



**MONASH** University  
Engineering

# Coding and information theory problems in DNA storage with nanopore sequencing

ENG4702: Final Year Project - Final Report

Author: Preet Patel (31467229)

Supervisor: Professor Emanuele Viterbo

Date of Submission: 26 May 2024

Project type: Research

# 1 Executive Summary

---

Storing digital data in DNA strands is an up-and-coming storage technique, with nanopore sequencing being one of the promising ways to efficiently read this data. This project investigates encoding data with codons and determines factors which can reduce the base error rate incurred through the sequencing process. The results from simulation show that the error rate is only weakly related to the popularity of the codons used, suggesting that there is little benefit from using codons as opposed to longer codewords. The error rate is found to have a stronger correlation with GC-content and homopolymerity, a relationship which is used to form a codebook which incurs half as many errors compared to an average codebook. In conducting this analysis, an empirical insight into contemporary basecalling software is also provided, finding Oxford Nanopore Technologies' basecallers Scrappie and Dorado to perform in the expected order, as suggested by their authors. Research was also conducted on the classification of real reads, and strategies to reduce the error rate in this domain.

---

## 2 Acknowledgement of Country

---

In the spirit of reconciliation, the author acknowledges the people of the Kulin Nations, on whose land Monash University operates. More broadly, the author acknowledges the Traditional Custodians of country throughout Australia and their connections to land, sea, and community. The author pays respect to their Elders past and present and extends that respect to all Aboriginal and Torres Strait Islander people today.

# Contents

---

3	Introduction	7
4	Aims and Objectives	9
4.1	Hypothesis	9
4.1.1	Rationale	9
4.2	Aims	9
4.3	Objectives	9
5	Literature Review	10
5.1	Information Theory	10
5.2	Data storage in DNA	11
5.3	DNA Synthesis	12
5.4	DNA Computing	13
5.5	DNA Sequencing	14
5.6	Coding for the Nanopore Channel	15
6	Methodology and Methods	16
6.1	Methodology	16
6.2	Method – Experiment 1	17
6.2.1	Purpose	17
6.2.2	Scope	17
6.2.3	Codebook generation	17
6.2.4	Simulation pipeline	18
6.2.5	Tx message generator	18
6.2.6	Squiggle generator: Scrappie	19
6.2.7	Squiggle generator: $k$ -mer tables	20
6.2.8	Basecaller: Scrappie	21
6.2.9	Basecaller: Dorado	21
6.2.10	Read accuracy calculator	21
6.3	Method – Experiment 2	23
6.3.1	Purpose	23
6.3.2	Popularity metric	23
6.3.3	GC-content	23
6.3.4	Homopolymerity metric	23
6.3.5	Results generation process	24
6.4	Method – Experiment 3	25
6.4.1	Purpose	25
6.4.2	Information about the reads	25

6.4.3	Classification	25
7	Results and Discussion	27
7.1	Experiment 1: Preliminary investigation	27
7.1.1	Preamble on testing setup	27
7.1.2	Results: R9 Scrappie generation and Scrappie basecalling	28
7.1.3	Results: R9 Scrappie generation and Dorado 'fast' basecalling	29
7.1.4	Results: R9 Scrappie generation and Dorado 'hac' basecalling	30
7.1.5	Results: R9 Scrappie generation and Dorado 'sup' basecalling	31
7.1.6	Results: <b>k</b> -mer generation and Dorado 'sup' basecalling – variable padding length	32
7.1.7	Results: R9 <b>k</b> -mer generation and Dorado 'sup' basecalling	33
7.1.8	Results: R10 <b>k</b> -mer generation and Dorado 'sup' basecalling	34
7.1.9	Discussion of preliminary results	35
7.2	Experiment 2: Investigating codebook metrics	37
7.2.1	Results: Popularity metric	37
7.2.2	Results: GC-content	37
7.2.3	Results: Homopolymer metric	38
7.2.4	Discussion of codebook metrics	39
7.2.5	Limitations and future work	40
7.3	Experiment 3	41
7.3.1	Results and Discussion: Classification	41
7.3.2	Results and Discussion: Improved classification	41
8	Conclusion	43
9	Reflection on Project Management	44
9.1	Project Scope	44
9.2	Project Plan & Timeline	44
9.2.1	Original timeline	45
9.2.2	Final timeline	46
9.2.3	Comparison of timelines	46
9.3	Reflection on Project	47
10	References	48
11	Appendices	51
11.1	Appendix A: Project Risk Assessment	51
11.2	Appendix B: Risk Assessment Matrix	52
11.3	Appendix C: Risk Management Plan	53
11.3.1	Task abstraction	53
11.3.2	Risk analysis	53
11.4	Appendix D: Sustainability Plan	55
ENG4702 Final Report		5

11.4.1	Sustainable Development Goals	55
11.4.2	DNA Storage and Sustainability	56
11.5	Appendix E: Generative AI Statement	57
11.6	Appendix F: DNA File Formats	58
11.6.1	FASTA	58
11.6.2	FAST5	58
11.6.3	POD5	58
11.7	Appendix G: Guide to Scrappie	59
11.7.1	What is Scrappie?	59
11.7.2	Scrappie installation	59
11.7.3	Scrappie squiggle generation	59
11.7.4	Scrappie basecalling	60
11.8	Appendix H: Guide to Dorado	61
11.8.1	What is Dorado?	61
11.8.2	Dorado basecalling	61
11.8.3	Dorado alignment	61
11.9	Appendix I: Guide to F5C	63
11.9.1	What is F5C?	63

### 3 Introduction

The digital age has seen and continues to see exponential growth in the quantity of stored information, with research playing a crucial role in ensuring we can efficiently and sustainably meet the demand of a data-driven future. An emerging and promising approach to storing digital information is writing to the nucleotide sequence of DNA strands. Storing information at a molecular level in this fashion yields unparalleled levels of data density, with the stored data remaining stable for millennia with virtually no energy consumption [1]. These aspects make DNA well suited to the archival storage of information and could one day compete with the currently popular tape-based methods. Furthermore, being able to efficiently modify and operate on DNA strands could push the boundaries of DNA computing and one day help realise a DNA computer. Achieving these milestones will require a significant collaborative effort between the fields of engineering, mathematics, biochemistry, and computer science.

There are several limitations in DNA storage that must be alleviated before the technology can traverse the boundaries of research. The primary limitation is the significant cost and time required to both synthesise and sequence DNA. Cheaper and faster synthesis technologies are available, but these are more prone to errors, requiring a robust error-correcting code to be implemented. While sequencing has historically been done using Sequencing By Synthesis (SBS), a quicker and cheaper alternative using nanopore technology has emerged, which is the focus of this project. A nanopore is a nanoscale organic structure that contains a pore through which a single strand of DNA can be fed. This process is controlled by a motor protein, which shifts the DNA strand through the nanopore, one nucleotide at a time. Figure 1 shows a diagram of an R9 nanopore, which is a common nanopore sold by Oxford Nanopore Technologies (ONT).

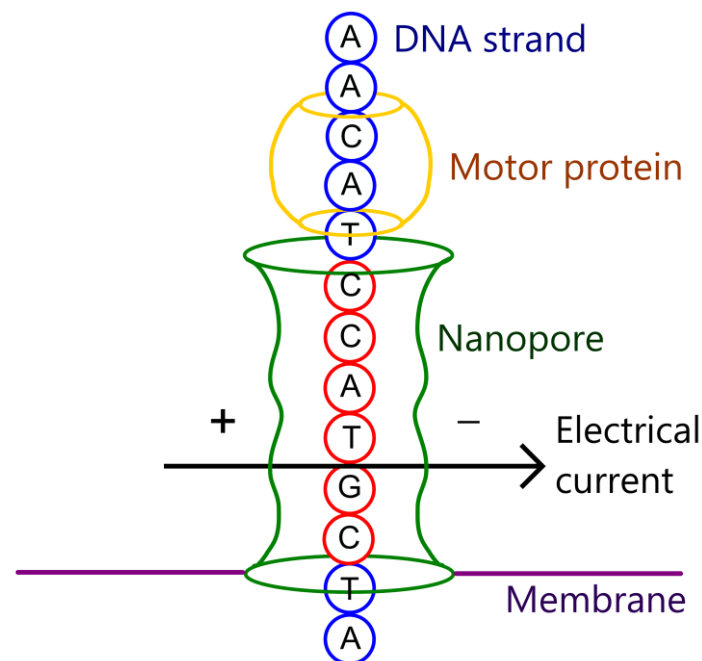


Figure 1: Diagram of an R9 nanopore by Oxford Nanopore Technologies. The nanopore sits on a membrane, and the motor protein shifts the single DNA strand through the nanopore, one nucleotide at a time. Adapted from [2].

As the bases pass through the nanopore, they alter the magnitude of an electrical current applied across the pore. By measuring this current and comparing it to levels when known nucleotide strands are passed through the pore, a neural network can determine the sequence of bases with an accuracy claimed to be over 99% [2].

As nanopore technology becomes more efficient and less erroneous, the feasibility of DNA storage grows. However, errors will be present in any DNA synthesis or sequencing process, and there are many errors which are unique to the nanopore channel. From an information theoretical perspective, one solution to ensuring the reliable recovery of data is to introduce logical redundancy prior to the synthesis process. Many studies have been done to produce an effective coding scheme for DNA storage with nanopore sequencing, however these often simplify the intricacies of a nanopore sequencer. The aim of this project will be to evaluate the feasibility of using a codebook of codons to encode digital data which is stored in DNA and sequenced using a nanopore. The codebooks will be tested against Scrappie and Dorado, which are two neural networks provided by ONT that perform basecalling, meaning that they take a raw current signal from a nanopore and output a sequence of bases.

The focus on codons in this project is for two main reasons. Firstly, the basecalling algorithms provided by ONT for their nanopore sequencers are neural networks which use training data based on DNA from humans, animals, plants, bacteria, and viruses [2]. These all contain genes, and therefore contain long sequences of codons. Neural networks tend to perform better when given data that is similar to their training data. We therefore hypothesise that neural networks will be more accurate when asked to sequence synthetic DNA formed from a codebook of frequent codons, compared to completely inorganic DNA. Next, a codon-based storage approach may aid DNA computing for a number of reasons. Enzymes that repair DNA in living organisms may one day be used to modify codon-based DNA strands, allowing data modification in DNA computers. A sequence of synthetically formed codons could also be used to synthesize a protein, which would allow an alternative way of representing and reading information that was originally stored in DNA.



## 4 Aims and Objectives

---

### 4.1 Hypothesis

It is hypothesised that Dorado will produce reads with a higher quality factor when basecalling nanopore signals of synthetic DNA built from a codebook of genetically frequent codons, compared to a codebook of infrequent codons, or a randomised codebook.

#### 4.1.1 Rationale

As Dorado is trained on DNA containing genetic information [2], it is expected that synthetic data which is built from frequent codons would more closely align with this training data, and therefore yield fewer errors when basecalled. Previous studies have focused on the performance of ONT's past basecallers on natural genomic DNA as opposed to synthetic DNA [3]. As Dorado is a relatively new basecaller, its capabilities are yet to be studied to the same level, and this research project aims to bridge this gap.

### 4.2 Aims

The principle aims of this research project are:

- [A.1] To evaluate the feasibility of using a codebook of popular codons to encode data which is stored in DNA and sequenced using a nanopore.
- [A.2] To investigate codebook metrics besides popularity and create an optimal codebook of codons.
- [A.3] To analyse and report on the performance of Dorado when basecalling synthetic DNA.
- [A.4] To document the techniques used to analyse the performance of codebooks and basecallers, to benefit other researchers in the field.

### 4.3 Objectives

In pursuit of these aims, the objectives of this project are:

- [O.1] To understand the steps in the storage of data in DNA, and the benefits and drawbacks of this approach when compared to conventional data storage systems.
- [O.2] To document the distinctive features of the nanopore sequencing process, including understanding existing models and existing research on channel capacity.
- [O.3] To understand the role of information theory in the nanopore channel.
- [O.4] To develop capability in using software such as Scrappie and Dorado to simulate squiggle generation and basecalling.
- [O.5] To test and analyse the capabilities of Dorado and to compare the performance of the various models provided by Dorado.
- [O.6] To provide documentation and explanation of the software provided by ONT, including guides and examples of usage.
- [O.7] To conduct preliminary analysis comparing the performance of a codebook of popular codons with a codebook of unpopular codons.
- [O.8] To publish a report on the preliminary findings.
- [O.9] To define and investigate the impact of codebook metrics besides popularity.
- [O.10] To create codebooks which have optimal values of these metrics, and hence find an optimal codebook.
- [O.11] To publish a report and presentation on the findings of the project.

## 5 Literature Review

### 5.1 Information Theory

A formal mathematical theory of communication was developed by Claude Shannon in 1948 [4]. In his groundbreaking paper, Shannon discussed a general communication system, as depicted in Figure 2, which can be used to model the nanopore channel.

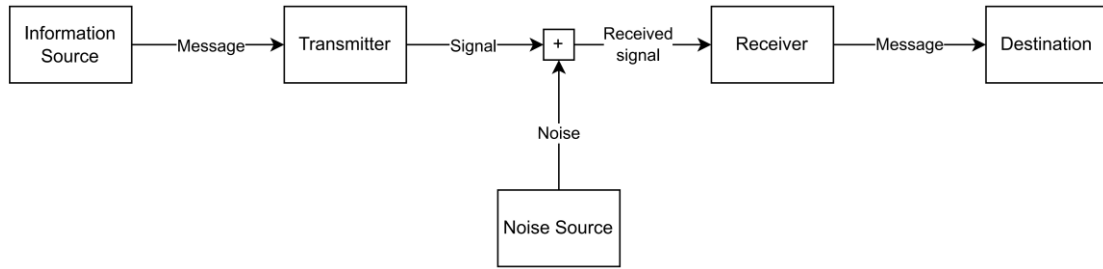


Figure 2: Block diagram showing components of a general communication channel. Adapted from [4].

The information source produces a sequence of symbols from a pre-defined alphabet  $\{S_1, S_2, \dots, S_n\}$ , which can be transmitted across the communication channel. In traditional computer storage, the alphabet is  $\{0, 1\}$ , while in paper-based storage, this is the alphabet, punctuation set and numerous other symbols. For DNA storage, the alphabet is the set of four nucleotides: adenine (A), cytosine (C), guanine (G) and thymine (T). Figure 3 shows an example of possible messages in each of these channels.



Figure 3: A comparison of messages found in three communication and data storage channels.

Assuming the information source is independent and identically distributed (IID), the quantity of information gained when the symbol  $S_i$  is output from the source is

$$I(S_i) = -\log_2 p_i,$$

where  $p_i$  is the probability that the output is  $S_i$ . The entropy of an information source is the average information gained from a single output of the information source. This is the weighted mean

$$H = -\sum_i p_i \log_2 p_i.$$

The entropy is maximised when  $p_i = \frac{1}{n}$ , and the maximal entropy value is  $H = \log_2(n)$ . This is a fundamental capacity on the information that can be conveyed through a single symbol. The unit is bits, so that when the alphabet is  $\{0,1\}$ , each symbol carries up to 1 bit of information.

The transmitter takes a symbol from the information source and creates a signal, which is a disturbance or imprint on some medium that is carried to the receiver to be recorded. While being carried, various sources

of noise may disturb the signal; these are collectively labelled the noise source, as depicted in Figure 2. The receiver obtains the signal, often at a specified sampling rate, and attempts to recover the original message. When there is significant noise in the channel, the receiver may incorrectly identify symbols in the received sequence. A coding scheme's objective is to carefully introduce redundancy in the channel so that errors are corrected, and no information is lost.

## 5.2 Data storage in DNA

The modern world is driven by data, with many organisations collecting and processing large swathes of data to gain a competitive advantage and improve customer experience. The amount of stored data has been growing and continues to grow exponentially as seen in Figure 4, with an estimated 175 ZB of data storage required by 2025 [5].

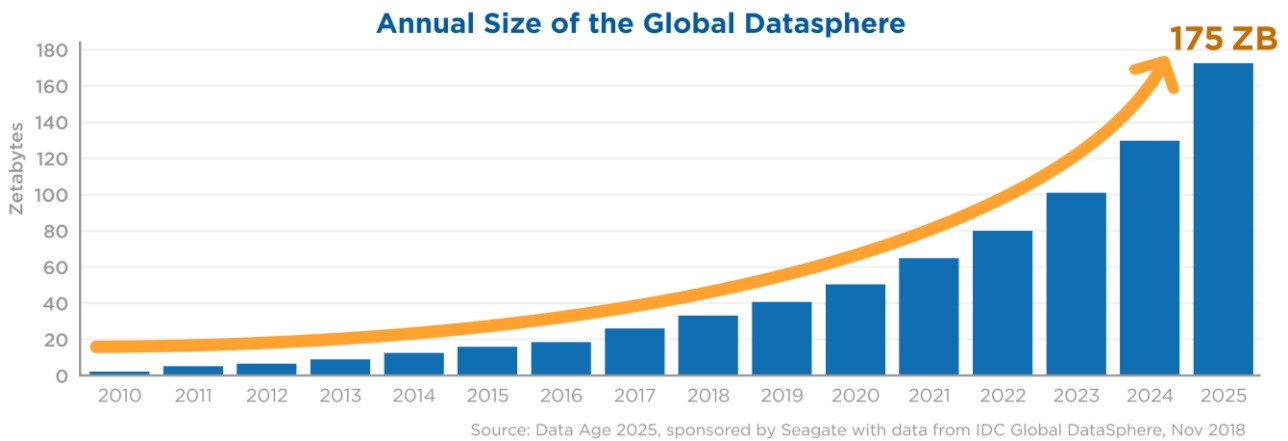


Figure 4: Expected growth in quantity of stored data until 2025. One zetabyte is  $10^{21}$  bytes. Source: IDC [5].

There are many conventional ways to store data, including in hard disk drives (HDD), solid-state drives (SSD), tape storage and paper-based methods. This research project focuses on a new approach, which is to store data in DNA. As there are four different nucleotides, a noiseless channel would have an information rate of 2 bit/nt. In practice, achievable information rates are much lower, due to a variety of reasons discussed in Section 5.6. A comparison of DNA storage with the aforementioned approaches is shown Table 1.

Table 1: Comparison of DNA-based storage with existing storage methods. Order of magnitude are approximate, and calculated based on market prices, given in USD. [1], [6], [7].

	DNA	Tape	HDD	SSD	Paper
<b>Cost to produce</b>	$10^8$ \$/TB	$10^1$ \$/TB	$10^1$ \$/TB	$10^2$ \$/TB	$10^4$ \$/TB
<b>Cost to store</b>	Low	Medium	High	High	Low
<b>Random access</b>	Difficult	Difficult	Easy	Easy	Difficult
<b>Longevity</b>	$10^5$ years	$10^1$ years	$10^1$ years	$10^1$ years	$10^2$ years
<b>Density</b>	Very high	High	Medium	Medium	Low
<b>Cost of duplication</b>	Low	Medium	Medium	Medium	High

DNA storage has many benefits over existing storage methods. At the forefront is the high information density it promises. Over  $10^8$  GB can be stored in a single gram of DNA [6], which is orders of magnitude higher than other technologies. In 2017, Erlich and Zielinski stored 215 MB of data in  $10^{-11}$ g of DNA, at an information rate of 1.58 bits/nt [6]. At this density, all the information in the world could be stored in a mass less than that of an average car. Another benefit of DNA storage is its longevity and low storage cost. In 2013, a team successfully sequenced 300,000 year old DNA from the bone of a cave bear [1]. In suitable conditions, DNA can last millennia with little degradation and low power consumption. To contrast, current data centres account for over 1% of global energy consumption [8], and this is expected to grow further. A third benefit is

that DNA is a fundamental component of life and will never grow obsolete. Research into DNA synthesis and sequencing techniques for biological purposes will continue, thereby improving the efficiency of DNA storage methods.

