University of Bergen
Department of informatics

Detecting inosine in nanopore sequencing data with machine learning

Author: Thomas Stautland
Supervisor: Eivind Valen
Advisors: Adnan Niazi

UNIVERSITETET I BERGEN
Det matematisk-naturvitenskapelige fakultet

August, 2021

Abstract

Detecting modifications in DNA has been a long-standing challenge in understanding the workings of the genome, particularly with regards to regulatory function. The currently most widely used sequencing technology, NGS, offers protocols to tackle these challenges but these are modification specific and involve convoluting preparation steps. As an alternative, nanopore sequencing offers the direct observation of such modifications. Though inosine has been demonstrated to be distinguishable from adenine in poly(A) RNA using nanopore sequencing, no framework has been proposed for the general detection of inosine presence in nanopore sequence data. In this thesis, I propose a test-based approach to use out-of-the-box classifiers to distinguish between sequences containing inosine and sequences that don't based on features present in nanopore sequencing data. The proposed model achieves a high accuracy on this classification task, providing avenues for further development of a self-contained inosine detector, as well as further exploration of the same approach to other modifications.

Acknowledgements

I would like to extend my sincerest gratitude to the supervisor for my master work, Eivind Valen, for introducing this project to me as well as for availability and feedback over the course of the project. Furthermore, I'm greatly appreciative for all the technical assistance and teachings I have recieved from Adnan Niazi, from the small beginnings of the project right up until the project's deadline. During my work on this project, I have greatly enjoyed and benefitted from being a part of the Valen group, a stimulating and constructive environment. For that, I would like to thank all the members of the group.

`Thomas Stauland`
12 August, 2021

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Biology is the study of life and living organisms. The information required to sustain life in an organism is stored in its genetic material. Genetic material takes the form of sequences of DNA and RNA, and is present in large quantities in all living organisms. Processing and extracting information from large datasets of biological relevance is the focus of bioinformatics. One particular focus of bioinformatics is sequencing and the subsequent analysis of biological sequences.

Analysis of biological sequences involves identifying the building blocks of various types of genetic material — be that DNA, RNA, or peptides. The process of identifying the contents of biological sequences is referred to as sequencing. This obtained sequence can then be analysed further to infer the behavioral or functional information carried by it.

One challenge within the domain of sequence analysis that has been the subject of extensive study, is the identification of modified nucleotides. These nucleotides have been chemically altered to make them different from the four standard (or canonical) bases — A, T, C, and G — of which DNA is composed. Such modifications can be indicative of functional relevance, or otherwise be of interest as markers in an experimental or clinical setting. In DNA, more than 17 such base modifications are known to exist [1]. Specialized sequencing techniques and analysis methods are needed to detect these modifications. This work uses a novel single-molecule sequencing technology developed by Oxford Nanopore Technologies (ONT) to identify modifications occurring within a DNA sequence. Specifically, the modification we aim to identify is the occurrence of a non-canonical base — inosine. In this thesis, I will detail the various approaches towards this end and reflect on the respective levels of success, or lack thereof.

In the following sections, I will motivate the rationale for the project and the biological relevance of modification detection before describing the technological framework used to produce data for analysis. In the "Experiment" chapters, I will then move on to describing the specific analyses carried out in this study and their results. Finally, I will offer concluding remarks and reflect on the experimental designs, the findings resulting from these and the potential avenues for further development.

1.1 Rationale

Before the advent of nanopore sequencing, base modification detection was done with short-read next-generation sequencing (NGS) approaches such as Illumina sequencing. However, Illumina sequencing cannot detect modifications directly. This is because Illumina sequencing requires PCR amplification of the original material which cannot conserve modifications present in the sequence. Furthermore, the synthesis step of sequencing-by-synthesis used in Illumina sequencing, generally does not allow for the selective addition of modified nucleotides where modification is present in the template.

To address these challenges with NGS, cumbersome workarounds were developed. For example, in bisulfite sequencing for detecting 5-methylcytosine (5mC) the DNA is treated with sodium bisulfite in such a way that modified cytosines remain intact in the sample, while unmodified cytosines are converted to uracil. When the sample is amplified and sequenced one can discriminate between modified and unmodified cytosines since all remaining cytosines in the template represent modified cytosines. However, in the scope of general modification detection, the primary shortcoming of this approach is evident as it is limited to detection of cytosine methylation. Though certainly useful — as 5-methylcytosine (5-mC) is one of the most commonly occurring and widely studied DNA base modifications — this method's specificity becomes a hurdle when considering base modifications in general.

In this respect, nanopore sequencing offers a great advantage. As the sequencing protocol does not rely on amplification and sequencing-by-synthesis, the molecule provided as input is directly sequenced. We therefore say that nanopore has the ability to sequence native molecules. The output of a nanopore sequencer is in the form of a series of measured current signals and the relative changes in this current as a DNA molecule translocates through the pore. These changes directly respond to the DNA as the current impedance is dependent on the mass of nucleotides passing through the pore. Modifications present in the sequence therefore directly influence data produced from a sequencing run. In analyzing data, modifications can then potentially be detected provided their impedance to the electrical current is sufficiently distinct from that of non-modified bases. The underlying rationale for this thesis follows directly from the direct native molecule sequencing feature offered by nanopore sequencing.

1.2 Objective

Based on the nanopore sequencing of canonical bases and the modified inosine base incorporated into synthetic DNA sequences, the aim of this work is to develop a method that can distinguish a sequence containing inosine from a sequence which does not.

Numerous studies have demonstrated the potential for using nanopore sequencing for detecting nucleotide modifications [2, 3]. We set out to further realize the modification-detection capabilities of this continually developing sequencing technology.

Specifically we want to investigate:

1. Is there a quantifiable difference between inosine (I) and the canonical bases (A, T, C, G) in the measurements resulting from sequencing with nanopore.
2. If there is such a difference, can distributions describing the distinct profiles of these five nucleotides (i.e., A, T, C, G, and I) be used to train an inosine-specific modification classifier. Such a classifier should be able to identify sequences in which an inosine is incorporated.

Chapter 2

Background

2.1 Nanopore sequencing

Nanopore sequencing is a novel technique enabling the direct sequencing of a native RNA or DNA molecule. In sequencing a native molecule using nanopore, a series of signal data is produced. Signal data is a direct representation of the sequence of the given molecule. This differs from previous generations of sequencing technology which rely on clonal amplification of a sequence of interest; in other words, an indirect representation of sequence data. In addition, the length of reads is extended from what was possible in previous generations. Through Illumina sequencing, which is the most widely used NGS technology, reads were 150 to 300 base pairs long. Nanopore sequencing can yield reads ranging from 500 bp to 2.3 Mb [4]. The actual resulting read length is highly dependent on sample and library preparation, but generally, the increased length means that nanopore sequencing gains an advantage when it comes to handling repeat regions. Based on these improvements, we say that nanopore sequencing belongs to the third generation of sequencing technology. While nanopore can process both DNA and RNA sequences, in this thesis we will be working exclusively with DNA sequences.

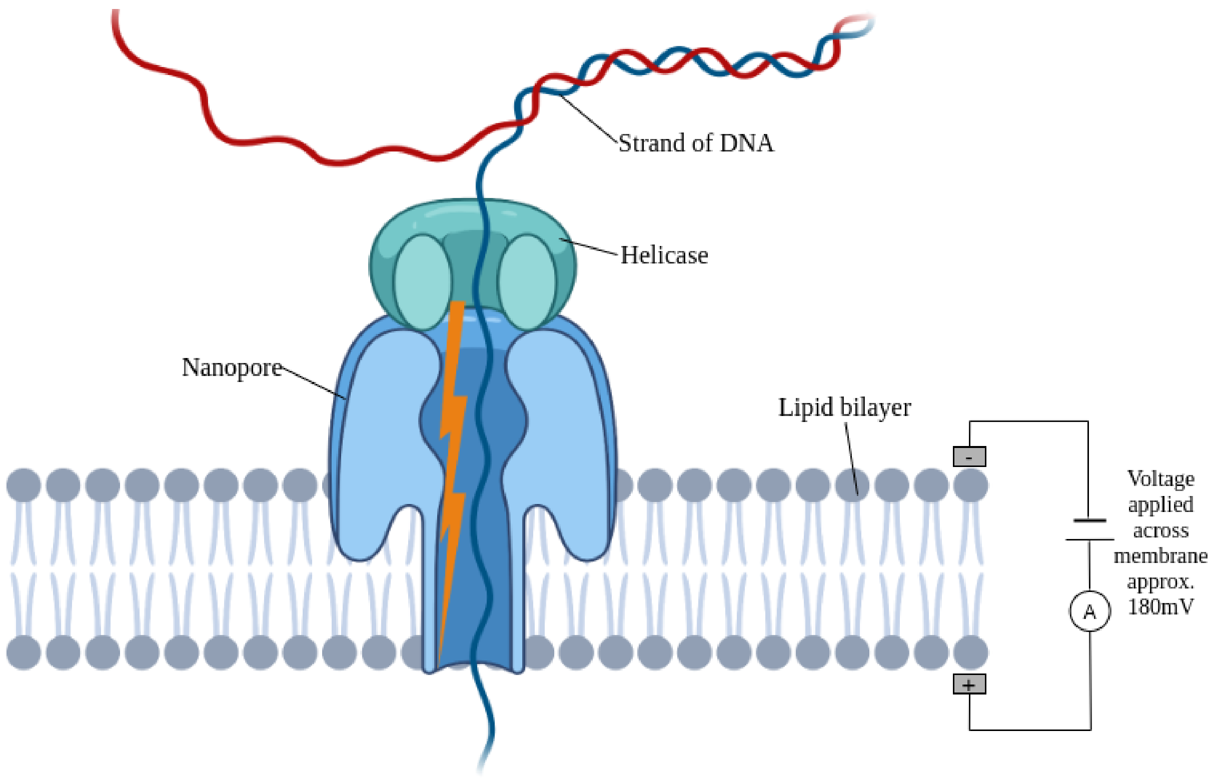


Figure 2.1: Illustration of nanopore sequencing. The double stranded DNA is unwinded by the enzyme sitting on top of the nanopore itself, and a single strand (the blue strand) passes through the nanopore. The lightning represents the current which also passes through the pore. As the current and the single strand of DNA passes through the pore simultaneously, the strand obstructs the current. This causes a change in the intensity of the current measured flowing through the pore, or a perturbation.

Credit: Illustration made with BioRender `https://biorender.com/`

To sequence DNA through a nanopore, special adapters are first ligated to the DNA to be sequenced. These adapters carry a motor protein that helps feed the molecule through the nanopore at a controlled speed. Next, the DNA with ligated adapters is fed into a flow cell that contains protein nanopores suspended in a lipid membrane across which a potential difference is applied. There is an ionic buffer on both sides of the membrane. The applied potential difference causes the ions to flow from one side to the other, resulting in a current. If anything blocks the pore, the current is perturbed; the larger the blockage, the larger the perturbation in the pore current, and vice versa (2.1). In this way, anything that passes through the pore creates a signature pattern in the current that can later be used to decode its identity.

2.1.1 Basecalling

As the DNA propagates through the nanopore, the current at any given time is impeded by the presence of five nucleotides (5-mer) occupying the pore; a given nucleotide is always read in the context of its surrounding nucleotides. So if we consider a construct containing only canonical bases, then 4^5 or 1024 unique combinations of nucleotides exist which can occupy the pore at any given time. Basecalling involves using the knowledge about the characteristic current signatures of these 1024 5-mers to predict which nucleotide sequence went through the pore. This is the working principle behind the use of nanopores for decoding DNA sequences [5].

Another metric we consider is the impact of the presence of a given nucleotide on the speed of a molecule traveling through the pore. We refer to the translocation rate as *dwell time* — the time a nucleotide spends in the nanopore. Because of the stochastic nature of motor protein and sequencing chemistry, DNA molecules have a varying translocation rate, and so this measure can sometimes be inconsistent. Nevertheless, it can be used — in addition to the current intensity — for deducing the bases in a k-mer. Trends in dwell time have also been proposed as an indicator for detecting sequence modifications such as 2'-O methylation [2]. Furthermore, this feature has been used to identify pseudouridine in RNA [6]. Together, the current intensity and the dwell time can serve as features with which the base identity can be inferred. Furthermore, information about one given nucleotide is gathered at all five possible positions it can occupy within the pore. The variance of this current signal over time is called the *squiggle*. These squiggles are stored as an array of numbers in FAST5 files. The task of identifying bases, or translating the squiggle into a sequence of bases, is referred to as *basecalling*. Tools that carry out basecalling are called basecallers. The basecaller used throughout my experiments is the ONT-provided tool for basecalling called *Guppy*.

Carrying out the procedure of basecalling is resource intensive and must meet several analytical challenges. The first challenge is to establish a correspondence between the continuous flow of current signal measurements and the discrete nucleotide sequence going through the pore. Generally, the approach to this is a procedure of segmentation. Old basecallers such as Albacore, used to provide a segmentation from raw current into what we call *events* prior to basecalling. This entailed looking for considerable changes in the current levels and using these to detect the eventuality of a new base entering the pore, and segment accordingly.

However, newer basecallers such as Guppy, have moved to a Connectionist Temporal Classification approach. This approach lets us classify a variable number of bases for a given window of signal measurements.

A consequence arising from the problem of segmenting current signals into events is also at the core of one of nanopore's shortcomings. This refers to basecalling of sequences that consist of a single nucleotide repeat — or *homopolymers*. In terms of events, homopolymers are described by a series of identical events following one another in succession. Since segmentation is based on changes to the current, these signals cannot be segmented in the same way. Moreover, the speed at which the construct is translocated through the nanopore is not constant, so this cannot be used to infer the number of events accurately. A common issue we can observe in basecalled nanopore data is therefore that such homopolymer segments exhibiting variable lengths from one read to another.

Another challenge of basecalling, is the task of actually labelling events with a sequence of bases. Various probabilistic frameworks can be used to tackle this problem, but Recurrent Neural Networks (RNNs) have been the most common approach. The model is trained on observed measurements of all possible k-mers. New observations that are being basecalled are then fed through the network and labelled according to the output. Finally, the basecalled sequence is produced by the merging of all labelled events. In the case of Guppy, we can view the results of this procedure in a .fastq or a .fast5 file.

2.2 Fast5 file format

The standard format for storing biological sequences that have been sequenced with a quality score is the FASTQ format. It is an uncomplicated format that describes a sequence in four lines: the first line is an identifier, the second line is the sequence itself, the third is a separator, and the fourth is the basecall quality. FAST5 format, on the other hand, is more complex. It is a type of hierarchical data format (HDF5), which is a flexible format for storing a set of associated data objects. A .fast5 file comes in two variations: a raw file and a basecalled file.

Whereas accessing information in a FASTQ format boils down to parsing lines of text, the FAST5 format consists of data objects contained within a set of folders. To navigate this

FAST5 format practically, some additional software or packages are required. With the help of these, the hierarchical structure of the file can then be navigated by directly querying the folders and the specific bits of information contained within.

In addition to metadata relating to the sequencing process, the raw file consists of a single dataset — the raw current signal measurements. After basecalling the raw file, another "Analyses" folder is attached to each file. This contains information pertaining to the segmentation of the signal, as well the sequence itself. There is also additional information of the squiggle and the transitions between events within this folder (2.2).

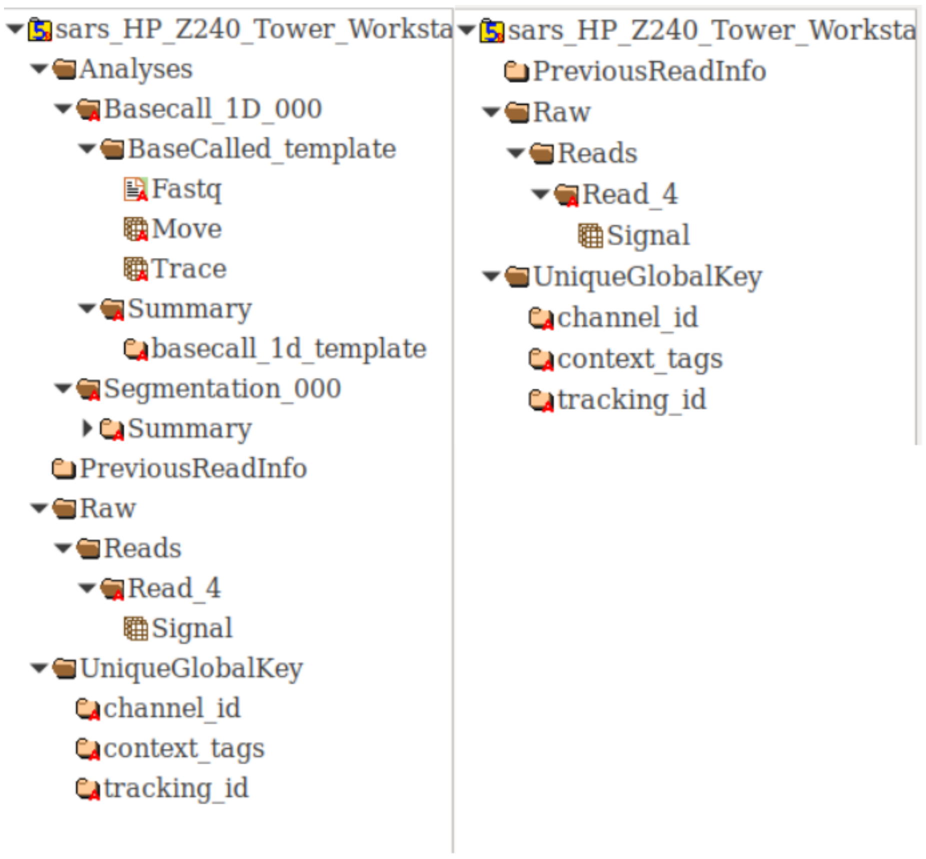


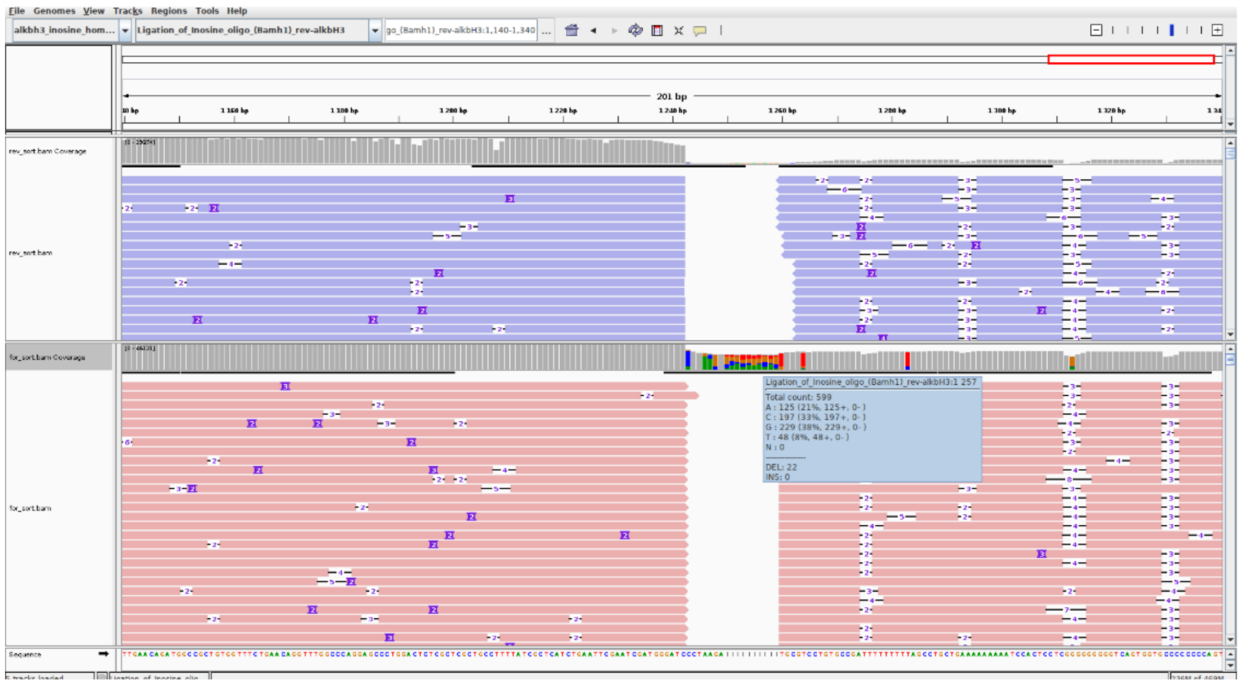
Figure 2.2: Side-by-side view of the structure of a basecalled and raw .fast5 file. The right side shows the basecalled file with an additional Analyses folder. Inside it, the BaseCalled_template folder holds information relating to the sequence of bases, how it corresponds to transitions of events, and the series of current signal measurements. On the left side, we see the raw file.

