English title

Exploring Oxford Nanopore Technologies Long-Read Sequencing Strategies and Haplotype Phasing Methods for Detecting the Length of HTT and FMR1 Tandem Repeat Expansions

Dansk titel

Udforskning af Oxford Nanopore Technologies Long-Read Sekventerings Strategier og Haplotype Fasnings Metoder til Bestemmelse af Tandem Repeat Ekspansion Længder for HTT og FMR1.

Tine Sørensen, February 2021

MSc of Computational Biomedicine (Biomedicinsk Informatik), Faculty of Science, University of Southern Denmark, Odense.

Department of Clinical Genetics, Odense University Hospital, Odense

Supervisors:

Assoc. Prof. Martin Jacob Larsen

Assoc. Prof. Lars Andersen

Assoc. Prof. Charlotte Brasch Andersen

Prof. Kedar Nath Natarajan

Number of characters in the dissertation: 112.754

Abstract

Short tandem repeats are prone to dynamic mutations which at times results in pathogenic expansions. The clinical tests for the diseases caused by repeat expansions focus on one disease per protocol, making it impossible to screen for multiple diseases at once. A panel consisting of 27 repeat diseases has been constructed. The panel has been tested in combination with adaptive targeted sequencing using long-read sequencing technology from Oxford Nanopore Technologies combined with the software ReadFish, as a greater depth is hypothesized to aid in the estimation of repeat lengths as well as in haplotype phasing the DNA-reads. Haplotype phasing is performed in order to more precisely separate the repeat length estimations from the two alleles. The adaptive targeted sequencing gave an enrichment with an average fold-change between 2.34 and 3.92 for the target panel, which falls within the expected range of 2.7-5.4x.

Four programs have been used for estimating the repeat lengths in HTT and FMR1: RepeatHMM, STRique, Tandem-Genotypes, and TRiCoLOR. Of these TRiCoLOR has the best performance and estimates close to correct repeat lengths. TRiCoLOR takes haplotype phased input files, and two methods for phasing has been used: LongShot and an in-house method created for this project. The different strategies in phasing use extraction of genotype information at the SNP position, which are either taken from less error-prone short-read sequencing (Illumina) or error-prone long-read sequencing (Oxford Nanopore). No significant difference was found in the results of the two phasing methods, and the phasing did not aid in separating the alleles with repeat lengths closer to equal.

Targeted sequencing is not ready for clinical use due to the sequencing errors and fluctuations in repeat length estimation that could have diagnostic consequences. Further improvements in sequencing, basecalling, and repeat estimation are necessary.

Resume

Short tandem repeats (STR) er tilbøjelige til at undergå dynamiske mutationer, hvilket kan resultere i patogene ekspansioner. De nuværende diagnostiske test for sygdomme forårsaget af ekspansioner fokuserer på én sygdom af gangen, hvilket gør det umuligt at screene for multiple sygdomme på én gang. Et panel af 27 STR sygdomme er blevet udviklet. Dette panel er blevet testet i kombination med adaptiv målrettet sekventering ved brug af long-read sekventeringsteknologi fra Oxford Nanopore Technologies sammen med softwaren ReadFish, eftersom bedre sekventeringsdybde menes at forbedre estimering af repeat længder såvel som ved haplotype fasning af DNA-reads. Haplotype fasning bliver udført for mere præcist at kunne separere estimationer af repeat længder af de to alleler. Adaptiv målrettet sekventering resulterede i berigelse med gennemsnitlig faktor-ændring på 2,34 til 3,92 for panelet, hvilket er indenfor det forventede interval af 2,7-5,4x.

Fire programmer er blevet brugt til at estimere repeat længder af HTT og FMR1: RepeatHMM, STRique, Tandem-Genotypes og TRiCoLOR. Af disse havde TRiCoLOR de bedste resultater, og estimerede tæt på den korrekte repeat længde. TRiCoLOR bruger haplotype fase input filer, og to metoder til fasning bliver benyttet: LongShot og en intern metode lavet til dette projekt. De forskellige strategier til fasning bruger udvinding af genotype informationer ved SNP positionen, hvilket enten er taget fra mere fejlfri short-read sekventerings metoder (Illumina) eller fra mere fejlfyldte long-read sekventerings metoder (Oxford Nanopore Technologies). Der var ingen signifikant forskel i resultaterne fra de to fasnings metoder, og selv alleler med næsten ens repeat længder havde tendens til at være svære at separere med fasning.

Målrettet sekventering er ikke klar til klinisk brug grundet sekventeringsfejl samt variation i repeat længde estimering, som kan have kliniske konsekvenser. Derfor er det nødvendigt med videre forbedringer af sekventering, basecalling og repeat estimering.

Table of content

Abbreviations

AF	Allele frequency
bp	Base pair
DNA	Deoxyribonucleic acid
DP	Depth
CNS	Central nervous system
CRISPR	Clustered Regularly-Interspaced Short Palindromic Repeats
crRNA	CRISPR RNA
FMR1	Fragile X Mental Retardation 1 gene
FXS	Fragile X Syndrome
HD	Huntington's Disease
HMM	Hidden Markov Model
HP	Haplotype Phasing
HTT	Huntingtin gene
IGV	Integrative Genomics Viewer
Indel	Insertion and/or deletion
JHD	Juvenile Huntington's Disease
ONT	Oxford Nanopore Technology
PacBio	Pacific Bioscience
PAGE	Polyacrylamide gel electrophoresis
PCR	Polymerase Chain Reaction
RNN	Recurrent neural networks
RNP	Ribonucleoprotein Complex
SNP	Single nucleotide polymorphism
SNV	Single nucleotide variant
STR	Short Tandem Repeat
STRique	Short Tandem Repeat Identification, Quantification, and Evaluation
SV	Structural Variant
tracrRNA	Trans-Activating CRISPR RNA
TRiCoLOR	Tandem Repeats Profiler for Long-Read Sequencing Data
UTR	Untranslated Region

WG	Whole-genome

1. Introduction

1.1 Variations in the Genome

Genetic variation in humans is part of what makes individuals unique. Genetic variation is a term describing the differences among individuals within a species in the composition of their genes or other DNA segments (Campbell et al. 2015). These variations arise in germlines as well as somatic cells and differ from the reference genome. Take any human genome, and it will differ from the reference genome about 4-5 sites on average (Eberle et al. 2017).

Genetic variation happens when the genetic code mutates or when recombination occurs in the germline during meiosis. In meiosis, the crossover happening during recombination is error-prone. In error-prone processes such as DNA-replication and DNA repair *de novo* mutations can arise (Nørby 2006).

Mutations that cause visible changes in the number and/or structure of chromosomes are called chromosomal mutations, while submicroscopic mutations often are called gene-mutations. Gene-mutations can be divided into four categories: (1) substitution of one base pair (bp), which are called single nucleotide polymorphisms/variations (SNPs/SNVs), (2) deletion of one or more bp, (3) insertion of one or more bp e.g., through duplication, and (4) inversion, which is a 180˚ turn of the orientation of a DNA-segment.

One common mechanism behind insertions and deletions (indels) is replication slippage and unequal crossing-over (meiotic or mitotic). There is a higher probability of unequal crossing-over in repeated areas of the genome, also called repetitive DNA (Nørby 2006).

1.1.1 Repeat Variations and Repeat Diseases

Repetitive DNA can be categorized into two groups: (1) interspersed repeats, and (2) tandem-repeated DNA. Interspersed repeats are randomly spread among the whole genome, while tandem-repeats consist of consecutive repeats (Nørby 2006).

Tandem-repeats can be divided into four groups depending on the size of the repeat unit: (1) mega satellite-DNA, (2) satellite-DNA, (3) mini satellite-DNA, and (4) micro satellite-DNA, also called short tandem repeats (STR). Tandem repeats are sections of the DNA that are repeated continuously, and these repeats are divided into short and long repeats (Nørby 2006).

Tandem repeats can occur in both the coding (exons) and noncoding part (introns) of the genome. In the coding region, about 9% of genes contain tandem repeats. In the noncoding region, about 12% of genes have tandem repeats in their promoter regions. Regions with tandem repeats are hyper-

mutable and are associated with several genetic diseases e.g., Huntington's Disease and Fragile X Syndrome. Tandem repeats have mutation rates that are 10- to 10,000-fold higher compared to other genomic elements (Ameur, Kloosterman, and Hestand 2019).

The sub-category of tandem repeats, STRs, comprise about 3% of the genome and rarely consist of a unit with a size of more than 4 bp in length. One characteristic of STRs is that the number of repeated units can vary, which is most likely due to errors in replication. STRs with units of length 3 is called trinucleotide repeats. Trinucleotide repeats are one example of genetic segments, where variations in repeat lengths are common (Nørby 2006).

Repetitive trinucleotide sequences are common and well known in many species and many of these compose polymorphisms. These polymorphisms are sequence variations in the population with repeat units varying within a limited range that does not cause disease. These repetitive trinucleotide polymorphisms are stable for the relevant chromosome, meaning that they are not subject to changes under mitosis nor meiosis and are inherited in a Mendelian manner (Nørby 2006).

Trinucleotide repeats are not always stable as they are subject to dynamic mutations. Trinucleotide repeats vary in length between generations, normally with a few repeat units. The number of repeat units can however be dramatically changed from one generation to the next, causing an elongation of multiple repeat units. The extend of elongation is disease- and sex-dependent. The dramatic elongation of repeat units can continue for multiple generations, and from one generation to the next, such a dramatic elongation can have severe health consequences. This elongation between generations can change the expression of the mutation and thereby often cause changes in the expression of disease. The expression of the disease will typically become more severe with elongated repeats and cause manifestation in earlier ages, compared to what is seen in previous generations. This type of dynamic mutation elongation in repeats between generations is called anticipation (Nørby 2006).

When a trinucleotide repeat is elongated beyond the specific gene's threshold, disease is likely to manifest itself due to instability. These diseases are generally caused by a few different outcomes, depending on the repeat location. These outcomes include a changed regulation of gene expression, the production of a toxic RNA, a defective protein, altered protein functions, or chromosome instability. A rule of thumb is, the larger the expansion the faster the onset, and the more severe the disease will be (Den Dunnen 2017).

Half of the trinucleotide repeat diseases are caused by expansions in the number of CAG-repeated units located in the protein-coding sequence of the relevant gene. In these genes, the CAG-repeats are in the same reading frame and codes for polyglutamine sequences. These proteins with

polyglutamine sequences are subject to a gain of function. A gain of function results in an increased or altered function of the protein compared to what the normal function of the protein is. These trinucleotide repeat diseases are therefore caused by a similar or shared mechanism. This mechanism includes neurodegeneration leading to neuronal dysfunction and neuronal death. The affected genes are expressed in all tissues but only leads to cell death in the central nervous system (CNS) which is specific for the individual diseases (Nørby 2006).

Trinucleotide repeat diseases that are due to repeat expansions in non-coding regions, like the 5-untranslated region (UTR), are multisystem diseases. These multisystem diseases involve dysfunction and degeneration in many tissues and organs. The repeat expansions in these trinucleotide diseases are larger and vary more than polyglutamine diseases. In these multisystem diseases, premutations occur. Premutations are smaller expansions that do not cause symptoms, but they can expand to fully pathogenic mutations under the gametogenesis (Parnell 2012; Nørby 2006).

1.2 Methods Used to Detects Repeat Diseases Clinically

Clinical test for repeat expansions are used when a repeat disease has been suspected in an individual. The clinical tests used for diseases with short repeats, like those in Huntington's disease (HD), are generally polymerase chain reaction (PCR) tests, which are visualized with electrophoresis. The clinical tests used for diseases with long repeats, like those in Fragile X Syndrome (FXS) are generally Southern blotting (Bahlo et al. 2018). The accuracy of repeat expansion estimation using repeat-primed PCR and Southern blotting are inversely correlated with repeat size and do not evaluate the nucleotide sequence or epigenetic modifications (De Roeck et al. 2019; Bahlo et al. 2018). Especially GC-rich regions are difficult to amplify with PCR and the difficulty increases with increasing repeat lengths. For this reason, many PCR methods do not attempt at detecting large alleles (Monaghan, Lyon, and Spector 2013). An additional issue is that both PCR and Southern blotting methods have a high turnaround time (De Roeck et al. 2019; Bahlo et al. 2018).

There are many PCR methods available for amplifying targets with repeat expansions which have a different set of primers, PCR conditions, and separation and detection methods. One of these methods are triple repeat-primed PCR, which in some assays enables accurate sizing up to 200 repeats (Monaghan, Lyon, and Spector 2013). In the department of clinical genetics at Odense University Hospital (OUH), a locus-specific repeat-primed PCR kit is used for FXS detection. This kit is called AmplideX® PCR/CE FMR1.

Southern blotting has several single- and double-enzymes as well as probe options available for repeat expansion determination (Monaghan, Lyon, and Spector 2013).

Diagnosis of HD is not performed at OUH, but samples are directed to Copenhagen University Hospital, where they use a fragment length analysis.

Fragment analysis is a combination of methods where DNA fragments are fluorescently labeled (PCR), separated by capillary electrophoresis, and the size is determined by comparison to an internal standard (Scientific). The workflow of fragment analysis is visualized in figure 1.

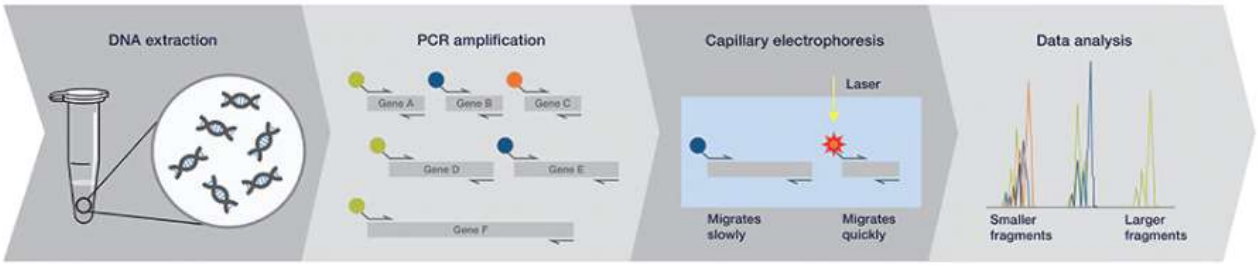


Figure 1: Fragment Analysis Workflow

First DNA is extracted from the sample, then the DNA is amplified with PCR, then the DNA fragments are visualized with capillary electrophoresis, and then the results are being analyzed. Image source: (Scientific)

With the current methods like PCR and Southern blotting, no comprehensive panel testing for all known repeat diseases is existing (Bahlo et al. 2018). Whole-genome analysis on Illumina sequenced data can be used to look for multiple repeat diseases and it is precise with short repeats, but Illumina is quite challenged with longer repeats such as the ones related to FXS. This is due to the inability of short reads to span repeats that are 200+ long, as they are in diseases such as FXS. To combat this disadvantage of short-read sequencing, one can use third-generation long-read sequencing. Therefore, long-read sequencing has been chosen for this project.

1.3 Project Foundation: Description of a Former Students Work

The work done in this project takes roots in the work of a former thesis student, Emilie Boye Lester.

Her project aimed at testing the potential of whole-genome (WG) sequencing in the determination of the repeat lengths in two repeat expansion diseases, namely HD and FXS. WG sequencing was performed as it has the potential of replacing standard methods and optimizing genotyping.

The WG sequencing was performed using both short-read and long-read sequencing platforms on six samples, which also are the samples used in this thesis (see materials and methods). The platforms used were Illumina NovaSeq 6000 and Oxford Nanopore Technologies (ONT) PromethION. This aim was created to find a *one fits all* analysis approach that would have the potential of lowering the cost of genetic screening, as diagnostics with the help of PCR can be time-consuming, and sometimes additional tests are necessary after doing PCR.

For Illumina sequenced data a repeat expansion analysis with ExpansionHunter was performed. For ONT sequenced data a repeat expansion analysis was done with STRique. She found that ExpansionHunter (Illumina) (Dolzhenko et al. 2019) was only precise in genotyping when the repeat length was below 40 repeat units. Genotyping with STRique (ONT) (Giesselmann et al. 2019) was capable of distinguishing normal alleles from pathogenic alleles, though the repeat length estimation was relatively inaccurate.

The former student concluded that the limiting factors in determining correct repeat length from short-read data were the read-length, which makes Illumina sequencing unsuitable for clinical diagnostics for most repeat expansion diseases.

The former student concluded that the limiting factors in determining correct repeat length from long-read data were low coverage, a high number of sequencing errors, and premature software packages.

1.4Repeat Diseases Chosen for this Project

To detect the repeat lengths in long-read sequencing data multiple programs will be used. Since the range of repeat units in repeat diseases varies substantially, the programs need to be tested in both short and long repeats to make sure that they are not lacking in one or the other.

As an example of a short read, HD has been chosen, and as an example of a long repeat, FXS has been chosen. These two diseases were also chosen in the project of the former thesis student, and therefore extra data is available by choosing these two diseases as subjects for analysis for this thesis. These two diseases are shortly described below, including their repeat length boundaries, which are quite different.

1.4.1 Short description of Huntington's Disease

Huntington's Disease (HD) is an autosomal dominant neurodegenerative disorder caused by a

pathologic extension of the CAG trinucleotide repeat on the short arm of chromosome 4p16.3 in the coding part of the Huntingtin (HTT) gene. HD typically manifests itself at age 30-50 with chorea as the basic symptoms and the following accompanying symptoms: Dystonia, cognitive deficits, psychiatric problems, and weight loss. A general rule of thumb is that the longer the repeat, the earlier the onset of the disease. HD onset before the age of 20 is called Juvenile Huntington's Disease (JHD) (Roos 2010; MacDonald et al. 1993; Den Dunnen 2017; Ross et al. 2014; Nørby 2006).

The repeat length boundaries for having HD are set to 36 CAG repeats or more, although individuals having between 27 and 35 repeats are said to have intermediate or mutable normal alleles, meaning that the alleles are unstable and prone to undergo elongation during reproduction. This is mostly seen in male reproduction and is called anticipation (Roos 2010; Kelly et al. 1999; Myers et al. 1993). The individuals with HD having an allele with repeat length between 36 and 39 often show incomplete penetrance or late-onset of disease (Trottier, Biancalana, and Mandel 1994), compared to the individuals having alleles with repeat lengths above 40, which is called full penetrance alleles (Brinkman et al. 1997; Nørby 2006). These repeat size boundaries are illustrated in Figure 2.

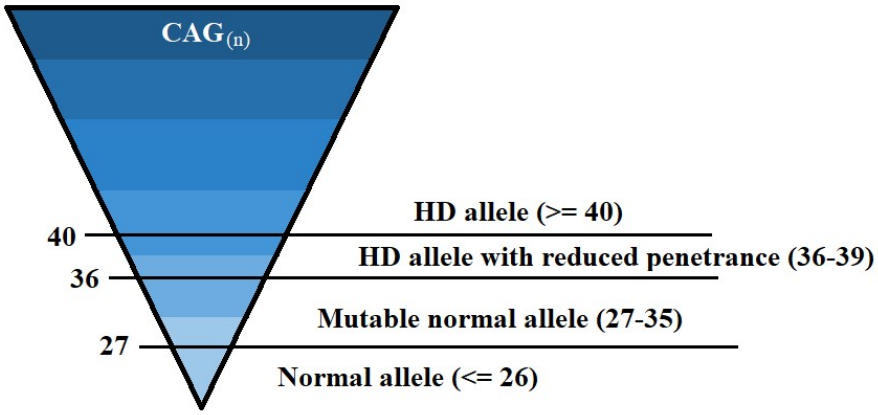


Figure 2: Diagram of the repeat size boundaries for HD.
Normal alleles are alleles with repeat sizes that are not pathologic and segregate as a stable repeat. Normal alleles contain repeats of sizes below or equal to 26. Mutable normal alleles are non-pathogenic but the size of the repeat makes them prone to unstable segregation. Mutable normal alleles have sizes between 27 and 35. Alleles with reduced penetrance are pathologic alleles with repeat sizes above or equal to 40. The figure is inspired by: (Bean, Bayrak-Toydemir, and Committee 2014).

1.4.2 Short description of Fragile X Syndrome

Fragile X Syndrome (FXS) is the most common form of inherited mental retardation and is visible as

a fragile site at the long arm of the X-chromosome, which is at location Xq27.3. FXS is inherited in an X-bound recessive manner (Bahlo et al. 2018; Jensen 2011; Nørby 2006). FXS has a prevalence of 1:4,000 in males and 1:6,000-7,000 in females (Turner et al. 1992; Verkerk et al. 1991; Jensen 2011). The basic symptom for FXS is mental retardation with the following accompanying symptoms: mood disorder, mutism, and autism (Den Dunnen 2017; Hagerman and Hagerman 2004). FXS is caused by expansions of a CGG trinucleotide repeat in the 5'-UTR of the fragile X mental retardation 1 (FMR1) gene (Bahlo et al. 2018; Den Dunnen 2017). A normal repeat length is up to 55 long. A premutation arises with repeats between 55 and 200 repeats, though some sources have set the full mutation boundary to be 230 repeats. A premutation can only expand to a full mutation through the female meiosis, therefore only females can transmit a full mutation to their children. A premutation can either expand or reduce by a few units when transmitted from a father to a daughter. A full mutation occurs when the repeats exceed 200 repeats. A full mutation generally leads to methylation of the repeat and promoter region leading to silencing of the FMR1 gene (Willemsen, Levenga, and Oostra 2011; Yu et al. 1991; Oberlé et al. 1991; Bell et al. 1991; Jensen 2011; Nørby 2006).

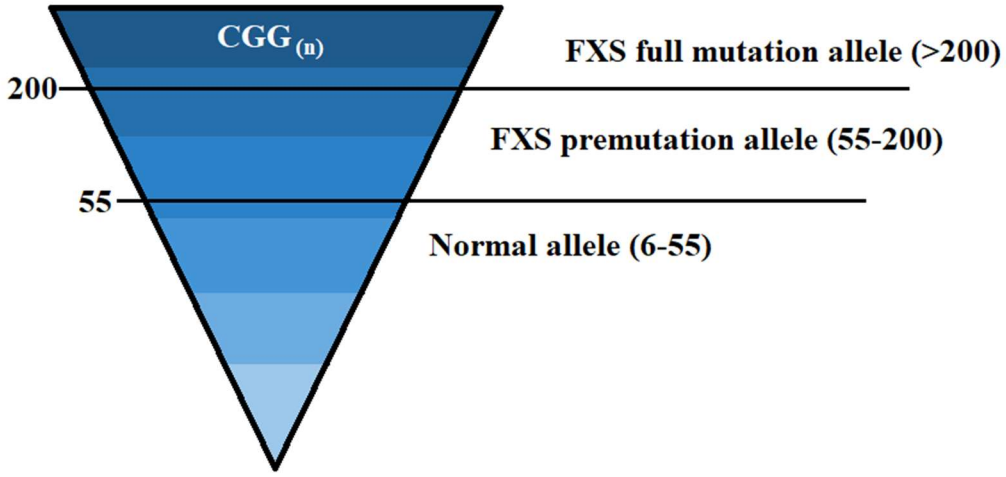


Figure 3: Diagram of the repeat size boundaries for FXS.
Normal alleles have repeat sizes that are not pathologic and segregate as a stable repeat. Normal alleles contain repeats of sizes up to 55. FXS premutation alleles are non-pathogenic but the size of the repeat makes them prone to unstable segregation. Premutation alleles have repeat lengths between 55 and 200. Alleles with a full mutation have repeat lengths above 200 units. The figure is inspired by: (Bean, Bayrak-Toydemir, and Committee 2014).

The pathogenic length of the FXS trinucleotide is far longer than that of HD, which is why FXS has been chosen together with HD for testing of repeat estimating programs. Before estimating the repeat lengths, samples must be sequenced.

1.5Long-Read Sequencing

As described in section 1.2, NGS such as Illumina can precisely determine the repeat length for short repeats such as for HD but are challenged with large repeats which occur in FXS. Third-generation sequencing, also known as long-read sequencing, has the advantage of easily covering large repeat regions, making repeat length determination possible for long repeats as well as small.

Third-generation sequencing (TGS) is the new era of sequencing, which is represented by two sequencing technologies, one developed by Pacific Bioscience (PacBio) and one developed by Oxford Nanopore Technologies (ONT). These two technologies sequence long DNA strands compared to NGS. TGS is also capable of live-sequencing allowing analysis of the sample as it is being sequenced. Long-read sequencing does not require PCR amplification (opposed to NGS), which is an advantage in CG-rich areas of the genome, like the CGG repeat in FMR1 (Bahlo et al. 2018).

The two TGS technologies have different methods of sequencing, which are visualized in Figures 4 and 5 below. ONT is a technology using nanopores, and PacBio is a technology using wells. ONT distinguishes electrical changes as seen in Figure 4, while PacBio detects fluorescent signals as shown in Figure 5. Of these two TGS technologies, ONT has been chosen for this project due to availability and cost as well as the ability for doing targeted sequencing with a computational approach. Both technologies have a high error-rate in base calling of ~5-20 % randomly distributed (Sedlazeck et al. 2018). However, PacBio has the advantage of sequencing the same DNA-strand multiple times lowering the error-rates compared to ONT sequencing error-rates.

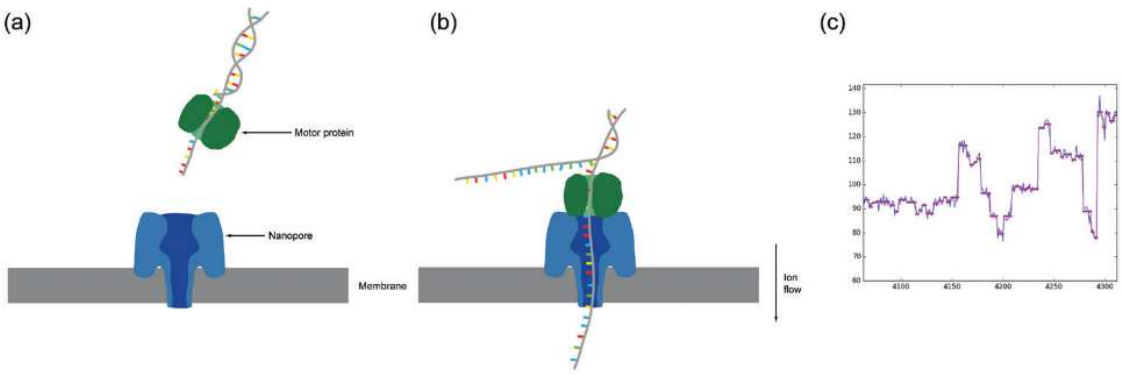


Figure 4: ONT Sequencing

When sequencing using Oxford Nanopore technologies (a) the priming of the flow cell includes adding tethers (not shown) which help bringing the DNA-molecules to the membrane. The DNA-strands have adapters attaches to the DNA ends which includes a leader sequence and a motor protein. The motor protein (b) docks with the pore translocate the DNA-molecule through it. The bases in the narrow part of the nanopore cause disruptions in the current (c), which are sequence-dependent.
Figure source: (Leggett and Clark 2017)

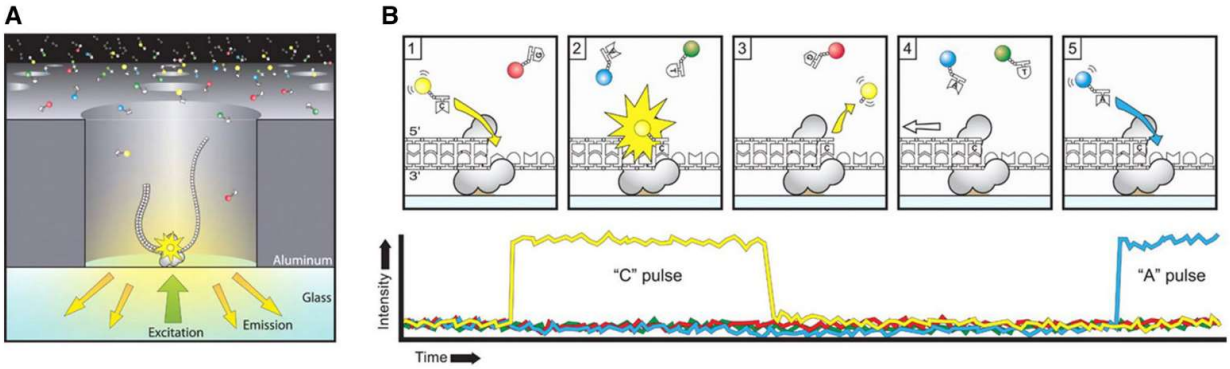


Figure 5: PacBio Sequencing

When sequencing using PacBio, chambers are used as seen in (a), where a DNA-molecule is being sequenced. During sequencing a phosphor-linked nucleotide is being connected to the strand being sequenced (B1), leading to a fluorescent signal being released. (B3/B4). Then the next phosphor-linked nucleotide can be added to the sequence (B2). The figure is from (Eid et al. 2009).

1.5.1 Oxford Nanopore Technology Sequencing

Oxford Nanopore Technologies (ONT) currently has three sequencing devices on the market: the MinION, the GridION, and the PromethION. The MinION, visualized in Figure 6, is a small transportable device with space for one flowcell. The GridION is a benchtop device allowing up to five flowcells. Finally, the PromethION is a benchtop device allowing up to 48 flow cells and comes with a compute module. These devices can both sequence DNA and RNA, and they can in addition also identify proteins.

The PromethION was used in the former thesis students project and the MinION is used in this thesis. The MinION device is shown and described in Figure 6 below.

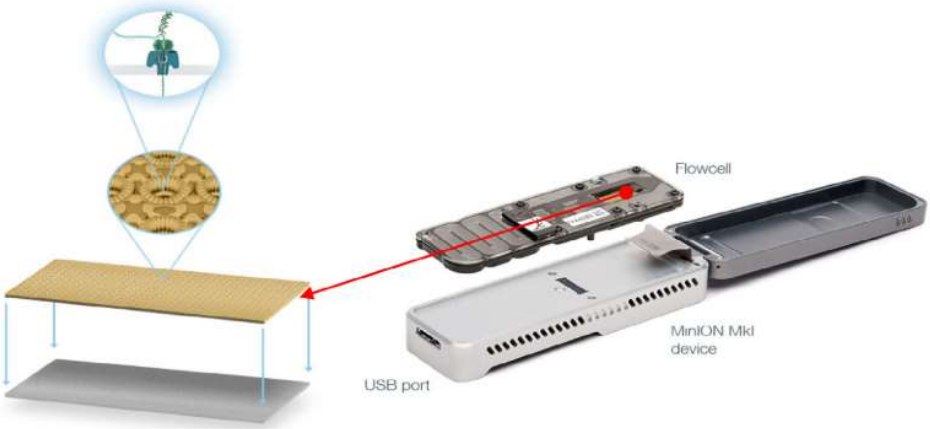


***Figure 6:* The MinION Sequencing Device**

The MinION is a portable device with a USB connection. A sample is added to the SPOT-ON port in the flow cell and the DNA-molecules will pass the pores in the membrane once an electrical current is enabled. The DNA-strand going through the pore is altering the current depending on which nucleotides are present inside the pores. These differences in current are what make up the raw

signal of nanopore sequencing. These signals are passed to the ASIC and MinKNOW. The MinION device is an array of 512 sensors, each connected to four nanopores of which only one is active at all times (Leggett and Clark 2017). Figure Source: (Lu, Giordano, and Ning 2016)

The computational interface used for launching a sequencing run is called MinKNOW, and this software controls the sequencing device, collects data in real-time, and processes it into basecalls (Technologies 2018a). From this interface the devices are connected, the flow cells are checked, and the sequencing is started and managed. During setup of a sequencing experiment, the laboratory kit used is selected as well as the choice of basecalling while sequencing. More choices are available.

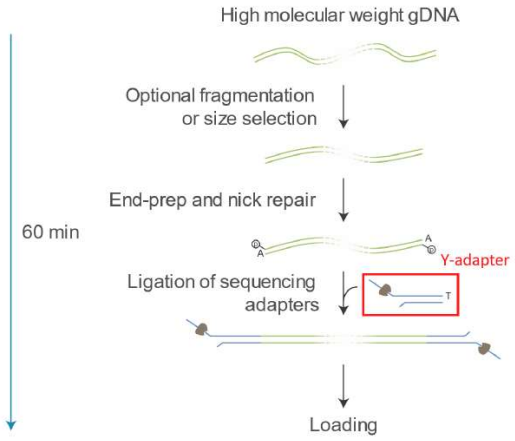


Figure 7: ONT Library Preparation

The steps in preparing the DNA library includes a selection of fragmentation sizes, repairing the ends of the DNA, preparing the ends of the DNA for adapter ligation, attaching sequencing adapters (highlighted in the red box) to the DNA ends, priming the flow cell, and loading the flow cell. Priming the flowcell includes adding flush buffer and flush tether. The flush tether helps bring the DNA-molecules to the membrane and the motor proteins in the adapter unzip dsDNA and translocate it through the membrane in the flow cell. Image source: (Technologies 2016b).

The flow cells, which are visualized in Figure 6, are the heart of the sequencing device. These flow cells are small chips with protein punctured membranes separating two compartments. This artificial membrane consists of synthetic polymers that are electrically resistant. The membrane is connected to an application-specific integrated circuit (ASIC) and when the electrical potential is applied, an ion current is going through the membrane pores. This ion current changes when DNA is passing through the pores that are embedded into the membrane.

Sequencing adapters, comprising of motor protein and leader sequence, are ligated to the DNA strands during the library preparation, which is the conversion of a sample into a format compatible with the sequencing system. When loading the flow cell there is added flush buffer and flush tether. These tethers help to bring the DNA-strands to the membrane of the flow cell. The motor proteins unzip dsDNA and translocate a single DNA-strand through the pore one base at a time. When the

template strand has passed through the pore the complement strand dissociates and may be captured by a nanopore later. The adapters are highlighted in Figure 7, where the overall steps of the library preparation are illustrated.

When DNA-strands are processed through the pores a squiggle pattern emerges as the result of altered electrical resistance inside the nanopore. The nucleotides and their epigenetic modification cause specific and individual changes in the current that is used when analyzing the raw current signal, which is referred to as squiggles. The squiggles are the raw sequencing data produced by ONT devices and are output as fast5-files. Fast5 files are a type of HDF5 files and contain all information from the sequencing, which are needed for analysis. These squiggles are translated by a basecaller called Guppy either in real-time or after ended sequencing (Leggett and Clark 2017; Technologies 2016c; Wick, Judd, and Holt 2019; Technologies 2016b).

The MinION device holds flow cells with 512 sensors, which all are connected to four individual nanopores. One of these four nanopores will always be active. These pores undergo different states, which overall consist of sequencing, not sequencing, being inactive, and recovering. Every 8 hours of sequencing, a mux scan is performed to identify the highest performing nanopore, which is then chosen for the next chunk of sequencing time. Sequencing runs will in general last up to 48 hours, though the first 24 hours are producing a higher yield as the flow cell performance gradually declines (Leggett and Clark 2017).

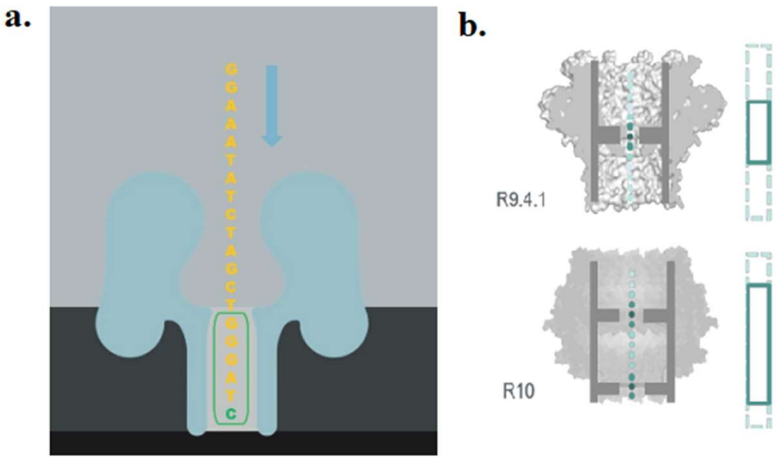


Figure 8: Oxford Nanopore Technologies Nanopores

The nanopore has a narrow part, called the reader, where the DNA-sequence is passing. The combination of nucleotides inside the nanopore (a) gives a distinct change in current. These current changes are the raw data of ONT and are called squiggles. There are multiple pore versions where the major difference is the number of readers in the pore (b). Version 9 has one reader while version 10 has two. Images modified from: (Technologies 2016c) and (Technologies 2016a)

ONT produces flow cells with different types of pores of which the R9.4.1. and R10 are worth mentioning. The structure of the pore determines the raw signal and this is one of the parameters available for optimization of sequencing (Technologies 2016c). Version 9 of the nanopore is a mutant of the CsgG lipoprotein from *E. coli* (Technologies 2016a).

The pores from ONT have one or more readers, which are the molecular constrictions in the pore that comes in touch with the DNA-sequence, as seen in Figure 8. The readers interact with the DNA and produce the changes in the ionic current. The R9.4.1 has one reader where the R10 pore has two. With more readers comes a larger interaction between the DNA and the pore, making the detection of homopolymeric regions easier. The R10 pore has better single-molecule accuracy when reading homopolymers as well as higher consensus accuracy (Technologies 2016c; Technologies 2016a). In this project older chips have been used with the R9.4.1 pores, since they were available and due to expire.

1.6Targeted Sequencing

Targeted sequencing is a sequencing strategy where specific regions of interest (ROI) are being enriched. The two types of methods used for targeted sequencing (enrichment) are either focused on laboratory practices or computational set-up. When selecting ROIs in the laboratory preparation, CRISPR/Cas9 technology can be used. When selecting ROIs in the sequencing phase, a program called ReadFish can be used. Both methods are described below.

The CRISPR/Cas9 kit from ONT was released in September 2020, while the ReadFish algorithm was ready for use in late February 2020. The CRISPR/Cas9 kit from ONT was not available early in the thesis making CRISPR/Cas9 enrichment a separate process that should be performed before the ordinary ONT library preparation of the samples and third-party reagents should be used. Regardless of which reagents one would use (third party or the ONT kit), CRISPR/Cas9 is a possible targeted enrichment strategy.

1.6.1 CRISPR/Cas Enrichment

CRISPR stands for clustered regularly-interspaced short palindromic repeats. The CRISPR/Cas9 complex contains a DNA-cleaving enzyme that can be programmed to cut specific DNA-sequences. ONT has developed a method to use the Cas9 enzyme to target an ROI, which leads to the attachment of sequencing chemistry as seen in Figure 9. This CRISPR/Cas9 prepared DNA can be added to a nanopore device for sequencing. The kit ONT has developed is called SQK-CS9109. One can also

perform a CRISPR/Cas9 experiment before using one of the standard sequencing kits from ONT called SQK-LSK109 and SQK-LSK110 (Technologies 2018a, 2020b, 2020a).

With SQK-CS9109 a coverage of >100x for the ROI is expected on a MinION MK1B flow cell, and one can have up to 100 crRNAs in a single panel. One ROI needs 4 crRNAs, which leaves us with 25 ROIs as the maximum for one experiment. A lower coverage is expected if the size for excision exceeds 10 kb. The method is PCR free meaning that there is less CG-bias and epigenetics are preserved (Technologies 2020a).

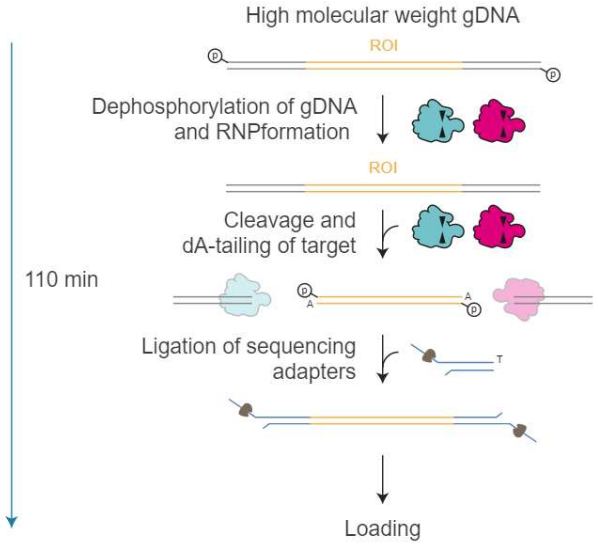


Figure 9: ONT Cas Sequencing Library Preparation

(1) The extracted DNA is dephosphorylated at the 5'ends to avoid ligating adapters to non-target strands. (2-3) The RNPs are added and they cleave the ROI leaving blunt ends with 5'phosphates that are ready for ligation. The DNA is dA-tailed at the 3'ends as preparation for ligation. (4) The adapters are ligated to the Cas9 cut sides. (5) The library is cleaned up to remove excess adapters and then the library is loaded to the flow cell for sequencing. This takes approximately 110 minutes.
Image source: (Technologies 2020a)

The CRISPR/Cas family comprise of DNA- or RNA-cleaving enzymes that are guided to the cutting site by short oligonucleotide sequences called CRISPR RNAs (crRNA). The crRNA is forming a ribonucleoprotein complex (RNP) together with trans-activating CRISPR RNA (tracrRNA). In the case of targeted enrichment of human DNA, it is this complex of RNA and protein that searches the genomic DNA for the target region (ROI) guided by the crRNA, and when the target region is found the Cas9 enzyme binds to the site and cleaves the genomic DNA.

The Cas9 enzyme cleaves the DNA 3 bp upstream of the protospacer adjacent motif (PAM). The crRNA sequence is the same as ROI, since the cut is not made inside the ROI but upstream of it.

