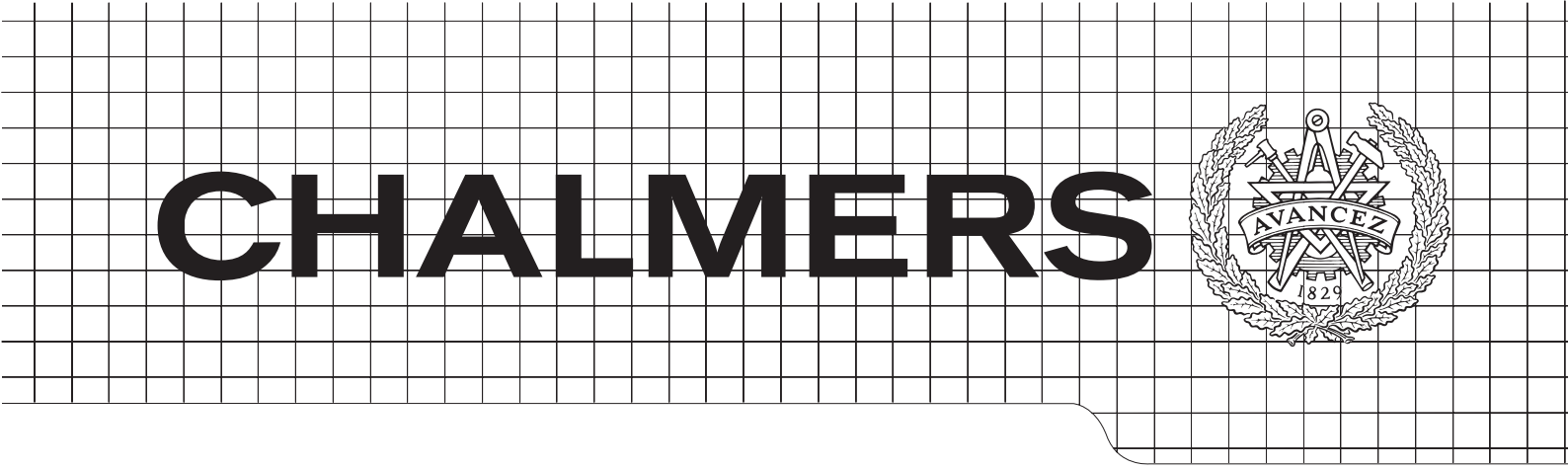

Population Based Microsatellite Genotyping

Master of Science Thesis in Algorithms, Languages & Logic

Snædís Kristmundsdóttir

Department of Computer Science & Engineering
Chalmers University of Technology
Gothenburg, Sweden 2014

MASTER'S THESIS

Population Based Microsatellite Genotyping

Snædís Kristmundsdóttir

JUNE 2014

Department of Computer Science & Engineering
Chalmers University of Technology
Göteborg, Sweden 2014
Examiner: Devdatt Dubhashi

Statistics
deCODE genetics
Reykjavík, Iceland
Supervisor: Bjarni Vilhjálmur Halldórsson

Population Based Microsatellite Genotyping
Snædís Kristmundsdóttir

Department of Computer Science & Engineering

Chalmers University of Technology
SE412-96 Gothenburg
Sweden

Abstract

Microsatellites, also known as short tandem repeats (STRs) are short DNA sequences containing repeated motifs ranging from 2-6 bases. The number of repeats varies between individuals and the numbers occurring in a population are known as the alleles of a microsatellite. Each individual carries two copies of each chromosome and hence two alleles of each microsatellite. There are at least 250.000 microsatellites that have a known location on a human reference genome, the most common form is dinucleotide repeats.

The range of applications for microsatellite analysis is very wide and includes among other things medical genetics, forensics and genetic genealogy. However, microsatellite variations are rarely considered in whole-genome sequencing studies in large due to a lack of tools capable of analyzing them.

The goal of this thesis is to create a microsatellite genotype caller which is faster and more accurate than others previously presented. In order to accomplish this goal two things were examined. First, we reduce by 87% the amount of sequencing data necessary for creating microsatellite profiles using previously aligned sequencing data. This was achieved by filtering the input to contain only reads aligned to known microsatellite locations and unaligned reads as these should be the ones useful for profiling. The results indicate that when performing microsatellite profiling using previously aligned data it is possible to significantly reduce running time with negligible effects on the resulting profile. Second, the accuracy of the microsatellite profiler was increased from 87.5% to 96.3%. The improvements included using population information to train microsatellite and individual specific error profiles. This was done by adding parameters to the model as well as using sequencing data from multiple individuals to improve parameter estimates. Combining these two procedures we were able to give a practical implementation of microsatellite genotyping which is both much faster and more accurate than previously presented solutions.

Acknowledgements

First, I would like to express my gratitude to deCODE genetics for making this thesis possible and to my supervisor at deCODE, Professor Bjarni Vilhjálmur Halldórsson for his patience and guidance. I would also like to thank my friend and colleague Dr. Birte Kehr for her valuable opinions and for always listening to my ideas. Last, I would like to thank my family for putting up with me during these past few months.

Contents

Chapter 1

Introduction

1.1 Genetic variation

Any two individuals differ in approximately $6 \cdot 10^6$ base pairs in their DNA sequence. Although this may appear to be a large number, it is only 0.1% of the entire human genome as the size of the human genome is around $3 \cdot 10^9$ base pairs and each individual carries two copies of each chromosome. Despite its small relative size this 0.1% can make a lot of difference and it is believed to be responsible for many of the observable differences between individuals.

The differences in genetic sequence between individuals are known as genetic variations and determining these differences is referred to as *genotyping*.

There exist many different types of genetic variations, the most common one being single nucleotide polymorphisms (SNPs). Single nucleotide polymorphisms are DNA sequence variations where a single nucleotide (A, T, G or C) is altered. Indels are another frequent type of genetic variation. Indels occur when one or more base pairs are present in some genomes but absent in others. Block substitutions are another type of variation and they arise when a string of adjacent nucleotides is different between two genomes. An inversion variant describes cases where the order of the base pairs is reversed in a defined section of a chromosome. The phenomenon when an identical or nearly identical sequence is repeated in some chromosomes but not in others is called a copy number variant. Normally indels and copy number variations are only a few bases long but can span entire chromosomes, an example of this are individuals with Down's syndrome which have an extra copy of chromosome 21 [19].

1.2 Microsatellites

Microsatellites are a form of repetitive DNA sequences in the human genome and are some of the most common form of variation. They occur when a short sequence of DNA (2-6 bases) is repeated a number of times [21]. The mutation rate for microsatellites is much higher than for other variations and its value has been estimated between $1 \cdot 10^{-4}$ and $1 \cdot 10^{-3}$ mutations per locus per generation [31]. The reason for this high mutation rate is their repetitive structure. It causes a secondary DNA conformation that makes replication slippage events more likely than in other locations of the genome [26]. Replication slippage is an error which occurs when DNA molecules are copied. The error results in either an increase or decrease in the number of motif repeats. During replication the copy strand being created and the original template strand get shifted in their relative positions which causes a part of the template to either be copied twice or not copied at all, see Figure 1.1. As a result of this error the newly replicated polynucleotide has either an extra repeat unit or is missing a repeat unit [12].

The effect of replication slippage is different depending on the cell it happens in and when it happens. If a slippage event occurs during replication of a sex cell it results in what is called a germline mutation. If the mutated sex cell takes part in fertilization then the mutation is passed on to the individual's offspring. Slippage events can also occur during the development of somatic tissue and this is called a somatic mutation. The mutated cell can divide and create more cells carrying the same mutation but the mutation will never be passed on to an offspring [20]. During some sequencing processes the DNA molecules in the sample are replicated by PCR amplification as a part of the sample preparation. If a slippage event occurs during this process it generates stutter noise in the sample which results in wrong sequencing results from the molecules containing the mutation. This makes the analysis of the sequencing results harder and may result in genotyping errors.

The high mutation rate causes the number of repeat motifs at a microsatellite locus, known as the alleles of a microsatellite, to vary greatly between individuals [31]. Each individual carries two copies of each chromosome and hence two alleles of each microsatellite. No pair of individuals alive today have the same combination of alleles for all microsatellites [22]. This makes it possible to create a unique genetic profile for every individual if enough microsatellites are profiled, with the exception of identical twins who share 100% of their DNA [3]. If both copies of the chromosome have the same allele of a microsatellite the individual is a homozygote at that microsatellite but if they are different he is a heterozygote.

Such high variability of repeat numbers between individuals makes the analysis of microsatellite variations highly desirable and makes it amenable to a wide range of ap-

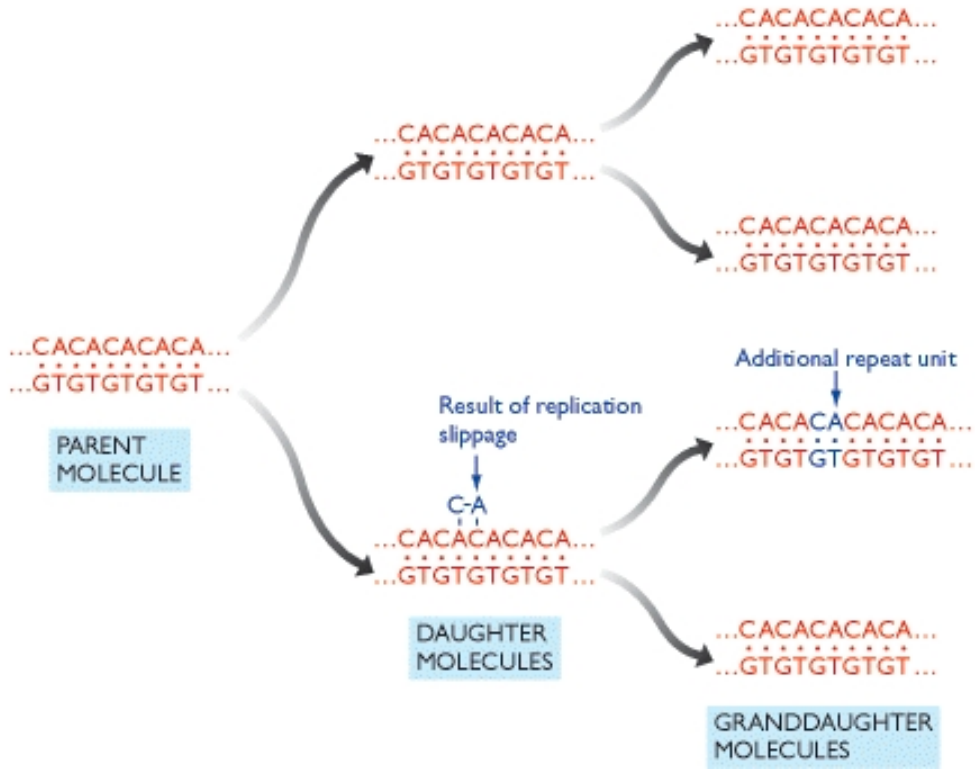

Figure 1.1: An extra repeat element added because of replication slippage [12].

plication. Despite this they remain comparatively understudied and this is in large part caused by a lack of tools capable of analyzing them [17]. Until 2006 microsatellites were the most commonly studied class of variation but with the development of new sequencing and genotyping techniques the focus has shifted to other genetic variations such as single nucleotide polymorphisms and indels.