2.3 Alignments

Following basecalling, the next analytical step is commonly to produce alignments. This is the process of matching the basecalled sequence to a known reference. For all synthetic constructs we sequence in this study, we always have a reference dictating what we expect the sequenced output to be. The alignment used in all analyses carried out here were done with minimap2 [7].

Minimap2 uses a seed-chain-align method. It indexes the reference using minimizers — a set of representative substrings which are stored in a hash table and can be used for efficient string matching and extension. These minimizers are used as queries to anchor the sequence to the reference [8]. Sets of collinear anchors are then identified, and a dynamic programming approach is taken to extend the ends of chains to fill in the gaps between them. This produces a sequence based alignment, giving an indication of what the high-level information of the sequence looks like. Alignments can then be viewed in IGV as shown below (2.3).

If needed, we can also produce an alignment including the low-level information like raw current signal measurements. For this, we use Nanopolish's eventalign function [9]. Instead of aligning the sequence itself, eventalign aligns the events of the basecalled data to a reference. This is useful when subsequent analyses require insight into the raw data underlying the sequence itself. It is often used in investigating the deviations from model values in current signal measurements arising due to sequence modifications.


File Genomes View Tracks Regions Tools Help
alkbh3_inosine_hom...
Ligation_of_Inosine_oligo_(Bamh1)_rev-alkbH3
go_(Bamh1)_rev-alkbH3:1,140-1,340
201 bp
rev_sort.bam Coverage
rev_sort.bam
for_sort.bam Coverage
for_sort.bam
Sequence

```
Ligation_of_Inosine_oligo_(Bamh1)_rev-alkbH3:1 257
Total count: 599
A : 125 (21%, 125+, 0- )
C : 197 (33%, 197+, 0- )
G : 229 (38%, 229+, 0- )
T : 48 (8%, 48+, 0- )
N : 0
---------------
DEL: 22
INS: 0
```

Figure 2.3: Two separate alignments viewed in IGV. The red bar toward the top shows us where in the reference we are located. Grey vertical bars on top of both tracks indicate read coverage. Hovering over these will give base counts at each position. Colored vertical bars, which can be seen above the lower construct, indicate that there is no overwhelming consensus for base identity at the given positions. The horizontal bars represent individual reads. When processing reads, aligners consider both the given reference and the reverse complement of the provided reference unless a single direction is specified. If a read aligns best with the given reference, it is labelled as a forward direction alignment. Conversely, if the read aligns to the reverse complement it is labelled as a reverse direction alignment. The direction of the read is indicated by a bit flag variable in the aligned output. In the figure above, blue horizontal bars indicate reverse alignment, and red indicate forward alignment. At the very bottom the reference sequence can be seen with small letters.

2.4 Modifications

Modified DNA nucleotides are nucleotides that are somehow chemically and structurally different from the canonical bases (A, T, C, and G). A rich variety of modifications exist, along with a variety of reasons why we may be interested in them. Some modifications are potential biomarkers of disease, while others can be utilized as targets for the treatment of disease. Others yet, serve important regulatory functions in the transcription process.

DNA modifications are an important topic of inquiry in the field of epigenetics that focuses on modes of inheritance that change the expressed genetic material. The methylation status of regions of DNA impacts its organization and accessibility. As a result, methylation is a well-studied phenomenon in epigenetics with a relatively greater amount of attention devoted specifically to 5-mC and 6-mA. DNA modifications are still continually being investigated for their potential epigenetic function.

Furthermore, modifications can be used as experimental markers. One such modification, which will be the focus of this thesis, is *inosine (I)*, which results from the deamination of deoxyadenosine to hypoxanthine [10]. Hypoxanthine without the ribose ring is, in other words, inosine's nucleobase. Chemically, this transformation occurs by the loss of an exocyclic amino group (2.4). While inosine may function as an analog to guanine and preferentially base pairs with cytosine, it can also base-pair with all other canonical bases. Inosines occur naturally in DNA at a relatively low rate but can lead to altered recognition sites and affect DNA expression. They have also been found at tissue-specific levels to occur in mRNA and have therefore been hypothesized to play an important role in regulating gene expression [11]. In fact, $A \rightarrow I$ editing events is the most common nucleotide modification event in the mammalian transcriptome. Most of these events are found in non-coding regions and suggest a regulatory function [12]. These modifications are driven by the ADAR (adenine deaminases acting on RNA) family of proteins [13]. Evidence points to these modifications influencing the sequence of synapses, and thus tuning nervous system function [14, 15]. ADAR-driven modifications have also proven essential in embryonic development [16, 17] and plays a role in differentiation decisions [18]. In addition, inosines have been hypothesized as a potential marker specific to, for example, cardiac ischemia [19].

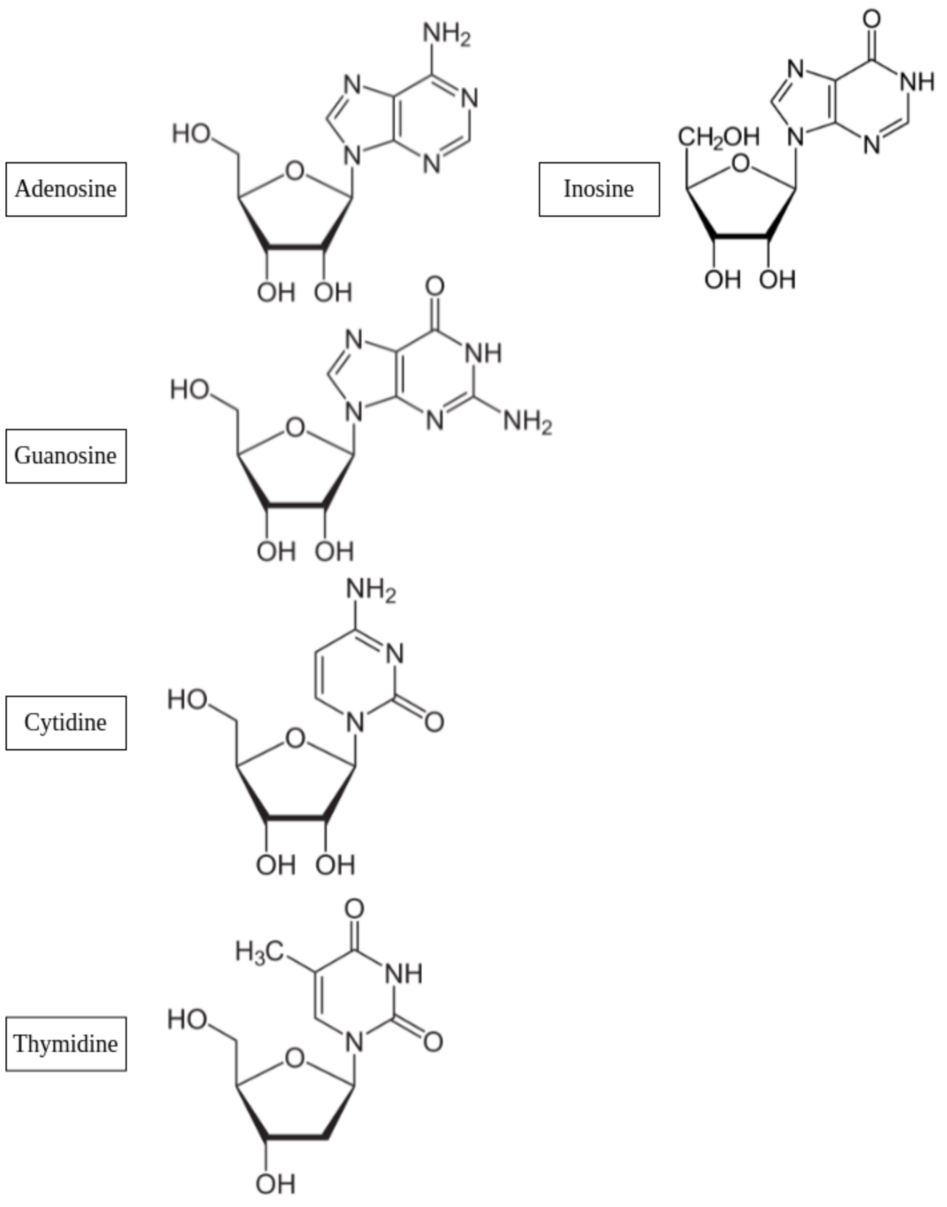


Figure 2.4: The chemical structure of the four canonical nucleosides and inosine. Structurally, we see that hypoxanthine, the nucleobase of inosine, is most similar to adenine. Nevertheless, the replication machinery treats inosine as if it were a guanine.

More relevant to the motivations of this thesis is inosine's ability to pair with all four canonical bases which makes it particularly useful for analysis of locations of DNA breakage. Substituting guanosine with inosine has been used to probe nucleotide properties and interpretation of binding studies [20]. One can envision an experimental design in which

DNA breaks are introduced at specific locations of interest or relevant functionality. These breaks can then be repaired by the introduction of inosine. In this way, a common marker for these breakage events is used for repair. By proxy, inosine becomes the marker for the underlying biological relevance informing the breakage. However, enabling the detection of this specific modification, and in fact any kind of modification, is inextricably dependent on the sequencing method.

2.5 Modification detection

As previously mentioned, detecting modifications using the sequencing-by-synthesis approach of short-read sequencing methods have inherent limitations. Their reliance on amplification makes it difficult to conserve modification-related information. In contrast, nanopore sequencing allows for the direct observation of modifications as they pass through the nanopore. The main advantage of this technique is that it enables us to detect multiple types of modifications without specified preparation protcols. These modifications need only be sufficiently structurally distinct from the canonical bases to enable their detection in nanopore sequence data. If this is the case, the modified nucleotide impedes the pore current differently than its unmodified counterpart, as well as all other bases that could potentially occupy a position in the sequence. By this direct observation, potentially complicating and confounding steps are eliminated from the procedure. In addition, specific preparation protocols for conserving the information held by the modification are circumvented. As a result, the experiment becomes faster to carry out and less convoluted in design.

As early as 2013, nanopore sequencing was shown capable of detecting cytosine methylation modifications [21]. However, there are still several factors that complicate the task of detecting modifications with nanopore sequencing. Importantly, as nanopore sequencing is still an emerging technology it is continuously being developed; there are basecalling error rates that are significantly higher than sequencing from earlier generations. Another consequence of the novelty of the technology is that the output data format is more limited in terms of applicable analysis pipelines and intermediate processing steps. The need for such steps becomes apparent as the analysis proceeds and thus scripts for carrying out these steps must be made on the fly. Finally, looking at the problem from a strictly numerical perspective, it is apparent that the task of classifying bases becomes more complex, even if we only consider a single modification. This is because instead of 1024 possible unique k-mers, we now have 5^5 or 3125 possible k-mers — more than three times as many possibilities compared to the canonical basecalling task. Further complications arise from the fact that the signal shift caused by the inclusion of a modified base in a given k-mer can be quite subtle [22]. As a result, the distribution of signals produced by a modifed k-mer can overlap with those produced by a corresponding unmodified k-mer.

2.6 Inosine detection and the current state of the art

Protocols analogous to the previously described bisulfite sequencing have been proposed for detecting inosine in RNA within an NGS framework [23]. One such approach, namely iSeq, works by treating the RNA with glyoxal. This leads to guanosines forming stable glyoxal conjugates. When the RNA is then treated with RNAse T1, which normally cleaves the strand after guanosines and inosines, guanosines are no longer recognized by the RNAse and the RNA is cleaved into segments with 3' inosines.

In a subsequent study, it was also pointed out that inosine can be detected in RNA from appearing in the sequencing profile as an A $\rightarrow$ G mutation [24]. Using a reverse transcription based sequencing technique such as Illumina, one would expect to identify an adenine by the presence of its complementary thymine in the reverse transcript. However, if the adenine has undergone A $\rightarrow$ I editing, the nucleotide would be treated as a G. This would be reflected by the inclusion of a C in the reverse transcript. Thus, this mutation signature could directly indicate the presence of an inosine. On its own, this discrepancy could easily be confused with single nucleotide polymorphisms or sequencing errors [25]. This can be alleviated by including a step of inosine cyanoethylation and increasing the depth of sequencing. This attaches acrylonitrile to the inosine, and as the reverse transcriptase reaches the inosine the procedure of reverse transcription is arrested resulting in a truncated RNA [26].

Both of these approaches are limited by the aforementioned challenges of detecting modifications using NGS sequencing. These are namely the modification-specific treatments and the indirectness of detection. The direct detection of modifications by nanopore sequencing eliminates the cost and potential error introduced by these shortcomings.

When it comes to detecting modifications in nanopore data, there have generally been two ways of approaching the challenge. One approach is to train a machine learning model on a specific set of modifications. The tool *DeepSignal* uses this approach to recognize 5-mC, 6-hmC, and 6-mA in synthetic samples [27]. While several studies have applied neural networks to this task, simpler machine learning algorithms have also been shown to be up to the task [28]. Another approach is to compare a sequence known to contain one or several modifications against a control sample which does not contain any modifications. This test-based method is used, for example, by the now archaic tool *nanoraw*, as well as by *Nanopolish* [29] and *NanoMod* [30]. Common to both approaches is that they compare current signals.

However, new developments within testing-based detection presents the possibility of de novo detection of any modification, that is without any a priori knowledge of the type of modification to detect. This potential was presented with nanoraw, which led to its successor *Tombo*. Tombo is a toolbox for modification detection, now owned and maintained by ONT. In addition to high-accuracy detection of specific bases, it provides a more error-prone functionality for de novo modification detection. In practice this is done by going over each position in a set of reads and comparing current signals against expected canonical signal levels. It also offers a function called level_sample_compare which compares between a modified sample and a control sample. This latter method has been used to detect inosine in nanopore sequenced data by introducing a chemical probe at the inosine modifications [31].

In the context of this study, the most pertinent advantage presented by nanopore is the direct observation of modifications. Consequently, the initially evident line of inquiry would then focus on nanopore's direct observation of inosine, and so indirectly on inosine's chemical structure. Prior studies have demonstrated the possibility of detecting DNA modifications in nanopore-sequenced data. Furthermore, one study in particular has shown the difference between inosine and adenine to be detectable within poly(A) RNA from analyzing and comparing current signal distributions [32]. The approach in our study is similar, but the task of detection will be more complicated as we will try to distinguish also from the other canonical bases. To address this, we will take advantage of other features from sequenced data which may aid in the classification.

Chapter 3

Experiments

To detect inosine with nanopore sequencing, three different nanopore sequencing experiments were performed during the course of this thesis. Observation from each experiment motivated the design of the subsequent experiment.

For each of these experiments, I will firstly describe the constructs designed and the rationale behind the contents of the constructs. Secondly, I will describe details regarding the analytical pipelines that we have used and the toolkits utilized in the process. Thirdly and finally, I will present the results of our analyses and provide a discussion surrounding the insights revealed by these analyses.

Each of the three experiments will be described in a separate chapter.

The first of the three is motivated by quantifying the difference in current signature between the canonical nucleotides and inosine. Here, we will investigate the potential for delineating between the "pure" signal of each of the five nucleotides by analyzing sequences of several instances of a single nucleotide. We will then look at the contribution of the nucleotides singularly incorporated in a more natural sequence context.

In the second experiment, we try to use an existing toolkit — Nanocompore — to identify constructs with incorporated inosines by comparing these modified samples against control samples, constructs where canonical nucleotides are present at the modified locations [33].

Finally, in the third experiment, we design our own machine learning framework for classifying the distinct modified and control samples used in experiment two. We will isolate a set of features from raw nanopore data, and train our classifiers on these data to identify the presence of inosine.

Chapter 4

Experiment 1

4.1 Rationale

Our aim with this first experiment was to explore if there is any detectable difference in the current signal level and dwell time between inosine and the canonical bases. Towards this end, we designed and sequenced two different DNA constructs as described below.

4.2 DNA constructs

4.2.1 Single-base context construct

As inosine can exist in nature sandwiched between different canonical bases, and because the nanopore current signature for a given base depends not only on the base itself but also on the two bases flanking both sides (the so called 5-mer context), we incorporated single inosine bases in three different flanking contexts in the construct used for this experiment. We will refer to this as the **single-base context construct** (4.1).

Five different constructs were designed containing either A, T, C, G, or I in three different contexts, also referred to as *variable* positions (see figure 5). A barcode at the 5'-end was incorporated to help distinguish these five flavours during analysis. Separating the three variable sites are spacer sequences, and upstream of the first variable position is a *GFP* sequence. By aligning first to the *GFP* region we can identify the single-base context construct reads. Next, by looking at the barcode region preceding the *GFP* region, we can identify which of the five varieties the sequence belongs to, and thus, which nucleotide is present at the three variable positions towards the 3'-end of the sequence. In making these constructs a mapping was established between the barcode and the base present at the variable sites. However, between the production of the constructs and the analysis presented here, this mapping was lost.

In the reference used to produce alignments for this construct, the variable sites are represented by N. We do this because we can't use I in reference sequences.

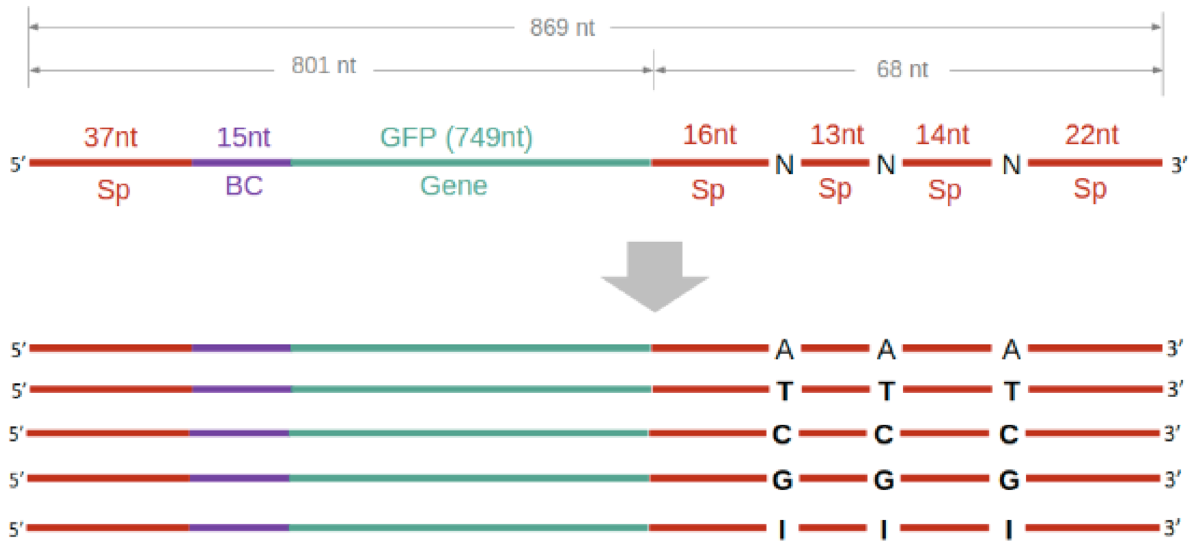


Figure 4.1: Single-base context construct. The construct in 5' to 3' direction consists of a 37-nt spacer, a 15-nt barcode, a *GFP* segment, a 16-base spacer, a single-nucleotide variable site, a 13-nt spacer, another single-nucleotide variable site, a 14-nt spacer, a third variable site, and finally a 22-nt spacer.