While DNA storage has many exciting benefits over existing technology, the current drawbacks mean it is not yet competitive for widespread usage. The largest of these drawbacks is the cost and time associated with the synthesis and sequencing of DNA. An estimated \$800 million USD would be needed to store 1TB of data, compared to \$15 USD for tape storage [1]. Furthermore, both synthesis and sequencing are complex processes requiring expensive infrastructure and trained personnel, making them almost inaccessible to a retail consumer. While several cheaper technologies exist, these introduce numerous errors in both the writing and reading stages, maintaining the drawbacks of the technology.

### 5.3 DNA Synthesis

DNA synthesis is usually performed in a four-step reaction using phosphoramidite chemistry. In this process, a large number of solid supports have a nucleotide chain growing on them in parallel. This is a very slow process which creates toxic waste and limits DNA strand lengths to 60-200nt [1]. Enzyme-based methods offer an alternative but are much less used and explored. This approach is expected to become cheaper and faster with ongoing research [1].

The costs of synthesis are often difficult to determine. As mentioned earlier, some estimates suggest that 1TB of data would cost \$800M USD to synthesise, and the time-cost is several seconds per nucleotide [1]. DNA storage has therefore been suggested as a form of archival storage, allowing data to be written once and read at a distant time in the future, without any modifications in the meantime.

While other cost-effective alternatives exist, these tend to introduce a large number of errors in the synthesised DNA strand. Creating efficient coding schemes that can correct these errors is a useful approach to improve the viability of these methods.

## 5.4 DNA Computing

One interesting application is the use of DNA as a platform for building computer systems. There are several benefits to a DNA-based approach over traditional semiconductor-based computers. The storage density of DNA would allow for hundreds of TB to be stored in a single gram, removing the need for large hard drives in computers [9]. The low energy use of DNA storage would give lower power consumption and longer sleep times. DNA strands could perform a large number of computations in parallel at great speeds, meaning it may one day rival the fastest supercomputers at a fraction of the energy cost.

Although we are far from realising DNA computers, important progress has been made in the field. A proof of concept was formulated by Adleman in 1994, when he used DNA strands of length 20 bases to encode a directed graph on 7 vertices. Ligations (joining of DNA strands) corresponded to paths and gel electrophoresis determined if a Hamiltonian path existed [10]. Entire operating systems and movies have successfully been stored in DNA strands [6], but without any infrastructure to run. DNA-based Boolean logic has previously been implemented [9], but the problem of scaling this to match semiconductor circuitry is still a major one.

One of the central aspects of computing is to perform operations on some stored data, and the viability of DNA as a computing platform is dependent on its efficacy for these operations. To begin, data in the form of DNA needs to be easily synthesised, a process that is currently expensive and slow. Due to the fragmented form of DNA, there needs to be a means of concatenating these fragments, which the process of ligation allows. The copying and duplication of data is made possible by techniques such as PCR. Another important operation is reading the data, which can be done in several ways. Information may be stored in the length of the DNA strands, which could be read using gel electrophoresis [11]. More relevant to this research project is the method of storing information in the nucleotide sequence itself. Nanopore sequencing would be naturally suited to this end, as its nanoscale size would allow many reads to be conducted in parallel in a DNA computer. This further prompts the research of efficient codes for DNA storage with nanopore sequencing.

The operations listed so far describe a very weak computer, which is capable of writing slowly, storing, duplicating and reading. Such a computer would be well-suited to the long-term storage of data. A much more practical computer could be formed if the data can be modified while in storage, for example by replacing individual bases without producing entirely new DNA strands. Enzymes that recognise and repair DNA exist in nature [12], and these could perhaps be used in a DNA computer to modify memory directly. Combined with enzymes that cleave and ligate DNA [12], a wider array of operations could be performed.

As such enzymes originate from life and interact with organic DNA, it is worthwhile formulating DNA coding schemes that adhere to the way DNA appears in life. Codons are sequences of three consecutive bases in genes which comprise 1-2% of the 3 billion base pairs of DNA [13]. Ribosome enzymes read one codon at a time and attach the corresponding amino acid to form the desired protein. This correspondence is depicted in Figure 5.

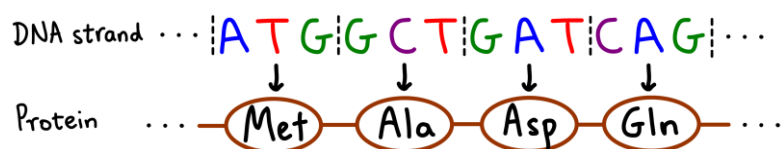


Figure 5: Diagram showing the codons in a DNA strand and the amino acids they code for.

The purpose of this project is to determine the efficiency and reliability of encoding data into DNA using a codebook of codons. If such a codebook is effective when sequenced with nanopores, then further research could be conducted to determine the ease by which codons can be modified in a DNA computer. Such a computer would also have the facility to represent codon-based DNA data as a protein by assembling the required amino acids. This would give a second way of representing the information in DNA and open new doors in the field of DNA computing.

## 5.5 DNA Sequencing

The conventional approach to DNA sequencing is *Sequencing By Synthesis* (SBS), a method developed in the 1990s and commercialised by Illumina in 2007 [14]. In this approach, a new strand of DNA is synthesised to match an existing strand. Fluorescent dyes are added to the process, so that each time a nucleotide is added to the new oligonucleotide chain, a flash of light is released. A large number of identical reactions occur in parallel, leading to a very low error rate in this approach [1].

A more recent approach developed by Oxford Nanopore Technologies is the use of a nanopore to read a single DNA strand. The nanopores currently used are biological in nature, created by pore-forming proteins collecting on a membrane [2]. A motor protein sits above the nanopore and controls the rate at which the DNA strand passes through the pore, as shown in Figure 1. In particular, the motor protein shifts the DNA strand one nucleotide at a time and pauses momentarily between each shift. During this process, an electrical current is passing through the nanopore, and the magnitude of this is sampled at a frequency of 4-5kHz. Small fluctuations in the current occur as the DNA passes through the nanopore, depending on which bases occupy the pore. Examining the current measurements allows a prediction of the original sequence of bases via neural networks.

The error rate is an important aspect of comparison between the two technologies. This is often represented by a quality score,  $Q$ , which is calculated as

$$Q = -10 \log_{10} e, \quad (1)$$

where  $e$  is the error rate. For example, Q20 corresponds to a 1% error rate. At the individual nucleotide level, there are three types of errors that may occur when a strand is sequenced, known as substitution, insertion and deletion errors. Figure 6 shows an example of each type of error.

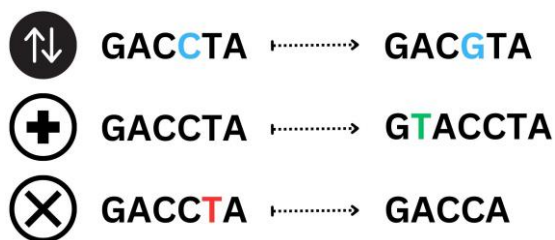


Figure 6: Illustration of substitution, insertion and deletion errors which may arise during any DNA sequencing process.

The latest single read accuracy advertised by Oxford Nanopore Technologies has a quality score of Q23 [2]. The company has also developed a new method called Duplex basecalling which achieves Q30. In this process, the DNA double strand is split by the motor protein and both strands are sent through the nanopore, one after the other, with the second strand being sent in reverse. In comparison, Illumina advertises that they achieve Q30 across their range of products.

Whilst sacrificing some accuracy, the nanopore approach has several benefits over SBS. Firstly, the process is significantly cheaper and easier to implement. A handheld MinION nanopore sequencer by ONT costs \$1000 USD [2], while a low-end Illumina sequencing system costs \$20000 USD [15]. While trained professionals are required to operate an Illumina system, a MinION can be used by anyone with a basic laboratory understanding. The MinION's portability is another benefit it offers, letting it be taken directly to the field if needed.

The SBS method is excellent at accurately detecting A, C, G and T bases, but other modified bases can also appear in DNA and RNA strands. Such modifications occur naturally in organisms, and nanopore sequencers are able to detect these whereas the SBS method cannot [16]. This is because the synthesis step in SBS adds one of the four bases to the existing oligonucleotide strand but does not have a mechanism to easily add modified bases in this process. To contrast, when a modified base passes through a nanopore, it alters the

current levels in a slightly different fashion to a regular base, and this discrepancy can be measured and understood.

Nanopore technology has made leaps and bounds in the last decade and holds a competitive presence in the world of DNA sequencing. It is important to note that there are many sources of errors in the current technology, some of which are unique to the nanopore sequencing process.

Firstly, the motor protein which pauses between each shift does not pause for a fixed amount of time. A single position of the DNA in the nanopore may last for many samples, a single sample, and sometimes not at all. The number of samples corresponding to a single position of the DNA strand is known as the dwell time. ONT's standard speed is 400 bases per second, at a sampling frequency of 4kHz, corresponding to a mean dwell time of 10 samples per nucleotide. The recent increase in the sampling rate from 4kHz to 5kHz means fewer nucleotides are expected to be completely missed.

A second cause of error is the noise in the current level measurements. When the noise is large enough, decoding the current pattern received becomes difficult, and the wrong pattern may be chosen, leading to an error. To work around these errors, an appropriate coding scheme should be used.

## 5.6 Coding for the Nanopore Channel

The four available standard bases give a theoretical maximum information rate of 2 bits/nt in a DNA storage channel. Some sequences of bases, such as long homopolymers, cannot be synthesised due to biochemical constraints. After this is considered, the maximum information rate reduces to 1.98 bits [6]. Furthermore, decay during storage and the requirement for some nucleotides to be used to index the strand reduce the maximum information rate further to approximately 1.83 bits/nt [6]. Rates close to this can be achieved when accurate synthesis and sequencing techniques are used, such as SBS. Due to the previously mentioned sources of noise in a nanopore sequencer, the channel capacity of a nanopore channel is significantly lower, and not well understood. Error-free recovery of information is important for the archival storage of data, and adding redundancy is a means to achieve this.

Many researchers have employed or considered error-correcting codes for DNA data storage. In 2014, Grass et al were able to recover 83kB of data error-free at an information rate of 1.14 bits/nt. They achieved error-correction by using a concatenated code, where both the inner and outer codes were independent Reed-Solomon codes [17]. This team used an Illumina sequencer, which on average had less than 1 error per 158nt length sequence. The results cannot be directly applied to this project, as nanopore sequencing would have a much higher error rate.

Organick et al made a significant contribution to DNA storage with nanopore sequencing, with the team using this method to store and recover over 200MB of data [18]. They also employed a Reed-Solomon outer code and XORed the message with a random sequence of bases as an inner code, and were able to achieve an information rate of 1.10 bits/nt. Two shortcomings of this experiment were that the sequences were read 36-80 times. More readings naturally increase the reliability of the information being transferred, and it is more fundamental to determine the information rate of a single read. Secondly, the code used didn't consider the specific sources of error or intricacies of a nanopore channel. In particular, the channel has memory and each position of the DNA is measured for a variable amount of time, meaning some sets of  $k$ -mers ( $k$  consecutive nucleotides) are more likely to be mistaken for one another than other sets. This is accounted for in Vidal, Wijekoon and Viterbo's paper, who propose a codebook with 16 codewords of length 4, giving an information rate of 1 bit/nt [19].

## 6 Methodology and Methods

### 6.1 Methodology

The research methodology chosen for this project is quantitative research, which is the natural choice for a project concerned with testing the efficiency and effectiveness of a codebook for a communication channel. Through quantitative analysis, the student will calculate information rates and error rates, which will help evaluate the effectiveness of the codebook.

An explicit example of quantitative methodology being used in a related research project is found in [19]. In this study by Vidal, Wijekoon and Viterbo, the codebook is evaluated by computing the pairwise error probabilities between codewords. This computation relies on a significant amount of theory developed earlier in the paper, and drawn from the wider literature. Once a codebook is chosen, numerical simulations on Scroppie are used to evaluate error performance. These computations and simulations show the importance of numerical analysis and quantitative research methodology when evaluating the feasibility of a codebook.

Another example is the work of Grass et al. [17], who evaluated their code by calculating how many errors the channel introduced, and what percentage of these errors were caught by each of the inner and outer code. A numerical summary of the work they performed included the amount of synthesised data, information density, number of oligonucleotides and physical density. Providing these statistics allows for easy comparison by other researchers in the field who are researching the same or similar problems. For example, Erlich et al. [6] were subsequently able to compare their code with many researchers including Grass et al., as shown in Table 2.

Table 2: Comparison of DNA storage coding schemes from 2012 to 2017 which used Sequencing By Synthesis. Source: Erlich et al. [6]

Parameter	Church et al. (3)	Goldman et al. (4)	Grass et al. (5)	Bornholt et al. (6)	Blawat et al. (7)	This work
Input data (Mbytes)	0.65	0.75	0.08	0.15	22	2.15
Coding potential (bits/nt)	1	1.58	1.78	1.58	1.6	1.98
Redundancy	1	4	1	15	113	107
Robustness to dropouts	No	Repetition	RS	Repetition	RS	Fountain
Error correction/detection	No	Yes	Yes	No	Yes	Yes
Full recovery	No	No	Yes	No	Yes	Yes
Net information density (bits/nt)	0.83	0.33	1.14	0.88	0.92	1.57
Realized capacity	45%	18%	62%	48%	50%	86%
Number of oligos	54,898	153,335	4,991	151,000	1,000,000	72,000
Physical density (Pbytes/g)	1.28	2.25	25	–	–	214

The student will therefore follow a quantitative methodology and draw upon published papers in the field for inspiration on which numerical results and methods to utilise. Once these results are calculated, they will be compared against published papers or benchmarks provided by ONT.



## 6.2 Method – Experiment 1

The code for all experiments can be found on GitHub upon request.

### 6.2.1 Purpose

Experiment 1 is a preliminary investigation, whose purpose is to both benchmark the accuracy of the software and models used, and to obtain an initial understanding of whether using a codebook of popular codons improves the error rate. This will contribute to aims [A.1] and [A.3], as listed in Section 4.2.

### 6.2.2 Scope

Two methods of squiggle generation will be studied – through Scrappie and through  $k$ -mer tables. Two basecallers called Scrappie and Dorado will be studied, however the latter has multiple models which will be studied separately.

We will investigate R9.4 and R10.4 nanopores, which are two commonly used nanopores from ONT. For simplicity, we will refer to these as R9 and R10. There is a slight technical point to be made here – R10 is actually the name of the original version of the nanopore released in 2019, while R10.4 is the latest revision released in 2022. The original R10 nanopore is scarcely used, with Dorado only offering support for R9.4 and R10.4 nanopores. Scrappie supports R9.4 and R10 nanopores, but we ignore its functionality of the latter given the minimal usage.

### 6.2.3 Codebook generation

ONT's basecallers have been trained using a variety of plant, animal, bacterial and viral DNA [2]. The codons in genes appear in varying frequencies, which were found on a peer-reviewed codon statistics database in TSV format [20]. In particular, the human genome was chosen as it was expected to best match the training data. The codebook lengths were fixed at 16, allowing 4 bits to be stored per 3 bases. Two codebooks were constructed from this frequency data. In the first codebook, the 16 most popular codons were chosen, while in the second codebook, the 16 least popular codons were taken.

Two other codebooks that did not rely on codon frequency statistics were also created in this step. The third codebook was a randomised control codebook. The fourth codebook was an attempt at a bad codebook, containing many repeated bases, and low GC-content. These codebooks are listed below. The software written also supports external codebooks with arbitrary length and structure to be used, potentially aiding other researchers in the field.

Codebook 1: AAA AAG AGC ATC ATG CAG CCC CTG GAA GAC GAG GAT GCC GGC GTG TTC,

Codebook 2: ACG ATA CCG CGA CGC CGT CTA GCG GGT GTA TAA TAG TCG TGA TGT TTA,

Codebook 3: AAT AGT ATA ATT CAA CCC CGG CTC CTT GAG GCC GGA GTG TCG TCT TGT,

Codebook 4: AAA AAC AAG AAT ACA AGA ATA CAA GAA TAA TTT TTA TTC TTG TAT TCT.

The performance of each codebook depended on the magnitude of global noise,  $\sigma$ , and the mean dwell time,  $K$  chosen at the signal generation step in the pipeline. For each codebook and each choice of  $\sigma$  and  $K$ , between 200 and 1000 trials were conducted using messages of length  $m = 200$  codons. The number of trials was chosen to be high enough so that the 95% confidence intervals in the quality score did not show significant overlap between the codebooks.

#### 6.2.4 Simulation pipeline

As depicted in Figure 7, there are three main stages in DNA storage with nanopore sequencing. In the first stage, a DNA strand containing information is synthesised and stored for a desired amount of time. In the next stage, the strand is split, and a single strand is fed through a nanopore. The current levels (in pA) are measured, and this signal is known as a *squiggle*. In the final stage, a basecaller passes the squiggle through a neural network which looks at the current fluctuations and outputs a sequence of bases.

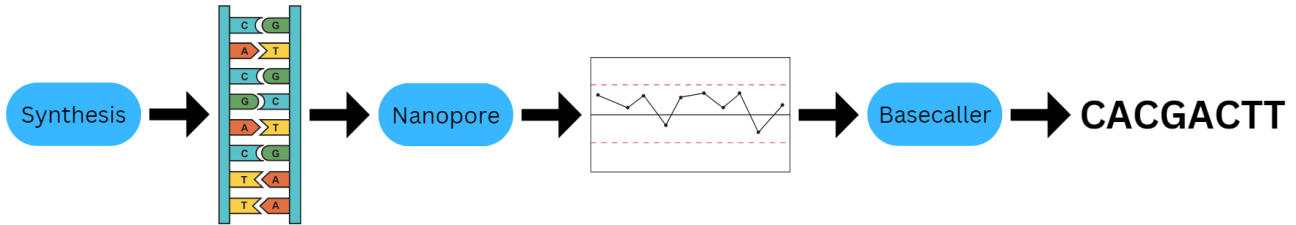


Figure 7: The three main stages of synthetic DNA storage with nanopore sequencing.

As mentioned previously, synthesis and sequencing are expensive operations, both in cost and time. This process will need to be run many times in this project, and therefore it is unfeasible to produce and use real DNA. Instead, various computer packages will be used to simulate the process, with the intended pipeline shown in Figure 8.

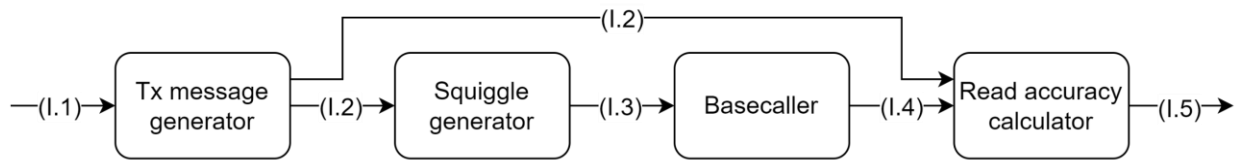


Figure 8: Block diagram of simulated nanopore sequencing process. Each block represents a major step in the process, and each arrow represents a transfer of data. Further information on the data corresponding to each label can be found in the succeeding sections.

