6.4, with the IQRs starting to overlap even at missing rate 0. The overlap between hard and soft sweeps increases with missing rate, with the IQR's being almost identical at missing rate 0.95. The quicker drop in the pdp of "H_1" may be caused by the poorer separation of "H_1". The predicted log odds declines quicker because the difference between hard sweeps and neutral simulations is smaller for "H_1" than it is for "H_3". The prediction also increases slightly between $(-0.8, 0.8)$ before dropping again. This kink is minor and is likely caused by statistical noise. Overall, the pdp for "H_1" suggests that the chance of a hard sweep increases as "H_1" gets negative.

6.4 Sweep Detection After DNA Ageing

In this chapter, we have simulated data under a constant population size model and aged the DNA. The ageing process simulated the technical challenges of working with ancient DNA such as missing information, pseudo-haplodisation, deamination and ascertainment bias. We trialled methods for imputing missingness and clustering pseudo-haplotype data. Based on our findings, we make the following remarks about detecting hard sweep in ancient DNA under this simple demographic model.

1. For selection coefficient 0.005, the sweep signal is too low to be picked up reliably by MARS.

2. For selection coefficients ≥ 0.0125, MARS coupled with random imputation can still retrieve a clear signal for detecting sweeps. The accuracy tends to decrease as missing rate goes up. Sweeps are easier to detect when the selection coefficient is high.

3. The accuracy of our models tended to drop when the missing rate went past 0.8. MARS coupled with random imputation and fixed clustering produced a classifier that maintained high accuracy across the missing rates (including 0.95, which is not unusual for ancient DNA [115]). This is our preferred method.

4. The most useful statistics are Tajima's D and Fay and Wu's H. h1 was used in our preferred model but it only played a minor role in the predictions.

6.5 Conclusion

This chapter explained the technical challenges of using ancient DNA for population genetic analysis, particularly in the context of detecting selective sweeps. We generated genomes from a constant population size model and simulated missing information, deamination, pseudo-haplodisation and ascertainment bias. We explored how to use supervised learning together with imputation and clustering techniques, to detect hard sweeps in ancient DNA. Our findings show that MARS used together with random imputation and fixed clustering can retrieve a clear sweep signal after DNA ageing. Tajima's D and Fay and Wu's H were the most useful statistics, although h1 may play a small role as well. Chapter 7 will frame our work in the wider context of ancient DNA research. Suggestions will be made for machine learning can be extended to finding sweeps in ancient DNA under more complicated scenarios.

Chapter 7

Conclusion

Chapter 1 outlined our project for using machine learning to detect selective sweeps. Here we also provided public links to our R package and pipeline which we wrote for this project. The biological background was covered in Chapter 2 which discussed key concepts such as the nature of DNA, evolution via natural selection, selective sweeps and population genetics summary statistics.

Chapter 3 gave a technical introduction to machine learning, explaining fundamental ideas such as supervised and unsupervised learning, classification and regression, the bias variance trade off, regularisation and how to train and evaluate our models. For unsupervised learning, we covered principal component analysis and k-means clustering which are useful techniques for exploratory data analysis. We also covered four machine learning classifiers:

1. Regularised logistic regression: A technique for modeling linear trends which is also simple to interpret.
2. MARS: A non-parametric technique for modelling more complex, non-linear patterns.
3. RDA: A decision boundary method based on a discriminant function.
4. Random forests: An ensemble method using classification trees.

This covered some of the main classes of classification techniques commonly used in machine learning. The concepts from the two background chapters were brought together in Chapter 4 which discussed current machine learning approaches for sweep detection. We identified two main knowledge gaps.

1. There is no standard machine learning protocol for detecting selective sweeps. Researchers tend to use methods in isolation based on hype or what they are familiar with. There is little understanding about which machine learning method is appropriate for any particular situation.

2. Current methods only use modern DNA which have been sampled from living individuals. We are interested in applying our methods to the emerging field of ancient DNA. Ancient DNA is useful in biology because it enables researchers to directly study the genomes of past populations. Using ancient DNA will enhance our understanding of how populations have evolved across time.

Chapter 5 addressed the first knowledge gap by testing the performance of a suite of machine learning classifiers. We focused on population bottleneck models since these are known to produce false positives in previous sweep detection methods. We designed a machine learning workflow which involved data preprocessing, hyperparameter tuning and comparing our models by considering their predictive accuracy, computational time and inherent model interpretability. We also used FIRM scores and partial dependence plots to understand how our different models made their predictions. Most of our models relied heavily on SFS-based statistics (D and H) although regularised logistic regression used ω_{max}.

We found that an accurate sweep detection model can be produced using simple, quick methods (*e.g.* regularised logistic regression) supported by data preprocessing. This shows that researchers should not immediately deploy elaborate and computationally intensive techniques such as neural networks. Instead we recommend researchers to first try some simple, quick techniques and only move onto more elaborate ones if the simple techniques prove to be insufficient.

Whilst our workflow focused on detecting hard sweeps, it can be generalised for other population genetic problems. Researchers can use our workflow to compare different machine learning methods and select one that is suitable for their problem. Our workflow ensures researchers not only build accurate and efficient predictive models but also understand how their models work.

Chapter 6 addressed the second knowledge gap. This chapter first reviewed the additional technical challenges of using ancient DNA due to various forms of post-mortem DNA damage. As a good starting point, we use genome data sampled from a constant population size model. We aged this data by simulating key characteristics of aDNA data such as missing information, deamination, pseudo-haplodisation and ascertainment bias. We explored some imputation and clustering strategies retrieve the sweep signal to account for deamination, missing data, pseudo-haplodisation and ascertainment bias

found in ancient DNA. Using our random imputation technique combined with fixed clustering, we built a MARS model which maintained a high accuracy even at missing rate 0.95. This shows that a clear sweep signal can be retrieved despite all the noise introduced via DNA ageing. The most important statistics were D and H, with the model predicting hard sweeps when these values were small. Although h1 was also used, it only played a minor role. The model distinguished hard sweeps by mostly looking for negative values of H and D.

Detecting selective sweeps is an important and active area of research in population genetics. Here are some reasons for why population geneticists may be interested in our work.

1. Researchers have previously used various machine learning classifiers in isolation for detecting sweeps in modern genomes. However, there is little understanding of which classifiers would be appropriate for given problem. In Chapter 5, we formulated a standard machine learning protocol for researchers to follow. Researchers can use our protocol to compare different methods and to select a suitable classifier for their specific research topic.

2. As we mentioned in Chapter 4, sometimes researchers are eager to use sophisticated and flexible machine learning classifiers which require immense computational resources (*e.g.* deep learning neural networks). At least for the range of demographic models we considered (constant population size and population bottlenecks), we showed that accurate models can be produced using simple, quick classifiers supported by summary statistics and data preprocessing. Although more sophisticated classifiers may still be required for more complicated demographic scenarios, our findings suggest that researchers should at least try some simpler methods first. This sets a performance baseline to compare with other more sophisticated methods if the simpler methods prove to be insufficient.