4.2.2 Homopolymer context construct

Because inosines can theoretically also occur as a stretch of two or more bases, we also wanted to investigate how homopolymer stretches of inosines differ from homopolymers of

A, T, C, and G in terms of current signal level and dwell time. Furthermore, this will allow us to get a representation of the pure signal contributed by inosine. And what that looks like versus the pure signal from the canonical bases.

To explore this we designed the **homopolymer context construct** (4.2) in which each of the canonical bases and inosine were incorporated in homopolymer segments. We incorporated inosine as a 10-nt homopolymer; thymine, adenine, and guanine as 9-nt homopolymers; and cytosine as an 8-nt homopolymer. Between the five homopolymers are spacer regions which simplifies separating data from each of the five homopolymer stretches. Preceding the inosine homopolymer, is a 29 base spacer sequence along with the sequence of the *ALKBH3* gene.

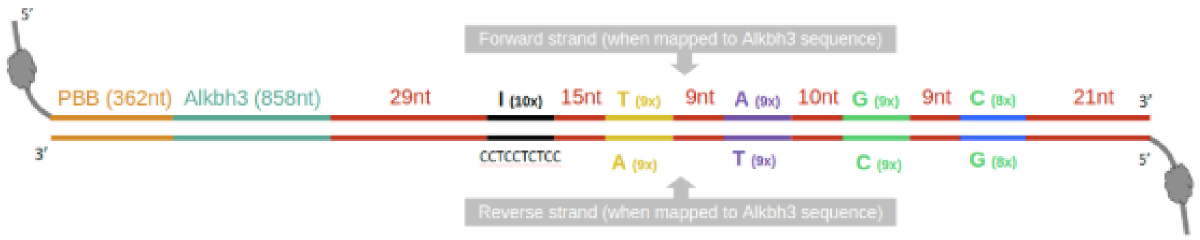


Figure 4.2: Homopolymer context construct. The construct in 5' to 3' direction on the forward strand consists of a plasmid backbone segment, an *ALKBH3* segment, a 29-nt spacer, a 10-nt inosine homopolymer, a 15-nt spacer, a 9-nt T homopolymer, a 9-nt spacer, a 9-nt A homopolymer, a 10-nt spacer, a 9-nt G homopolymer, a 9-nt spacer, an 8-nt C homopolymer, and finally a 21-nt spacer.

Since we sequenced the single-base context construct and the homopolymer context together, reads from the two constructs were pooled together and had to be separated. By aligning to a reference containing *ALKBH3* we can therefore identify the homopolymer context construct.

As mentioned, a reference cannot contain I, therefore we again represent the inosine homopolymer by a series of ten N's. We also systematically substitute each of the canonical bases in the reference at the inosine homopolymer and perform alignments to see how this affects the aligned data.

4.3 Sequencing

The two constructs used for analysis in this experiment were pooled together and sequenced on a single MinION using SQK-LSK108 kit. Pooling was done so that both samples could be sequenced in the same run on the same device so as to avoid any run-specific biases.

Sequencing produced 1153078 reads over the course of 1 hour and 45 minutes.

4.4 Initial pre-processing

Initially, we basecalled sequenced data using ONT's own basecaller — Guppy. Next, we did a quality control of the data in order to verify that the reads have an acceptable read quality and expected read length distribution using NanoPlot [34]. Next, we separated the two construct variants by aligning the data to both complete construct references using minimap2 (4.3). This allows us to check for each read whether it contains one of the two genes, and then subset the file containing all reads into two separate files depending on which of the two gene sequences (*ALKBH3* or *GFP*) is present. Furthermore, we subsetted these files again with the intention of sorting reads based on direction as well as removing unmapped reads. We do this by filtering on the bit flag variables produced by alignment. Here, we use sequence alignment toolkit samtools [35]. After these subsetting steps we are left with four files to use for further analysis.

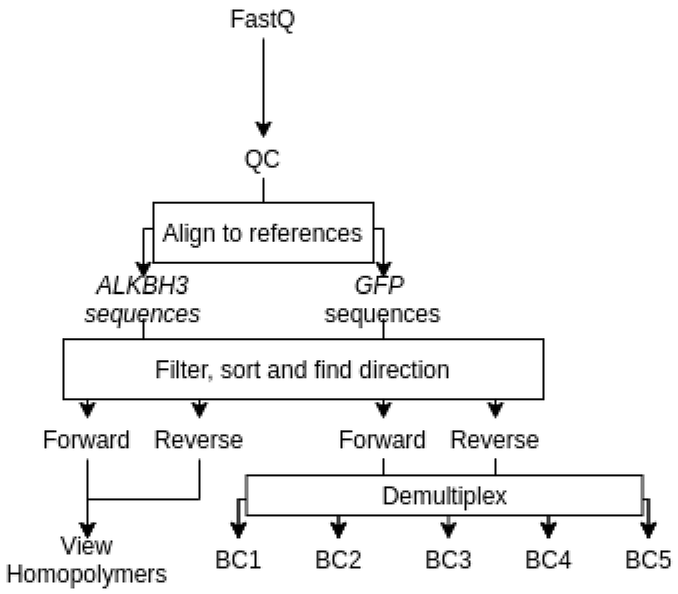


Figure 4.3: Pipeline for processing *ALKBH3* and *GFP* reads. We quality controlled sequences before aligning to the genetic references. After separating, we filtered reads by mapping and direction, and sorted the reads. We demultiplexed the *GFP*-aligned reads by barcode, and directly viewed the *ALKBH3*-aligned reads.

The separation with regards to direction of the reads is important since only the forward reads of the homopolymer context construct contain inosine (4.2). When quantifying inosine's characteristic current signature, the reverse strand does not contain relevant data. On the contrary, we are able to use the complementary regions for the canonical-base homopolymers for signal analysis. This is because the complementary regions here are homopolymers consisting of the complementary base to the canonical base in question.

4.5 Demultiplexing

The *GFP*-aligned single-base context reads also require further pre-processing as we must establish a mapping between the barcode sequence and the base present at the three variable sites towards the 3'-end of the construct. In the process of establishing this mapping we will sort reads by which barcode is present in the read. This process is called *demultiplexing.* Due to high error rates in nanopore sequence data, demultiplexing reads based on barcodes is challenging [36]. In fact, even with dedicated software for demultiplexing, as many as 20%

of barcoded reads can remain unassigned to any barcode and are therefore useless [37]. In our case, we are not concerned so much with the percentage of reads we are able to assign to the various barcodes, but when considering nanopore sequence data any demultiplexing procedure must incorporate some level of leniency when matching a basecalled sequence to the barcode sequences.

In order to demultiplex the reads first we convert the sequence alignment files — containing forward- and reverse-aligned sequences — into FASTQ format. This allows us to directly access the sequence information.

Next, we locate the barcode sequence. To do this, we first do a pairwise alignment of the 5'-end of the *GFP* sequence with the read sequence to home in to the 5'-end of *GFP* sequence in the read. If we find that this segment is present in the sequence, then 15 bases immediately upstream of this segment should correspond to the barcode. Therefore, in this restricted region of the sequence upstream of the *GFP* 5'-end, we then search for all five barcodes to see which of the five barcodes matches best by doing a pairwise alignment with each of the five barcodes. The barcode with the highest alignment score above a threshold was considered to be present in the read. This procedure leads to five .fastq files, each of which is a subset of the input .fastq file. In each of these .fastq files the barcode segments are judged to be the same, and, by the same logic, the three variable sites toward the 3'-end of the sequences will be the same for all sequences in any one of the five files. In this way, the five different flavours of reads were demultiplexed and their identity decoded despite the loss of true mapping information between barcode and what each barcode encoded for.

4.6 Producing consensus

The final step of pre-processing here is then to complete the mapping between the barcodes and the 3'-end variable sites. In order to do this we create a consensus for each of the five separate .fastq files. We produce this consensus by aligning to the *GFP* reference. This reference has an ambiguous sequence (N) in the barcode segment. After making this alignment, we firstly expect to be able to observe an agreement across the demultiplexed reads with regards to the barcode segment. Secondly, we expect to see that the consensus sequence has a clear majority base count of one nucleotide at the three variable positions toward the 3'-end. For each of the five sets of reads resulting from demultiplexing, the majority base should be the same across the three variable sites. Furthermore, we anticipate that the basecall made for inosine will reveal something about how the inosine bases are interpreted by a normal basecaller.

We also produce alignments for the *ALKBH3*-aligned reads to investigate if the inosine homopolymer segment coincides with our expectations. This will also give some idea regarding how inosine is interpreted by a standard basecaller. After doing so, we move on to isolating and analyzing data from each of the five homopolymers present in this construct. Expanding on how inosine is interpreted by regular processing framework, this will allow us to quantify inosine's underlying characteristic current signature. We will then compare this current signature against those belonging to the four canonical bases.

4.7 Processed current signal analysis

In the analytical portion of this experiment, we focus on extracting and compiling data which will allow us to quantifiably differentiate between I-nucleotides and the canonical nucleotides. For this step, we used reads generated from the **homopolymer context construct**. We initially focus on the homopolymers since these segments are long enough to span the entire nanopore (i.e. they have a length greater than 5-nt). This means that we can obtain current measurements resulting from only one type of nucleotide occupying the pore. Thus, we can quantify the "pure" current perturbation exerted by each of the five types of nucleotides analyzed here. We isolate current measurements and dwell-times of each of the five different homopolymer segments, and then do a comparison of the five.

In order to extract low-level signal information we use eventalign. From the resulting output we are then able to see which measurements contribute to the basecalled k-mer at all positions in the reference.

Nanopolish provides a pipeline for running eventalign. Firstly, reads are indexed with Nanopolish and aligned to the reference using minimap2. To produce the eventalign output, we provide as input our reads, a sorted and filtered alignment file, and a reference. This gives us a tab-separated values (.tsv) file as output. Since we are unsure if Nanopolish accepts references containing N, we perform the procedure with several different references: one containing N at the inosine homopolymer, one containing A at the inosine homopolymer, and a third reference with C in the same location etc. Within the output file, the sequence of the reference is divided into 6-mers. For each of the 6-mers present, we have access to several descriptive statistics such as:

1. The index position of the 6-mer relative to the reference
2. The sequence of nucleotides in the 6-mer
3. The read id
4. Statistics describing current signal mean and standard deviation
5. Length of the event, or the event's dwell time
6. Corresponding statistics for the model 6-mer which was used to inform the basecalling of the event
7. All the individual sample measurements that were used to calculate the statistics of the event

We are interested only in the signal from the five homopolymer segments for this analysis. Therefore, we eliminate superfluous data by defining five intervals of index thresholds and discarding all data that falls outside these thresholds. This way we retain only data describing the exact segments we are interested in. We then compile and calculate descriptive statistics for all events of each interval. For each read containing the interval of interest, we calculate the following statistics: mean current signal, the average standard deviation, and dwell time.

By further aggregating and calculating descriptive statistics from this output, we obtain metrics for each of the intervals. This can be seen as quantifying the contribution to the current signal perturbation of the single nucleotide the respective homopolymers consist of. We will then compare these statistics for all five nucleotides to see the differences in their effect on the nanopore current. To investigate the validity of this analysis, we also carry the analysis out on data that have been aligned to references with different bases present at the inosine homopolymer. To indicate a robust analysis, we would expect the measurements contributing to the statistics reported for the inosine homopolymer to be similar for all alignments.

4.8 Raw current signal analysis

Thus far, we have only analyzed data at the sequence level that have already been aligned. Now, we analyse the raw data. By this we mean data that has not undergone processing as a result of the eventalign procedure. However, extracting this data and ensuring it corresponds to the relevant events involves several more steps than described in the previous analysis.

We iterate through all .fast5 files, reading and extracting information from each. This renders data contained in each .fast5 file easily navigable. Here, the most interesting points of data are raw signal data, read id, and basecalled sequence. A useful structure we refer to as the **event data table** enables an association between specific segments of the sequence with intervals in the array of raw signal data. After finding a segment of interest in the sequence, the table allows us to extract the raw signal measurements corresponding to this segment. Importantly, we must also extract read-specific metadata pertaining to the sampling procedure. These are: block stride, digitisation, range, and offset; all are parameters to be used in the normalization of raw data.

Locating the five homopolymer segments and using corresponding raw data measurements as descriptive statistics gives us an idea of how each of the five homopolymer segments are processed by the nanopore. As nanopore sequencing is known to struggle with the length of homopolymers, we must expect these segments to appear in an inconsistent manner in the FASTQ sequence. Therefore, we locate them by proxy of flanking segments. We also subject the reverse complementary reads to the same analysis. In order to limit the influence of surrounding bases not belonging to the homopolymer, we excise the 25% first and last signal measurements. We then use the compiled measurements to calculate statistics on the normalized raw signal.

To address complications that arose with our prepared construct (see chapter 4.10.1), we generated an additional dataset from the same construct that we ran through the same analytical pipeline. In the results this is referred to as the second set of homopolymer context reads.

4.9 Training a classifier

After compiling these descriptive data points of raw read information, we use them to train a set of classifiers for distinguishing the five different homopolymers. In this preliminary classification we use the variables extracted, current signal mean and standard deviation, as the features to train on. The classification task is then to predict the base of which a homopolymer consists based on these input features. We trained several different learning models, while modifying the hyperparameters for some. For these purposes we use out-of-the-box keras models simply to survey their respective performances given the acquired features [38]. The machine learning algorithms we used were SVM, Decision Tree, kNN, Random Forest, and GBM.

4.10 Results

4.10.1 Producing consensus results

When producing consensus reads for each of the five files resulting from the demultiplexing, the results were not quite as expected. We observed an agreement for the barcode segment indicating that the demultiplexing itself was successful (4.4), but there was no consistent majority consensus for the three variable sites toward the 3'-end (4.5). Finding no viable explanation for this observation, we discarded the single-base context construct dataset as flawed.

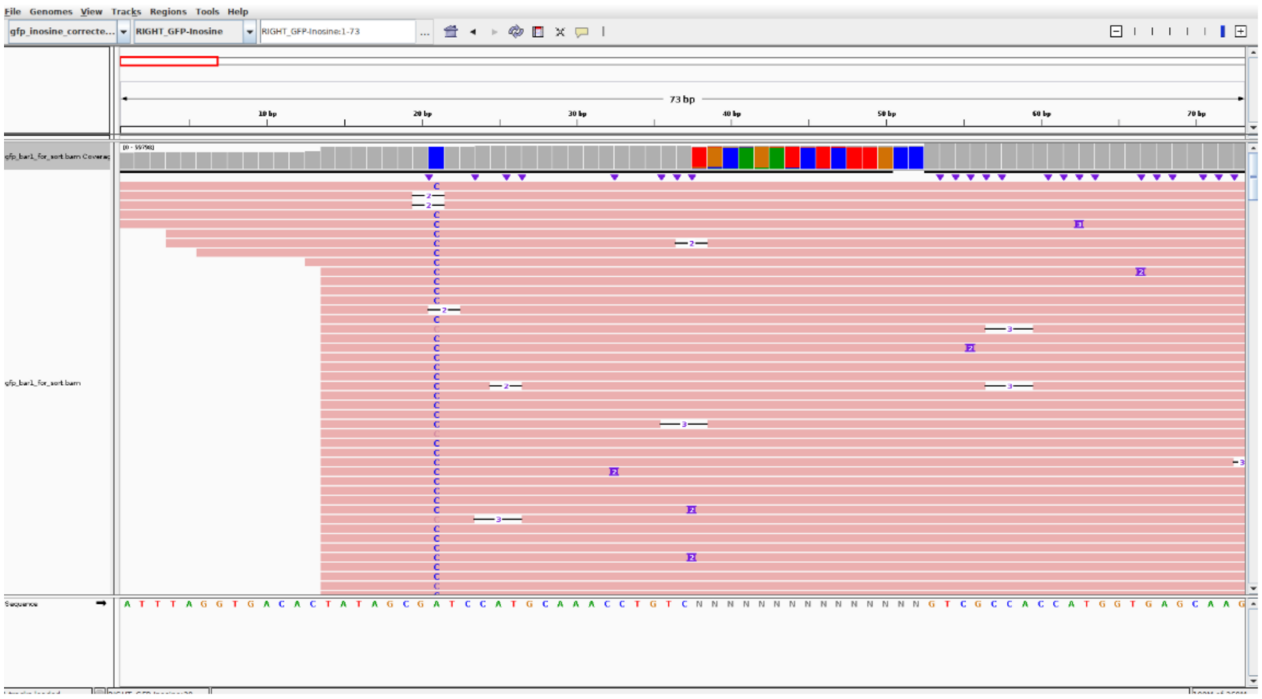
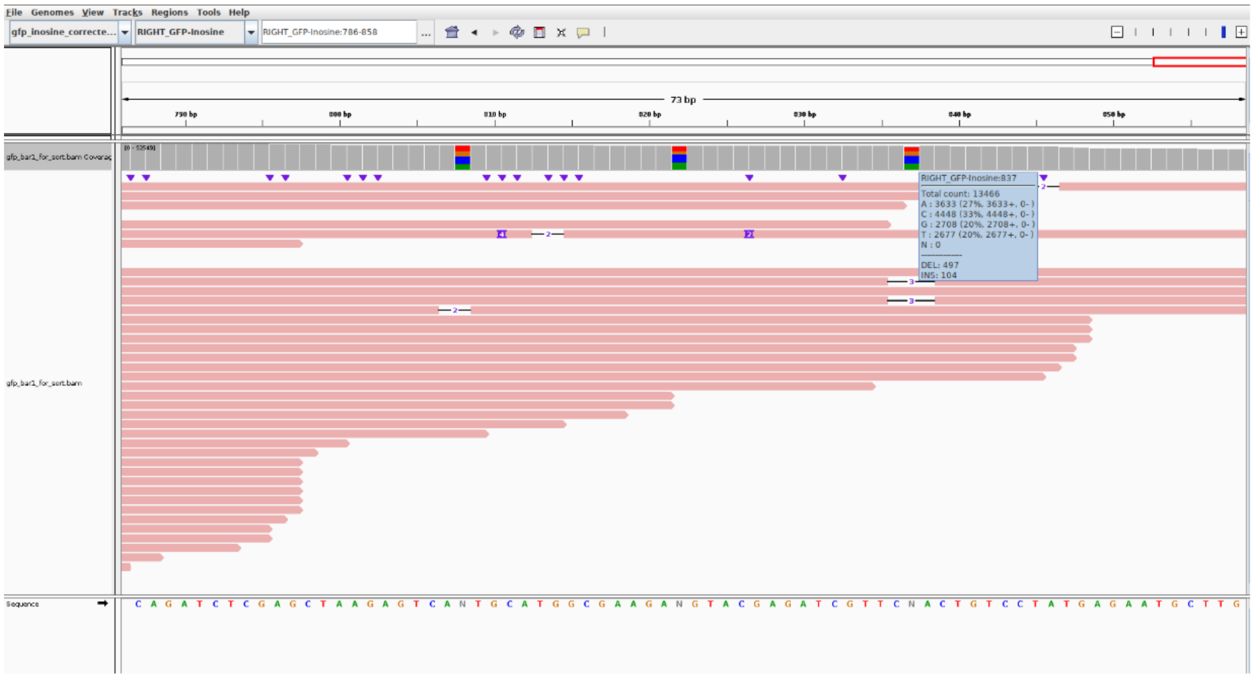

File Genomes View Tracks Regions Tools Help
gfp_inosine_correcte...
RIGHT_GFP-Inosine
RIGHT_GFP-Inosine:1-73
73 bp
10 bp
20 bp
30 bp
40 bp
50 bp
60 bp
70 bp
gfp_bar1_for_sort.bam Coverag
gfp_bar1_for_sort.bam
Sequence
ATTTAGGTGACACTATAGCGATCCATGCAAACCTGTCNNNNNNNNNNNNNNNGTCGCCACCATGGTGAGCAAG

Figure 4.4: 5'-end of consensus for demultiplexed single-base context reads. Here we show the alignment view for reads in which the second barcode was detected. The colored vertical bars denote the barcode segment. Because alignment score threshold allows for a mismatch or two in individual reads, there are reads containing unexpected bases at each of the positions as indicated by the colored bars. Nevertheless, each position shows a clear majority base. Therefore, we would expect the three downstream variable sites to show a similar agreement.