#### 6.2.5 Tx message generator

The transmission message generation step mimics the synthesis step when working with synthetic DNA. It takes a codebook and randomly samples and concatenates codewords to form a transmission message. For example, for the dummy codebook {AAA, CCC, GGG, TTT}, a possible output would be AAATTTAAACCCCC.

The inputs (I.1) of this block are shown in Table 3.

Table 3: List of inputs and their formats, corresponding to (I.1) in Figure 8.

Input	Format
codebooks	Text files containing codewords in a single line, separated by a space.
padding	String of bases.
message length	Integer.

The *message length*,  $m$ , is the number of codewords which should be randomly concatenated to form the transmission message. Preliminary investigations found that squiggle-generation and basecalling is less reliable at the start and end of strands. To account for this, copies of the *padding* input are added before and after the transmission message. With a padding of length  $k$ , the padded transmission message will have a length of  $L = 3m + 2k$ . The final transmitted message is stored in a FASTA file for later analysis.

Table 4: List of outputs and their formats, corresponding to (I.2) in Figure 8.

Output	Format
Transmitted message	FASTA format, as described in Appendix F.

### 6.2.6 Squiggle generator: Scrappie

The squiggle generation step mimics the nanopore signal generation step when working with synthetic DNA. This project investigates two methods for squiggle generation – through Scrappie and through  $k$ -mer tables.

Scrappie is a technology demonstrator which was first released in 2017 and has been archived since 2022. It simulates both the nanopore squiggle generation process as well as the basecalling process through the commands *squiggle* and *raw* respectively.

The FASTA file from the transmission message generation step is passed into Scrappie. Instead of directly providing a squiggle, several statistics are outputted from which a squiggle can be made. These come in five columns in the format shown in Figure 9, which corresponds to a transmission message beginning with TGTTCAT.

#	Sequence	1			
pos	base	current	sd	dwel1	
0	T	0.419587	0.345852	0.713681	
1	G	1.023567	0.318915	0.794324	
2	T	0.079655	0.306535	1.033633	
3	T	-0.728703	0.242368	2.603098	
4	C	1.033638	0.181615	4.248816	
5	T	0.463943	0.149272	4.893713	
6	C	1.642555	0.178201	6.842926	
7	A	0.567149	0.197912	7.023503	
8	T	1.148610	0.125626	7.985809	

Figure 9: Scrappie output. Source: Scrappie [21].

The *current* column shows the expected normalised current level  $\mu_i$  as the  $i^{th}$  base passes through the nanopore. In practice, the current fluctuates with a mean of around 70pA and a standard deviation of around 12pA, but Scrappie and other basecallers use normalised data to rule out the significant variability between nanopores. One would expect that for a large number of bases, the mean of the *current* column would be 0, while the standard deviation would be 1. Upon measuring, the mean *current* was actually found to be -0.03 and the standard deviation was 0.8. One explanation for the discrepancy is that normalisation may have been conducted on the natural DNA on which Scrappie was trained, and this may give greater variability in current levels when passed through a nanopore. The *sd* column indicates the normalised noise intensity  $\sigma_i$ , which is the magnitude of Gaussian noise which should be added to  $\mu_i$  when generating the squiggle.

The bases pass through the nanopore in a ratchet-like fashion, as the motor protein holds the DNA strand for a variable amount of time. The number of samples each base is held for is estimated by Scrappie and displayed in the final ‘dwell’ column. For the bulk of the sequence, the dwell time is 6-9 samples, however as seen in Figure 9, the dwell time is significantly smaller at the start and end of sequences. The dwell time provided is simply the expected dwell time. The producers of the software recommend a geometric distribution to generate dwell times at the signal generation step [21].

We will now describe mathematically how a squiggle is generated from these statistics. For the  $i^{th}$  base, let  $\mu_i$ ,  $\sigma_i$  and  $K_i$  be the *current*, *sd* and *dwell* as found in the  $i^{th}$  row of the table. Firstly, we define  $T_i$  to be a geometric random variable (starting at 1) with expectation

$$\mathbb{E}[T_i] = K_i.$$

The squiggle will contain  $T_i$  samples corresponding to the  $i^{th}$  base. Let these be  $(Z_1^i, Z_2^i, Z_3^i, \dots, Z_{T_i}^i)$ . Each of these samples are drawn from a normal distribution as

$$Z_j^i \sim \mathcal{N}(\mu_i, \sigma_i^2).$$

The squiggle is then the concatenation of these samples for each base, written as

$$(Z_1^0, \dots, Z_{T_0}^0, Z_1^1, \dots, Z_{T_1}^1, \dots, Z_1^{L-1}, \dots, Z_{T_{L-1}}^{L-1})$$

The nature of this distribution means many bases will only persist for 1-2 samples and may therefore be missed during the basecalling stage of the pipeline. The significant inter-symbol interference would counteract this to some degree and allow such bases to be recovered.

Figure 10 shows an example of the first 8 bases of a generated squiggle.

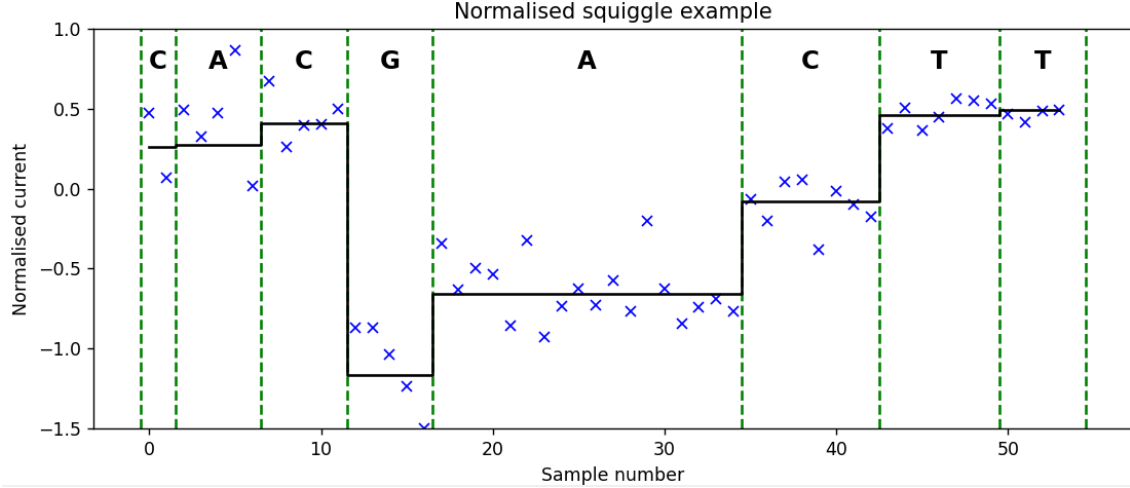


Figure 10: Plot of normalised current deviations for an R9 nanopore simulated by Scrappie. The black line is the mean current level, while the blue data is the generated squiggle, which hovers around this mean.

The green dashed lines separate the bases – in this realisation we have  $T_0 = 2$  samples, showcasing the low reliability of Scrappie at the start of a sequence. Thereafter, the current level fluctuates around a mean value for each base.

Although Scrappie provides a distinct *sd* and *dwell* for each base, it is more useful to override these with a global noise standard deviation,  $\sigma$ , and mean dwell time,  $K$ . These can be altered to understand the effect the parameters have on the quality score of the basecall.

Before storing, the squiggle is digitised by scaling and converting each sample to an integer between 0 and 8191. This is stored in a FAST5 or POD5 file, along with a swathe of metadata about the read. The output of the squiggle generation step is the following.

Table 5: List of outputs and their formats, corresponding to (1.3) in Figure 8.

Output	Format
Squiggle	FAST5 or POD5 format, as described in Appendix F.

### 6.2.7 Squiggle generator: $k$ -mer tables

A second method to generate squiggles is with  $k$ -mer tables. A  $k$ -mer is a sequence of  $k$  consecutive bases. As a DNA strand passes through a nanopore, several ( $k$ ) bases are contained inside the nanopore at any particular time and these all have some impact on the measured current level. With a table mapping  $k$ -mers to expected current levels, we can predict the current in the nanopore without needing Scrappie.

Such tables are provided by ONT for public use [22]. For R9 pores, 6-mer tables are provided as 6 bases explains most of the variation in current. For the larger R10 pores, 9-mer tables are provided, which still seem to be insufficient. There is some downside in using  $k$ -mer tables instead of Scrappie since the latter employs a neural network which more effectively models the nuances of a nanopore and can take into account more than 6 bases at a time. However, Scrappie only generates R9 squiggles, and  $k$ -mer tables must be used for R10.

### 6.2.8 Basecaller: Scrappie

The first basecaller we investigate is Scrappie. We use the *raw* function, which takes a squiggle in FAST5 format and outputs a sequence of bases, which are predicted through a neural network.

Table 6: List of outputs and their formats, corresponding to (I.4) in Figure 8.

Output	Format
Received message	FASTA format, as described in Appendix F.

### 6.2.9 Basecaller: Dorado

Dorado is a more powerful basecaller which replaces many other basecallers created by ONT. The *dorado basecaller* command is analogous to *scrappie raw*. The authors of Dorado recommend squiggles in POD5 format, and the steps to create such a file are included in Appendix F. An example of a Dorado call is

```
dorado basecaller --emit-fa dna_r9.4.1_e8_sup@v3.6 squiggle.pod5 > calls.fa
```

In this command, the output format chosen is FASTA, which is simpler than the default binary option BAM and its readable counterpart SAM. A range of basecalling models besides *dna\_r9.4.1\_e8\_sup@v3.6* can be chosen, with a full list found on the Dorado git page [23]. The tag *sup* is short for ‘super accuracy’, which is the most accurate model but also the most computationally demanding [2]. *hac* and *fast* are also offered, which stand for ‘high accuracy’ and ‘fast’ respectively. In previous research with another basecaller Guppy, *hac* was found to reduce error rates by about 2% compared to *fast* [24]. We will investigate how these models compare in Dorado.

### 6.2.10 Read accuracy calculator

Previous researchers who have used or proposed a codebook have listed error rates or described algorithms to determine bounds on error rates [6] [17] [19]. One of the evaluation criteria for the proposed codebooks in this project will be the read accuracy they achieve.

Traditionally, the Hamming distance is used to calculate the error rate in a communication medium. For example, if

```
tx_msg = 'AAAACCCCGGGGTTTT'
```

and

```
rx_msg = 'AAATCCCGGGGATTT'
```

then the Hamming distance between the sequences is 2, due to the errors in positions 4 and 13. This metric is useful if the only type of error is substitution error. However, in the nanopore channel, bases are often erroneously inserted or deleted when basecalling. For example, we may have

```
tx_msg = 'ACACACACACACACA'
```

```
rx_msg = 'GACACACACACACA'
```

The Hamming distance between these is 16, which is the entire length. The sequences are nonetheless similar, related by an insertion error at the beginning of the sequence and a deletion error at the end.

A more useful metric is the Levenshtein distance (also called edit distance), which counts the number of *edits* that need to be made to one string to get to another [25]. An edit refers to a substitution, insertion or deletion, as described in Figure 6. For example, the Levenshtein distance between *tx\_msg* and *rx\_msg* above would be 2. Initially, we implemented the Levenshtein algorithm in python, but found this to be slow, having a time complexity of  $O(n^2)$ . A faster implementation was found and subsequently used from the python package *editdistance* [26].

From the edit distance  $ED$ , a quality score for the trial can be calculated using the formula

$$Q = -10 \cdot \log_{10} \frac{ED}{3m},$$

where  $3m$  is the length of the transmitted message without padding. Finally, a summary of the entire run is stored in a text file. When  $N$  repeated trials are conducted, the program also outputs the sample mean quality score,  $\bar{Q}$ , and the uncertainty in this number.

*Table 7: List of outputs and their formats, corresponding to (1.5) in Figure 8.*

<b>Output</b>	<b>Format</b>
Run information	Text file, containing codebook, $m$ , $N$ , $\sigma$ , $K$ , $\bar{Q}$ , $sd(\bar{Q})$ .

## 6.3 Method – Experiment 2

### 6.3.1 Purpose

Experiment 1 will provide important information about the simulation and basecalling software used and give preliminary information about the performance of the various codebooks. The purpose of experiment 2 is to investigate some metrics and find an improved codebook. We will use a single set of operating conditions for all codebooks tested, summarised in the following parameter block.

Table 8: Parameter block, showing all control variables for experiment 2.

Squiggle generation	Basecaller	Model	BPS	Sample rate	$\sigma$	$K$	Pad length
$k$ -mer table	Dorado	R10 sup	400	5000 Hz	0.10	10	5b

In this experiment, we will define and test three different metrics, and then create a codebook with optimal values of these metrics which achieves a maximal quality score.

### 6.3.2 Popularity metric

We will continue to investigate the impact of codon popularity on the quality score. For each codon, its frequency per 1000 codons is available in the TSV data in [20]. We define the popularity metric of a codebook as

$$p = \frac{\text{ave. frequency per 1000 of codons in codebook}}{\text{ave. frequency per 1000 of all codons}}.$$

By the nature of its definition,  $p$  will be close to 1 for most codebooks. Across a large number of codebooks, it is observed to span from 0.5 to 1.7.

### 6.3.3 GC-content

A second metric we will use is the GC-content of the codebook, which is an established metric that is known to impact the stability of a DNA strand [27]. Delahaye found that reads with a low GC-content have fewer errors than reads with a high GC-content, when basecalled using another one of ONT's basecallers named Guppy [24]. The GC-content of a codebook can be written as a proportion or a percentage, and is defined as

$$g = \frac{\text{total G/C bases in codebook}}{\text{total bases in codebook}}.$$

By symmetry, the GC-content of a randomly formed codebook would be 0.5. In theory, this metric may range from 0 to 1. However, there are only 8 codons which can be formed using G and C alone, while our codebook has length 16, meaning at least 8 of the codewords must contain an A or a T. Therefore,  $g$  will lie in the interval  $\left[\frac{1}{6}, \frac{5}{6}\right]$ . It is worth noting that the GC-content of the human genome is 41% [28], which may explain any local peak in the quality score near  $g = 0.41$ .

### 6.3.4 Homopolymerity metric

Research has also shown that nanopores tend to be more erroneous in homopolymer regions, which are regions where the same base appears in succession two or more times [29]. We are evaluating codebooks of codons, so the only possible homopolymer lengths in a codon are 2 or 3. This provides a very coarse scale with which to judge a codebook, and it is unable to distinguish the codebooks {ACA, ATA} and {ACT, ACG}, for which it is clear that the former will permit homopolymers in the transmission message while the latter will not. We therefore develop a homopolymer metric as the probability of a random 2-mer in the transmission message being a homopolymer. A 2-mer can arise in three ways, with equal probability:

- First two bases of a single codon.
- Second two bases of a single codon.
- Last base of one codon, and first base of another codon.

We can work out the probabilities of each of these cases separately and sum them to give a homopolymer metric. For a codebook  $\mathcal{C}$ , this can be defined explicitly as

$$h = \frac{1}{3} \cdot \frac{\sum_{c \in \mathcal{C}} \delta(c[1], c[2])}{|\mathcal{C}|} + \frac{1}{3} \cdot \frac{\sum_{c \in \mathcal{C}} \delta(c[2], c[3])}{|\mathcal{C}|} + \frac{1}{3} \cdot \frac{\sum_{c, d \in \mathcal{C}} \delta(c[3], d[1])}{|\mathcal{C}|^2},$$

where  $\delta$  is the Kronecker delta function.

### 6.3.5 Results generation process

We expect that each of the metrics  $p$ ,  $g$  and  $h$  will have some impact on the quality score of a codebook. To analyse the impact of each metric individually, we create 200 codebooks with a range of values of that metric, whilst the other metrics are kept within control bounds. These bounds are

$$0.95 < p < 1.05,$$

$$0.46 < g < 0.54,$$

$$0.27 < h < 0.33.$$

Once the relationship between the quality score and each metric is understood, 400 codebooks are produced with favourable values of each metric, and the best codebook is taken from these.



## 6.4 Method – Experiment 3

### 6.4.1 Purpose

Experiments 1 and 2 investigate the performance of basecallers and metrics on the quality score of a codebook and conclude by forming a codebook which is close to optimal. Due to the large testing throughput required, this was all done in simulation and is limited in the sense that it does not use real nanopore sequencing data.

To contrast, experiment 3 examines real data obtained from R10 nanopores for another research project being undertaken at Monash University, which is investigating classification directly from squiggles. In their experiment, eight template sequences with length 182nt and their eight reverse complements were synthesised in large volumes. These were then fed through an R10 nanopore and the squiggles were collected, corresponding to one of 16 possible types, which we call reference sequences. The author of that experiment utilised an algorithm to take any particular squiggle and determine which sequence and orientation it best matched, skipping the basecalling step entirely.

The purpose of this experiment is to conduct the same classification using Dorado, and maximise the classification accuracy. Real data also provides us with an opportunity to test some of the assumptions made when the entire pipeline was simulated, and therefore reflect on the reliability of our previous conclusions.

### 6.4.2 Information about the reads

There are 51614 reads in total, conducted on R10 nanopores. Each read is in FAST5 format and is therefore stored as a sequence of integers instead of the actual pA levels. Each read also has a set of channel information, which includes the following:

- Channel number: From 1 to 128 (however many channels are not found – perhaps blocked).
- Digitisation,  $D$ : The number of discrete integer levels used to store the signal. This is always 8192.
- Offset,  $o$ : An integer offset which needs to be added to the digitised sequence as explained below. This is dependent on the channel, and for our reads it remains between 1 and 20.
- Range,  $r$ : The range of current levels (in pA) which can be described by the digitisation. For our reads, this is always 1842.912109375 (regardless of channel).
- Sampling rate: For our reads, this is always 5000Hz.

According to [30], we can convert the digitised sequence stored in the FAST5 read back to the pA measurements using the equation

$$I_{pA} = (I_{dig} + o) \cdot \frac{r}{D}.$$

Therefore, although not stored explicitly, we can recover the sequence of raw current measurements from a given read in a FAST5 file.

### 6.4.3 Classification

We will basecall each read with Dorado's R10 *sup* model and calculate the edit distance between the basecalled sequence and each of the 16 reference sequences. Classification will be with minimum distance decoding, using edit distance.

For improved classification, we will identify and remove reads which we believe are more likely to be in error. We will do this by calculating a *strength*,  $s$ , from the edit distances found, using the equation

$$s = \frac{\text{second smallest edit distance}}{\text{smallest edit distance}} \geq 1.$$

For a read, a large strength indicates that it is much closer (in edit distance) to one reference sequence than to any other reference sequence. To contrast, a strength close to 1 indicates that the read is approximately equally close to two reference sequences. In the latter case, it is more likely that a misclassification has occurred, and we would like to filter out such reads.

## 7 Results and Discussion

---

### 7.1 Experiment 1: Preliminary investigation

#### 7.1.1 Preamble on testing setup

The expected noise for the R9 pore has magnitude 0.14, which was found by averaging the *sd* column of Scrappie's squiggle generator output for 1000 bases. The mean dwell time for the R9 pore is 10 samples per base, as this corresponds to sampling 400 bases per second at 4kHz sample rate, which is the former rate used by ONT, and the rate Scrappie was trained on [21]. The newest R10 pore is sampled at 5kHz, corresponding to a mean dwell time of 12.5 samples per base.

To standardise, we define the *operating condition* to be  $\sigma = 0.10$ ,  $K = 10$ . When one of  $\sigma$  or  $K$  is the independent variable being tested, the other is kept at operating condition.