Microsatellites have a number of attributes which make their analysis using high throughput sequencing analysis pipelines very hard. High throughput sequencing is the most widely used sequencing technology today. This technique generates short DNA fragment sequences called reads which are randomly sampled from the individuals genome. The reads are then assembled to create the genome of the individual being sequenced. Reads can be anywhere from 35 base pairs long and up, depending on the details of the sequencing technique used but the most common length is around 100 base pairs. For a read to be useful in microsatellite genotyping it must completely encompass the microsatellite. If a read only contains a portion of the microsatellite it can only give a lower bound on the number of repeats and thus only reads fulfilling this condition can be used for determining the number of repeats [21]. Another challenge is how most popular read-to-reference aligners (see Section 2.1.1) trade-off between the tolerance of insertions/deletions and running time. A reference genome is the sequence of a template individual DNA used

as reference when finding where on the genome a read originates from. Once reads have been aligned one can identify differences to the reference individual. Aligning a read to a reference genome becomes very hard if it contains a microsatellite, even if the microsatellite only has small number of added/deleted repeat motifs compared to the genome used as reference. This is due to the gaps that the extra/missing base pairs create in the alignment [24]. PCR amplification, a routine step in sample preparation for whole genome sequencing, can create stutter noise. When this happens the DNA amplicons contain a false repeat length caused by successive slippage events during amplification. It results in reads declaring a false allele which damages the results of genotyping [18]. The conclusion to be drawn from this is that despite its desirability, the analysis of microsatellites is very challenging using the sequencing and analysis techniques available today.

1.3 Outline

The goal of this thesis is to increase the speed and accuracy of microsatellite profiling. First, we reduce the running time of a microsatellite profiler when running it on previously aligned sequencing data without damaging its results. A microsatellite profiler is a program that given sequencing data, determines an individuals' number of repeats at a microsatellite locus.

Second, we create a microsatellite genotype caller using population information that is more accurate than previously presented callers of this kind.

We show that the running time of a microsatellite profiler can be reduced by running it on previously aligned data and using the output of premapped BAM-files.

We provide a novel microsatellite genotype caller that uses population information to construct an error model for profiling microsatellites. The model uses the population information to first determine what distinguishes a sequencing error read or a misplaced read from a good read and then trains microsatellite and individual specific error models. Comparison with results from the previously mentioned microsatellite profiler reveal that the accuracy of genotyping is increased from 87.5% to 96.3%. This is due to the new approach used to identify different types of sequencing errors and the quality of parameter estimation.

The disection of this thesis is as follows. Chapter 2 gives an introduction to sequencing techniques, formats for storing genetic data, some of the previous work done on microsatellite profiling and the algorithmic methods applied. A description and motivation of the data sets used is given in Chapter 3 along with a description of the capillary electrophoresis genotypes used for benchmarking. The input reduction process used to lower the running time of a microsatellite profiler and its results are presented in Chapter 4. The

results indicate that a significant decrease in running time is possible with negligible effect on the generated profiles when using previously aligned data for profiling. The average running time reduction achieved for a set of 80 individuals was 74.4% and the average effect on results was less than 2%. Chapter 5 gives a description of both the error model applied in the microsatellite profiler examined in Chapter 4 and the error model implemented with the intention of increasing the genotyping accuracy. A comparison of their performance relative to the capillary electrophoresis genotypes is covered as well. The comparison shows an substantial improvement in genotyping accuracy for 362 individuals on 185 microsatellites.

Chapter 2

Background

Here we give an introduction to concepts necessary while reading the following chapters. We introduce methods and techniques for sequencing DNA molecules and aligning short sequences to reference genomes. Formats for storing and annotating genetic data are introduced and defined. We give a detailed description of the microsatellite profiler used for running time reduction and for comparison with the genotype caller presented. Last, we give a brief summary of the algorithmic methods applied.

2.1 Sequencing methods and reference genomes

The goal of DNA sequencing is to determine the order of nucleotides in a DNA molecule. All sequencing technologies presently available use the same high level process.

First, a sufficient number of DNA molecules is obtained, either through replication of a DNA molecule from the sample or by using a sample containing an ample amount of DNA molecules. Next, the molecules are broken into small fragments at random locations and the order of bases in the fragments is determined. These sequenced fragments are called reads.

During sequencing every base is given a PHRED quality value which is a log value of the probability of sequencing errors, this probability is estimated during sequencing and the PHRED quality value is computed in the following way

$$PHRED\,score = -10 \cdot log10(error\,probability) \tag{2.1}$$

Bases with a low PHRED score are therefore more likely to be the result of errors during sequencing. The PHRED scores are one of the attributes considered here when classifying error reads from true reads.

As the reads are randomly broken, their location is not known and they must be

aligned to a reference genome, the DNA sequence of a template individual, to determine their genomic location. Aligning reads to a reference genome is a non trivial and very time consuming task, it is prone to errors and for some reads an alignment to the reference genome can not be found. Alignment to reference genomes is covered in more detail in section 2.1.1. When the reads where alignment is possible have been aligned to the reference genome, the DNA sequence has been determined as well as the sample allows. What varies between sequencing technologies are among other things the *read length* and the number of times on average each location in the sample is sequenced, or the sequencing *coverage*. The most commonly used sequencing technologies and the ones used at deCODE break the DNA molecules into fragments that are around 500 base pairs long and sequence around 100 base pairs from each end of the fragment. The number of times the sample is sequenced determines the coverage of the sequencing which is defined as the average number of pieces overlapping each base. In high coverage sequencing data an error read is less likely to impact genotyping results since many other reads cover the same genomic area.

2.1.1 Reference genomes and alignment to them

A reference genome is the DNA sequence of a representative individual used to determine the genomic location of reads generated during the sequencing of another individual. Most individuals will not have a sequence very different from the reference sequence and genetic variants can be discovered by considering the difference between the individual being sequenced and the reference genome. The first human reference genomes were introduced in February 2001 when the Human Genome Project (HGP) and Celera Genomics published their results to that date. The HGP presented a 90 percent complete sequence of all three billion base pairs in the human genome [1] and Celera presented a nearly complete sequence of 92% of the human genome [33]. Both were of comparable quality and since then many improvements and additions have been made to this sequence, the latest version was released in December 2013.

Once a fragment of DNA has been sequenced the next step of analysis is to align it to a reference genome to reveal its genomic location. Alignment programs search the entire reference genome to find where the sequence being aligned fits best. When reads are aligned to a reference genome they don't always match completely, this could be due to genetic variations, misplacement of the read or sequencing errors for example. The edit distance from the read to the reference is the smallest number of substitutions, insertions, and deletions of bases that can be used to transform the read to the reference. Error-tolerant alignment can reveal whether the sequence has a variation relative to the reference

and hence find the differences between the reference and the individual being sequenced but the procedure is not perfect. To determine where on the genome the read fits best the alignment programs use a scoring scheme. These scoring schemes give each possible alignment a score which increases for each base matching the reference and decreases for each gap or mismatch introduced in the alignment. The alignment with the highest score is then reported as the location of the read. To be considered valid the score of an alignment must have a minimum value which can usually be set by the user along with the parameters of the scoring scheme. The scoring scheme often gives the correct location, but this problem is particularly challenging in for microsatellites.

2.1.2 Single and paired end sequencing

The two main techniques used in DNA sequencing are single end sequencing and paired end sequencing which produce single-end reads and paired-end reads, respectively [7]. While the single end approach sequences the DNA fragment from one end only, the paired-end one sequences it from both ends. This produces read pairs containing reads from both ends of the DNA fragment. The distance between read pairs is short and follows a tightly bound distribution. The usage of paired-end reads can help detect re-arrangements, and because the approximate distance between reads in a pair is known, alignment algorithms can use this knowledge to align the reads covering repetitive regions more precisely, see Figure 2.1. This improves the overall alignment quality, especially in repetitive genomic regions where alignment of reads is hard [6]. All sequencing data

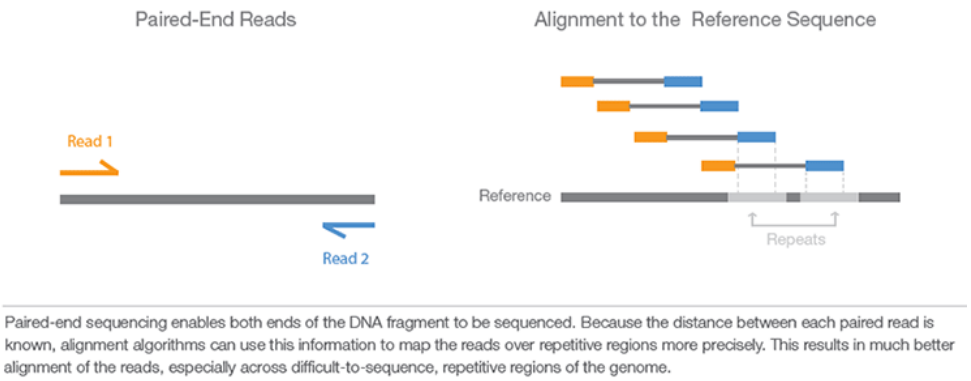

Figure 2.1: Both ends of the DNA fragment are sequenced with a known distance between then which improves alignment. [6].

analyzed in this thesis comes from paired end sequencing performed using Illumina sequencers and when discussing *mates* of reads this refers to the other member in the read pair. Paired end sequencing is currently the most widely used form of sequencing.