File Genomes View Tracks Regions Tools Help
gfp_inosine_correcte...
RIGHT_GFP-Inosine
RIGHT_GFP-Inosine:786-858
73 bp
790 bp
800 bp
810 bp
820 bp
830 bp
840 bp
850 bp
[0 - 52549]
gfp_bar1_for_sort.bam Coverag
gfp_bar1_for_sort.bam
Sequence
C A G A T C T C G A G C T A A G A G T C A N T G C A T G G C G A A G A N G T A C G A G A T C G T T C N A C T G T C C T A T G A G A A T G C T T G

```
RIGHT_GFP-Inosine:837
Total count: 13466
A : 3633 (27%, 3633+, 0- )
C : 4448 (33%, 4448+, 0- )
G : 2708 (20%, 2708+, 0- )
T : 2677 (20%, 2677+, 0- )
N : 0
---------------
DEL: 497
INS: 104
```

Figure 4.5: 3'-end of consensus for demultiplexed single-base context reads. Here we show the alignment view for the variable sites from the same set of reads as seen in figure 4.4. The three vertical colored bars indicate the three variable sites. We see there is no pronounced majority base at any of the three positions. The proportion of base frequencies is about equal for all four canonical nucleotides.

Next we produced consensus for the homopolymer context construct and observed that this matched our expectations to a greater degree. However, instead of a consensus for all positions within the I homopolymer region we observed a consensus sequence of GGAG-GAGAGG. This corresponds to the reverse complement of the CCTCCTCTCC segment incorporated in the reverse read at this location (4.6). This was caused by a ligation error in preparing the construct that produced the first set of reads. The segment from the start of the inosine homopolymer to the 3'-end of the forward strand was unsuccessfully incorporated, and as a result the forward strand was fixed by extending from the breakage point. In this situation, the forward strand only contains the reverse complement of the reverse strand which would explain why we see the reverse complement pattern where we would expect to see a pattern indicating the presence of our inosine homopolymer.

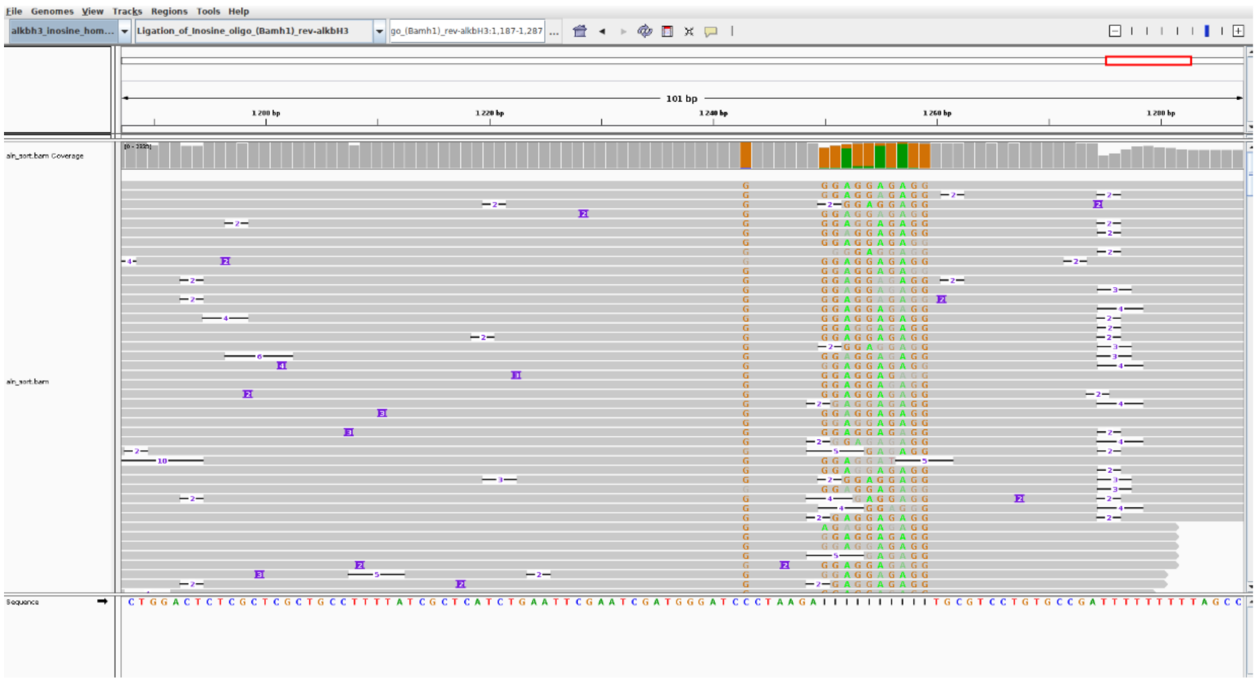


Figure 4.6: 3'-end of consensus for homopolymer context reads. The inosine homopolymer is indicated by the stretch of colored vertical bars. This stretch indicates a consensus of GGAGGAGAGG, the reverse of what is present on the reverse complement of the strand containing the inosine homopolymer.

In response to this unexpected observation, we generated a second set of homopolymer context reads. We prepared and sequenced another oligo following the same design as previously. When producing an alignment for this newly sequenced construct we saw no such reverse complementary pattern at the inosine homopolymer. Rather, we could see that all positions in the homopolymer construct exhibited some level of uncertainty. For all positions, the base counts were relatively similar, with either guanine or adenine being the most frequently observed base (4.7). This aligns with what we would expect to see in a construct containing an inosine homopolymer.

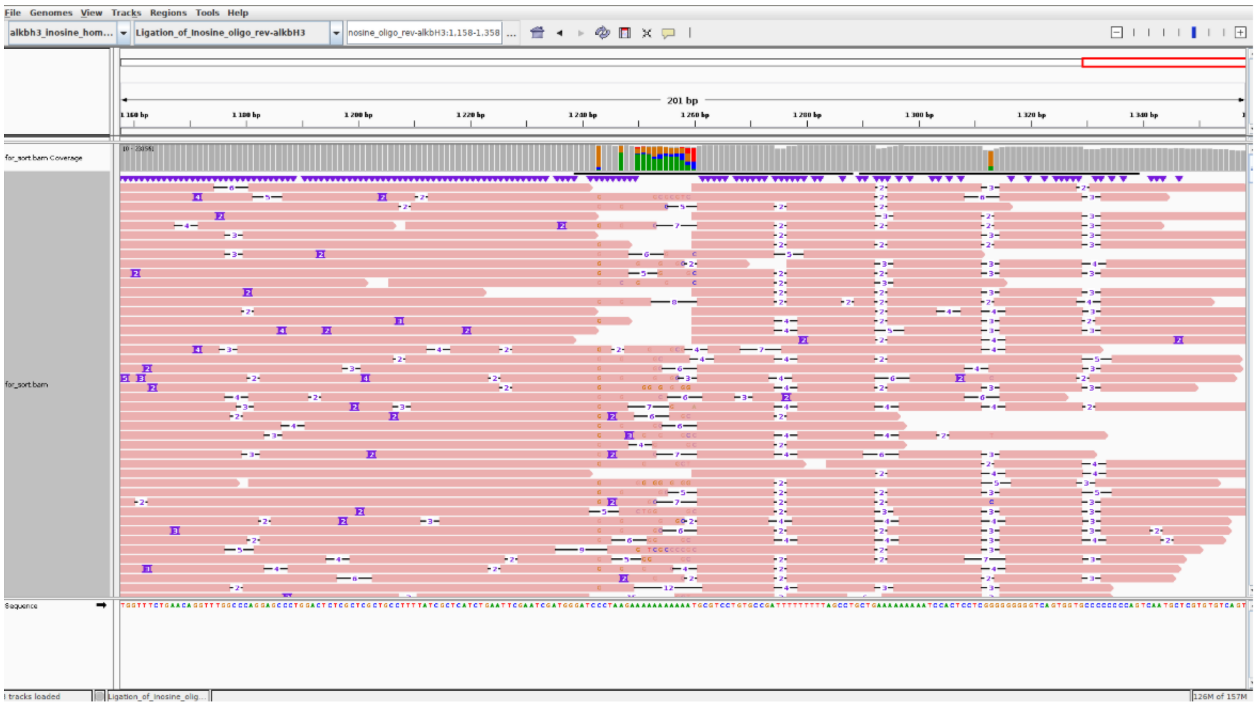

File Genomes View Tracks Regions Tools Help
alkbh3_inosine_hom...
Ligation_of_inosine_oligo_rev-alkbH3
201 bp
for_sort.bam Coverage
for_sort.bam
Sequence

Figure 4.7: 3'-end of second consensus for homopolymer context reads. Here we view alignments produced for the forward *ALKBH3* reads. The colored vertical bars again indicate the segment corresponding to the inosine homopolymer. We see that the bars are predominantly green and yellow, respectively indicating a majority base count of adenine and guanine.

4.10.2 Processed current signal results

After locating the signal-intervals that correspond to the five different homopolymers and calculating descriptive statistics, we obtained the following results. Notably, these results were produced by using a reference containing N in the I homopolymer in the eventalign step. We obtained a mean current signal value of 86.83 pA and mean dwell time of 0.00381 for the I homopolymer. For T homopolymer, mean current signal value was 90.51 pA and dwell time was 0.00808. The A homopolymer had a mean current signal of 87.52 pA and a dwell time of 0.00817. The G homopolymer had a mean current signal of 74.20 pA and a dwell time of 0.00970. Finally, for the C homopolymer, we observed a mean current signal of 98.85 pA and a dwell time of 0.00466.

After gathering the same statistics for the output produced with A in the reference in the I homopolymer, we see virtually no change in any of the homopolymer mean current signal values or dwell times. This is to be expected for the canonical base homopolymers, but it is more surprising with regards to the I homopolymer. In this case, the I homopolymer reports a mean current signal value of 86.83 pA and a mean dwell time of 0.00379.

Substituting the series of A-s representing the I homopolymer with all C-s in the reference, we redo the analysis once more. While the current signal means and dwell times for the canonical bases remains unchanged, the same does not hold true for the I homopolymer. In this case we obtain a mean current signal of 99.71 pA and a dwell time of 0.00306 for the I homopolymer.

4.10.3 Raw current signal results

After obtaining the raw signal measurements corresponding to the homopolymers in the first set of reads produced from the homopolymer context construct we analyzed this data. When we observed that the results did not align with our expectations, and considered this in conjunction with the unexpected alignment observations (4.6), we realized that an experimental failure had occurred.

Having observed a pattern for the inosine homopolymer more in line with our expectations in the second set of homopolymer context reads, we also analyze these further. We obtained the following statistics: I homopolymer mean current signal was 71.79 pA (n = 688970) with a standard deviation of 8.75, T homopolymer mean current signal was 81.54 pA (n = 754234) with a standard deviation of 8.69, G homopolymer mean current signal was 66.16 pA (n = 249554) with a standard deviation of 8.32, C homopolymer mean current signal was 91.94 pA (n = 179336) with a standard deviation of 9.78, and for the A homopolymer the mean current signal was 76.16 (n = 512014) with a standard deviation of 8.96 (4.8).

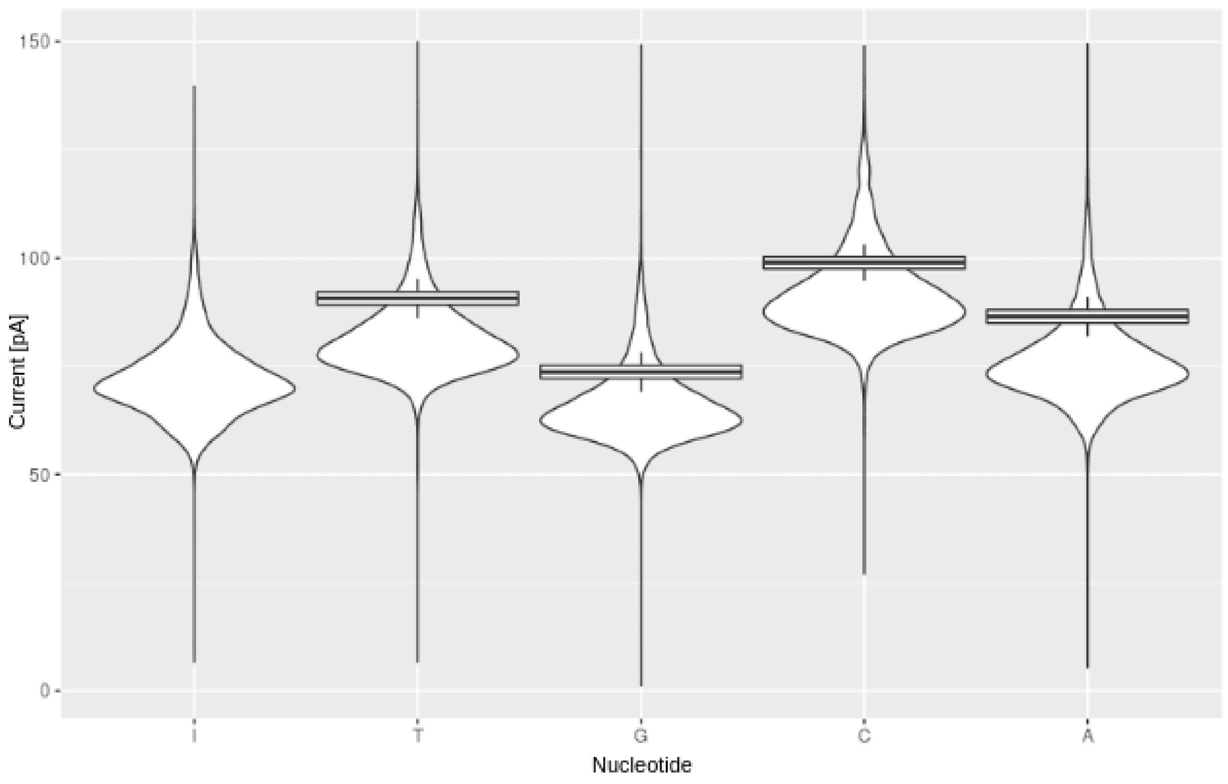


Figure 4.8: Violin plot showing distributions of the current signals obtained for each of the five homopolymers. We overlay the boxplots, which represent ONT provided expected current signal means for the four canonical bases. From these we can see a systematic bias that is consistent for all four: current signal measurements are shifted slightly lower. We attribute this to a difference in the voltage applied across the membrane between our sequencing runs and those that produced the nanopore reference data. Difference in voltage can cause systematic shifts like those exhibited here.

4.10.4 Classification results

Performance of the various classifiers were relatively similar, though GBM and SVM consistently outperformed the others in terms of accuracy (4.1). Hyperparameter tuning, which was carried out for SVM and kNN, had little impact on the accuracy of the former and a slight effect on that of the latter. For the various classifiers trained, accuracy generally plateaued right below 49%. We also generated confusion matrices to identify which bases most frequently were incorrectly classified (4.2). Based on the performance of the various

	accuracy	MCC
SVM	47.74%	0.35
Decision Tree	44.95%	0.31
kNN	43.88%	0.30
Random Forest	45.91%	0.34
GBM	48.30%	0.35

Table 4.1: Models used to classify homopolymers and their performances. SVM was evaluated with various kernels, regularization parameters, kernel coefficients and decision function shapes. The model represented above uses an RBF kernel with a coefficient of 10, a regularization parameter of 100, and an one-vs-one decision function. KNN was evaluated with various values for n. The chosen model here had n = 9.

		Predicted class				
		I	T	G	C	A
True class	I	418	167	117	88	209
	T	44	503	5	254	194
	G	191	73	520	10	56
	C	0	193	1	399	47
	A	181	296	28	166	327

Table 4.2: Confusion matrix for five-way classification with GBM. The accuracy for this classification was 48.30%, and the MCC was 0.35. We see inosine most commonly misclassified as adenine.

classifiers used, we use as an example the confusion matrix resulting from the best performing classifier: GBM. Unsurprisingly, signal means from the inosine homopolymer were found to be confused most frequently with those of the guanine and adenine homopolymers.

4.11 Discussion

In observing the statistics produced from analyzing the processed current signals from the homopolymer context construct, we see that the statistics for the canonical bases coincide with expected values. However, the minimal difference between the I homopolymer and the A homopolymer is a disconcerting finding. To be sure of the validity of our results we ran the pipeline again, with N substituted for A at the I homopolymer in the reference. This produced almost identical results as the previous run of the analysis. The current signal means for the inosine homopolymer deviated by 0.02 pA, and dwell time by 0.00002 units.

From this we can draw one of two conclusions. Either Nanopolish's eventalign function does not handle the usage of N in the reference, or the difference between the I and A homopolymers is in fact minimal. To verify which of these scenarios is the case we repeat the analysis pipeline once more with another of the canonical bases (C) substituted for the I homopolymer in the reference.

When considering the statistics obtained from the data produced with C in the reference at the inosine homopolymer, we note that the mean current signal observed for the I homopolymer differs considerably from that which we observed with A and N in the reference. Furthermore, it is much closer to the measurements observed for the C homopolymer. This would indicate that of the two previously considered scenarios the former is the case. When using N in the reference in the eventalign procedure we cannot reliably consider statistics obtained for the I homopolymer. Meanwhile, when using a substitute base in the reference at the I homopolymer, some a bias is clearly introduced into data.

We also look at the individual sample measurements used to calculate the mean current signal for the relevant events. Both the number of signal samples and their values were different for the same events in identical reads aligned with different references. The problematic effect is reflected in the statistics reported for the inosine homopolymer, which tend to be skewed significantly toward signal values which would be expected for the base used as a substitute for inosine in the reference. The difference in signal samples indicates that reads processed with eventalign in the manner described previously cannot reliably be analyzed for our purposes.

In the results from the raw current analysis of the second homopolymer dataset, we observe a shift compared to the ONT provided values which are expected for the canonical homopolymers. This shift is consistently proportional for all four canonical homopolymers. We see that inosine's average signal is most similar to that of guanine and adenine. This is expected as inosine is structurally most similar to these nucleotides. In addition, viewing the consensus results for the second homopolymer dataset also indicates that a basecaller interprets inosine most frequently as adenine or guanine. Regardless, it is promising to find each of the five homopolymers exhibiting evenly spaced current signal means (4.8). This is what we try to demonstrate in training a classifier on obtained data and subsequently evaluating it.