The two methods of squiggle generation being tested are Scrappie and  $k$ -mer tables.

The two basecallers being tested are Scrappie and Dorado. The latter has three models, which we test separately.

In one of our tests, we vary the padding length to find the optimal number of bases.

For each test, a parameter block is included to describe the constant and variable parameters in the test. Note that quality score is the dependent variable in all of the following plots, however the vertical scale used differs from plot to plot.

### 7.1.2 Results: R9 Scrappie generation and Scrappie basecalling

Table 9: Parameter block.

Squiggle generation	Basecaller	Model	BPS	Sample rate	$\sigma$	$K$	Pad length
Scrappie	Scrappie	R9	400	4000 Hz	Varied	10b	5b

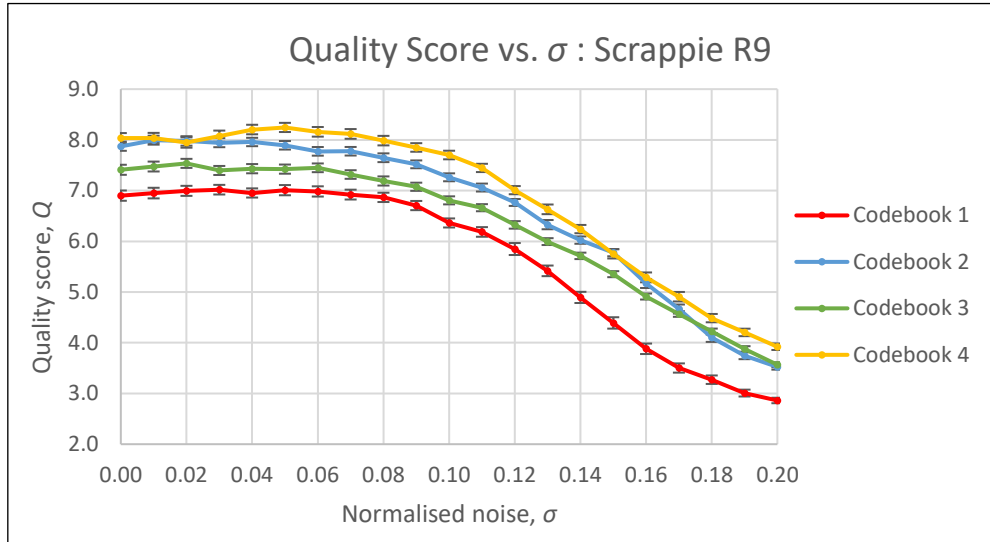


Figure 11: Quality score vs.  $\sigma$  for the four chosen codebooks. Vertical error bars denote a 95% confidence interval.

Table 10: Parameter block.

Squiggle generation	Basecaller	Model	BPS	Sample rate	$\sigma$	$K$	Pad length
Scrappie	Scrappie	R9	400	4000 Hz	0.10	Varied	5b

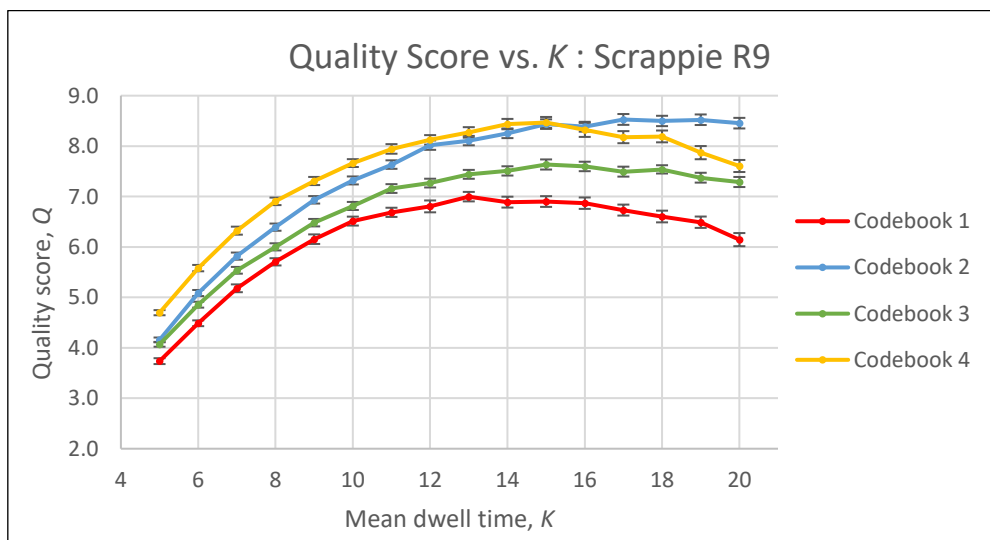


Figure 12: Quality score vs.  $K$  for the four chosen codebooks. Vertical error bars denote a 95% confidence interval.

### 7.1.3 Results: R9 Scrappie generation and Dorado 'fast' basecalling

Table 11: Parameter block.

Squiggle generation	Basecaller	Model	BPS	Sample rate	$\sigma$	$K$	Pad length
Scrappie	Dorado	R9 fast	400	4000 Hz	Varied	10	5b

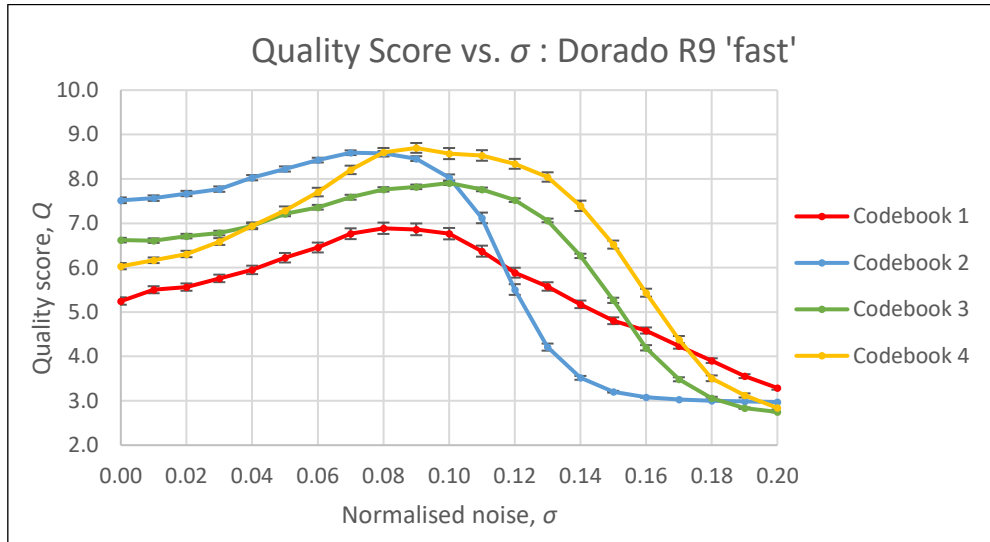


Figure 13: Quality score vs.  $\sigma$  for the four chosen codebooks. Vertical error bars denote a 95% confidence interval.

Table 12: Parameter block.

Squiggle generation	Basecaller	Model	BPS	Sample rate	$\sigma$	$K$	Pad length
Scrappie	Dorado	R9 fast	400	4000 Hz	0.10	Varied	5b

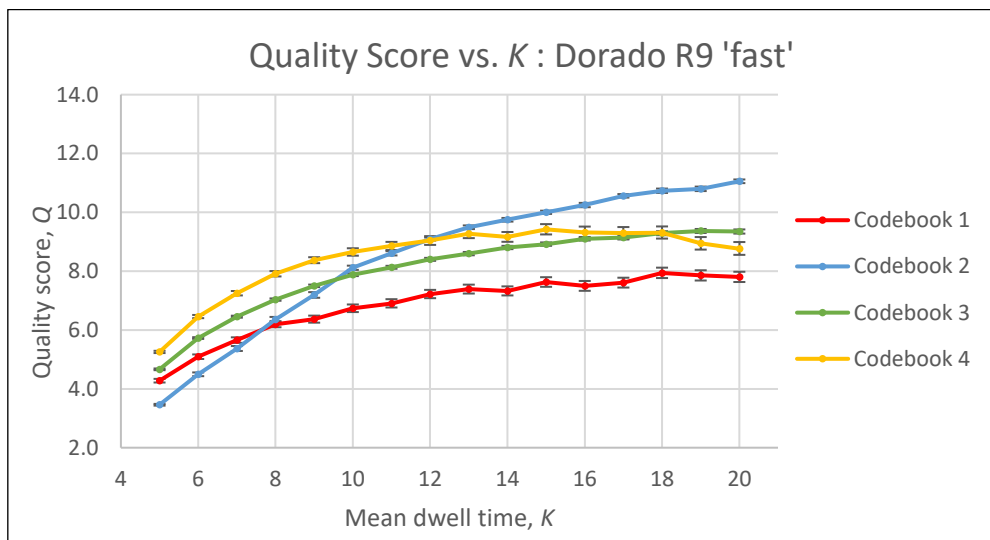


Figure 14: Quality score vs.  $K$  for the four chosen codebooks. Vertical error bars denote a 95% confidence interval.

#### 7.1.4 Results: R9 Scrappie generation and Dorado 'hac' basecalling

Table 13: Parameter block.

Squiggle generation	Basecaller	Model	BPS	Sample rate	$\sigma$	$K$	Pad length
Scrappie	Dorado	R9 hac	400	4000 Hz	Varied	10	5b

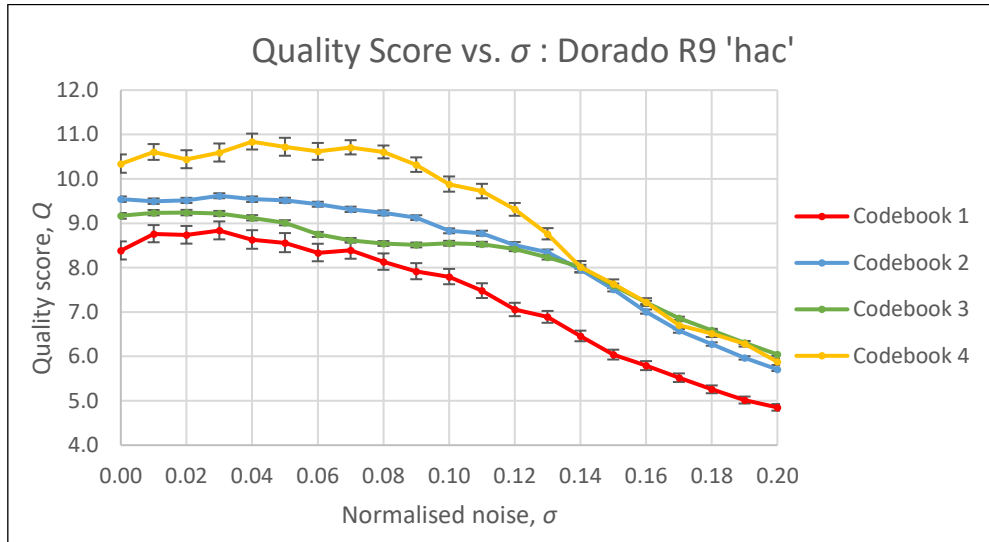


Figure 15: Quality score vs.  $\sigma$  for the four chosen codebooks. Vertical error bars denote a 95% confidence interval.

Table 14: Parameter block.

Squiggle generation	Basecaller	Model	BPS	Sample rate	$\sigma$	$K$	Pad length
Scrappie	Dorado	R9 hac	400	4000 Hz	0.10	Varied	5b

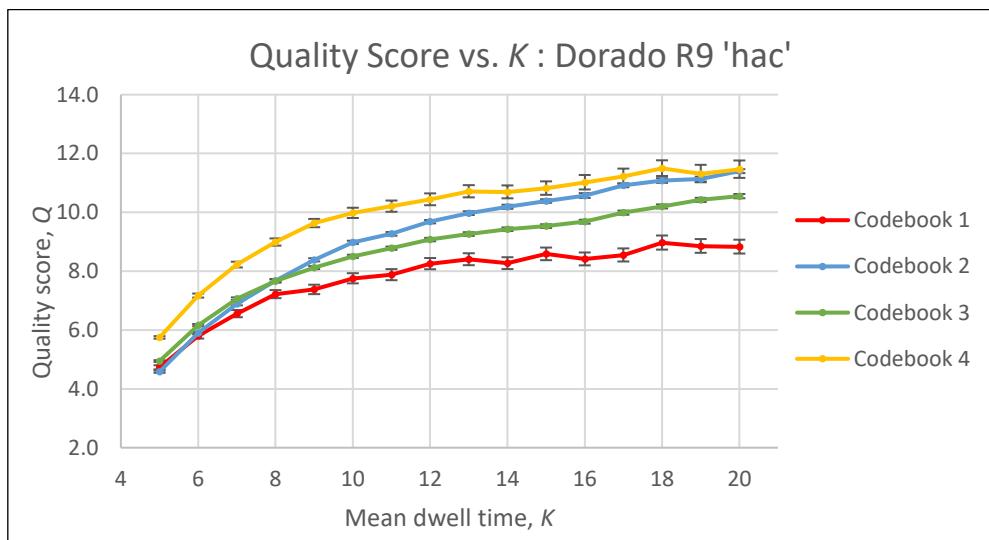


Figure 16: Quality score vs.  $K$  for the four chosen codebooks. Vertical error bars denote a 95% confidence interval.

### 7.1.5 Results: R9 Scrappie generation and Dorado 'sup' basecalling

Table 15: Parameter block.

Squiggle generation	Basecaller	Model	BPS	Sample rate	$\sigma$	$K$	Pad length
Scrappie	Dorado	R9 sup	400	4000 Hz	Varied	10	5b

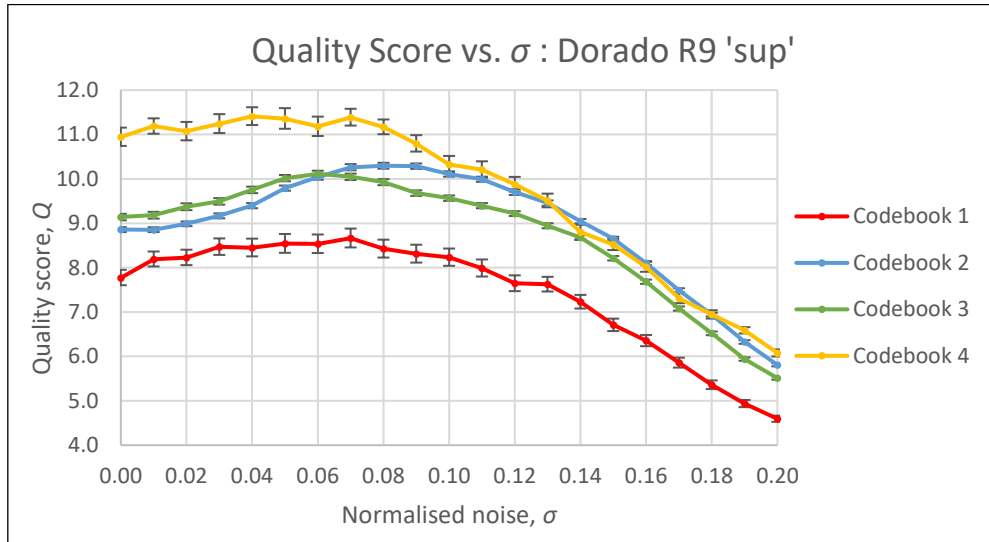


Figure 17: Quality score vs.  $\sigma$  for the four chosen codebooks. Vertical error bars denote a 95% confidence interval.

Table 16: Parameter block.

Squiggle generation	Basecaller	Model	BPS	Sample rate	$\sigma$	$K$	Pad length
Scrappie	Dorado	R9 sup	400	4000 Hz	0.10	Varied	5b

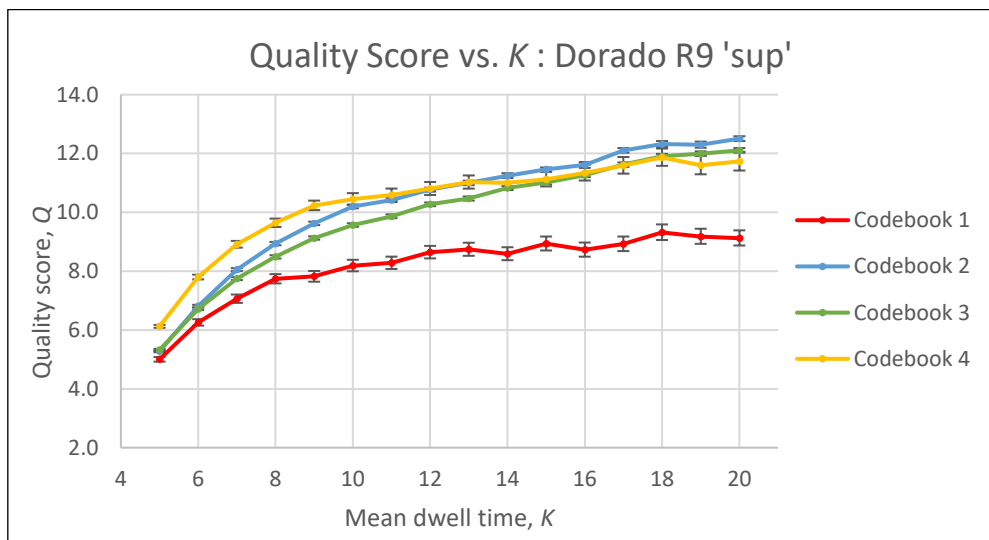


Figure 18: Quality score vs.  $K$  for the four chosen codebooks. Vertical error bars denote a 95% confidence interval.

### 7.1.6 Results: $k$ -mer generation and Dorado 'sup' basecalling – variable padding length

Table 17: Parameter block.

Squiggle generation	Basecaller	Model	BPS	Sample rate	$\sigma$	$K$	Pad length
$k$ -mer table	Dorado	R9 sup	400	4000 Hz	0.10	10	Varied

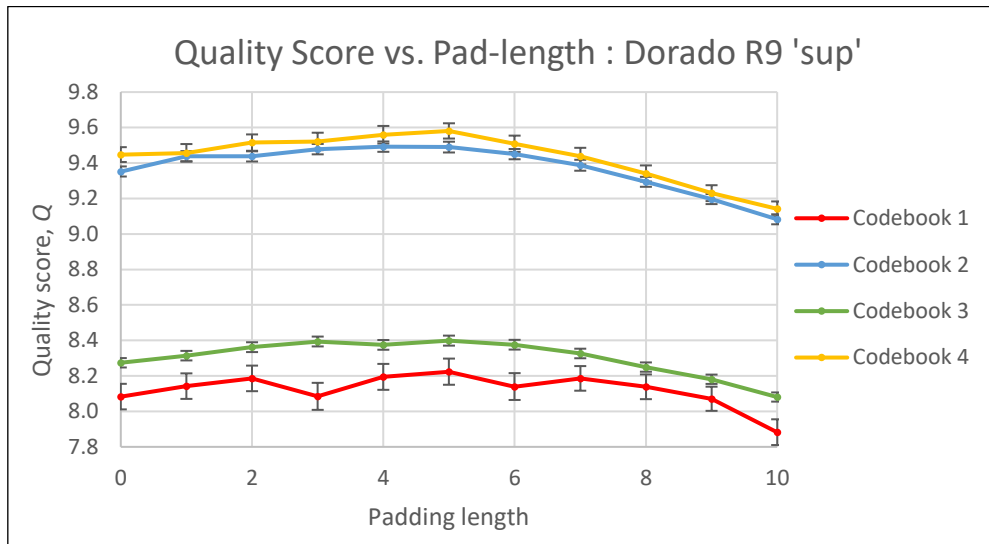


Figure 19: Quality score vs. padding length for the four chosen codebooks. Vertical error bars denote a 95% confidence interval.