When both strands are cut the exposed ends are called ‘deprotection’ by ONT, and this is the basis of the selectivity of the Cas-mediated enrichment method.

The CRISPR/Cas complex can be seen in Figure 10. It is the crRNAs that one needs to design for one’s experimental needs. The designing of crRNA can be done by bioinformatic searches of reference genomes. A useful tool for designing crRNA is CHOPCHOP (Technologies 2018b; Labun et al. 2019).

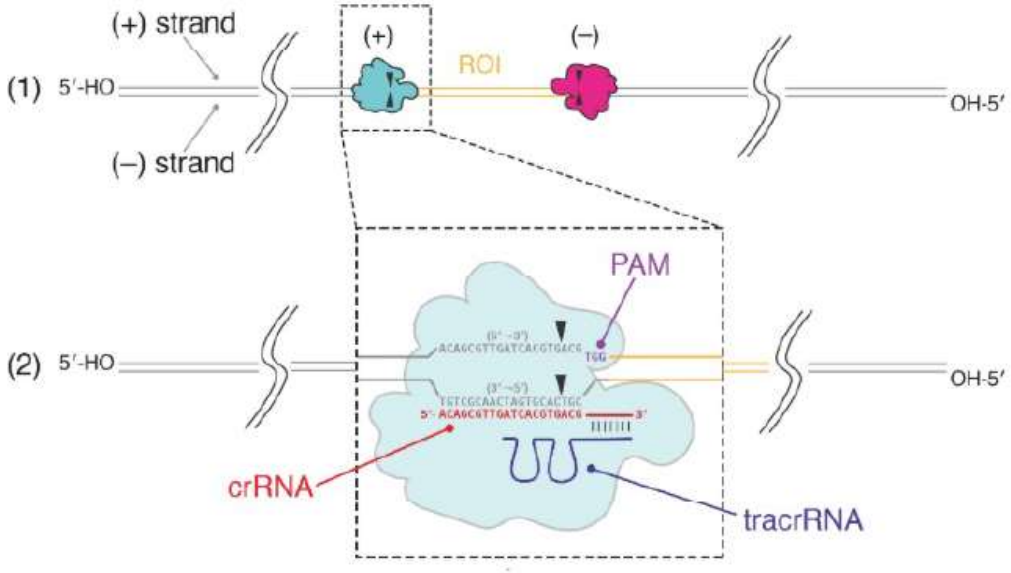


Figure 10: The RNP Complex Bound to The Target Sequence
(1) The CRISPR/Cas RNP complex highlighted is bound to the ROI. (2) The RNP melts the DNA duplex upon hybridization with the complementary crRNA-strand, which is upstream of the PAM. Image source: (Technologies 2018b)

There are three CRISPR/Cas9 approaches: excision, tiling, and single cut and read out. In the excision approach, both sides of the ROI are targeted for cutting and is generally recommended when the user has one ROI.

In the tiling approach, multiple crRNA probes are needed and they are designed along an ROI of 5-10 kb overlapping chunks that allows for even coverage across the ROI. The tiling approach is recommended when the ROI is larger than 20 kb.

In the single cut and read out approach, one crRNA probe is designed to cut at one side of the ROI. This leads to the sequencing of the ROI and the rest of the sequence on that DNA-strand. This method is recommended when the ROI is larger than 20 kb and only one side of the ROI is known (Technologies 2018b).

1.6.2 Targeted Sequencing with ONT and ReadFish

ReadFish is compatible with the MinION device from ONT as it is possible to reverse the voltage across specific nanopores, forcing the pore to unblock the DNA strand. It is the Read Until API software from ONT that makes the targeted sequencing possible (see Figure 11). This API provides a stream of raw current samples from the sequencing pores. To manage this software, a second software called ReadFish has been used as a client to manage the setup of the experiment (Payne et al. 2020a). ReadFish is a python3 package that integrates with a slightly modified version of Read Until API.

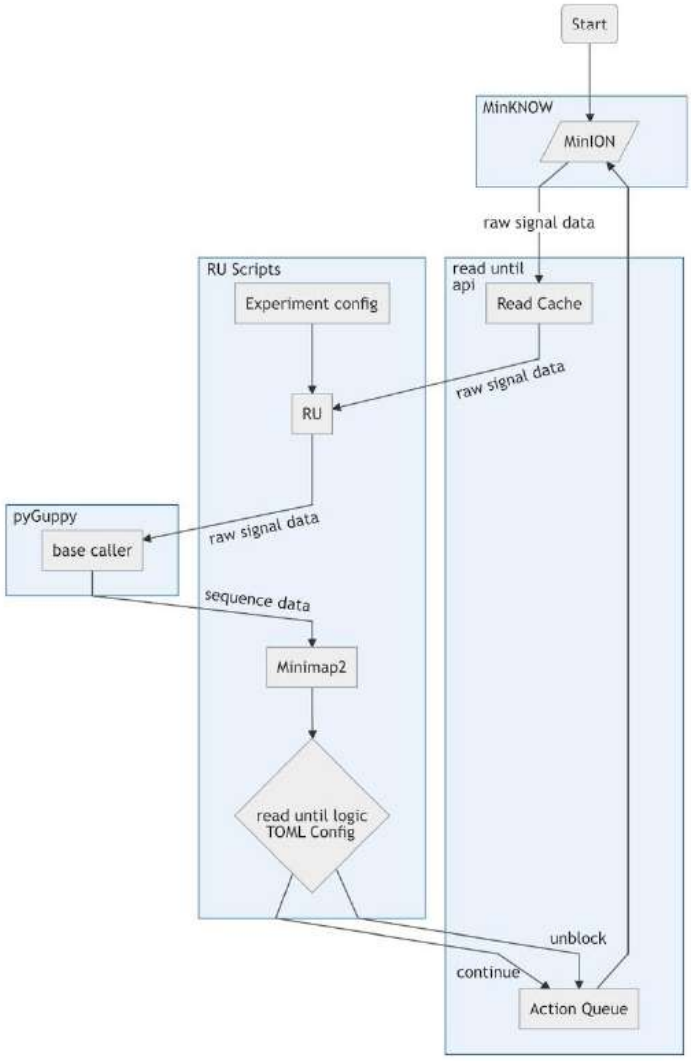


Figure 11: Flow Diagram of the Core Read Until Program.

The Read Until program manages the basecalling and aligning of read fragments from the MinION. First, a run is set up and sequencing starts, then a chunk of data is being processed by Read Until (RU) scripts and fed to pyGuppy. The basecalled sequences can then be mapped by Minimap2 and depending on the experiment setup, an action is given to the action queue which feeds back to the MinION device. Image modified from (Payne et al. 2020a).

Read Until API makes it possible to obtain read data in real-time by connecting it to a MinKNOW server, making real-time analysis and actions possible. A basic way of describing the method behind ReadFish is that chunks of data are taken from the Read Until API, then these chunks of data are processed in one batch by converting the raw signal into compatible reads for Guppy, followed by the data chunk being basecalled using python bindings to Guppy (ONT 2020). Then a python API for minimap2, called mappy, is being used (Li 2018). A flow diagram of the Read Until API is presented in Figure 11 (Payne et al. 2020a).

Depending on the output wanted, different actions happen from here. In this project, targeted sequencing experiments are performed and three actions happen to the read fragments: stop receiving, proceed, and unblock. Stop receiving means that the API stops receiving information for the remainder of that read. The action 'stops receiving' happens when a read fragment maps to multiple locations including the region of interest (multi_on) or if the read fragment only maps to the region of interest (single_on). The action 'proceed' means that data continues to sequence and serve through the API for later decisions. When a read fragment maps to multiple locations, not including the region of interest (multi_off), or if the read fragment does not map to the reference, the read is given the action 'proceed'. The action unblock happens when a read maps to one location but it is not a region of interest (single_off) and then the read is rejected (Payne et al. 2020a).

If a read cannot be mapped in the first chunk a read cache has been implemented by ReadFish to store chunks from the same read such that the adjacent signal data from the same read can be basecalled and mapped. Therefore, the action 'proceed' makes sense. The team behind ReadFish finds that 90% of reads can be called and mapped within 3 chunks (3*0.4 seconds = 12 seconds) using this method (Payne et al. 2020a).

Besides targeted sequencing, Read Until makes depletion and balancing of mixed samples possible, and the team behind ReadFish have made it easy to design the actions of rejecting and continual sequencing, through TOML configuration files (Payne et al. 2020a).

ReadFish is being used in this project to enrich a target panel consisting of 27 tandem repeats, expecting to obtain a greater depth at the targets than possible with whole-genome sequencing. A greater depth at the targets is advantageous further down the analysis pipeline, where multiple software are being used to estimate the number of repeat extensions.

The team behind ReadFish made some selective sequencing experiments using a target panel and

reached an effective enrichment of 2.7-5.4x depending on the flow cell (Payne et al. 2020a). This is what is expected from our targeted sequencing experiments.

1.6.3 Choice of Enrichment Method

When doing a CRISPR/Cas experiment 1-10 ug DNA is needed. The samples left from the former thesis student varied in concentration and volume, making it unfeasible to do a CRISPR experiment. Though, more extracted DNA for the same samples could be ordered.

When using uncut DNA there is more sequence available on both sides of the targets, making more SNPs available for haplotype phasing (HP). It is unknown how many SNPs are needed for the haplotype phasing (explained in sections 1.7.4, 2.5, and 2.6) and therefore how large the target regions should be. Though, when doing a CRISPR/Cas9 enrichment, one can include large DNA-flanks on both sides of the ROI if a drop in coverage is acceptable. The degree of this drop is unknown.

More optimization and testing would be required if CRISPR/Cas9 were chosen for enrichment compared to using ReadFish, as there is an optimization challenge in designing crRNA probes as well as the CRISPR/Cas9 approach chosen. ReadFish includes the ability to change the target regions (ROI) on the spot as well as other parameters to optimize the targeted enrichment in a single run.

If CRISPR/Cas9 was chosen for the experiments, the target panel chosen for the enrichment experiments would have to be shortened as the maximum of ROIs are 25, and the panel chosen for the experiments consist of 27 known repeat diseases.

ReadFish was chosen for this experiment due to the flexibility of choosing targets and to add a computational challenge instead of a laboratory challenge, as bioinformatics is of focus in this thesis. ReadFish furthermore requires no custom sample preparation, meaning the standard protocols from ONT can be used. The computational requirements were also met in the lab making ReadFish available.

1.7Long-Read Data Processing

Long-read data processing contains a few steps, where the general steps are basecalling and alignment. Other steps are added to the analysis pipeline depending on the goal of the analysis. This project has the additional steps of estimating repeat counts and haplotype phasing.

1.7.1 Basecalling

Basecalling is the computational process of translating raw sequencing signals into nucleotide

sequences, also referred to as bases. ONT outputs squiggles which the basecaller Guppy translates into nucleotides. The Guppy toolkit contains the basecalling algorithm from ONT's production as well as several bioinformatic postprocessing features. It is featured in the sequencing instrument software MinKNOW (ONT 2003 Updated 2018; ONT). The input for Guppy is fast5 files, which contain sequencing information and metadata for the reads. The output of Guppy is fastq files, which contain information for 4000 reads. Basecalling of ONT device signals (squiggles) is a challenging machine learning problem as the nanopore holds multiple nucleotides while sequencing. The *R9.4 pore* holds approximately 5 nucleotides yielding $4^5 = 3125$ possible states, for a standard 4-base model. When these bases are modified with methylations and/or other epigenetic marks, the number of possible states increases to $5^5, 6^5, 7^5$ etc. The models that Guppy uses are based on recurrent neural networks (RNN) as well as other options and algorithms (Wick, Judd, and Holt 2019; ONT 2003 Updated 2018).

1.7.2 Genome Alignment

Aligning the sequences is the process of taking a DNA-read and figuring out where in the genome that sequence is located, hence which chromosome and exact position it came from. This process is also called mapping, as an algorithm tries to map the DNA sequence to a reference genome.

In this project, Minimap2 is used for aligning to the reference genome as it is one of the fastest, and is well-known for aligning error-prone long-read data. The input of Minimap2 is fastq files, which are being processed to sam-files and later bam-files (Li 2018; Zhou, Lin, and Xing 2019).

Another alignment strategy used in this project is LAST alignment. This was chosen for the repeat estimating program called Tandem-Genotypes, as the input of the program demands specific pre-processing of the sequenced data, which are different compared to the other three repeat estimating programs used in this project.

1.7.3 Repeat Length Estimation

Repeat estimation is the process of evaluating the number of repeat units in the genome, sometimes at specific locations. This is of interest when researching diseases such as HD and FXs as their repeat lengths vary from the reference genome. Precise repeat counting is important when diagnosing patients as small and erroneous variations can be of clinical importance. The input of repeat estimating programs is many including fast5, fastq, and bam files.

The programs used for estimating repeat lengths in this project are RepeatHMM, STRique, Tandem-Genotypes, and TRiCoLOR. These programs are described in Materials and Methods (section 2.8).

1.7.4 Haplotype Phasing

Haplotype phasing is the process of figuring out which of the chromosomes in a chromosomal pair a specific DNA-read belongs to. This is of interest since the chromosomes in a chromosomal pair differ from each other and possibly also differ in areas such as the ones causing HD and FXS.

Humans have diploid genomes consisting of homologous chromosomal pairs, where chromosome A in the pair is inherited from parent A and chromosome B in the pair is inherited from parent B. One characteristic of having a diploid genome is that every human carries two versions of every gene, with some exceptions of the genes found on the sex-chromosomes.

Females carry two X chromosomes (XX), and males carry one X and one Y chromosome (XY), where only small parts of the X and Y chromosome are homologous. Therefore, most genes on the male sex-chromosomes only have one variant, making the inherited character depend on one version of the gene (Campbell et al. 2015).

One variant of a gene is called an allele, and every gene in a human consists of two alleles which can either be heterozygous (different from each other) or homozygous (identical). A group of alleles inherited together from one parent can construct a haplotype. These haplotypes can be constructed bioinformatically by looking at SNPs. In other words, a haplotype can be described as the combination of alleles at SNP sites on a chromosome, and haplotypes represent the complete information on DNA variation in an individual (Edge, Bafna, and Bansal 2017).

When sequencing DNA, small chunks of both chromosomes A and B are separated into two strands, giving four possible DNA sequences for a specific gene. When analyzing repeat lengths, it is important to assign the repeat count to either chromosome A or B to distinguish the repeat counts, especially when the repeat lengths are close to equal. When using error-prone technology like ONT it is vital to separate the reads into chromosome A or B, since the result of an analysis tool can be more complicated to analyze compared to a less error-prone sequencing method, like Illumina, was used. For example, it is hard to distinguish a case where the repeat lengths in HTT only differ with 3 repeats as sequencing errors and analysis errors happen.

To figure out the chromosomal origin of the DNA reads, markers are essential. The most common DNA identifiers are SNPs, which often are inherited together in blocks of DNA called haplotypes,

which often cover multiple genes. This makes SNPs useful in distinguishing DNA-reads covering the same gene, making genotyping possible. This means that phasing of variants is possible if they are present on the reads. Longer reads can cover many variants, making long-read sequencing an advantage (Cretu Stancu et al. 2017).

A read needs at least two heterozygous SNPs to be useful for haplotype phasing (HP). Reads from ONT typically cover multiple heterozygous sites and therefore provide partial haplotype information (Edge, Bafna, and Bansal 2017). This haplotype information from heterozygous SNP sites can be used to phase the reads as a SNP on a haplotype is expected to segregate with reads from the same haplotype (Edge and Bansal 2019). However, one should keep sequencing errors in mind.

When HP ONT reads, parental information is not required as the two methods used here can phase variants that are specific for the individual, which is partly due to the long reads (Edge, Bafna, and Bansal 2017).

The two HP methods used in this project are LongShot and an in-house Illumina-Guided HP. Longshot is a program used for HP ONT reads, which combines variant detection and haplotyping (Edge and Bansal 2019). The in-house Illumina-Guided algorithm is a haplotype phasing pipeline developed for this project. It was developed on the assumption that Illumina is less error-prone and more reliable than LongShot. The in-house Illumina-Guided method obtains SNPs from Illumina sequenced data instead of obtaining SNPs from ONT sequenced data. When basing the HP on Illumina chosen SNPs, the haplotypes should theoretically be more precise compared to Longshot which bases HP on SNPs found in ONT data. The hypothesis is that the Illumina VCF file, which contains SNP information, can be used to find true SNPs in the ONT reads, and thereby be used to make more valid haplotypes on ONT reads. Illumina sequenced data are not used alone to make haplotypes since short reads make it difficult to link distant SNPs and thereby creating haplotypes (Edge, Bafna, and Bansal 2017). Read length, sequencing errors and fluctuating coverage could be major limiting factors that may induce false positive and true negative SNPs (Cretu Stancu et al. 2017).

1.8 Aims of the Thesis

The overall aim of the project is to explore the potential of long-read Nanopore sequencing for the detection of pathogenic tandem repeat expansions.

The first specific aim is to develop a data analysis pipeline by testing and combining different software tools for repeat expansions, as a former master student concluded premature software as a limiting factor. The software used in this project is RepeatHMM, STRique, Tandem-Genotypes, and TRiCoLOR. The goal of using these programs is to determine how precisely one can estimate repeat lengths from error-prone ONT sequenced data.

To address the problem of low coverage, found in the former student thesis, the second specific aim is to test targeted sequencing using ReadFish. Targeted sequencing also has the potential of increasing sequencing depth for a target panel and with such a target panel, ONT sequencing has the potential of being a 'one method fits all' in terms of diagnosing repeat diseases.

The third specific aim is to HP the samples in order to more precisely separate data from the two alleles, as this is difficult when the repeat lengths are close to equal. Haplotype phasing is also necessary for using the repeat estimating program TRiCoLOR. A hypothesis is that some of the SNPs found by sequencing with ONT are false positives, an in-house Illumina guided haplotype phasing method is being created to borrow the precise SNPs from Illumina sequencing when phasing the reads from ONT.

2 Materials and Methods

2.1 Sample Characteristics and Whole Genome Data

WG sequencing was performed on commercial immortalized lymphoblastoid cell lines (*Coriell,* USA), using both short-reads (Illumina NovaSeq 6000) and long-reads (Oxford Nanopore Technologies PromethION) in a previous project. The data generated from the previous project was used in this project in combination with long-read targeted sequencing data, generated in the current project. The former student extracted DNA from the cell lines and stored extracted DNA in a fridge, which has been used for targeted sequencing in this project.

The commercial cell lines have been characterized by Coriell, including patient information (see table 1). Automatic fluorescent sequencing, capillary electrophoresis, and polyacrylamide gel electrophoresis (PAGE) were used for determining the number of repeats. The commercial samples have been analyzed thoroughly by *Coriell* and their repeat count values are used as expected values.

The length of the repeats linked to HD and FXS vary for the sample ID's, as seen in table 1. There are two samples selected for repeat expansion characterization in HTT (HD) and four samples

selected for repeat expansion characterization in FMR1 (FXS). The ID's for the two HD samples are GM04284 and GM05538 (Coriell ; Coriell). The ID's for the four FXS samples are GM20239, GM06891, GM07541, and GM07861 (Coriell ; Coriell ; Coriell ; Coriell).

Table 1: Sample Characteristics and Expected Values

Characteristics of samples from Coriell included in the study: (1) sample ID, (2) disease type, (3) repeat motif, (4) chromosomal location, (5) disease onset or clinical diagnosis, (6) patient age at sampling, (7) sex (M = Male, F = Female), and (8) repeat count given as [allele 1/allele 2]. Since HD develops later in life, and FXS develops at conception, there is a difference in whether the onset age is given or the clinical diagnosis. This means that the two samples with HD are clinically affected at the onset age and unaffected, but not disease-free, before that age. A reference to the commercial web-page for the samples has been made under Coriell and the sample ID.

i. *Clinically unaffected but fully mutated.*

ii. *The 9 labs testing the repeat size reached no consensus on allele 2.*

iii. *Clinically normal.*

iv. *The DNA from this patient was sequences in triple and the average of repeats in the FMR1 gene allele 1 was 118. The repeat length for allele 2 is not stated and no explanation is given.*

v. *Mental retardation. The variant is fully mutated/methylated.*

Sample ID	Disease Type	Repeat Motif	Chromosomal Location	Disease Onset OR Clinical Diagnosis	Patient Age at Sampling	Sex	Repeat Count [Allele 1/Allele 2]
GM04284	HD	CAG	4	27 years	40 years	M	39/50
GM05538	HD	CAG	4	2 years	10 years	M	22/101
GM06891	FXS	CGG	X	Normal[iii]	29 years	M	118[iv]
GM07541	FXS	CGG	X	Normal[iii]	37 years	F	31/29
GM07861	FXS	CGG	X	Affected[v]	25 years	M	351-400
GM20239	FXS	CGG	X	Unaffected[i]	42 years	F	20/183-193[ii]

2.2Targeted Sequencing

For this project, two different ligation and sequencing protocols were used. Ligation sequencing kit SQK-LSK109 (Technologies 2018a) and SQK-LSK110 (Technologies 2020b) was used according to the manufacturer's recommendations with the following modifications: (1) in step 3 of DNA and End-prep, the amount of input DNA was increased from 1 µg to 3 µg if possible, and (2) in step 6 of DNA and End -prep, the DNA repair mix was incubated for 15 minutes at 20°C and 60°C.

Some of the reagents are different between the two kits but the protocol is identical. It is necessary to repair the DNA, prepare the DNA ends for adapter attachment, attach the adapters to the DNA ends, prime the flow cell, and load the DNA library into the flow cell.

The companion module (NEBNext® Companion Module for Oxford Nanopore Technologies® Ligation Sequencing (cat # E7180S)) from NEB (*New England Biolabs*, United Kingdom) was used in the ligation steps, and the compatible flow cell priming kit and wash kit from ONT was used as well.

2.2.1 Needle Shearing

Needle shearing was performed on some samples to break the longest reads (>60kb) due to previous experiences in the lab, suspicion has been raised on whether these long reads can disrupt the sequencing. The idea is that there will be a greater output with shorter fragments, but they should not become so short that they cannot span an extended repeat region.

The samples went through the needle 20 times on average.

2.2.2 DNA Repair and End Preparation

DNA repair was performed to repair nicks of the DNA strand, which can lead to blocking of sequencing pores. For DNA repair, reagents from the sequencing ligation kit (SQK-LSK109 or SQK-LSK110) were mixed in a PCR tube to a final volume of 59 µL according to the manufacturer's recommendations.

As DNA input, 3 µg was diluted with nuclease-free water to a total volume of 49 µL. The tube was mixed gently by flicking and subsequently spun down, followed by incubation for 15 minutes at 20°C, and 15 minutes at 65°C.

2.2.3 Bead Purification Round 1

This clean-up step is important for adapter ligation efficiency. It also reduces the prevalence of chimeric reads and leads to more pores being available for sequencing.

The heat-treated sample was transferred to a LoBind Eppendorf tube, where 59 µL of Agencourt AMPure XP beads was added. The sample was incubated at a Hula Mixer for 5 min at room temperature.

The sample is then briefly centrifuged and put on a magnetic rack for pellet formation. The supernatant is removed, and the pellet is washed twice with 70% ethanol. After letting the pellet dry for 30 seconds it is resuspended in nuclease-free water and incubated at room temperature for 2 min. Then the eluate is moved to a clean LoBind Eppendorf tube.

2.2.4 Adapter Ligation and Bead Purification Round 2

To the LoBind Eppendorf tube containing the purified DNA, the following reagents are added according to the manufacturer's recommendations: ligation buffer (LNB), NEBNext Quick T4 DNA ligase, and the adapter mix (AMX/AMX-F). The tube is incubated at room temperature for 10 min. After incubation, Agencourt AMPure XP beads are added, and Long Fragment Buffer (LFB) is used to wash the sample, as the LFB is designed to enrich for DNA fragments >3kb. The pellet is then resuspended in Elution Buffer (EB) and incubated at 37°C for 10 min to improve the recovery of long fragments. After incubation, the eluate containing the DNA library is moved to a clean LoBind Eppendorf tube.

2.2.5 Qubit Concentration Measurement

After the first and second bead purification, the concentration of the sample is measured using the broad-range kit from Qubit (*ThermoFisher Scientific*, Qubit™ dsDNA BR Assay Kit).

This is a quality step where the last concentration measurement is used to determine how many µL DNA library to use for loading since it depends on the fmoles available in the total sample. The number of DNA molecules (in fmoles) is calculated by giving the mass of the DNA library and the length of the DNA (Biolabs 2020).

2.2.6 Priming and Loading the ONT MinION SpotON Flow Cell

The flow cell arrives with a storage buffer inside which is used to check the number of active pores in the flow cell. This is done with a program module in MinKNOW.

Without introducing air into the flow cell, a large portion of the priming mix is loaded into the priming port. After 5 minutes, the SpotON port is opened and a smaller amount of priming mix is then introduced to the flow cell through the priming port. This pushes air out of the area where the chip is, making it easier to load the DNA library. They recommend loading between 5-50 fmol of DNA from the final library. The volume of the DNA is diluted with elution buffer, which is mixed with sequencing buffer and loading beads. This loading library is then added dropwise in the SpotON port right after the second introduction of the priming mix.

2.2.7 Washing and Re-loading the Flow Cell

The flow cell wash kit (*ONT*, EXP-WSH003) is used to wash out the library either for storing the flow cell or to load a new or more of the same DNA library. The wash kit contains DNase I which digests the remaining DNA in the flow cell. This clears up pores otherwise categorized as recovering or unavailable, improving the sequencing output.

Before washing, the sequencing is paused and the waste channel is emptied. Then without introducing air, the wash mix, prepared according to the manufacturer's recommendations, is loaded in the priming port. After 30 minutes, the waste channel is again emptied, leaving it ready for a new DNA library or storage solution.

2.3 Computational Processes for Targeted Sequencing using ReadFish

ReadFish is the software controlling the adaptive sequencing process and is started when the sequencing has been set-up. To run ReadFish, a conda environment named *readfish_guppy3* needs to be activated, and then the shell file *run_readfish.sh* is called to start ReadFish. The list of packages in the environment is listed in Appendix 2.

The targets are given in a toml file, which contains a panel of short tandem repeat loci (see Github repository, Appendix 1, or Figure 15).

2.3.1 Mosdepth: Target Depth Extraction

To detect the depth at target regions Mosdepth (Pedersen and Quinlan 2018) has been used with standard settings. For details see the GitHub repository for the R script processing the tables from Mosdepth as well as the genome depth calculations.

Mosdepth is installed inside a conda environment with the command:

```
conda install -c bioconda mosdepth
```

The usage of Mosdepth:

```
mosdepth [options] <prefix> <BAM – or – CRAM>
```

Example for GM06891:

> Mosdepth –by ~/regions_file.bed GM06891 ~/GM06891.bam

> cd ~/Mosdepth_folder

> zcat GM06891.regions.bed.gz

Output:

Chr	Start	Stop	Disease	Mean Depth
1	57782716	57882916	SCA37	4.00
3	63848361	63948561	SCA7	7.57
3	128841420	128941620	DM2	4.36
4	3026604	3126770	HD	6.23

See full output in Appendix 4

2.3.2 Fold-Change Plots and Sequencing Illustrations

When making plots showing the fold-change in target depth compared to the mean genome depth, the following formulas have been used, which also can be seen in the R script below that creates the plots.

To get the number of reads mapped, Samtools have been used in the following way:

```
> samtools view -c -F 260 ~/sample.bam
```

$$average\ read\ length = \frac{Estimated\ bases\ generated}{Number\ of Reads\ mapped}$$

$$Mean\ coverage = \frac{number\ of\ reads\ mapped * average\ read\ length}{total\ sequence\ length}$$

Example for GM06891:

$$average\ read\ length = \frac{7{,}380{,}000{,}000\ bp}{13{,}482{,}122\ reads} = 527.29\frac{bp}{read}$$

$$Mean\ coverage = \frac{13{,}482{,}122\ reads * 527.29\frac{bp}{read}}{3{,}101{,}788{,}170\ bp} = 2.29\ bp$$

Then the Mosdepth output of the regions.bed file is loaded into R and the mean depth of the targets is divided by the genome depth to get the fold change.

$$GM06891_{targets\$fold_{chang}} = c(GM06891\_targets\$Mean\_Depth/GM06891\_genome\_depth)$$

Then the column with the disease and the fold-change is subsetted into a new table and the columns are labeled x and y, respectively. This subset table is used with ggplot to create a figure.

R plotting example with GM06891 (experiment 1):

```
ggplot(GM06891_targets_subset, aes(x=x, y=y)) +
  geom_segment( aes(x=x, xend=x, y=0, yend=y), color="skyblue") +
  geom_point( color="blue", size=4, alpha=0.6) +
  theme_light() +
  coord_flip(ylim=c(0,5)) +
  theme(panel.grid.major.y = element_blank(),
    panel.border = element_blank(),
    axis.ticks.y = element_blank()) +
  labs(x = "", y = "Fold-Change") +
  ggtitle("Experiment 1") +
  theme(plot.title = element_text(hjust = 0.4))
```

For visualization of aligned reads, Integrative Genomics Viewer (IGV) has been used to show the sequencing depth in and around targets.

MinKNOW produces a report with several figures, which are used as well to explain the effect of ReadFish.

2.4 Process of Nanopore Data

Some basic bioinformatic processes have been used such as basecalling, alignment, and merging of files.

2.4.1 Basecalling

Basecalling is the process where the raw files from Nanopore Sequencing are being translated into bases (ATCG).

For the whole-genome data, the sequencing software MinKNOW has basecalling enabled. MinKNOW uses Guppy as a basecaller.

For targeted sequencing basecalling was used for Readfish and the reads were not fully basecalled and stored. Basecalling was done separately after sequencing. Below is an example with HAC basecalling.

```
> guppy_basecaller -i ${workdir}/fast5 -s ${workdir}/fastq/ -c dna_r9.4.1_450bps_hac.cfg -x "cuda:all"
```

2.4.2 Alignment

Alignment is the process where the basecalled data is lined up against a known reference genome. This has been done for both the whole-genome data and the targeted data.

For alignment, Minimap2(Li 2018) has been used with parameter presets for Oxford Nanopore genomic reads:

```
> minimap2 -ax map -ont ref.fa ont.fq.gz > aln.sam
```

Samtools was used to convert sam files to bam files.

```
> samtools sort input -sam -o output.bam
```

```
> samtools index output.bam
```

2.4.3 Merging Files

WG sequencing and targeted sequencing data have been merged into combined data for further analyses. To gather fast5 files, the tool *mv* was used to collect all the fast5 files in one folder.

For joined fastq files, the combined fast5 folders were basecalled to fastq files. Otherwise, *cat* can be used to join existing fastq files.

The tool *Samtools merge* was used to join bam files.

2.5 Illumina-Guided Haplotype Phasing

An in-house haplotype phasing method called "Illumina" has been developed. Below is a description of how it works.

To phase the reads, we chose to use variants identified by Illumina genome sequencing of the samples to make sure to use accurate SNP positions for haplotype phasing. The method works in two parts: (1) constructing a data frame for each sample with the bases/nucleotides of all the relevant SNP positions from all the reads of that sample, and (2) the creation of haplotypes and haplotype assignment of the reads.

2.5.1 Part 1: Creating Data Frames with SNP Information

To make sure that the tables used for establishing haplotypes contain enough information, a window of 40,000 bp (+/-) extending from the target was used for SNP detection in combination with an allele frequency (AF) of 0.3-0.7. An AF of 0.3 means that at least 30 % of the reads are different from the reference. The interval of 0.3-0.7 was chosen to make sure to find heterozygote SNPs. An AF of 0.9 could easily be a homozygote variation with a few sequencing errors. When the AF is 0 there is no variation from the reference, and when the AF is 1 there is a homozygous variant, and an AF between 0 and 1 is indicative of a heterozygous variant. A minimum depth of 10 reads was chosen to make sure that there was enough coverage to trust the SNP selection. The DP is the depth of the variant, meaning the number of reads covering the chromosomal position where the variant is found.

A larger window than 40,000 bp only gives SNPs further away from the target. A window of 40,000 bp was chosen to make sure that all relevant reads were included. However the reads are not expected to extend more than 20,000 kb away from the target. When creating the haplotypes, the closest SNPs are of most importance since we use these SNPs as a way of filtering the reads (see more below). Another reason for choosing a window of 40,000 is that there is only a very small SNP gain when increasing the window and unfortunately there is no SNP gain around HTT for samples GM04284 and GM07861, which are the ones which are difficult to haplotype phase due to a low number of SNPs.

For part 1, a shell and python script has been written and run.
To run the shell and python script the following command was run in the terminal. The shell script calls on the python script.

```
Bash /nanoporeData/kommandoer/Illumina_Guided_HP_FMR1_and_HTT_3.sh
```

The shell script: Extraction and Filtering of Illumina SNP Genotype Information

The shell script contains four global variables: *window*, *AF_low*, *AF_high*, and *DP*.
To go through both the HTT and FMR1 gene, an if-else statement has been created to save values in the variables *start_position, end_position,* and *chromosome*. The start position is the location for the start of the repeat minus the window, and the end position is the location for the end of the repeat plus the window. These three variables are used when calling SAMtools to make a subset of the BAM files for the samples such that the BAM file only contains reads ranging from *start_position* to *end_position*. Then the read-names, as well as the chromosomal position from that BAM file, is saved in a text file.

The *start_position* and *end_position* are also used to subset the Illumina VCF file, which is done with BCFtools. Then the *AF_low*, *AF_high*, and *DP* are used to filter the VCF subset. From this filtered vcf, the Illumina genotypes are being extracted for the SNPs chosen. The genotypes consist of a reference and an altered base. The Illumina genotypes are saved as a text file. A small example is given in table 2 below.

Table 2: Illumina Genotypes for the In-House Haplotype Phasing Method

This table gives a small presentation of the Illumina genotype data frames. The chromosome, position, the reference base, and the altered base is given.

CHROM	POS	REF	ALT
4	3036630	T	C
4	3038112	T	G
4	3038415	A	G
4	3038551	T	C

The python script: Extraction of ONT Genotype Information at Illumina SNP positions

First, the SNP positions from the filtered Illumina VCF file are loaded into a set, and the read-names are loaded into a list. Then an empty data frame is created with dimensions equal to the number of SNPs and number of reads. Next, the data frame is filled with bases corresponding to the specific read-name at the specific SNP position. This is done by using a pileup on the ONT BAM file. To finish it off, all columns and rows only containing NAs are removed as they are not contributing with any information for haplotype phasing. The NAs appear when the read is not covering the SNP position. The resulting data frame is saved as a text file whereof a small example is given below in table 3.

Table 3: Data Frame Example of the SNPs for the ONT DNA Reads

This table gives a small presentation of how the resulting data frame from part 1 of the in-house Illumina haplotype phasing method. The hyphen is used where there are deletions and NA is used where there is missing information.

Read-IDs	**SNP Positions**			
	3041938	3043512	3043513	3050925
ec0c2dd1-5283-4f2a-b401-ccf5ccda1281	C	C	A	-
d48315a7-479f-4612-aea8-19e8fe5a7ce2	T	G	C	-
fda80971-40db-4fd5-a1a9-de12952592a4	T	G	A	NA
276d7de3-d8bd-4352-8ae5-81552dc4301c	C	C	A	NA
66e35965-020b-4568-87d2-b1b822e42815	T	G	C	NA
cbc13953-4f33-4482-86be-775e98e90398	C	NA	NA	NA
f51b128f-7c95-4f65-bd12-ff8f2c5740b0	-	C	A	C
04e1de4d-002f-4457-92fc-e53a60593701	T	G	C	T
62d44a0d-f2f1-41da-9fb4-803c0d163785	T	G	C	T
2e31f898-22bc-45a7-bd9a-45df2bf434cf	C	C	A	C
2d1ad48c-ab35-4a99-8b5c-fe0ca752d905	G	C	A	C

2.5.2 Part 2: Creating and Assigning Haplotypes

The next step is the actual haplotype phasing, where reads containing one of the closest SNVs on either side of the repeat are used for establishing the haplotypes. All the reads that cannot be used for establishing the haplotypes i.e., reads covering only one SNP, are attempted to be recovered by matching them to the established haplotypes.

The R script: Haplotype Creation and Read Assignment

Nine helper functions have been made in the script: (1) Get_gene_information, (2) remove_distant_reads, (3) remove_uniformative_reads, (4) num_reads_cover_both_snps, (5) gower_distance, (6) hierarchical_clustering, (7) construct_haplotype, (8) get_statistics, and (9) check_snp_in_hap.

The main pipeline of phasing the resulting data frames from the python script consists of the following steps:

1. The input files are loaded into a list of files. For every file (data frame) in the list, the sample name and the gene are saved in the variables *sample* and *gene*. Then the *Get_gene_information* function is called to retrieve the chromosomal start- and end position.
2. The SNV coordinates are extracted for the sample around the gene of interest. The closest SNP on the left and right (upstream and downstream) from the repeat are saved in the variables 'snp_left' and 'snp_right'. Then all reads not covering both of these variants are removed, as two SNPs are necessary for HP. This is done with the helper functions *remove_distant_reads* and *remove_uniformative_reads*.
3. Then the genotypes for the Illumina variants are loaded into a table called genotypes.
4. There are two scenarios for phasing. Either we have reads covering the closest SNP on both sides of the repeat or we have reads that do not. This comes down to how long the reads are and how far from the repeat that the closest SNP is. To check which of the scenarios we have we use the helper function *num_reads_cover_both_snps*. If reads exist on both sides (scenario 1), then the reads are clustered into two groups by using the *hierarchial_clustering* function, and the two most likely haplotypes are constructed by the *construct_haplotype* function. If no reads are covering both sides of the repeat (scenario 2), then the data set is split into two sets containing reads on each side. Then both sets/sides are clustered, and the haplotypes are constructed for both sides. This leaves us with phased reads on both sides of the reads and with scenario two where it can be difficult to correlate the repeat separated haplotypes.

5. Then all the reads that only covered one SNP are recovered and clustered. The read is recovered if it contains SNPs that agrees with one of the haplotypes.
6. The output files are written out to a local folder. The output consists of (1) a file showing the haplotypes, and (2) a file with the read IDs and their haplotype assignment. Below are examples of output files for sample GM06891 for HTT.

read_ids	haplotype	origin
c20360cf-8732-4fd8-a9e9-807d4f2d17d3	hap1	hap1
ea15d0f0-6191-4081-837f-54c39f6c249f	hap1	hap1
bdf99f7a-8c29-4f3a-8d27-e01e9159315d	hap2	hap2
10084b0e-e6c1-41ac-9676-78200ce7985f	hap2	hap2
29b0255e-5e3d-4c27-8e45-9acbccf27303	hap1	hap1
bbe49fdd-3230-4cfe-a8b2-5140f6e46f7f	hap1	hap1
d165cfa8-af28-427b-8832-12ef87944a49	hap1_left	hap1_left_recovered
82473b02-cd20-4067-8d82-0671892166cd	hap2_left	hap2_left_recovered
53efe9a6-5cc5-4075-8594-06e6ab6ac1bd	hap1_right	hap1_right_recovered
1ffa75af-c228-4f93-8f4c-788c3ce25f32	hap1_right	hap1_right_recovered

Haplotype 1:

C C NA C A A G C/T G A A A C G C G C T A T C T C A C A/G T G G C T A A G G G G T G C A G C C A G T G A G C T A A G G T A C C G A NA NA NA NA NA NA NA NA

Haplotype 2:

NA NA NA NA NA NA A C A G G G G A T T T G G C T A T G T G C A A C C G G C A C A C A G G A T T G A A A G A G C G A A A C T NA NA NA NA NA NA NA NA NA NA NA NA

2.6 Longshot Haplotype Phasing

Longshot is a variant caller that takes BAM files as input and outputs a phased VCF file with variant information. Longshot can also output a BAM file with haplotype tags assigned to the reads that the program successfully phased.

Example on LongShot usage:

```
longshot -r 4 --bam ~/sample.bam  --ref ~/reference.fasta \
--out_bam ~/sample_chr4_longshot_hp.bam \
--min_cov 4 --min_alt_count 2 --out ~/sample_chr4_longshot_hp.vcf
```

- Parameters chosen/adapted:
  - -r, --region: Region in format <chrom> or <chrom:start-stop> in which to call variants.
  - -c, --min_cov: Minimum coverage (of reads passing filters) to consider position as a potential SNP. [default: 6]
  - -e, --min_alt_count: Require a potential SNP to have at least this many alternate allele observations. [default: 3]

The parameters are adapted due to low coverage in some samples.

When running Longshot every sample will have an HP-tagged BAM file and a VCF file as output for each ROI.

2.7 Create Haplotype Phased BAM files

For the programs used to count the repeat extension, haplotype phased BAM reads are either useful or necessary to compare the two haplotype phasing methods.

STRique is the only count program that outputs the read-names, making it possible to phase the results from the program. Repeat-HMM and Tandem-Genotypes do not include the read-names in their output making it important to input phased bam-files such that the results can be analyzed with phases. TRiCoLOR needs haplotype-phased bam-files as the input making this step necessary.

The Longshot BAM file with haplotype tags needs to be split into two BAM files with one for each haplotype. The output files from the in-house Illumina-guided HP with read IDs and haplotype tags are used for making new BAM files.