Given the current signal measurement distributions, and their overlaps (4.8) it is also perhaps not very surprising that the classifiers were unable to sufficiently distinguish between these five classes. The conclusion, based on the insight from this classification task, is that either a different approach must be taken or more features are required. Specifically, if it was possible to extract dwell time while we extracted signals to be used for classification this might have provided further discriminative information for our classifiers. However, since 25% of the first and last signal measurements were excised arbitrarily, there was no way to extract the dwell time for only the retained signal measurement. Going forward we keep these insights in mind, but move on to a scenario that more closely resembles a real-life application. This is a scenario in which one has to distinguish between one construct that has inosine in it and another which does not contain inosine.

Chapter 5

Experiment 2

5.1 Rationale

The approach of the second experiment was further geared towards identifying the difference in characteristic nucleotide signature explored in Experiment 1 and distinguishing based on these. Here, however, we take a test-based approach by doing pairwise comparisons between sequences where inosine and the canonical bases have been incorporated at three locations within the same contexts. The basis for this experiment is a more realistic use-case scenario. In this scenario one sample, containing inosine, is considered the modified sample while the others serve as control samples. We aim to observe the differences between the modified sample and all four control samples reflected in data. Based on these differences, we wish to draw conclusions with regards to the realizability of an accurate classification in this comparative setting.

5.2 DNA constructs

The idea of five different construct variants is also the basis for the third and final dataset presented here. In this case, the five different variants were all of different length, but with an invariant segment of 69 nucleotides at the 3'-end of the construct. The five constructs also differ at three specific sites within this invariant segment (5.1). Upstream of the invariant segment is a stretch of varying length that is unique to the five different constructs. These can be used to identify which of the five constructs a given read belongs to. The idea behind this data is that we can do a pairwise comparison between the invariant segments across the different constructs. By comparing the three variable positions we can assess whether it is possible to tell a construct containing inosine at these three positions from a construct containing one of the canonical bases here. These constructs will be referred to by the base occupying the three variable sites, i.e. the construct containing inosine will be referred to as construct I.

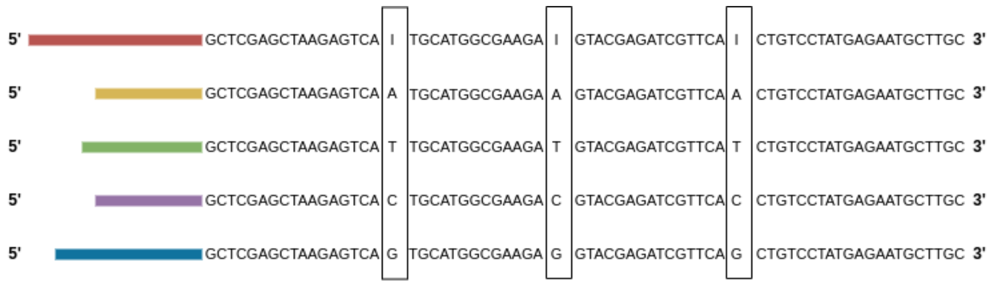

5' GCTCGAGCTAAGAGTCA I TGCATGGCGAAGA I GTACGAGATCGTTCA I CTGTCCTATGAGAATGCTTGC 3'
5' GCTCGAGCTAAGAGTCA A TGCATGGCGAAGA A GTACGAGATCGTTCA A CTGTCCTATGAGAATGCTTGC 3'
5' GCTCGAGCTAAGAGTCA T TGCATGGCGAAGA T GTACGAGATCGTTCA T CTGTCCTATGAGAATGCTTGC 3'
5' GCTCGAGCTAAGAGTCA C TGCATGGCGAAGA C GTACGAGATCGTTCA C CTGTCCTATGAGAATGCTTGC 3'
5' GCTCGAGCTAAGAGTCA G TGCATGGCGAAGA G GTACGAGATCGTTCA G CTGTCCTATGAGAATGCTTGC 3'

Figure 5.1: Five constructs with three variable sites embedded in an invariant 3'-segment. The figure shows the five varieties of the construct analyzed in experiments 2 and 3. Toward the 5'-end is a segment of variable length and content indicated by colored horizontal bars of varying lengths. The rest of the construct is identical with the exception of the three variable sites accentuated by the rectangles.

5.3 Sequencing

The construct was pooled with another construct used for a different analysis and sequenced on a single MinION using SQK-LSK109 kit. The sequencing produced 3104181 reads over the course of 5 hours.

5.4 Basecalling and quality check

The sequenced files are, as in the previous experiment, first basecalled using ONT's Guppy. Again, the results of said basecalling procedure are investigated with NanoPlot, and found to generally hold a consistently acceptable quality. As a result, no reads are filtered from the batch of basecalled reads.

5.5 Alignments

Producing alignments presents the first considerable challenge of this analysis. We want to identify and separate the five distinct constructs from one another, but we need to make sure construct I is handled in a way that does not complicate downstream steps. The challenge is: *what base should be present in the reference at the three variable positions when producing alignments for construct I?* In later steps of the analysis, we use Nanopolish again which, as we have preiously demonstrated, can not reliably handle ambiguous nucleotides (N) in reference. The way we address the problem of reference is to produce four alignments, one for each of the canonical bases to be substituted at the inosine position. Data resulting from these four different alignments will then be used for four separate comparisons. When comparing construct I with construct A for example, we would use data from construct I that had been aligned with A present at the three variable sites. This is a way of simulating a natural scenario where we have a known reference and relative to the known reference we check for modifications. In terms of handling the challenge introduced by the aligner trying to find the best fit for the data versus the reference, this will allow for the most robust analysis when the comparison stage is reached. All in all, we set out to produce 9 alignments; one for each construct with the exception of construct I for which we create five (5.2).

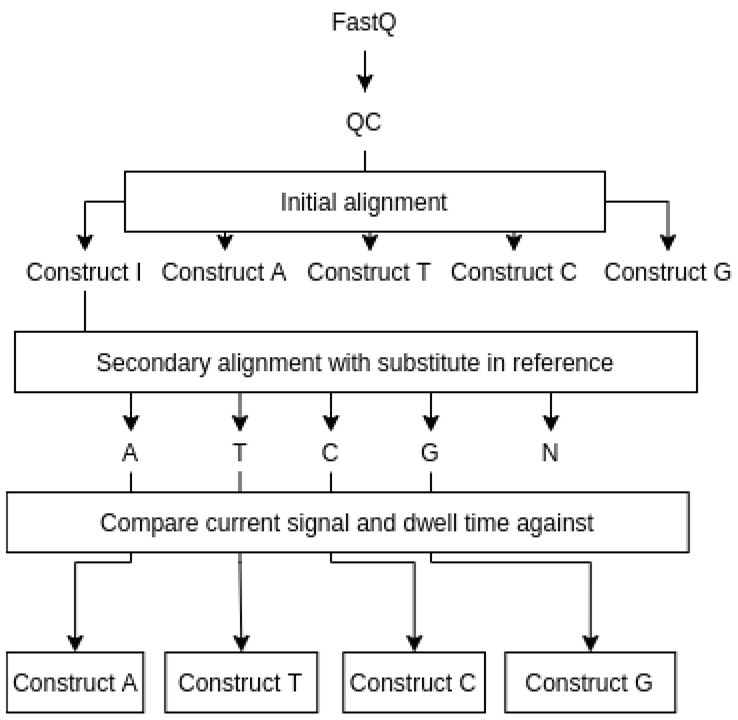


Figure 5.2: Procedure of generating alignments for pairwise comparison. We first identify the five unique constructs through an initial alignment. Then we produce five different alignments for construct I with different bases present at the three inosine positions. Four of these are intended to be compared against specific constructs. We also align with N in reference to check if this might work despite our previous experience with ambiguous positions in reference with Nanopolish.

Another set of alignments were produced to format data for the five different constructs as similarly as possible. These are alignments where only the invariant 69-nt segment towards the 3'-end was used as a reference. This differs from the previous set of alignments which used the whole constructs as reference. With the exception of the three variable positions within this segment, the reference is now identical for all constructs. We must still find a way to separate the five constructs from one another. To enable this distinction the input for these alignments are the outputs of the initial alignments which separated reads based on which construct they belonged to. One potentially useful consequence of this alignment is the fact that all read information is relative to references that are the same length. This may make the task of doing position-by-position comparisons between constructs less complicated.

The alignments are again made with minimap2, and further indexed, filtered, and sorted with samtools.

5.6 Initial pre-processing: preparing comparable files

To reiterate, the goal of this analysis is a position-by-position current signal and dwell time comparison between two constructs. Comparing two constructs, we expect the current signal and dwell time of the two to be identical at all positions with the exception of the three variable sites.

The first step in enabling this comparison is to format the read information so it can be interrogated on a position-by-position basis. Again, we use Nanopolish' eventalign function to output a .tsv file which describes every event of every read. The software we use for the actual comparison, Nanocompore, requires some further processing.

We have to collapse the eventalign output by k-mer. This is done with a tool called NanopolishComp which has an eventalign_collapse command. As the name of the tool indicates, it collapses an eventalign file. In some cases, nanopore may interpret a single k-mer as several different events. Eventalign_collapse takes the information regarding such a k-mer and creates a single representative event. Each k-mer in the reference then occurs once in each read of the collapsed eventalign file. The collapsed events are consolidated into one composite of the events corresponding to this k-mer from a single read. In addition to the current signal mean for the events, the output of eventalign_collapse also contains dwell time for each k-mer in the reference.

The central assumption made by Nanocompore for a position-by-position comparison of two constructs, is that the two constructs are identical with the exception of potential modifications. In other words, a single reference is used for both constructs. Here however, we are working with constructs which, in their entirety, are variable in length and content. We must therefore prepare data in such a way that Nanocompore understands that we are only interested in the identical segments common to all five constructs. Namely, the 69 invariant nucleotides towards the 3'end of the construct. Having created collapsed eventalign files for the all five constructs in their entirety, we are not able to make comparisons of these files directly since the commonalities between them are not apparent. A number of approaches were taken to address this issue.

The first approach was to prune all excess information. By excess information, we mean all information describing events prior to the invariant 69 nt segment toward the 3'-end. We use the reference index numbers to only retain events occurring in the invariant segment. Though we are able to eliminate excess information, the eventalign_collapse procedure also outputs a corresponding .tsv.idx file required by Nanocompore. This file cannot be modified to reflect the modifications made by the pruning script. And so, downstream analysis carried out on the collapsed and pruned eventalign triggers errors.

To circumvent the issue introduced by the .tsv.idx file, we opted for a simple workaround. This is to swap the steps of pruning and collapsing. Instead of pruning the collapsed eventalign file, we would now be pruning the original eventalign file. The subsequent collapsing step should then produce a .tsv.idx file which corresponds to all the information held in the collapsed eventalign file. We make slight modifications to the procedure, but the idea of execution is identical: to simply excise the irrelevant information contained in each read. This enables the comparisons of sequences perceived as identical.

5.7 Comparison and various preparative steps

Comparisons in this experiment were carried out in a pairwise manner. Construct I was compared with each of the other constructs. This resulted in a total of four comparisons.

We carried out the actual sample comparisons with the tool Nanocompore, which has a function called SampComp. This is a tool specifically developed for the comparisons of two RNA constructs, one being the control containing no modifications and the other containing modifications. After contacting one of the developers of this tool, Tommaso Leonardi, we were informed that a workflow with Nanocompore on DNA data should be feasible and similar to a standard protocol on RNA data.

The output of running Nanocompore is three files: A *results* file that holds the statistical results carried out for each event in the two constructs; a *shift_stats* file that contains basic descriptive statistics for both constructs per position; and a *database wrapper object* that contains the underlying data from which the statistical results are calculated.

A recurring issue in running the comparison for our eventalign files was that the SampComp output files were filled with exclamation marks. This appeared to be an error-response put in place to notify of mismatches between reference and construct positions. We employed multiple pruning strategies to ensure that the reference used for comparison was common for both constructs, and that these were the same references used to align to the constructs that were to be compared. Even after ensuring this to be the case, the exclamation marks persisted.

We noticed that after tweaking the SampComp scripts slightly, though the exclamation marks were still present in the results output, data present in the shift_stats output looked like the sort of output we were expecting. After cleaning up the results output and removing the exclamation, we were left with a file that could be used for plotting these results.

We produced line graphs describing the change in current for all constructs. The plot was made with bokeh, using a template plot-maker script provided in the Nanocompore documentation. Only case-specific modifications were made to this script. Using bokeh allowed for an interactive plot where the currents of all constructs could be selectively viewed.

5.8 Results

When viewing the plot of all comparisons made to visualize the results of the position-by-position current comparison, we expected to see a difference between the inosine construct and the construct that it was being compared to at the inosine positions. Furthermore, since the constructs were identical with the exception of the three variable sites, we expected to see no difference in current signal mean at all positions except these variable sites.

Selecting a view corresponding to the pairwise comparisons, for example, viewing construct I aligned with A in the reference along with construct A, we saw no such difference. The two constructs look identical for all positions (5.3). This is the case for all other pairwise comparisons made here (5.4).

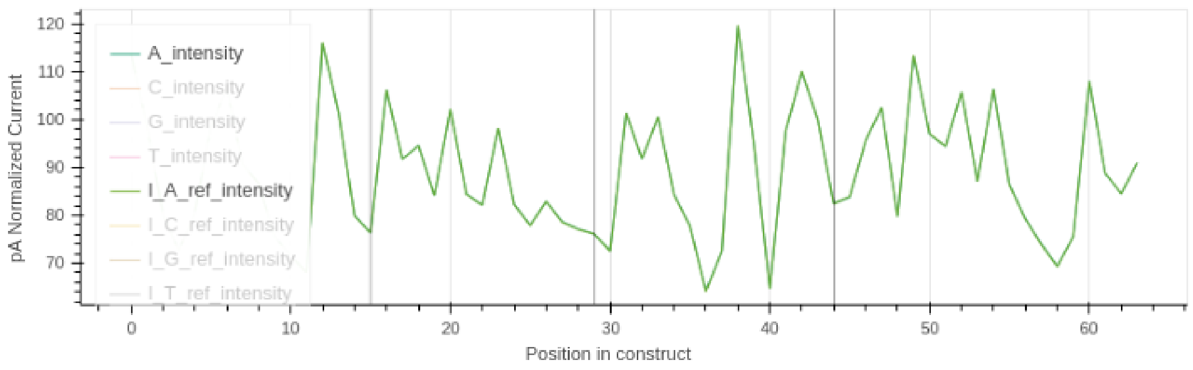


Figure 5.3: Pairwise comparison of construct I and construct A. The plot shows the graphs of mean current levels resulting for construct I (aligned with A present in the reference at the inosine positions) and construct A. The two lines are so similar that they overlap each other and hence only a single line can be seen in the plot. The bolded vertical grey lines indicate the three variable positions.

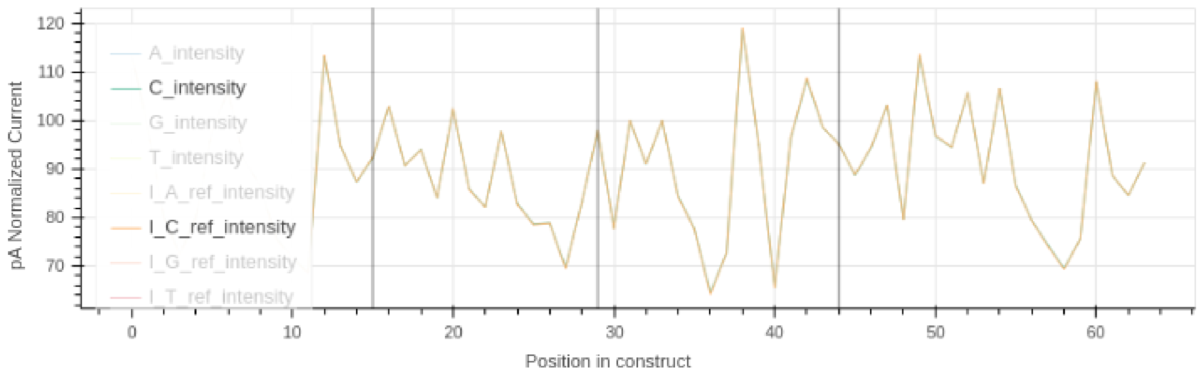


Figure 5.4: Pairwise comparison of construct I and construct C. Here construct I has been aligned with C present in the reference at the three inosine positions to prepare for comparison with construct C. Again, we see there is no detectable difference in current signal between the two constructs, which are known to be distinct at the three positions indicated by the vertical bolded grey lines.

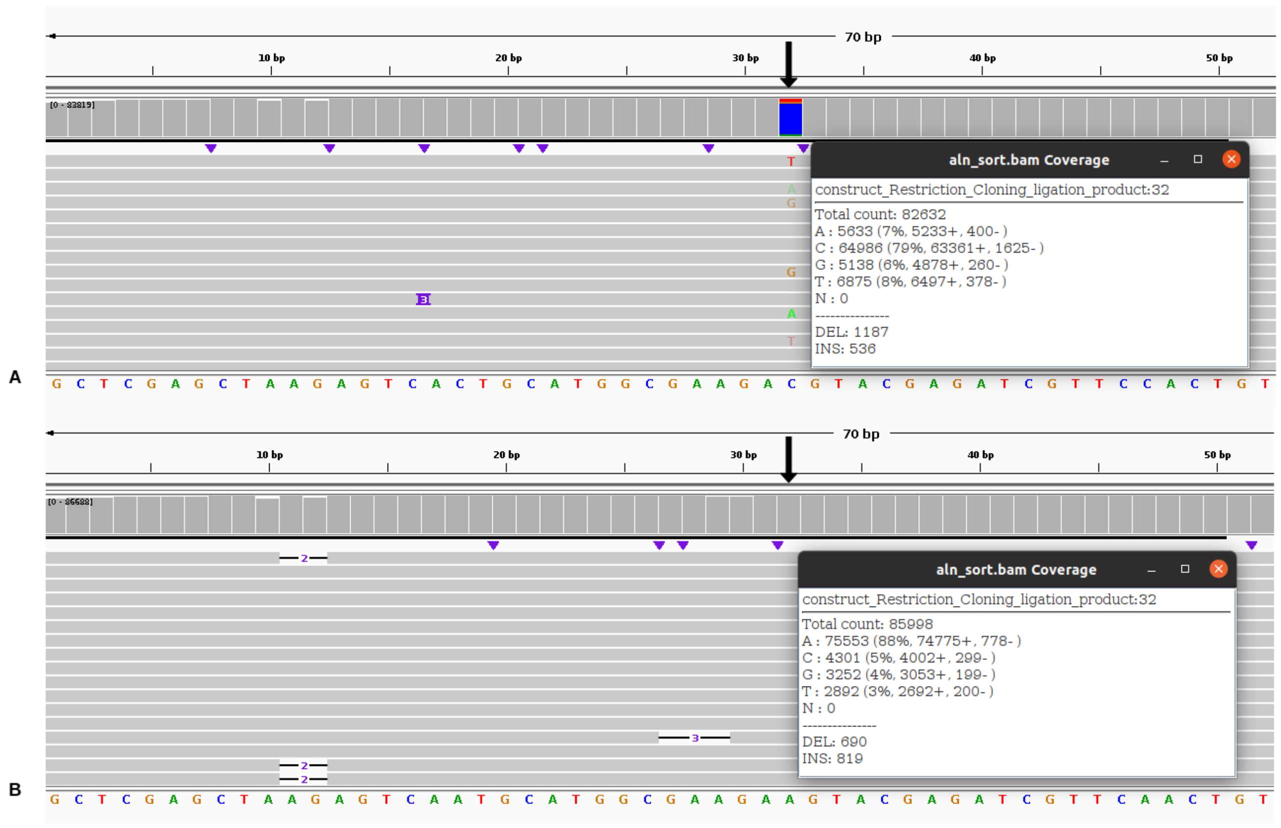
5.9 Discussion

While proposing the design of this experiment, we had expected to enable the comparison between two DNA constructs known to be different from one another at exactly three locations. While we were able to make these comparisons, the resulting comparisons showed no difference when looking at an inosine-containing construct versus a construct with one of the canonical bases present at the same position. Comparing the signal difference at one of these variable sites versus one of the invariant sites does not disclose a tendency toward a larger current measurement difference at the variable site relative to the rest of the construct. Since we know that the two constructs contain different nucleotides at the variable location, this is highly unexpected.