3. We incorporated novel techniques to investigate variables of importance in our workflow. It is noteworthy that these techniques are model agnostic. This enables researchers to not only make accurate predictive models but also to understand how their models make predictions, not matter which machine learning algorithm they are using. By unravelling the relationships between the predictors and the response, researchers can study the different kinds of genetics patterns that are produced by selective sweeps. This information allows researchers to better understand the robustness of their methods towards demographic model misspecification and to improve their classifiers with summary statistics or preprocessing regimes.

4. Previous sweep detection methods only leverage modern DNA samples. A key contribution is that we extended machine learning to detect sweeps in ancient DNA. This was achieved by designing and implementing novel imputation and clustering strategies for ancient DNA data. Given the caveat that we focused on a simple demographic model, we showed that a clear sweep signal can still be achieved when much of the DNA is absent due to damage (missing rate > 0.8). Whilst more research is needed to extend this to more realistic and complicated scenarios, our work shows the potential of using machine learning to detect sweeps in ancient DNA.

7.0.1 Future Research Directions

Chapter 4 showed that many previous applications of machine learning on sweep detection focused on deploying a specific classifier onto a particular data set. A novel aspect of our project is that we built machine learning models for sweep detection from the ground up. This involved reviewing a suite of classifiers and designing a workflow to compare, select, assess and interpret different models. This section will discuss the limitations of our project and identify several future research directions which can extend our methods to work with more complicated, real world data sets.

7.0.1.1 Simplifying Assumptions

1. We made simplifying assumptions to our simulated data such as fixed mutation and recombination rates. In practice, these rates may vary across the genome [117]. In the future, we could simulate data from a distribution of rates and investigate how they may affect model accuracy. To ensure our methods are robust, we could also include additional summary statistics such as the nSL statistic which have been shown to be robust to different recombination rates [118]. We did not use nSL in our project because it is computationally demanding to compute and our models performed well without it. For more complicated demographies, we could investigate the trade-off between making the model more robust to recombination rate variation vs. computational efficiency.

2. As a starting point, we considered simple demographic models where there is either a constant population size or a single population bottleneck. Bottlenecks were included because they are known to produce false positives [75]. Real populations can have more complex demographies which may include expanding/shrinking population sizes (rather than immediate changes as with bottlenecks) or interbreeding

between relatively separate populations (admixture) [119, 120]. In order to extend our methods to account for realistic scenarios, we could select a population of interest and estimate a set of feasible demographic histories using demographic modelling tools. We can then simulate data under these scenarios and analyse the results using our proposed workflow.

3. In our simulated data, all our hard sweeps were fixed at the time of sampling for the sake of simplicity. This explains the high accuracy of our predictive models. In real biological settings, the hard sweeps are likely to have been fixed at a range of different times. Since sweep patterns decay over time, older sweeps are more difficult to detect. We could use a range of fixation times in our simulations to train a suite of classifiers. We can then investigate how fixation time affects prediction and at what point is a sweep too old to be accurately distinguished.

4. To simplify the problem, we treated the missing data in ancient DNA to be missing completely at random [116] (Chapter 1). This process was simulated by independently converting each element in the genome matrix to NA with some specified probability. In practice, not every sample will have the same proportion of missing data. Some regions may be more difficult to map than others, resulting in heterogeneity in missing rates across different sites [121]. An improvement would be to use empirical data to estimate a probabilistic model of how missingness is distributed between individuals and across different sites. We can then simulate missingness in our data under this probabilistic model.

5. In Chapter 6, we focused on four key aspects of ancient DNA; namely deamination, missing information, ascertainment bias and pseudo-haplotypes. There are other important technical challenges we have not considered such as contamination from foreign individuals. We could estimate a probabilistic model of contamination and use it simulate DNA ageing [122]. This would involve allowing some individuals in the sample to come from a modern population (*e.g.* a modern European individual instead of a ancient human).

   Population geneticists may wish to use a combination of ancient and modern genomes to enhance their analyses. Since ancient specimens vary in age, this means the data will contain sequences taken from a range of time points. A potential solution is to first estimate the demographic history using population genetic tools. We input the histories into discoal and simulate sequences according to their estimated dates. DNA damage can be introduced into the ancient genomes using the method described in Chapter 6. We can partition the genome matrix so that each partition only contains sequences from the same time. Summary statistics are computed separately for each partition. Thus for data with k different time

points and some statistic f, we compute $f_1, f_2, \ldots, f_k$. We can then use models which can account for repeated measures [123].

7.0.1.2 Expanding To Other Types Of Selection

Our project focused on hard sweeps because it is the most well-studied form for selection. However, population geneticists are also interested in other kinds of selection such as soft sweeps and balancing selection [124]. Supervised learning could be used to develop a multinomial classifier for selection. This tool would scan through the genome and output a predicted probability for each class (all types of selection + neutral). This would involve incorporating additional summary statistics for detecting the new types of selection. For example, we could use $\frac{h2}{h1}$, $\frac{h12}{h1}$ and $\frac{h123}{h1}$ for soft sweeps and the B statistics for balancing selection [125].

For a classification task with n classes, a classifier can output $n-1$ predicted probabilities since the total probability must sum to one. This makes investigating variables of importance more complex for multinomial models. Suppose we made a classifier to distinguish hard sweeps, soft sweeps, balancing selection and neutral simulations. First we set "neutral" as the reference class. For each predictor, we can output three pdp's (or ICE plots) corresponding to the predicted probabilities of three selection patterns. Similarly, a FIRM score can be computed for each selection pattern.

7.0.1.3 Model Assessment

In both Chapter 5 and Chapter 6, we tuned our models by picking the highest cross validation accuracy and assessed our final models using AUC. This approach has the following limitations.

1. Both performance metrics take a balanced approach towards penalising the amount of false positives (FP) and false negatives (FN) (Figure 3.5). From a population genetics perspective, minimising FP is arguably more important than minimising FN. Genomes are typically very large relative to the 1Mb windows we used for sweep detection (*e.g.* human haploid genome is over 3 billion bases long) [8]. This means that a mode with FPR ~ 0.98 will incorrectly identifying $\sim 600,000$ neutral regions as sweeps. This is inefficient and can mislead researchers into wasting resources on studying neutral regions which they think are sweeps. False negatives are not as harmful provided the models can correctly identify some selective sweeps. A solution would be to modify our performance metrics into putting more weight on the FPR rather than FNR. We could experiment on different weightings to

see how it may affect hyperparameter tuning, variable importance and overall predictions.

2. We made the same number of simulations for each of our seven selection coefficients, one of which is 0. This produces imbalanced classes because $\frac{6}{7}$ of the data are hard sweeps. CV accuracy can be a misleading metric because a naive model which classifies all observations as hard sweeps will still have an accuracy of $\sim 85\%$. An improvement would be to tune and evaluate our models using a metric designed to handle imbalanced class problems such as Matthew's correlation coefficient [126]. To address the previous point, we could also modify the metric to put more weight on FP.