2.7.1 Creation of LongShot Haplotype Phased BAM files

The BAM-file produced by LongShot which contains HP-tags needs to be indexed. This is done with *Samtools* (see example below). Then the BAM-file is subsetted with *Samtools* to region 4:3076604-3076666 or X:146993569-146993628, which is the repeat region for HD and FXS. This results in a BAM-file with the reads that span the repeat. This BAM-subset is indexed as well. Next, the read IDs are pulled out of the BAM subset by cutting field 1 and 22 out of the BAM-file and piping it into a text-file. Then a text-file is created with read names belonging to haplotype 1, and another text-file is created with read names belonging to haplotype 2. This is done by using *Awk* to look for "HP:i:1" for haplotype 1 and "HP:i:2" for haplotype 2. Then the resulting rows from the *Awk* search are cut such that only field 1 with the read name is saved into a new text-file. Then grep is used to pull out all the read names from the text-file with haplotype 1 read names along with the BAM header to create a new BAM-file. Finally, the new HP BAM-file is indexed.

Example of creating a BAM-file with reads belonging to LongShot haplotype 1 for chr4:

```
# Index the output bam from LongShot
samtools index ~/sample_combined_longshot_hp_chr4.bam

# Subset the output bam from LongShot
samtools view -bh  ~/sample_combined_longshot_hp_chr4.bam "4:3076604-3076666" \
> ~/sample_combined_longshot_hp_chr4_subset.bam

# Index the subset bam
samtools ~/sample_combined_longshot_hp_chr4_subset.bam

# Pull out the hp-tag from the subset bam along with the read-names
samtools view ~/sample_combined_longshot_hp_chr4_subset.bam | cut -f 1,22 > \
~/sample_combined_longshot_hp_chr4_readnames.txt

# Create txt files with readnames belonging to hp1
awk '/HP:i:1/' ~/sample_combined_longshot_hp_chr4_readnames.txt | cut -f 1 > \
~/sample_combined_longshot_hp1_chr4_readnames.txt

# Create a new bam-file with haplotype 1
samtools view -h ~/combined_data_sample.bam | grep -e '^@' -f \
~/sample_combined_longshot_hp1_chr4_readnames.txt | \
samtools view -bS - > ~/sample_combined_HTT_longshot_hp1.bam

# Index the new hp1 bam
samtools index ~/sample_combined_HTT_longshot_hp1.bam
```

2.7.2 Creation of In-House Illumina-Guided Haplotype Phased BAM files

In R, the read names and their haplotype-tag are saved in a text-file. This file needs to be transformed into a tabs-separated file instead of a space-separated file. This transformation is done with *cat* and *tr* (see example below).

Then *awk* is used to create two new text-files, each with the read names belonging to a haplotype.

Again *Samtools* is used to create two new BAM-files, each with the reads belonging to a haplotype, which as a final step is indexed.

Example of creating a BAM-file with reads belonging to in-house Illumina haplotype 1 for chr4:

```
# transform space separates files to tabs separated files
cat ~/sample_HTT_reads.txt  | tr ' ' '\t' >  ~/sample_HTT_reads_tabs.txt

# Create txt files with readnames belonging to hp1
awk '/hap1/' ~/sample_HTT_reads_tabs.txt | cut -f 1 > \
~/sample_combined_illumina_hp1_chr4_readnames.txt

# Create a new bam-file with haplotype 1
samtools view -h ~/sample_combined.bam | grep -e '^@' -f \
~/sample_combined_illumina_hp1_chr4_readnames.txt | samtools view -bS > \
~/sample_combined_HTT_illumina_hp1.bam

# Index the new hp1 bam
samtools index ~/sample_combined_HTT_illumina_hp1.bam
```

2.8 Programs for Repeat Length Estimation

Four programs are used to estimate repeat lengths for HD and FXs: RepeatHMM, STRique, TandemGenotypes, and TRiCoLOR.

2.8.1 RepeatHMM

RepeatHMM uses a combination of strategies and tools to estimate repeat expansion lengths. One strategy used is split-and-align which improves alignments as well as error correction, where a specific one for ONT sequenced data is available. To detect the repeats, RepeatHMM utilizes a Hidden Markov Model (HMM) and a peak calling algorithm based on a Gaussian mixture model (Liu et al. 2017).

This program outputs an array of repeat length as well as the number of reads covering this repeat length. From this array, the program tries to find an optimal Gaussian model. A warning is given if it cannot. For FMR1 RepeatHMM might filter the estimation at the last step due to the tendency of having very few reads with the same count. In these cases, the output is something like [0,0]. When making plots for this results section only the array is used and not the estimated repeat length on each allele.

Example for RepeatHMM usage:

 # No HP

```
python repeatHMM.py BAMinput --Onebamfile ~/sample_targeted_combined.bam  \
--hg hg19 --hgfile  ~/resources/human_g1k_v37_decoy.fasta \
--repeatName HTT --MinSup 1 --FlankLength 150 --SeqTech Nanopore \
--UserDefinedUniqID Sample_HTT_Combined_Unphased;
```

 # HTT HP1 LongShot

```
python repeatHMM.py BAMinput --Onebamfile ~/GM06891_combined_HTT_longshot_hp1.bam  \
--hg hg19 --hgfile  ~/resources/human_g1k_v37_decoy.fasta \
--repeatName HTT --MinSup 1 --FlankLength 150 --SeqTech Nanopore \
--UserDefinedUniqID Sample_HTT_Combined_Longshot_hp1;
```

 # Sample HTT HP2 LongShot

```
python repeatHMM.py BAMinput --Onebamfile ~/GM06891_combined_HTT_longshot_hp2.bam \
--hg hg19 --hgfile  ~/resources/human_g1k_v37_decoy.fasta \
--repeatName HTT --MinSup 1 --FlankLength 150 --SeqTech Nanopore \
--UserDefinedUniqID Sample_HTT_Combined_Longshot_hp2;
```

 # Sample HTT HP1 Illumina

```
python repeatHMM.py BAMinput --Onebamfile ~/GM06891_combined_HTT_illumina_hp1.bam  \
--hg hg19 --hgfile  ~/resources/b37/human_g1k_v37_decoy.fasta \
--repeatName HTT --MinSup 1 --FlankLength 150 --SeqTech Nanopore \
--UserDefinedUniqID Sample_HTT_Combined_illumina_hp1;
```

 # Sample HTT HP2 Illumina

```
python repeatHMM.py BAMinput --Onebamfile ~/GM06891_combined_HTT_illumina_hp2.bam \
--hg hg19 --hgfile  ~/resources/b37/human_g1k_v37_decoy.fasta \
--repeatName HTT --MinSup 1 --FlankLength 150 --SeqTech Nanopore \
--UserDefinedUniqID Sample_HTT_Combined_illumina_hp2;
```

2.8.2 STRique

Short Tandem Repeat Identification, Quantification, and Evaluation (STRique) works in two phases, where the first phase is signal alignment, and the second phase is repeat quantification (Giesselmann et al. 2019).

A read which spans the repeat location is identified by aligning the conventional base-called sequence to a reference. This mapping includes a signal alignment algorithm. When the sequence is mapped a hidden Markov Model (HMM) is used on the raw fast5 data, also known as squiggles, to quantify the number of repeats (Giesselmann et al. 2019). The result files have several parameters, whereof the target ID, count, prefix, and suffix score and methylation are of interest in this project.

STRique is used through a docker. To use STRique fofn-files are created, which are index files over the squiggles in the fast5 files. A configuration file is needed with the following headers: chr, begin, end, name, repeat, prefix, suffix. Then the python script is run like the example below.

Examle for STRique usage:

 # Activate udocker

```
udocker run -v /work:/work strique /bin/bash
```

 # Create fofn files

```
python3 scripts/STRique.py index ~/fast5/sample/ > ~/fast5/sample/sample.fofn
```

 # Use the STRique python script to estimate repeat lengths

```
cat ~/sample.sam | python3 /app/scripts/STRique.py count \
~/fast5/sample_targeted.fofn /app/models/r9_4_450bps.model \
~/strique_config.tsv --config /app/configs/STRique.json \
--mod_model /app/models/r9_4_450bps_mCpG.model > \
~/strique/sample_count_results.txt
```

2.8.3 Tandem-Genotypes

Tandem-Genotypes detects changes in repeat length compared to the reference. The output is the number of changed repeats, meaning that the reference repeat count must be added for the total repeat

length. Tandem genotypes align and prioritize the reads in multiple steps and estimate the repeat count from insertion sizes (Mitsuhashi et al. 2019).

Before running Tandem-Genotypes, the reference genome and the sequenced data should be prepared according to the guidelines given in their GitHub repository. *Windowmasker* is used to mask repeats in the reference genome, as it reduces the computation time and memory when aligning. *Lastdb* is used to index the reference genome. *Last-train* is used to determine the rates of insertion, deletion, and substitution between the sample reads and the reference genome. A subset of the fastq files is used to limit processing time. The output is par-files. Then *lastal* is used to align the reads, with maf-files as output. Now Tandem-Genotypes can be used together with a text-file giving the repeats of interest. See the examples below.

Example on preparing the reference genome for usage of Tandem-Genotypes:

```
# Preparing a reference genome with repeat-masking
windowmasker -mk_counts -in ~/Homo_sapiens_assembly19.fasta \
> ~/hg19_homo_sapiens_assembly19.wmstat

windowmasker -ustat ~/hg19_homo_sapiens_assembly19.wmstat \
 -outfmt fasta -in ~/Homo_sapiens_assembly19.fasta \
>  ~/Homo_sapiens_assembly19_wm.fasta

# Index the genome
lastdb -P8 -uNEAR -R11 -c mydb ~/Homo_sapiens_assembly19_wm.fasta
```

Example on alignment of LongShot haplotype phased files:

 # Determine rates of insertions, deletions and substitutions

```
last-train -P24 -Q0 mydb ~/sample_combined_chr4_hp1_longshot.fastq.gz \
> ~/samlpe_combined_chr4_hp1_longshot.par
```

 # Aligning DNA sequences

```
lastal -P24 -p ~/sample_combined_chr4_hp1_longshot.par mydb \
~/sample_combined_chr4_hp1_longshot.fastq.gz | last-split -fMAF | gzip \
> ~/sample_combined_chr4_hp1_longshot.maf.gz
```

Example for Tandem-Genotypes usage:

```
tandem-genotypes microsat.txt ~/sample_combined_chr4_hp1_longshot.maf.gz \
> ~/sample_combined_chr4_hp1_longshot_results.txt
```

2.8.4 TRiCoLOR

Tandem Repeats Profiler for Long-Read Sequencing Data (TRiCoLOR) (Bolognini et al. 2020) takes HP files as input and is therefore discussed in this section.

TRiCoLOR can detect *de novo* repetitive regions and estimate the repeat length for the two alleles. Repetitive regions are detected as low entropy regions, and partial order alignment is used to compute haplotype-specific consensus sequences. A fast regular expression-based approximate string-matching algorithm is then used to find the repeat motif and estimate the number of repeats (Bolognini et al. 2020).

TRiCoLOR contains four modules of which the following two have been used: (1) a de novo repetitive regions identifier called the *Shannon Entropy Scanner* (SENSoR), and (2) a repeat length estimator called the *Repeats Finder* (REFER).

Example on TRiCoLOR usage on LongShot haplotype phased files to estimate HTT repeat lengths:

 # SENSOR

```
TRiCoLOR SENSoR -bam sample_combined_HTT_longshot_hp1.bam \
sample_combined_HTT_longshot_hp2.bam --chromosomes 4 -c 5 -l 15 \
-id 100 -od 100 -s 15 -e 1.23 -o ~/TRiCoLOR/sample_SENSoR_chr4_Longshot
```

 # REFER

```
grep -P "^4\t3075|^4\t3076|^X\t14698" ~/TRiCoLOR/sample_SENSoR_Longshot/TRiCoLOR.srt.bed > ~/TRi-
CoLOR/GM06891_SENSoR_Longshot/TRiCoLOR.srt.fltrd.bed

TRiCoLOR REFER -g ~/resources/b37/human_g1k_v37_decoy.fasta \
 -bam  sample_HTT_Longshot_HP_1.bam sample_HTT_Longshot_HP_2.bam \
-bed ~/TRiCoLOR/sample_SENSoR_Longshot/TRiCoLOR.srt.fltrd.bed -s 20 \
-o ~/TRiCoLOR/sample_REFER_Longshot
```

2.9Logos Plots

Logos plots are created as a quality control of haplotype phasing, which is compared to how well the repeat lengths are separated by the repeat estimating programs.

Logos plots are a specific way of showing consensus sequences by giving a visual image of the entropy at the SNP positions. The consensus sequence is a representation of the most frequent residues at specific positions in a sequence alignment. (Schneider and Stephens 1990). In logos plots, the entropy is contained in a measurement called bits, which is seen on the y-axis. The height of the letter stack at each position represents the information content measured in bits at that position. Positions with high entropy will have low stacks and positions with low entropy will have high stacks. Therefore, the stack height is proportional to the relative entropy. The maximum of bits available at a position is two. AT homozygous SNPs 2 bits are expected unless there are a lot of reads with NAs. The height of each nucleotide in a specific position in the logos plot is proportional to each other showing their frequency at that specific SNV, having the most frequent nucleotide on top. Two SNV positions can have the same nucleotide distribution but have a different stacking height due to entropy (bits) (Wagih 2017; Schneider and Stephens 1990). Higher stacks should be better for haplotype phasing as they contain more information.

Before phasing the reads, the sequence logos will be full of noise, as both alleles will be presented. The unphased sequence logo of both haplotypes will show which SNPs are noisy as we only expect to see two nucleotides stacked at each SNP position.

The logos plots of the phased reads will show the noise on the two alleles separately and can be a picture of how noisy ONT sequencing is.

For LongShot logos plots, the first step is to adapt the Illumina-Guided method part 1 into taking LongShot VCF files, as the resulting data frames with bases for the filtered SNP positions and read IDs are needed. Examples of logos plots can be seen in Appendix 7.

2.9.1 Logos Plot Creation in R

These data frames are loaded into R and NAs and deletions (indicated by -) are exchanged with zeros. Then an empty character vector is made for sequences to be added onto. The sequence being added i s the combination of bases for all the SNPs for each read. For example, "ACG0TGGCGCA0GG00A 0". This character vector is used to create the logos plot containing both haplotypes, also called the t otal logos plot in this dissertation.

Then the tables with read names and haplotype-tags are loaded into R, where they are subs etted into two data frames, one for each haplotype. These two data frames with read names and hapl otype tags are used to subset the SNP data frame by using the function *merge()*. This leaves us with t wo data frames with SNP information for the reads belonging to either haplotype 1 or 2. Then new c haracter vectors are made, and sequences added onto as done for the total data and new logos plots a re made. See the example below.

LongShot Logos Plot Creation Example:

 # Load Libraries

```
require(ggplot2)

require(ggseqlogo)
```

 # Loading the table

```
GM06891_HTT <- read.table("longshot_GM06891_HTT_het_bases_window40000_DP10.txt", header = TRUE, sep = ",")

colnames(GM06891_HTT)[1] <- c("read_id")
```

 # Create tables with zeros instead of NA and deletions (-)

```
GM06891_HTT[is.na(GM06891_HTT)] <- 0

GM06891_HTT[GM06891_HTT == "-"] <- 0
```

 # Make a character vector for the sequences to be added to

```
Character_vector_GM06891_HTT = character(nrow(GM06891_HTT))
```

 # Add the sequences to the character vector

```
i <- 1

while(i < (nrow(GM06891_HTT)+1)) {

  Character_vector_GM06891_HTT[i] <- paste(c(as.character(as.character(GM06891_HTT[i, 2:ncol(GM06891_HTT)]))),collapse = "")

  i <- i + 1

}
```

 # Create the total logos plot logo

```
ggseqlogo(Character_vector_GM06891_HTT, method = 'bits', seq_type = 'dna')
```

 # Load Read-Names for the alleles

```
GM06891_HTT_readnames <- read.table("GM06891_longshot_hp_chr4_readnames.txt", header = F)

colnames(GM06891_HTT_readnames) <- c("read_id","hp_tag")
```

Example continued

 # Subset Longshot data frames into haplotypes

```
GM06891_HTT_longshot_hp1 <- subset(GM06891_HTT_readnames[GM06891_HTT_readnames$hp_tag == "HP:i:1",])

GM06891_HTT_longshot_hp2 <- subset(GM06891_HTT_readnames[GM06891_HTT_readnames$hp_tag == "HP:i:2",])
```

 # Create subset of the data frames for allele logos plots

```
GM06891_HTT_allele_1 <- merge(GM06891_HTT, GM06891_HTT_longshot_hp1)

GM06891_HTT_allele_2 <- merge(GM06891_HTT, GM06891_HTT_longshot_hp2)
```

 # Make a character vector for the sequences to be added to for the allele logos plots

```
Character_vector_GM06891_HTT_allele_1 = character(nrow(GM06891_HTT_allele_1))

Character_vector_GM06891_HTT_allele_2 = character(nrow(GM06891_HTT_allele_2))
```

 # Add the sequences to the character vector

```
i <- 1

while(i < (nrow(GM06891_HTT_allele_1)+1)) {

  Character_vector_GM06891_HTT_allele_1[i] <- paste(c(as.character(as.character(GM06891_HTT_allele_1[i, 2:ncol(GM06891_HTT_allele_1)]))),collapse = "")

  i <- i + 1

}

i <- 1

while(i < (nrow(GM06891_HTT_allele_2)+1)) {

  Character_vector_GM06891_HTT_allele_2[i] <- paste(c(as.character(as.character(GM06891_HTT_allele_2[i, 2:ncol(GM06891_HTT_allele_2)]))),collapse = "")

  i <- i + 1

}
```

 # Create the sequence logo

```
ggseqlogo(Character_vector_GM06891_HTT_allele_1, method = 'bits', seq_type = 'dna')

ggseqlogo(Character_vector_GM06891_HTT_allele_2, method = 'bits', seq_type = 'dna')
```

3 Results & Discussion

3.1 Adaptive Targeted Sequencing

Targeted sequencing was performed to increase the sequencing depth compared to the whole-genome sequencing performed in an earlier study by a former student. The theory is, that a greater depth at target will provide greater precision of results from the count programs and haplotype phasing. To perform adaptive targeted sequencing a program called ReadFish (Payne et al. 2020b) is enabled during the run. When running a Nanopore MinION experiment with ReadFish enabled there is expected a greater depth at the target loci and a lower depth everywhere else.

The panel used in the adaptive targeted enrichment experiments consists of 27 STRs, which are associated with diseases. The repeat location associated with the following diseases are being enriched in the targeted sequencing experiments: HD, SBMA, SCA1, SCA2, SCA3, SCA6, SCA7. SCA17, DRPLA, HDL2, FRAXA, FRAXE, DM1, FRDA, DM2, FTDALS1, SCA36, SCA10, EPM1, SCA12, SCA8, SCA31, FAME1, FECD3, OPMD, and EIEE1.

Targeted sequencing could be used clinically for the detection of known diseases caused by extensions of STRs. For this, a target panel of repeat diseases is useful. It is also advantageous for this enrichment analysis to have more than two repeat diseases (HD and FXS) since two target regions can be subjected to randomness in a sample. However, only repeat lengths in HD and FXS are analyzed.

Table 4: Targeted Sequencing Experiment Overview.

Four different experiments were performed with five adjustable parameters. The ligation sequencing kit was changed from SQK-LSK109 to SQK-LSK110 in experiment 4. Needle shearing was applied in experiments 3 and 4. The small target region was chosen in experiments 1 and 2. The large target region was chosen in experiments 3 and 4. The fast version of Guppy was applied in experiment 1 and the HAC version of Guppy was applied in experiment 2-4. Washing of the flow cell was not performed for experiment 2 due to the suspicion of not having enough material for loading since there have been practical problems in loading. The green text indicates the changes from the prior experiment, meaning that the parameter values for experiment 1 are set as default as this is the baseline, we worked from trying to improve targeted sequencing.

	Sample ID	Ligation Sequencing Kit	Needle Shearing	Target Region	Guppy Config File	Wash
Experiment 1	GM06891	SQK-LSK109	No	51,000 bp	Fast	Yes
Experiment 2	GM07541	SQK-LSK109	No	51,000 bp	**HAC**	**No**
Experiment 3	GM20239	SQK-LSK109	**Yes**	**1,000,000 bp**	HAC	**Yes**
Experiment 4	GM07861	**SQK-LSK110**	Yes	1,000,000 bp	HAC	Yes

Four experiments were performed where five parameters were changed in-between experiments: (1) the basecalling configuration file: fast or high accuracy (HAC) basecalling, (2) the size of the target region given to ReadFish, (3) the use of needle shearing, (4) the ligation sequencing kit used, and (5) washing of the flow cell. Table 4 summarizes the five parameters and differences in experiments. The parameters were adjusted in-between experiments in an attempt to increase the sequencing depth at the target regions. The four experiments are described below.

When comparing the experiments in the following subsections, it is important to note that the differences in enrichment could be due to sample variation and not the parameter variations, as four different samples from four different individuals have been used.

Experiment overview:

- **Experiment 1:** The first sample (GM06891) was prepared with kit SQK-LSK109. When running the experiment, the small target region was chosen as well as the fast version of Guppy.

- **Experiment 2:** This sample (GM07541) was also prepared with kit SQK-LSK109. The small target region was also selected but the HAC version of Guppy was chosen as the time it took to reject the reads with the fast version of Guppy were far below the threshold of 0.4 seconds, which ReadFish requires.

- **Experiment 3:** This sample (GM20239) was also prepared with kit SQK-LSK109 but was also exposed to needle shearing in an attempt to break some of the long DNA strands, which for all the samples have shown peaks at >60.000 bp with a Tapestation analysis. Unfortunately, the Tapestation did still show a peak at >60.000 bp after the needle shearing. The large target region was chosen for this run as well as the HAC version of Guppy as the time it took to reject the reads in experiment two were below 0.4 seconds.

- **Experiment 4:** This sample (GM07861) was prepared with a brand-new kit called SQK-LSK110 and were exposed to needle shearing. The large target region as well as the HAC version of Guppy were chosen.

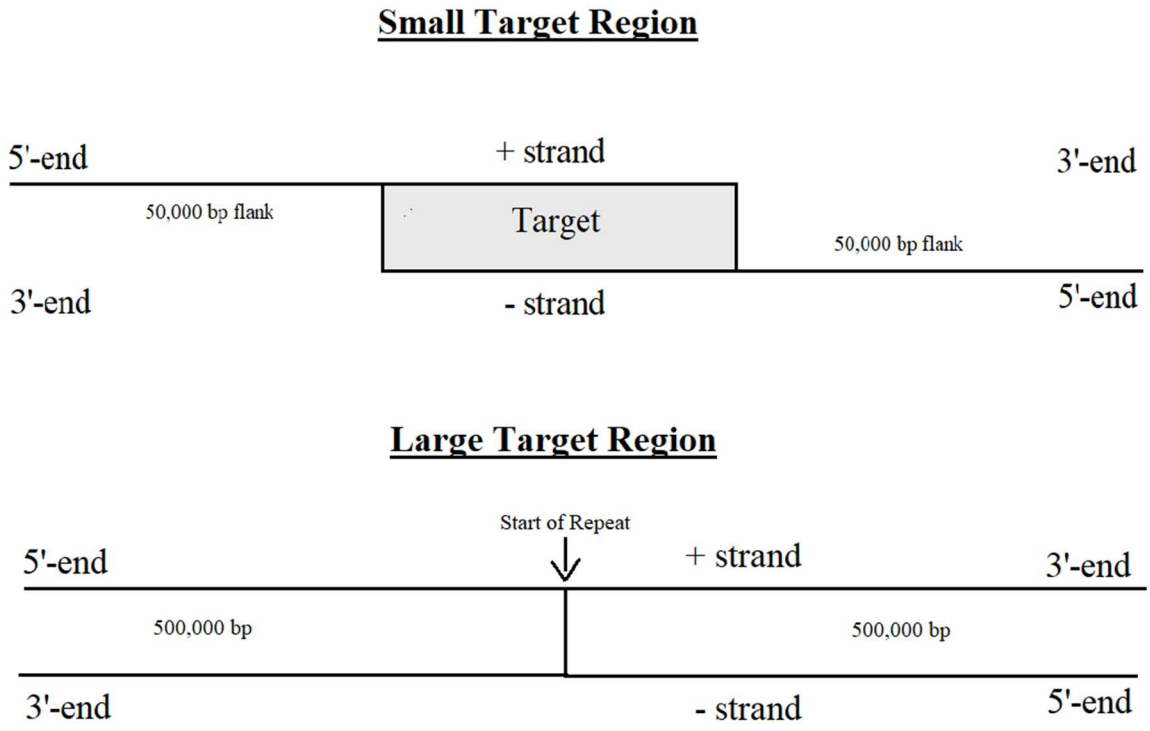


Figure 12: The Target Regions for ReadFish.
The small target regions include the repeat and a 50,000 bp flank on each side of the target repeat. The large target regions contain 500,000 bp on either side of the start location of the target repeat on both strands.

The target regions are chromosomal locations given in a file to ReadFish. A small and a large target region of 51,000 bp and 1,000,000 bp, respectively, were chosen for each target in the target panel,

where the target is the center (see Figure 12). The small target regions have a total size of ~2,700,000 bp and the large target regions have a total size of ~27,000,000 bp. The target panel is illustrated in Figure 12.

The big difference between the small and the large target region besides the size is that the small target region has different chromosomal positions given for the plus- and the minus-strand, where the large target region has the positions given for the two strands (see Figure 12). For the small target region, an area covering the target repeat was chosen with an additional 50,000 bp towards the 5'-end of both strands. This means that a target would have flanks on either side such that the beginning of reads located before the target would get sequences but reads starting after the repeat would not (see the file in Github repository, Appendix 1).

The large target was chosen for experiments 3 and 4 to test if a larger total target size would result in greater output, as a theory is that a small total target region would ruin the pores faster than a larger one due to a higher frequency of rejecting reads.

3.1.1 ReadFish for Targeted Sequencing

When starting ReadFish the pore occupancy drops, which can be seen in the channel states panel (see Figure 13). When starting ReadFish, the pore occupancy drops. This is due to the quick rejection of reads, which also leads to a greater number of recovering pores compared to runs without ReadFish enabled. The two channel state panels in Figure 13 are from different runs but are representative of how the sequencing reacts to ReadFish. The run shown in the panel to the right was slightly underloaded and an air bubble has destroyed part of the chip. The panel to the right would be expected to have a slightly higher pore occupancy if it were from the same run as the panel to the left.

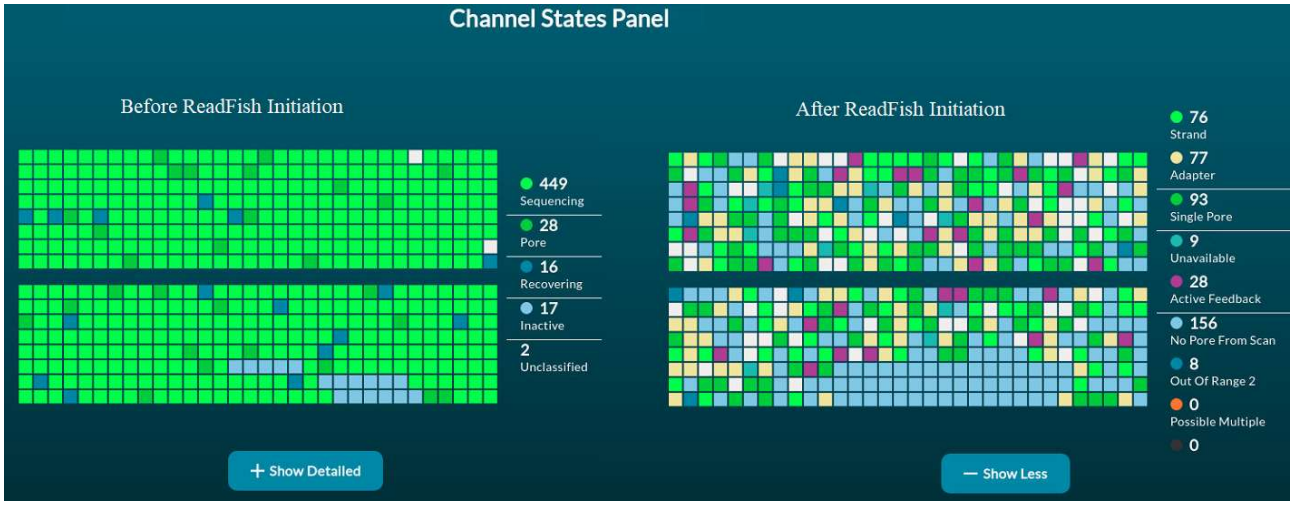


Figure 13: Channel States with and without ReadFish enabled

These two channel states are from different runs, but they serve the purpose of showing the pattern of how the sequencing changes upon ReadFish initiation. The channel states before ReadFish initiation show much higher pore occupancy since 449 pores is sequencing. The channel states after ReadFish initiation shows a lower pore occupancy and a lot of the sequencing pores are labeled as adapters.

The plot of the duty time in Figure 14 is another way of illustrating what is shown with the channel state panels in Figure 13. When sequencing without ReadFish enabled the number of pores with a strand present is high. In this experiment, about 77 % of the 512 available pores are occupied by a strand. The number of pores occupied by adapter sequences is about 8 %. When starting ReadFish the total number of occupied pores drops, and the ratio between the different pore activities changes. There is a greater percentage of pores occupied by adapter sequences compared to the percentage of pores occupied by strands when ReadFish is enabled, even though there still are fewer pores occupied. When ReadFish is started about 38 % of the pores are occupied by strands and about 18 % of the pores are occupied by adapter sequences. This means that there is a loss of pore occupancy of about 29 %.

Another difference between sequencing without and with ReadFish is that the active feedback increases from about 2 % to 10 %. There is also an increase in unclassified pore states, which increases from about 2 % to 15 %. The number of pores not sequencing (single pores) increases from about 10 % to 20 %.

All this is due to the rejection of strands through active feedback, which is communicated between ReadFish, the ReadUntil API, and MinKNOW.

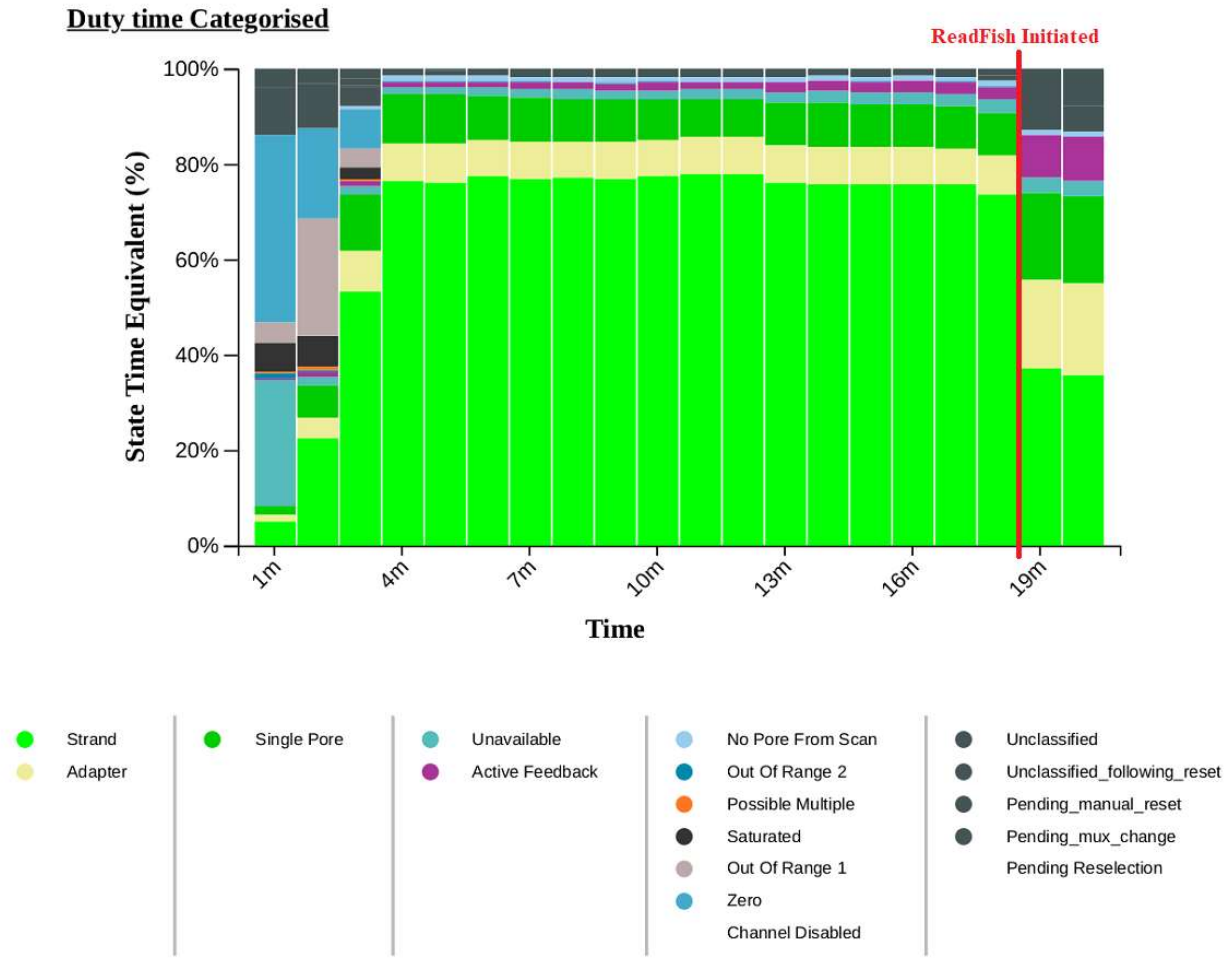


Figure 14: Duty Time of run Experiment 4

This plot describes the distribution of pore activities over time. The red vertical line represents the timepoint where ReadFish was started. There is a clear shift in pore states when ReadFish is running versus when the sequencing is running with default settings. Pore activity that decreases after starting ReadFish is the pore state called strand, meaning the number of pores occupied by a DNA-strand. The activities increasing after starting ReadFish are adapter, single pore, active feedback, and unclassified. The three first pillars of the duty time plot are looking odd due to the mux scan of the pores.

The average length of the DNA sequenced is also clearly affected by the initiation of ReadFish, which makes the read length histogram from the MinKNOW software a useful plot to visualize the effect of ReadFish. The read histograms from the MinKNOW software describe the read length of the rejected reads, as the N50 reflects the overall efficiency of read rejection (Payne et al. 2020a). The read length histogram for all four experiments is present in appendix 3 but their N50 values are also present in table 6.

When the sequencing ran with ReadFish deactivated, the sample shown in Figure 15 has an N50 of 5,060 bases. When the sequencing was modified with ReadFish, the sample shown in Figure 15 has an N50 of 545 bases. This is a huge difference and shows that ReadFish efficiently rejects the reads

not matching the target regions given. It would be expected that the N50 would increase with time if ReadFish was not initiated, as many longer reads (classified as outliers in the read length histogram) would be sequenced. When running ReadFish the number of seconds it takes for a read to be rejected is shown in the Linux terminal. This amount of time should be below 0.4 seconds, according to the manual.

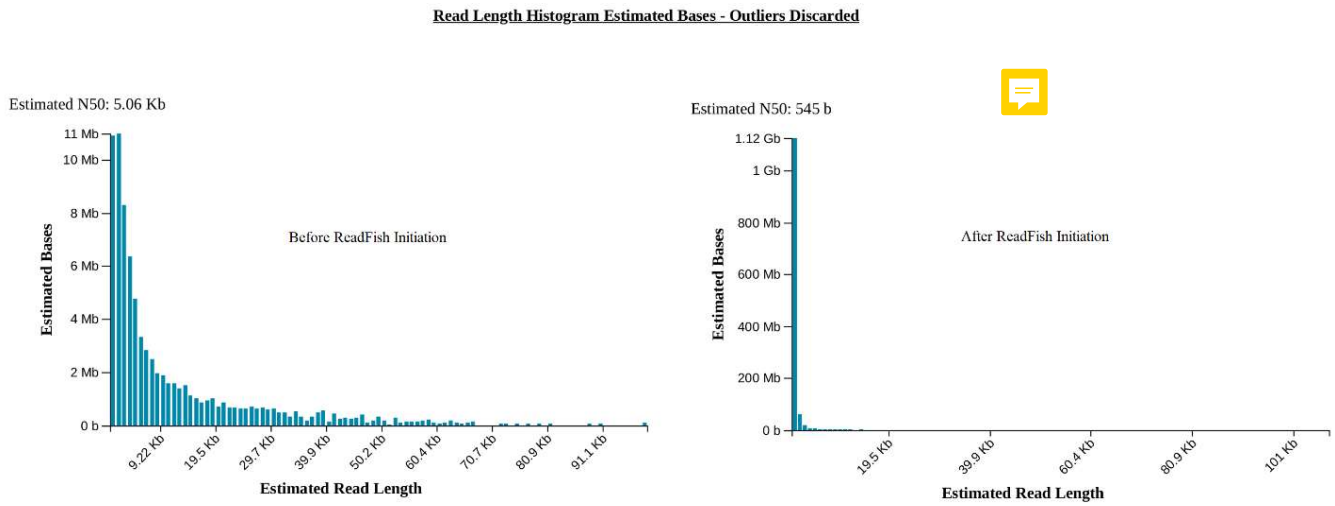


Figure 15: Read Length Histogram of Experiment 2.

The y-axes show the estimated bases in byte size. The x-axes show the estimated read length.

On the left, there is a read length histogram showing how the read length was distributed before the initiation of ReadFish. Before the initiation of ReadFish, the N50 was on 5.060 bases. On the right, there is a read length histogram showing how the read length was distributed after the initiation of ReadFish. After the initiation of ReadFish, the N50 was on 545 bases.

3.1.2 The Effect of Fast vs High Accuracy Basecalling

Of the four experiments, experiment 1 was the only run with the fast version of the basecalling. The only difference between experiments 1 and 2 is the version of basecalling, and whether the flow cell was washed. During this thesis, washing has proved to give a longer runtime with more available pores, making it an advantage for experiment 1.

When looking at the average target enrichment and the average fold-change in table 5, there is a slight increase for experiment 2 compared to experiment 1. The average target enrichment for experiment 1 is 5.35x and 5.50x for experiment 2. The average fold-change for experiment 1 is 3.24 and 3.37 for experiment 2. The fold change is the target depth normalized to the average genome depth, making it

a great number for comparison between experiments. The average target enrichment is the average target depth subtracted from the average genome depth.

Table 5: Enrichment Results from the Targeted Sequencing Experiments

This table contains statistics on the average genome depth, the average target depth, the average target enrichment, and the average fold-change in the target regions. The full enrichment tables for all the experiments can be seen in Appendix 3.

	Avg. Genome Depth	Avg. Target depth	Avg. Target enrichment	Avg. Fold-Change in Target Regions
Experiment 1	2.29x	5.35x	3.24x	2.34x
Experiment 2	2.13x	5.50x	3.37x	2.58x
Experiment 3	4.97x	19.50x	14.53x	3.92x
Experiment 4	5.39x	20.27x	14.88x	3.76x

Table 6: MinKNOW Report Data from the Targeted Sequencing Experiments

In this table, descriptive information about the four experiments is presented. The input DNA is given in ng, the run length is given in hours (h) and minutes (m), the N50 is given in bp, and the estimated bases sequenced in the run are given in Gb.

** This run crashed after a few hours due to storage issues, which results in two reports from MinKNOW. The N50 estimated 3 hours after sequencing on day 1 was on 545 bp. The N50 estimated at day 3 after running for ~48 hours were on 603 bp. See appendix 4 for histograms of the read length for these two runs made on the same sample. Since the second sequencing attempt ran for ~48 hours, the true N50 is expected to be closer to 603 than 545 bp.*

	Input DNA	Run Length	N50	Estimated Bases
Experiment 1	1242.6 ng	70h 40m	531 bp	7.38 Gb
Experiment 2	1075 ng	50h 45m	603 bp*	6.82 Gb
Experiment 3	1752 ng	68h 11m	656 bp	15.86 Gb
Experiment 4	418 ng	72h 48m	595 bp	17.46 Gb

The average genome depth is lower for experiment 2 compared to experiment 1, as well as the input DNA and run length (see Table 5 and 6). One would expect a sequencing run with higher DNA input, longer run time, and greater average genome depth (as in experiment 1 compared to experiment 2) to give a greater target enrichment, but in this case, it does not. Therefore, it indicates that HAC base-calling is superior to fast basecalling when using ReadFish.

There are a lot of factors that determine how long a sample can be sequenced on a flow cell, including the number of available pores at the start of sequencing, the occupancy of the pores (as the sequencing

pores dies when they do not have enough work to do), as well as whether the flow cell is being washed during sequencing followed by reloading with a fresh DNA library. The runs were not terminated before the flow cell had less than approximately 10 sequencing pores left. The input of experiment 2 is not the limiting factor as experiment 4 has greater results with lower input.

The cost of using HAC basecalling compared to fast is a slight increase in N50, suggesting that it takes longer for the basecalling signal to be produced which entails that the DNA-strand is sequenced for longer. This increase in N50 is small compared to the N50 of sequencing without ReadFish, which has shown to be about 5,000 bp or more. One would also expect the increased decision time for the reads to give a decreased enrichment fold-change, but this is not the case.

One can ponder whether more precisely basecalled sequences will make the alignment more precise, leading to more sequences being aligned to the target regions.

3.1.3 The Effect of Target Region Size and Needle Shearing

Experiment 1 and 2 was run with the small target region, while experiment 3 and 4 were run with the large target region. Experiment 3 and 4 has also undergone needle shearing in the DNA preparation, which makes it hard to assign the effect of these two parameters to either one of them.