It is challenging to pinpoint exactly why this appears to be the case. Possibly, there could be an undisclosed error in the Nanocompore framework that does not handle DNA data in the same way as it does RNA data. However, also curious to note are some potentially complicating observations that were apparent when the second round of alignments were carried out to prepare construct I for comparison and these were made in IGV. As an illustrating example, when aligning to the construct I the resulting alignment file should have only data describing the inosine construct. Since inosine is an ambiguously interpreted base the three variable sites should reflect this in the base identity frequency. In other words, when looking at the most frequently observed base at each of these three sites, we expect to see some disagreement. Specifically, we would expect the frequencies for observed adenine and guanine to be higher relative to cytosine and thymine. This is because we have observed inosine to be more structurally similar to the purines.

What we actually observed at these sites did not match these expectations (5.3). We could illustrate with an example of the inosine construct aligned to a reference with adenine present at the variable sites. Bear in mind that this construct is then being prepared to be compared to a construct that in fact has adenine present at these three sites. When looking at the base frequency at each of the three variable sites there is no apparent disagreement or ambiguity with regards to base identity. The most ambiguity observed is for the second of the three variable sites, where 88% of reads are reported to contain adenine at this position. Meanwhile, over 95% observed adenines are reported for the first and the third variable sites. This could also be a factor of the two rounds of alignments, first to the whole construct and then to only the invariant segment which is used for comparison. When viewing the alignment produced by using the whole construct as a reference, a certain level of ambiguity, ranging from 79% - 84% majority base, is present for all three sites.

Regardless, the conclusion we can draw is that the base present in the reference influences the signal through the alignment process leading to bias later in the analyses. We can see this by viewing the alignments produced for the inosine construct with another of the canonical bases present at the variable sites. The base frequency statistics for these alignments consistently report a majority of reads containing the base substituted for inosine in the reference (5.5).


70 bp
10 bp
20 bp
30 bp
40 bp
50 bp
[0 - 83819]
A
G C T C G A G C T A A G A G T C A C T G C A T G G C G A A G A C G T A C G A G A T C G T T C C A C T G T
[0 - 86688]
B
G C T C G A G C T A A G A G T C A A T G C A T G G C G A A G A A G T A C G A G A T C G T T C A A C T G T

aln_sort.bam Coverage

construct_Restriction_Cloning_ligation_product:32

Total count: 82632
A : 5633 (7%, 5233+, 400-)
C : 64986 (79%, 63361+, 1625-)
G : 5138 (6%, 4878+, 260-)
T : 6875 (8%, 6497+, 378-)
N : 0

DEL: 1187
INS: 536

aln_sort.bam Coverage

construct_Restriction_Cloning_ligation_product:32

Total count: 85998
A : 75553 (88%, 74775+, 778-)
C : 4301 (5%, 4002+, 299-)
G : 3252 (4%, 3053+, 199-)
T : 2892 (3%, 2692+, 200-)
N : 0

DEL: 690
INS: 819

Figure 5.5: Construct I aligned with different references. In the top half of this figure view an alignment of construct I with C in the reference (**A**). The construct is known to contain three inosines, but has been aligned with a reference containing cytosine at those positions. By looking at the base counts, we can tell one inosine position as indicated by the colored vertical bar. This position is also indicated by a black arrow. Here 79% of the bases are cytosines. At both of the other inosine positions 90% of the bases are cytosines, and so these do not have the same vertical colored bar due to overwhelming reference agreement. Notably, our analyses have indicated that cytosine's signature current signal is the one out of all four canonical bases that is the least similar to inosine. Below, we view an alignment of the same construct with a reference containing A (**B**). Here all three inosine positions have 88% adenine bases or more. This illustrates the problem that arises with accounting for the presence of inosine in a reference. The aligner finds the best fit between the given reads and the reference. Thus, the aligned reads will appear to be closer to the reference. When we compare construct I aligned with A in the reference to construct A, the two will appear indistinguishable since both alignments are reported to contain adenine at the three variable positions.

Since a surprisingly low level of ambiguity is present with regards to the three variable sites in construct I after two alignments, this could cause the Nanocompore comparison to be unable to distinguish between the variable positions in the modified and control constructs. The goal of the subsequent experiment will then be to circumvent this alignment step, to avoid introducing changes into data as a result of processing.

Chapter 6

Experiment 3

6.1 Rationale

In experiment 2, we found that we were unable to detect a difference in current signal when comparing construct I with any of the other constructs. From our analyses, it seems that substituting one of the canonical bases in the reference which we align construct I to introduces a bias in the data after alignment. This bias makes the current signal values for the two constructs we wish to compare too similar, and thus we are unable to distinguish between the two. In this experiment, we address this and avoid this bias by analyzing data belonging to construct I that has been aligned to a reference that instead contains N at the three inosine positions.

The approach in the third experiment can be conceived of as a hybrid of the two prior experiments. As we did in the first experiment, we focus on the raw current signal. And similarly to the second experiment, we are looking at data on a position-by-position basis. The central idea is to focus on the three variable positions and the surrounding nucleotides. Specifically, we limit the scope of our focus to the four bases immediately upstream and downstream of a central variable position. We do this since these are all the positions at which the central variable nucleotide has a direct influence on the measurements made by the nanopore. From raw basecalled data we then extract descriptive statistics, then move on to compare the five constructs against one another. Finally, we train a machine learning model to differentiate and correctly classify a 9-base segment based on construct. If this classification is successful, we can then infer the identity of the central variable position based solely on data gathered from these nine positions.

The bulk part of the processing prior to analysis is directed at identifying the set of reads that contain an alignment for all the three variable sites and the surrounding positions. In each read we check for alignments at each of the three variable sites. Data describing one of the variable sites and its immediate context is only used if a given read contains data corresponding to each of the nine positions in a segment of interest.

6.2 DNA constructs

The DNA constructs used for this experiment are the same ones that were used for the second experiment. Thus, we have five constructs with a variable segment followed by an invariant segment of 69 bases. Again, there are 3 variable positions within the invariant segment. Where we in the second experiment focused on the entirety of the invariant segment, here we further limit our focus to only the three variable sites within this segment along with the four nucleotides on either side of each of the variable sites.

6.3 Basecalling, alignment, and further pre-processing

We use the construct-specific alignments produced in the previous experiment to identify the five different constructs. For construct I, we use alignments specific to this construct but this time containing N in the reference at the three inosine positions to avoid any bias introduced in alignment. The alignments are used to inform a subsequent filtering step aimed to identify reads containing data which we are interested in using for comparison. After generating a subset of raw reads, only retaining those reads we have found to contain the relevant information, do we basecall this subset of reads for further analysis. However, the steps pertaining to basecalling and aligning proceed identically to prior descriptions. Since we are using the same data as in the previous experiment, no further quality controlling steps are taken.

We then prepare for downstream position-by-position analysis by converting alignment files (.bam) to .tsv files using sam2tsv [39]. In preparation for using the sam2tsv java archive based tool, we must also create a sequence dictionary which is required by sam2tsv. To do this we use another java archive based tool, picard [40]. The result of these steps is five separate files which correspond to each of the five constructs.

6.4 Filtering and further processing

Each of the resulting files from the previous step contains a magnitude of information, most of which is not relevant to the analysis we are carrying out. The first step in condensing data to contain only the relevant information is to identify which reads contain information for the three variable sites and their surrounding context. In order to not exclude valuable data, we consider each of the three variable sites and their respective surrounding bases individually.

For these purposes, we check for alignments at sequence indices relative to the reference specified to correspond to the segments that we are interested in. We determine for each read that is processed, whether it contains a complete alignment (i.e. a sequence match for all positions) to one of the three 9-nt segments of interest. If this is the case, the read ID of the read is appended to a list of IDs that are known to contain at least one segment we are interested in. The procedure is carried out once per construct for each of the three potential segments of interest.

Two sanity checks are then applied to the input. These respectively ensure that each segment is represented by exactly 9 reads with “M” as the CIGAR value and that no read appears twice in the output. Once the output is verified to contain the information we are interested in, another script writes all read IDs to a separate list. For each construct we then have three such lists corresponding to the three variable sites with their contexts.

In order to access raw data describing these reads, we subset the bulk .fast5 directory. Using ont-fast5-api, we create another directory containing only the reads as indicated by the list of read IDs. At this stage, we basecall only the subset of .fast5 files obtained in this step.

Next, we traverse the bulky format of the .fast5 files to extract the salient points of information. We use the importable get_fast5_file from the ont_fast5_api for the traversal, and to localize regions of interest we use a pairwise alignment method provided by Biopython [41]. In the process, we also use Nanopolish's pA normalization step to enable comparisons with similarly processed data. The points of information we gather here are: read id, base present in read, mean current signal, current signal standard deviation, dwell time, and basecall quality score. The specific raw signal measurements are again extracted by navigating something resembling the event data table discussed in Experiment 1.

6.5 Position-by-position comparison

When examining the differences between the same 9-nt segment across five constructs, we look at the differences present in the data individually at each position. The output we produced in the previous step contains data from all nine positions in bulk. To make subsequent analysis easier, we separate data from each of the nine positions in each of the three segments of interest into separate files. We compare constructs based on mean current signal, current signal standard deviation, dwell time, and basecall quality, then produce plots to visualize these comparisons. Furthermore, we look at the base count for each position of the three segments in construct I.

6.6 Construct classification

Following the visualization of data by position, we prepare the data for classification. For all nine positions in each of the three segments we include the following features: mean current signal, current standard deviation, base identity, dwell time, and basecall quality. A single column in the table indicates the construct a single row belongs to. We train a random forest classifier on this table providing all features as predictors, and the construct label as the target variable. We also test our classifier with a subset of these features, to evaluate the relative importance and information gain of the different features. This will tell us how well we are able to discriminate all five constructs from one another. We refer to this as the five-way classification task. For a more coarse classification on whether the construct is an inosine-containing construct or not, we add another label indicating whether a construct is construct I or any of the other constructs. We train another model on this classification as well. This we will refer to as the two-way classification task.

To robustly evaluate our classifier, we trained the classifier on the first segment and evaluated the performance of the model on the second segment. This will assess whether the model is able to recognize a pattern which is not merely specific to a single context. Since we are using features from all nine nucleotides in the segment, this is particularly important. We wish to minimimze the possibility of the classification task being too greatly influenced by learning features from the surrounding bases rather than the central variable position we wish to classify. To do this, we perform another round of classifications where the values for current measurements and dwell times have been normalized by the expected value for these variables. This is done by using 10% of gathered data to estimate a mean for both these values for each position in each segment. The mean is then subtracted from the values recorded for the other 90% of data. Data from construct I was overrepresented in terms of total samples available. Due to this class imbalance, we used the built in balance_classes function of the H2O Distributed Random Forest model [42]. This upsamples the under-represented classes.

6.7 Results

6.7.1 Position-by-position comparison results

In comparing the average pA normalized current signal from the first 9-nt segment of all five constructs, we observed that the signal measurements for the two nucleotides on the ends of either side of the 9-nt long segment almost did not differ at all. However, on the third nucleotide of the segment, we start to see the contours of diverging signal measurements. Continuing the trend, the difference of the five constructs is greater at the fourth nucleotide and even more so at the fifth and sixth nucleotides. On the seventh nucleotide the signal averages start to converge again, and on the last two nucleotides the signals are almost identical again (6.1).

The second and third segments exhibited the same trend with slight variation. In the second segment, though there is certainly a converging tendency also in the seventh nucleotide position (T), a greater level of variance persists between the five constructs (6.1). While in the third segment the plot visualizing these differences looks slightly different. This is due to a change in the reference sequence where it seems two nucleotides have switched positions. We observe an apparent shift in data describing constructs I and A, relative to the other three (6.1).

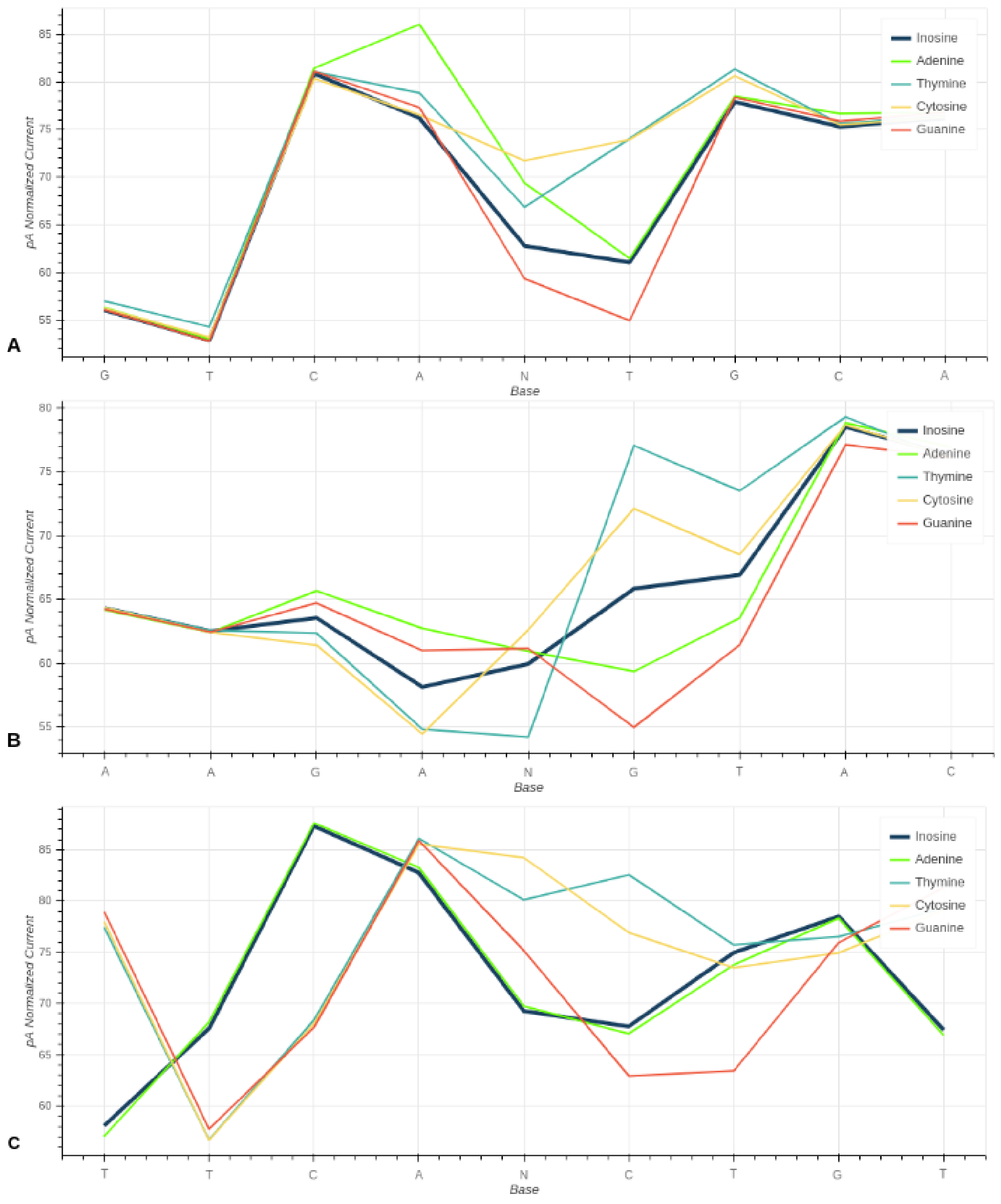


Figure 6.1: Average current per position for three segments for all five constructs. The general trend of these plots is most clearly illustrated by the comparisons of the first and the second segment (**A** and **B**). Here we can see very little difference in mean current for the first and the last bases in the segments. As we move toward the central variable position, the differences between the constructs increase, with the most pronounced difference being at the central base or one of its neighbors. In the plot made for the third segment, we also see the same trend (**C**). However, the curves for construct I and construct A seem to have been offset by a single base relative to the other three constructs. We also see almost identical curves for these two constructs.

Comparisons of standard deviation in the signals also agree with the results of comparing signal averages. We see a definite agreement across all constructs in the first two nucleotides, a tendency towards greater difference on the third nucleotide, followed by three nucleotides of a pronounced difference in standard deviations. At the seventh nucleotide, we again see a tendency towards greater similarity across constructs, and then less difference at the last two positions of the 9-nt segment (6.2). Though the general trend is the same for all three segments, we disregard the third segment as it was affected by the reference shift.

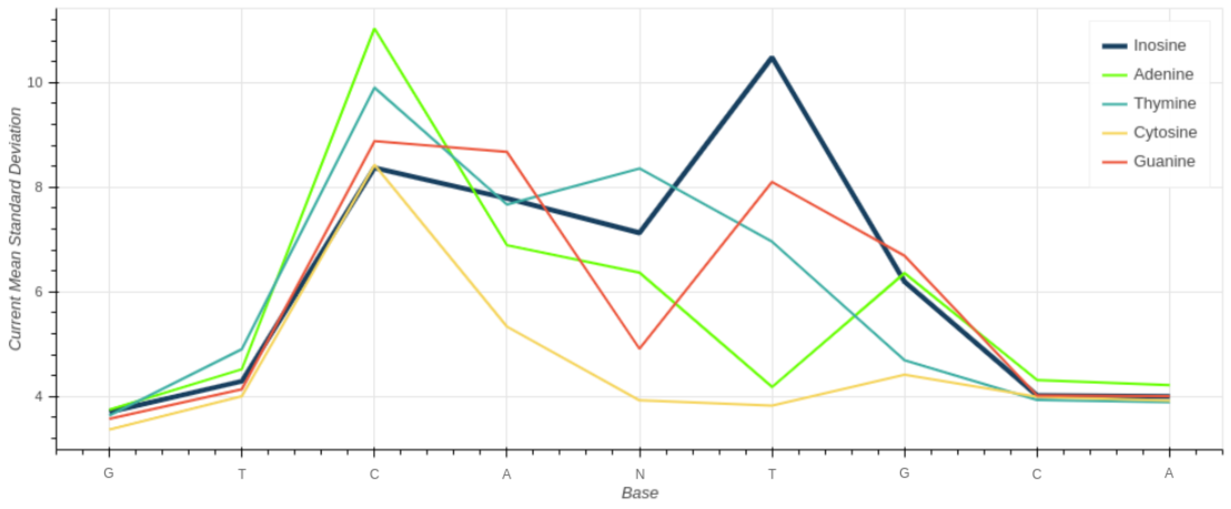


Figure 6.2: Standard deviation of mean current per position of the first segment. Here we also see a clear tendency of a greater level of variance at and around the central variable position.