Appendix A

Chapter 4 Appendix

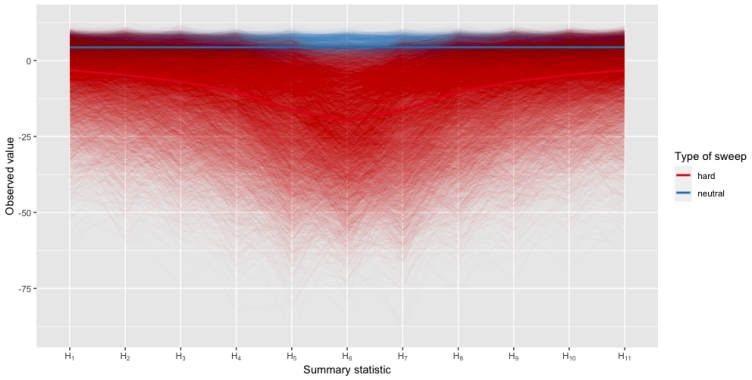

FIGURE A.1: Parallel coordinates plot for Fay and Wu's H. Each faded line is a simulation. Red lines are hard sweeps. Blue lines are neutral simulations. The solid lines represent the mean values for hard and neutral simulations.

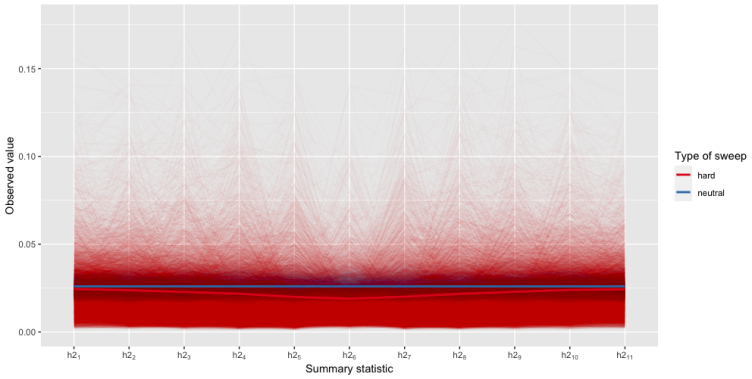

FIGURE A.2: Parallel coordinates plot for h2. Each faded line is a simulation. Red lines are hard sweeps. Blue lines are neutral simulations. The solid lines represent the mean values for each group.

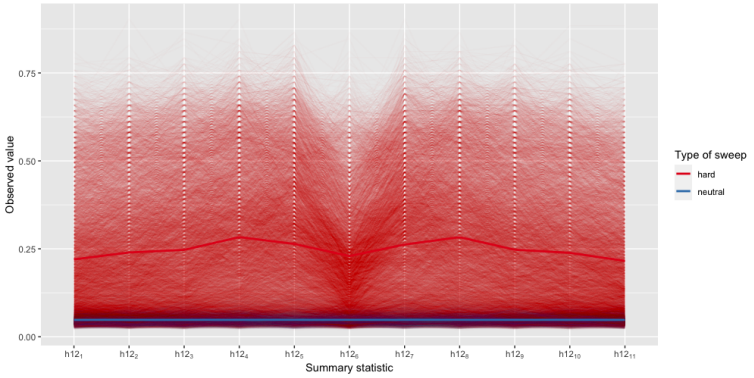

FIGURE A.3: Parallel coordinates plot for h12. Each faded line is a simulation. Red lines are hard sweeps. Blue lines are neutral simulations. The solid lines represent the mean values for hard and neutral simulations.

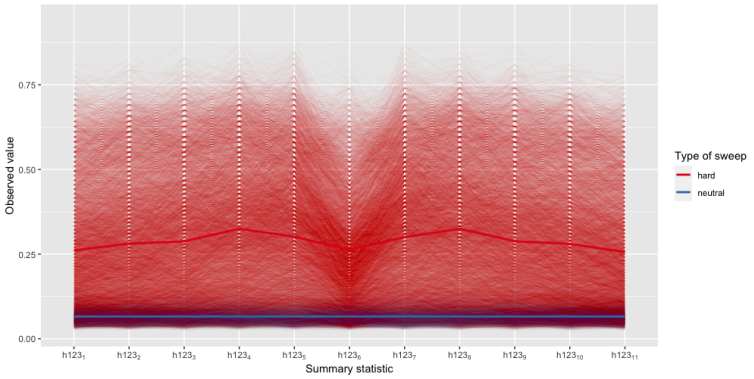

FIGURE A.4: Parallel coordinates plot for h123. Each faded line is a simulation. Red lines are hard sweeps. Blue lines are neutral simulations. The solid lines represent the mean values for hard and neutral simulations.

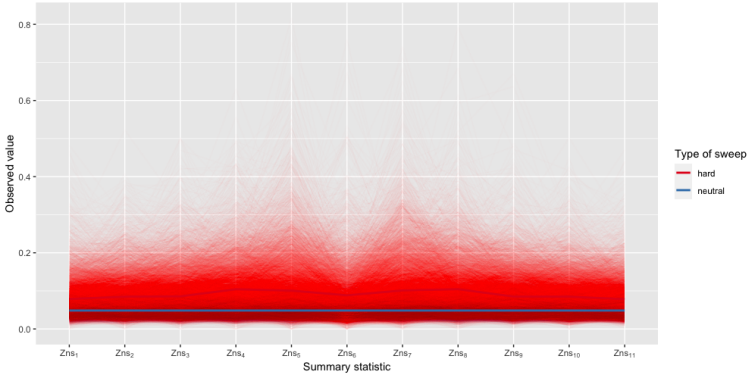

FIGURE A.5: Parallel coordinates plot for Kelly's Z_{nS}. Each faded line is a simulation. Red lines are hard sweeps. Blue lines are neutral simulations. The solid lines represent the mean values for hard and neutral simulations.

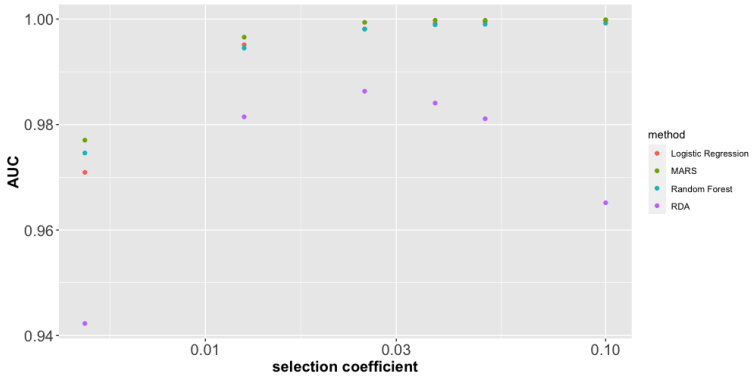

FIGURE A.6: Plot of the AUC achieved by each classifier, across different missing rates.