2.1.3 Chain termination sequencing - Sanger sequencing

The Sanger sequencing method uses a purified DNA polymerase enzyme to synthesize DNA chains that vary in length. The key feature in the reaction mixture used in this method is including chemically modified nucleotides that terminate the elongation of the DNA strands. This gives DNA fragments of varying length which are then separated by size using gel or capillary tube electrophoresis. In this separation method an electric field pulls molecules across a substrate or hairlike capillary fiber and it is sensitive enough to distinguish DNA fragments with only a single nucleotide length difference. The fragments line up by size and the base at the end of each fragment can be determined. From this, the DNA sequence can easily be read. Each reaction can result in a read between 750-1000 base-pairs long using the latest machines [11].

2.1.4 Microsatellite STR Analysis using capillary electrophoresis

This method knows beforehand the location of the microsatellite. A unique sequence on either side of the microsatellite is identified and a primer is used to amplify the region containing the microsatellite. Fragments containing the microsatellite sequence are cut and analyzed using capillary electrophoresis. Since this process can distinguish DNA fragments with only a single nucleotide length difference this enables one to correctly identify the allele of the microsatellite based on the length of the sequence. The results from this procedure however are not perfect since the resulting eletropherograms have to be manually inspected to accurately determine the genotype. Software has been developed with the intention of automating this inspection process as much as possible. An example of this is the *Decode-GT* program which classifies genotyping decisions into good, bad and ambiguous decisions based on the properties of their electropherograms. Only the ambiguous decisions would then need manual inspection while the bad ones are discarded and the good ones are taken as truth without inspecting them [27].

2.1.5 Next generation sequencing

Next generation sequencing is a synonym for technologies that parallelize the sequencing process and produce thousands or millions of sequences concurrently. The reads obtained using this method can be anywhere from 35 base pairs long and up, but the sequencing error rate increases as the read length increases. The most widely used next generation sequencing technology was developed by Illumina and generates reads that currently have a length of around 100 base pairs. All data used in this thesis was generated using the Illumina sequencing technology. Most current next generation sequencing methods apply

the paired end sequencing technology [29].

2.2 Formats for genetic data

2.2.1 FASTA and FASTQ files

The FASTA format is text-based and used to represent both amino-acid and nucleotide sequences. In these files nucleotides and amino acids are represented using a single letter to encode them. It is possible to have sequence names and/or comments precede the sequences [9]. The format was originally created by Bill Pearson as an input format for his FASTA suite of tools, but has become a standard in bioinformatics [28]. A FASTA file stores sequences that all begin with a description line which is distinguished from the sequence data by a ">" symbol as its first letter. Following this symbol is a sequence identifier and following the identifier is a sequence description. Both the sequence identifier and description are optional but if they are present there can not be a space between the ">" symbol and the identifiers first letter. All lines of text in a FASTA file should be shorter than 80 characters. The end of a sequence is identified by the beginning of the next one, i.e. a ">" symbol, or of course the end of the file itself [9]. Figure 2.2 shows an example of a sequence in the FASTA format

```
>gi|186681228|ref|YP_001864424.1| phycoerythrobilin:ferredoxin oxidoreductase
MNSERSDVTLYQPFLDYAIAYMRSRLDLEPYPIPTGFESNSAVVGKGKNQEEVVTTSYAFQTAKLRQIRA
AHVQGGNSLQVLNFVIFPHLNYDLPFFGADLVTLPGGHLIALDMQPLFRDDSAYQAKYTEPILPIFHAHQ
QHLSWGGDFPEEAQPFFSPAFLWTRPQETAVVETQVFAAFKDYLKAYLDFVEQAEAVTDSQNLVAIKQAQ
LRYLRYRAEKDPARGMFKRFYGAEWTEEYIHGFLFDLERKLTVVK
```

Figure 2.2: An example of a sequence in the FASTA format

The FASTQ format is a simple extension to the FASTA format which makes it possible to assign a numeric quality score to each element in a sequence [13]. This format is only used for nucleotide sequences because the quality score is defined for sequences created using next generation high throughput sequencing techniques which are only used for sequencing DNA, not proteins. In FASTQ files the description line is identified by an "@" symbol as its first letter and the next line contains the actual sequence. The third line must start with a "+" symbol and then optionally repeats the description line. The fourth and last line contains the qualities which are defined in terms of the estimated probability of error during sequencing [13]. Figure 2.3 shows an example of a sequence in the FASTQ format

```
@SRR014849.1 EIXKN4201CFU84 length=93
GGGGGGGGGGGGGGGCTTTTTTTGTTTGGAACCGAAAGGGTTTTGAATTTCAAACCCTTTTCGGTTCCAA
CCTTCCAAAGCAATGCCAATA
+SRR014849.1 EIXKN4201CFU84 length=93
3+&$#""""""""""""7F@71,'";C?,B;?6B;:EA1EA1EA5'9B:?:#9EA0D@2EA5':>?:\%A;A
8A;?9B;D@/=<?7=9<2A8==
```

Figure 2.3: An example of a sequence in the FASTQ format

2.2.2 SAM and BAM-files

SAM stands for Sequence Alignment/Map format and SAM-files store aligned sequencing data, i.e. sequencing data that has been aligned to a reference genome. The SAM format is a TAB-delimited text format that contains an optional header section, where all lines start with a "@" symbol. Following the header is an alignment section and each alignment line represents the alignment of a single DNA-sequence read to a reference genome. Each line in the alignment section must have 11 mandatory fields that contain necessary alignment information, these include for example mapping position and quality. In addition, alignment lines can have a variable number of optional fields for other information that might be aligner specific. The mandatory fields can have the values '0' or '*' (depending on the field) if the corresponding information is unavailable [23]. BAM files are compressed binary versions of SAM files and one can translate from one format to the other relatively easily using for example the Samtools software. Most aligned sequencing data is stored in the BAM format due to the size convenience it offers relative to files in the SAM format [23].

2.2.3 VCF-files

Variant Call Format (VCF) is a generic text format used to store genetic variation data such as SNPs, insertions, deletions and structural variants. VCF files are normally stored in a compressed manner. The format was developed for the 1000 genomes project and has since then become the standard format for representing genetic variations. [14]. All VCF files contain the following three sections. Meta-information lines that all begin with "##", one header line beginning with "#CHROM" and data lines that contain the variation type and position along with genotype data, one variant per line. Each VCF record(data line) has the same number of tab-separated fields as the header line. The symbol "." is used to denote missing data.[4]

2.2.4 BED-files

BED stands for Browser Extensible Data and BED is a TAB-delimeted text format for defining genomic regions. It is used by the UCSC genome browser and defines one genomic region, called a BED record, per line [2]. BED records have three required fields and nine additional fields which are optional. The required BED fields are the name of the chromosome, the starting position of the feature in the chromosome and the ending position of the feature in the chromosome. The first base in a chromosome is numbered 0 and the ending position is not included in the display of the feacture. For example, the first 100 bases of a chromosome are defined with a starting position of 0 and an ending position of 100 [8].

2.3 Previous work

A number of attempts have been made to develop software capable of reliably genotyping microsatellites. These include

- *REDEEM* (Read Error DEtection and Correction via Expectation Maximization). The software attempts to model and correct errors in short reads generated by next generation sequencing technology with a repeat aware approach. This would simplify downstream analysis of the sequencing data since it can assume absense of errors with reasonable confidence [34].

- *lobSTR* takes in raw sequencing data, identifies reads possibly containing microsatellites, determines the repeat motifs of those reads and aligns them to a reference genome. The set of reads aligned to each microsatellite is passed to an error model provided with the software which determines the final genotype [21]. Further defined in Section 2.3.1.

- *RepeatSeq* uses informed error profiles to determine human microsatellite genotypes. The method applies a fully Bayesian approach and uses three attributes to determine which error profile to apply when genotyping. These are the length of the repeat in the reference genome, the length of the repeat motif and the average base quality of the bases in the reads. The error profiles to choose from are provided with the software and were created using near homozygous fly genomes [22].

- *GenoTan* uses 2 regression steps to determine the inherited genotype at a microsatellite loci. The first roughly estimates genotypes for the microsatellites using a NLS (Non-linear Least Square) fitting method. The genotypes from the first step are

then used to estimate parameters used in the second step where a final genotyping decision is made [32]. This method only determines genotypes for microsatellites containing homopolymer runs, i.e. when the repeat motif is a single base.

We chose to focus on lobSTR both for reducing running time and for comparison with genotyping accuracy since it is the most highly cited of these. We now give a detailed description of the software.

2.3.1 lobSTR

LobSTR is a microsatellite profiler for personal genomes that works on raw sequencing data in the form of FASTA files, FASTQ files or BAM files. It produces a VCF file specifying an individuals' alleles for all microsatellite loci covered in the input. Before reaching this result the program performs two steps. The first step identifies reads possibly containing microsatellites, determines their repeat sequence and aligns them to a reference genome. The second step determines the alleles for each microsatellite based on the reads aligned to it in the first step [21].