A larger target region would in theory only add ultra-long reads to the target region while needle shearing changes the length of the DNA strands in the sample by breaking the long strands. The breakage of long strands has been reported to give a greater number of DNA-strands to sequence. Though the unblocking of DNA could be detrimental to the pores, but since the runtime for experiment 1 is similar to that of experiments 3 and 4, it is unlikely to have a significant effect. It has shown to be more detrimental to the pores to be underloaded with DNA as a fifth run with very low pore occupancy ruined the flow cell in 3-4 hours. A final argument is that ReadFish could be designed in a way where it runs better with larger target regions, which would support that the large target region has a greater effect than needle shearing. The effect we see in experiments 3 and 4 might as well be a combination of effects from both these parameters.

The needle shearing, due to the increased amount of DNA strands available is however likely to be of greater importance.

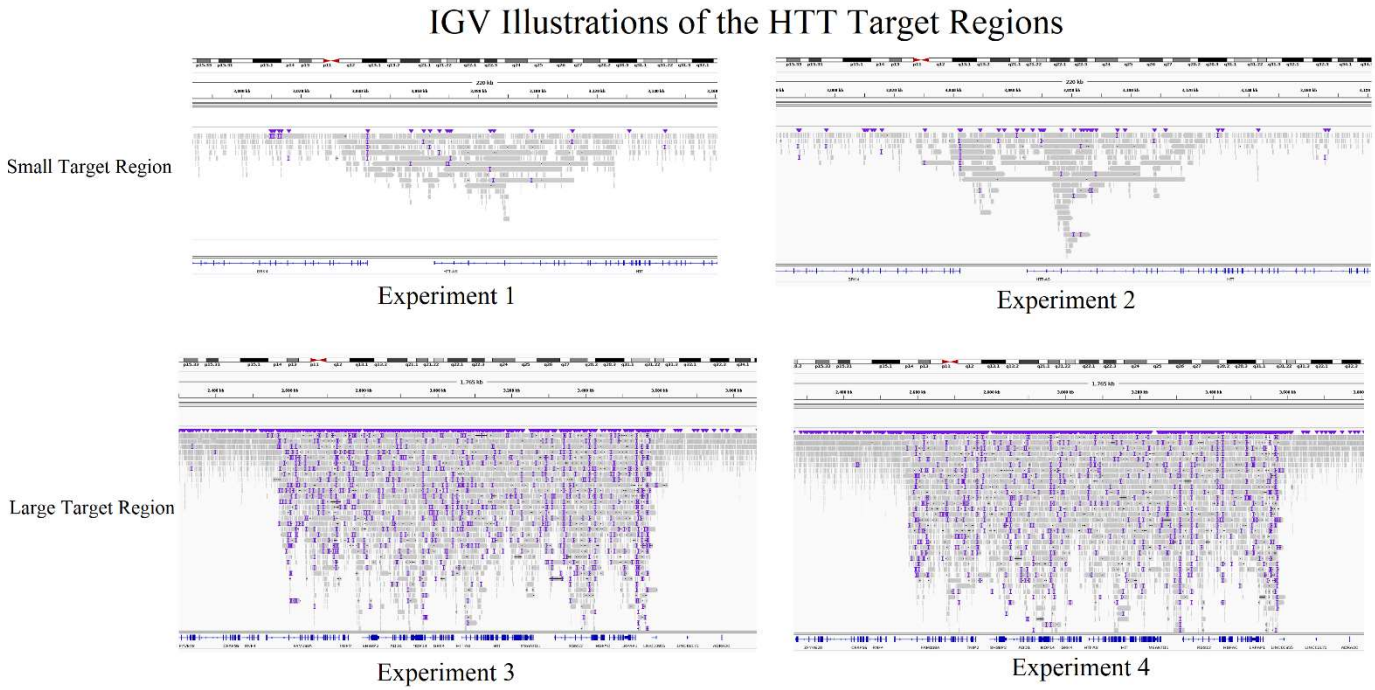


Figure 16: IGV Illustrations of the HTT Target Regions.

These images are taken from IGV in the target region around HTT. In the middle where the long reads are present, we have the target region. Around the target region there is a lot of small reads. Experiment 1 had a ~2.7-fold enrichment in the HTT target region. Experiment 2 had a ~3.1-fold enrichment in the HTT target region. Experiment 3 had a ~4.17-fold enrichment in the HTT target region. Experiment 4 had a ~4.03-fold enrichment in the HTT target region.

Experiments 3 and 4 have higher outputs of gigabases compared to experiment 1 and 2, which are listed in table 6. This increase indicates independence of the amount of input DNA and run-length, as experiment 3 ran for less than experiment 1, and experiment 4 had lower input DNA than experiment 1 and 2. The input DNA can though be deceiving as the fmol of input DNA is dependent on DNA length. The fmol indicates how many DNA strands are available for sequencing, which is more useful to know than the input DNA, as the number of available DNA-strands tethered to the membrane would increase and thereby most likely increase the overall amount of bp sequenced.

A Tapestation was used to measure the DNA length of the samples before and after needle shearing. All the samples (experiment 1-4) had a dominating peak at >60,000 bp. After needle shearing on the samples used for experiments 3 and 4 it showed the same dominating peak at >60,000 bp. There was an indication of the concentration of the samples being out of range making it difficult to rely on these results. The shorter the DNA-strands the greater the fmol loaded into the flow cell will be.

The average fold-change for experiments 3 and 4 was between 3.76x and 3.92x, while it only was between 2.34x and 2.58x for experiments 1 and 2. This can also be seen in Figure 16, where the runs have been visualized with IGV centered around HTT. The target region is obvious as there are long reads inside it. There are short reads around the target regions. For all experiments, there is a tendency for short reads to appear inside the target region even though they are not expected.

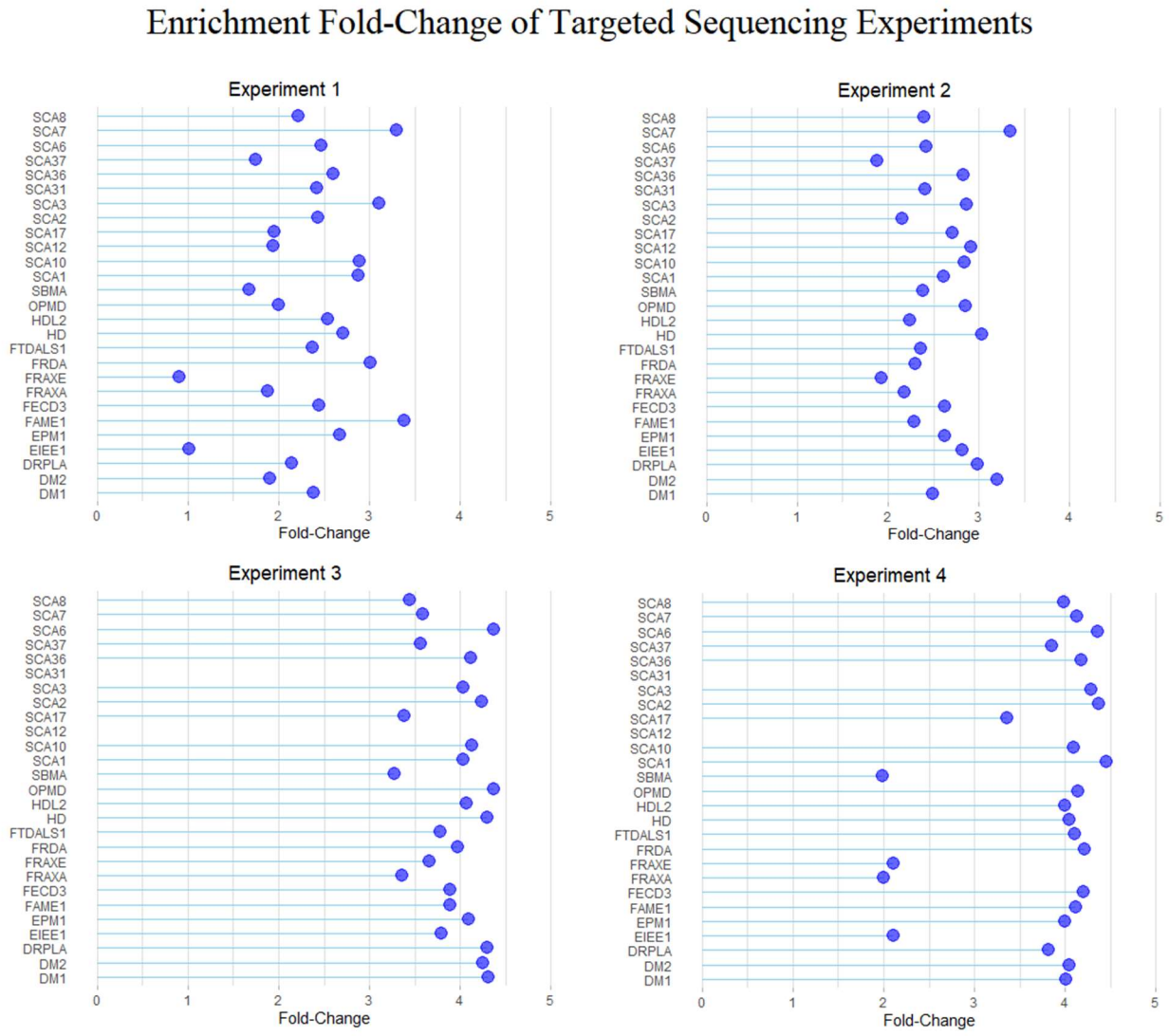


Figure 17: The Average Fold-Change in the Target Regions.

The blue dots show the value in fold-change for the individual targets. Experiments 1 and 2 had a small target region, where each target had a size on ~100.000 bp. Experiments 3 and 4 had a large target region, where each target has a size on ~1.000.000 bp. There has been a typo for the target region of SCA12 and SCA31, leading to a much lover enrichment compared to the other targets. SCA12 and SCA31 have been excluded from the plots for experiments 3 and 4.

The effect of the large target region and the needle shearing is also clearly illustrated in Figure 17, where the target panel is listed on the y-axis and the fold-change normalized to the genome depth is given at the x-axis. Experiment 3 and 4 has an increased enrichment compared to experiment 1 and 2.

3.1.4 The Effect of Ligation Sequencing kit SQK-LSK109 and SQK-LSK110

The preparation of experiments 1-3 was done with ligation sequencing kit SQK-LSK109, while experiment 4 was prepared with the newer kit, SQK-LSK110, to test if the newer version of the reagents would give a greater enrichment. In Figure 17 the enrichment is shown to be between ~3- and 4.5-fold for experiment 3, while the fold-change is between ~2 and 4.5 for experiment 4. The enrichment for experiment 4 seems more scattered compared to experiment 3, making the average enrichment slightly worse even though the output of the flow cell was greater with 17.46 Gb for experiment 4 and 15.86 Gb for experiment 3. When looking at the average target depth, experiment 4 has a depth on 20.27x while experiment 3 has a depth on 19.50x. It is difficult to conclude that the newer sequencing kit is better or worse than the old as the average enrichment is better for experiment 4 despite the average fold-change being slightly lower than experiment 3.

3.1.5 The Effect of Washing and Reloading the Flow Cell

Only the flow cell in experiment 2 was not washed and reloaded with a freshly prepared DNA library. Washing unblocks pores and therefore makes the pore availability higher, giving a higher sequencing yield. Experiment 2 was not washed since there had been some practical issues with loading the flow cell and it was prioritized to get enough DNA into the flow cell instead of loading it twice. This means that the estimated bases generated by that run could have been greater.

Sub-Conclusion for Targeted Sequencing

Targeted sequencing, with ReadFish, met our expectations, as 2.7-5.4x enrichment was expected. The enrichment achieved was a fold-change between 2.34 and 3.92. The targeted sequencing indicated maximum enrichment with the following parameters: HAC basecalling with Guppy, Needle shearing,

a larger target region on 1,000,000 bp, washing and reloading the flow cell (preferably more than once).

3.2 Repeat Count Methods

Nanopore sequencing is more error-prone than Illumina sequencing and is a rather new technology and therefore lacks a golden standard tool for detection of repeat expansions. Nanopore has the advantage of sequencing long reads, making it ideal for detecting long repeat expansions like the ones in people with FXS. There are multiple programs released by research groups worldwide, whereof some are still under maintenance. The programs used in this project are RepeatHMM (Liu et al. 2017), STRique (Giesselmann et al. 2019), Tandem-Genotypes (Mitsuhashi et al. 2019), and TRiCoLOR (Bolognini et al. 2020). TRiCoLOR takes haplotype phased bam-files as input and is not analyzed in this section.

In this section, the three programs mentioned above are presented with a focus on one gene at a time. The WG sequencing data from the former student are presented as well as the targeted data produced in this project. These two sequencing methods are also combined to have the largest amount of data available to test the performance of the programs.

The expected repeat counts from the company where the cell-lines have been bought, Coriell, is given in table 7. Coriell have only informed the HTT repeat count for sample GM04284 and GM05538 and only FMR1 for the rest of the samples. A former student sequenced the samples on Illumina and used ExpansionHunter to determine the repeat length of HTT and FMR1 for all the samples. Illumina is precise with short repeats like HTT but is challenged with larger repeats such as FMR1. The former student has used the AmplideX® PCR/CE FMR1 kit from AsuraGen to determine the repeat lengths for all the samples. All these expected values are given in table 7.

Table 7: Expected Repeat Count Values

This table shows the expected repeat length values for the samples at HTT and FMR1. The company where the cell-lines are bought, Coriell, has informed some of the repeat lengths needed for this analysis. Supplementary is a former student work with AmplideX® PCR/CE FMR1 kit and analysis of Illumina data with ExpansionHunter on HTT. The males are homozygote for FMR1 while the females are heterozygote. A hyphen indicates a lack of information.

** From whole-genome data*

*** Values from Illumina data analyzed with ExpansionHunter for whole-genome data*

**** The PCR method cannot determine repeats longer than 200*

***** The reads did not cover and span the entire repeat*

Sample	Sex	Gene	Coriell Informed Values	PCR Values*	Illumina Values**
GM04284	M	HTT	39/50	-	39/47
		FMR1	-	20	20
GM05538	M	HTT	22/101	-	22/112
		FMR1	-	20	20
GM06891	M	HTT	-	-	9/20
		FMR1	118	118	72****
GM07541	F	HTT	-	-	17/22
		FMR1	29/31	29/31	29/31
GM07861	M	HTT	-	-	19/20
		FMR1	351-400	>200***	61****
GM20239	F	HTT	-	-	18/18
		FMR1	20/183-193	20/196-199	20/91****

3.2.1 Program Comparison for HTT Repeat Count Precision

The three programs used in this section and for the plots in Figure 18 are RepeatHMM, STRique, and Tandem-Genotypes. Fully sized plots are available in Appendix 8.

RepeatHMM is the poorest one to estimate correct repeat lengths for HTT as it always has what is either a lot of outliers or an allele that is estimated to have a much higher repeat length compared to what the other programs are indicating.

STRique estimates repeat lengths close to the expected values. However, sometimes there is a large variation on the estimated repeat lengths.

Tandem-Genotypes estimates repeat lengths close to the expected values as well and have less variation compared to STRique.

Unphased Repeat Length Estimation for HTT

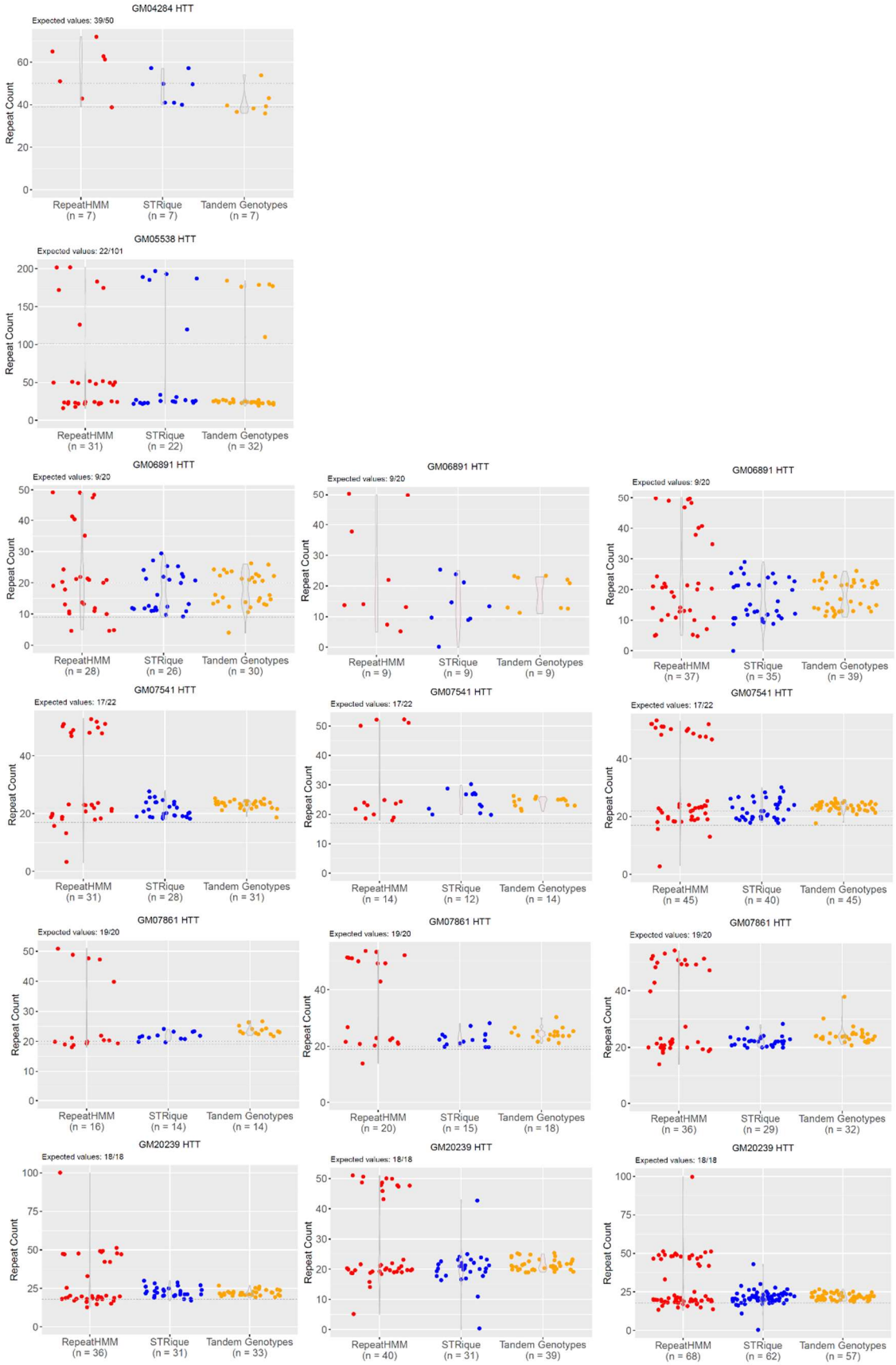


Figure 18: Unphased Repeat Length Estimation for HTT

Whole-genome data have been sequenced on PromethION, targeted data have been sequenced on Minion and the combined data are the whole-genome and targeted data merged. Column one is whole-genome data, column two is targeted data, and column 3 is combined data. Each row has plots belonging to one sample. The x-axis in the plots is divided into 3 sectors, where sector one is for RepeatHMM data (red dots), the second is for STRique data (blue dots), and the third one is for Tandem-Genotypes data (orange dots). The Y-axis is the repeat counts and all the plots are forced to start at 0 but they vary the maximum repeat size. Every plot has the expected values written as subheader, as well as the expected values indicated inside the plot by a dashed horizontal line.

All three of the repeat estimating programs tend to estimate the repeats to be longer than what is expected. It is easy to distinguish between the alleles when they are far from equal in repeat length. The contrary is the case when the alleles are close to equal. It is clear that individuals having alleles with 5 repeats or less in difference are hard to distinguish, this is likely due to the high nucleotide indel error rate seen in Nanopore data.

In the plot for sample GM06891 with combined data, the two alleles are visibly separated for STRique and Tandem-Genotypes. RepeatHMM has data points covering the same range as allele 1 and 2 is covering for STRique and Tandem-Genotypes, but additionally has 10 data points in the range 35-50. This makes it seem like there is an allele with repeat size around 15 and 45, where it is at 9 and 20.

In the plot for GM20239 with combined data, the two alleles have the same repeat length at 18 in the HTT gene. STRique and Tandem-Genotypes have data points that are inseparable at repeat count 16-29 when you see away from the outliers. RepeatHMM does also have repeat length in the area just described but additionally have repeat lengths plotted around 42-51. If RepeatHMM was used on its own, one would be tricked into believing that there would be a normal allele and an allele in the affected area (equal to or above 40).

3.2.2 Program Comparison for FMR1 Repeat Count Precision

Based on the variation in repeat lengths and the precision, Tandem-Genotypes is the most precise program at estimating repeat lengths for FMR1, with STRique as second, and lastly RepeatHMM.

For HTT, RepeatHMM tends to overestimate the repeat lengths which is not the case for FMR1, as illustrated in Figure 19. Fully sized plots are available in Appendix 9.

Unphased Repeat Length Estimation for FMR1

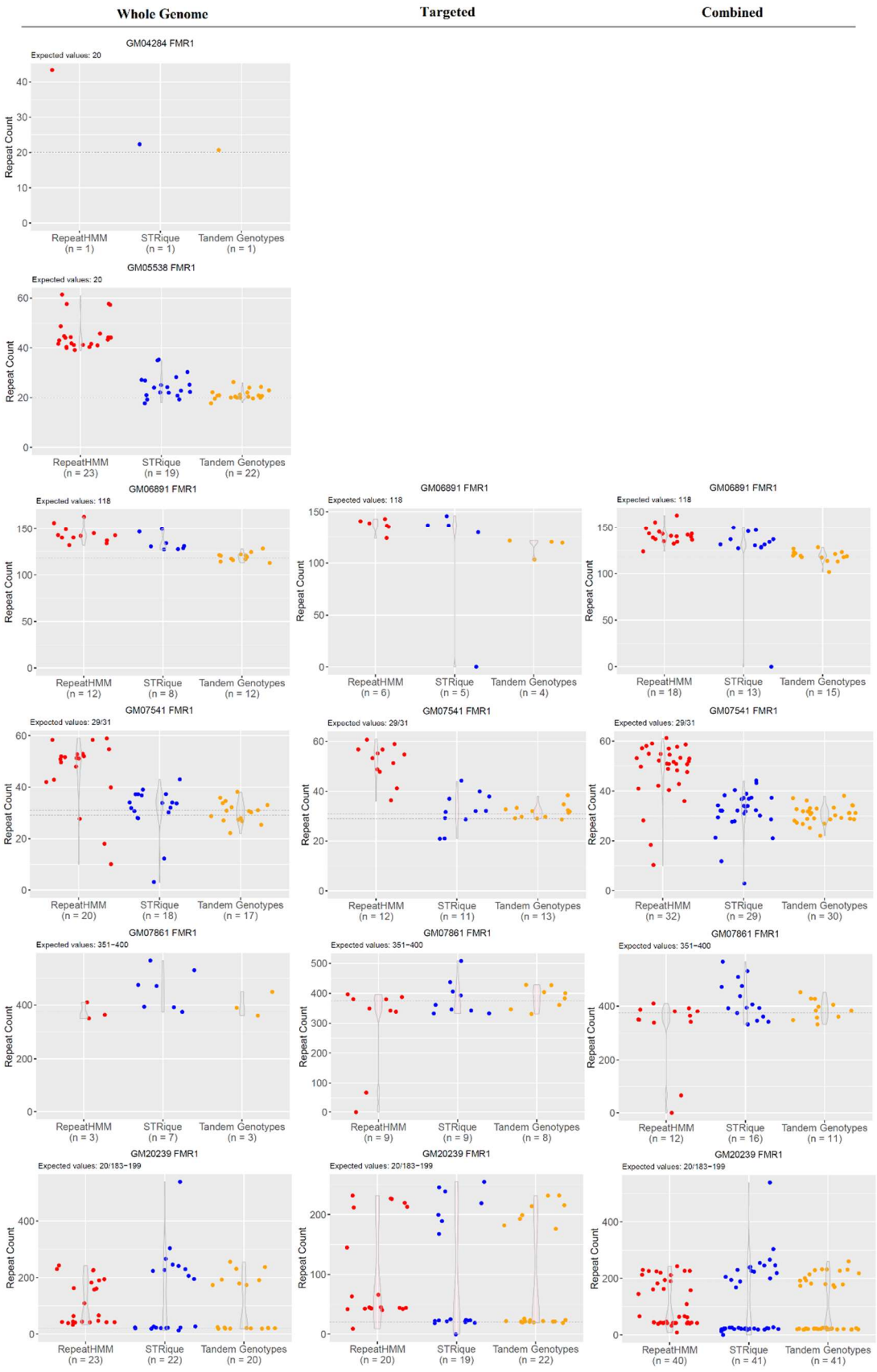


Figure 19: Repeat Count Comparison for FMR1

Whole-genome data have been sequenced on PromethION, targeted data have been sequenced on Minion and the combined data are the whole-genome and targeted data merged. Column one is whole-genome data, column two is targeted data, and column 3 is combined data. Each row has plots belonging to one sample. The x-axis in the plots is divided into 3 sectors, where sector one is for RepeatHMM data (red dots), the second is for STRique data (blue dots), and the third one is for Tandem-Genotypes data (orange dots). The Y-axis is the repeat counts and all the plots are forced to start at 0 but they vary the maximum repeat size. Every plot has the expected values written as subheader, as well as the expected values indicated inside the plot by a dashed horizontal line.

The programs are also challenged when separating alleles in cases where the repeat count is almost equal, as seen for HTT (illustrated in the plot for GM07541).

However, when the repeat lengths are vastly different, the programs clearly separate the alleles for both genes (illustrated in the plot for GM20239).

Data suggests that the depth needed for estimating correct repeat lengths for both HTT and FMR1 is lower than expected, as the same degree of variation is present in the targeted data and the WG data. Therefore, the combined data show the same as the WG and targeted data, although with greater support. The data suggests that approximately 10 reads are enough to give repeat length estimations that are as correct as can be for the individual programs.

As an example, the targeted data from Tandem-Genotypes for GM06891 at FMR1 vary more from the expected with a read number of 4, as compared to the variation from the WG data with a read number of 12.

Sample GM07861 at FMR1 shows that for STRique and Tandem-Genotypes the variations are the same for WG and targeted sequencing. However, for RepeatHMM there is a greater variation for the targeted data compared to the WG sequenced data. This indicates that more reads do not necessarily equal a more uniform repeat estimation.

For the plots showing repeat length in HTT, the same degree of variation is present in the WG sequenced data and the targeted data. When these are combined the variation does not differ significantly and the extra number of reads compared to WG sequenced and targeted data do not result in a much clearer indication of the repeat lengths.

Sub-Conclusion for Repeat Count Methods

Correct repeat length estimation of Oxford Nanopore sequencing data is possible but is challenged in cases where the alleles are differing with less than 5 repeat units. Unphased repeat length estimation was performed by three programs: RepeatHMM, STRique, and Tandem-Genotypes. Tandem-Genotypes and STRique outperformed RepeatHMM. RepeatHMM tends to overestimate the repeat length

for a significant proportion of the DNA-reads. RepeatHMM was especially challenged when estimating repeat length in HTT, which indicates that the program has different challenges depending on the repeat motif. For a reliable performance of the programs, data suggests that around 10 reads are enough, especially if analyzing with Tandem-Genotypes. However, there are variations in the repeat length results for all samples, which are expected to become smaller with increased depth.

3.3 Haplotype Phased Count Results

Some of the samples have alleles with close repeat lengths. When plotting them they are inseparable unless one knows which allele each data point belongs to. Separating DNA reads into haplotypes is called haplotype phasing (HP) and this has been done so that every sample would have repeat lengths estimated for each allele.

Nanopore sequencing is error-prone and our theory was that basing the HP on SNPs from Nanopore data would lead to erroneous haplotypes. The samples had been sequenced on Illumina which is more accurate, and for that reason, the SNPs were chosen from Illumina sequencing. These SNPs were then used to create two haplotypes and all the reads were assigned a haplotype. This method is referred to as "Illumina" in the plots.

There are existing methods for HP Nanopore data, of which LongShot was chosen. LongShot bases its haplotype phasing solely on the Nanopore data inputted.

In the previous section, three programs' ability to estimate repeat length were tested. One more program has also been used, which is TRiCoLOR (Bolognini et al. 2020).

3.3.1 TRiCoLOR performance vs. RepeatHMM, STRique, and Tandem-Genotypes

STRique and Tandem-Genotypes were able to visually separate the two alleles for HTT in sample GM06891 (Figure 18). Though it was not clear whether one was more precise than the other. In Figure 20 the mean of the HTT alleles for the programs have been plotted as well as given in table 8 below, showing that STRique is a tad better in this case compared to Tandem-Genotypes. TRiCoLOR is though superior.

Table 8: Mean repeat length estimations of phased data

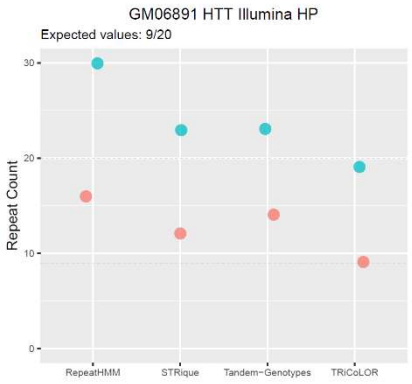
This table gives the mean of the repeat lengths estimated for the DNA reads by the four different programs phased wither by LongShot (L) or in-house Illumina-guided HP (I).

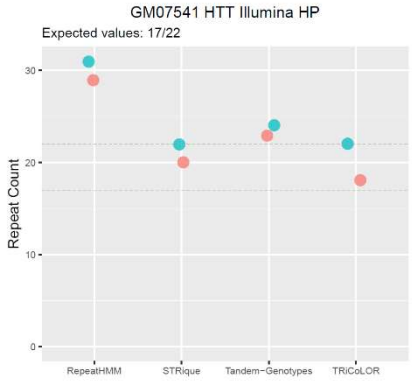
			Expected	RepeatHMM		STRique		Tandem-Genotypes		TRiCoLOR	
				L	I	L	I	L	I	L	I
Combined	*GM06891*	*HTT*	*9/20*	15/30	16/30	12/21	12/23	14/23	14/23	9/19	9/19
	GM07541	*HTT*	*17/22*	30/30	29/31	20/25	20/22	22/24	23/24	18/22	18/22
		FMR1	*29/31*	53/45	51/45	36/29	37/28	33/31	31/31	31/27	30/15
	GM07861	*HTT*	*19/20*	33/38	34/34	22/22	22/21	24/24	24/24	19/21	19/20
	GM20239	*HTT*	*18/18*	28/28	28/30	21/22	23/22	21/23	21/23	19/19	19/19
		FMR1	*20/183-199*	182/49	174/53	227/22	246/22	207/21	212/21	182/21	171/21

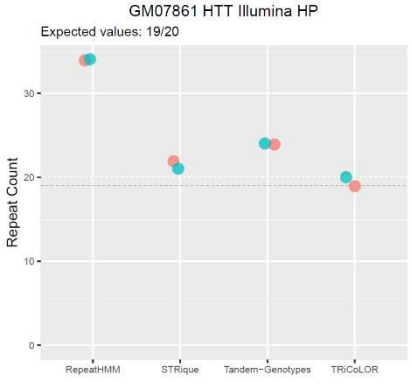
In Figure 21 the mean of the FMR1 alleles for the programs have been plotted and it shows that with LongShot HP data, TRiCoLOR is good at estimating the repeat length for the alleles for sample GM07541. The same cannot be said with Illumina HP as the repeat length estimation is bad. The repeat length estimation is good for both phasing methods for sample GM20239.

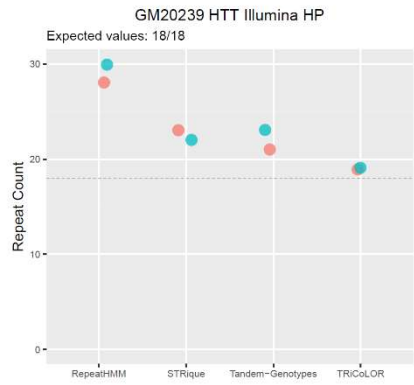
Haplotype Phased HTT Repeat Count Program Comparison

Illumina HP

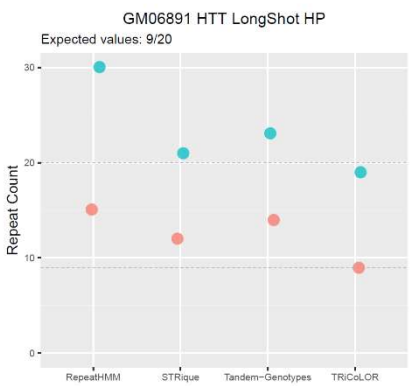


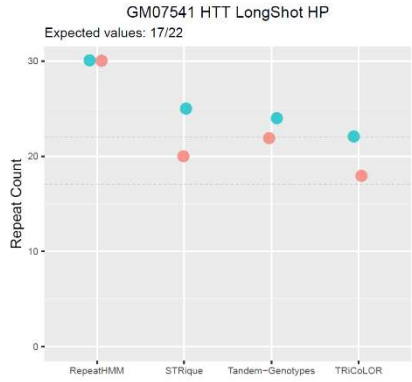






LongShot HP





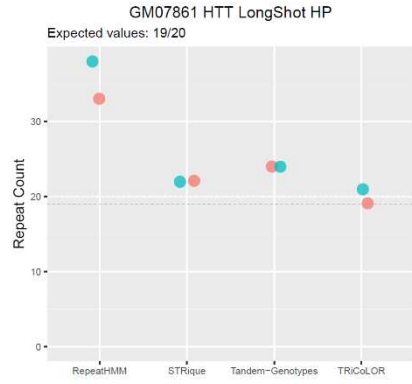


Figure 20: Haplotype Phased HTT Repeat Count Program Comparison

The four samples with targeted data have been merged with whole-genome data and these samples are shown here. In the first column, the sample data has been phased with the in-house Illumina HP method, and in the second column, the sample data has been phased with LongShot. The mean of the alleles is plotted for RepeatHMM, STRique, and Tandem-Genotypes, as well as the estimation from TRiCoLOR.

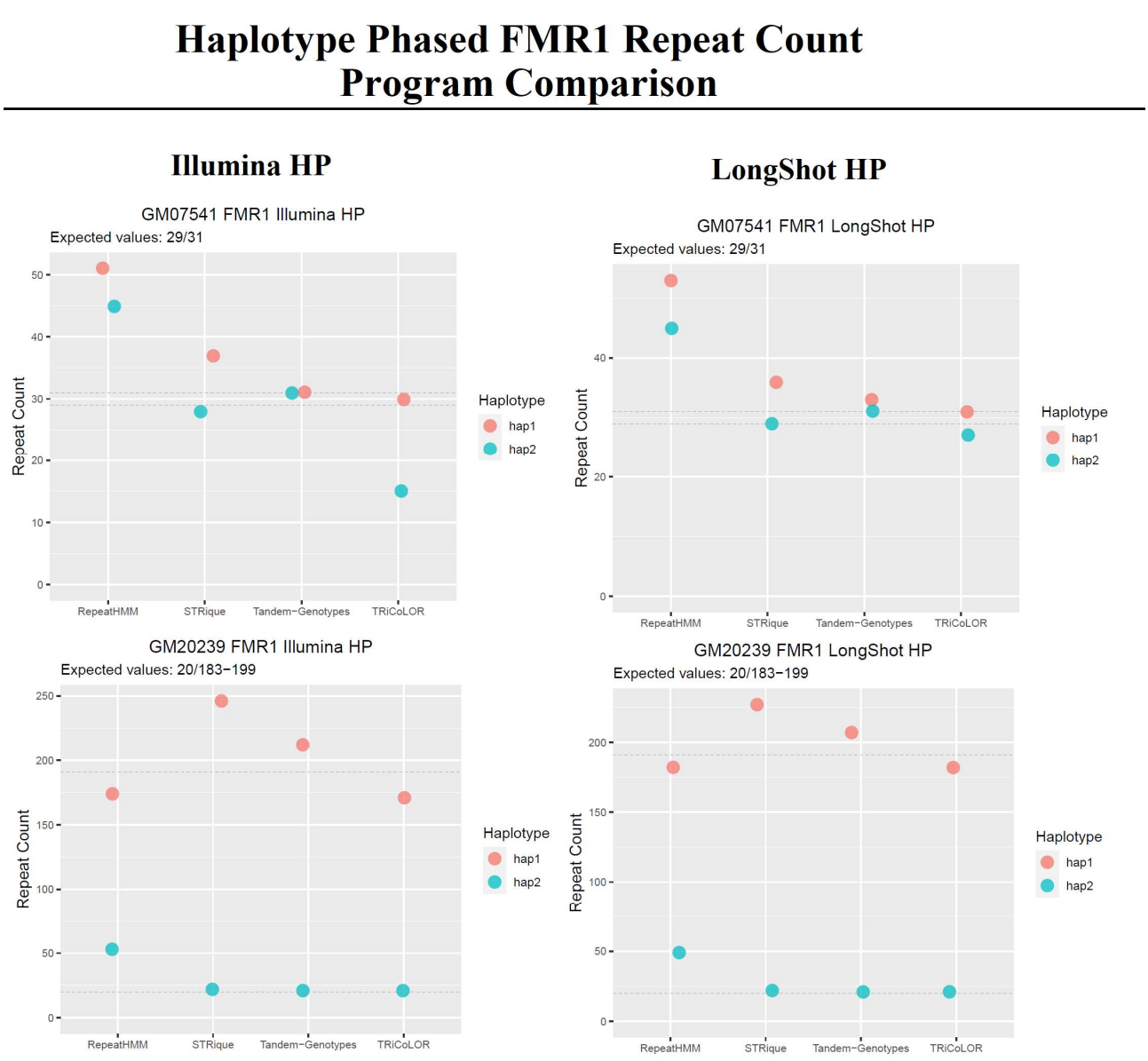


Figure 21: Haplotype Phased FMR1 Repeat Count Program Comparison

The four samples with targeted data have been merged with whole-genome data and these samples are shown here. In the first column the sample data has been phased with the in-house Illumina HP method, and in the second column the sample data has been phased with LongShot. The mean of the alleles is plotted for RepeatHMM, STRique, and Tandem-Genotypes, as well as the estimation from TRiCoLOR.

3.3.2 Haplotype Phasing Aids in Determining Repeat Lengths

Some of the plots were difficult to separate into two alleles, which is why haplotype phasing was performed. Sample GM20239 had clear allele separation for FMR1 (Figure 21 & 22) which is made even clearer by the haplotype phasing (Figure 22). Though, there are a few data points that are tagged with the wrong haplotype. Sample GM06891 also had clear allele separation for HTT (Figure 18) which also is clear in the phased plots in Figure 23.

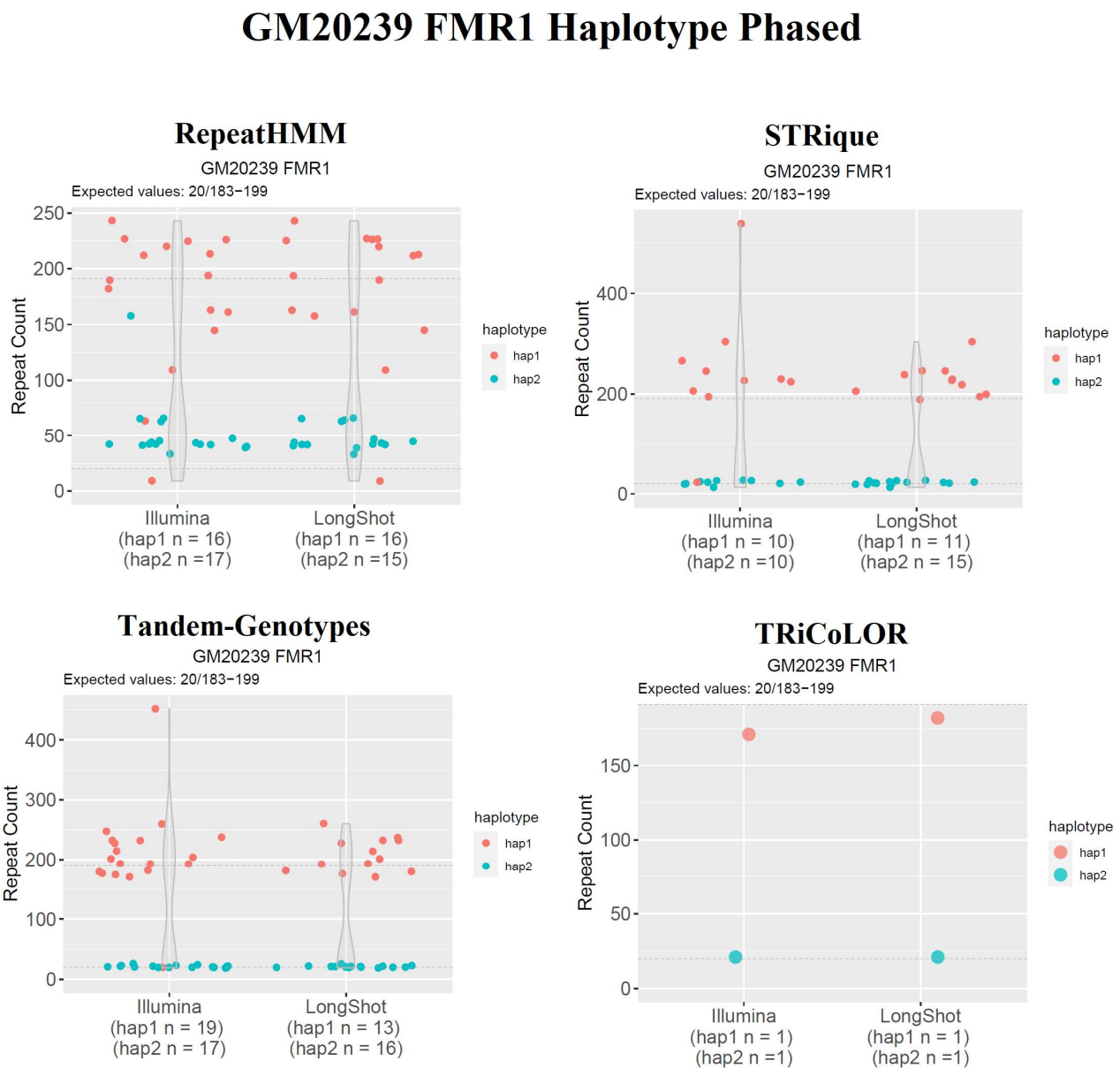


Figure 22: GM20239 FMR1 Repeat Counts Haplotype Phased

The combined data has been haplotype phased and run with the following four programs: RepeatHMM, STRique, Tandem-Genotypes, and TRiCoLOR. The first column inside each plot presents the sample data phased with the in-house Illumina HP method, and the second column presents the sample data phased with LongShot. The expected repeat length is indicated with dashed horizontal lines.