Mean dwell time comparisons show the same trend, but to a lesser extent. Here, the biggest differences are limited to the central variable nucleotide with slight difference in its immediate neighbor nucleotides on either side (6.3). While differences in dwell time standard deviations are present across all positions with the exception of the first and last positions in the segment.

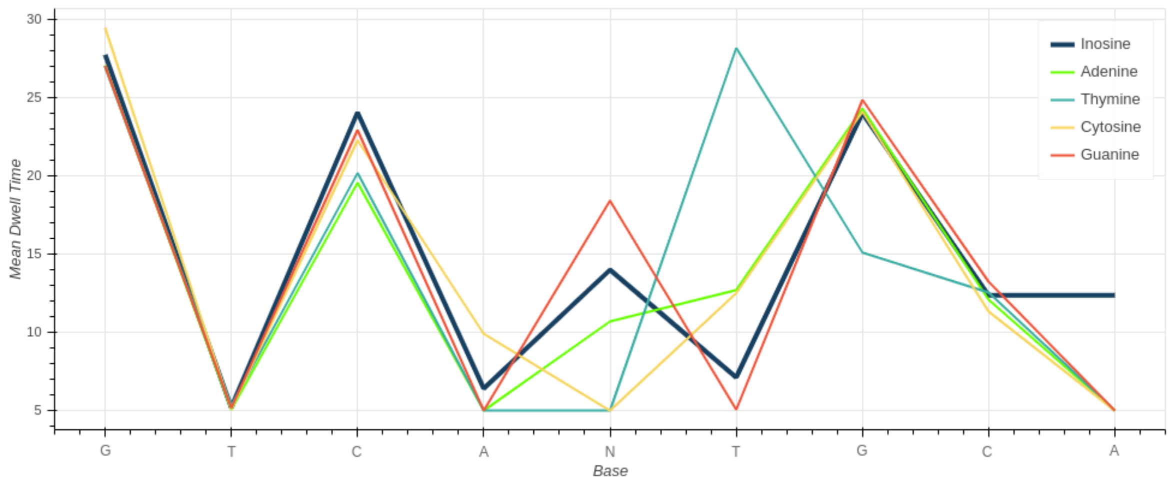
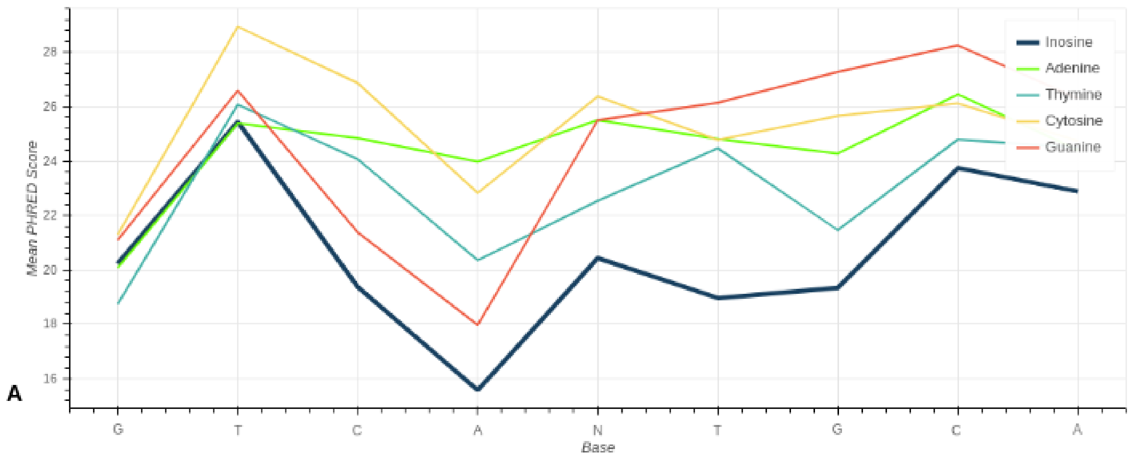


Figure 6.3: Mean dwell time per position for the first segment for all five constructs. We see a similar trend to previous plots of increased variance toward the variable base.

For basecall quality, we again observe a trend congruent with the mean current signal comparisons. More similarity is present at the two first and last nucleotides, with a more pronounced difference being apparent in all five central positions of the segment. In the first segment construct I has a worse score than all other constructs for almost all bases (6.4). We do not observe the same quality scores for this construct in the second segment (6.4).



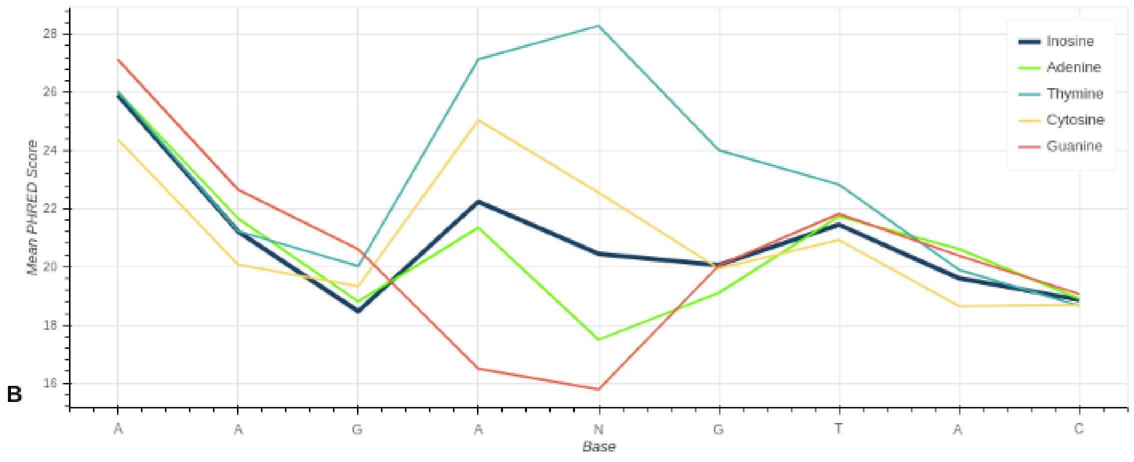


Figure 6.4: Basecall quality score per position for the first and second segment for all constructs. In the first segment, we can see that construct I has a lower average mean bascall quality for almost every position (**A**). However, in the second segment this trend cannot be seen (**B**). Similar for both plots is the relatively greater degree of agreement toward the start and the end of the segments. It isn't as clearly pronounced as in current comparisons but still present. It is particularly clear toward the end of the second segment.

The base identity for all positions except the central variable nucleotide are in agreement with the base indicated by the reference. In the first segment of construct I, the central inosine was in the majority of reads basecalled as G, but also often as C (6.5). Looking at the base count for the second segment, inosine was identified as each of the four canonical bases by even proportions (6.5). In the third segment, the central inosine was basecalled as adenine in the majority of reads (6.5). It should be noted that there is a small frequency of mismatches in the four positions before and after the central variable nucleotide. This is an almost insignificant fraction, but it is slightly larger for the four bases after the variable position.

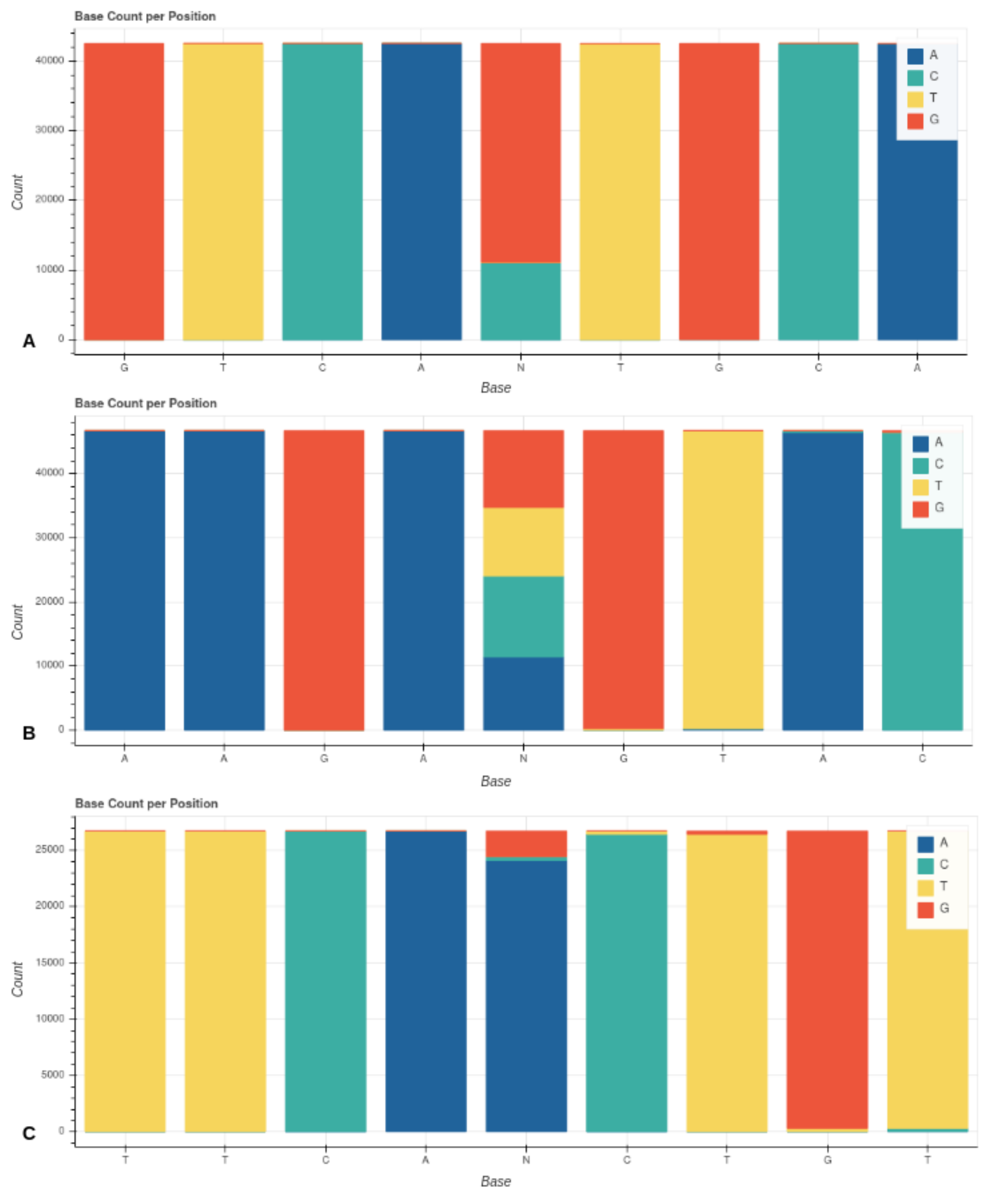


Figure 6.5: Base count per position for all three segments of construct I. In the first segment we observe that the central inosine is most frequently basecalled as guanine, but also quite frequently as cytosine (**A**). However, in the second segment inosine is almost evenly basecalled as all four canonical bases (**B**). Here we also see a slightly greater fraction of misclassifications occurring after the variable base. This tendency is even more pronounced in the third segment (**C**). In this case, the inosine was most frequently basecalled as adenine.

6.7.2 Construct classification results

6.7.2.1 Training and testing on the same segment

We trained the first random forest classifier on the first segment, using all five extracted features. The model was specified to generate 20 trees, with a max depth of 7. After training on 80% of the data, we then validated on 20% of the data from the same segment. Evaluating the classification of inosine presence, we were able to correctly classify 62367 out of 68304 samples (6.1). This is a success-rate of 91.31%. The Matthews Correlation Coefficient (MCC) for this classification was 0.83. We also performed a five-way classification with this training and testing approach. Using the same model specifications, in this case we were able to classify the correct construct for 125798 out of 170551 samples (6.2). This yields a success-rate of 73.76% and an MCC of 0.67.

	Predicted class		
True class		False	True
	False	32057	2140
	True	3797	30310

Table 6.1: Confusion matrix for two-way classification after training and testing on the same segment. The rows indicate the true class of a sample, while the columns indicate the predicted class. This classification achieved an accuracy of 91.31% and a MCC of 0.83.

	Predicted class					
True class		construct I	construct A	construct T	construct C	construct G
	construct I	32402	0	0	1680	26
	construct A	823	33286	0	0	2
	construct T	60	0	34015	0	0
	construct C	9097	0	0	25017	0
	construct G	33065	0	0	0	1078

Table 6.2: Confusion matrix for five-way classification after training and testing on the same segment. This classification achieved an accuracy of 73.76% and an MCC of 0.67.

6.7.2.2 Training and testing on different segments

When evaluating the performance of the same model trained on data from the first segment, but tested on data from the second segment, we also started by evaluating the performance on the two-way classification task. Here, we use the average current signal, dwell time, and basecall quality score for each of the 9 positions as features for classification. We were able to correctly classify 62332 out of 68183 samples, yielding an accuracy of 91.42% and MCC of 0.83 (6.3). Meanwhile, on the five-way classification task we were able to correctly classify 140625 out of 170405, giving an accuracy of 82.52% and MCC of 0.78 (6.4).

	Predicted class		
True class		False	True
	False	32384	1691
	True	4160	29948

Table 6.3: Confusion matrix for two-way classification after training on the first segment and testing on the second segment. This classification achieved an accuracy of 91.42% and an MCC of 0.83.

	Predicted class					
True class		construct I	construct A	construct T	construct C	construct G
	construct I	31633	0	4	2215	256
	construct A	1131	32974	5	1	1
	construct T	203	0	33838	7	0
	construct C	1275	0	0	32783	0
	construct G	24680	1	0	1	9397

Table 6.4: Confusion matrix for five-way classification after training on the first segment and testing on the second segment. This classification achieved an accuracy of 82.52% and an MCC of 0.78.

6.7.2.3 Training and testing on different segments with normalized data

We also applied this same training and testing framework of training on the first segment and testing on the second segment to normalized data. Also here, are we using three features: average current signal, dwell time, and basecall quality. The former two have been normalized. Starting with a focus on the two-way classification, we are able to classify 61315 out of 61414 samples correctly (6.5). This gives an accuracy of 99.84% and an MCC of 0.99. For the five-way classification, we were able to classify 143532 out of 153628 correctly, an accuracy of 93.42% and MCC of 0.92 (6.6).

		Predicted class	
		False	True
True class	False	30640	46
	True	53	30675

Table 6.5: Confusion matrix for two-way classification after normalizing data, training on the first segment, and testing on the second segment. This classification achieved an accuracy of 99.84% and an MCC of 0.99.

		Predicted class				
		construct I	construct A	construct T	construct C	construct G
True class	construct I	30719	0	1	5	4
	construct A	1221	26637	964	734	1185
	construct T	1235	70	27728	1325	351
	construct C	996	16	217	29384	106
	construct G	1425	2	42	197	29064

Table 6.6: Confusion matrix for five-way classification after normalizing data, training on the first segment, and testing on the second segment. This classification achieved an accuracy of 93.42% and an MCC of 0.92.

6.8 Discussion

We observe a definite trend of increasing variation seen at the central variable position and the surrounding nucleotides. This is exhibited across the comparisons for almost all values for the three segments, and suggests that the different constructs exhibit considerable variation that can be helpful in classifying the constructs. While this difference is consistent for all comparisons, the tendency of this difference is not consistent across the segments. That is, we don't see the influence of inosine having a similar impact on the measurements across different contexts. This may not be particularly surprising when it comes to the mean current signal since the kmers influenced by the variable base differ, but it is somewhat more surprising when it comes to basecall quality. We would expect construct I to on average have a lower basecall quality, since it contains a base that doesn't conform to the nucleotides expected by the basecallers. This is the case for the first and the third segments, where we see a considerably lower basecall quality for the bases around the variable nucleotide. In the second segment we don't see this tendency, with both construct G and construct T exhibiting lower basecall quality for the same positions.

Nevertheless, when it comes to classification we need to ensure that we aren't introducing sequence-dependent bias of the contextual underlying data distributions. This is the reason for training on one segment and evaluating on another, since these differ in their surrounding context; it is also the reason for normalizing data which can remove the influence of the surrounding context. After doing both of these we are left with a classifier that is equipped to differentiate between all five constructs. If we examine the confusion matrix (6.3), the major difficulty for the five-way classification prior to normalization is the correct classification of construct G. A large portion of these samples are classified as construct I. Given the observations with regards to nanopore's processing of inosine in contrast to the canonical bases, this supports the finding that inosine has a profile most similar to adenine and guanine. After normalizing, the model gets a considerable boost in performance for both classification tasks. But for the five-way classification the errors are more evenly spread across all constructs (6.4). In fact, after normalizing it seems that construct A presents the most confusing samples for our classifier.

However, we are more interested in the confusion with regards to adenine in a two-way classification. This is because in a natural setting inosines occur in DNA through the deamination of adenine. As indicated by the confusion matrix from the 5-way classification, there is little confusion in the classification of construct I. There is still some confusion in the classification of construct A, where a considerable portion of samples are classified as construct I. This is likely due to the over-representation of construct I in the data.

Looking at the confusion matrix for the 2-way classification however, we see there are very few misclassified samples (6.5). This tells us that when the objective is to detect the presence of inosine, this framework for classification is sufficiently discriminating.

Chapter 7

Discussion

7.1 Discussion

In this thesis, we have introduced the experimental relevance of detecting Inosine in DNA sequence data. We have generally discussed methods for detecting modifications, the advantages nanopore sequencing offers in this task over previous generations of sequencing, and the current state of the modification detection enterprise. This formulated the motivating impetus of the study. We then proposed proceedings for realizing the potential application of nanopore sequencing in detecting inosine and introduced briefly the technical aspects of the data we would be working with.

We then presented a framework for identifying the presence of inosine within a specific window of DNA sequence data. Following our proposal, we have surveyed the profile of inosine as it is processed by nanopore. We attempted to run pairwise comparisons for our constructs using Nanocompore on DNA data. Not being able to distinguish constructs using this approach, we do position-by-position comparisons of constructs to see the difference in signal measurements and other sequence data without the influence of other analytical tools. Observing this difference, we define a random forest classifier which successfully distinguishes between constructs containing inosines and those that don't. Though not fully integrated into a pipeline, the potential for applying a machine learning classifier to the task of inosine detection is demonstrated. While detection of other modifications by similar approaches have been demonstrated previously, this is the first work addressing the detection of inosine in DNA. Specifically, the proposed framework shows utility in a natural setting. This is a scenario in which adenine has been modified to an inosine — a classification task which our model performs well on.

One limitation of the current framework is that it has only been tested in a setting where the relevant sites for modification are known. In future work, an aim would be to further develop the approach to become a self-contained pipeline. Given raw data from a sequence containing either adenine or inosine and a known reference, the model, having been trained on normalized data containing both of these bases, could then classify whether inosine is present in a construct or not. Potential modifications could be localized by surveying the construct at relevant positions and using a sliding window approach to evaluate the signal data from the local region. Continuing the development of this tool we would then focus efforts on developing this functionality.

Further, it would be interesting to generalize the classification framework and see its performance on other modifications. Several modifications are known to exhibit a similar variance as we have observed for Inosine, where measurements show deviation from model values at and around a modified base. We would be able to use the same framework to detect such modifications. This could be an avenue to pursue in an effort to extend this framework towards a more general modification detection tool for nanopore data.

In developing a general modification detection tool, we would also need to put more focus on the performance of the framework relative to other existing frameworks. This would involve benchmarking accuracy and efficiency against other established tools for detection modification.