Appendix B

Chapter 5 Appendix

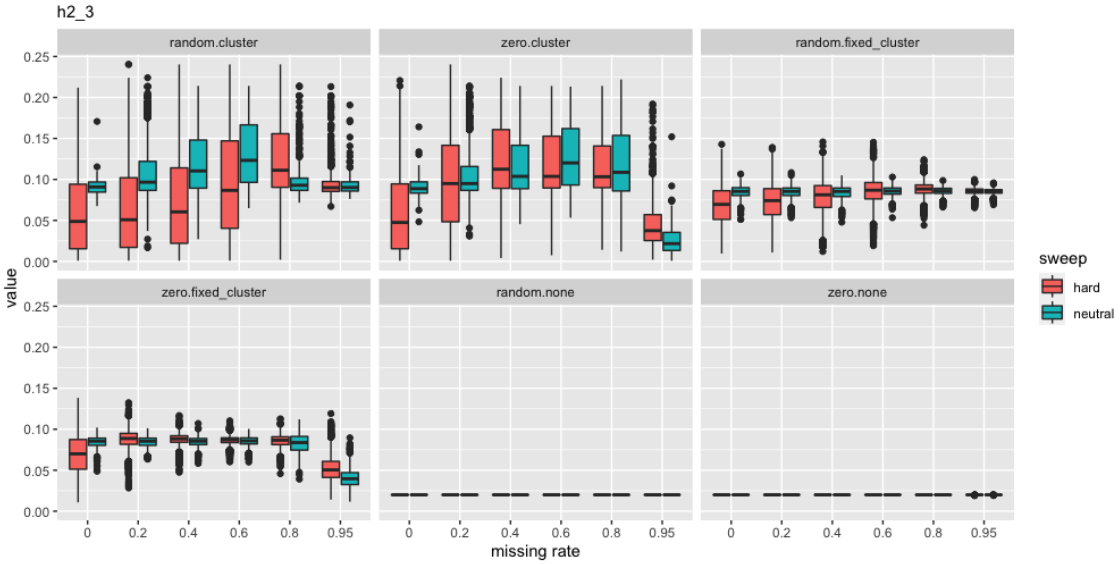

FIGURE B.1: Boxplot showing how h2 on the central window changes with different missing rates for all 6 processing techniques (imputation and clustering). The imputation methods are random and zero impute. The clustering methods are fixed clustering with k = 10 and silhouette clustering. "none" indicates no clustering was done.

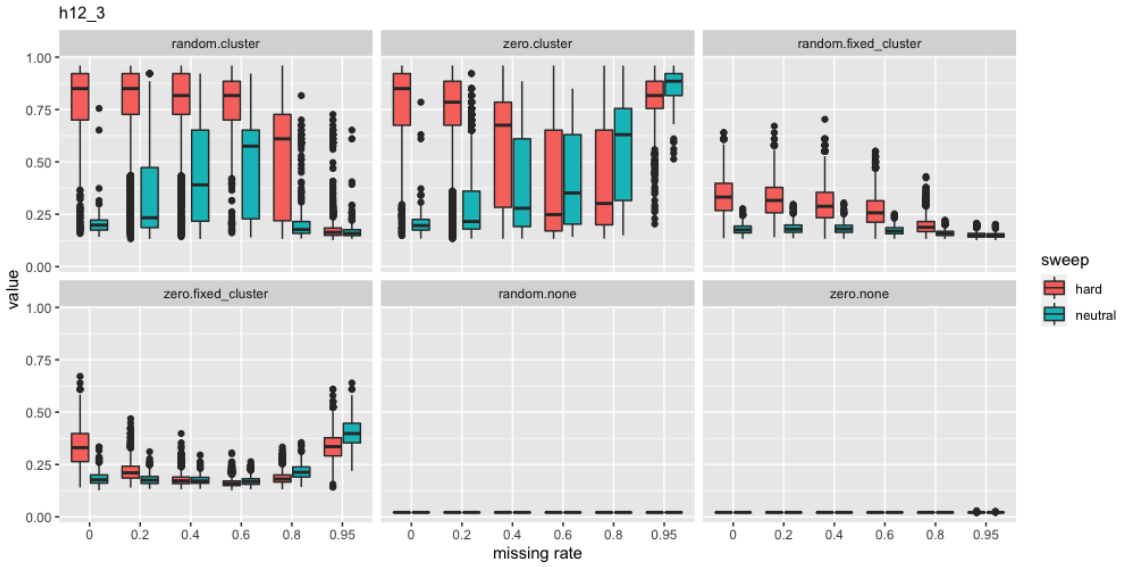

FIGURE B.2: Boxplot showing how h12 on the central window changes with different missing rates for all 6 processing techniques (imputation and clustering). The imputation methods are random and zero impute. The clustering methods are fixed clustering with k = 10 and silhouette clustering. "none" indicates no clustering was done.

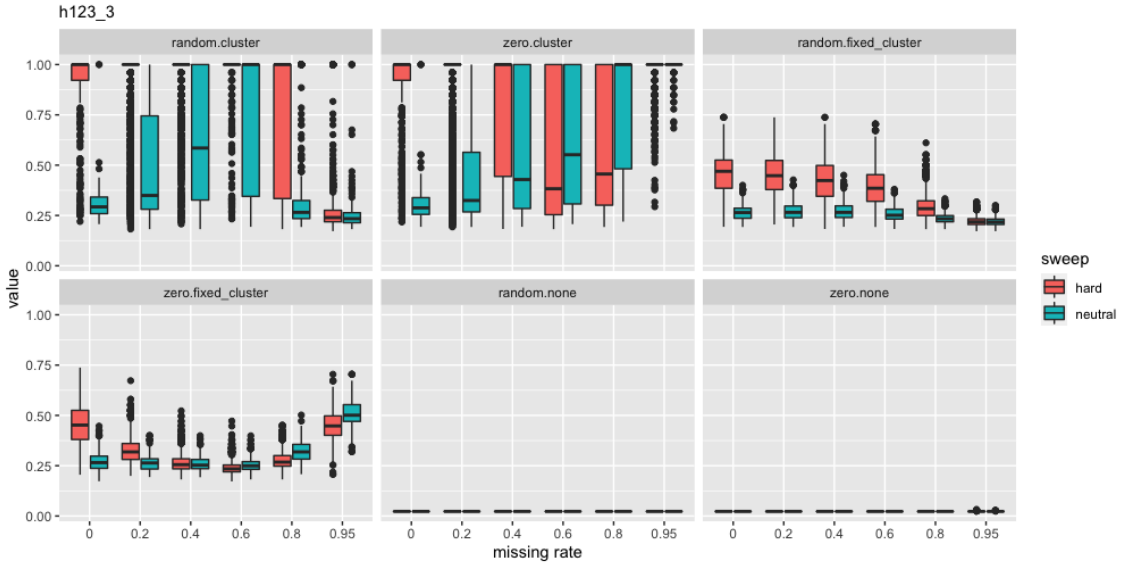

FIGURE B.3: Boxplot showing how h123 on the central window changes with different missing rates for all 6 processing techniques (imputation and clustering). The imputation methods are random and zero impute. The clustering methods are fixed clustering with k = 10 and silhouette clustering. "none" indicates no clustering was done.

Bibliography

[1] Allison Marie Horst, Alison Presmanes Hill, and Kristen B Gorman. *Palmerpenguins: Palmer Archipelago (Antarctica) Penguin Data*, 2020. URL https://allisonhorst.github.io/palmerpenguins/. R package version 0.1.0.

[2] A. D. Hershey and Martha Chase. Independent functions of viral protein and nucleic acid in growth of bacteriophage. *The Journal of General Physiology*, 36(1): 39–56, 1952. doi: 10.1085/jgp.36.1.39. URL http://jgp.rupress.org/content/jgp/36/1/39.full.pdf.