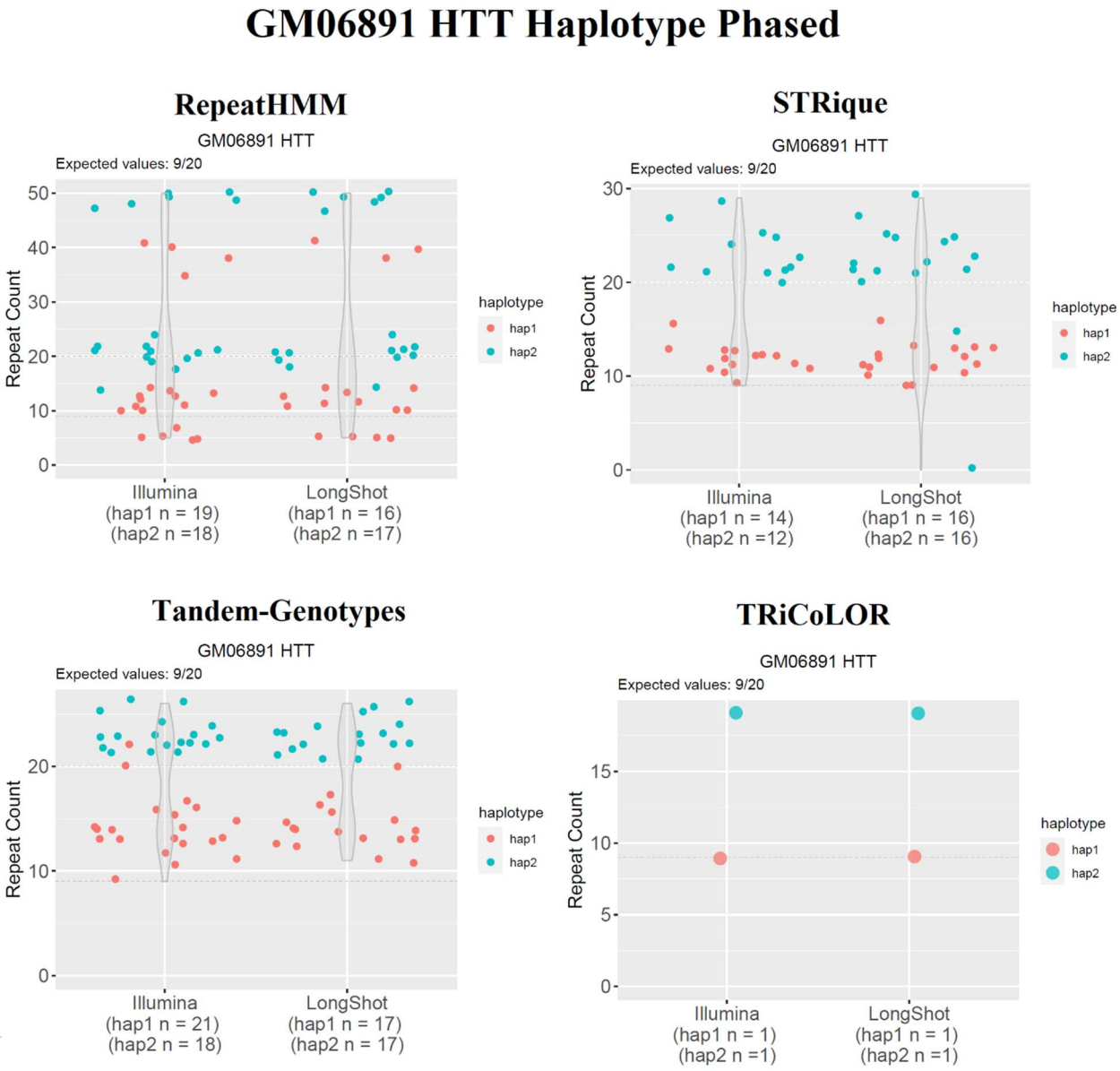


Figure 23: GM06891 HTT Repeat Counts Haplotype Phased

The combined data has been haplotype phased and run with the following four programs: RepeatHMM, STRique, Tandem-Genotypes, and TRiCoLOR. The first column inside each plot presents the sample data phased with the in-house Illumina HP method, and the second column presents the sample data phased with LongShot. The expected repeat length is indicated with dashed horizontal lines.

The alleles were impossible to separate in the unphased section for FMR1 in sample GM07541 due to the repeat lengths being close. RepeatHMM, STRique, and Tandem-Genotypes are not able to separate the alleles with HP data. See Figure 24. This is not due to the haplotype phasing as can be seen from Figure 25 where logos plots of the whole-genome data are presented. There clearly are two distinct haplotypes, even if it is hard to see since there are a lot of SNP positions compressed into Figure 25. Though we would still expect the haplotypes to be separated inside the cluster. Since they are not, the sequencing must be to blame.

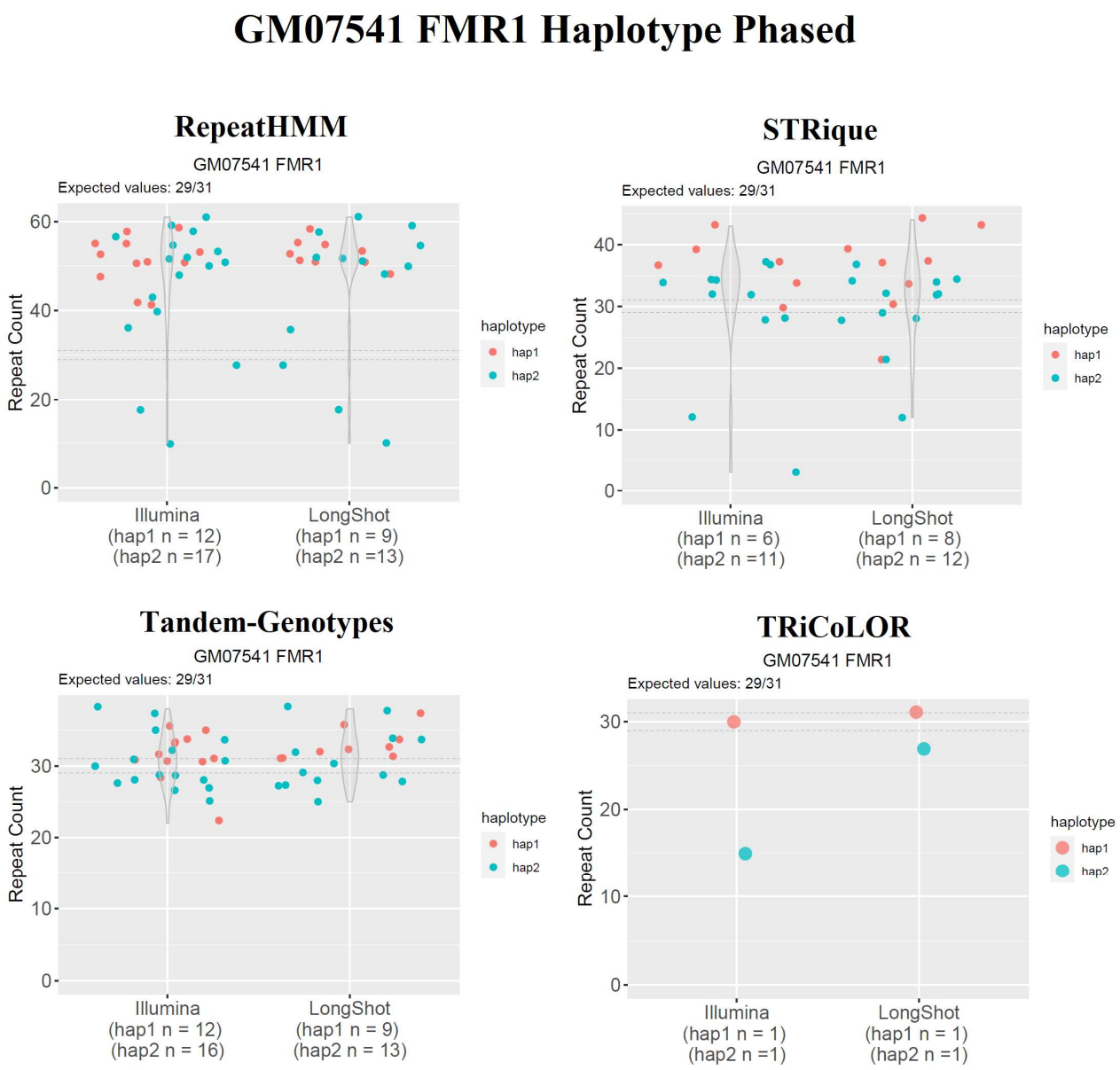


Figure 24: Haplotype Phased GM07541 FMR1 Repeat Count Program Comparison

The combined data has been haplotype phased and run with the following four programs: RepeatHMM, STRique, Tandem-Genotypes, and TRiCoLOR. The first column inside each plot presents the sample data phased with the in-house Illumina HP method, and

the second column presents the sample data phased with LongShot. The expected repeat length is indicated with dashed horizontal lines.

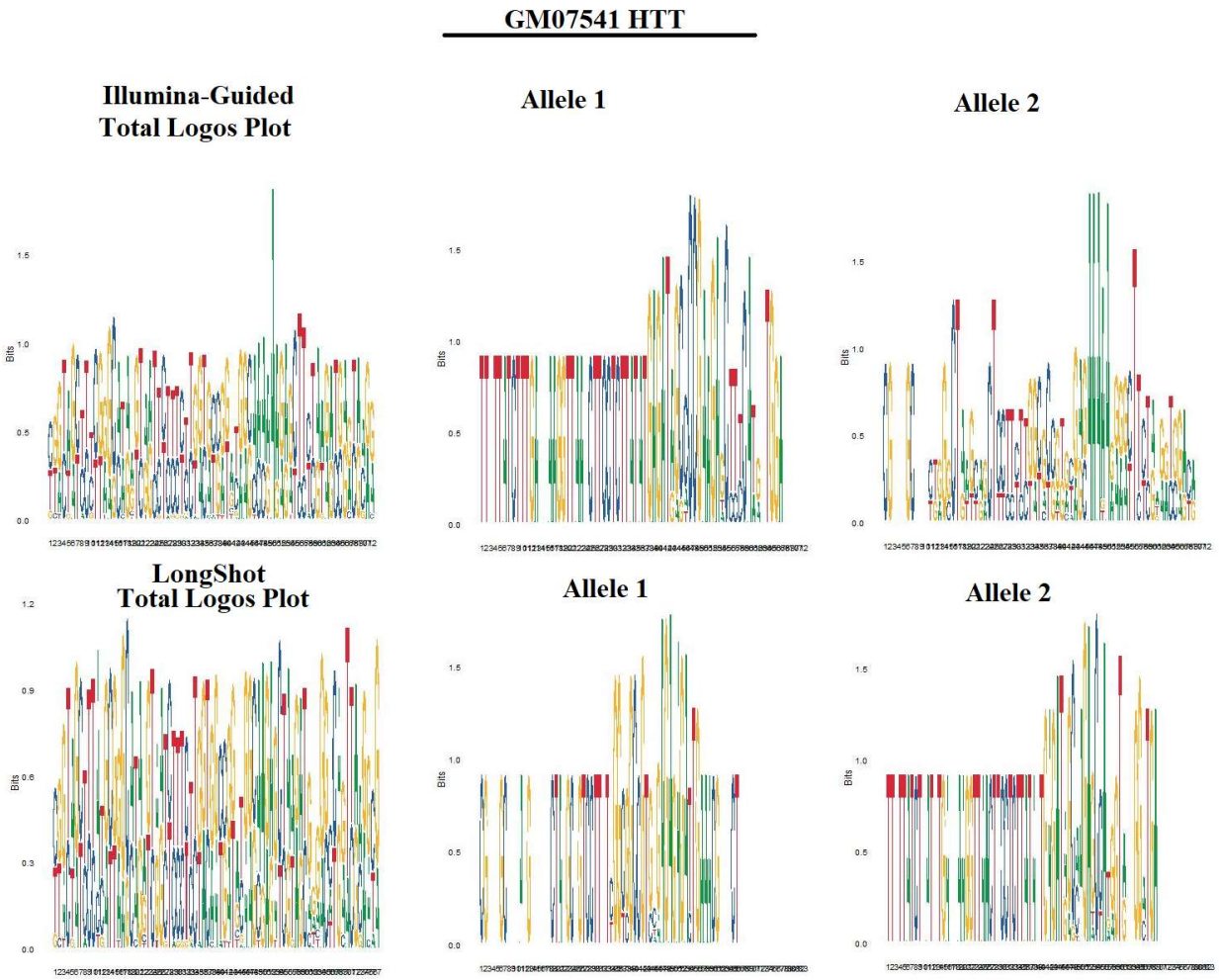


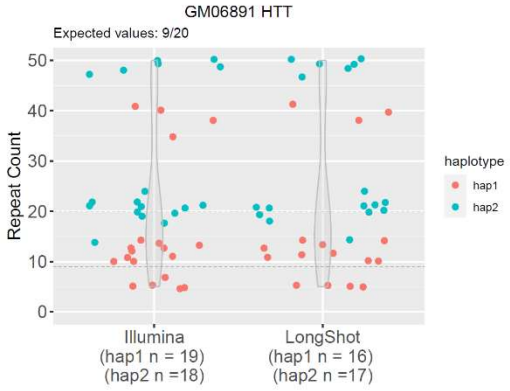
Figure 25: Logos Plot of Whole-Genome Data from GM07541 HTT

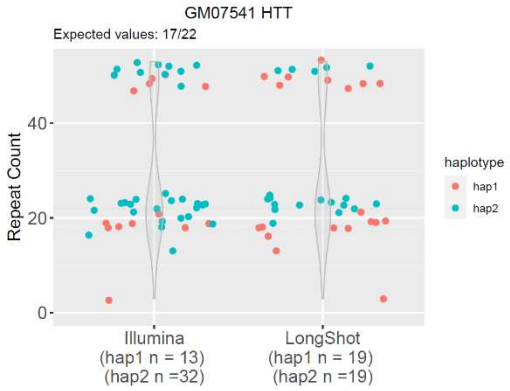
The whole-genome data have been haplotype phased with both methods, whereof the in-house Illumina-guided HP method is plotted in the first row and LongShot in the second row. Even though the SNPs for the two haplotypes are different it is clear that allele 1 for Illumina HP is matching allele 2 from LongShot. It is less obvious that allele 2 from Illumina HP is matching allele 1 from LongShot HP. Though it is clear that there are two different haplotypes (alleles) with both HP methods used. The total logos plot shows this as two nucleotides most often are stacked upon each other. At the x-axis in the plots is the SNP number and at the y-axis is the number of bits.

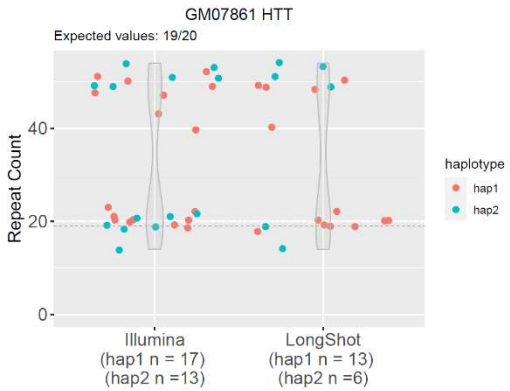
RepeatHMM performed badly with unphased data at HTT. Looking at the phased data, it seems like RepeatHMM are phased correctly but then have an extra set of alleles with a too high repeat count. Haplotype 1 and 2 are still at their respective positions, one having higher repeat counts than the other, but the repeat counts are off. If I were to use this program again, I would ignore the outliers which in the unphased section seemed like one of the alleles. For examples, see GM06891 HTT and GM07541 HTT in Figure 26.

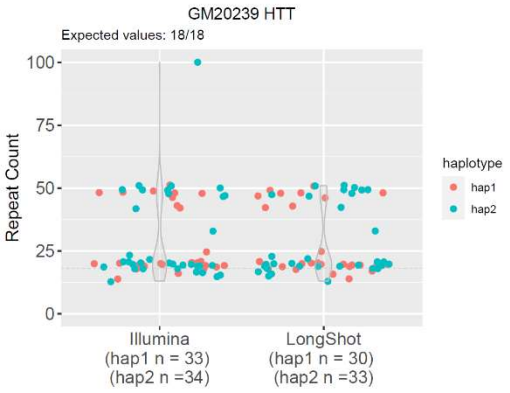
Haplotype Phased RepeatHMM Repeat Counts

HTT

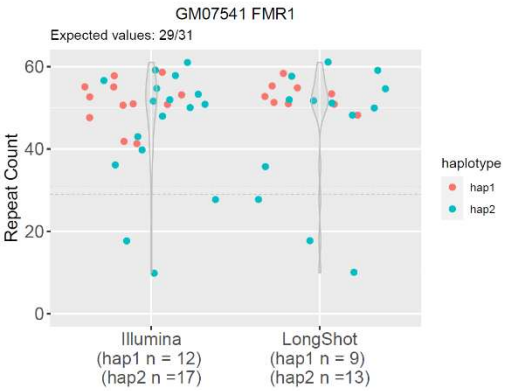








FMR1



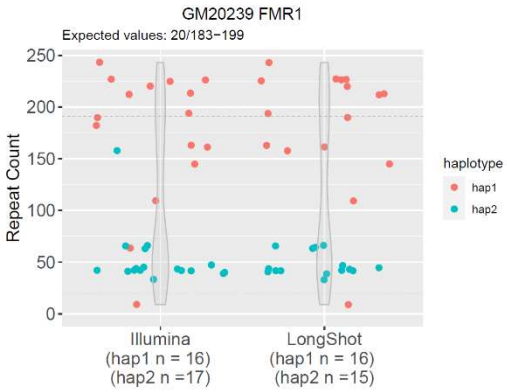


Figure 26: Haplotype Phased Combined Data Repeat Lengths

The combined data has been haplotype phased and run with the following four programs: RepeatHMM, STRique, Tandem-Genotypes, and TRiCoLOR. The first column inside each plot presents the sample data phased with the in-house Illumina HP method, and the second column presents the sample data phased with LongShot. The expected repeat length is indicated with dashed horizontal lines.

The alleles were impossible to separate in the unphased section for HTT in sample GM07541 due to the repeat lengths being close. TRiCoLOR estimated nearly correct repeat counts for this sample. For the three other programs, it is possible to detect multiple data points belonging to one haplotype on the bottom or top of the cluster, but they are not completely separated. This is probably due to sequencing errors. See Figure 27.

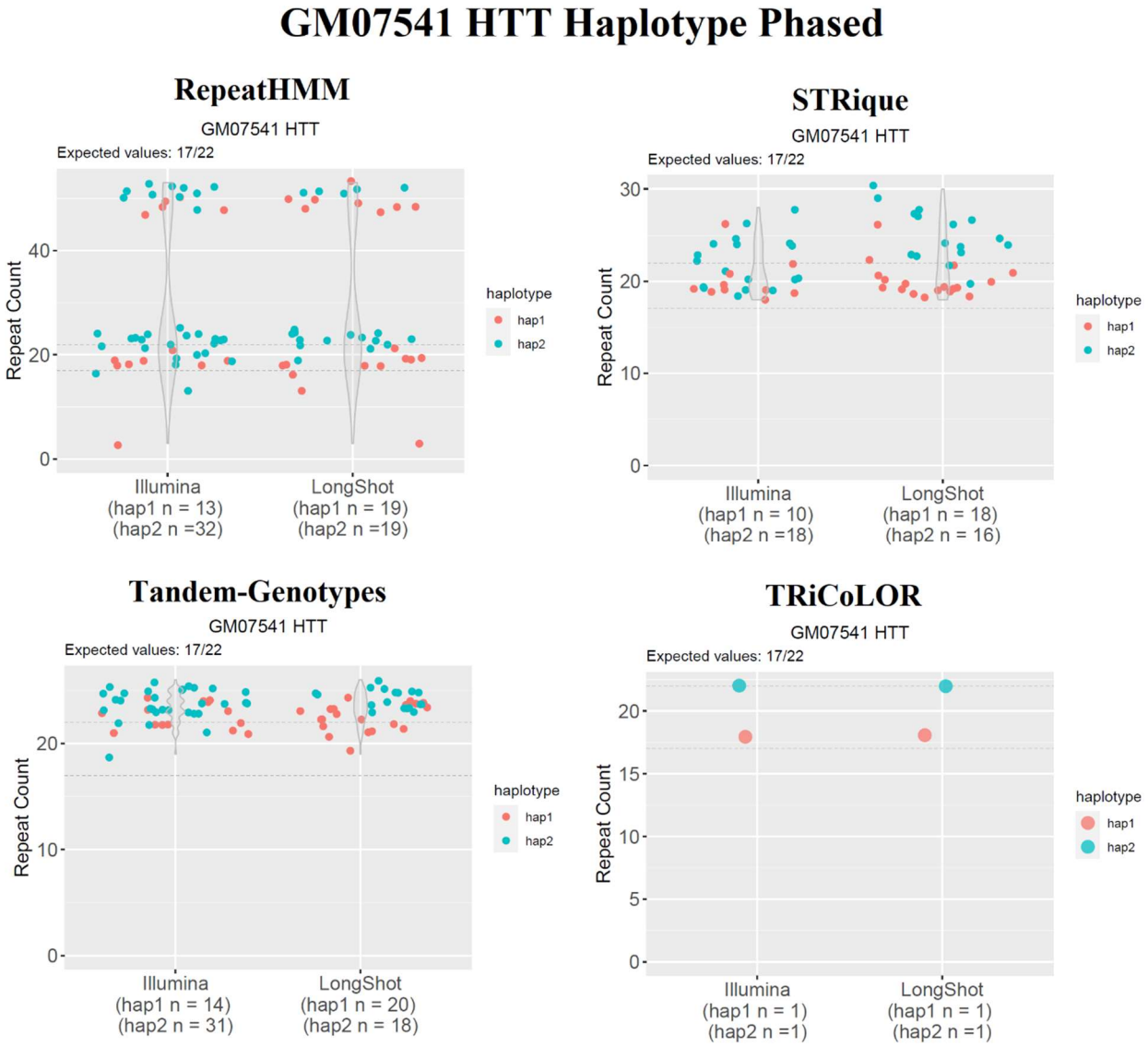


Figure 27: GM07541 HTT Repeat Counts Haplotype Phased

The combined data has been haplotype phased and run with the following four programs: RepeatHMM, STRique, Tandem-Genotypes, and TRiCoLOR. The first column inside each plot presents the sample data phased with the in-house Illumina HP method, and the second column presents the sample data phased with LongShot. The expected repeat length is indicated with dashed horizontal lines.

The alleles were impossible to separate in the unphased section for HTT in sample GM07861 due to the repeat lengths being close (Figure 28). TRiCoLOR estimated correct repeat counts for this sample. For the three other programs, the hp has not helped in separating the two alleles. This is due to them only being 1 repeat length apart in combination with sequencing errors. The haplotypes are distinctive and can be seen in Appendix 7.

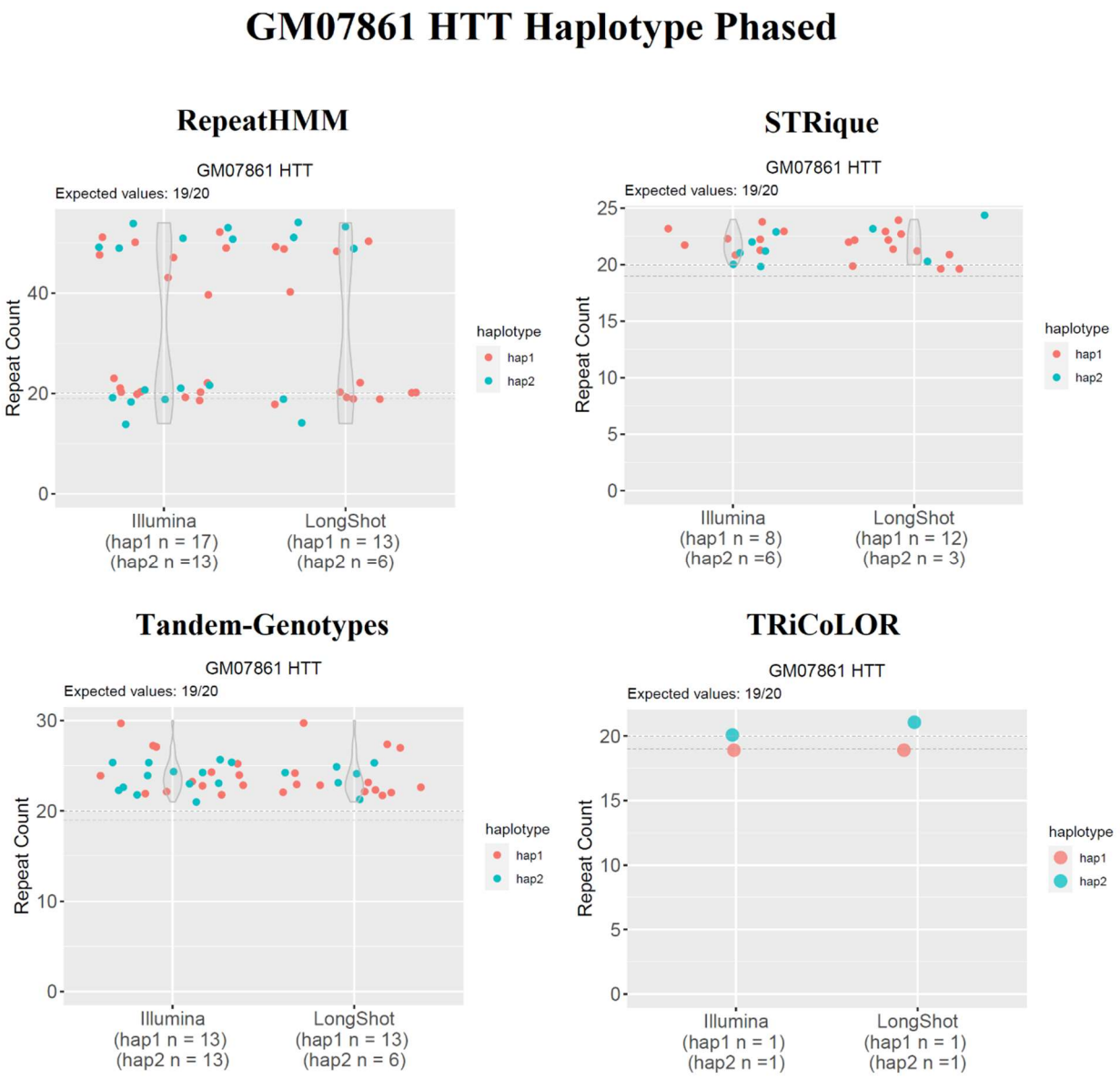


Figure 28: GM07861 HTT Repeat Counts Haplotype Phased

The combined data has been haplotype phased and run with the following four programs: RepeatHMM, STRique, Tandem-Genotypes, and TRiCoLOR. The first column inside each plot presents the sample data phased with the in-house Illumina HP method, and the second column presents the sample data phased with LongShot. The expected repeat length is indicated with dashed horizontal lines.

The alleles were impossible to separate in the unphased section for HTT in sample GM20239 as the repeat lengths are the same. TRiCoLOR estimated nearly correct repeat counts for this sample. For the three other programs, the HP has not helped RepeatHMM or STRique in separating the two alleles (see Figure 29). This is probably due to them having the same repeat length. Though, for the plot with

Tandem-Genotypes repeat counts the alleles do seem separated as more of one haplotype is dominating at either the bottom or top of the cluster. This could be explained by the comparison plots where the mean repeat count from the two haplotypes is considerably more separated in size compared to the other programs' mean repeat count for the haplotypes. It is expected that the two haplotypes would be looking like they do in the plot for STRique. The haplotypes are distinctive even though they have the same repeat length (see Appendix 7).

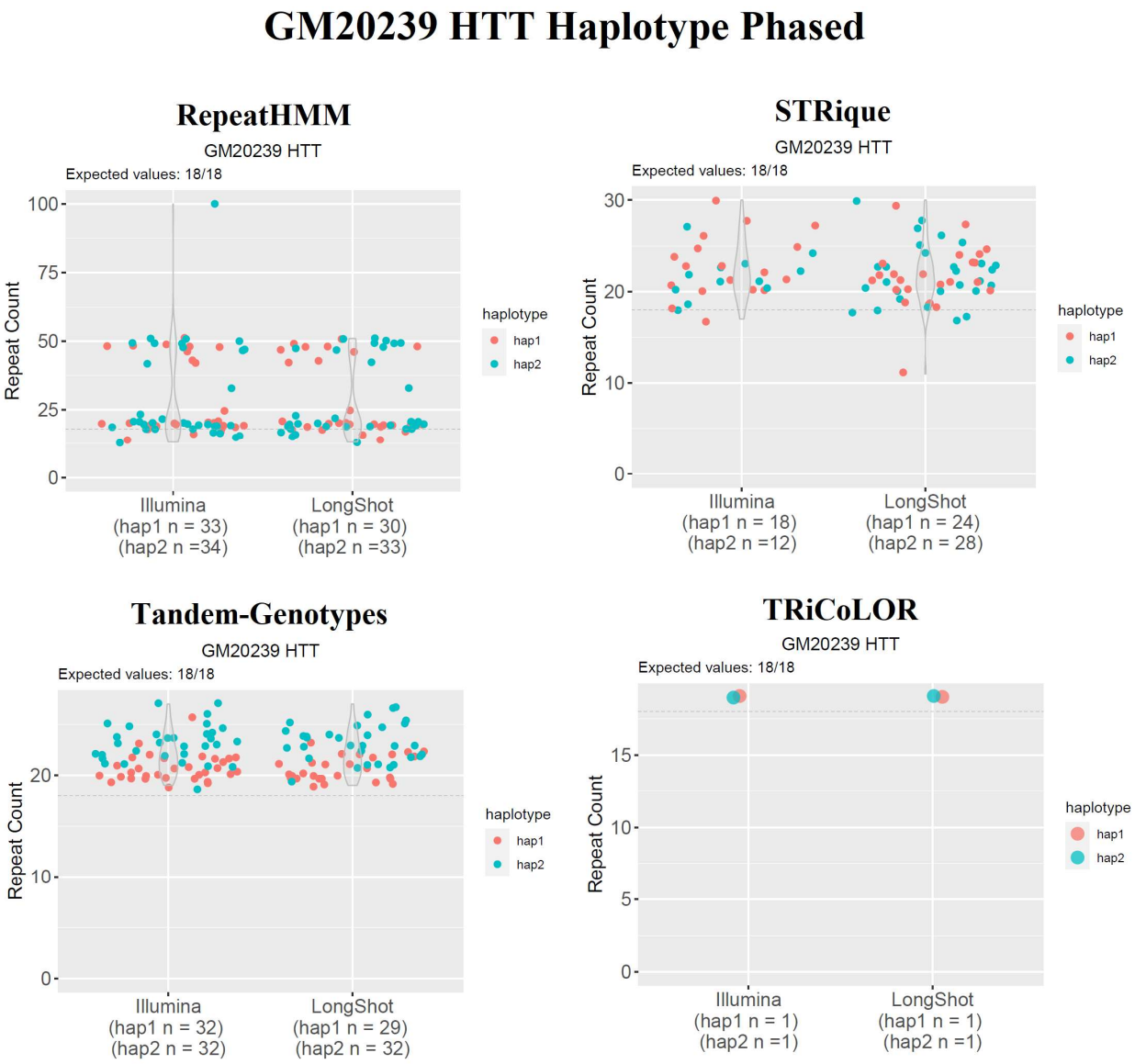


Figure 29: GM07541 HTT Repeat Counts Haplotype Phased

The combined data has been haplotype phased and run with the following four programs: RepeatHMM, STRique, Tandem-Genotypes, and TRiCoLOR. The first column inside each plot presents the sample data phased with the in-house Illumina HP method, and the second column presents the sample data phased with LongShot. The expected repeat length is indicated with dashed horizontal lines.

3.3.3 Long-read Haplotype Phased Results Impact on Clinical Diagnostics

In this case, none of the samples had HP results that would have changed the clinical diagnosis of the patient regarding HD, except if one were to use RepeatHMM for estimation of repeat lengths. None of the samples have mean alleles where the repeat length shifts from being a normal allele to being an affected allele.

Though, the better repeat estimating programs (STRique, Tandem-Genotypes, TRiCoLOR) are at times estimating up to 6 repeat lengths off in the HTT gene, which could have clinical implications. Nanopore is not ready for clinical use for short repeats like HTT. More testing and possibly improvements would be needed.

One of the samples could have clinical implications regarding FXS. This sample is GM20239. For the FMR1 gene, there are up to 6 repeat lengths off normal alleles but there is a much higher variation on longer repeats. For GM20239 one allele is supposed to have a repeat length between 183 and 199, but one of the programs estimates the allele to be 246 repeats long which is 37-49 repeats too high. This could have clinical implications, especially if there were a patient with a normal allele on 45 repeats since they would be diagnosed as a premutation, which can lead to health implications such as FXTAS or primary ovarian insufficiency. If a patient had a premutation allele with length 180 they would be wrongly diagnosed with FXS.

It is though important to remember, that a clinical evaluation of a patient considers more parameters than repeat length such as disease symptoms. A patient that has symptoms that could be the outcome of multiple diseases would though be at risk of getting a wrong diagnosis if Nanopore was being used as the only diagnostic technique since the repeat lengths can be off.

Sub-Conclusion for Haplotype Phased Count Results

Haplotype phasing (HP) was performed for two reasons: (1) to run TRiCoLOR, which takes phased files as input, and (2) to separate alleles with repeat length close to equal. The performance of TRiCoLOR is excelling compared to RepeatHMM, STRique, and Tandem-Genotypes, making HP a necessary step in the analysis pipeline for optimal repeat estimations.

Alleles having close to equal repeat length are separated for all programs when taking the mean of the alleles. The alleles are not visually separable when looking at the total data from RepeatHMM, STRique, and Tandem-Genotypes. These inseparable alleles are most likely the result of sequencing errors and not poorly constructed haplotypes. Haplotypes constructed with SNPs from

Illumina data do not excel over haplotypes constructed with SNPs from Oxford Nanopore data. The haplotypes constructed with LongShot are therefore recommended over the in-house algorithm, as Longshot only requires long-read sequencing data.

4 Conclusion and Perspective

In this project there are three sub-topics: (1) targeted long-read ONT sequencing using ReadFish, (2) estimating repeat counts, and (3) phasing sequenced data for better repeat length estimation. Targeted sequencing with ReadFish gave an enrichment with an average fold-change between 2.34 and 3.92, for the target panel consisting of 27 known repeat diseases, which falls within the expected range of 2.7-5.4x.

Repeat length estimation is not reliable when using RepeatHMM as there is an indication of disease-specific performance since it does a better job at estimating repeat lengths in FMR1 compared to HTT. RepeatHMM cannot be recommended as it overestimates the repeat length of a good portion of DNA-reads. Correct repeat length estimation is possible, though alleles differing with less than 5 repeat units are hard to distinguish. Repeat lengths were estimated close to correct when using TRiCoLOR, which requires haplotype phased input making it a necessary step in the analysis pipeline. For an optimal analysis pipeline, LongShot is the recommended program since using SNP positions from Illumina sequencing does not give a better performance in haplotype phasing compared to using SNP positions from ONT sequencing.

Haplotype phasing did not aid in separating the alleles when the repeats lengths were almost equal unless the mean repeat length was taken for the programs, which one would do in a clinical setting as one number for the repeat lengths is needed. The inability of haplotype phasing alleles with close to equal repeat lengths are most likely due to error-prone sequencing causing fluctuations in the repeat length estimations from the same allele. This makes haplotype phasing unnecessary if analyzing repeats with STRique or Tandem-Genotypes, which had performances close to that of TRiCoLOR. The errors in long-read sequencing are though preferred compared to using short-read sequencing, such as Illumina, when estimating repeat lengths for alleles with an FXS full mutation since short-reads often do not span the entire repeat.

Targeted sequencing of repeat regions is not ready for clinical use. At the moment, the current methods such as PCR and Southern blotting are irreplaceable as they are more precise than long-read

sequencing. Further testing, sequencing improvements, and program development is needed as a precise genotype is necessary for diagnostics. To improve the genotyping accuracy of repeats using ONT it would be relevant to test the ability of the CRISPR/Cas9 method for enrichment. The new model of flow-cells could improve sequencing and possibly the estimation of repeat lengths. One could furthermore analyze methylation patterns with STRique, when appropriate for the disease. The accuracy of the programs could also be evaluated by running an experiment with a DNA-library consisting of DNA-fragments with known repeat lengths.

Literature

Ameur, A., W. P. Kloosterman, and M. S. Hestand. 2019. 'Single-Molecule Sequencing: Towards Clinical Applications', *Trends Biotechnol*, 37: 72-85.

Bahlo, M., M. F. Bennett, P. Degorski, R. M. Tankard, M. B. Delatycki, and P. J. Lockhart. 2018. 'Recent advances in the detection of repeat expansions with short-read next-generation sequencing', *F1000Res*, 7.

Bean, Lora, Pinar Bayrak-Toydemir, and on behalf of the Acmg Laboratory Quality Assurance Committee. 2014. 'American College of Medical Genetics and Genomics Standards and Guidelines for Clinical Genetics Laboratories, 2014 edition: technical standards and guidelines for Huntington disease', *Genetics in Medicine*, 16: e2-e2.

Bell, M. V., M. C. Hirst, Y. Nakahori, R. N. MacKinnon, A. Roche, T. J. Flint, P. A. Jacobs, N. Tommerup, L. Tranebjaerg, U. Froster-Iskenius, and et al. 1991. 'Physical mapping across the fragile X: hypermethylation and clinical expression of the fragile X syndrome', *Cell*, 64: 861-6.

Biolabs, New England. 2020. 'nebiocalculator version 1.12.0'. https://nebiocalculator.neb.com/.

Bolognini, Davide, Alberto Magi, Vladimir Benes, Jan O Korbel, and Tobias Rausch. 2020. 'TRiCoLOR: tandem repeat profiling using whole-genome long-read sequencing data', *Gigascience*, 9.

Brinkman, R. R., M. M. Mezei, J. Theilmann, E. Almqvist, and M. R. Hayden. 1997. 'The likelihood of being affected with Huntington disease by a particular age, for a specific CAG size', *Am J Hum Genet*, 60: 1202-10.

Campbell, Neil A., Jane B. Reece, Lisa A. Urry, Michael L. Cain, Steven Alexander Wasserman, Peter V. Minorsky, and Robert B. Jackson. 2015. *Biology : a global approach*.

Coriell. 'GM04284: LCL from B-Lymphocyte', Accessed 31-01-2020. https://www.coriell.org/0/Sections/Search/Sample_Detail.aspx?Ref=GM04284&Product=CC.

———. 'GM05538: LCL from B-Lymphocyte', Accessed 31-01-2020. https://www.coriell.org/0/Sections/Search/Sample_Detail.aspx?Ref=GM05538&Product=CC.

———. 'GM06891: LCL from B-Lymphocyte', Accessed 31-01-2020. https://www.coriell.org/0/Sections/Search/Sample_Detail.aspx?Ref=GM06891&Product=CC.