Table 18: Parameter block.

Squiggle generation	Basecaller	Model	BPS	Sample rate	$\sigma$	$K$	Pad length
$k$ -mer table	Dorado	R10 sup	400	5000 Hz	0.10	10	Varied

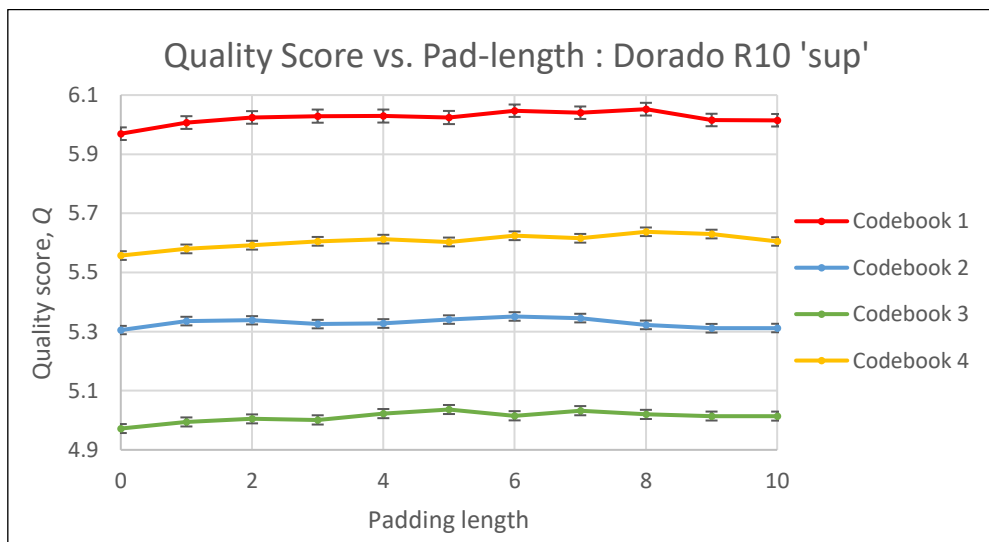


Figure 20: Quality score vs. padding length for the four chosen codebooks. Vertical error bars denote a 95% confidence interval.



### 7.1.7 Results: R9 $k$ -mer generation and Dorado 'sup' basecalling

Table 19: Parameter block.

Squiggle generation	Basecaller	Model	BPS	Sample rate	$\sigma$	$K$	Pad length
$k$ -mer table	Dorado	R9 sup	400	4000 Hz	Varied	10	5b

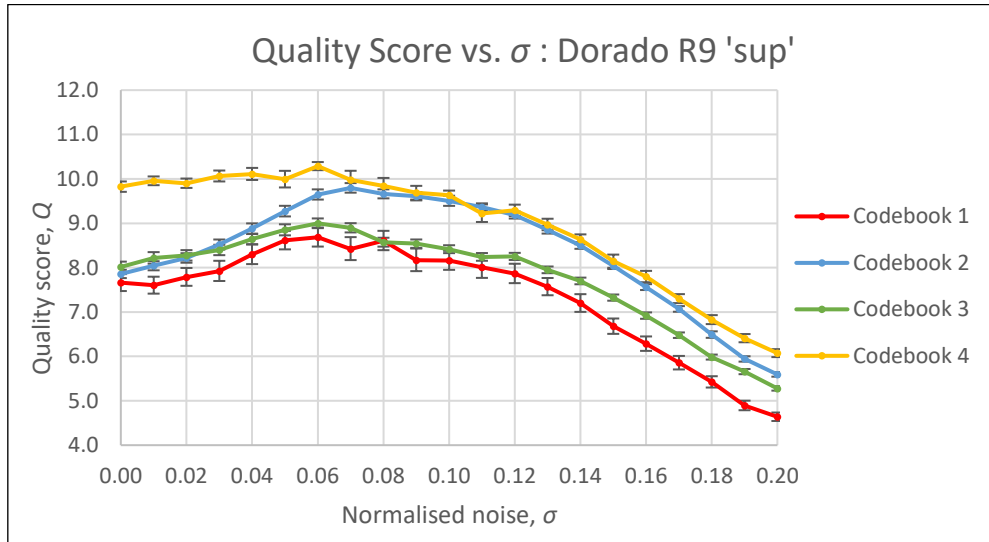


Figure 21: Quality score vs.  $\sigma$  for the four chosen codebooks. Vertical error bars denote a 95% confidence interval.

Table 20: Parameter block.

Squiggle generation	Basecaller	Model	BPS	Sample rate	$\sigma$	$K$	Pad length
$k$ -mer table	Dorado	R9 sup	400	4000 Hz	0.10	Varied	5b

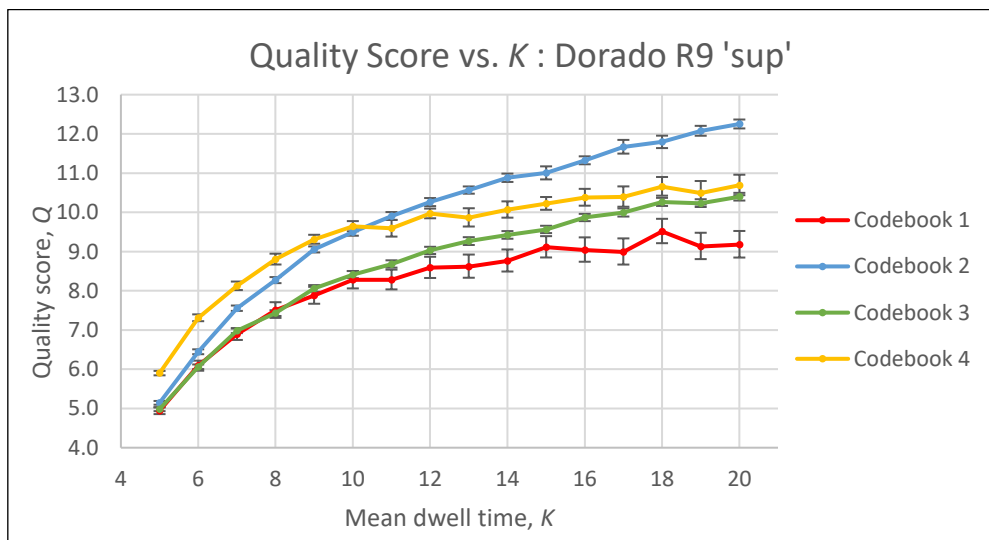


Figure 22: Quality score vs.  $K$  for the four chosen codebooks. Vertical error bars denote a 95% confidence interval.

### 7.1.8 Results: R10 $k$ -mer generation and Dorado 'sup' basecalling

Table 21: Parameter block.

Squiggle generation	Basecaller	Model	BPS	Sample rate	$\sigma$	$K$	Pad length
$k$ -mer table	Dorado	R10 sup	400	5000 Hz	Varied	10	5b

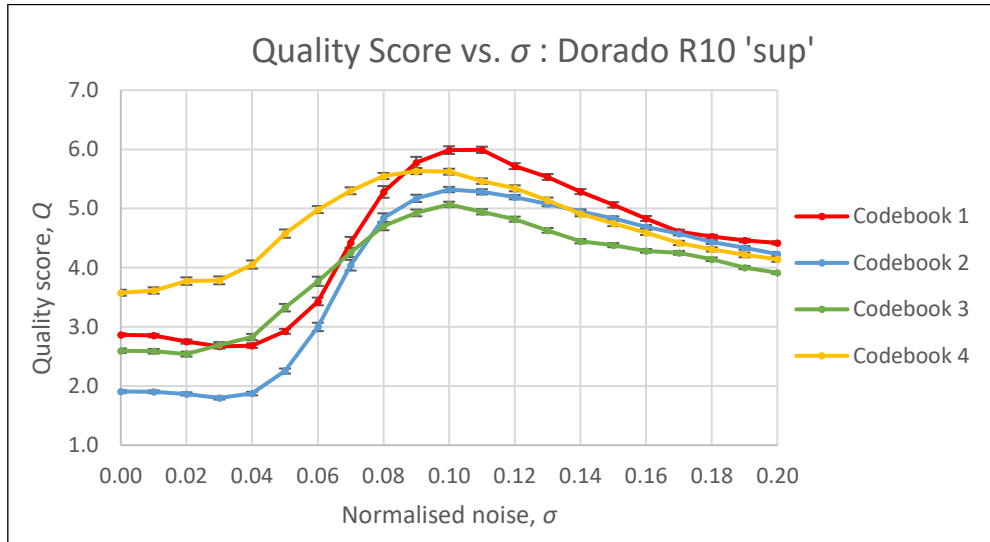


Figure 23: Quality score vs.  $\sigma$  for the four chosen codebooks. Vertical error bars denote a 95% confidence interval.

Table 22: Parameter block.

Squiggle generation	Basecaller	Model	BPS	Sample rate	$\sigma$	$K$	Pad length
$k$ -mer table	Dorado	R10 sup	400	5000 Hz	0.10	Varied	5b

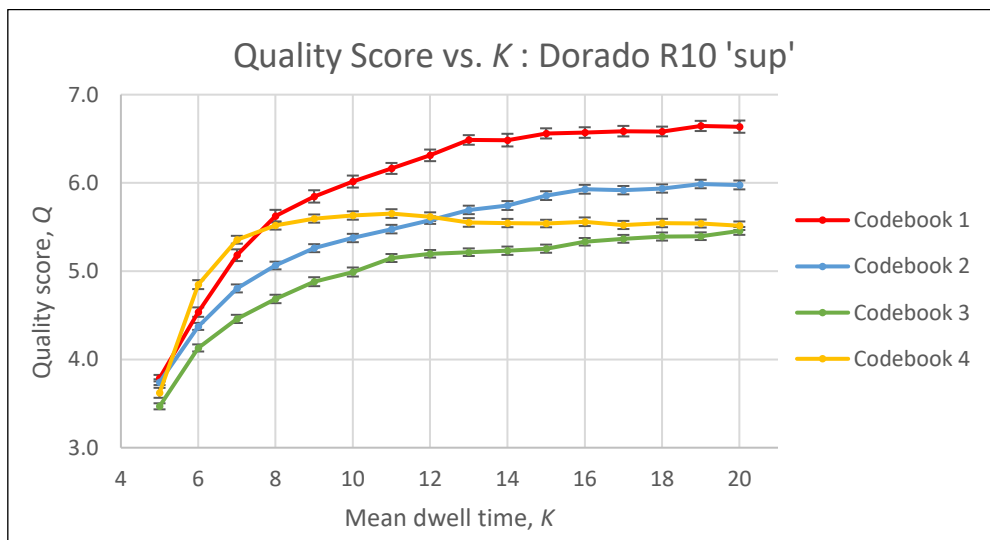


Figure 24: Quality score vs.  $K$  for the four chosen codebooks. Vertical error bars denote a 95% confidence interval.

### 7.1.9 Discussion of preliminary results

The data obtained in Experiment 1 give evidence for several conclusions which we discuss.

#### ***A codebook of more popular codons does not necessarily reduce the error rate.***

The impact of codon popularity is observed by comparing codebooks 1, 2 and 3, which are the popular, unpopular and control codebooks respectively. It was hypothesised that Scrappie and Dorado basecallers would perform better when given these codons.

For R9 pores, we observe the opposite effect. Figure 11 to Figure 22, which correspond to the R9 pore, show that codebook 1 has the lowest quality score across almost all noise levels and dwell times, while codebook 2 has the highest. This is found unanimously across the two squiggle generation models and various basecalling models.

To contrast, for the R10 pore our hypothesis is supported. Figure 23 and Figure 24 show R10 Dorado basecalling with  $k$ -mer squiggle generation, in which codebook 1 largely outperforms the unpopular and control codebooks.

The preliminary results for R9 and R10 are therefore in opposition, and we cannot conclude that using more popular codons leads to improved quality of a codebook. It is likely that there are other codebook metrics which play an important role in codebook performance.

#### ***A padding of 5 bases is optimal for synthetic R9 sequences simulated using $k$ -mer tables.***

As mentioned in Section 6.2.5, a padding is required as basecalling software is less reliable at the start and end of sequences. We fixed the padding length at 5 bases for all experiments. This choice is justified by the experiment in which padding is varied, with the results shown in Figure 19.

#### ***Dorado models perform in the order expected, and the 'fast' model performs similarly to Scrappie.***

Figure 25 shows the noise performance curves of the control codebook extracted from Figures Figure 11, Figure 13, Figure 15 and Figure 17, which correspond to R9 sequences with Scrappie squiggle generation.

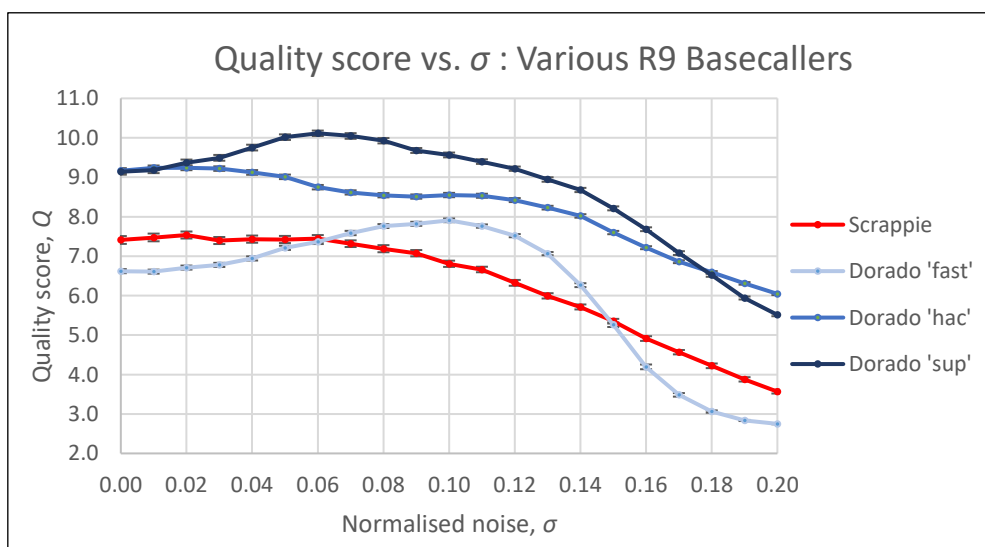


Figure 25: Quality score vs.  $\sigma$  for the four basecallers. Vertical error bars denote a 95% confidence interval.

At the operating point of  $\sigma = 0.10$ , we see that *sup* gives a quality score which is Q1.0 higher than *hac*, which is in turn Q0.6 higher than *fast*. All three Dorado models performs better than the Scrappie basecaller, whose quality score is Q1.1 lower than the *fast* model. This is in line with expectations, with Dorado being the currently used software and Scrappie being archived two years ago.

### ***Performance varies continuously with noise, mean dwell time and the codebooks themselves.***

If we consider any single codebook in any of the performance plots, we see that its quality score does not change significantly as a result of a small change in noise or mean dwell time. The relationship is continuous as opposed to chaotic, and a benefit of this is that we can fix a noise and mean dwell time in experiment 2 without worrying that the exact choice will greatly impact the behaviour of the codebooks.

Another question of interest is whether the relationship between quality score and the codebooks themselves is continuous. Codebooks are not numerical variables, but we can attempt to answer by taking a codebook, altering a single base in a single codon, and observing how the quality score changes. We generated 6 perturbations of the control codebook, and tested these with Dorado R10 *sup*, with *k*-mer squiggle generation. The quality score changed by very little for each perturbation, with the largest change observed being Q0.16. To contrast, the markedly different codebooks tested in experiment 1 had quality scores differing by Q0.3 or more at operating conditions. This suggests that quality score is indeed continuous with respect to the codebooks themselves.

### ***Dorado performs better with some noise than with no noise.***

From Figure 23, we see that in Dorado R10 *sup*, the quality score improves by approximately 3 as the noise is increased from 0.04 to 0.10. This contradicts the general rule in communication theory that increasing noise power will increase the error rate, and thus reduce the quality score. However, it is not surprising in this situation as the Dorado models have been trained on real data which would possess some level of noise.

On the other hand, from Figure 11 we see that the quality score of R9 reads which are basecalled using Scrappie fall as the noise level is increased. As Scrappie is a more primitive basecaller than Dorado, it is not surprising that it doesn't exhibit the same degree of tuning to the expected noise. In practice, noise is produced in the physical nanopore itself and therefore is not a parameter we can vary. These results show that it is important to use basecalling models which are trained on the exact pore generation being used, and to retrain these models when new pores, perhaps with lower noise, are released.

### ***R10 squiggle generation is less reliable than R9 squiggle generation when using *k*-mer tables.***

From Figure 23, at the operating point we see that Dorado R10 *sup* has an average quality score of Q5.5 across the four codebooks, while Figure 21 shows that this is Q9.0 for R9. One difference between these two runs is that a 9-mer table is used to simulate the R10 pore, while a 6-mer table is used for the smaller R9 pore. The data therefore suggests that 9 bases is insufficient in modelling the R10 pore during the signal generation step. We cannot conclude whether 6 bases is sufficient for the R9 pore, but observe that it provides more reliable output than the former case.

To combat this problem, we would ideally want to use a 10-mer table or higher for R10, but these are not yet available. To generate such a table, one would need to create strands containing all possible 10-mers, which even if done efficiently would require at least 1 million bases. Furthermore, to achieve accurate and reliable current levels, many trials would need to be conducted, perhaps requires billions of bases, which would be impractical to synthesise. A possibility would be to use the human genome which has already been sequenced. This has approximately 3 billion bases but may still miss some 10-mers [13].

## 7.2 Experiment 2: Investigating codebook metrics

### 7.2.1 Results: Popularity metric

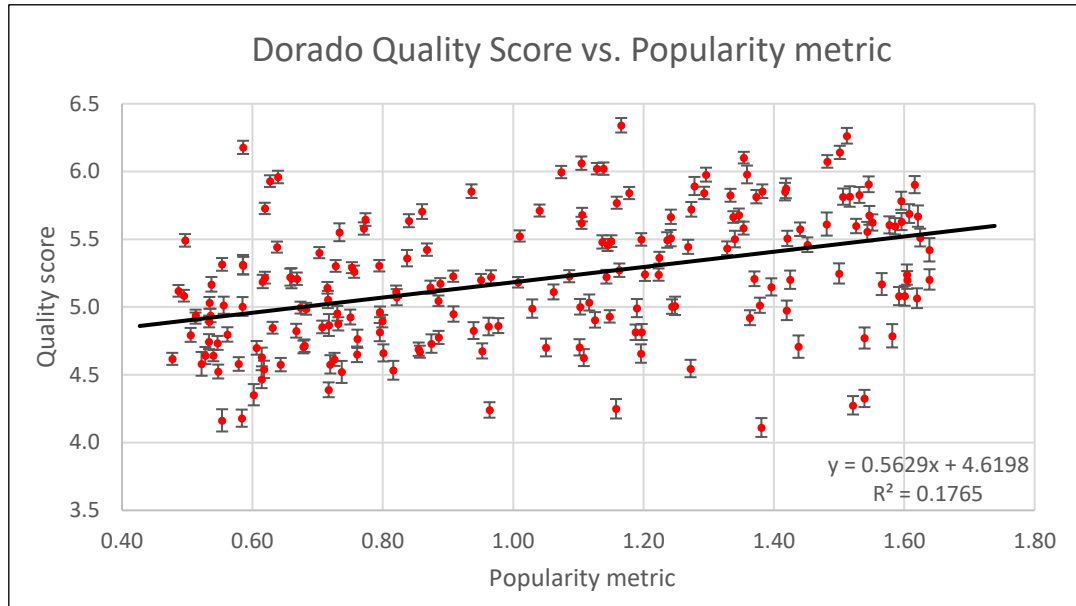


Figure 26: Quality score vs popularity. The quality score of 200 codebooks is plotted against their popularity metric.