[3] Martin Thanbichler, Sherry C. Wang, and Lucy Shapiro. The bacterial nucleoid: A highly organized and dynamic structure. *Journal of Cellular Biochemistry*, 96(3):506–521, 2005. ISSN 0730-2312. doi: 10.1002/jcb.20519. URL https://doi.org/10.1002/jcb.20519.

[4] RG Jensen and JA Bassham. Photosynthesis by isolated chloroplasts. *Proceedings of the National Academy of Sciences of the United States of America*, 56(4):1095, 1966.

[5] Katrin Henze and William Martin. Essence of mitochondria. *Nature*, 426(6963): 127–128, 2003.

[6] Colin M. Hammond, Caroline B. Strømme, Hongda Huang, Dinshaw J. Patel, and Anja Groth. Histone chaperone networks shaping chromatin function. *Nature Reviews Molecular Cell Biology*, 18:141, 2017. doi: 10.1038/nrm.2016.159. URL https://doi.org/10.1038/nrm.2016.159.

[7] Aylwyn Scally and Richard Durbin. Revising the human mutation rate: implications for understanding human evolution. *Nature Reviews Genetics*, 13(10): 745–753, 2012.

[8] Allison Piovesan, Maria Chiara Pelleri, Francesca Antonaros, Pierluigi Strippoli, Maria Caracausi, and Lorenza Vitale. On the length, weight and gc content of the human genome. *BMC Research Notes*, 12(1):106, 2019.

[9] Chris P Ponting and Ross C Hardison. What fraction of the human genome is functional? *Genome Research*, 21(11):1769–1776, 2011.

[10] Chris M Rands, Stephen Meader, Chris P Ponting, and Gerton Lunter. 8.2% of the human genome is constrained: variation in rates of turnover across functional element classes in the human lineage. *PLoS Genetics*, 10(7):e1004525, 2014.

[11] Matthew S Meselson and Charles M Radding. A general model for genetic recombination. *Proceedings of the National Academy of Sciences*, 72(1):358–361, 1975.

[12] Marie-Theres Gansauge, Tobias Gerber, Isabelle Glocke, Petra Korlević, Laurin Lippik, Sarah Nagel, Lara Maria Riehl, Anna Schmidt, and Matthias Meyer. Single-stranded dna library preparation from highly degraded dna using t4 dna ligase. *Nucleic Acids Research*, 45(10):e79–e79, 2017.

[13] Marie-Theres Gansauge and Matthias Meyer. Single-stranded dna library preparation for the sequencing of ancient or damaged dna. *Nature Protocols*, 8(4):737–748, 2013.

[14] Sebastian Jünemann, Fritz Joachim Sedlazeck, Karola Prior, Andreas Albersmeier, Uwe John, Jörn Kalinowski, Alexander Mellmann, Alexander Goesmann, Arndt Von Haeseler, Jens Stoye, et al. Updating benchtop sequencing performance comparison. *Nature Biotechnology*, 31(4):294–296, 2013.

[15] Wenmin Zhang, Ben Jia, and Chaochun Wei. Pass: a sequencing simulator for pacbio sequencing. *BMC Bioinformatics*, 20(1):1–7, 2019.

[16] Michael A Quail, Miriam Smith, Paul Coupland, Thomas D Otto, Simon R Harris, Thomas R Connor, Anna Bertoni, Harold P Swerdlow, and Yong Gu. A tale of three next generation sequencing platforms: comparison of ion torrent, pacific biosciences and illumina miseq sequencers. *BMC Genomics*, 13(1):1–13, 2012.

[17] Heng Li, Jue Ruan, and Richard Durbin. Mapping short dna sequencing reads and calling variants using mapping quality scores. *Genome Research*, 18(11):1851–1858, 2008.

[18] Deanna M Church, Valerie A Schneider, Tina Graves, Katherine Auger, Fiona Cunningham, Nathan Bouk, Hsiu-Chuan Chen, Richa Agarwala, William M McLaren, Graham RS Ritchie, et al. Modernizing reference genome assemblies. *PLoS Biology*, 9(7):e1001091, 2011.

[19] Heng Li. Aligning sequence reads, clone sequences and assembly contigs with bwa-mem. *arXiv preprint arXiv:1303.3997*, 2013.

[20] Alexander Dilthey, Charles Cox, Zamin Iqbal, Matthew R Nelson, and Gil McVean. Improved genome inference in the mhc using a population reference graph. *Nature Genetics*, 47(6):682–688, 2015.

[21] Benedict Paten, Adam M Novak, Jordan M Eizenga, and Erik Garrison. Genome graphs and the evolution of genome inference. *Genome Research*, 27(5):665–676, 2017.

[22] RJ Berry. Industrial melanism and peppered moths (biston betularia (l.)). *Biological Journal of the Linnean Society*, 39(4):301–322, 1990.

[23] H Bernard D Kettlewell. Selection experiments on industrial melanism in the lepidoptera. *Heredity*, 9(3):323–342, 1955.

[24] John Maynard Smith and John Haigh. The hitch-hiking effect of a favourable gene. *Genetical Research*, 23(1):23–35, 1974. ISSN 0016-6723. doi: 10.1017/S0016672300014634. URL https://www.cambridge.org/core/article/hitchhiking-effect-of-a-favourable-gene/918291A3B62BD50E1AE5C1F22165EF1B.

[25] R. Nielsen and M. Slatkin. *An Introduction to Population Genetics: Theory and Applications*. Sinauer, 2013. ISBN 9781605351537. URL https://books.google.com.au/books?id=Iy08kgEACAAJ.

[26] Pavlos Pavlidis and Nikolaos Alachiotis. A survey of methods and tools to detect recent and strong positive selection. *Journal of Biological Research-Thessaloniki*, 24(1):7, 2017.

[27] Tom R Booker, Benjamin C Jackson, and Peter D Keightley. Detecting positive selection in the genome. *BMC Biology*, 15(1):98, 2017.

[28] Jeffrey D Jensen. On the unfounded enthusiasm for soft selective sweeps. *Nature Communications*, 5(1):1–10, 2014.

[29] Rebecca B Harris, Andrew Sackman, and Jeffrey D Jensen. On the unfounded enthusiasm for soft selective sweeps ii: Examining recent evidence from humans, flies, and viruses. *PLoS Genetics*, 14(12):e1007859, 2018.

[30] Jonathan K Pritchard, Joseph K Pickrell, and Graham Coop. The genetics of human adaptation: hard sweeps, soft sweeps, and polygenic adaptation. *Current biology*, 20(4):R208–R215, 2010.

[31] Y chromosome nih: Genetics home reference. https://ghr.nlm.nih.gov/chromosome/Y Accessed: 2020-04-20.

[32] 1000 Genomes Project Consortium et al. A global reference for human genetic variation. *Nature*, 526(7571):68–74, 2015.

[33] Steven W Criscione, Yue Zhang, William Thompson, John M Sedivy, and Nicola Neretti. Transcriptional landscape of repetitive elements in normal and cancer human cells. *BMC Genomics*, 15(1):1–17, 2014.