———. 'GM07541: LCL from B-Lymphocyte', Accessed 31-01-2020. https://www.coriell.org/0/Sections/Search/Sample_Detail.aspx?Ref=GM07541&Product=CC.
———. 'GM07861: LCL from B-Lymphocyte', Accessed 31-01-2020. https://www.coriell.org/0/Sections/Search/Sample_Detail.aspx?Ref=GM07861&Product=CC.
———. 'GM20239: LCL from B-Lymphocyte', Accessed 31-01-2020. https://www.coriell.org/0/Sections/Search/Sample_Detail.aspx?Ref=GM20239&Product=CC.
Cretu Stancu, M., M. J. van Roosmalen, I. Renkens, M. M. Nieboer, S. Middelkamp, J. de Ligt, G. Pregno, D. Giachino, G. Mandrile, J. Espejo Valle-Inclan, J. Korzelius, E. de Bruijn, E. Cuppen, M. E. Talkowski, T. Marschall, J. de Ridder, and W. P. Kloosterman. 2017. 'Mapping and phasing of structural variation in patient genomes using nanopore sequencing', *Nat Commun*, 8: 1326.
De Roeck, A., W. De Coster, L. Bossaerts, R. Cacace, T. De Pooter, J. Van Dongen, S. D'Hert, P. De Rijk, M. Strazisar, C. Van Broeckhoven, and K. Sleegers. 2019. 'NanoSatellite: accurate characterization of expanded tandem repeat length and sequence through whole genome long-read sequencing on PromethION', *Genome Biol*, 20: 239.
Den Dunnen, W. F. A. 2017. 'Trinucleotide repeat disorders', *Handb Clin Neurol*, 145: 383-91.
Dolzhenko, E., V. Deshpande, F. Schlesinger, P. Krusche, R. Petrovski, S. Chen, D. Emig-Agius, A. Gross, G. Narzisi, B. Bowman, K. Scheffler, Jjfa van Vugt, C. French, A. Sanchis-Juan, K. Ibáñez, A. Tucci, B. R. Lajoie, J. H. Veldink, F. L. Raymond, R. J. Taft, D. R. Bentley, and M. A. Eberle. 2019. 'ExpansionHunter: a sequence-graph-based tool to analyze variation in short tandem repeat regions', *Bioinformatics*, 35: 4754-56.
Eberle, M. A., E. Fritzilas, P. Krusche, M. Källberg, B. L. Moore, M. A. Bekritsky, Z. Iqbal, H. Y. Chuang, S. J. Humphray, A. L. Halpern, S. Kruglyak, E. H. Margulies, G. McVean, and D. R. Bentley. 2017. 'A reference data set of 5.4 million phased human variants validated by genetic inheritance from sequencing a three-generation 17-member pedigree', *Genome Res*, 27: 157-64.
Edge, P., V. Bafna, and V. Bansal. 2017. 'HapCUT2: robust and accurate haplotype assembly for diverse sequencing technologies', *Genome Res*, 27: 801-12.
Edge, P., and V. Bansal. 2019. 'Longshot enables accurate variant calling in diploid genomes from single-molecule long read sequencing', *Nat Commun*, 10: 4660.
Eid, J., A. Fehr, J. Gray, K. Luong, J. Lyle, G. Otto, P. Peluso, D. Rank, P. Baybayan, B. Bettman, A. Bibillo, K. Bjornson, B. Chaudhuri, F. Christians, R. Cicero, S. Clark, R. Dalal, A. Dewinter, J. Dixon, M. Foquet, A. Gaertner, P. Hardenbol, C. Heiner, K. Hester, D. Holden, G. Kearns, X. Kong, R. Kuse, Y. Lacroix, S. Lin, P. Lundquist, C. Ma, P. Marks, M. Maxham, D. Murphy, I. Park, T. Pham, M. Phillips, J. Roy, R. Sebra, G. Shen, J. Sorenson, A. Tomaney, K. Travers, M. Trulson, J. Vieceli, J. Wegener, D. Wu, A. Yang, D. Zaccarin, P. Zhao, F. Zhong, J. Korlach, and S. Turner. 2009. 'Real-time DNA sequencing from single polymerase molecules', *Science*, 323: 133-8.
Flanagan, J. M., V. Popendikyte, N. Pozdniakovaite, M. Sobolev, A. Assadzadeh, A. Schumacher, M. Zangeneh, L. Lau, C. Virtanen, S. C. Wang, and A. Petronis. 2006. 'Intra- and interindividual epigenetic variation in human germ cells', *Am J Hum Genet*, 79: 67-84.
Giesselmann, P., B. Brandl, E. Raimondeau, R. Bowen, C. Rohrandt, R. Tandon, H. Kretzmer, G. Assum, C. Galonska, R. Siebert, O. Ammerpohl, A. Heron, S. A. Schneider, J. Ladewig, P. Koch, B. M. Schuldt, J. E. Graham, A. Meissner, and F. J. Muller. 2019. 'Analysis of short

tandem repeat expansions and their methylation state with nanopore sequencing', *Nat Biotechnol*, 37: 1478-81.
Hagerman, Paul J., and Randi J. Hagerman. 2004. 'The Fragile-X Premutation: A Maturing Perspective', *The American Journal of Human Genetics*, 74: 805-16.
Jensen, Peter K. A. 2011. *Kromosomafvigelser hos mennesket* (Gyldendal: Kbh.).
Kelly, T. E., P. Allinson, R. C. McGlennen, J. Baker, and Y. Bao. 1999. 'Expansion of a 27 CAG repeat allele into a symptomatic huntington disease-producing allele', *Am J Med Genet*, 87: 91-2.
Labun, Kornel, Tessa G Montague, Maximilian Krause, Yamila N Torres Cleuren, Håkon Tjeldnes, and Eivind Valen. 2019. 'CHOPCHOP v3: expanding the CRISPR web toolbox beyond genome editing', *Nucleic Acids Research*, 47: W171-W74.
Leggett, R. M., and M. D. Clark. 2017. 'A world of opportunities with nanopore sequencing', *J Exp Bot*, 68: 5419-29.
Li, H. 2018. 'Minimap2: pairwise alignment for nucleotide sequences', *Bioinformatics*, 34: 3094-100.
Liu, Q., P. Zhang, D. Wang, W. Gu, and K. Wang. 2017. 'Interrogating the "unsequenceable" genomic trinucleotide repeat disorders by long-read sequencing', *Genome Med*, 9: 65.
Lu, H., F. Giordano, and Z. Ning. 2016. 'Oxford Nanopore MinION Sequencing and Genome Assembly', *Genomics Proteomics Bioinformatics*, 14: 265-79.
MacDonald, Marcy E., Christine M. Ambrose, Mabel P. Duyao, Richard H. Myers, Carol Lin, Lakshmi Srinidhi, Glenn Barnes, Sherryl A. Taylor, Marianne James, Nicolet Groot, Heather MacFarlane, Barbara Jenkins, Mary Anne Anderson, Nancy S. Wexler, James F. Gusella, Gillian P. Bates, Sarah Baxendale, Holger Hummerich, Susan Kirby, Mike North, Sandra Youngman, Richard Mott, Gunther Zehetner, Zdenek Sedlacek, Annemarie Poustka, Anna-Maria Frischauf, Hans Lehrach, Alan J. Buckler, Deanna Church, Lynn Doucette-Stamm, Michael C. O'Donovan, Laura Riba-Ramirez, Manish Shah, Vincent P. Stanton, Scott A. Strobel, Karen M. Draths, Jennifer L. Wales, Peter Dervan, David E. Housman, Michael Altherr, Rita Shiang, Leslie Thompson, Thomas Fielder, John J. Wasmuth, Danilo Tagle, John Valdes, Lawrence Elmer, Marc Allard, Lucio Castilla, Manju Swaroop, Kris Blanchard, Francis S. Collins, Russell Snell, Tracey Holloway, Kathleen Gillespie, Nicole Datson, Duncan Shaw, and Peter S. Harper. 1993. 'A novel gene containing a trinucleotide repeat that is expanded and unstable on Huntington's disease chromosomes', *Cell*, 72: 971-83.
Mitsuhashi, S., M. C. Frith, T. Mizuguchi, S. Miyatake, T. Toyota, H. Adachi, Y. Oma, Y. Kino, H. Mitsuhashi, and N. Matsumoto. 2019. 'Tandem-genotypes: robust detection of tandem repeat expansions from long DNA reads', *Genome Biol*, 20: 58.
Monaghan, K. G., E. Lyon, and E. B. Spector. 2013. 'ACMG Standards and Guidelines for fragile X testing: a revision to the disease-specific supplements to the Standards and Guidelines for Clinical Genetics Laboratories of the American College of Medical Genetics and Genomics', *Genet Med*, 15: 575-86.
Myers, R. H., M. E. MacDonald, W. J. Koroshetz, M. P. Duyao, C. M. Ambrose, S. A. Taylor, G. Barnes, J. Srinidhi, C. S. Lin, W. L. Whaley, and et al. 1993. 'De novo expansion of a (CAG)n repeat in sporadic Huntington's disease', *Nat Genet*, 5: 168-73.
Nørby, Søren

Nielsen, K.A. Peter. 2006. *Medicinsk genetik* (FADL: Kbh.).
Oberlé, I., F. Rousseau, D. Heitz, C. Kretz, D. Devys, A. Hanauer, J. Boué, M. F. Bertheas, and J. L. Mandel. 1991. 'Instability of a 550-base pair DNA segment and abnormal methylation in fragile X syndrome', *Science*, 252: 1097-102.
ONT. 'MinKNOW Github Page'.
———. 2003 Updated 2018. "Nanopore Protocol: Guppy Software Overview." In.

———. 2020. 'ONT pyGuppy Client', Accessed 27/10-2020. https://pypi.org/project/ont-pyguppy-client-lib/#description.
Parnell, L. D. 2012. 'Advances in technologies and study design', *Prog Mol Biol Transl Sci*, 108: 17-50.
Payne, Alexander, Nadine Holmes, Thomas Clarke, Rory Munro, Bisrat Debebe, and Matthew Loose. 2020a. 'Nanopore adaptive sequencing for mixed samples, whole exome capture and targeted panels', *bioRxiv*: 2020.02.03.926956.
Payne, Alexander, Nadine Holmes, Thomas Clarke, Rory Munro, Bisrat Debebe, and Matthew W Loose. 2020b. 'Nanopore adaptive sequencing for mixed samples, whole exome capture and targeted panels', *bioRxiv*: 2020.02.03.926956.
Pedersen, B. S., and A. R. Quinlan. 2018. 'Mosdepth: quick coverage calculation for genomes and exomes', *Bioinformatics*, 34: 867-68.
Roos, R. A. 2010. 'Huntington's disease: a clinical review', *Orphanet J Rare Dis*, 5: 40.
Ross, C. A., E. H. Aylward, E. J. Wild, D. R. Langbehn, J. D. Long, J. H. Warner, R. I. Scahill, B. R. Leavitt, J. C. Stout, J. S. Paulsen, R. Reilmann, P. G. Unschuld, A. Wexler, R. L. Margolis, and S. J. Tabrizi. 2014. 'Huntington disease: natural history, biomarkers and prospects for therapeutics', *Nat Rev Neurol*, 10: 204-16.
Schneider, T. D., and R. M. Stephens. 1990. 'Sequence logos: a new way to display consensus sequences', *Nucleic Acids Res*, 18: 6097-100.
Scientific, ThermoFisher. 'What is fragment analysis?', Accessed 20-01-2021. https://www.thermofisher.com/dk/en/home/life-science/sequencing/sequencing-learning-center/capillary-electrophoresis-information/what-is-fragment-analysis.html.
Sedlazeck, F. J., P. Rescheneder, M. Smolka, H. Fang, M. Nattestad, A. von Haeseler, and M. C. Schatz. 2018. 'Accurate detection of complex structural variations using single-molecule sequencing', *Nat Methods*, 15: 461-68.
Technologies, Ocxford Nanopore. 2016a. "Nanopore Sensing - How it Works." In *Nanopore technical document*.
Technologies, Oxford Nanopore. 2016b. "Chemistry Technical Document." In *Nanopore Technical Document*.
———. 2016c. "Data Analysis: Basecalling Overview." In *Nanopore Technical Document*.
———. 2018a. 'Ligation Sequencing Kit SQK-LSK109', Oxford Nanopore Technologies. https://store.nanoporetech.com/eu/sample-prep/ligation-sequencing-kit.html.
———. 2018b. "Targeted, amplification-free DNA sequencing using CRISPR/Cas." In *Nanopore Info sheet*.
———. 2020a. "Cas9 Sequencing Kit (SQK-CS9109)." In *Nanopre protocol*.
———. 2020b. 'Ligation Sequencing Kit SQK-LSK110', Oxford Nanopore Technologies. https://store.nanoporetech.com/eu/ligation-sequencing-kit110.html.
Trottier, Y., V. Biancalana, and J. L. Mandel. 1994. 'Instability of CAG repeats in Huntington's disease: relation to parental transmission and age of onset', *J Med Genet*, 31: 377-82.
Turner, G., H. Robinson, S. Laing, A. Goddard, M. van den Berk, S. Sherman, A. Colley, and M. Partington. 1992. 'Population screening for fragile X', *The Lancet*, 339: 1210-13.
Verkerk, A. J., M. Pieretti, J. S. Sutcliffe, Y. H. Fu, D. P. Kuhl, A. Pizzuti, O. Reiner, S. Richards, M. F. Victoria, F. P. Zhang, and et al. 1991. 'Identification of a gene (FMR-1) containing a CGG repeat coincident with a breakpoint cluster region exhibiting length variation in fragile X syndrome', *Cell*, 65: 905-14.
Wagih, O. 2017. 'ggseqlogo: a versatile R package for drawing sequence logos', *Bioinformatics*, 33: 3645-47.

Wick, R. R., L. M. Judd, and K. E. Holt. 2019. 'Performance of neural network basecalling tools for Oxford Nanopore sequencing', *Genome Biol*, 20: 129.
Willemsen, R., J. Levenga, and B. A. Oostra. 2011. 'CGG repeat in the FMR1 gene: size matters', *Clin Genet*, 80: 214-25.
Yu, S., M. Pritchard, E. Kremer, M. Lynch, J. Nancarrow, E. Baker, K. Holman, J. C. Mulley, S. T. Warren, D. Schlessinger, and et al. 1991. 'Fragile X genotype characterized by an unstable region of DNA', *Science*, 252: 1179-81.
Zhou, A., T. Lin, and J. Xing. 2019. 'Evaluating nanopore sequencing data processing pipelines for structural variation identification', *Genome Biol*, 20: 237.

Appendix 1: GitHub Appendix

The link below takes you to my GitHub repository created as an online Appendix for my code.

https://github.com/Tine-dk/Master_Thesis

Appendix 2: Readfish Environment Package List

Name	Version	Build	Channel
_libgcc_mutex	0.1	conda_forge	conda-forge
_openmp_mutex	4.5	1_gnu	conda-forge
attrs	20.2.0	pypi_0	pypi
beautifulsoup4	4.9.1	pypi_0	pypi
biopython	1.78	pypi_0	pypi
bzip2	1.0.8	h516909a_3	conda-forge
ca-certificates	2020.6.20	hecda079_0	conda-forge
certifi	2020.6.20	py37hc8dfbb8_0	conda-forge
chardet	3.0.4	pypi_0	pypi
flatbuffers	1.11	pypi_0	pypi
google	3.0.0	pypi_0	pypi
grpcio	1.32.0	pypi_0	pypi
h5py	2.10.0	pypi_0	pypi
idna	2.10	pypi_0	pypi
importlib-metadata	2.0.0	pypi_0	pypi
jsonschema	3.2.0	pypi_0	pypi

libffi	3.2.1	he1b5a44_1007	conda-forge
libgcc-ng	9.3.0	h24d8f2e_16	conda-forge
libgomp	9.3.0	h24d8f2e_16	conda-forge
libstdcxx-ng	9.3.0	hdf63c60_16	conda-forge
mappy	2.17	pypi_0	pypi
minknow-api	4.0.4	pypi_0	pypi
ncurses	6.2	he1b5a44_1	conda-forge
numpy	1.19.2	pypi_0	pypi
ont-fast5-api	3.1.6	pypi_0	pypi
openssl	1.0.2u	h516909a_0	conda-forge
packaging	20.4	pypi_0	pypi
pandas	1.1.2	pypi_0	pypi
pathtools	0.1.2	pypi_0	pypi
pip	20.2.3	py_0	conda-forge
progressbar33	2.4	pypi_0	pypi
protobuf	3.13.0	pypi_0	pypi
pyguppyclient	0.0.6	pypi_0	pypi
pyparsing	2.4.7	pypi_0	pypi
pyrsistent	0.17.3	pypi_0	pypi
python	3.7.0	hd21baee_1006	conda-forge
python-dateutil	2.8.1	pypi_0	pypi
python_abi	3.7	1_cp37m	conda-forge
pytz	2020.1	pypi_0	pypi
pyzmq	17.1.2	pypi_0	pypi
read-until	2.1.0	pypi_0	pypi
readfish	0.0.5a3	pypi_0	pypi
readline	7.0	hf8c457e_1001	conda-forge
requests	2.24.0	pypi_0	pypi
setuptools	49.6.0	py37hc8dfbb8_1	conda-forge
six	1.15.0	pypi_0	pypi
soupsieve	2.0.1	pypi_0	pypi
sqlite	3.28.0	h8b20d00_0	conda-forge
tk	8.6.10	hed695b0_0	conda-forge
toml	0.10.1	pypi_0	pypi
urllib3	1.25.10	pypi_0	pypi

watchdog	0.10.3	pypi_0	pypi
wheel	0.35.1	pyh9f0ad1d_0	conda-forge
xz	5.2.5	h516909a_1	conda-forge
zipp	3.2.0	pypi_0	pypi
zlib	1.2.11	h516909a_1009	conda-forge

Appendix 3: Targeted Sequencing Supplementary Figures

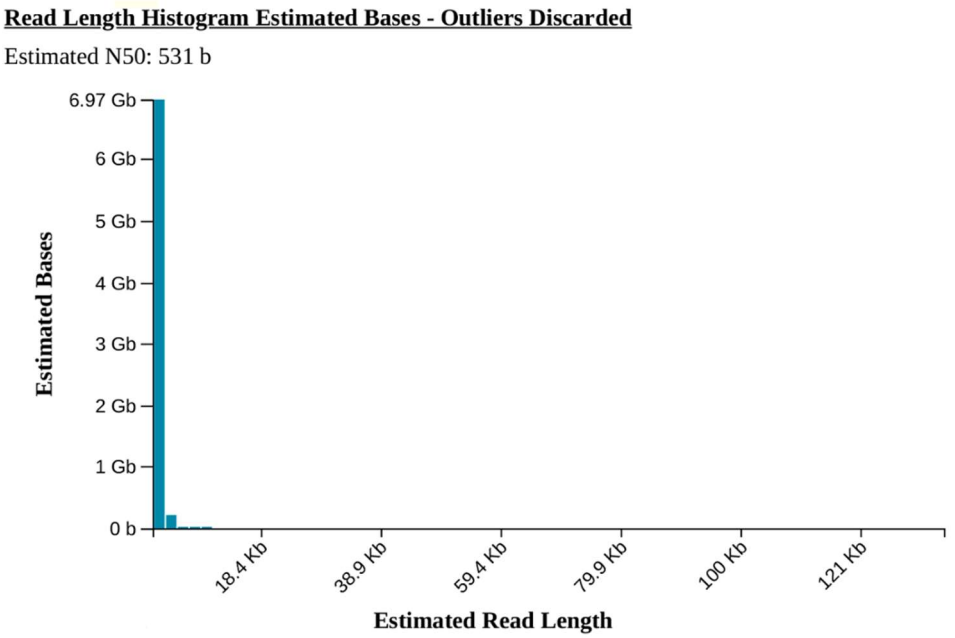


Figure 30: Read Length Histogram of Experiment 1.

This histogram shows the estimated bases on the y-axis and the estimated read length on the x-axis. The estimated N50 is 531 bp. There are bases covering the whole x-axis but the number of bases at the low-end is skewing the proportions of the cones in the histogram. This histogram is taken from the final MinKNOW report.

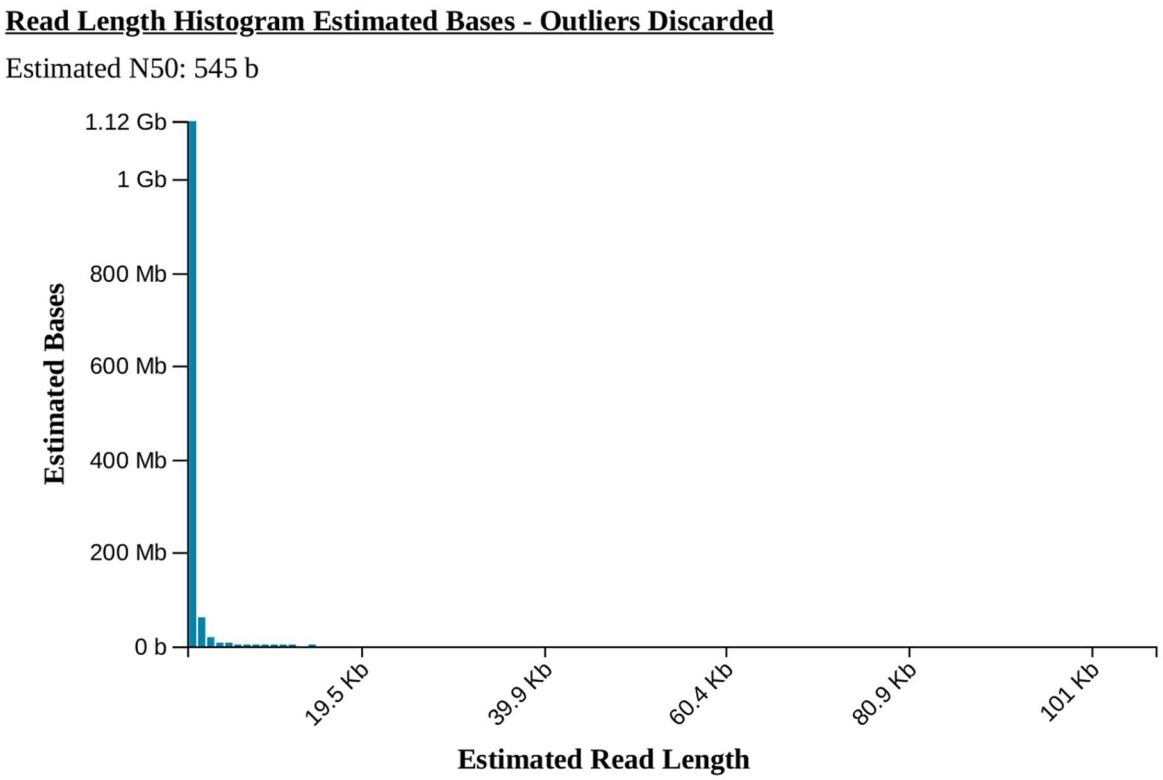


Figure 31: Read Length Histogram of Experiment 2 day 1.

This histogram shows the estimated bases on the y-axis and the estimated read length on the x-axis. The estimated N50 is 603 bp. There are bases covering the whole x-axis but the number of bases at the low-end is skewing the proportions of the cones in the histogram. This histogram is taken from the MinKNOW report of day 1 after ~3 hours of sequencing.

Read Length Histogram Estimated Bases

Estimated N50: 608 b

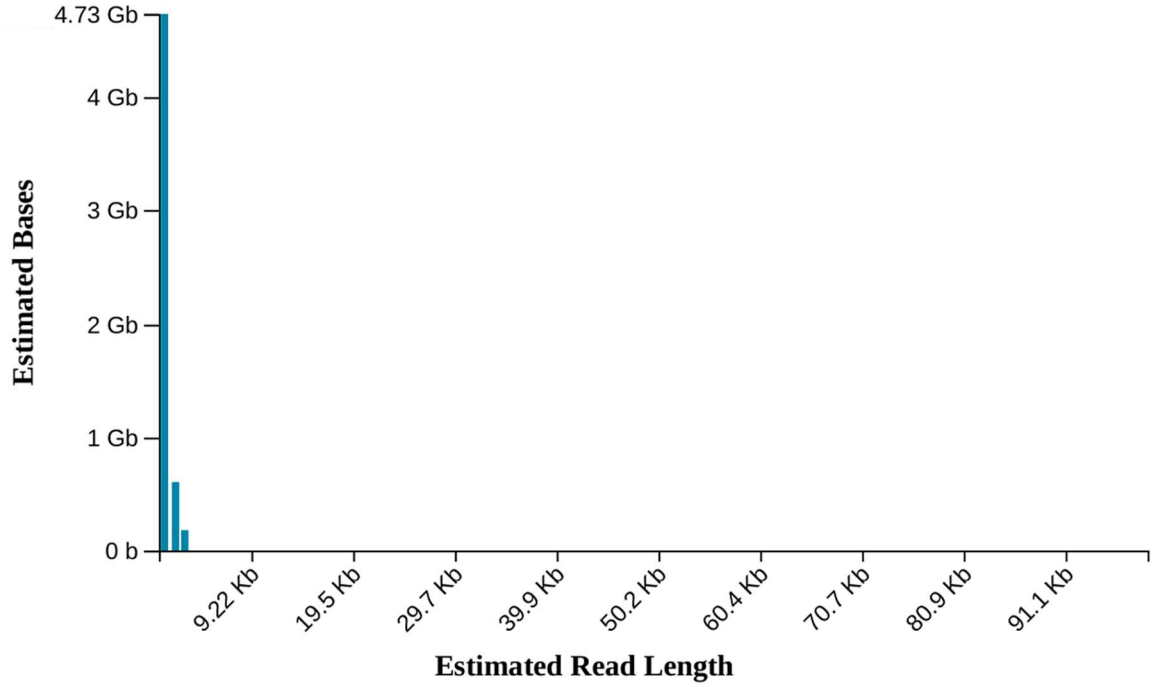


Figure 32: Read Length Histogram of Experiment 2 day 3.

This histogram shows the estimated bases on the y-axis and the estimated read length on the x-axis. The estimated N50 is 603 bp. There are bases covering the whole x-axis but the number of bases at the low-end is skewing the proportions of the cones in the histogram. This histogram is taken from the MinKNOW report of day 3 after ~48 hours of sequencing.

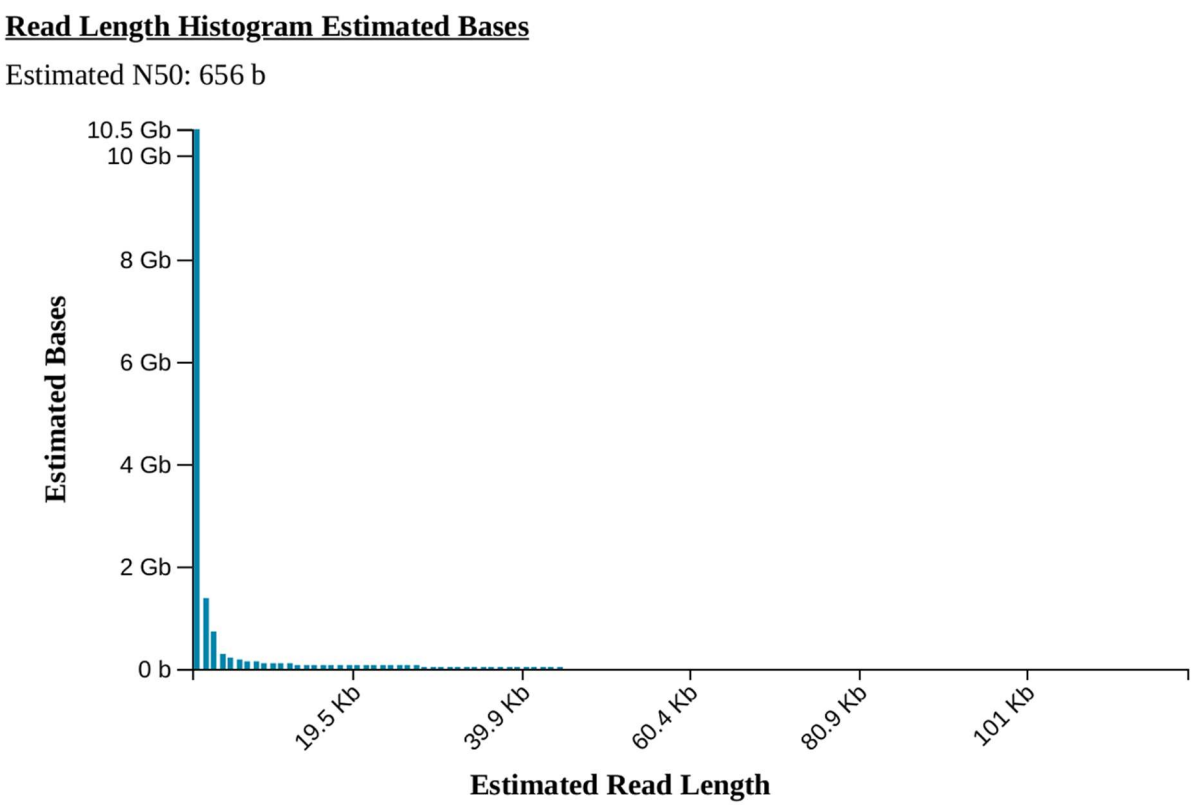


Figure 33: Read Length Histogram of experiment 3.

This histogram shows the estimated bases on the y-axis and the estimated read length on the x-axis. The estimated N50 is 656 bp. There are bases covering the whole x-axis but the number of bases at the low-end is skewing the proportions of the cones in the histogram. This histogram is taken from the MinKNOW report.

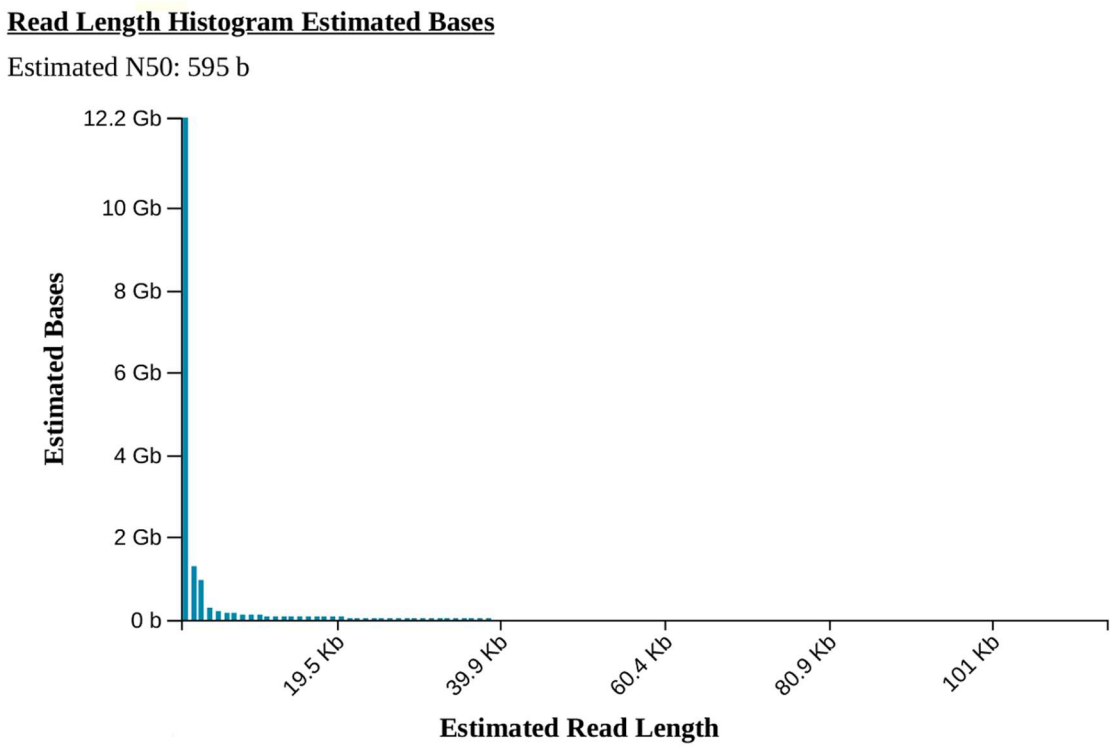


Figure 34: Read Length Histogram of experiment 4.
This histogram shows the estimated bases on the y-axis and the estimated read length on the x-axis. The estimated N50 is 595 bp. There are bases covering the whole x-axis but the number of bases at the low-end is skewing the proportions of the cones in the histogram. This histogram is taken from the MinKNOW report.

Appendix 4: Targeted Sequencing Depth Tables

These tables are created with MosDepth.

Table 1: GM06891 targets depth. *This table is the output from Mosdepth given a bedfile with the targets. The table shows the depth at each region. Huntinton's Disease and Fragile X is marked with bold as it is these two diseases I have been focusing on when analysing the whole genome data. Fast Guppy basecalling was used for this sample together with ReadFish.*

Chr	Start	Stop	Disease	Mean Depth
1	57782716	57882916	SCA37	4.00
3	63848361	63948561	SCA7	7.57
3	128841420	128941620	DM2	4.36
4	**3026604**	**3126770**	**HD**	**6.23**
5	146208291	146308491	SCA12	4.46
6	16277865	16378050	SCA1	6.60
6	170820995	170921200	SCA17	4.47
8	119329055	119429255	FAME1	7.75

9	27523483	27623683	FTDALS1	5.45
9	71602201	71702401	FRDA	6.91
12	6995880	7096100	DRPLA	4.92
12	111986754	112086950	SCA2	5.59
13	70663516	70763716	SCA8	5.08
14	23740682	23840682	OPMD	4.58
14	92487355	92587555	SCA3	7.14
16	66474302	66574502	SCA31	5.55
16	87587899	87688100	HDL2	5.83
18	53203385	53303585	FECD3	5.62
19	13268673	13368873	SCA6	5.67
19	46223463	46323663	DM1	5.47
20	2583379	2683579	SCA36	5.98
21	45146324	45246524	EPM1	6.14
22	46141235	46241435	SCA10	6.62
X	24981771	25081971	EIEE1	2.33
X	66715159	66815360	SBMA	3.85
X	**146943555**	**147043755**	**FRAXA**	**4.30**
X	147532159	147632359	FRAXE	2.07

Table 1: GM07541 targets depth. *This table is the output from Mosdepth given a bedfile with the targets. The table shows the depth at each region. Huntinton's Disease and Fragile X is marked with bold as it is these two diseases I have been focusing on when analysing the whole genome data. HAC Guppy basecalling was used for this sample together with ReadFish.*

Chr	Start	Stop	Disease	Mean Depth
1	57782716	57882916	SCA37	4.00
3	63848361	63948561	SCA7	7.13
3	128841420	128941620	DM2	6.83
4	**3026604**	**3126770**	**HD**	**6.46**
5	146208291	146308491	SCA12	6.22
6	16277865	16378050	SCA1	5.57
6	170820995	170921200	SCA17	5.78
8	119329055	119429255	FAME1	4.87
9	27523483	27623683	FTDALS1	5.04
9	71602201	71702401	FRDA	4.90
12	6995880	7096100	DRPLA	6.37
12	111986754	112086950	SCA2	4.59
13	70663516	70763716	SCA8	5.11
14	23740682	23840682	OPMD	6.09
14	92487355	92587555	SCA3	6.11
16	66474302	66574502	SCA31	5.13
16	87587899	87688100	HDL2	4.78
18	53203385	53303585	FECD3	5.58
19	13268673	13368873	SCA6	5.16
19	46223463	46323663	DM1	5.30

20	2583379	2683579	SCA36	6.02
21	45146324	45246524	EPM1	5.59
22	46141235	46241435	SCA10	6.06
X	24981771	25081971	EIEE1	5.99
X	66715159	66815360	SBMA	5.07
X	**146943555**	**147043755**	**FRAXA**	**4.64**
X	147532159	147632359	FRAXE	4.11

Table 1: GM20239 targets depth. *This table is the output from Mosdepth given a bedfile with the targets. The table shows the depth at each region. Huntinton's Disease and Fragile X is marked with bold as it is these two diseases I have been focusing on when analysing the whole genome data. HAC Guppy basecalling was used for this sample together with ReadFish.*

chr	start	stop	disease	mean_depth
1	57332716	58332716	SCA37	17.72
3	63398361	64398361	SCA7	17.84
3	1,28E+08	1,29E+08	DM2	21.10
4	**2576604**	**3576604**	**HD**	**21.36**
5	1,46E+08	1,51E+08	SCA12	NA
6	15827865	16827865	SCA1	20.01
6	1,7E+08	1,71E+08	SCA17	16.82
8	1,19E+08	1,2E+08	FAME1	19.30
9	27073483	28073483	FTDALS1	18.75
9	71152201	72152201	FRDA	19.73
12	6545880	7545880	DRPLA	21.35
12	1,12E+08	1,13E+08	SCA2	21.05
13	70213516	71213516	SCA8	17.07
14	23290682	24290682	OPMD	21.67
14	92037355	93037355	SCA3	20.03
16	66474302	66574502	SCA31	NA
16	87137889	88137889	HDL2	20.22
18	52753385	53753385	FECD3	19.29
19	12818673	13818673	SCA6	21.71
19	45773463	46773463	DM1	21.38
20	2133379	3133379	SCA36	20.47
21	44696324	45696324	EPM1	20.30
22	45691235	46691235	SCA10	20.52
X	24531771	25531771	EIEE1	18.80
X	66265159	67265159	SBMA	16.27
X	**1,46E+08**	**1,47E+08**	**FRAXA**	**16.65**
X	1,47E+08	1,48E+08	FRAXE	18.16

Table 1: GM07861 targets depth. *This table is the output from Mosdepth given a bedfile with the targets. The table shows the depth at each region. Huntinton's Disease and Fragile X is marked with bold as it is these two diseases I have been focusing on when analysing the whole genome data. HAC Guppy basecalling was used for this sample together with ReadFish.*

chr	start	stop	disease	mean_depth
1	57332716	58332716	SCA37	20.79
3	63398361	64398361	SCA7	22.29
3	1,28E+08	1,29E+08	DM2	21.82
4	**2576604**	**3576604**	**HD**	**21.80**
5	1,46E+08	1,51E+08	SCA12	NA
6	15827865	16827865	SCA1	24.04
6	1,7E+08	1,71E+08	SCA17	18.14
8	1,19E+08	1,2E+08	FAME1	22.23
9	27073483	28073483	FTDALS1	22.12
9	71152201	72152201	FRDA	22.75
12	6545880	7545880	DRPLA	20.56
12	1,12E+08	1,13E+08	SCA2	23.57
13	70213516	71213516	SCA8	21.46
14	23290682	24290682	OPMD	22.31
14	92037355	93037355	SCA3	23.13
16	66474302	66574502	SCA31	NA
16	87137889	88137889	HDL2	21.53
18	52753385	53753385	FECD3	22.66
19	12818673	13818673	SCA6	23.52
19	45773463	46773463	DM1	21.59
20	2133379	3133379	SCA36	22.55
21	44696324	45696324	EPM1	21.58
22	45691235	46691235	SCA10	22.10
X	24531771	25531771	EIEE1	11.36
X	66265159	67265159	SBMA	10.70
X	**1,46E+08**	**1,47E+08**	**FRAXA**	**10.77**
X	1,47E+08	1,48E+08	FRAXE	11.37

Appendix 5: In-House Illumina Haplotype Phasing Part 1 Code

The Shell file:

```
1. ### Illumina-Guided HP
2. cd /work/sdularsen/tine/nanoporeData/
3.
4. Window=40000
5. AF_low=30
6. AF_high=70
7. DP=10
8.
9. # Go through the listed genes
10.  for gene in "FMR1" "HTT"; do
11.          if [ $gene == "FMR1" ]; then
```

```
12.                 # FMR1
13.                 chromosome=X
14.                 Start_Position=$(( 146993569 - $Window ))
15.                 End_Position=$(( 146993598  + $Window ))
16.         elif [ $gene == "HTT" ]; then
17.                 # HTT
18.                 chromosome=4
19.                 Start_Position=$(( 3076603 - $Window ))
20.                 End_Position=$(( 3076774 + $Window ))
21.         else
22.                 echo "Gene $gene not supported"
23.                 break
24.         fi
25.
26.         for file in Illumina_data/*vcf.gz; do
27.                 vcf=`basename $file`
28.                 SAMPLE=${vcf%_truseq-pcr-free-genome_*.vcf.gz}
29.                 echo $SAMPLE $gene
30.
31.                 ### SAMtools
32.                 # Extracting a smaller BAM
33.                 samtools view -bh combined_data/new_combined_bam/${SAM-
   PLE}_combined.bam "$chromosome:${Start_Position}-${End_Position}" > com-
   bined_data/illumina_hp/ONT_BAM_subset/${SAMPLE}_$(Flanagan et
   al.)_window${Window}_AF_0_${AF_low}_to_0_${AF_high}_DP${DP}.bam
34.                 samtools index combined_data/illumina_hp/ONT_BAM_sub-
   set/${SAMPLE}_$(Flanagan et
   al.)_window${Window}_AF_0_${AF_low}_to_0_${AF_high}_DP${DP}.bam
35.
36.                 # Export the readnames into a txt file
37.                 samtools view combined_data/illumina_hp/ONT_BAM_sub-
   set/${SAMPLE}_$(Flanagan et
   al.)_window${Window}_AF_0_${AF_low}_to_0_${AF_high}_DP${DP}.bam | cut -f
   1,3,4 > combined_data/illumina_hp/ONT_BAM_subset_readnames/${SAM-
   PLE}_$(Flanagan et
   al.)_Read_Name_window${Window}_AF_0_${AF_low}_to_0_${AF_high}_DP${DP}.txt
38.
39.
40.
41.                 ### BCFtools
42.                 # Creation of an index file
43.                 INDEX_FILE=Illumina_data/${vcf}.tbi
44.                 if [ ! -f "$INDEX_FILE" ]; then
45.                         tabix -p vcf Illumina_data/${vcf}
46.                 fi
47.
48.                 # Subset Illumina Dragen VCF file to ROI +/- the window
49.                 bcftools filter -r $chromosome:${Start_Position}-
   ${End_Position} Illumina_data/${vcf} > combined_data/illumina_hp/illu-
   mina_VCF_subset/${SAMPLE}_$(Flanagan et
   al.)_Illumina_Dragen_window${Window}_AF_0_${AF_low}_to_0_${AF_high}_DP${DP
   }.vcf
50.
51.                 # Filter VCF file
52.                 bcftools filter -i "FORMAT/DP>=${DP} && FOR-
   MAT/AF>=0.${AF_low} && FORMAT/AF<=0.${AF_high} && STRLEN(REF) == 1 &&
```

```
   STRLEN(ALT) == 1 && N_ALT==1" combined_data/illumina_hp/illumina_VCF_sub-
   set/${SAMPLE}_$(Flanagan et
   al.)_Illumina_Dragen_window${Window}_AF_0_${AF_low}_to_0_${AF_high}_DP${DP
   }.vcf \
53.                 > combined_data/illumina_hp/illumina_VCF_subset_fil-
   tered/${SAMPLE}_$(Flanagan et
   al.)_Illumina_Dragen_Filtered_SNVs_window${Window}_AF_0_${AF_low}_to_0_${A
   F_high}_DP${DP}.vcf
54.
55.                 #Extract Illumina Genotypes from the filtered VCF file
56.                 grep -v '^##' combined_data/illumina_hp/illumina_VCF_sub-
   set_filtered/${SAMPLE}_$(Flanagan et
   al.)_Illumina_Dragen_Filtered_SNVs_window${Window}_AF_0_${AF_low}_to_0_${A
   F_high}_DP${DP}.vcf | cut -f 1,2,4,5 | sed 's/#//g' > \
57.                 combined_data/illumina_hp/illumina_VCF_genotypes/${SAM-
   PLE}_$(Flanagan et
   al.)_Illumina_Dragen_genotypes_window${Window}_AF_0_${AF_low}_to_0_${AF_hi
   gh}_DP${DP}.txt
58.
59.                 # Number of lines/SNPs left after filtering
60.                 snvs=$(bcftools filter -i "FORMAT/DP>=${DP} && FOR-
   MAT/AF>=0.${AF_low} && FORMAT/AF<=0.${AF_high} && STRLEN(REF) == 1 &&
   STRLEN(ALT) == 1 && N_ALT==1" combined_data/illumina_hp/illumina_VCF_sub-
   set/${SAMPLE}_$(Flanagan et
   al.)_Illumina_Dragen_window${Window}_AF_0_${AF_low}_to_0_${AF_high}_DP${DP
   }.vcf | grep -v '^#' | wc -l)
61.                 echo $snvs
62.
63.
64.                 ### SNV Bases
65.                 if [ $snvs -gt 0 ]; then
66.                         python /work/sdularsen/tine/nanoporeData/komman-
   doer/SNV_Base_Identification_combined_data.py \
67.                                 --sample $SAMPLE --window $Window --gene
   $gene \
68.                                 --chr $chromosome --start $Start_Position
   --end $End_Position \
69.                                 --AF_low $AF_low --AF_high $AF_high --DP
   $DP
70.                 fi
71.         done
72.  done
```