A linear regression was conducted, finding very weak positive correlation with  $R^2 = 0.1765$  and a fit given by

$$\hat{Q} = 0.5629p + 4.6198.$$

### 7.2.2 Results: GC-content

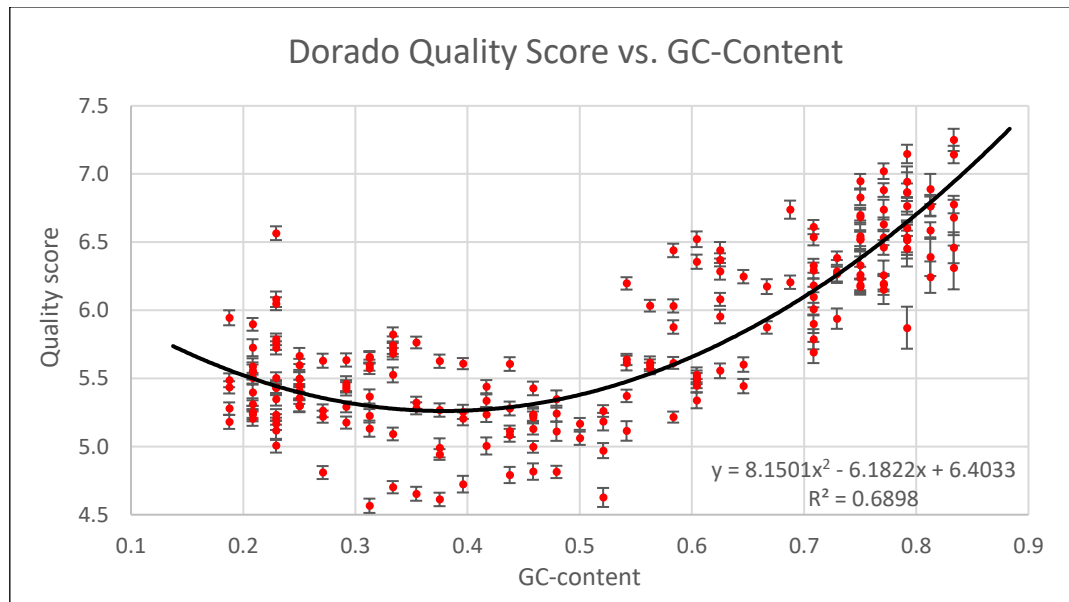


Figure 27: Quality score vs GC-content. The quality score of 200 codebooks is plotted against their GC-content.

A quadratic regression was conducted, finding moderate correlation with  $R^2 = 0.6898$  and a fit given by

$$\hat{Q} = 8.1501g^2 - 6.1822g + 6.4033.$$

### 7.2.3 Results: Homopolymer metric

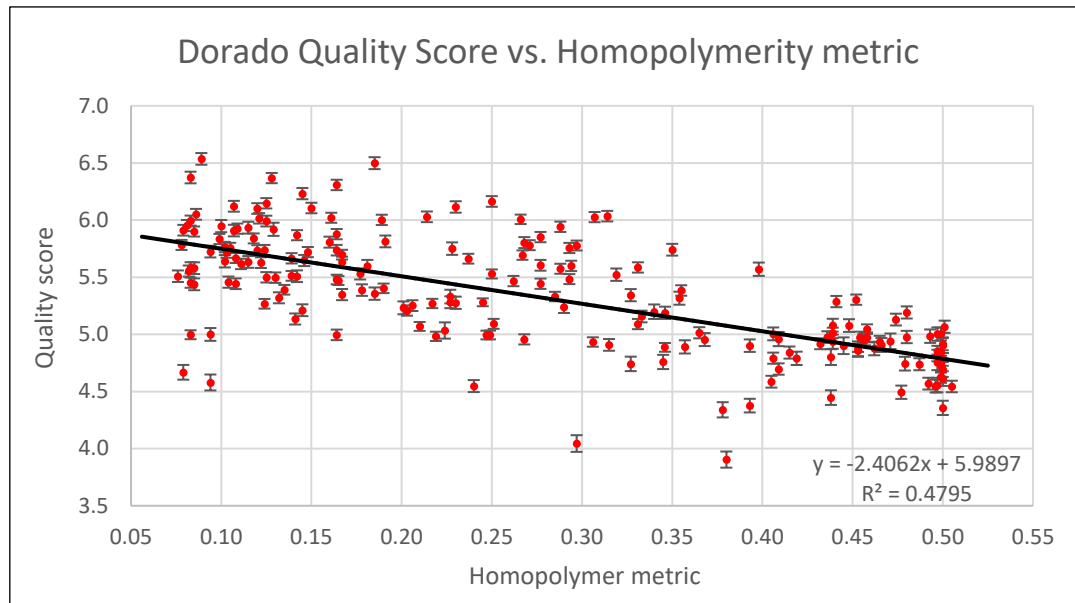


Figure 28: Quality score vs homopolymerity. The quality score of 200 codebooks is plotted against their homopolymerity metric.

A linear regression was again conducted, finding moderate negative correlation with  $R^2 = 0.4795$  and a fit given by

$$\hat{Q} = -2.4062h + 5.9897.$$

#### 7.2.4 Discussion of codebook metrics

The results found varying levels of correlation with each of the metrics.

##### ***Popularity is weakly positively correlated with the quality score.***

The experiment found that popularity has a positive correlation with the quality of the codebook. The codebooks with the highest popularity have a quality score which is on average Q0.6 higher than those with the least popularity. However, with an  $R^2$  value of 0.1765, the correlation observed is very weak. The primary aim of this project was to determine whether a codebook of frequent (popular) codons would be optimal. This result weakly supports the hypothesis that using more popular codons improves the error performance, but we cannot conclude that such a choice is optimal.

These conclusions are in line with experiment 1, which found opposing results from R9 and R10 as to whether quality score improves with popularity. A possible explanation for why popularity doesn't have a significant impact is that the nanopore and basecaller do not examine the strand one codon at a time and do not know where codons start and end.

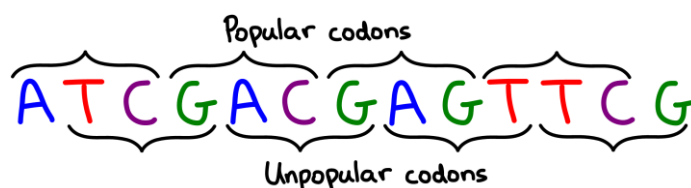


Figure 29: Diagram showing how a lack of synchronisation makes popular and unpopular codons indistinguishable.

Figure 29 shows a part of a DNA strand which was formed by concatenating popular codons. However, if we segment this differently, it becomes a concatenation of unpopular codons. Furthermore, any attempt at synchronization is impeded by the possibility that bases are inserted or deleted during synthesis. This justifies our observation that codon popularity does not significantly impact the quality score.

##### ***GC-content is moderately non-linearly correlated with the quality score.***

Figure 27 shows that quality score is highest when GC-content is maximised, and lowest when GC-content is within 40-50%. A quadratic fit has been chosen, although there is no physical reason for this to be the correct fit – another possibility would be a two-piece linear fit.

These results contrast those found previously with Guppy [24], in which higher GC-content yielded a higher error rate, and thus a lower quality score. Those results were with the R9 pore, suggesting that the effects of GC-content have varying impact, and need to be studied for each new nanopore and basecaller individually. Optimising GC-content to improve codebooks is therefore unlikely to be a sustainable solution, as DNA data storage is proposed for long-term archival storage and the set of nanopores and basecallers being used in the distant future will certainly be different to those used today.

The human genome has a GC-content of 41% [28], so we expected the basecaller to perform better for reads with GC-contents close to this. We see however that the opposite occurs, with performance being worst in this region. Dorado is also trained on plant and animal DNA more broadly, but this also has a GC-content of 41% [31], and so does not explain the inverted results. We can conclude that training bias does not play a significant role in influencing the effect of GC-content.

##### ***Homopolymerity is moderately negatively correlated with the quality score.***

Figure 28 shows that quality score is negatively linearly correlated with homopolymerity, which is in concurrence with previous studies. The phenomenon likely to be present here is that the nanopore current fluctuates by a smaller amount when the same base is encountered twice in succession. This fluctuation is therefore more likely to be missed and an error ensues.

***Properties of a codebook besides those discussed here also impact the quality score.***

In this experiment, we created and tested three metrics we believed would impact codebook performance, but did not suggest these to be comprehensive. In each of performance plots in experiment 2, we vary one of the metrics whilst holding the other two nearly constant. Nonetheless, there is significant variation in the results found, with quality score residuals of each fit having a standard deviation of roughly Q0.4. This suggests that other properties of a codebook also influence the quality score, and further investigation is necessary to understand the relationship better. We have looked at GC-content, but a more refined approach would be to look at A, C, G and T concentrations independently. Each base has a unique impact on the nanopore current, and therefore may have more subtle effects on the quality score achieved. Additionally, our homopolymerity metric is defined on 2-mers but this could be broadened to encountering  $k$  consecutive bases of the same type for  $k \geq 3$ .

***An optimal codebook can be found by creating codebooks with high GC-content and low homopolymerity metric.***

After testing the metrics, 400 *good* codebooks were created with  $g > 0.70$  and  $h < 0.20$ . These were then sent through the pipeline, and all were observed to have a quality score above Q6.5. In contrast, most of the codebooks generated when testing the metrics individually have a quality score between Q4.5 and Q6.5, so we are confident that producing codebooks with optimal parameters does indeed lead to improved performance. The highest quality score obtained from this set was Q8.1, belonging to the following codebook.

Codebook: AGC CAC CCA CCG CCT CGA CGC CTC CTG GAC GCA GCC GCG GCT GTC TCG

Metric values:  $p = 1.06, g = 0.75, h = 0.19$ .

This provides a solution to our search for an optimal codebook. We note that there are on the order of  $10^{14}$  possible codebooks, and we have tested merely 1000 of these. Assuming the quality scores of codebooks are normally distributed, the expected number of codebooks with a quality score above Q8.1 is on the order of  $10^9$ . Hence, it is almost certain that the above codebook is not truly optimal, but is certainly quite exceptional.

### 7.2.5 Limitations and future work

The major limitation of the first two experiments is that they are entirely done in simulation. Despite adding noise and a geometric dwell time to the squiggles we generate, this would not perfectly model the nanopore. Furthermore, when real nanopores read DNA, there is a degree of uncertainty as to the exact moment when the strand begins or finishes passing through the nanopore, which we have not fully accounted for. In this regard, the results we generate may have higher quality scores than in practice.

On the other hand, as mentioned in experiment 1, the 9-mer tables do not seem to fully model the complex R10 nanopore, and our generated squiggles may be of low quality due to this shortcoming. Not having 10-mer tables or higher is therefore another limitation which may mean our quality scores are lower than in practice. The two limitations have contrasting impacts, and it is difficult to determine which affects the quality score more significantly.

Future work could create a larger number of metrics such as those described previously in this section. A selection of the best and worst codebooks found in simulation could be synthesised into real DNA, and basecalled using Dorado models. If they perform in the order expected, this would give greater confidence to the results obtained in these two experiments.



## 7.3 Experiment 3

### 7.3.1 Results and Discussion: Classification

The classification process used minimum edit distance decoding to match each of the 51614 reads against one of 16 reference sequences. The number of misclassified reads was 532, giving a classification error of 1.03%. The error rate for each individual sequence is found in Table 23.

Table 23: Classification error by strand and direction, in %. The dataset does not have an equal number of reads for each sequence, so the simple average differs significantly from the weighted average.

Strand	Reverse	Template
1	0.14	1.89
2	0.00	25.88
3	20.31	0.73
4	0.26	1.64
5	0.50	0.79
6	0.18	6.77
7	0.21	19.68
8	0.17	0.50
Overall (simple average)	2.72	7.24
Overall (weighted average)	0.526	4.892

For most of the strands and directions, we see that the classification error is lower than 1%. A high classification error is observed for *reverse\_3*, *template\_2* and *template\_7*, potentially making these sequences unfit for communication. The dataset does not have an equal proportion of each reference sequence, meaning that the average classification error is different to the weighted average, where the weighting is by number of reads. For example, *reverse\_1* and *reverse\_6* together comprise half of the dataset, so the weighted average is biased significantly towards these two sequences.

### 7.3.2 Results and Discussion: Improved classification

While 1.03% is a formidable error rate, we attempt to improve this by filtering out bad reads before classification. Previously, classification was done by taking a read, calculating its edit distance to each reference sequence, and choosing the minimum. Here we calculate a *strength* for each read in the manner defined in Section 6.4.3. Setting a minimum strength threshold reduces the classification error, but also means that we lose some of our reads. The relationship between classification error and reads lost is shown in Figure 30.

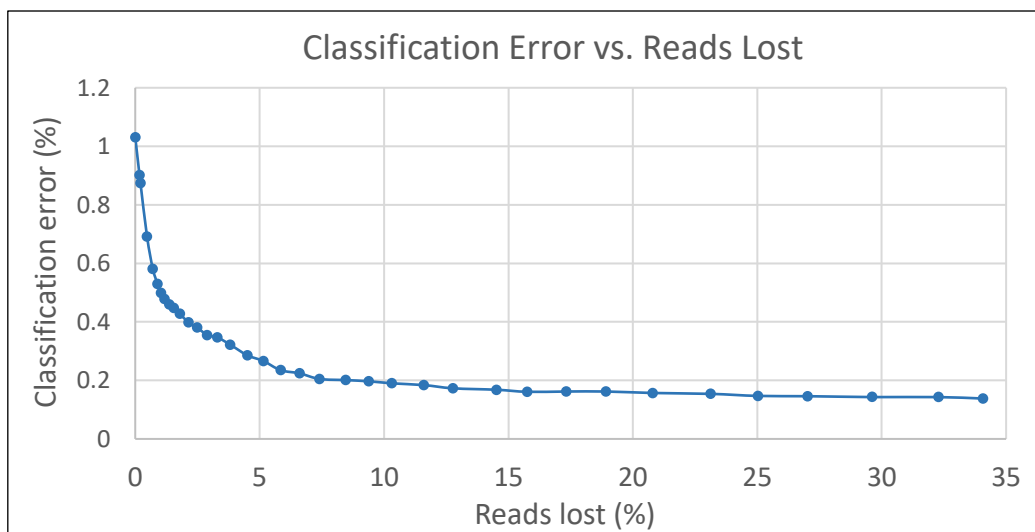


Figure 30: Classification error vs. reads lost.

We observe that classification error can be halved by dropping 1% of reads and can be reduced to below 0.2% by dropping 10% of reads. Increasing the strength threshold further has a diminishing improvement on the classification accuracy.

## 8 Conclusion

---

We have seen that DNA allows energy efficient and long-lasting data storage at a molecular level. In this project, we have considered the problem of coding for the DNA channel with nanopore sequencing, focusing on basecallers Scrappie and Dorado. In particular, we have investigated the usage of codons for creating transmission messages, and the error landscape of such a choice.

In our preliminary investigation, we found that using more popular codons does not directly lead to improved codebook performance, and that there are other factors at play. We also found that noise in the current signal does not have a conformal relationship with the error rate, with basecallers requiring some noise to perform optimally, likely due to their training data being noisy. The quality score was observed to vary continuously with noise, mean dwell time and the codebooks themselves, meaning it is sensible to search for an optimal codebook without worrying that slight changes in the operating conditions will vastly change the results.

In our second experiment, we found that Dorado R10 *sup* performance improved slightly with codon popularity, however with very low correlation. To contrast, GC-content and homopolymerity had stronger correlation, with high GC-content and low homopolymerity creating the best codebooks. These two metrics are not unique to codons, and can be defined for any codebooks, suggesting that there is no additional benefit from coding using codons. A large number of codebooks were constructed with these favourable metrics, with the best among these having half as many errors as an average codebook, with a quality score of Q8.1.

Our final experiment was detached from the others, looking at the problem of classifying reads. We were able to achieve mostly successful classification, with an error of 1.03%. This error rate could be halved by discounting the worst 1% of reads.

Despite not finding any significant benefit from using codons, this experiment provides strong evidence that certain aspects of a codebook impact the error rate it incurs in the nanopore channel. When designing codebooks for practical use, the channel cannot be abstracted away as a symmetric channel, and metrics such as GC-content and homopolymerity need to be addressed to ensure minimal error rate and optimal performance. Future research on this topic could explore a larger number of metrics and keep track of whether the observed relationships change when newer basecallers or nanopores are examined.

## 9 Reflection on Project Management

---

### 9.1 Project Scope

The scope of this project is controlled by the aim and objectives set out in Section 4. In particular, it is expected that a report will be produced with an in-depth investigation on the use of codons for a codebook used in DNA-based data storage with nanopore sequencing.

This investigation should include analysis of data and simulations from Scrappie and Dorado, but it is not expected to draw from a real nanopore sequencer due to the costs and labwork involved in implementing this. The report should include some discussion of the accuracy and limitations of the software used, and potential impacts of this on the suitability of the coding scheme.

Another major task within the scope of the project is documenting the methods used to a level that allows others to reproduce the work conducted. This will aid other researchers in the field and protect the student from legal repercussions from external parties who wish to discredit the results.

It is not in the scope of this project to produce a coding scheme that is an improvement to existing coding schemes. Instead, a codon-based coding scheme should be investigated, and some justifiable optimisations made. It is intended that the methods and ideas developed may aid future research and formulations of improved coding schemes.

### 9.2 Project Plan & Timeline

This final-year project began at the start of semester 2, 2023, and was completed at the end of semester 1, 2024. The year-long endeavour provided an opportunity to conduct a deep study of the field and analyse at length the feasibility of a codon-based coding scheme. An approximate timeline is shown in Table 24, along with the objectives to which each time-period corresponds.

The focus of semester 2, 2023 was to perform background research on DNA data storage using nanopore sequencing, and to collect data and analyse the performance of a codebook of codons. The first major deliverable was the progress report in mid-November. This showcased the analysis conducted on an unoptimized codebook of codons.

The summer break was to be spent researching deeper into the field and reading a wider array of papers, with an aim of finding inspiration to refine and optimise the codebook of codons. During this time, at least one full day was to be spent each week working on the project, with a goal of completing a significant portion of the work before the final semester. The student would also gain further capability in using Dorado and associated software. The student purchased a computer with a GPU, so that research could continue even when the student was not at their university campus. Four new codebooks would be proposed, implemented and tested before the start of semester 1, 2024.

The next major deliverables were the project poster, video presentation and this report, due at the end of semester 1, 2024. The focus of this semester was conducting further analysis on the refined codebook of codons, in both simulation and theory. Several weeks had been kept free in case the need arises to revise the codebook multiple times. Other tasks are expected to arise which are not currently known, so it is expected there will be sufficient work to do during the gaps in the original timeline.