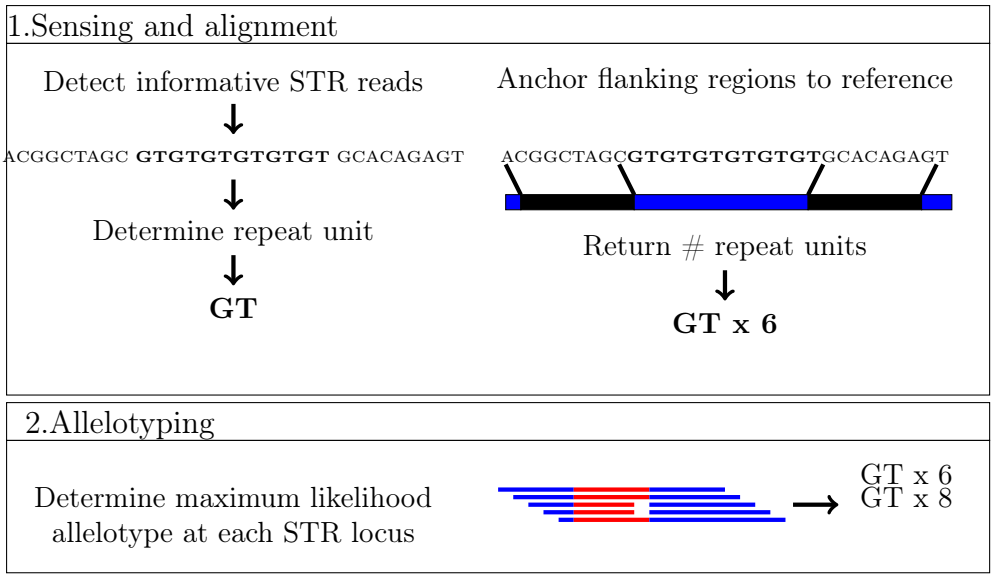

Figure 2.4: Step 1: Find microsatellite reads and align them to a reference genome. Step 2: Determine the genotype [21]

In the first step, entropy calculations are performed on all reads in the raw sequencing data to predict whether a read contains a microsatellite. This is done by splitting each read into overlapping windows of user specified size (default is 24 base pairs with a 12 base pair overlap). One or more consecutive windows that have entropy levels below a certain threshold and have windows with normal levels of entropy on each side are the ones of interest. To compute the length of the repeat motif in reads marked as interesting,

the algorithm uses a spectral analysis approach. First, all consecutive windows in the read, having an entropy score below the threshold are merged and then a fast Fourier transform is performed along columns of a matrix representation of the merged sequence. Because microsatellites have a unique fingerprint in the frequency domain this enables the algorithm to determine the length of the repeat motif. To finally determine the motif sequence all possible sub-sequences of the length returned by the spectral analysis are recorded and the one occurring most frequently is chosen.

Alignment of the reads is achieved by comparing the flanking windows with normal entropy scores to reference flanking areas of all considered microsatellites with the identified repeat sequence and the ones achieving the best alignment are the result. The minimum flanking region length for alignment can be specified by the user and has a default size of 8 base pairs. The reference flanking areas are stored in an index consisting of one file for each repeat motif and a BED-file containing the genomic coordinates of all microsatellites considered. There are two and indices available with the lobSTR software. The smaller one contains the reference flanking areas for microsatellites and the microsatellite genomic coordinates BED-file in the hg18 build of the human reference genome. The bigger one is an extension of the smaller one and the BED-file and the reference flanking areas are all given in the hg19 build of the human reference genome. This method of alignment eliminates the problem of a gapped alignment which occurs frequently in other alignment programs when dealing with repetitive areas in the genome. A BAM file containing the alignments of all chosen reads as well as their repeat motifs is the output of this step along with another file containing statistics on the results. In the second step, lobSTR performs genotyping using microsatellite aligned reads from the BAM-file generated in the first step along with an error model that models the probability that a read is a result of stutter noise. The model is provided with the software and can be used directly but the user can also choose to train a new model on his own data. This is however only possible on male genomes containing a large number of reads aligned to microsatellites located on the sex chromosomes. Since males have only one copy of each sex chromosome homozygosity can be assumed and used to identify error reads. Included in the noise model as influential factors are motif length, microsatellite region length, GC content of flanking regions and microsatellite purity. This model is described in detail in Chapter 5. A visualization of the whole process can be seen in Figure 2.4.

2.4 Algorithmic methods

2.4.1 Expectation maximization

The expectation maximization algorithm is used for estimating parameters of probabilistic models with incomplete data by alternating between 2 steps, the E-step and the M-step. In the E-step the algorithm computes an expected probability distribution of missing data given the current model. In the M-step the algorithm then re-estimates the model parameters using the data completion obtained in the other step. The name of the E-step is derived from the fact that it is not necessary to compute the probability distribution over the completions explicitly, it only has to compute the **E**xpected statistics over the completions. In the same way the name of the M-step is derived from the fact that re-estimating the model is actually **M**aximizing the expected log-likelihood of the data [15].

Here we will use expectation maximization in error model training. We will train classifiers to classify between true reads and error reads caused by undefined errors and estimate rates of slippage errors at each microsatellite location considered, see Section 5.2.3.

2.4.2 Logistic regression

Logistic regression is used for estimating the probability of an event. It uses the knowledge obtained from relevant explanatory variable/s to predict the probability p of an event occurring. This is different from linear regression where the event we want to predict is a precise numerical value, for example the height of a person in a specific population. The relationship between the predicted probability and explanatory variables/s is not linear and is described instead using the logistic regression function where p is the probability we want to predict and x is the independent relevant variable:

$$p = \frac{e^{b_0+b_1 x}}{1 + e^{b_0+b_1 x}} \tag{2.2}$$

During logistic regression one estimates the values of b_0 and b_1 using the explanatory variable [10]. The logistic curve is frequently used to model population growth, survival from a disease or the spread of a disease [30].

Here we will use logistic regression to fit a regression curve for each microsatellite considered using expectation maximization. The logistic regression will compute for each read a probability that this read supports a true allele.

2.4.3 Feature selection

Feature selection is the process of selecting a subset of the attributes available in the training set and using this subset as variables in classification. The training set is a set of observations for which we know the labeling, i.e. we know the class of each sample. Each sample then has characteristics that are called attributes and if many samples within the same class have the same value of a specific attribute then it characterizes the class. The purpose of a classifier is to determine the class of the sample/s given as input. Feature selection has two main purposes. First, it makes the training and application of a classifier more efficient by decreasing the size of the effective attributes. This is very important for classifiers that are expensive to train. Second, feature selection often increases classification accuracy by eliminating noise features. A noise feature is a feature that increases the classification error on new data when it is added as an attribute. For example an attribute can contain no information about a class but it happens to have the same value for all examples from a given class. Including this attribute might then result in a classifier that misassigns test samples containing this value of the attribute to this class. When an accidental property of an attribute in the training set causes an incorrect generalization like this it is called overfitting.

We can view feature selection as a method for replacing a complex classifier (using all features) with a simpler one (using a subset of the features) [25].

Here we will use a form of feature selection to determine the optimal subset of features for fitting a logistic regression curve that predicts the probability that a read reports a true allele.

Chapter 3

Data description and arguments for data selection

The DNA sequence data used in this thesis is whole genome, paired end, sequence data of Icelanders generated at deCODE genetics.

3.1 Data for read selection

The data used to examine the effects of input reduction on the microsatellite profiling results came from 80 BAM-files, each containing sequencing data for one individual. The reads in the BAM-files were all aligned to the second latest version of the human genome reference (hg19). We chose a set of 80 individuals and the average file size before input reduction was 58.9GB. Figure 3.1 shows the size distribution of these files.

3.2 Data for error model

The error model training was performed using data stored in 362 BAM-files, each containing sequencing reads from one individual. The reads were aligned to the third latest version of the human genome reference (hg18). Error models were trained for 185 microsatellites coming from all chromosomes.

The coverage of sequencing data is a good attribute to measure its quality. High coverage data gives genotyping results that are more conclusive and reliable. A definition of sequencing data coverage is given in Chapter 2. We chose a set of 412 individuals whose BAM files contained sequencing data with coverage greater than 40x.

deCODE genotypes which were obtained by microsatellite analysis using capillary electrophoresis were used as benchmarking data to estimate the accuracy of the error

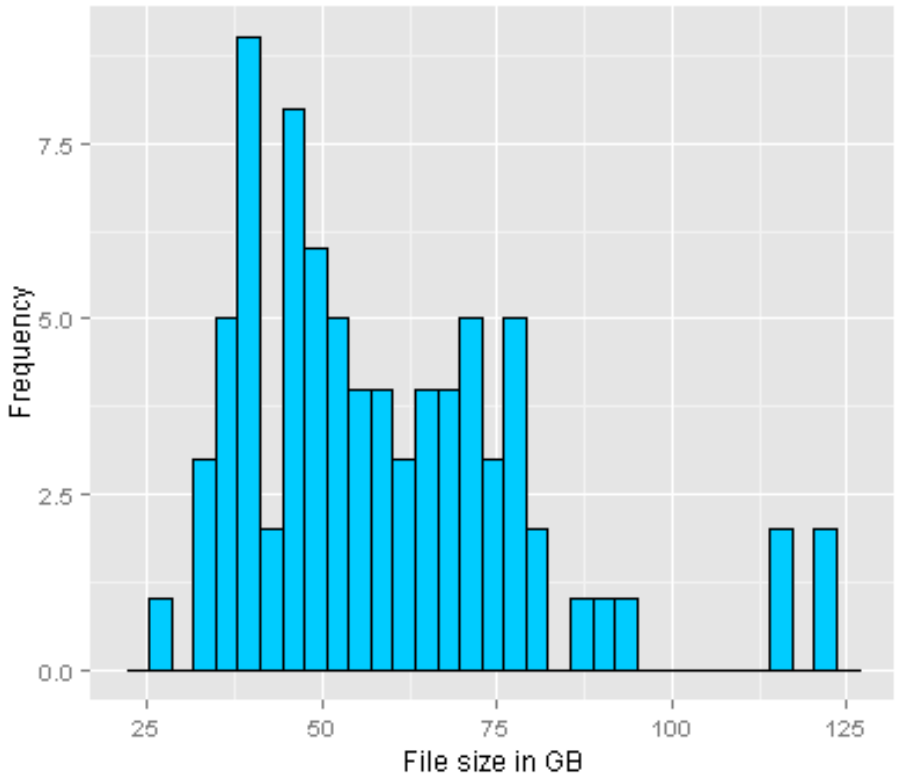

Figure 3.1: Size distribution of BAM-files used for read selection.

model, these are described in Section 3.3. This microsatellite analysis method has been estimated to be around 99% accurate so they were taken to be the truth. No benchmarking data turned out to be available for a number of the original 412 high-coverage individuals and some files were inaccessible which reduced the set to 383 individuals. Furthermore, inconsistencies between capillary electrophoresis genotyping rounds were discovered for 21 out of these 383 so the final set contained 362 individuals. These inconsistencies are explained further in Section 3.3.