[34] GA Watterson. On the number of segregating sites in genetical models without recombination. *Theoretical population biology*, 7(2):256–276, 1975.

[35] Adam D Leaché and Jamie R Oaks. The utility of single nucleotide polymorphism (snp) data in phylogenetics. *Annual Review of Ecology, Evolution, and Systematics*, 48:69–84, 2017.

[36] Motoo Kimura. The number of heterozygous nucleotide sites maintained in a finite population due to steady flux of mutations. *Genetics*, 61(4):893, 1969.

[37] Richard R Hudson. Generating samples under a wright–fisher neutral model of genetic variation. *Bioinformatics*, 18(2):337–338, 2002.

[38] J. Wakely. *Coalescent Theory: An Introduction*. Macmillan Learning, 2016. ISBN 9780974707754. URL https://books.google.com.au/books?id=x30RAgAACAAJ.

[39] F Tajima. Statistical method for testing the neutral mutation hypothesis by dna polymorphism. *Genetics*, 123(3):585–595, 1989. ISSN 0016-6731. URL https://www.genetics.org/content/123/3/585.

[40] Justin C Fay and Chung-I Wu. Hitchhiking under positive darwinian selection. *Genetics*, 155(3):1405–1413, 2000.

[41] Nandita R Garud, Philipp W Messer, Erkan O Buzbas, and Dmitri A Petrov. Recent selective sweeps in north american drosophila melanogaster show signatures of soft sweeps. *PLoS Genetics*, 11(2), 2015.

[42] Yuseob Kim and Rasmus Nielsen. Linkage disequilibrium as a signature of selective sweeps. *Genetics*, 167(3):1513–1524, 2004.

[43] Montgomery Slatkin. Linkage disequilibrium—understanding the evolutionary past and mapping the medical future. *Nature Reviews Genetics*, 9(6):477–485, 2008.

[44] John K Kelly. A test of neutrality based on interlocus associations. *Genetics*, 146 (3):1197–1206, 1997.

[45] Guy S Jacobs, Timothy J Sluckin, and Toomas Kivisild. Refining the use of linkage disequilibrium as a robust signature of selective sweeps. *Genetics*, 203(4): 1807–1825, 2016.

[46] G. James, D. Witten, T. Hastie, and R. Tibshirani. *An Introduction to Statistical Learning: with Applications in R*. Springer Texts in Statistics. Springer New York, 2013. ISBN 9781461471387. URL https://books.google.com.au/books?id=qcI_AAAAQBAJ.

[47] D.P. Kroese, Z.I. Botev, T. Taimre, and R. Vaisman. *Data Science and Machine Learning: Mathematical and Statistical Methods*. Chapman & Hall/CRC machine learning & pattern recognition. CRC Press, 2019. ISBN 9781138492530. URL https://books.google.com.au/books?id=hVu6xwEACAAJ.

[48] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, Second Edition*. Springer Series in Statistics. Springer New York, 2009. ISBN 9780387848587. URL https://books.google.com.au/books?id=tVIjmNS3Ob8C.

[49] F. Chollet and J.J. Allaire. *Deep Learning with R*. Manning Publications, 2018. ISBN 9781617295546. URL https://books.google.com.au/books?id=xnIRtAEACAAJ.

[50] Ron Kohavi et al. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Ijcai*, volume 14, pages 1137–1145. Montreal, Canada, 1995.

[51] Peter J Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20: 53–65, 1987.

[52] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1):1–22, 2010. URL http://www.jstatsoft.org/v33/i01/.

[53] Noah Simon, Jerome Friedman, Trevor Hastie, and Rob Tibshirani. Regularization paths for cox's proportional hazards model via coordinate descent. *Journal of Statistical Software*, 39(5):1–13, 2011. URL http://www.jstatsoft.org/v39/i05/.

[54] JR Leathwick, D Rowe, J Richardson, Jane Elith, and T Hastie. Using multivariate adaptive regression splines to predict the distributions of new zealand's freshwater diadromous fish. *Freshwater Biology*, 50(12):2034–2052, 2005.

[55] M. Kuhn and K. Johnson. *Applied Predictive Modeling*. SpringerLink : Bücher. Springer New York, 2013. ISBN 9781461468493. URL https://books.google.com.au/books?id=xYRDAAAAQBAJ.

[56] Tom Fawcett. An introduction to roc analysis. *Pattern recognition letters*, 27(8): 861–874, 2006.

[57] Christoph Molnar. *Interpretable Machine Learning*. 2019. https://christophm.github.io/interpretable-ml-book/.

[58] Brandon M Greenwell, Bradley C Boehmke, and Andrew J McCarthy. A simple and effective model-based variable importance measure. *arXiv preprint arXiv:1805.04755*, 2018.

[59] Christian A Scholbeck, Christoph Molnar, Christian Heumann, Bernd Bischl, and Giuseppe Casalicchio. Sampling, intervention, prediction, aggregation: A generalized framework for model-agnostic interpretations. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 205–216. Springer, 2019.

[60] Alex Goldstein, Adam Kapelner, Justin Bleich, and Emil Pitkin. Peeking inside the black box: Visualizing statistical learning with plots of individual conditional expectation. *Journal of Computational and Graphical Statistics*, 24(1):44–65, 2015.

[61] Kirill S Korolev, Melanie JI Müller, Nilay Karahan, Andrew W Murray, Oskar Hallatschek, and David R Nelson. Selective sweeps in growing microbial colonies. *Physical biology*, 9(2):026008, 2012.

[62] Benjamin A Wilson, Pleuni S Pennings, and Dmitri A Petrov. Soft selective sweeps in evolutionary rescue. *Genetics*, 205(4):1573–1586, 2017.

[63] Kao Lin, Haipeng Li, Christian Schlötterer, and Andreas Futschik. Distinguishing positive selection from neutral evolution: boosting the performance of summary statistics. *Genetics*, 187(1):229–244, 2011.

[64] Peter Bühlmann, Torsten Hothorn, et al. Boosting algorithms: Regularization, prediction and model fitting. *Statistical Science*, 22(4):477–505, 2007.

[65] Daniel R. Schrider and Andrew D. Kern. S/hic: Robust identification of soft and hard sweeps using machine learning. *PLoS Genetics*, 12(3):1–31, 03 2016. doi: 10.1371/journal.pgen.1005928. URL https://doi.org/10.1371/journal.pgen.1005928.

[66] Pierre Geurts, Damien Ernst, and Louis Wehenkel. Extremely randomized trees. *Machine learning*, 63(1):3–42, 2006.

[67] Michael DeGiorgio, Christian D Huber, Melissa J Hubisz, Ines Hellmann, and Rasmus Nielsen. Sweepfinder2: increased sensitivity, robustness and flexibility. *Bioinformatics*, 32(12):1895–1897, 2016.

[68] Jeffrey Chan, Valerio Perrone, Jeffrey Spence, Paul Jenkins, Sara Mathieson, and Yun Song. A likelihood-free inference framework for population genetic data using exchangeable neural networks. In *Advances in neural information processing systems*, pages 8594–8605, 2018.

[69] Lex Flagel, Yaniv Brandvain, and Daniel R Schrider. The unreasonable effectiveness of convolutional neural networks in population genetic inference. *Molecular biology and evolution*, 36(2):220–238, 2019.