### 9.2.1 Original timeline

Table 24: Original project timeline

Timeframe	Tasks	Objectives
Pre semester – 10 July 2023 to 23 July 2023		
July	Read about probability and communication. Learn basics about DNA storage.	[O.1], [O.3]
ENG4701 – 24 July 2023 to 19 November 2023		
Weeks 1-2	Read about information theory and DNA storage	[O.1], [O.2], [O.3]
Weeks 3-4	Read more broadly about DNA storage. Familiarise with Scrappie.	[O.1], [O.2], [O.5]
Weeks 5-6	Summarise research on DNA storage and nanopores. Write project proposal.	[O.1], [O.2], [O.3], [O.4]
Weeks 7-8	Learn about existing models. Implement codon-code in Scrappie and basecall with Dorado.	[O.3], [O.4], [O.5]
Weeks 9-10	Test simplistic codon-code. Analyse performance. Determine theoretical performance.	[O.5], [O.6]
Weeks 11-12	Compare theoretical and practical performance. Write progress report.	[O.7], [O.8]
SWOTVAC and Exams	Collect papers to read over the Summer.	
Summer break – 20 November 2023 to 25 February 2024		
November	Read papers on coding for nanopore sequencing.	
December	Explore similar problems in other fields.	[O.9]
January	Refine and propose at least two codebooks.	[O.9]
February	Implement and test codebooks with Scrappie and Dorado.	[O.9]
ENG4702 – 26 February 2024 to 26 May 2024		
Weeks 1-2	Conduct theoretical analysis of codebooks. Compare theoretical and actual results. Refine codebook further if necessary.	[O.9], [O.10]
Weeks 3-4	Gap for unforeseen circumstances.	
Weeks 5-6	Conduct further analysis of codebook. Research simulation-related issues of codebook.	[O.10]
Weeks 7-8	Gap for unforeseen circumstances.	
Week 9	Review, discuss and finalise project with supervisor.	
Weeks 10-11	Create project poster and video. Write final report.	[O.11]
Week 12	Write final report.	[O.11]
SWOTVAC	Poster presentation	[O.11]

## 9.2.2 Final timeline

Table 25: Final project timeline

Timeframe	Tasks	Objectives
Pre semester – 10 July 2023 to 23 July 2023		
July	Read about probability and communication. Learn basics about DNA storage.	[O.1], [O.3]
ENG4701 – 24 July 2023 to 19 November 2023		
Weeks 1-2	Read about information theory and DNA storage	[O.1], [O.2], [O.3]
Weeks 3-4	Read more broadly about DNA storage. Familiarise with Scrappie.	[O.1], [O.2], [O.5]
Weeks 5-6	Summarise research on DNA storage and nanopores. Write project proposal.	[O.1], [O.2], [O.3], [O.4]
Weeks 7-8	Learn about existing models. Implement codon-code in Scrappie and basecall with Dorado.	[O.3], [O.4], [O.5]
Weeks 9-10	Test simplistic codon-code. Analyse performance. Determine theoretical performance.	[O.5], [O.6]
Weeks 11-12	Compare theoretical and practical performance. Write progress report.	[O.7], [O.8]
SWOTVAC and Exams	Collect papers to read over the Summer.	
Summer break – 20 November 2023 to 25 February 2024		
November	Read papers on coding for nanopore sequencing.	[O.2]
December	Install and learn to use Dorado.	[O.4]
January	Conduct preliminary investigation with Dorado	[O.4], [O.5], [O.7]
February	Complete experiment 1.	[O.7]
ENG4702 – 26 February 2024 to 26 May 2024		
Weeks 1-2	Research and formulate new metrics with which to evaluate codebook performance.	[O.9]
Weeks 3-4	Test codebooks against these new metrics.	[O.9]
Weeks 5-6	Create a large set of improved codebooks. Complete experiment 2.	[O.9], [O.10]
Weeks 7-8	Investigate classification problem with Dorado.	
Week 9	Investigate F5C consensus problem with Dorado	
Week 10	Review, discuss and finalise project with supervisor.	[O.11]
Week 11	Create project poster and video. Write final report.	[O.11]
Week 12	Finish final report.	[O.11]
SWOTVAC	Poster presentation	[O.11]

## 9.2.3 Comparison of timelines

The original timeline underestimated the difficulty in getting Dorado to work. There was an added step of creating synthetic POD5 files which took a significant amount of time to understand, and the Summer break was largely spent doing this. Towards the end of the break, a large number of trials were conducted and experiment 1 was finalised. In the original timeline, refined codebooks were required by the beginning of the second semester. However, finding these refined codebooks was not a simple problem, and this eventually grew into another standalone experiment which took the first six weeks of the final semester. The original timeline had gaps for unforeseen circumstances, but the extra time wasn't needed. It was offered to another researcher at Monash by assisting them in using Dorado and classifying their reads.

## 9.3 Reflection on Project

### ***Reflection on individual performance.***

There was a steep learning curve for ONT's software and Linux as a whole. This is summarised by one researcher, who wrote "Oxford Technology Nanopore communicates little about the precise characteristics of its devices and softwares" [24]. At times, I spent days trying to understand a single command with little success and became demotivated and unsure if it would ever work. However, I refused to believe that published software would be dysfunctional and endeavoured through this, eventually using many packages such as Scrappie, Dorado, Samtools, F5C, pod5, fast5-research and some others. I was proud of my work and glad to have seen it through.

### ***Reflection on project success.***

Due to uncertainty in the software involved, at times I was doubtful of the project's success. Achievements often came abruptly but eventually I was able to collect the data together and show it only weakly supported the initial hypothesis. At this point, I realised that I needed to widen the scope of my project, which I achieved by initially investigating other metrics besides popularity, and later by offering assistance to another researcher in this field. I was able to succeed in these endeavours, however it felt like three smaller successes rather than one significant success, which would have arisen if the experiment supported the hypothesis more strongly.

### ***Reflection on timeline.***

In the original timeline, I was to spend time over the Summer break working on the project. I had planned to do this concurrently with a demanding internship in Sydney, not estimating how difficult this would be. In the end, I spent less time than planned on the project over the Summer and had to spend a significant amount of time at the start of the second semester to catch up. After finishing the main component of the project, I had a spare three weeks and took on extra work to assist other researchers, an excerpt of which was included as experiment 3 in this report.

### ***Strategies for future improvement.***

When I encountered problems, I often took a long time before applying for help from others, which became a problem as I was completing this project individually. In future, I would ask for assistance earlier, drawing from my supervisor, other staff members and other students, where this is allowed. I would also be more organised with my note-taking, as I often placed important information in a large number of text files which I could not find later on. Instead, it would be useful to dotpoint directly into a draft report, so I could see all the information I had in one place and organise it effectively.

## 10 References

---

- [1] D. e. al, "Emerging Approaches to DNA Data Storage: Challenges and Prospects," *ACS Nano*, vol. 16, no. 11, November 2022.
- [2] "Oxford Nanopore Technologies," [Online]. Available: <https://nanoporetech.com/>. [Accessed 23 August 2023].
- [3] R. R. Wick, L. M. Judd and K. E. Holt, "Performance of neural network basecalling tools for Oxford Nanopore sequencing," *Genome Biology*, vol. 20, no. 1, pp. 129-138, 2019.
- [4] C. E. Shannon, "A Mathematical Theory of Communication," *The Bell System Technical Journal*, vol. 27, pp. 379-423, 623-656, July 1948.
- [5] D. Reinsel, J. Gantz and J. Rydning, "The Digitization of the World - From Edge to Core," International Data Corporation, Framingham, 2018.
- [6] Y. Erlich and D. Zielinski, "DNA Fountain enables a robust and efficient storage architecture," *Science*, vol. 355, no. 6328, pp. 950-954, February 2017.
- [7] DNA Data Storage Alliance, "An Introduction to DNA Data Storage," DNA Data Storage Alliance, 2021.
- [8] V. Rozite, "Data Centres and Data Transmission Networks," International Energy Agency, 11 July 2023. [Online]. Available: <https://www.iea.org/energy-system/buildings/data-centres-and-data-transmission-networks>. [Accessed 23 August 2023].
- [9] T. Kumar and S. Namasudra, "Chapter One - Introduction to DNA computing," in *Perspective of DNA Computing in Computer Science*, Elsevier, 2023, pp. 1-38.
- [10] L. M. Adleman, "Molecular computation of solutions to combinatorial problems.," *Science*, vol. 266, no. 5187, pp. 1021-1024, 1994.
- [11] J. Watada and R. Bakar, "DNA Computing and Its Applications," *ICIC*, vol. 2, no. 1, pp. 101-108, 2008.
- [12] L. Rittie and B. Perbal, "Enzymes used in molecular biology: a useful guide," *Journal of Cell Communication and Signaling*, vol. 2, no. 1, pp. 25-45, 2008.
- [13] *Base Pair*, National Human Genome Research Institute, 2024.
- [14] T. Fatima and A. Ebertz, "A Journey Through The History Of DNA Sequencing," Eurofins Genomics, [Online]. Available: <https://the-dna-universe.com/2020/11/02/a-journey-through-the-history-of-dna-sequencing/>. [Accessed 23 August 2023].
- [15] "Illumina," [Online]. Available: <https://sapac.illumina.com/>. [Accessed 24 08 2023].
- [16] A. Smith, M. Jain, L. Mulrone, D. Garalde and M. Akeson, "Reading canonical and modified nucleobases in 16S ribosomal RNA using nanopore native RNA sequencing," *PLoS*, vol. 14, no. 5, 2019.




- [17] R. N. Grass, R. Heckel, M. Puddu, D. Paunescu and W. J. Stark, "Robust Chemical Preservation of Digital Information on DNA in Silica with Error-Correcting Codes," *Angewandte Communications*, vol. 54, pp. 2552-2555, 2015.
- [18] L. Organick et al, "Random access in large-scale DNA data storage," *Nature Biotechnology*, vol. 36, no. 3, pp. 242-248, 2018.
- [19] A. Vidal, V. B. Wijekoon and E. Viterbo, "Error Bounds for Decoding Piecewise Constant Nanopore Signals in DNA Storage," *IEEE International Symposium on Information Theory (ISIT)*, pp. 358-363, 2023.
- [20] K. Subramanian, B. Payne, F. Feyertag and D. Alvarez-Ponce, "The Codon Statistics Database: A Database of Codon Usage Bias," *Molecular Biology and Evolution*, vol. 39, no. 8, 2022.
- [21] *Scrappie*, Oxford Nanopore Technologies, 2017.
- [22] "kmer\_models," Oxford Nanopore Technology, 2024.
- [23] *Dorado*, Oxford Nanopore Technologies, 2022.
- [24] C. Delahaye and J. Nicolas, "Sequencing DNA with nanopores: Troubles and biases," *PLoS ONE*, vol. 16, no. 10, 2021.
- [25] B. Young, T. Faris and L. Armogida, "Levenshtein distance as a measure of accuracy and precision in forensic PCR-MPS methods," *Forensic Science International: Genetics*, vol. 55, p. 102594, 2021.
- [26] H. Tanaka, *Editdistance*, 2013.
- [27] B. O.F., S. A.K., C. B.K and T. N.A, "Relative stability of AT and GC pairs in parallel DNA duplex formed by a natural sequence," *FEBS Letters*, vol. 322, no. 3, pp. 304-306, 1993.
- [28] A. Piovesan, M. Pelleri and F. Antonaros, "On the length, weight and GC content of the human genome," *BMC Research Notes*, vol. 12, no. 106, 2019.
- [29] M. K. S. M. K. e. a. Jain, "Nanopore sequencing and assembly of a human genome with ultra-long reads," *Nature Biotechnology*, vol. 36, no. 4, pp. 338-345, January 2018.
- [30] H. G. e. al., "Fast nanopore sequencing data analysis with SLOW5," *Nature Biotechnology*, vol. 40, pp. 1026-1029, 2022.
- [31] X.-Q. Li, "Variation, evolution, and correlation analysis of C+G content and genome or chromosome size in different kingdoms and phyla.," *PLoS One*, vol. 9, no. 2, p. e88339, February 2014.
- [32] "Transforming our world: the 2030 Agenda for Sustainable Development," United Nations, New York, 2015.
- [33] I. U. W. B. W. IEA, "Tracking SDG 7: The Energy Progress Report," World Bank, Washington DC, 2023.
- [34] "FAST5 Examples," Oxford Nanopore Technologies, 2017. [Online]. Available: [https://nanoporetech.github.io/fast5\\_research/examples.html](https://nanoporetech.github.io/fast5_research/examples.html).
- [35] "Docker ONT guide," Oxford Nanopore Technologies, 2020.
- [36] *FAST5 to POD5 converter*, Oxford Nanopore Technologies.

- [37] J. Henninger, "The 99 percent of the human genome," 1 October 2012. [Online]. Available: <https://sitn.hms.harvard.edu/flash/2012/issue127a/>. [Accessed 29 August 2023].
- [38] B. McBain, E. Viterbo and J. Saunderson, "Finite-State Semi-Markov Channels for Nanopore Sequencing," *IEEE International Symposium on Information Theory - Proceedings*, Vols. 2022-June, pp. 216-221, 2022.
- [39] S. Zhang, B. Wang, L. Wan and L. M. Li, "Estimating Phred scores of Illumina base calls by logistic regression and sparse modeling," *BMC Bioinformatics*, vol. 18, no. 1, p. 335, 2017.
- [40] "Sequence Alignment/Map Format Specification," The SAM/BAM Format Specification Working Group, 2023.
- [41] *POD5 File Format*, Oxford Nanopore Technologies, 2022.
- [42] B. McBain and E. Viterbo, "An Information Theoretic Approach to Nanopore Sequencing for DNA Storage (Unpublished)," 2023.

# 11 Appendices

## 11.1 Appendix A: Project Risk Assessment

The following is a printed summary of the risk management plan, as submitted on the Monash S.A.R.A.H platform.

52003	RISK DESCRIPTION	STATUS	TREND	CURRENT	RESIDUAL
	ENG4701_CL_2023_S2_PreetPatel_CodingAndInformationTheoryProblemsInDNAStorageWithNanoporeSequencing	Draft		Medium	Low
RISK TYPE					
1. Activity or Task Based Risk Assessment					
RISK OWNER		RISK IDENTIFIED ON		LAST REVIEWED ON	
PREET JIGNESH PATEL		23/08/2023			
RISK FACTOR(S)	EXISTING CONTROL(S)	CURRENT	PROPOSED CONTROL(S)	TREATMENT OWNER	DUE DATE
Sitting in a chair for long periods of time. This may cause circulatory, nerve or musculoskeletal stress due compression in the legs and lower back as a result of being seated for large periods of time, or having inadequate posture.	Control: Currently, the student stretches their back to avoid pain from sitting too long. Control Effectiveness:	Medium	In addition to stretching, the student will stand up and move around regularly, and work standing up wherever possible. If sitting for a long period of time is necessary, the student will use a donut pillow to alleviate stress on the lower back.	PREET JIGNESH PATEL	24/08/2023
Looking at a computer screen for long periods of time. This can have several detrimental impacts on the human eye. Some are temporary, such as eye strain and computer vision syndrome, while others are permanent/semi-permanent, such as myopia.	Control: Currently, the student uses their computer for long periods of time. Control Effectiveness:	Medium	The student will adopt a 20-20-20 rule, where they look at least 20 feet away for 20 seconds, every 20 minutes.	PREET JIGNESH PATEL	24/08/2023
Stress and burnout from workload. Completing a final year project on top of a full time workload and multiple jobs may cause burnout if not managed correctly. Some possible consequences of this are failure in meeting project outcomes, failure of unit(s), dismissal from work and depression.	Control: The student has organised their timetable and attempts to meet tasks on a week-to-week basis. Control Effectiveness:	Low	The student will avoid burnout and stress by creating a priority list of tasks and maintaining communication with supervisor and other teams. Social activities will be restricted to once per week.	PREET JIGNESH PATEL	24/08/2023

# 11.2 Appendix B: Risk Assessment Matrix

Table 26: Risk Assessment Matrix where (L) is low risk, (M) is medium risk, (S) is substantial risk, (H) is high risk and (E) is extreme risk.

Consequence and Likelihood	Insignificant	Minor	Serious	Disastrous	Catastrophic
Rare	L	L	L	L	M
Unlikely	L	L	M	M	S
Possible	L	M	M	S	H
Likely	L	M	S	H	E
Almost Certain	M	S	H	E	E

## 11.3 Appendix C: Risk Management Plan

### 11.3.1 Task abstraction

The tasks directly associated with this project are reading books and journal articles, conducting research, coming up with hypotheses, testing these using simulation software, analysing their performance using further software, and writing reports. This will largely be done on a computer, on paper, or on a whiteboard and either at Monash University Clayton Campus or at the student's home.

### 11.3.2 Risk analysis

These activities are deemed low-risk as they do not introduce any new or significant risks of physical or psychological harm to the student. Nonetheless, the OHS risks that do exist have been documented in S.A.R.A.H, which is Monash University's risk management platform. A printout of this can be found in Appendix A. A wider range of risks which may affect either the student or the project have been identified and collated in Table 27. The risk level is determined using a risk assessment matrix, which can be found in Appendix B.

*Table 27: Projects risks.*

Project Risk	Risk	Likelihood	Consequence	Risk level	Mitigation	Residual Risk
Legal action from Oxford Nanopore Technologies	If the quality scores found significantly contradict the published quality scores on the ONT website, the company may scrutinise the research or ask for it to be retracted.	Unlikely	Serious	M	Including a disclaimer that the research findings apply only to synthetic DNA, and cannot be compared to ONT's published benchmarks.  Clearly showing the data production and analysis process, so that it is repeatable and can be independently verified.	Risk is rare, and consequence is deemed minor, as the disclaimer provides legal footing to the student.
Dangerous DNA sequence is produced	If the DNA encoding scheme allows some particular base orders to be synthesised, this could create new/dangerous viruses.	Rare	Serious	L	The student will research tables of unallowed nucleotide sequences, and attempt to prevent their code ever generating them.	This is quite a rare risk, and with the mitigation mentioned is not expected to occur.

Sitting in a chair for long periods of time.	This may cause circulatory, nerve or musculoskeletal stress due compression in the legs and lower back as a result of being seated for large periods of time, or having inadequate posture.	Possible	Minor	M	Stretch back to avoid pain from sitting too long. Additionally, stand up and move around regularly, and work standing up wherever possible. If sitting for a long period of time is necessary, the student will use a donut pillow to alleviate stress on the lower back.	The same risk still exists, but has lower likelihood. The risk level is L.
Looking at a computer screen for long periods of time.	This can have several detrimental impacts on the human eye. Some are temporary, such as eye strain and computer vision syndrome, while others are permanent/semi-permanent, such as myopia.	Likely	Insignificant	M	The student will adopt a 20-20-20 rule, where they look at least 20 feet away for 20 seconds, every 20 minutes.	Again, the same risk still exists, but has lower likelihood. The risk level is L.
Stress and burnout from workload.	Completing a final year project on top of a full time workload and multiple jobs may cause burnout if not managed correctly. Some possible consequences of this are failure in meeting project outcomes, failure of unit(s), dismissal from work and depression.	Possible	Insignificant	L	The student will avoid burnout and stress by creating a priority list of tasks and maintaining communication with supervisor and other teams. Social activities will be restricted to once per week.	Again, the same risk still exists, but has lower likelihood. The risk level is L.
Similar competing research arises.	If another researcher publishes a paper that covers many of the same research areas, then the importance and credibility of this project may be impacted.	Rare	Serious	L	The student will keep a record of research as it is conducted, using the project journal to summarise weekly activities.	The risk is alleviated because keeping a record will allow this project paper to be published, regardless of the competing paper. The likelihood doesn't change.
Loss of work.	Crashing or malfunction of computers or loss of paper notes could lead to lost work, and introduce delays in the project.	Unlikely	Minor	L	The student will primarily keep notes on a word document, and only use paper for highlighting key information in papers, and for temporary calculations. This will be backed up on OneDrive continuously.	As the project notes are backed up on OneDrive, the risk of lost work is rare.