3.2.1 Choosing microsatellites

Care had to be taken when choosing microsatellites because comparison of genotyping results to the benchmark data had to be possible. The BAM-files of the chosen individuals had to contain reads covering the microsatellite and benchmark data had to be available for the individuals at this microsatellite as well.

Error models should be trained for microsatellites where matching benchmarking and training data but they should also be confirmed previously as good genetic markers. Because of this the set of microsatellites considered contained only ones that had been

verified at deCODE to be good markers in the paper *A direct characterization of human mutation based on microsatellites* [31].

The first criteria for choosing microsatellites from the set considered was that benchmark data had to be available for at least 150 of the 362 individuals. Furthermore, BAM-files of at least 350 individuals had to contain reads aligned to the microsatellites. This criteria returned 457 microsatellites.

Next, the quality of the sequencing data stored in the BAM-files used for training was assessed. The number of instances where the data never could have given the correct result was computed for all 457 microsatellites. The correct result for a microsatellite in a given individual was considered possible when there existed read/s reporting the right allele/s in the set of reads covering the microsatellite in the individual. The results showed that the average ratio where the correct result was never possible was 3.6% per microsatellite with a maximum of 60.1% and a minimum of 0%. The microsatellites retained were the ones for which this ratio was 1% or less since this is the expected value for this number. This reduced the final set size to 185 microsatellites.

Two C++ programs were written using a sequencing analysis library called SeqAn to identify the microsatellites satisfying the criteria from the set considered and to extract relevant attributes. The first of the two programs identified the microsatellites satisfying the data availability criteria and filtered reads aligned to them from the BAM-files. The second one took as input the filtered reads and extracted from them the attributes chosen for model training. These programs will now be described in detail.

3.2.1.1 Pre-processing of training data

To identify and align reads containing microsatellites the sensing and alignment step of lobSTR was run on the 362 high-coverage individuals. The training data was extracted from the BAM-files generated by this step as they contained only reads aligned to microsatellites.

The sequencing method used to obtain the benchmarking genotypes returns the location of primers close to the microsatellite rather than its exact coordinates. Because of this the genomic coordinates for microsatellites in the training data did not match the ones for the benchmarking microsatellites.

To translate between these two types of microsatellite coordinates a C++ program was written. The program goes through a BAM-file generated in the sensing and alignment step of lobSTR and checks for all reads if the coordinates of the microsatellite they are aligned to fall between the coordinates of any benchmarking microsatellite. When such a read is encountered a translation between the two coordinates for that microsatellite is obtained. To find all "translate-able" microsatellites and determine how many individu-

als they occur in, this program was run on the BAM-files generated in the sensing and alignment step of lobSTR for all chosen individuals. A BED-file containing benchmarking microsatellite coordinates for all microsatellites considered was passed along with them. Each time a translation was found it was written to an output file common to all individuals. This made it possible to count for all translate-able microsatellites how many individuals they occurred in. The ones occurring in more than 350 individuals were cross-referenced with the microsatellites where benchmarking data was available for more than 150 individuals. The intersection of those two contained 457 microsatellites. To then create BAM-files containing only reads aligned to these chosen microsatellites the same program was run again on same BAM-files but this time the BED-file passed along with them contained only the coordinates of the chosen microsatellites. An option was added to the program to specify whether the matching reads should be written to an output file for each individual and switched on for this second run.

At this point the data was still in BAM-format so some further manipulations were made before the data was imported into R where the error model was implemented. To achieve this and to get an rough estimation of genotypes used as error model initialization a second C++ program was written, again using SeqAn. This program computed, for each read in a BAM-file, the various attributes thought to indicate whether the read represented a true allele, was a result of replication slippage or a result of some other error. To initialize the error model training all reads supporting the most frequent allele at each location were labelled as representing a true allele. If the number of reads supporting the second most frequent allele was more than 15% of the number of reads supporting the most frequent one then those were marked as representing a true allele as well and the individual was considered a heterozygote. All reads supporting an allele with one motif-repeat less than either of the true alleles at each location were marked as a result of replication slippage during PCR amplification and the rest were marked as error reads. All the computed values and the labeling were then written to an output file to be passed to R where the data quality estimation which identified the cases where a correct result was never possible was performed.

3.3 Benchmarking data from capillary electrophoresis

Genotypes obtained using microsatellite analysis by capillary electrophoresis available at deCODE were used as benchmarking data. Prior to 2006 microsatellites were the most common form of genetic variants studied at deCODE and most of the data is generated before that date. A file containing all genotypes available for the initial 412 individuals was generated and this revealed the individuals for which no benchmarking data was available. As mentioned earlier, the removal of these individuals and the ones with inaccessible BAM-files reduced the set size to 383 individuals.

The benchmarking data from the 383 individuals was examined further and inconsistencies in results between genotyping rounds were identified in 21 individuals. These rounds were all stored in the same database but only one copy of results for each individual was initially extracted since these inconsistencies were not anticipated. The inconsistencies makes it impossible to decide which genotype is the true one and thus these individuals could not be included. Causes for this may include sample mix ups or other types of man-made errors during sample preparation, sequencing and genotyping. The final set therefore contained 362 individuals.

The file containing all available genotypes for the final set of 362 individuals was used to determine the microsatellites where data was available for more than 150 of the individuals. The microsatellites where benchmarking data was available for more than 150 was cross referenced with the microsatellites where training data was available for more than 350 individuals and the intersection returned 457 microsatellites. Since the coordinates for the deCODE-microsatellites are not exact coordinates but rather boundaries on a genomic region containing them these coordinates did not match the ones returned by lobSTR exactly. A translation scheme was created in the C++ program mentioned in the previous section to handle this. The quality assessment of the data described in Section 3.2.1 then determined the 185 microsatellites included in the final set.

Chapter 4

Read selection

Here we explain the input filtering process used to reduce the running time of a microsatellite profiler. We present the results which are very conclusive and show an average running time reduction of 74.4% with very little effect on the microsatellite profile generated, less than 2% on average. For unfiltered input files containing sequencing data for 80 individuals the average running time was over 100 hours. This would sum up to 2.000.000 CPU hours for sequencing data of 20.000 individuals which is not feasible. Since the running time has had a prohibitive effect on the usage of this tool at deCODE this type of data pre-processing could possibly enable its usage in the future.

4.1 Input manipulation

We select reads from BAM-files to perform microsatellite profiling using the fact that they are aligned. This filtering reduces the input size for profiling and as a result the running time. We compare the final genotyping results with results obtained using the entire contents of the BAM-file.

The input reduction utilizes that because the sequencing data has already been aligned it is possible to select only reads aligned to known microsatellite locations as well as reads that have not been properly aligned. We use the BED-file provided with the microsatellite profiler. This file contains locations of all microsatellites considered by the profiler and allows us to choose only reads with alignments intersecting these locations, along with their mates. We will also include all unaligned reads along with their mates. The unaligned reads are included since we don't know if they contain microsatellites or not. The sequences already aligned to a non-microsatellite sequence are unlikely to contain to a microsatellite while sequences that are unaligned may in fact contain a microsatellite but have not been aligned because they are too different from the reference.

In addition to this, the input reduction procedure throws away low quality reads, i.e.

the ones that fail platform or vendor quality checks and reads that are PCR or optical duplicates. This should increase the result quality since the reads being removed could possibly affect the results in a bad way. We run the microsatellite profiler on only the filtered reads and since this decreases the input size it decreases the running time as well. This is done without damaging the genotyping results because the reads that are discarded very likely do not contain microsatellites.

The high level idea behind the filtering process is to simultaneously scan the BAM-file and BED-file given as input and compare all reads in the BAM-file with the current BED-file microsatellite coordinates. If a read intersecting a microsatellite is found it is filtered along with its mate. When the BAM-file scan reaches reads with alignment locations greater than the current BED-file microsatellite coordinates it updates them to the next one behind it.

The results indicate that the difference between results obtained from filtered input and unfiltered input is very little and can be explained in large by the removal of the low quality reads performed by the filter. Thus, it should be safe to conclude that the benefits from filtering the input are greater than the loss and that this is a good way to reduce running time when generating microsatellite profiles using previously aligned sequencing data.

Pseudocode . We let r_i^s and r_i^e denote the start and end positions of the alignment for read number i, respectively. We set m_j^s and m_j^e as the start and end positions of microsatellite number j, respectively. For properties regarding the read we let r_i^u be a logical value which is true if read i is unaligned, r_i^q be set to true if the read passes quality checks and r_i^d be set to true if the read is an optical duplicate. Both the reads and microsatellites must then be sorted in the same way such that $\forall i, k$ if $i < k$ then r_i comes before r_k and $\forall j, l$ if $j < l$ then m_j comes before m_l in the sorting. Given a BAM-file containing reads $r_1 \dots r_n$ and a BED-file containing the genomic coordinates of microsatellites $m_1 \dots m_b$ we will start with $j = 1$, consider each read from 1 to n separately and check for the following conditions

1. Is r_i^q = FALSE? If yes, discard it and set $i = i + 1$.
2. Is r_i^d = TRUE? If yes, discard it and set $i = i + 1$.
3. Is r_i^u = TRUE? If yes, print r_i and its mate to the output and set $i = i + 1$.
4. Is $r_i^e < m_j^s$? If yes, discard it and set $i = i + 1$.
5. Is $r_i^s > m_j^e$? If yes, set $j = j + 1$ and start from 4. again.