[70] Jeffrey D Wall and Laurie S Stevison. Detecting recombination hotspots from patterns of linkage disequilibrium. *G3: Genes, Genomes, Genetics*, 6(8):2265–2271, 2016.

[71] Andrew D Kern and Daniel R Schrider. Discoal: flexible coalescent simulations with selection. *Bioinformatics*, 32(24):3839–3841, 2016.

[72] Laure Ségurel, Minyoung J Wyman, and Molly Przeworski. Determinants of mutation rate variation in the human germline. *Annual review of genomics and human genetics*, 15, 2014.

[73] Augustine Kong, Daniel F Gudbjartsson, Jesus Sainz, Gudrun M Jonsdottir, Sigurjon A Gudjonsson, Bjorgvin Richardsson, Sigrun Sigurdardottir, John Barnard, Bjorn Hallbeck, Gisli Masson, et al. A high-resolution recombination map of the human genome. *Nature Genetics*, 31(3):241–247, 2002.

[74] Brenna M Henn, L Luca Cavalli-Sforza, and Marcus W Feldman. The great human expansion. *Proceedings of the National Academy of Sciences*, 109(44):17758–17764, 2012.

[75] Christian D Huber, Michael DeGiorgio, Ines Hellmann, and Rasmus Nielsen. Detecting recent selective sweeps while controlling for mutation rate and background selection. *Molecular ecology*, 25(1):142–156, 2016.

[76] Richard R Hudson et al. Gene genealogies and the coalescent process. *Oxford surveys in evolutionary biology*, 7(1):44, 1990.

[77] John M Braverman, Richard R Hudson, Norman L Kaplan, Charles H Langley, and Wolfgang Stephan. The hitchhiking effect on the site frequency spectrum of dna polymorphisms. *Genetics*, 140(2):783–796, 1995.

[78] Nicola F Müller, David A Rasmussen, and Tanja Stadler. The structured coalescent and its approximations. *Molecular biology and evolution*, 34(11):2970–2981, 2017.

[79] M. Kuhn and K. Johnson. *Feature Engineering and Selection: A Practical Approach for Predictive Models*. Chapman & Hall/CRC Data Science Series. CRC Press, 2019. ISBN 9781351609463. URL https://books.google.com.au/books?id=q5alDwAAQBAJ.

[80] Carsten F Dormann, Jane Elith, Sven Bacher, Carsten Buchmann, Gudrun Carl, Gabriel Carré, Jaime R García Marquéz, Bernd Gruber, Bruno Lafourcade, Pedro J Leitao, et al. Collinearity: a review of methods to deal with it and a simulation study evaluating their performance. *Ecography*, 36(1):27–46, 2013.

[81] Philipp Probst and Anne-Laure Boulesteix. To tune or not to tune the number of trees in random forest. *The Journal of Machine Learning Research*, 18(1): 6673–6690, 2017.

[82] Andrew Nguyen, Tobin South, Nigel Bean, Jonathan Tuke, and Lewis Mitchell. Podlab at SemEval-2019 task 3: The importance of being shallow. In *Proceedings of the 13th International Workshop on Semantic Evaluation*, pages 292–296, Minneapolis, Minnesota, USA, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/S19-2050. URL https://www.aclweb.org/anthology/S19-2050.

[83] Shenglin Liu and Michael M Hansen. Psmc (pairwise sequentially markovian coalescent) analysis of rad (restriction site associated dna) sequencing data. *Molecular ecology resources*, 17(4):631–641, 2017.

[84] Stephan Schiffels and Ke Wang. Msmc and msmc2: the multiple sequentially markovian coalescent. In *Statistical Population Genomics*, pages 147–166. Humana, New York, NY, 2020.

[85] Svante Pääbo. Preservation of dna in ancient egyptian mummies. *Journal of Archaeological Science*, 12(6):411–417, 1985.

[86] Russell Higuchi, Barbara Bowman, Mary Freiberger, Oliver A Ryder, and Allan C Wilson. Dna sequences from the quagga, an extinct member of the horse family. *Nature*, 312(5991):282–284, 1984.

[87] Stephanie Marciniak and George H Perry. Harnessing ancient genomes to study the history of human adaptation. *Nature Reviews Genetics*, 18(11):659, 2017.

[88] Ludovic Orlando, Aurélien Ginolhac, Guojie Zhang, Duane Froese, Anders Albrechtsen, Mathias Stiller, Mikkel Schubert, Enrico Cappellini, Bent Petersen, Ida Moltke, et al. Recalibrating equus evolution using the genome sequence of an early middle pleistocene horse. *Nature*, 499(7456):74–78, 2013.

[89] Caitlin Simone Mudge, Rebecca Dallwitz, Bastien Llamas, and Jeremy Austin. Using ancient dna analysis and radiocarbon dating to determine the provenance of an unusual whaling artifact. *Frontiers in Ecology and Evolution*, 8:303, 2020.

[90] Richard E Green, Johannes Krause, Adrian W Briggs, Tomislav Maricic, Udo Stenzel, Martin Kircher, Nick Patterson, Heng Li, Weiwei Zhai, Markus Hsi-Yang Fritz, et al. A draft sequence of the neandertal genome. *science*, 328(5979):710–722, 2010.

[91] Montgomery Slatkin and Fernando Racimo. Ancient dna and human history. *Proceedings of the National Academy of Sciences*, 113(23):6380–6387, 2016. ISSN 0027-8424. doi: 10.1073/pnas.1524306113. URL https://www.pnas.org/content/113/23/6380.

[92] David Enard and Dmitri A Petrov. Evidence that rna viruses drove adaptive introgression between neanderthals and modern humans. *Cell*, 175(2):360–371, 2018.

[93] Richard E Green, Adrian W Briggs, Johannes Krause, Kay Prüfer, Hernán A Burbano, Michael Siebauer, Michael Lachmann, and Svante Pääbo. The neandertal genome and ancient dna authenticity. *The EMBO journal*, 28(17):2494–2502, 2009.

[94] Alan Cooper and Hendrik N Poinar. Ancient dna: do it right or not at all. *Science*, 289(5482):1139–1139, 2000.

[95] Adrian W Briggs, Udo Stenzel, Philip LF Johnson, Richard E Green, Janet Kelso, Kay Prüfer, Matthias Meyer, Johannes Krause, Michael T Ronan, Michael Lachmann, et al. Patterns of damage in genomic dna sequences from a neandertal. *Proceedings of the National Academy of Sciences*, 104(37):14616–14621, 2007.

[96] Tomas Lindahl. Instability and decay of the primary structure of dna. *nature*, 362 (6422):709–715, 1993.

[97] Jesse Dabney, Matthias Meyer, and Svante Pääbo. Ancient dna damage. *Cold Spring Harbor perspectives in biology*, 5(7):a012567, 2013.

[98] Svante Pääbo. Ancient dna: extraction, characterization, molecular cloning, and enzymatic amplification. *Proceedings of the National Academy of Sciences*, 86(6): 1939–1943, 1989.