The Python file:

```
1. # Import libraries
2. import pysam
3. import pandas as pd
4. import sys
5. import argparse
6. path = "/work/sdularsen/tine/nanoporeData/"
7.
8. # Add variables used in the code
9. parser = argparse.ArgumentParser(description='Process some integers.')
10.  parser.add_argument('--sample', type = str, help = 'The sample to ana-
   lyze')
```

```
parser.add_argument('--window', type = str, help = 'The window around the repeat')
parser.add_argument('--gene', type = str, help = 'The gene of interest')
parser.add_argument('--chr', type = str, help = 'The chromosome where the gene of interest is located')
parser.add_argument('--start', type = int, help = 'Start position on chromosome')
parser.add_argument('--end', type = int, help = 'End position on chromosome')
parser.add_argument('--AF_low', type = str, help = 'low end of AF interval')
parser.add_argument('--AF_high', type = str, help = 'high end of AF interval')
parser.add_argument('--DP', type = str, help = 'read depth from Illumina Data')
args = parser.parse_args()
sample = args.sample
gene = args.gene
window = args.window
chr = args.chr
start = args.start
end = args.end
AF_low = args.AF_low
AF_high = args.AF_high
DP = args.DP

# Load SNV positions into a set to be used in the data frame
snv_positions = set()
with open (path + "combined_data/illumina_hp/illumina_VCF_subset_filtered/"+sample+"_"+gene+"_Illumina_Dragen_Filtered_SNVs_window"+window+"_AF_0_"+AF_low+"_to_0_"+AF_high+"_DP"+DP+".vcf", "r") as f:
    for line in f:
        if not line.startswith("#"):
            snv_positions.add(int(line.split("\t")[1]))

# Load read names into a list to be used in the data frame
read_names = list()
with open (path + "combined_data/illumina_hp/ONT_BAM_subset_readnames/"+sample+"_"+gene+"_Read_Name_window"+window+"_AF_0_"+AF_low+"_to_0_"+AF_high+"_DP"+DP+".txt", "r") as f:
    for line in f:
        read_names.append(line.split("\t")[0])

# Create DataFrame object
snv_df = pd.DataFrame(pd.NA, columns = sorted(list(snv_positions)), index=read_names)

# Fill the dataframe by going through the SNV's and reads
bamfile = pysam.AlignmentFile(path + "combined_data/illumina_hp/ONT_BAM_subset/"+sample+"_"+gene+"_window"+window+"_AF_0_"+AF_low+"_to_0_"+AF_high+"_DP"+DP+".bam", "rb")
for pileupcolumn in bamfile.pileup(chr, start, end, min_base_quality=0):
    # python is 0-indexed, vcf-file is 1-indexed
    snv_pos = pileupcolumn.pos+1
    if snv_pos in snv_positions:
        for pileupread in pileupcolumn.pileups:
            read_name = pileupread.alignment.query_name
            ## checking for deletions
            if not pileupread.is_refskip:
```

```
                    if pileupread.is_del:
                        snv_df.loc[read_name, snv_pos] = "-"
                    else:
                        snv_df.loc[read_name, snv_pos] = pileupread.align-
    ment.query_sequence[pileupread.query_position]

    bamfile.close()

    # Remove columns with only NAs
    snv_df = snv_df.dropna(axis=1,how="all")
    # Remove rows with only NAs
    snv_df = snv_df.dropna(axis=0,how="all")


    # Print data frame while keeping the NAs
    print(snv_df, sep='\n')

    # Save the data frame as a txt file
    snv_df.to_csv(path + "combined_data/illumina_hp/results_data-
    frames/"+sample+"_"+gene+"_bases_window"+win-
    dow+"_AF_0_"+AF_low+"_to_0_"+AF_high+"_DP"+DP+".txt", sep=',',
    mode='w',na_rep='NA')
```

Appendix 6: In-House Illumina Haplotype Phasing Part 2 Code

The R file:

```
library(dplyr)
library(proxy)
library(gower)

# Set working directory
setwd("C:/Users/Taine/OneDrive/Skrivebord/Nanopore/combined_data/illu-
mina_hp/dataframes_and_genotypes")



#########################
### Helper functions ###
#########################

get_gene_information <- function(gene = "HTT"){
  # HTT gene information
  if (gene == "HTT"){
    chr <<- 4
    start_pos <<- 3076604
    end_pos <<- 3076666
  } else if (gene == "FMR1"){
    # FMR1 gene information
    chr <<- "X"
```

```
23.      start_pos <<- 146993569
24.      end_pos <<- 146993628
25.    } else {
26.      print("gene not supported")
27.    }
28.  }
29.
30.  # Remove snps not in close proximity to gene
31.  remove_distant_reads <- function(data, snp_left, snp_right){
32.
33.    # Get reads with information on nearest snp on either side of gene
34.    if (nrow(snp_left)>0){
35.      reads_left = !is.na(data[,which(names(data)==snp_left$snp)])
36.    } else {
37.      reads_left = F
38.    }
39.    if (nrow(snp_right)>0){
40.      reads_right = !is.na(data[,which(names(data)==snp_right$snp)])
41.    } else {
42.      reads_right = F
43.    }
44.
45.    # Keep only reads near or spanning gene
46.    data = data[reads_left|reads_right,]
47.    return(data)
48.  }
49.
50.  # Keep only reads covering at least two snps
51.  remove_uninformative_reads <- function(dat,min_snps){
52.    rows_to_keep=apply(dat, 1, function(row){
53.      # Check if at least "min_snps" SNPs exists in a read
54.      # note: snp_name is not NA and therefore counts +1 extra
55.      if (sum(!is.na(row))>=(min_snps+1)){
56.        return(TRUE)
57.      } else {
58.        return(FALSE)
59.      }
60.    })
61.    rows_to_keep
62.  }
63.
64.  # Check reads cover both snps
65.  num_reads_cover_both_snps <- function(data2, snp_left, snp_right){
66.
67.    # Get reads with information on nearest snp on either side of gene
68.    if (nrow(snp_left)>0 && nrow(snp_right)>0){
69.      return(sum(!is.na(data2[,snp_left$snp]) &
   !is.na(data2[,snp_right$snp])))
70.    } else if (nrow(snp_left)>0){
71.      return(sum(!is.na(data2[,snp_left$snp])))
72.    } else if (nrow(snp_right)>0){
73.      return(sum(!is.na(data2[,snp_right$snp])))
74.    } else {
75.      return(0)
76.    }
77.
78.  }
79.
80.  # Compute gower's distance
```

```
gower_distance <- function(x,y){
  # Check if no overlapping snps between two reads
  # if no overlapping snps, return 0.5 as their distance
  if (sum(!(is.na(x) | is.na(y)))==0){
    return(0.5)
  }
  # Compute and return gower's distance between two reads
  return(gower_dist(x,y))
}

# Hierarchical clustering of reads
hierarchical_clustering <- function(data, k=2){
  ## Hierarchical clustering
  dst = proxy::dist(data[,-1],method = gower_distance)
  hc <- hclust(dst, method = "ward.D2")
  cluster = cutree(hc, k=k)
  dend = as.dendrogram(hc)
  plot(dend)
  return(cluster)
}

# Construct haplotype from subset of reads
construct_haplotype <- function(h){
  apply(h[,-1], 2, function(col){
    d = sort(table(col), decreasing = T)
    # print(d)
    if (length(d)==0){
      return(NA)
    } else if (length(d)==1 || d[1]>d[2]){
      return(names(d[1]))
    } else if (d[1]==d[2]){
      return(paste0(names(d)[1], "/", names(d)[2]))
    }
  })
}

# Print some stats of the data
get_statistics <- function(data){
  # Count num snps for each read
  snps_per_read = apply(data,1,function(row){
    sum(!is.na(row))
  })
  print("Number of SNPs covered in each read")
  ## Print only snps_per_read count
  print(snps_per_read)
  ## Print read_id and snps_per_read count
  print(data.frame(Read_ID=data$X, snps_per_read))

  # Count num reads covering every set of two snps
  # - if any is 0, then the reads must be split into separated groups
  reads_covering_set_two_snps <- sapply(2:ncol(data)-1, function(i){
    sum(!is.na(data[,i]) & !is.na(data[,i+1]))
  })
  print("Number of reads covering every 2-set of snps")
  print(reads_covering_set_two_snps)
}

# Check if snp is found in haplotype
check_snp_in_hap <- function(read, hap, pos){
```

```
140.   if (!is.na(read[pos]) && read[pos] %in% strsplit(hap[pos], split =
   "/")[[1]]){
141.     return(TRUE)
142.   } else {
143.     return(FALSE)
144.   }
145. }
146.
147. ##########
148. ## Main ##
149. ##########
150.
151. input_files = list.files(pattern = "_bases_win-
   dow40000_AF_0_30_to_0_70_DP10.txt")
152. for (file in input_files){
153.
154.   # Define sample and gene from input file
155.   sample = strsplit(file, split = "_")[[1]][1]
156.   gene = strsplit(file, split = "_")[[1]][2]
157.
158.   # read input
159.   data = read.table(file, header = T, sep = ",", na.strings = "NA")
160.
161.   # Remove duplicate lines (that exist for some reason)
162.   data = distinct(data)
163.   # Get dimensionality of data
164.   dim(data)
165.
166.   get_gene_information(gene)
167.
168.   # Guess the most probable genotype at each snp
169.   bases = apply(data[,-1], 2, function(col){
170.     d = sort(table(col), decreasing = T)
171.     c(names(d)[1], names(d)[2])
172.   })
173.
174.   # Get some stats on the data
175.   get_statistics(data)
176.
177.   # Extract snp coordinates from SNP names
178.   snp_coords = data.frame(snp=names(data[,-1]), coord = as.inte-
   ger(gsub(x = names(data[,-1]), pattern = "X", replacement = "")))
179.   # Get nearest snp to gene on either side
180.   snp_left = filter(snp_coords, coord <= start_pos) %>% slice_max(coord)
181.   snp_right = filter(snp_coords, coord >= end_pos) %>% slice_min(coord)
182.   # Remove reads not covering at least one of the closest snp on either
   side of the gene
183.   data = remove_distant_reads(data, snp_left, snp_right)
184.
185.   # read genotypes
186.   genotypes = read.table(gsub("bases", "Illumina_Dragen_genotypes",
   file), header = T)
187.   for(i in 1:nrow(genotypes)){
188.     genotypes_in_snp = c(genotypes$REF[i],genotypes$ALT[i])
189.     data[,i+1][!(data[,i+1] %in% genotypes_in_snp)] = NA
190.   }
191.
192.   # Get some stats on the data
193.   get_statistics(data)
```

```
194.
195.   # Remove reads covering only one SNP
196.   rows_to_keep = remove_uninformative_reads(data, 2)
197.   data2 = data[rows_to_keep,]
198.
199.   # Check if any reads exist covering both SNPs closest on either side
   to the gene, or only SNPs on one side exists
200.   reads_covering_both_snps = num_reads_cover_both_snps(data2, snp_left,
   snp_right)
201.   if (reads_covering_both_snps>0){
202.
203.     # Cluster reads into two groups
204.     cluster = hierarchical_clustering(data2, 2)
205.
206.     # Construct most likely haplotypes - possibly use logos plots
207.     hap1 = construct_haplotype(data2[cluster==1,])
208.     hap2 = construct_haplotype(data2[cluster==2,])
209.
210.   } else {
211.
212.     # Split data into two sets containing reads on either side of the
   gene
213.     data_left = data2[!is.na(data2[,snp_left$snp]),]
214.     data_right = data2[!is.na(data2[,snp_right$snp]),]
215.
216.     # Cluster reads in each dataset into two groups
217.     cluster1 = hierarchical_clustering(data_left, 2)
218.     cluster2 = hierarchical_clustering(data_right, 2)
219.
220.     # Construct haplotypes for cluster1
221.     hap1_left = construct_haplotype(data_left[cluster1==1,])
222.     hap2_left = construct_haplotype(data_left[cluster1==2,])
223.     # Construct haplotypes for cluster2
224.     hap1_right = construct_haplotype(data_right[cluster2==1,])
225.     hap2_right = construct_haplotype(data_right[cluster2==2,])
226.   }
227.
228.   # Recover and cluster "uninformative" reads i.e. reads covering only
   one snp
229.   if (sum(!rows_to_keep)>0){
230.     uninformative_reads = data[!rows_to_keep,]
231.     uninformative_reads$haplotype = NA
232.
233.     if (reads_covering_both_snps>0){
234.       for (i in 1:nrow(uninformative_reads)){
235.         # Get position of snp not being NA
236.         pos = which(!is.na(uninformative_reads[i,-1]))
237.         # If no snp found ignore read
238.         if (length(pos)==0){
239.           next
240.         }
241.
242.         # Check if snp matches both haplotypes, if so we cannot assign
   it to any cluster and we ignore it
243.         if (check_snp_in_hap(uninformative_reads[i,-1], hap1, pos) &&
   check_snp_in_hap(uninformative_reads[i,-1], hap2, pos)){
244.           next
245.           # Check if snp matches hap1, if so assign it to cluster1
```

```
246.         } else if (check_snp_in_hap(uninformative_reads[i,-1], hap1,
   pos)) {
247.           uninformative_reads[i,"haplotype"] = "hap1"
248.           # Check if snp matches hap2, if so assign it to cluster1
249.         } else if (check_snp_in_hap(uninformative_reads[i,-1], hap2,
   pos)) {
250.           uninformative_reads[i,"haplotype"] = "hap2"
251.           # If no matches we ignore the read
252.         } else {
253.           next
254.         }
255.       }
256.     } else {
257.       for (i in 1:nrow(uninformative_reads)){
258.         # Get position of snp not being NA
259.         pos = which(!is.na(uninformative_reads[i,-1]))
260.         # If no snp found ignore read
261.         if (length(pos)==0){
262.           next
263.         }
264.
265.         # Check if snp matches both haplotypes, if so we cannot assign
   it to any cluster and we ignore it
266.         if (check_snp_in_hap(uninformative_reads[i,-1], hap1_left, pos)
   && check_snp_in_hap(uninformative_reads[i,-1], hap2_left, pos)){
267.           next
268.         }
269.         if (check_snp_in_hap(uninformative_reads[i,-1], hap1_right, pos)
   && check_snp_in_hap(uninformative_reads[i,-1], hap2_right, pos)){
270.           next
271.         }
272.
273.         # Check if SNP matches hap1_left, if so assign it to hap1_left
274.         if (check_snp_in_hap(uninformative_reads[i,-1], hap1_left, pos))
   {
275.           uninformative_reads[i,"haplotype"] = "hap1_left"
276.           # Check if SNP matches hap2_left, if so assign it to hap2_left
277.         } else if (check_snp_in_hap(uninformative_reads[i,-1],
   hap2_left, pos)) {
278.           uninformative_reads[i,"haplotype"] = "hap2_left"
279.           # Check if SNP matches hap1_right, if so assign it to
   hap1_right
280.         } else if (check_snp_in_hap(uninformative_reads[i,-1],
   hap1_right, pos)) {
281.           uninformative_reads[i,"haplotype"] = "hap1_right"
282.           # Check if SNP matches hap2_right, if so assign it to
   hap2_right
283.         } else if (check_snp_in_hap(uninformative_reads[i,-1],
   hap2_right, pos)) {
284.           uninformative_reads[i,"haplotype"] = "hap1_right"
285.         }
286.       }
287.     }
288.   }
289.
290.   # Write output files
291.   output_dir = "../Output_haplotype_phasing/heterozygote/"
292.   dir.create(output_dir, showWarnings = F)
293.
```

```
294.   output_files_for_haplotype <- function(data){
295.
296.     # Add origin column
297.     data$origin = data$haplotype
298.     uninformative_reads$origin = paste0(uninformative_reads$haplo-
   type,"_recovered")
299.     uninformative_reads = select(uninformative_reads, read_ids=X, haplo-
   type, origin)
300.     uninformative_reads = filter(uninformative_reads, !is.na(haplotype))
301.
302.     data = rbind(data,uninformative_reads)
303.
304.     # Output reads related to haplotype 1
305.     write.table(data, file = paste0(output_dir, sam-
   ple,"_",gene,"_reads.txt"), quote = F, row.names = F)
306.   }
307.
308.   if (reads_covering_both_snps>0){
309.     # Output haplotypes
310.     hap1_out = paste(hap1, collapse = " ")
311.     hap2_out = paste(hap2, collapse = " ")
312.     write.table(paste0("Haplotype 1:\n",hap1_out,"\n\nHaplotype 2:\n",
   hap2_out),
313.                 file = paste0(output_dir, sample,"_",gene,"_haplo-
   types.txt"), quote = F, row.names = F, col.names = F)
314.
315.     # Add haplotype information
316.     data2$haplotype = paste0("hap",cluster)
317.     data2 = select(data2, read_ids=X, haplotype)
318.     # Write reads to files
319.     output_files_for_haplotype(data2)
320.
321.   } else {
322.     # Output haplotypes
323.     hap1_left_out = paste(hap1_left, collapse = " ")
324.     hap2_left_out = paste(hap2_left, collapse = " ")
325.     hap1_right_out = paste(hap1_right, collapse = " ")
326.     hap2_right_out = paste(hap2_right, collapse = " ")
327.     write.table(paste0("hap1_left:\n",hap1_left_out,"\n\nhap2_left:\n",
   hap2_left_out, "\n\nhap1_right:\n",hap1_right_out,"\n\nhap2_right:\n",
   hap2_right_out),
328.                 file = paste0(output_dir, sample,"_",gene,"_haplo-
   types.txt"), quote = F, row.names = F, col.names = F)
329.
330.     # Write reads to files
331.     data2 = data.frame(read_ids=c(data_left$X, data_right$X),haplo-
   type=c(paste0("hap",cluster1,"_left"),paste0("hap",cluster2,"_right")))
332.
333.     # Write reads to files
334.     output_files_for_haplotype(data2)
335.   }
336. }
```

Appendix 7: Logos Plot Haplotype Comparisons Showing of the Whole-Genome Data

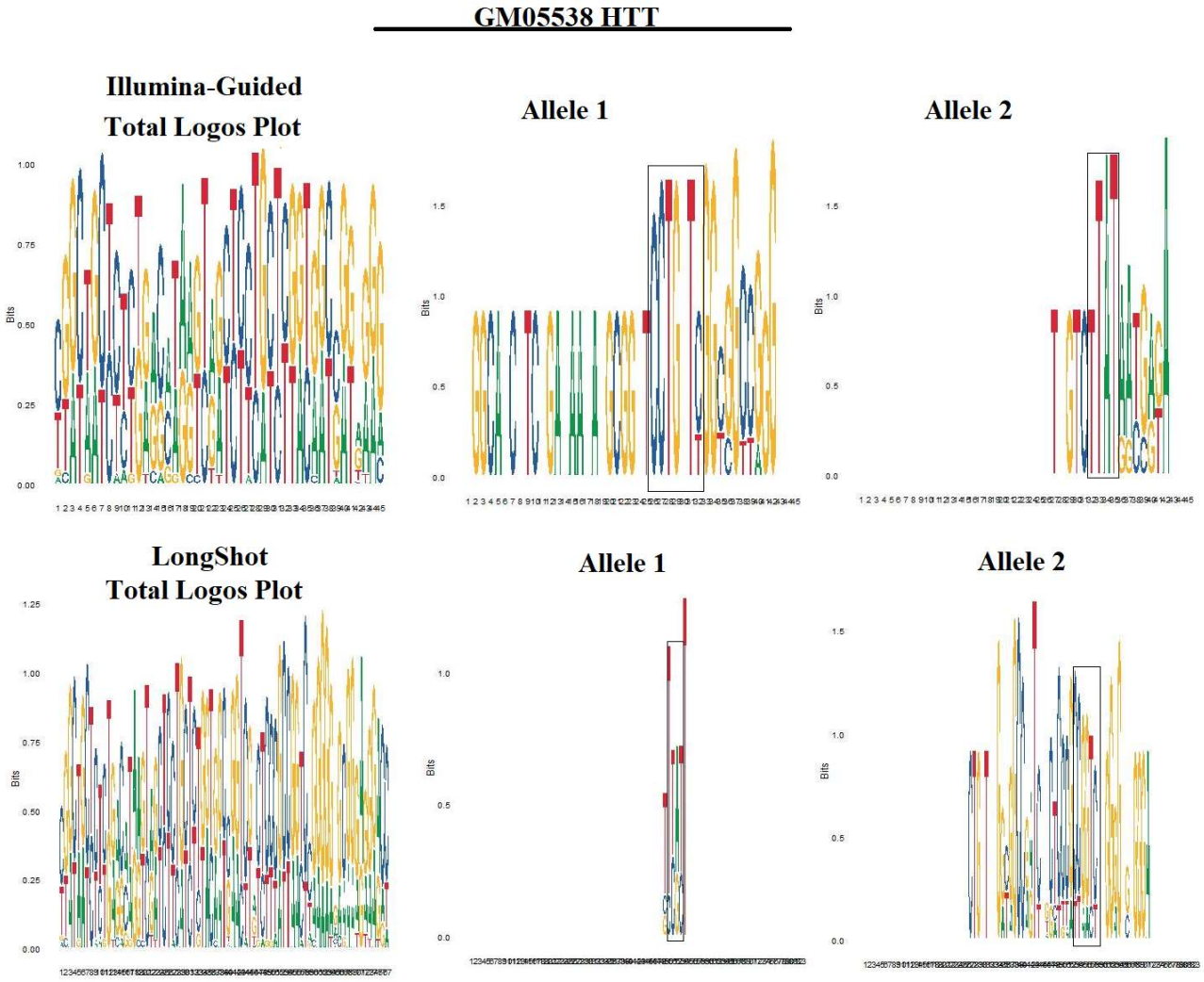


Figure 35: Logos Plot Comparison for Sample GM05538 at HTT.

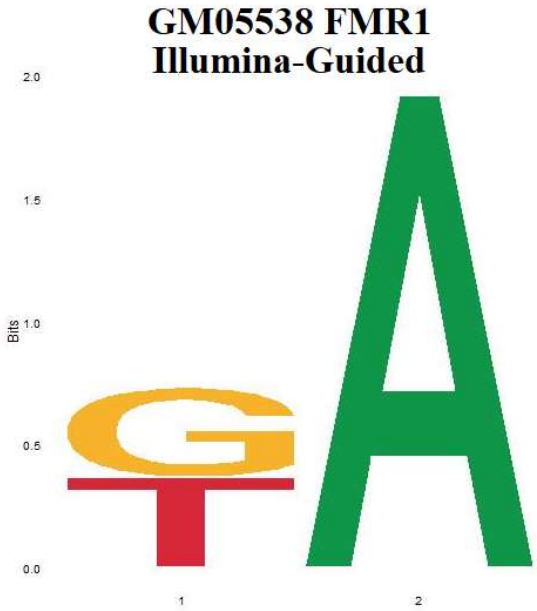


Figure 36: Illumina-Guided Logos Plot for GM05538 at FMR1.

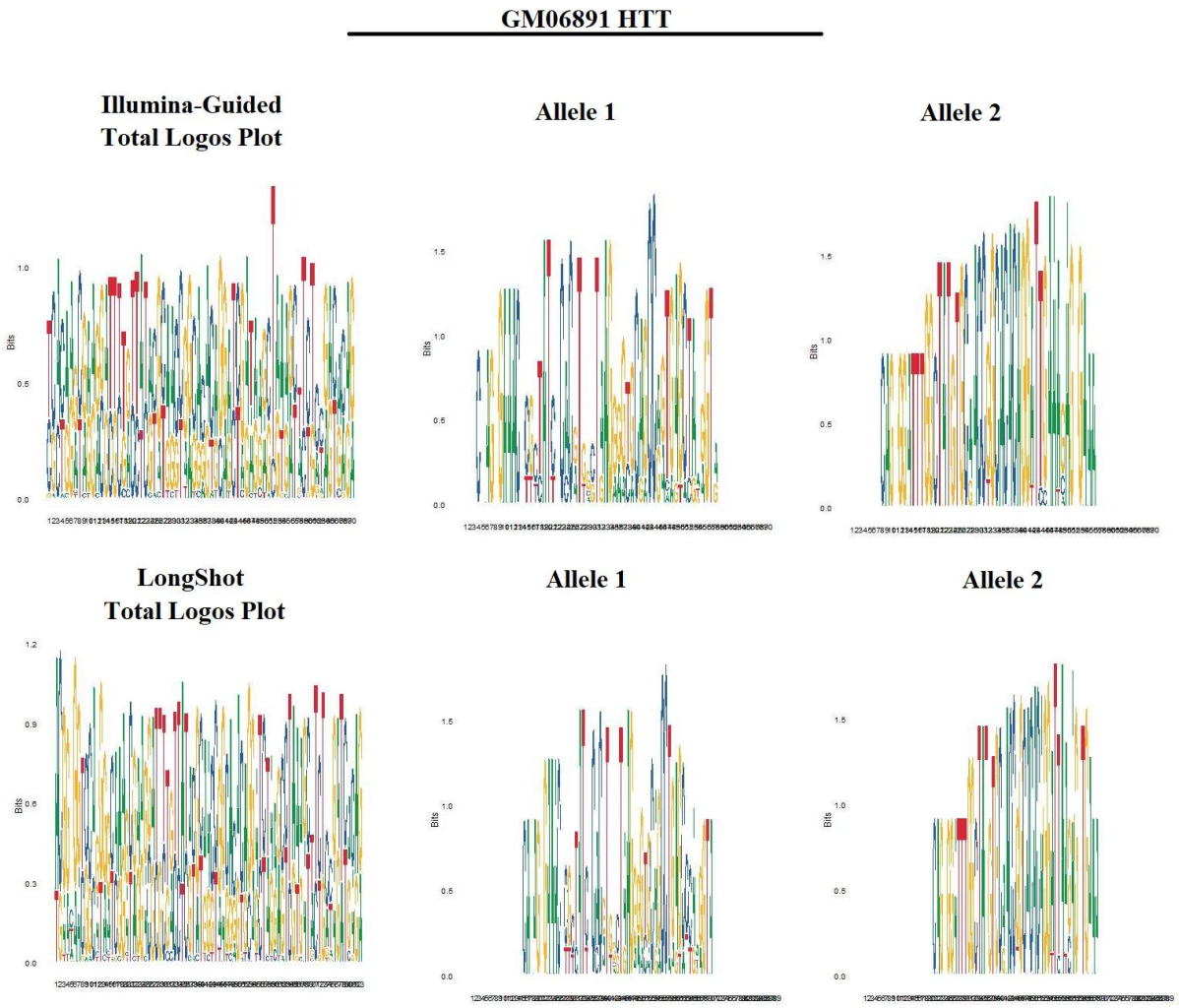


Figure 37: Logos Plot Comparison for Sample GM06891 at HTT.

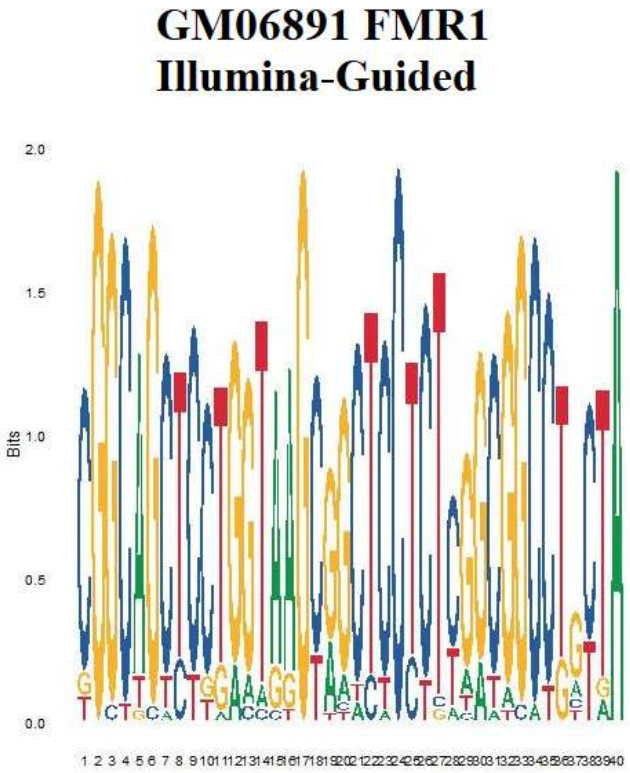


Figure 38: Illumina-Guided Logos Plot for GM06891 at FMR1.

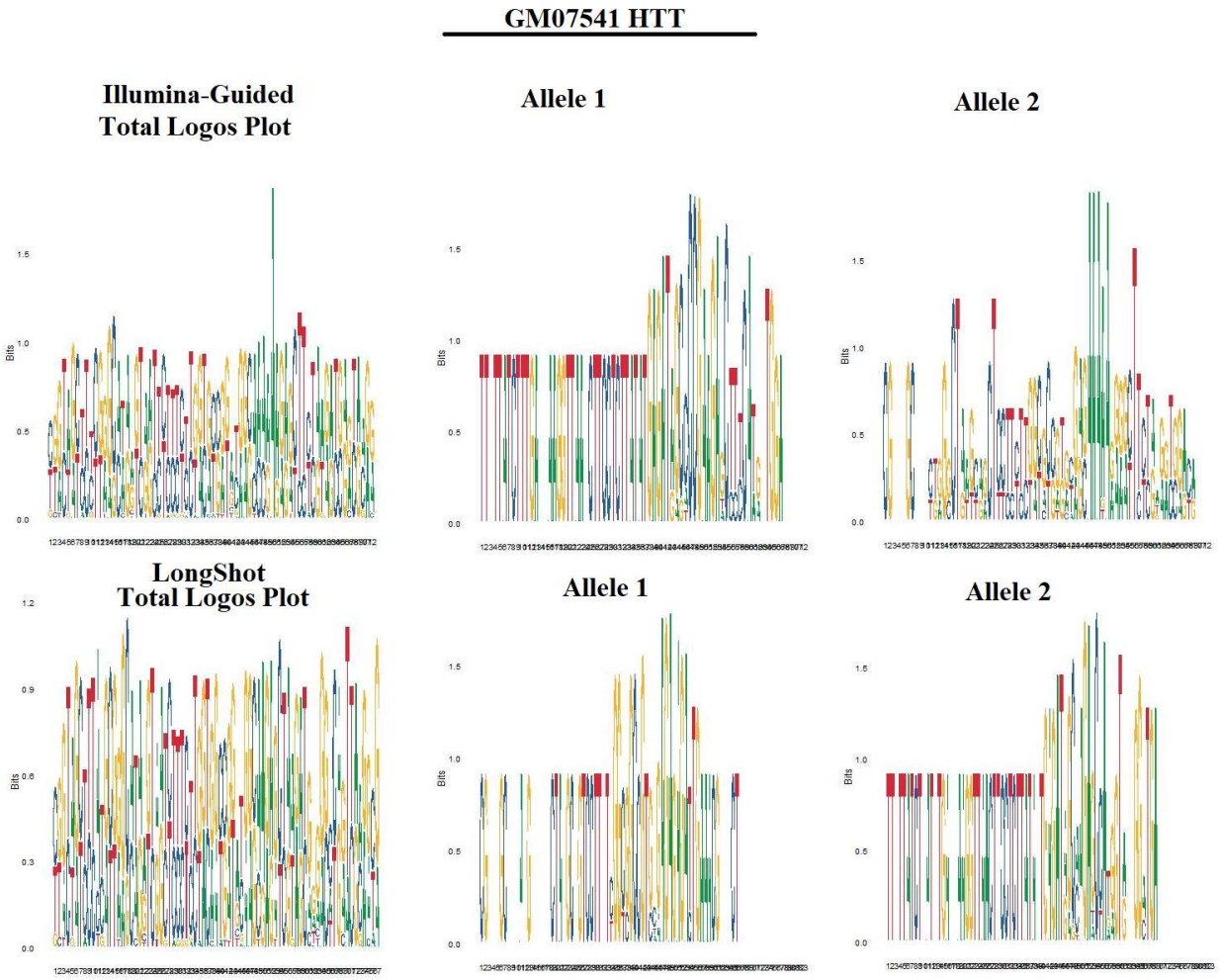


Figure 39: Logos Plot Comparison for Sample GM07541 at HTT.

GM07541 FMR1

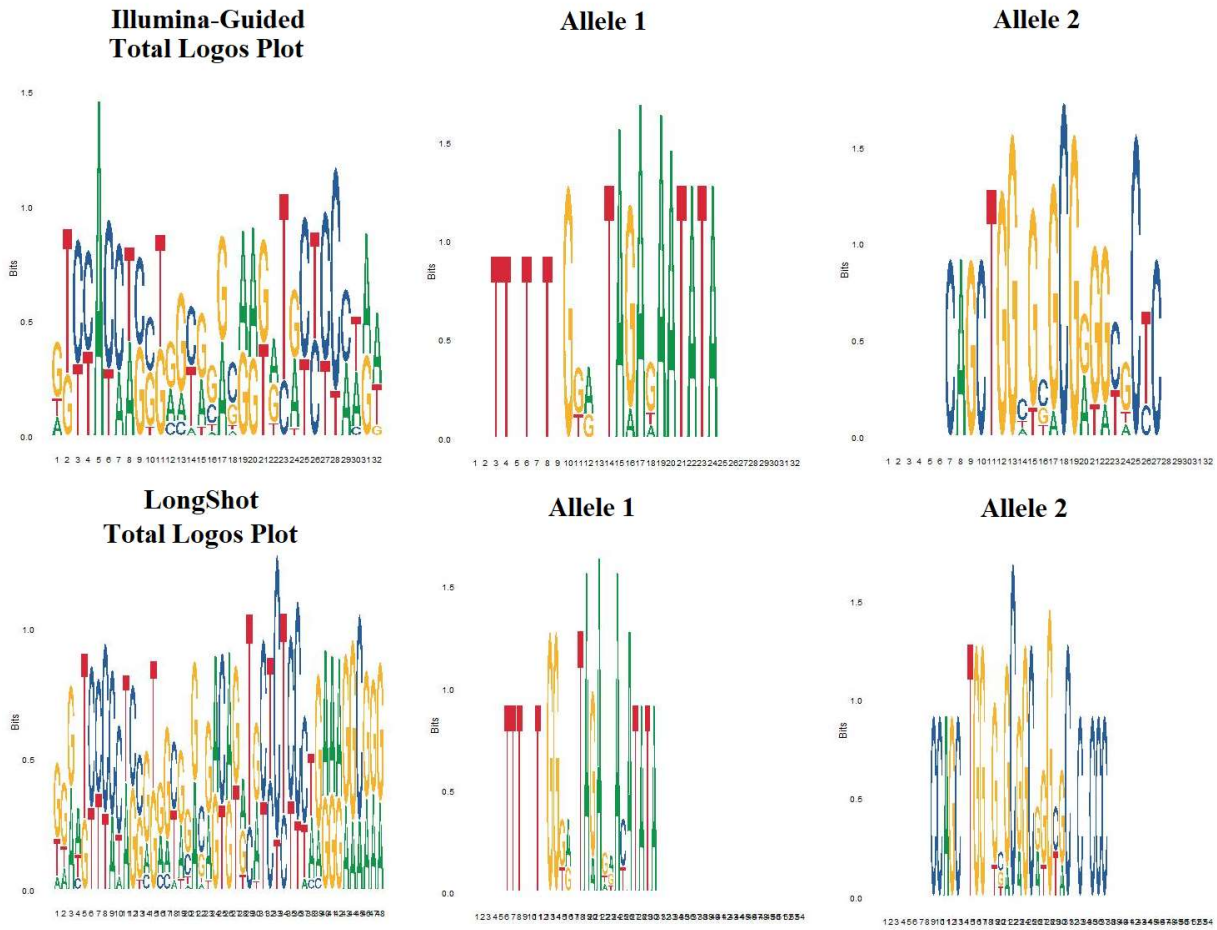


Figure 40: Logos Plot Comparison for Sample GM07541 at FMR1.

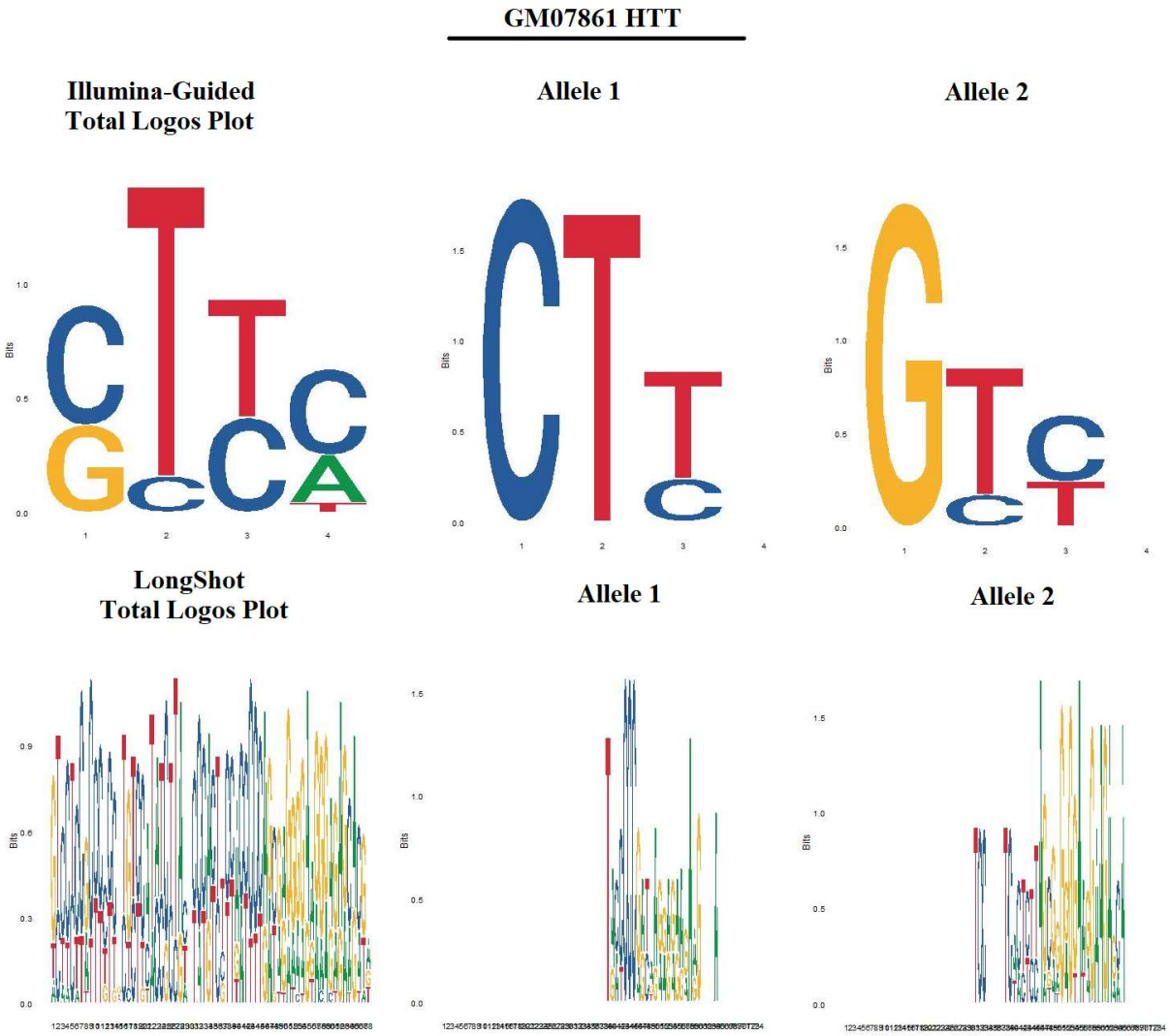


Figure 41: Logos Plot Comparison for Sample GM07861 at HTT.