6. Is $r_i^e \geq m_j^s$ and $r_i^e \leq m_j^e$? If yes, print r_i and its mate to the output and set $i = i+1$.

7. Is $r_i^s \geq m_j^s$ and $r_i^s \leq m_j^e$? If yes, print r_i and its mate to the output and set $i = i+1$.

8. Is $r_i^s \geq m_j^s$ and $r_i^e \leq m_j^e$? If yes, print r_i and its mate to the output and set $i = i+1$.

9. Is $r_i^s \leq m_j^s$ and $r_i^e \geq m_j^e$? If yes, print r_i and its mate to the output and set $i = i+1$.

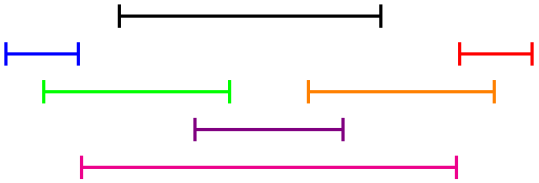

Figure 4.1: The black line at the top shows the microsatellite region. Condition 4 deals with the blue read, condition 5 with the red one, condition 6 with the green one, condition 7 with the orange one, condition 8 with the purple one and condition 9 with the pink one.

Implementation The input reduction program was written in C++ using the sequence analysis library SeqAn which allows for easy reading and manipulation of data stored in BAM-files [16]. After sorting and preprocessing the BED-file containing the microsatellite locations it can be passed to the program along with a BAM file and a name for the output BAM-file. The program goes through the input BAM-file sequentially in parallel with the location file comparing all read alignments to the current microsatellite location. If we find an unaligned read or a read with an alignment intersecting a microsatellite then we mark it and its mate for writing to the output BAM-file. When a read with an alignment located after the current microsatellite is read from the BAM-file then the value of the current microsatellite is updated to the next one. Because of this it is very important to consider the sorting of the BAM-file and make sure this sorting matches the location BED-file.

4.2 Results

To examine the effects of these input manipulations we now reduce the size of the 80 BAM-files chosen earlier, see Chapter 3. The average running time of the filtering program was 1.33 hours (maximum 2.69 hours, minimum 0.63 hours). File size reduction decreased the average BAM-file size from 58.9GB (maximum 123GB, minimum 28GB) to an average of 7.7GB (maximum 15GB, minimum 3.8GB). This gives an average imput size reduction of 86.9% (maximum 93.9%, minimum 84.5%). Figures 4.2 and 4.3 visualize this. The first one shows the file size before reduction in GB on the red line and the size for the corresponding file after reduction in GB on the green line and the second one shows a histogram of the size reduction percentages.

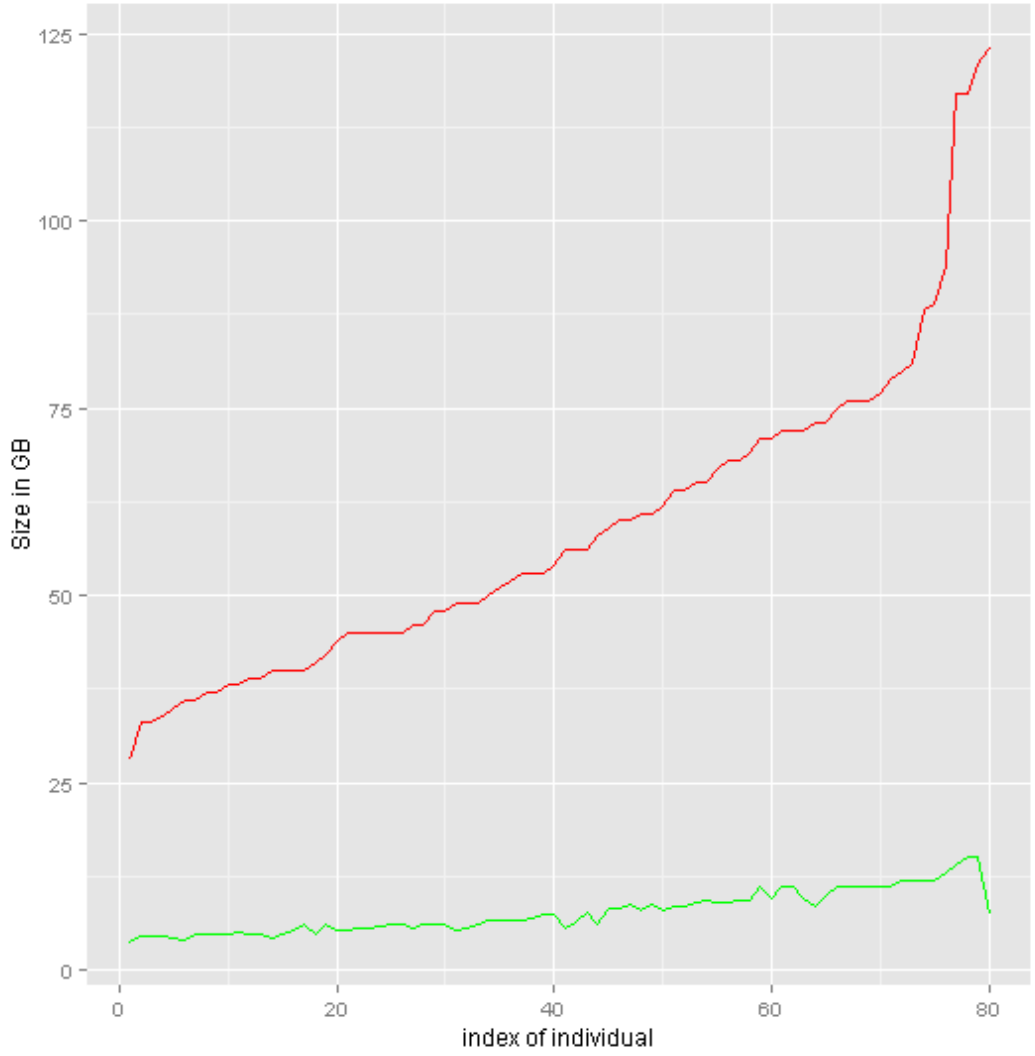

Figure 4.2: The red line shows file sizes before filtering and the green one after filtering for the corresponding files.

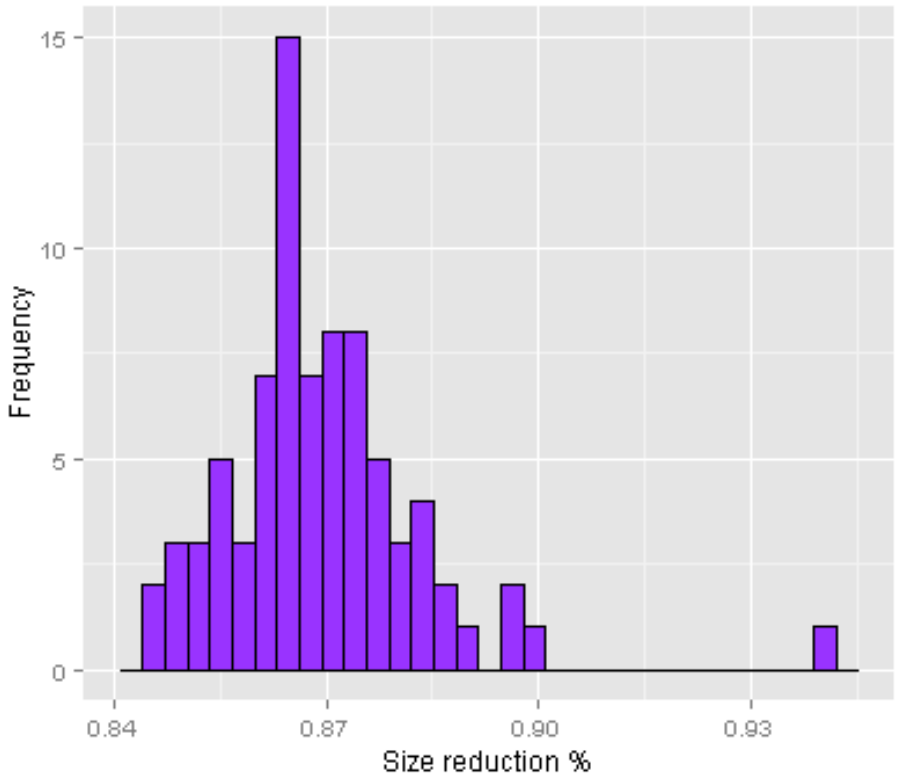

Figure 4.3: Histogram of size reduction in percentages

We run the lobSTR sensing and alignment step on both the filtered and original BAM-files. Because it would have been very computationally expensive to run the program using the larger BED-file and index the smaller one was chosen.

Effects on size of generated files. We estimate the effect that the filtering has on the BAM-files generated in the sensing and alignment step. We do it by computing the size ratio between a file generated using the whole BAM-file of an individual and the filtered BAM-file of the individual for all 80 individuals. This shows an average size reduction of 5.4% which indicates that most of the reads identified by lobSTR as microsatellite-containing were present in the filtered input but not all. Figure 4.4 shows the size of the generated BAM-files, red line representing the size when using the original BAM-files as input and the green line the size when using the reduced input. The four drops (at individuals: $\approx 17, 44, 72$ and 80) observable in the green line were examined further and the BAM-files of these individuals turned out to have an unusually high content of low quality and duplicate reads which the filter removed but were still present in the output from the unfiltered BAM-files.