7.2 Conclusion

In this thesis, we set out to classify the presence of inosine in a given sequence. Firstly, we looked at the possibility of distinguishing the pure signal of inosine from that of the four canonical DNA nucleotides. Furthermore, we attempted to train a classifier to distinguish between the pure signals of these five nucleotides based only on current signal mean and standard deviation. We moved on to use the sample comparison functionality of Nanocompore, a tool designed for RNA modification detection, to distinguish between one modified and four non-modified DNA sequences. Finally, we isolated the relevant data from our modified and non-modified DNA constructs. We used these to train a classifier on distinguishing the five constructs. In addition, we trained another classifier to address a two-class problem of distinguishing modified and non-modified sequences.

In the initial phase of this work, we present a description of inosine's characteristic current signal profile. We also demonstrate that the pure signal signature of inosine is indeed sufficiently distinct from that of the four canonical DNA nucleotides. However, it is not so distinct that one can readily distinguish an inosine homopolymer from the four canonical homopolymers based only on current signal mean and standard deviation values.

Despite prolonged efforts, we were not able to demonstrate modification detection in DNA sequence data with Nanocompore. Where we expected to be able to see a considerable difference in current signal measurements when comparing modified and non-modified samples, no such difference could be observed.

In the final phase of the project, we construct our own framework for sample comparison and are then able to observe the difference we expected when comparing modified and non-modified samples. Based on these observations we construct a classifier which with high accuracy distinguishes constructs containing inosine and those that do not. We demonstrate, as the first work to do so, the successful classification of inosine presence in nanopore sequence data from DNA.

Further work is required to integrate our proposed approach into a stand-alone toolkit for inosine detection. In doing so, there are several interesting avenues for expanding functionality that can be explored. Such avenues could include generalizing our approach to other modifications or trying to detect modifications without training for one particular modification. This would require generating new datasets and further refinement of the overall approach. Thus, we would continue to expand upon the promising results presented in this work.

Bibliography

[1] Raiber Eun-Ang, Robyn Hardisty, van Delft Pieter, and Balasubramanian Shankar. Mapping and elucidating the function of modified bases in DNA. *Nature Reviews. Chemistry*, 1(9), September 2017. doi: http://dx.doi.org.pva.uib.no/10.1038/s41570-017-0069. URL `http://www.proquest.com/docview/2389664451/abstract/136BB822819A4F34PQ/1`. Place: London, United States Publisher: Nature Publishing Group.

[2] William Stephenson, Roham Razaghi, Steven Busan, Kevin M. Weeks, Winston Timp, and Peter Smibert. Direct detection of RNA modifications and structure using single molecule nanopore sequencing. *bioRxiv*, page 2020.05.31.126763, June 2020. doi: 10.1101/2020.05.31.126763. URL `https://www.biorxiv.org/content/10.1101/2020.05.31.126763v1`. Publisher: Cold Spring Harbor Laboratory Section: New Results.

[3] Qian Liu, Li Fang, Guoliang Yu, Depeng Wang, Chuan-Le Xiao, and Kai Wang. Detection of DNA base modifications by deep recurrent neural network on Oxford Nanopore sequencing data. *Nature Communications*, 10(1):2449, June 2019. ISSN 2041-1723. doi: 10.1038/s41467-019-10168-2. URL `https://www.nature.com/articles/s41467-019-10168-2`. Number: 1 Publisher: Nature Publishing Group.

[4] Shanika L. Amarasinghe, Shian Su, Xueyi Dong, Luke Zappia, Matthew E. Ritchie, and Quentin Gouil. Opportunities and challenges in long-read sequencing data analysis. *Genome Biology*, 21(1):30, February 2020. ISSN 1474-760X. doi: 10.1186/s13059-020-1935-5. URL `https://doi.org/10.1186/s13059-020-1935-5`.

[5] Miten Jain, Hugh E. Olsen, Benedict Paten, and Mark Akeson. The Oxford Nanopore MinION: delivery of nanopore sequencing to the genomics community. *Genome Biology*,

17(1):239, November 2016. ISSN 1474-760X. doi: 10.1186/s13059-016-1103-0. URL `https://doi.org/10.1186/s13059-016-1103-0`.

[6] Aaron M. Fleming, Nicole J. Mathewson, and Cynthia J. Burrows. Nanopore dwell time analysis permits sequencing and conformational assignment of pseudouridine in SARS-CoV-2. *bioRxiv*, 2021. doi: 10.1101/2021.05.10.443494. URL `https://www.biorxiv.org/content/early/2021/05/10/2021.05.10.443494`. Publisher: Cold Spring Harbor Laboratory _eprint: https://www.biorxiv.org/content/early/2021/05/10/2021.05.10.443494.full.pdf.

[7] Heng Li. Minimap2: pairwise alignment for nucleotide sequences. *Bioinformatics*, 34 (18):3094–3100, September 2018. ISSN 1367-4803. doi: 10.1093/bioinformatics/bty191. URL `https://doi.org/10.1093/bioinformatics/bty191`.

[8] Michael Roberts, Wayne Hayes, Brian R. Hunt, Stephen M. Mount, and James A. Yorke. Reducing storage requirements for biological sequence comparison. *Bioinformatics*, 20 (18):3363–3369, December 2004. ISSN 1367-4803. doi: 10.1093/bioinformatics/bth408. URL `https://doi.org/10.1093/bioinformatics/bth408`.

[9] Nicholas J. Loman, Joshua Quick, and Jared T. Simpson. A complete bacterial genome assembled de novo using only nanopore sequencing data. *Nature Methods*, 12(8):733–735, August 2015. ISSN 1548-7105. doi: 10.1038/nmeth.3444. URL `https://www.nature.com/articles/nmeth.3444`. Number: 8 Publisher: Nature Publishing Group.

[10] Ingrun Alseth, Bjørn Dalhus, and Magnar Bjørås. Inosine in DNA and RNA. *Current Opinion in Genetics & Development*, 26:116–123, June 2014. ISSN 0959-437X. doi: 10.1016/j.gde.2014.07.008. URL `https://www.sciencedirect.com/science/article/pii/S0959437X14000811`.

[11] M S Paul and B L Bass. Inosine exists in mRNA at tissue-specific levels and is most abundant in brain mRNA. *The EMBO Journal*, 17(4):1120–1127, February 1998. ISSN 0261-4189. doi: 10.1093/emboj/17.4.1120. URL `https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1170460/`.

[12] John S. Mattick, Paulo P. Amaral, Marcel E. Dinger, Tim R. Mercer, and Mark F. Mehler. RNA regulation of epigenetic processes. *BioEssays*, 31(1):51–59, 2009. ISSN 1521-1878. doi: 10.1002/bies.080099. URL

https://onlinelibrary.wiley.com/doi/abs/10.1002/bies.080099. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/bies.080099.

[13] Brenda L. Bass. RNA Editing by Adenosine Deaminases That Act on RNA. *Annual review of biochemistry*, 71:817–846, 2002. ISSN 0066-4154. doi: 10.1146/annurev.biochem.71.110601.135501. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1823043/.

[14] Bernd Sommer, Martin Köhler, Rolf Sprengel, and Peter H. Seeburg. RNA editing in brain controls a determinant of ion flow in glutamate-gated channels. *Cell*, 67(1):11–19, October 1991. ISSN 0092-8674. doi: 10.1016/0092-8674(91)90568-J. URL https://www.sciencedirect.com/science/article/pii/009286749190568J.

[15] Louis Valente and Kazuko Nishikura. ADAR Gene Family and A-to-I RNA Editing: Diverse Roles in Posttranscriptional Gene Regulation. In *Progress in Nucleic Acid Research and Molecular Biology*, volume 79, pages 299–338. Academic Press, January 2005. doi: 10.1016/S0079-6603(04)79006-6. URL https://www.sciencedirect.com/science/article/pii/S0079660304790066.

[16] Miyoko Higuchi, Stefan Maas, Frank N. Single, Jochen Hartner, Andrei Rozov, Nail Burnashev, Dirk Feldmeyer, Rolf Sprengel, and Peter H. Seeburg. Point mutation in an AMPA receptor gene rescues lethality in mice deficient in the RNA-editing enzyme ADAR2. *Nature*, 406(6791):78–81, July 2000. ISSN 00280836. doi: http://dx.doi.org.pva.uib.no/10.1038/35017558. URL http://www.proquest.com/docview/204488012/abstract/8914543D14D64A22PQ/1. Num Pages: 4 Place: London, United States Publisher: Nature Publishing Group.

[17] Q. Wang, J. Khillan, P. Gadue, and K. Nishikura. Requirement of the RNA Editing Deaminase ADAR1 Gene for Embryonic Erythropoiesis. *Science*, 290(5497):1765–1768, 2000. ISSN 0036-8075. URL https://www.jstor.org/stable/3081770. Publisher: American Association for the Advancement of Science.

[18] Sivan Osenberg, Nurit Paz Yaacov, Michal Safran, Sharon Moshkovitz, Ronit Shtrichman, Ofra Sherf, Jasmine Jacob-Hirsch, Gilmor Keshet, Ninette Amariglio, Joseph Itskovitz-Eldor, and Gideon Rechavi. Alu Sequences in Undifferentiated Human Embryonic Stem Cells Display High Levels of A-to-I RNA Editing. *PLoS ONE*, 5(6): e11173, June 2010. ISSN 1932-6203. doi: 10.1371/journal.pone.0011173. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2888580/.

[19] Don E Farthing, Christine A Farthing, and Lei Xi. Inosine and hypoxanthine as novel biomarkers for cardiac ischemia: From bench to point-of-care. *Experimental Biology and Medicine*, 240(6):821–831, June 2015. ISSN 1535-3702. doi: 10.1177/1535370215584931. URL `https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4935215/`.

[20] Zachary E. Ferris, Qiushi Li, and Markus W. Germann. Substituting Inosine for Guanosine in DNA: Structural and Dynamic Consequences. *Natural Product Communications*, 14(5):1934578X19850032, May 2019. ISSN 1934-578X. doi: 10.1177/1934578X19850032. URL `https://doi.org/10.1177/1934578X19850032`. Publisher: SAGE Publications Inc.

[21] Jacob Schreiber, Zachary L. Wescoe, Robin Abu-Shumays, John T. Vivian, Baldandorj Baatar, Kevin Karplus, and Mark Akeson. Error rates for nanopore discrimination among cytosine, methylcytosine, and hydroxymethylcytosine along individual DNA strands. *Proceedings of the National Academy of Sciences*, 110(47):18910–18915, November 2013. ISSN 0027-8424, 1091-6490. doi: 10.1073/pnas.1310615110. URL `https://www.pnas.org/content/110/47/18910`. Publisher: National Academy of Sciences Section: Biological Sciences.

[22] Ina Anreiter, Quoseena Mir, Jared T. Simpson, Sarath C. Janga, and Matthias Soller. New Twists in Detecting mRNA Modification Dynamics. *Trends in Biotechnology*, 39(1):72–89, January 2021. ISSN 0167-7799, 1879-3096. doi: 10.1016/j.tibtech.2020.06.002. URL `https://www.cell.com/trends/biotechnology/abstract/S0167-7799(20)30166-9`. Publisher: Elsevier.

[23] Pierre B. Cattenoz, Ryan J. Taft, Eric Westhof, and John S. Mattick. Transcriptome-wide identification of A $>$ I RNA editing sites by inosine specific cleavage. *RNA*, 19(2):257–270, February 2013. ISSN 1355-8382. doi: 10.1261/rna.036202.112. URL `https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3543087/`.

[24] Tsutomu Suzuki, Hiroki Ueda, Shunpei Okada, and Masayuki Sakurai. Transcriptome-wide identification of adenosine-to-inosine editing using the ICE-seq method. *Nature Protocols*, 10(5):715–732, May 2015. ISSN 17542189. doi: http://dx.doi.org.pva.uib.no/10.1038/nprot.2015.037. URL `http://www.proquest.com/docview/1672119395/abstract/461748E750F7433BPQ/1`. Num Pages: 18 Place: London, United States Publisher: Nature Publishing Group.

[25] Schraga Schwartz and Yuri Motorin. Next-generation sequencing technologies for detection of modified nucleotides in RNAs. *RNA Biology*, 14(9):1124–1137, September 2017. ISSN 1547-6286. doi: 10.1080/15476286.2016.1251543. URL `https://doi.org/10.1080/15476286.2016.1251543`. Publisher: Taylor & Francis _eprint: https://doi.org/10.1080/15476286.2016.1251543.

[26] Masayuki Sakurai, Takanori Yano, Hitomi Kawabata, Hiroki Ueda, and Tsutomu Suzuki. Inosine cyanoethylation identifies A-to-I RNA editing sites in the human transcriptome. *Nature Chemical Biology*, 6(10):733–40, October 2010. ISSN 15524450. doi: http://dx.doi.org.pva.uib.no/10.1038/nchembio.434. URL `http://www.proquest.com/docview/751845195/abstract/7613904F23024F2FPQ/1`. Num Pages: 8 Place: Cambridge, United States Publisher: Nature Publishing Group.

[27] Peng Ni, Neng Huang, Zhi Zhang, De-Peng Wang, Fan Liang, Yu Miao, Chuan-Le Xiao, Feng Luo, and Jianxin Wang. DeepSignal: detecting DNA methylation state from Nanopore sequencing reads using deep-learning. *Bioinformatics*, 35(22):4586–4595, November 2019. ISSN 1367-4803. doi: 10.1093/bioinformatics/btz276. URL `https://doi.org/10.1093/bioinformatics/btz276`.

[28] Alexa B. R. McIntyre, Noah Alexander, Kirill Grigorev, Daniela Bezdan, Heike Sichtig, Charles Y. Chiu, and Christopher E. Mason. Single-molecule sequencing detection of N 6-methyladenine in microbial reference materials. *Nature Communications*, 10 (1):579, February 2019. ISSN 2041-1723. doi: 10.1038/s41467-019-08289-9. URL `https://www.nature.com/articles/s41467-019-08289-9`. Number: 1 Publisher: Nature Publishing Group.

[29] Marcus Stoiber, Joshua Quick, Rob Egan, Ji Eun Lee, Susan Celniker, Robert K. Neely, Nicholas Loman, Len A. Pennacchio, and James Brown. De novo Identification of DNA Modifications Enabled by Genome-Guided Nanopore Signal Processing. *bioRxiv*, page 094672, April 2017. doi: 10.1101/094672. URL `https://www.biorxiv.org/content/10.1101/094672v2`. Publisher: Cold Spring Harbor Laboratory Section: New Results.

[30] Qian Liu, Daniela C. Georgieva, Dieter Egli, and Kai Wang. NanoMod: a computational tool to detect DNA modifications using Nanopore long-read sequencing data. *BMC Genomics*, 20(1):78, February 2019. ISSN 1471-2164. doi: 10.1186/s12864-018-5372-8. URL `https://doi.org/10.1186/s12864-018-5372-8`.

[31] Soundhar Ramasamy, Vinodh J. Sahayasheela, Zutao Yu, Takuya Hidaka, Li Cai, Hiroshi Sugiyama, and Ganesh N. Pandian. Chemical probe-based Nanopore Sequencing to Selectively Assess the RNA modifications. *bioRxiv*, page 2020.05.19.105338, May 2020. doi: 10.1101/2020.05.19.105338. URL `https://www.biorxiv.org/content/10.1101/2020.05.19.105338v2`. Publisher: Cold Spring Harbor Laboratory Section: New Results.

[32] Rachael E. Workman, Alison D. Tang, Paul S. Tang, Miten Jain, John R. Tyson, Philip C. Zuzarte, Timothy Gilpatrick, Roham Razaghi, Joshua Quick, Norah Sadowski, Nadine Holmes, Jaqueline Goes de Jesus, Karen L. Jones, Terrance P. Snutch, Nicholas Loman, Benedict Paten, Matthew Loose, Jared T. Simpson, Hugh E. Olsen, Angela N. Brooks, Mark Akeson, and Winston Timp. Nanopore native RNA sequencing of a human poly(A) transcriptome. *bioRxiv*, page 459529, November 2018. doi: 10.1101/459529. URL `https://www.biorxiv.org/content/10.1101/459529v1`. Publisher: Cold Spring Harbor Laboratory Section: New Results.

[33] Adrien Leger, Paulo P. Amaral, Luca Pandolfini, Charlotte Capitanchik, Federica Capraro, Isaia Barbieri, Valentina Migliori, Nicholas M. Luscombe, Anton J Enright, Konstantinos Tzelepis, Jernej Ule, Tomas Fitzgerald, Ewan Birney, Tommaso Leonardi, and Tony Kouzarides. RNA modifications detection by comparative Nanopore direct RNA sequencing. *bioRxiv*, 2019. doi: 10.1101/843136. URL `https://www.biorxiv.org/content/early/2019/11/15/843136`. Publisher: Cold Spring Harbor Laboratory _eprint: https://www.biorxiv.org/content/early/2019/11/15/843136.full.pdf.

[34] Wouter De Coster, Svenn D'Hert, Darrin T. Schultz, Marc Cruts, and Christine Van Broeckhoven. NanoPack: visualizing and processing long-read sequencing data. *Bioinformatics (Oxford, England)*, 34(15):2666–2669, August 2018. ISSN 1367-4811. doi: 10.1093/bioinformatics/bty149.

[35] Heng Li, Bob Handsaker, Alec Wysoker, Tim Fennell, Jue Ruan, Nils Homer, Gabor Marth, Goncalo Abecasis, and Richard Durbin. The Sequence Alignment/Map format and SAMtools. *Bioinformatics*, 25(16):2078–2079, August 2009. ISSN 1367-4803. doi: 10.1093/bioinformatics/btp352. URL `https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2723002/`.

[36] Ryan R. Wick, Louise M. Judd, and Kathryn E. Holt. Deepbinner: Demultiplexing barcoded Oxford Nanopore reads with deep convolutional neural networks. *PLOS Com-*

putational Biology, 14(11):e1006583, November 2018. ISSN 1553-7358. doi: 10.1371/journal.pcbi.1006583. URL `https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1006583`. Publisher: Public Library of Science.

[37] Ryan R. Wick, Louise M. Judd, Claire L. Gorrie, and Kathryn E.YR 2017 Holt. Completing bacterial genome assemblies with multiplex MinION sequencing. *Microbial Genomics*, 3(10):e000132. ISSN 2057-5858,. doi: 10.1099/mgen.0.000132. URL `https://www.microbiologyresearch.org/content/journal/mgen/10.1099/mgen.0.000132`. Publisher: Microbiology Society,.

[38] Francois Chollet et al. Keras, 2015. URL `https://github.com/fchollet/keras`.

[39] Pierre Lindenbaum. JVarkit: java-based utilities for Bioinformatics. May 2015. doi: 10.6084/m9.figshare.1425030.v1. URL `/articles/journal_contribution/JVarkit_java_based_utilities_for_Bioinformatics/1425030/1`. Publisher: figshare.

[40] Broad Institute. Picard tools. `http://broadinstitute.github.io/picard/`, (Accessed: 2018/02/21; version 2.17.8).

[41] Peter J. A. Cock, Tiago Antao, Jeffrey T. Chang, Brad A. Chapman, Cymon J. Cox, Andrew Dalke, Iddo Friedberg, Thomas Hamelryck, Frank Kauff, Bartek Wilczynski, and Michiel J. L. de Hoon. Biopython: freely available Python tools for computational molecular biology and bioinformatics. *Bioinformatics*, 25(11):1422–1423, June 2009. ISSN 1367-4803. doi: 10.1093/bioinformatics/btp163. URL `https://doi.org/10.1093/bioinformatics/btp163`.

[42] H2O.ai. H2o, August 2021. URL `https://github.com/h2oai/h2o-3.git`. Version: 3.32.1.1.

Appendix A

A.1 Source code

The code used to produce the data presented here can be found at this Github repo (https://github.com/Stautis/detecting-inosine-master-thesis).

A.2 Data availibility

Data used for analysis can be made available upon request.