[99] Pontus Skoglund, Bernd H Northoff, Michael V Shunkov, Anatoli P Derevianko, Svante Pääbo, Johannes Krause, and Mattias Jakobsson. Separating endogenous ancient dna from modern day contamination in a siberian neandertal. *Proceedings of the National Academy of Sciences*, 111(6):2229–2234, 2014.

[100] Kristiina Ausmees, Federico Sanchez-Quinto, Mattias Jakobsson, and Carl Nettelblad. An empirical evaluation of genotype imputation of ancient dna, 2019.

[101] R DeSalle. Implications of ancient dna for phylogenetic studies. *Experientia*, 50 (6):543–550, 1994.

[102] M Thomas P Gilbert, Jonas Binladen, Webb Miller, Carsten Wiuf, Eske Willerslev, Hendrik Poinar, John E Carlson, James H Leebens-Mack, and Stephan C Schuster. Recharacterization of ancient dna miscoding lesions: insights in the era of sequencing-by-synthesis. *Nucleic acids research*, 35(1):1–10, 2007.

[103] Michael Knapp and Michael Hofreiter. Next generation sequencing of ancient dna: requirements, strategies and perspectives. *Genes*, 1(2):227–243, 2010.

[104] Meredith L Carpenter, Jason D Buenrostro, Cristina Valdiosera, Hannes Schroeder, Morten E Allentoft, Martin Sikora, Morten Rasmussen, Simon Gravel, Sonia Guillén, Georgi Nekhrizov, et al. Pulling out the 1%: whole-genome capture for the targeted enrichment of ancient dna sequencing libraries. *The American Journal of Human Genetics*, 93(5):852–864, 2013.

[105] Peter B Damgaard, Ashot Margaryan, Hannes Schroeder, Ludovic Orlando, Eske Willerslev, and Morten E Allentoft. Improving access to endogenous dna in ancient bones and teeth. *Scientific Reports*, 5:11184, 2015.

[106] Michael Hofreiter, Johanna LA Paijmans, Helen Goodchild, Camilla F Speller, Axel Barlow, Gloria G Fortes, Jessica A Thomas, Arne Ludwig, and Matthew J Collins. The future of ancient dna: Technical advances and conceptual shifts. *BioEssays*, 37(3):284–293, 2015.

[107] Alon Keinan, James C Mullikin, Nick Patterson, and David Reich. Measurement of the human allele frequency spectrum demonstrates greater genetic drift in east asians than in europeans. *Nature Genetics*, 39(10):1251–1255, 2007.

[108] Michel Delseny, Bin Han, and Yue Ie Hsing. High throughput dna sequencing: the new sequencing revolution. *Plant Science*, 179(5):407–422, 2010.

[109] Iain Mathieson, Iosif Lazaridis, Nadin Rohland, Swapan Mallick, Nick Patterson, Songül Alpaslan Roodenberg, Eadaoin Harney, Kristin Stewardson, Daniel Fernandes, Mario Novak, et al. Genome-wide patterns of selection in 230 ancient eurasians. *Nature*, 528(7583):499–503, 2015.

[110] Sharon R Browning and Brian L Browning. Haplotype phasing: existing methods and new developments. *Nature Reviews Genetics*, 12(10):703–714, 2011.

[111] Jose Manuel Monroy Kuhn, Mattias Jakobsson, and Torsten Günther. Estimating genetic kin relationships in prehistoric populations. *PLoS One*, 13(4):e0195491, 2018.

[112] Jack N Fenner. Cross-cultural estimation of the human generation interval for use in genetics-based population divergence studies. *American Journal of Physical Anthropology: The Official Publication of the American Association of Physical Anthropologists*, 128(2):415–423, 2005.

[113] Monika Karmin, Lauri Saag, Mário Vicente, Melissa A Wilson Sayres, Mari Järve, Ulvi Gerst Talas, Siiri Rootsi, Anne-Mai Ilumäe, Reedik Mägi, Mario Mitt, et al. A recent bottleneck of y chromosome diversity coincides with a global change in culture. *Genome Research*, 25(4):459–466, 2015.

[114] Joseph Lachance and Sarah A Tishkoff. Snp ascertainment bias in population genetic analyses: why it is important, and how to correct it. *Bioessays*, 35(9): 780–786, 2013.

[115] Éadaoin Harney, Nick Patterson, David Reich, and John Wakeley. Assessing the performance of qpadm: A statistical tool for studying population admixture. *bioRxiv*, 2020.

[116] Stef Van Buuren. *Flexible imputation of missing data*. CRC press, 2018.

[117] Michael W Nachman. Variation in recombination rate across the genome: evidence and implications. *Current opinion in genetics & development*, 12(6):657–663, 2002.

[118] Anna Ferrer-Admetlla, Mason Liang, Thorfinn Korneliussen, and Rasmus Nielsen. On detecting incomplete soft or hard selective sweeps using haplotype structure. *Molecular biology and evolution*, 31(5):1275–1291, 2014.

[119] Joseph K Pickrell and David Reich. Toward a new history and geography of human genes informed by ancient dna. *Trends in Genetics*, 30(9):377–389, 2014.

[120] Adam H Freedman, Ilan Gronau, Rena M Schweizer, Diego Ortega-Del Vecchyo, Eunjung Han, Pedro M Silva, Marco Galaverni, Zhenxin Fan, Peter Marx, Belen Lorente-Galdos, et al. Genome sequencing highlights the dynamic early history of dogs. *PLoS Genetics*, 10(1):e1004016, 2014.

[121] Kevin Cheeseman, Etienne Rouleau, Anne Vannier, Aurélie Thomas, Adrien Briaux, Cedrick Lefol, Pierre Walrafen, Aaron Bensimon, Rosette Lidereau, Emmanuel Conseiller, et al. A diagnostic genetic test for the physical mapping of

germline rearrangements in the susceptibility breast cancer genes brca1 and brca2. *Human mutation*, 33(6):998–1009, 2012.

[122] J Víctor Moreno-Mayar, Thorfinn Sand Korneliussen, Jyoti Dalal, Gabriel Renaud, Anders Albrechtsen, Rasmus Nielsen, and Anna-Sapfo Malaspinas. A likelihood method for estimating present-day human contamination in ancient male samples using low-depth x-chromosome data. *Bioinformatics*, 36(3):828–841, 2020.

[123] Joost DJ Plate, Rutger R van de Leur, Luke PH Leenen, Falco Hietbrink, Linda M Peelen, and MJC Eijkemans. Incorporating repeated measurements into prediction models in the critical care setting: a framework, systematic review and meta-analysis. *BMC Medical Research Methodology*, 19(1):1–11, 2019.

[124] Philip W Hedrick and Glenys Thomson. Evidence for balancing selection at hla. *Genetics*, 104(3):449–456, 1983.

[125] Xiaoheng Cheng and Michael DeGiorgio. Flexible mixture model approaches that accommodate footprint size variability for robust detection of balancing selection. *Molecular Biology and Evolution*, 37(11):3267–3291, 2020.

[126] Sabri Boughorbel, Fethi Jarray, and Mohammed El-Anbari. Optimal classifier for imbalanced data using matthews correlation coefficient metric. *PLoS One*, 12(6): e0177678, 2017.