## 11.4 Appendix D: Sustainability Plan

### 11.4.1 Sustainable Development Goals

Engineering practices and projects have widespread and lasting impacts on both people and the planet. In any engineering project, it is important to consider the benefit and detriment that may be caused to society, the economy and the environment. To this end, the United Nations have developed and published *The 2030 Agenda for Sustainable Development* [32], which lists 17 Sustainable Development Goals (SDGs) and 169 targets underlying these goals. These aim to end poverty and hunger, build peaceful and just societies, protect human rights, promote gender equality, protect the planet and its resources and create sustainable economic growth and prosperity for all. Engineering can help achieve these goals in many ways, both by developing innovative technology and solutions, and by improving existing solutions.

While multiple SDGs align with the aims of this project and DNA storage technology more broadly, the one which resonates the strongest is Goal 7.

*Goal 7. Ensure access to affordable, reliable, sustainable and modern energy for all.* [32]

The most relevant target under this goal is Target 7.3.

*Target 7.3. By 2030, double the global rate of improvement in energy efficiency.* [32]

We are currently midway through the UN's 15-year plan, yet Goal 7 is not on track to be reached by 2030 according to *Tracking SDG7: The Energy Progress Report* [33]. The key indicator for Target 7.3 is energy intensity, which is the ratio of energy consumed to GDP. This has improved at a rate of 1.8% per year, which falls short of the 2.6% benchmark [33]. To achieve this target, energy-exhaustive technologies must be replaced with energy-efficient ones. Furthermore, the handover process from old technology to new should not use arbitrarily high amounts of energy, as this would counter the purpose of the handover in the first place.

#### 11.4.2 DNA Storage and Sustainability

As discussed in Section 5.2, one of the key benefits of using DNA to store digital data is its compactness and low energy consumption. The compactness of DNA storage would reduce the need for large data centres which require significant amounts of energy to run. Much less energy would be needed to achieve the same or higher levels of data storage, thereby contributing to improved energy intensity. The exponential growth in stored data [5] means that making this process more energy efficient could have significant contributions to improving global energy intensity and achieving Target 7.3.

A significant drawback of DNA storage is that a large number of laboratory materials are consumed during both the sequencing and synthesis stages of DNA storage with nanopore sequencing. For example, one of ONT's products is the Flongle, which costs 90USD and can sequence an estimated 2.8 billion bases [2], which is less data than an average movie. Flongles cannot be reused and end up as waste, amounting to high energy and resource costs for nanopore sequencing. To compare, memory card readers can be used thousands of times with little degradation.

While the energy cost of producing nanopore sequencers cannot be recovered, ONT offers a recycling program for depleted sequencers [2]. Sequencers are sent back to ONT who can recycle some of the electrical components, thereby reducing environmental impact. More avenues of recycling such as this are necessary before DNA storage can become widespread.

The role of this project in achieving Goal 7 is in improving understanding of the performance of ONT's basecallers on synthetic DNA. By doing so, researchers could develop error-correcting codebooks targeted at these basecallers which can achieve higher information rates than existing codebooks. More robust codebooks would allow cheaper synthesis techniques to be used without comprising data integrity, overall increasing the competitiveness of DNA storage technology.



## 11.5 Appendix E: Generative AI Statement

**Email**

[ppat0019@student.monash.edu](mailto:ppat0019@student.monash.edu)

**Name**

Preet Patel

**Are you part of a team?**

No

**Campus**

Clayton

**Host**

Department Electrical and Computer Systems Engineering

**Supervisor**

Emanuele Viterbo

**This project has been conducted using AI tools**

In this project, there will be no use of generative artificial intelligence (AI). All content in relation to the assessment task has been produced by the authors.

**The use of Generative AI has been discussed with and approved by my academic supervisor.**

No

## 11.6 Appendix F: DNA File Formats

### 11.6.1 FASTA

The FASTA file format is commonly used for storing a sequence of bases in a readable format. The file extensions used are .fa or .fasta. The following is an example of a FASTA file.

> Sequence 1

```
TGTTCTCATCAGCTATTAGGTTGTAGCCAGATTTTCGAGCTTTTGATCTCAAGATCAGCTGTGCATTTCAAAAACTTACATATGCCT
GGGCTATATACTGGACTTCAGAATCTGAGTGTACAAGTCTTGGTGGGCATTTGCATTTCTAGAACCCCTGAATAAGAAGTGCCCCC
AAAATATATATTGAATGGCTTAGGGGGATTCTATATCGGAT
```

> Sequence 2

```
CTTTTCCATGCACACACAGTGCAAGCTGTTGAATCCTGCCATTCTGGGGGTACCAGCAAGAGCGCCGTGCTGATGGGCGAGAA
```

A FASTA file can be created directly in an editor like VSCode, or can be generated in python. If the description desc and transmission message tx\_msg are specified, this can be created using the code

```
with open("message.fa", "w") as fasta_file:
    fasta_file.write(f">{description}\n")
    fasta_file.write(tx_msg + "\n")
```

### 11.6.2 FAST5

A FAST5 file holds the squiggle data outputted by a nanopore, as shown in Figure 7. The easiest method to generate a FAST5 file is using the python fast5\_research package and following the instructions found in [34]. The Git repo associated with this project contains some example code.

Note that there are two types of FAST5 files – *single-read* and *multi-read*. Only the latter will be accepted by Dorado.

### 11.6.3 POD5

A POD5 file also holds squiggle data, and replaces the FAST5 format. Hence, while Scrappie uses FAST5, the newer Dorado prefers POD5. The Git repo demonstrates one method for generating POD5 files using the pod5 package, which can be installed from PyPi. Other packages needed include uuid and pytz.

The code takes normalised squiggles, generates a pod5 read and adds this read to a new file 'new.pod5'. The writing step will fail if the file 'new.pod5' already exists.

## 11.7 Appendix G: Guide to Scrappie

### 11.7.1 What is Scrappie?

Scrappie is a software package provided by ONT which provides squiggle generation and basecalling facilities for R9.4 and R10 nanopores. Since R10 was released in 2019, there have been two major revisions – R10.3 in 2020 and R10.4 in 2022. These are not included in Scrappie as its final release was in 2019. The repository was archived in 2022, and further information can be found on its Git page [21]. Scrappie runs on Linux and can be called in command line or directly from python.

The following description of Scrappie is found using the `scrappie help` command, explaining its at-times confusing design choices.

*“This project began life as a proof (bet) that a base caller could be written in a low level language in under 8 hours. Some of the poor and just plain odd design decisions, along with the lack of documentation, are a result of its inception. In keeping with ONT’s fish naming policy, the project was originally called Crappie.”*

### 11.7.2 Scrappie installation

There are several ways to run Scrappie.

A Docker image containing Scrappie can be created by following the instructions in [35]. Rather than directly using the run or mount commands found on the website, it is advisable to also give an appropriate name using the following run command.

```
docker run -it --name scrappie -v <path-to-data-folder>:/data scrappie
```

The ‘-v’ tag specifies that the contents of ‘<path-to-data-folder>’ on the user’s computer will be shared with the ‘data’ folder in the docker image.

To run Scrappie in python, it must be installed using the command

```
pip install scrappie
```

When importing in python, the alternative alias `scrappy` must be used, as in

```
import scrappy
```

### 11.7.3 Scrappie squiggle generation

Scrappie generates squiggle statistics for a given DNA sequence in the text format given in Figure 31.

# Sequence 1				
pos	base	current	sd	dwell
0	T	0.419587	0.345852	0.713681
1	G	1.023567	0.318915	0.794324
2	T	0.079655	0.306535	1.033633
3	T	-0.728703	0.242368	2.603098
4	C	1.033638	0.181615	4.248816
5	T	0.463943	0.149272	4.893713
6	C	1.642555	0.178201	6.842926
7	A	0.567149	0.197912	7.023503
8	T	1.148610	0.125626	7.985809

Figure 31: Scrappie output. Source: Scrappie [21].

From these statistics, a squiggle can be generated using the process described in Section 0.

To generate these statistics, `scrappie squiggle` must be called through command line or in python. An example command line prompt is

```
scrappie squiggle --model squiggle_r94 --output stats.txt sequence.fa
```

For R10, the model name is “squiggle\_r10”. The *sequence.fa* input file should be in FASTA format, as specified in Appendix F. One of the options is *--rescale* or *--no-rescale*. As stated in the documentation, “the output of the squiggle prediction network is scaled into natural coordinates”, meaning that the natural logarithm of the sd and dwell are used internally within Scrappie. When calling *scrappie squiggle*, the default option is *--rescale*, which means the sd and dwell are rescaled to comparable, real-world values, as needed in the process described in Section 0.

An example use of the squiggle generation functionality in python is

```
test = scrappy.sequence_to_squiggle(sequence, rescale=True)
data = test.data(as_numpy=True)[:,[1,2,0]]
```

*rescale=True* is needed because the default option in python is *--no-rescale*. The first command returns a *Scrappy matrix*, however the order of the columns is sd, dwell, mean, which is different to the command-line output as seen in Figure 31. The second line converts the data to a numpy array and reorders the columns so it matches the command line output.

#### 11.7.4 Scrappie basecalling

Scrappie can basecall a raw signal in FAST5 format. An example command line prompt is

```
scrappie raw --format fasta --model raw_r94 --output calls.fa squiggle.fast5
```

The output format can be *fasta* or *sam*. There are several basecalling models, listed on the Git page. To avoid the step of creating a FAST5 file when simulating synthetic reads, the python package can be used.

An example use of basecalling functionality in python is

```
rx_msg = scrappy.basecall_raw(squiggle)[0]
```

Here, *squiggle* is a numpy array of type int16, containing the digitised current levels. The *basecall\_raw* function returns the basecalled sequence as a string, as well as some metadata.

## 11.8 Appendix H: Guide to Dorado

### 11.8.1 What is Dorado?

Dorado is a software package provided by ONT which provides basecalling, alignment, barcode classification and several other facilities for R9.4.1 and R10.4.1 nanopores. It is more powerful and versatile than previous basecallers, including Scrappie. The package is continually being updated and re-released, with the latest being on 21<sup>st</sup> May 2024. Dorado runs on all major operating systems, however it requires a GPU-enabled device due to its intensive processes. There is currently no functionality to use Dorado in python directly, however this can be done indirectly using the *os* package, for example with

```
os.system(dorado_command)
```

### 11.8.2 Dorado basecalling

The following is an elaboration of Section 6.2.9, which introduced Dorado basecalling.

The authors of Dorado recommend squiggles in POD5 format, and the steps to create such a file are included in Appendix F. An example command is

```
dorado basecaller --emit-fa dna_r9.4.1_e8_sup@v3.6 squiggle.pod5 > calls.fa
```

In this command, the output format chosen is FASTA, which is the simplest format. This overrides the default binary option BAM and its readable counterpart SAM. Dorado can also accept squiggles in FAST5 format, although these must be of *multi-read* format. The authors of Dorado recommend POD5 input, and the files can be converted online [36].

A range of basecalling models besides *dna\_r9.4.1\_e8\_sup@v3.6* can be chosen, with a full list found on the Dorado git page [23]. In the latest release, v5 models are available, offering improved accuracy. The tag *sup* is short for ‘super accuracy’, which is the most accurate model but also the most computationally demanding [2]. *hac* and *fast* are also offered, which stand for ‘high accuracy’ and ‘fast’ respectively. R10.4 models were previously offered at 400bps and 260bps, although the latter has been discontinued.

### 11.8.3 Dorado alignment

Alignment refers to aligning a basecalled read to a reference sequence. Dorado uses Minimap2 to align reads, and can either align as a separate step or together with basecalling. An example command line prompt is

```
dorado basecaller -v dna_r9.4.1_e8_fast@v3.4 multi.pod5 > calls.bam --reference ref.fa
```

FASTA is not designed for alignment information, so we use the default BAM output. Each read in *multi.pod5* is basecalled and then aligned to the reference sequence *ref.fa*.

To view these aligned sequences, we must use samtools, which can be installed by downloading from <http://www.htslib.org/download/> and running the commands

```
$ cd samtools-1.18
$ ./configure --without-curses --disable-bz2 --disable-lzma && make
```

An alternative installation method is running the command

```
sudo apt install samtools
```

We first need to sort and index the reads using the commands

```
samtools sort calls.bam -o sorted.bam
samtools index sorted.bam
```

Finally, we can view the aligned reads using the command

```
samtools tview sorted.bam ref.fa
```

```

1      11      21      31      41
AAAAAAGAGTACACCATGCAGATCGAGGTGATCC*A*GTACAT*
|
.K.....-.....
.....C.*.*.....G.*.**.*
.....*****.....*.*.**.*
.....*.....*.....G.*.**.*
.....*.....*.*.**.*
.....*.*.*A.G...

```

The first line is the reference genome. The second line is presumably the consensus sequence. The lines below this are the individual basecalled reads, arranged by start index. A dot represents a match. An asterisk in the reference sequence represents an insertion. An asterisk in the basecalled sequence represents a deletion. A letter in the basecalled sequence represents an insertion or a substitution.

## 11.9 Appendix I: Guide to F5C

### 11.9.1 What is F5C?

F5C is a package that gives information about which parts of the squiggle correspond to which bases. In particular, it assigns *events* to *k*-mers, and has an *eventalign* command which we use. As this is a stand-alone guide, some hyperlinks are included directly instead of being added to the main set of references.

An *event* is a sequence of consecutive current measurements during which the current is relatively constant. (The F5C package has a program which claims to implement event detection. So, I believe event detection is done at this stage and not earlier, but I cannot find confirmation on this). We consider an example sequence.

- The example sequence has somewhere over 451 events.
- Each event corresponds to a minimum of 3 and an average of 5.2 current measurements (samples).

An example use of the command is

```
f5c eventalign --print-read-names --signal-index --collapse-events --scale-events
--samples -b sorted.bam -g reference.fa -r reverse_1.fa --pore r10 > aligned.tsv'
```

The meaning of some of the options and terminology are:

- **Eventalign:** This means that each event is matched up with a 9mer in the reference sequence.
- **Collapse-events:** Some 9mers will have multiple events associated with them. *Collapsed* means that events corresponding to a single 9mer are aggregated into a single row of the F5C output file.
- **Scale-events:** Each read is individually shifted and scaled so that it matches the model mean and standard deviation.

F5C will first detect the events and store them for later use. Each event has the following information associated with it:

- Event index
- Event level mean – average of current measurements (scaled).
- Event standard deviation – standard deviation of current measurements (scaled).
- Start index – index of first sample corresponding to that event (inclusive).
- End index – index of last sample corresponding to that event (exclusive).
- Set of current measurements (scaled samples) corresponding to that event, obtained using *--scaled* option.
- Duration of event (a multiple of 1/5000 seconds for R10.4)

The F5C package also has R9 and R10 *k*-mer model data stored in `f5c/src/model.h`. For R10, this model data is essentially a dictionary where each row corresponds to a single 9-mer, and gives the mean current level observed for that 9-mer, as well as standard deviation. This model is distinct from the ONT model [see: [https://github.com/nanoporetech/kmer\\_models/tree/master/dna\\_r10.4.1\\_e8.2\\_400bps](https://github.com/nanoporetech/kmer_models/tree/master/dna_r10.4.1_e8.2_400bps)]. It seems that the F5C authors have used the ONT model as a base, and then conducted further training on top of that using data from a PromethION flow cell [<https://hasindu2008.github.io/f5c/docs/r10train>].

The current levels in any given read are not comparable to the current levels found in the model, due to variation in nanopores. F5C will by default scale and shift the model current levels to match the read current levels. However, by passing the `--scale-events` tag into the `eventalign` command, F5C will instead scale and shift the read current levels to match the model current levels. This is preferred because it means we can easily compare measurements from different reads, since they have all been scaled to match the model.

- For our example read, we have  $shift = 10.40$ ,  $scale = 0.95$ . This is printed by F5C if we include the `--print-scaling=yes` tag.
- $scale$  and  $shift$  are chosen by F5C, supposedly to minimise squared error between read and model.
- A  $var$  is also printed individually for each read. This is used to calculate the standardised level, but we will later find a consensus without worrying about the  $var$  or standardised level, so no need to worry about this. The formula for  $var$  seems to be the following, found by diving into the source code.

$$var = \sqrt{\text{average} \left( \left[ \frac{current\_level\_mean - (scale \times model\_mean + shift)}{model\_stdv} \right]^2 \right)}$$

- $var$  is a measure of how much the scaled currents differ from the model, after an optimal scaling is applied. For our example read, the  $var$  is 0.66. The  $var$  is always below 1.

The relationship between raw and scaled current levels is the following: [see: [https://hasindu2008.github.io/f5c/docs/output#:~:text=\(event\\_level\\_mean%2Dscaling.shift\)/scaling.scale](https://hasindu2008.github.io/f5c/docs/output#:~:text=(event_level_mean%2Dscaling.shift)/scaling.scale)]

$$scaled\_current = \frac{raw\_current - shift}{scale}$$

- We observe that the  $shift$  is always positive ( $\approx 10$ ) and the  $scale$  is always less than 1.

F5C uses the ABEA algorithm to align the events to a reference sequence. To do this, it requires several inputs:

- Reference sequence.
- FAST5 file of read.
- Basecalled read in BAM format.
- Basecalled read in FASTA format.

Often, multiple events correspond to a single 9-mer, and less frequently, a single event corresponds to multiple 9-mers. We use the `--collapse-events` tag to combine multiple events when they correspond to a single 9-mer. The latter case is evident in the output data when we notice a 9-mer is skipped.

F5C outputs a large number of columns. The important ones (with `--scale-events` tag) are:

- `position`: index of 9-mer within reference sequence.
- `k-mer`
- `read index` or `read name`
- `event_level_mean`: average of scaled current measurements.
- `event_stdv`: std\* of current measurements
- `event length`: multiple of 1/5000 seconds.
- `model_mean`
- `model_stdv`
- `standardised_level`

The standardised level is calculated as

$$standardized\_level = (event\_level\_mean - model\_mean) / (model\_stdv * \sqrt{var})$$



A low var occurs when the scaled currents do not closely match the model currents. This gives a larger standardized level.

To find the consensus squiggle we will take the weighted average of the scaled event\_level\_means.

- Scaled: We use the scaled means as these have removed the individual characteristics of each nanopore, and are comparable.
- Weighted: Reads that linger on a 9-mer for longer have more samples of that 9-mer, and we expect them to be more reliable. Duration-weighted will give these reads a greater weight.

Suppose we have  $n$  reads. Suppose the scaled event\_level\_mean for the  $i^{th}$  read and  $j^{th}$  9-mer is  $E_{ij}$ . Suppose this persists for a duration of  $T_{ij}$  and has event standard deviation  $\sigma_{ij}$ . Then the consensus mean for this 9-mer is calculated as

$$C_j = \frac{\sum_{i=0}^n E_{ij} \cdot T_{ij} / \sigma_{ij}}{\sum_{i=0}^n T_{ij} / \sigma_{ij}}$$

The Git repo contains the files used to conduct this research. In particular, the *demo.ipynb* file provides an interactive explanation of the processes described in this section.