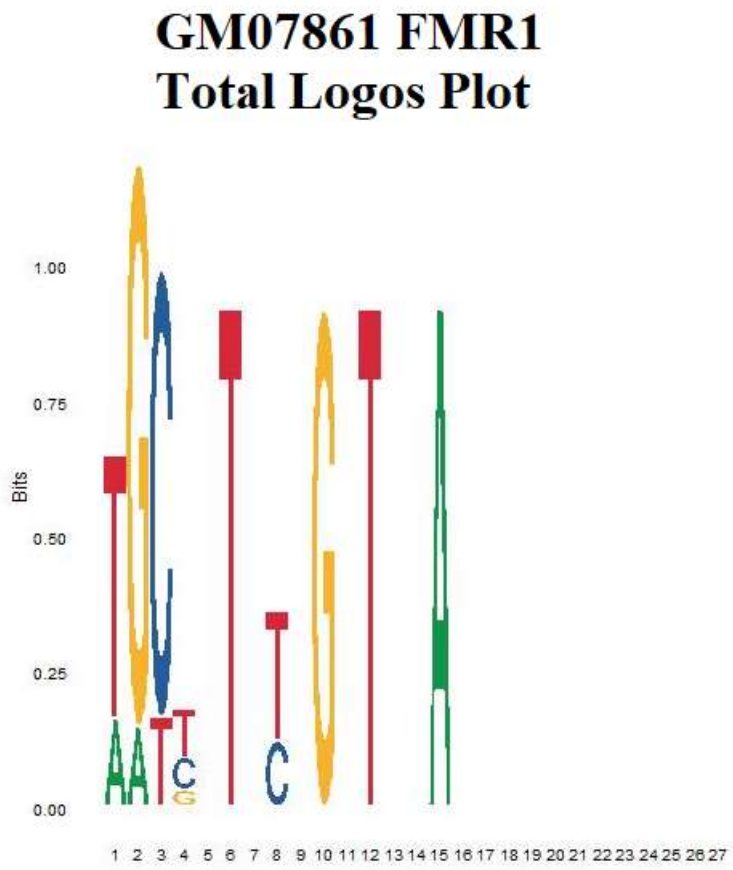


Figure 42:Illumina-Guided Logos Plot for GM07861 at FMR1.

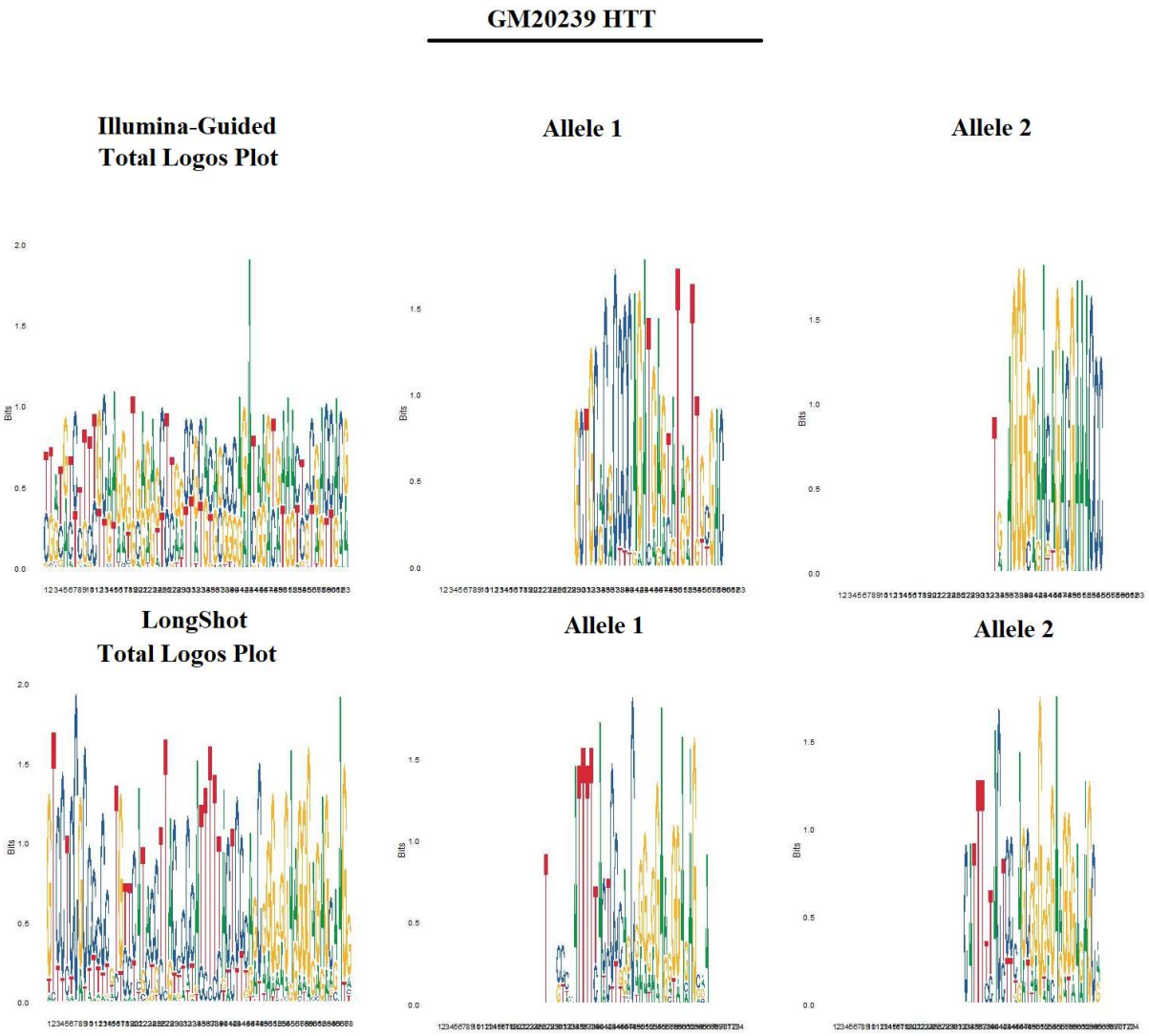


Figure 43:Logos Plot Comparison for Sample GM20239 at HTT.

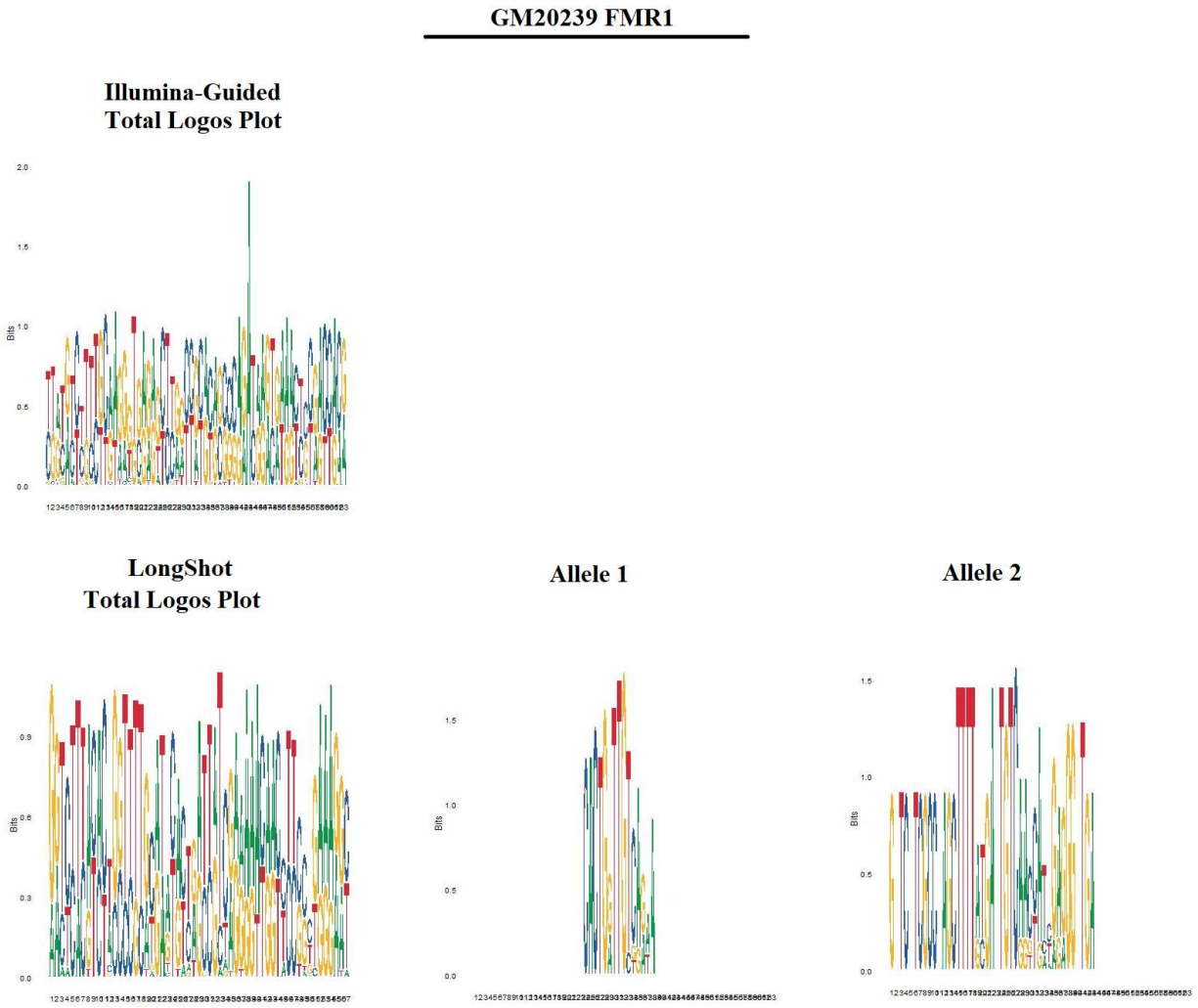
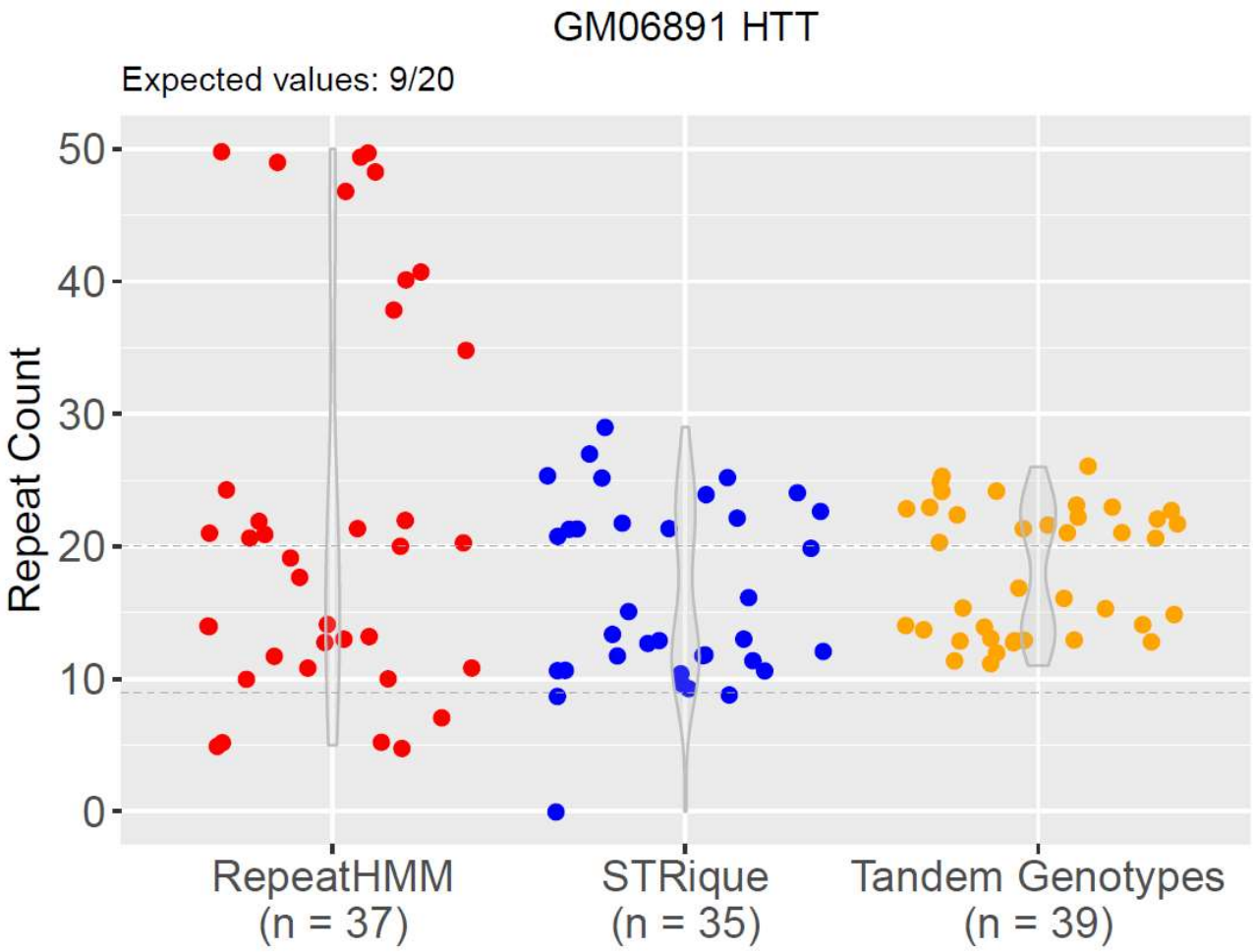
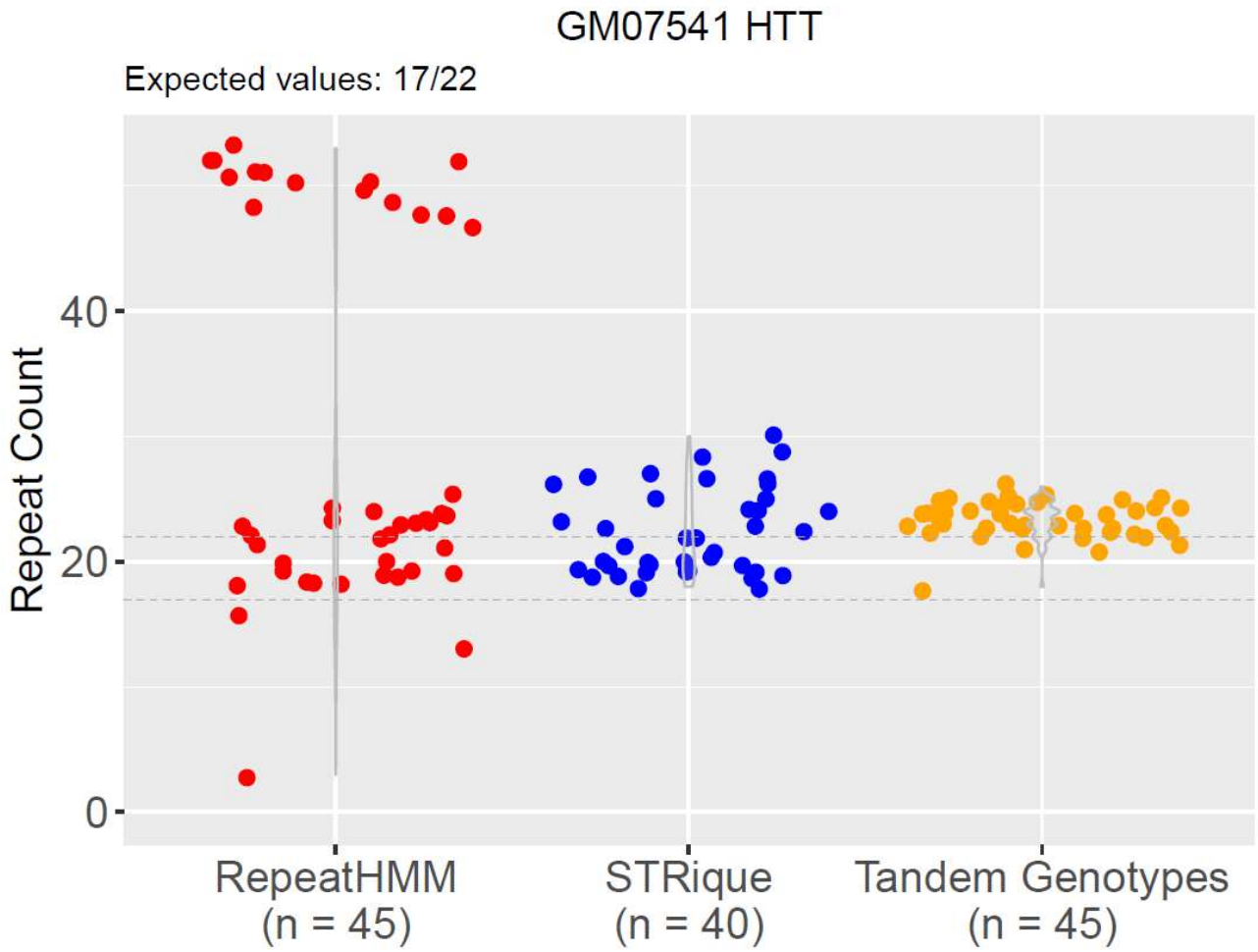
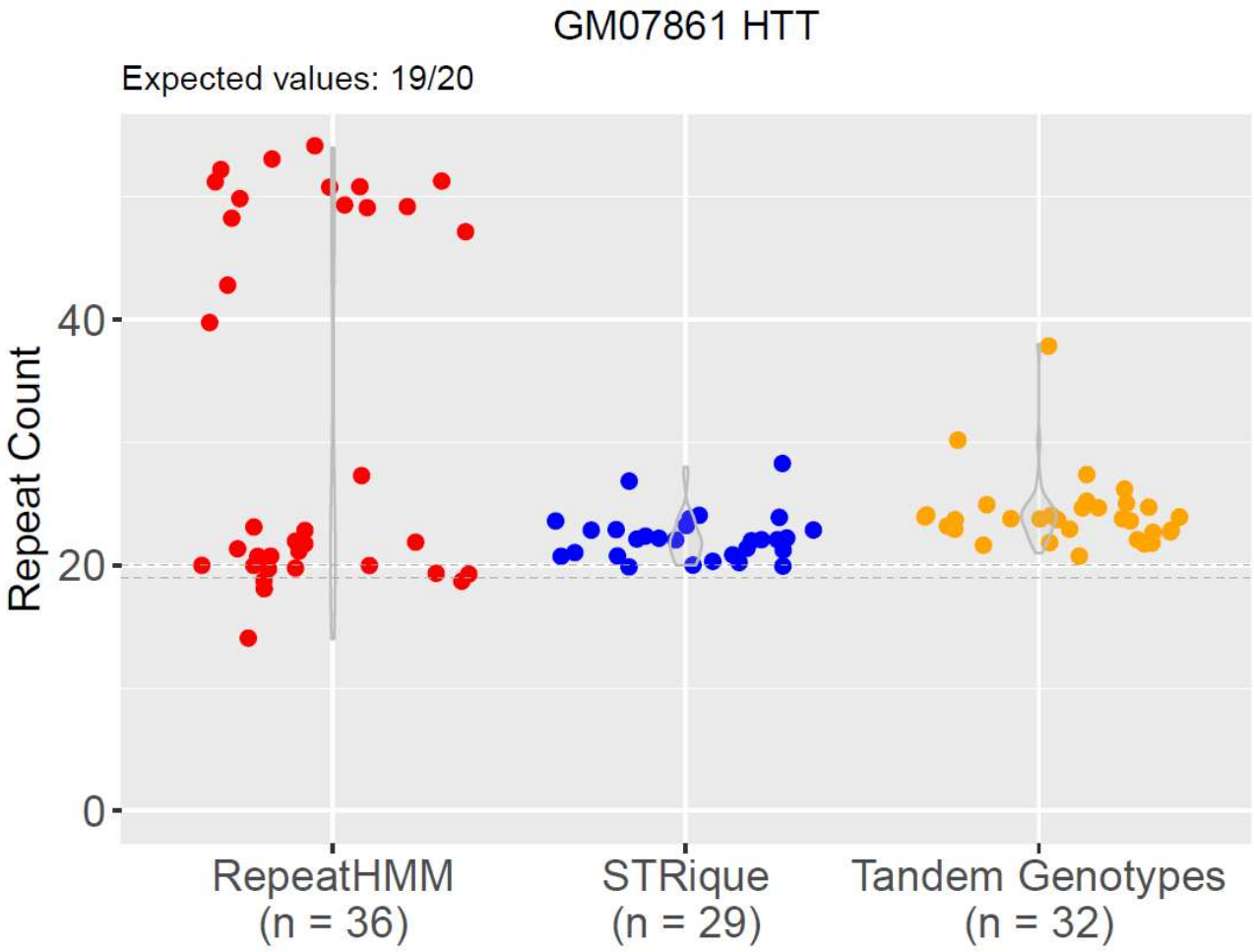


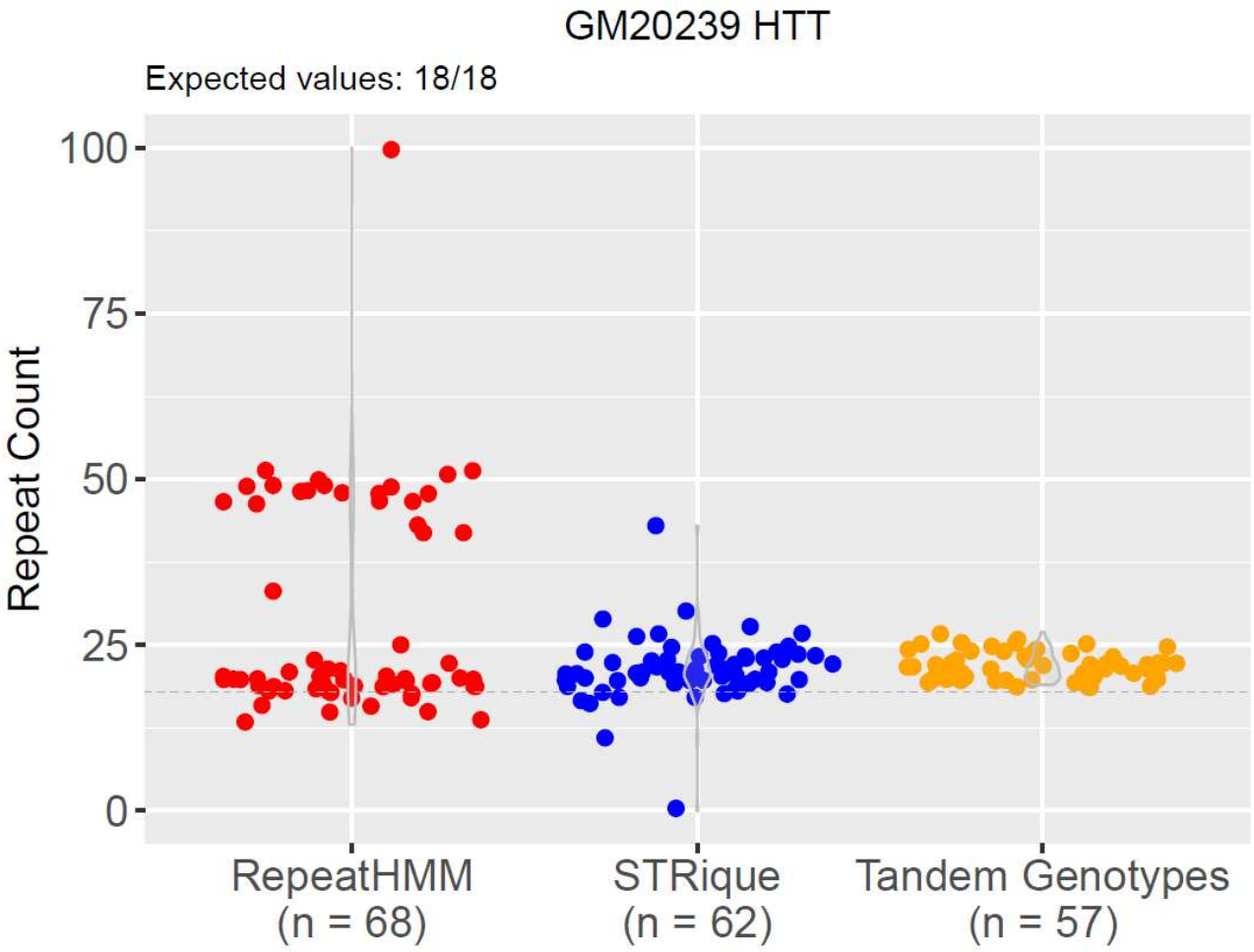
Figure 44: Logos Plot Comparison for Sample GM20239 at FMR1.

Appendix 8: Unphased Combined Repeat Count Figures for HTT

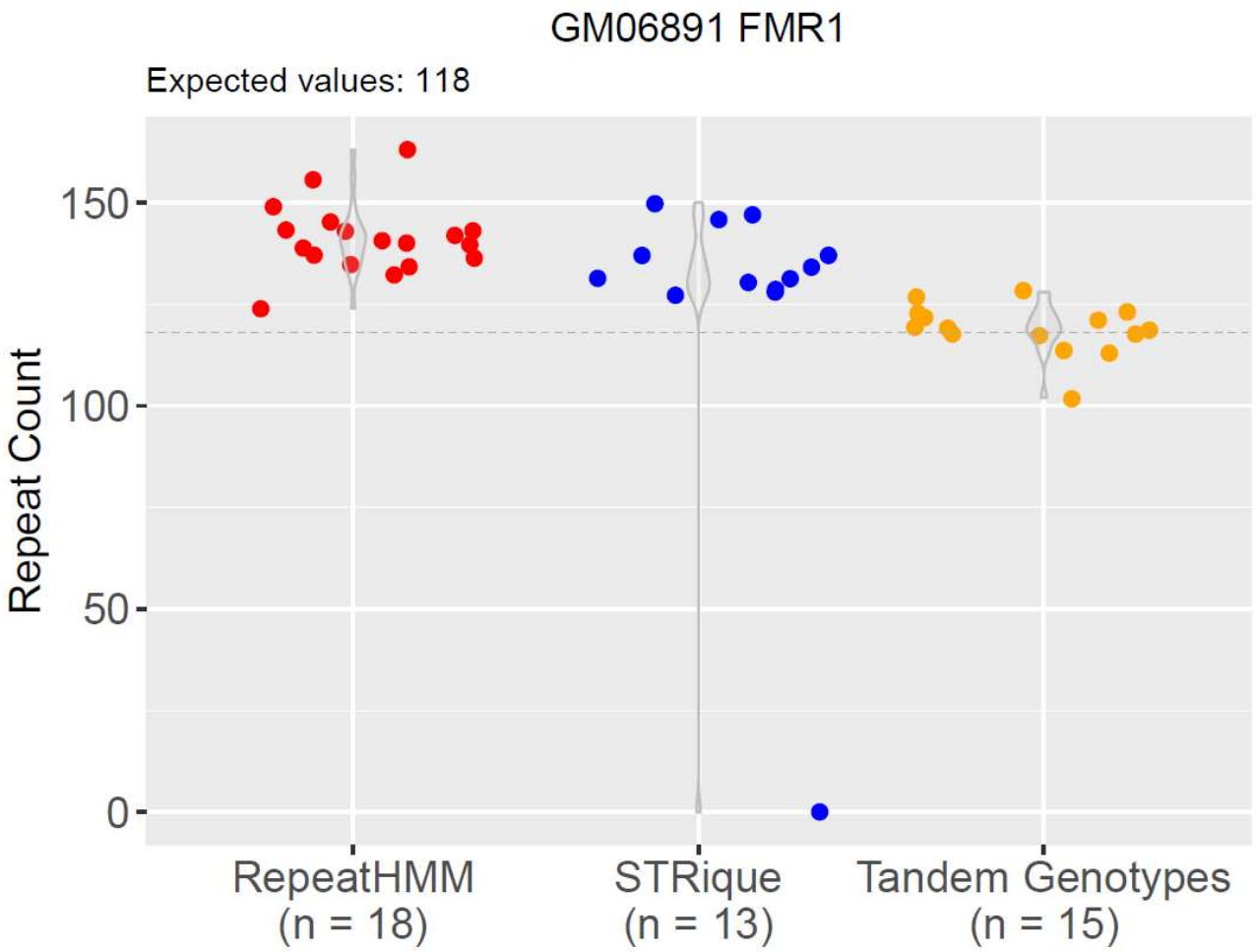

GM06891 HTT
Expected values: 9/20
Repeat Count
50
40
30
20
10
0
RepeatHMM
(n = 37)
STRique
(n = 35)
Tandem Genotypes
(n = 39)

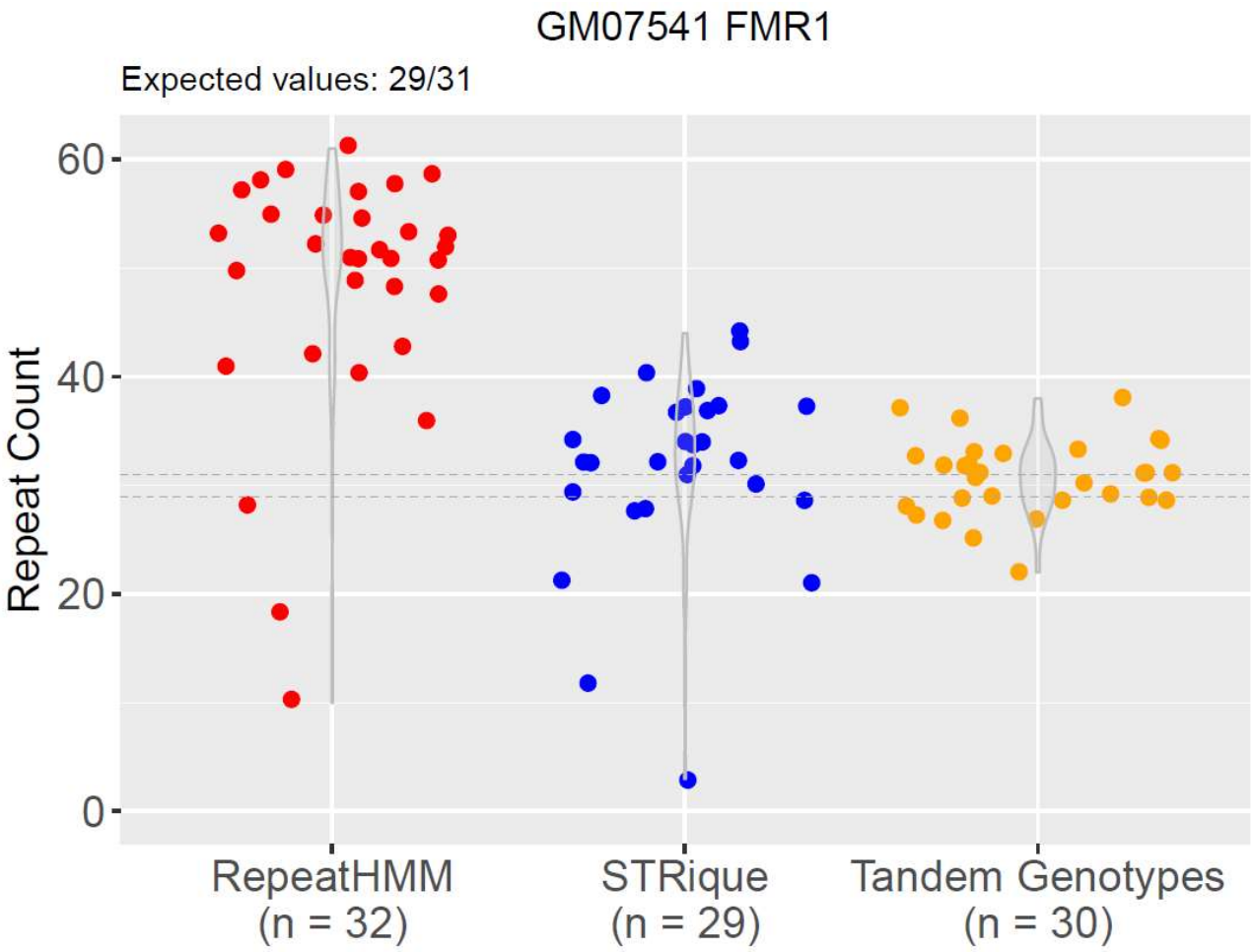

GM07541 HTT
Expected values: 17/22
Repeat Count
40
20
0
RepeatHMM
(n = 45)
STRique
(n = 40)
Tandem Genotypes
(n = 45)

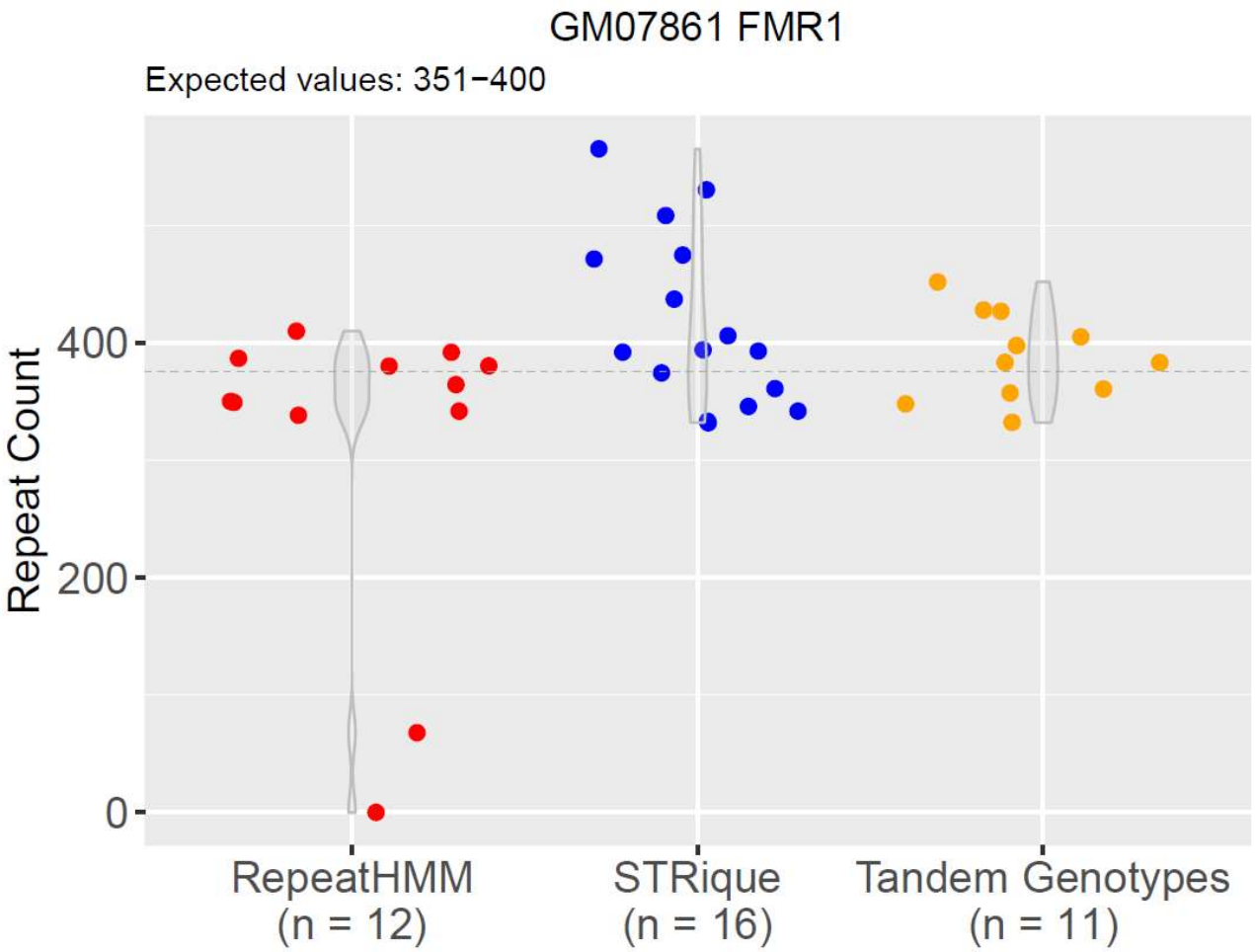

GM07861 HTT
Expected values: 19/20
Repeat Count
40
20
0
RepeatHMM
(n = 36)
STRique
(n = 29)
Tandem Genotypes
(n = 32)

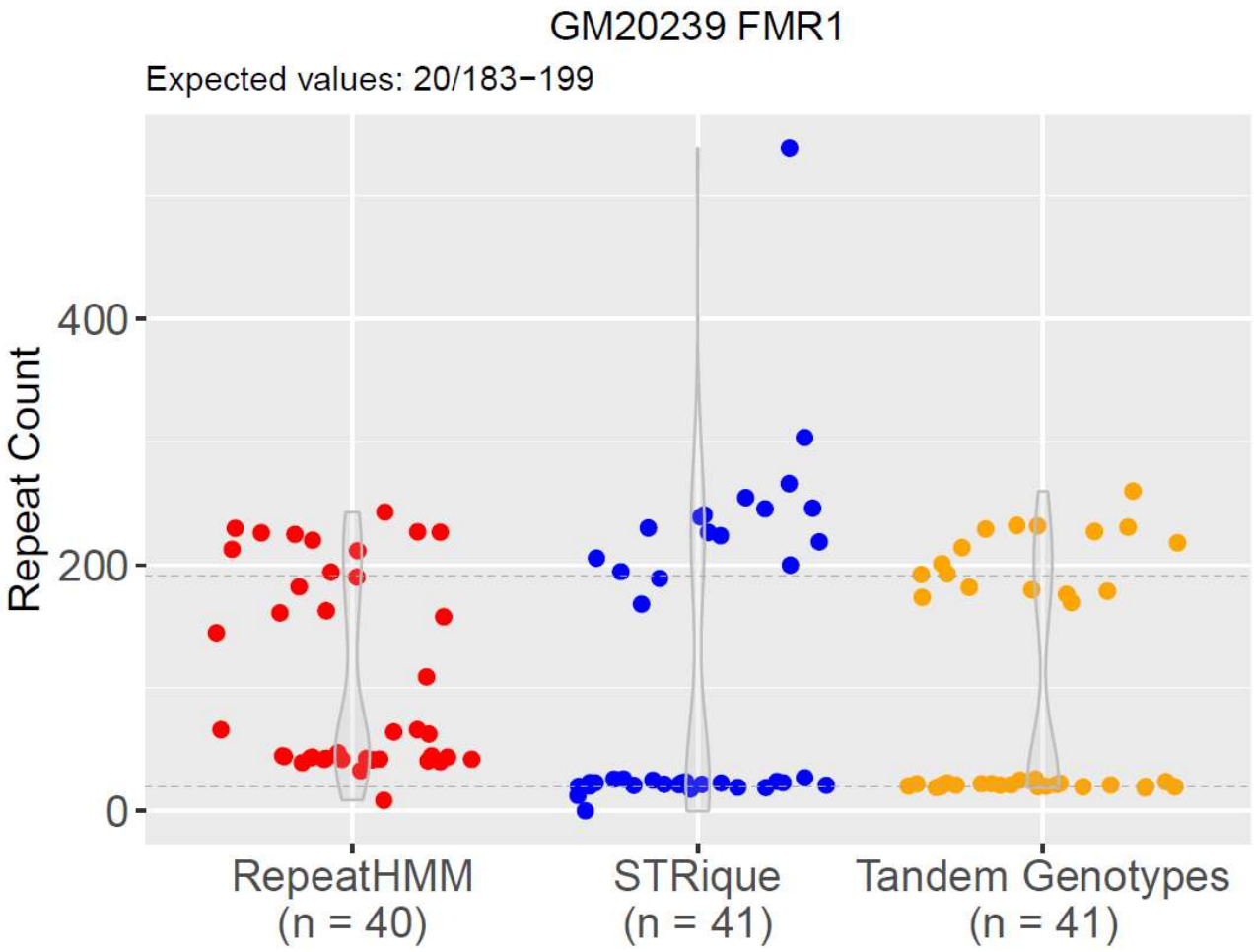

GM20239 HTT
Expected values: 18/18
Repeat Count
100
75
50
25
0
RepeatHMM
(n = 68)
STRique
(n = 62)
Tandem Genotypes
(n = 57)

Appendix 9: Unphased Combined Repeat Count Figures for FMR1




GM07541 FMR1
Expected values: 29/31
Repeat Count
60
40
20
0
RepeatHMM
(n = 32)
STRique
(n = 29)
Tandem Genotypes
(n = 30)


GM07861 FMR1
Expected values: 351−400
Repeat Count
400
200
0
RepeatHMM
(n = 12)
STRique
(n = 16)
Tandem Genotypes
(n = 11)


GM20239 FMR1
Expected values: 20/183−199
Repeat Count
400
200
0
RepeatHMM
(n = 40)
STRique
(n = 41)
Tandem Genotypes
(n = 41)