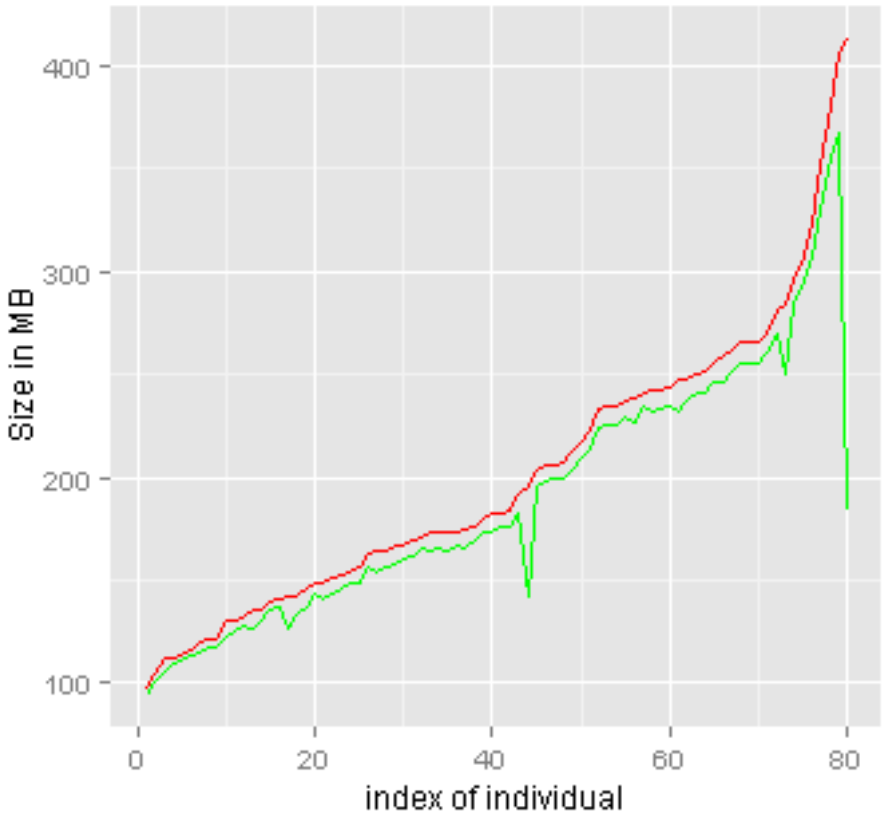

Figure 4.4: Red line shows output size from original files and green line from filtered files.

Effects on running time. We consider the running time and for the original BAM-files the average was 119.5 hours (maximum 253.3 hours, minimum 60 hours) while the average was 30.1 hours (maximum 55 hours, minimum 16.5 hours) for the filtered BAM-files. This gives an average of 74.4% reduction in running time (maximum 88.5%, minimum 69%), this is visualized in Figure 4.5 where the running time for the original BAM-files is shown on the red line and the running time for the same BAM-files after size reduction is shown on the green line.

Effects on number of profiled microsatellites. Since the desired result is of course a microsatellite profile for each of the 80 individuals in the form of a VCF-file the next step is to pass the two versions of BAM-files generated in the first step to the second step of lobSTR which performs the genotyping. Here the input sizes between the BAM-files generated from the filtered input and the unfiltered input do not differ enough to influence the running time so it is not documented. We compare the resulting VCF-files, on average the BAM-file generated from the reduced input returned genotypes for 155400 microsatellites (maximum 169126, minimum 116051) while the BAM-file generated from

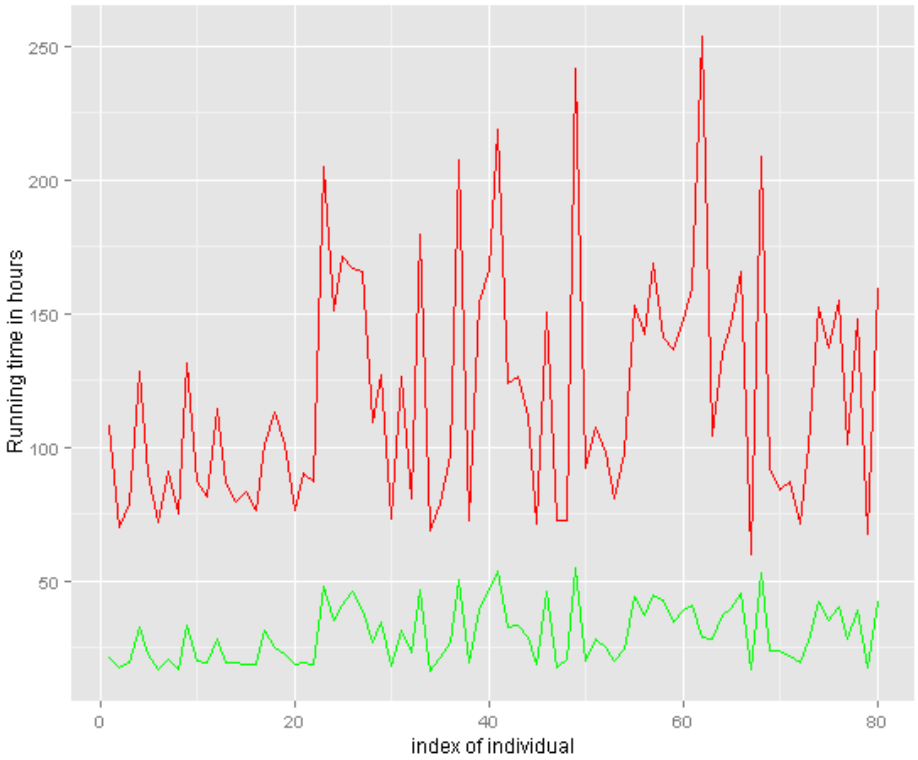

Figure 4.5: Red line shows running time for original files and green line for filtered files.

the original input returned genotypes for 158084 microsatellites on average (maximum 172145, minimum 117986). On average the number of microsatellites genotyped using the reduced input was 98.3% of the number of microsatellites genotyped using the original input. Figure 4.6 shows this comparison per individual. Of the microsatellites found in both cases the VCF files agreed on 99.4% of them on average. For a total comparison the smaller input found and agreed with 97.8% of the microsatellites from the bigger input on average.

Analyzing differences in results. To better understand the difference in number of genotyped microsatellites we analyze both versions of BAM files generated in the sensing and alignment step, i.e the one generated using the filtered input and using the unfiltered input. In both files we check the coverage of every microsatellite genotyped in the VCF-file generated using the unfiltered BAM-file as input. The results were that on average the BAM-file generated in the sensing and alignment step using the reduced input covered 98,4% of the loci in the VCF file while the one using the original input covered all of them. To see if the microsatellites were covered by a similar number of reads in each file this ratio between covering reads was computed for all microsatellites in all individuals. The left side of Figure 4.7 shows a histogram of this ratio for all covered microsatellites

Figure 4.6: Red line shows number of microsatellites genotypes for original files and green line for filtered files.

in a sample individual.

To explain any difference in coverage ratio both BAM-files generated in the sensing and alignment step were analyzed further. The reads only present in the file generated using the original BAM-file as input were identified. These were located in the original BAM-file, checked for properties indicating low quality reads and optical duplicates and removed if such were present. The coverage ratio between microsatellites in both files generated in the first step was then computed again. The right side of Figure 4.7 shows the results of this. Close to 100% of the covered microsatellites now have a coverage ratio of 1 between the files generated using the original and reduced BAM-files. This indicates that the difference in results was due to low quality reads present in the files generated using the unfiltered input but not in the ones generated using the filtered input and that the filtering had no effect on the resulting genotype for almost all loci. This procedure also raised the average size ratio between the BAM-files generated by the first step from 94.6% to 97%

Last, we examine was why some microsatellites were genotyped differently when using the reduced BAM-file as input and the original BAM-file as input. The average number of microsatellites the VCF-files disagreed was 847 microsatellites out of the 158084 microsatellites genotyped on average. This is an average ratio of 0.5%. We looked at the reads covering these locations that are only present in the BAM-file resulting from

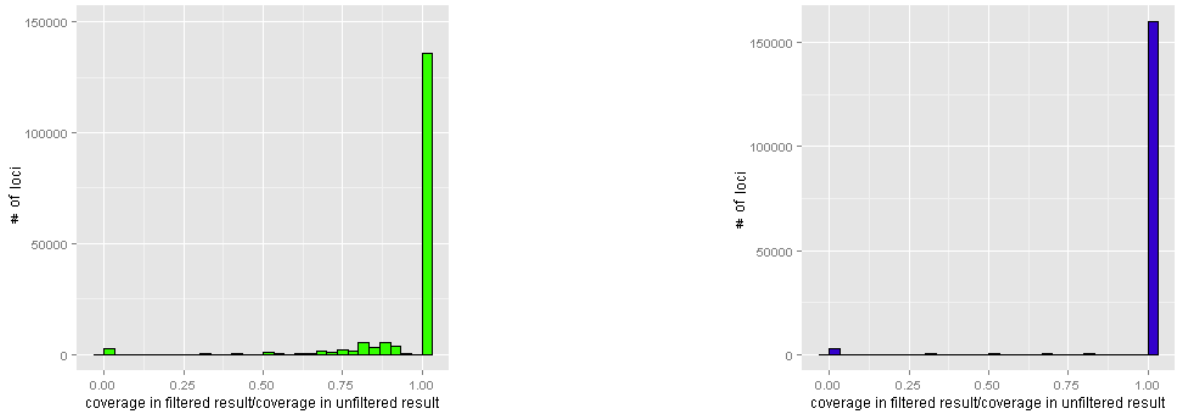

Figure 4.7: Coverage ratio between BAM-files, original on left and on right when low quality and duplicate reads have been removed.

running the first step using the original BAM-file as input. We went further on to locate these reads in the original BAM file and checked their quality and location to understand why they are not a part of the filter's output. On average 12% of the reads covering these locations only in the output generated using the original files as input were discovered to either have low quality or were PCR optical duplicates. It is possible that this contributed to the difference in results.

Chapter 5

Modelling error

Here we will start by giving a detailed definition of the error model provided with lobSTR which we will use for comparison with our error model. Possible problems with the model will be discussed. Next we give a step by step description of the development of our model. Last, results from both gentype callers will be presented and compared to each other and the benchmark data. Our results improve the genotyping accuracy from 87.5% to 96.3%.

5.1 lobSTR error model

5.1.1 Definition

The error model from lobSTR is trained using data aligned to sex chromosomes on male samples since they are hemizygous, meaning that a male individual has only one copy of each sex chromosome.

The model is generative and considers two aspects of stutter noise, the first one addresses the probability s_j that a read covering a microsatellite j is a result of stutter noise. The second one expresses the distribution of error lengths, $D(e)$, i.e. the probability of error length e given that there is stutter noise. Four characteristics are used to predict s_j for each microsatellite j. These are the length of the repeat motif, the STR region length (which refers to the length of the entire region spanned by the microsatellite), the GC content of the microsatellite's flanking regions and STR purity. STR purity is a score that measures the purity of the STR sequence and is based on the suggested Tandem Repeats Finder scoring scheme with match=2, mismatch=-7 and indel=-7. Therefore the maximum possible score for a perfectly pure STR sequence (e.g. ATATATATATAT) is $2 \cdot$ (length of STR region). These characteristics are modelled as having the same effect on stutter probability in every location, i.e. there is only one model for all microsatellites.

To compute s_j for a read covering microsatellite j the model plugs in the value of these four attributes at that microsatellite.

Stutter noise where the length is a multiple of the repeat motif length, is modelled as being Poisson distributed. Stutter noise creating non unit errors is modelled to have a geometric distribution with $p = \frac{1}{\bar{x}+1}$ where $\bar{x}$ is the average error modulo the repeat motif length. Stutter noise tends to delete repeat units rather than increase their number and because of this an extra parameter q is added to the model representing the probability that stutter will increase the true allele length. The final expression for the probability of error length e given a repeat motif length of m then becomes

$$D(e, m) = Pois(\lambda = m) \cdot Z \cdot Geom\left(p = \frac{1}{\bar{x}+1}\right) \cdot Y \tag{5.1}$$

where Z is an indicator variable that is 1 if $e \neq 0$, else 0 and Y is another indicator variable equal to q if $e > 0$, else $1 - q$.

This model computes the probability, $P(L|A, m)$, of generating a read with a microsatellite of length L when the true microsatellite length is A in a hemizygous locus, i.e. only one chromosome copy is present, with a repeat motif length of m. For a microsatellite locus where two copies of a chromosome are available and the true lengths are A and B the probability of this is $P(L|A, B, m) = \frac{1}{2} \cdot (P(L|A, m) + P(L|B, m))$ since each read has equal possibility of coming from either allele. Since all PCR rounds are independent, the probabilities of stutter noise between reads are also independent and we can multiply them together to compute the likelihood of getting this set of reads given a specific genotype. This kind of independence allows the usage of log-likelihood, the sum of logs of probabilities for all reads covering the same microsatellite is maximized with respect to A and B in order to determine the alleles present in the individual being genotyped. Along with the score, lobSTR requires 20% of reads to support an alternative allele for a location to be classified as heterozygous and 50% of the reads must support the resulting alleles [5].

5.1.2 Possible problems

Because the model was trained only on microsatellites of male samples on the sex chromosomes it is not able to capture per location error, i.e. the four characteristics used to predict the probability of stutter noise at each microsatellite location might not have the same effect on other chromosomes. As we have data available for a large number of individuals we develop and apply a more complex estimation of the slippage rate at each microsatellite. Furthermore, the quality of alignment is not considered in the classification but this quality heavily contributes to how reliable a read is for genotyping. In this

model all reads are considered real or a result of stutter noise. Thus the only type of error accounted for is stutter noise created due to slippage events during sample preparation. Other types of sequencing errors or misplacement of reads are not considered.

5.2 Improved error model

We start with an error model that uses a simple heuristic to estimate the rate of slippage events on a per microsatellite basis. The first update is to add a logistic regression classification to the model. This accounts for the fact that slippage events are not the only possible type of errors and the regression is trained to identify other types of errors. Next we update our estimate of the slippage rate to include the contribution of the individual by estimating a slippage rate for each individual. We then model the slippage in each case to be the sum of the individual and microsatellite contributions. Because this can lead to overestimation of slippage rates we update the estimate by weighing contributions of individuals and microsatellites with their variance to reduce the effect of noisy data in the final estimates. The method used for determining the genotype is refined twice. First by increasing the tolerance for error reads in the genotype and second by considering the fact that slippage events are more likely to delete repeat units than to increase their number.

5.2.1 Stepwise improvements

Estimating slippage rates for each microsatellite. The frequency of slippage errors that cause stutter noise varies between microsatellites. To account for this we estimate a specific slippage rate for each location using the training data. This gives an accurate estimation since it includes the characteristics of each microsatellite instead of using the same expression for all of them. We estimate the slippage rate at microsatellite i by dividing the number reads at the microsatellite marked as error reads by the total number of reads aligned to the microsatellite. The reads get their initial labelling as true or error reads in the pre-processing of the data and the labelling is iteratively updated during the estimation process. The following expression is used to estimate S_i^M, the slippage rate at microsatellite i

$$S_i^M = \frac{\sum n_i^e}{\sum n_i} \tag{5.2}$$

where n_i^e stands for the number of reads at microsatellite i marked as error reads and n_i stands for the total number of reads aligned to microsatellite i. The distribution of this slippage rate per location is shown in Figure 5.3 in Section 5.3.2. This expression however ignores the contribution of the individual to the slippage rate and considers only the microsatellite contribution.

Adding logistic regression. To account for errors, other than the ones caused by stutter noise, we train a logistic regression classifier for each microsatellite. We train it using only reads reporting true alleles and error reads that are not a result of stutter noise. This enables us to give each read a probability of reporting a true allele. The probability is computed by classifying the read using the classifier trained for the microsatellite it is aligned to. By including this probability in the expression used to determine the genotype we can also identify error reads caused by undefined errors. The attributes used by the classifier are various but all have in common their ability to indicate the quality of a read in one way or the other. We compute a number of attributes and consider all of them but an optimum subset is selected using feature selection. The attributes considered are summarized in Table 5.1 and the process used for feature selection is described in Section 5.2.3

Estimating slippage rates for each microsatellite and each individual. To increase the accuracy of our slippage rate estimates, we update the expression of the estimate to contain the sum of slippage that is caused by the microsatellite location and the slippage that is caused by the individual. A system of equations is used to estimate the slippage rate. Here, S_{ij} represents the slippage rate for the pair of microsatellite i and individual j, S_i^M represents the slippage contributed by microsatellite i and S_j^P represents the slippage that individual j contributes.

$$S_{ij} = S_i^M + S_j^P \tag{5.3}$$

We denote the probability that a read aligned to microsatellite i, marked as an error read is actually reporting a true allele as p_i^e. The probability that a read aligned to microsatellite i, regardless of how it is marked, reports a true allele is represented by p_i. For reads grouped by individuals we let p_{ij} denote the probability of reporting a true allele for a read aligned to microsatellite i from individual j. To estmate the slippage rate at microsatellite i we then use the following expression

$$S_i^M = \frac{\sum p_i^e}{\sum p_i} - \sum_j S_j^P \cdot \frac{\sum p_{ij}}{\sum p_i} \tag{5.4}$$

In the expression used for estimating the slippage of individual j we let p_j^e denote the probability that an error read from individual j reports a true allele and p_j denote the probability that a read from individual j reports a true allele, regardless of how it is

marked. So the expression becomes

$$S_j^P = \frac{\sum p_j^e}{\sum p_j} - \sum_i S_i^M \cdot \frac{\sum p_{ij}}{p_j} \tag{5.5}$$

The equations depend on each other and are updated iteratively using a basic initialization. The updating process is described in Section 5.2.3.

Although this estimate improves the previous expression it however allows for very high slippage rate estimates in some cases since all individuals contribute equally to the slippage at each position and vice versa without regard to their variance. Thus if an individual or a microsatellite has a high variance they will cause overestimation of the slippage rate.

Weighing contributions of individuals and microsatellites with their variance To decrease the weight of individuals and microsatellites with noisy reads in the slippage rate estimates we update the estimate once more and weigh the contributions of each individual and microsatellite with the inverse of their variance. This gives microsatellites and individuals with low variance higher weights in the estimate and should thus make it more stable and reliable. The slippage rate at a microsatellite essentially represents the probability of getting an slippage error read at the given microsatellite and it is a binary distributed variable. We define the variance of a microsatellite as its slippage rate and use the definition of a variance for one trial on a binary distributed variable, given by

$$v = p(1-p) \tag{5.6}$$

After this update, the expression for the slippage rate at microsatellite i becomes

$$S_i^M = \frac{\sum p_i^e}{\sum p_i} - \sum_j S_j^P \cdot \frac{\frac{1}{S_j^P(1-S_j^P)} \cdot \frac{\sum p_{ij}}{\sum p_i}}{\sum_j \frac{1}{S_j^P(1-S_j^P)} \cdot \frac{\sum p_{ij}}{\sum p_i}} \tag{5.7}$$

and similarly the slippage rate for individual j becomes

$$S_j^P = \frac{\sum p_j^e}{\sum p_j} - \sum_i S_i^M \cdot \frac{\frac{1}{S_i^M(1-S_i^M)} \cdot \frac{\sum p_{ij}}{\sum p_j}}{\sum_i \frac{1}{S_i^M(1-S_i^M)} \cdot \frac{\sum p_{ij}}{\sum p_j}} \tag{5.8}$$

Using these estimates we include the overall slippage rate of the individual or microsatellite being considered along with the contribution of each microsatellite to the individual and the contribution of each individual to the microsatellite. These contributions are however weighed with the inverse of their variance so noisy individuals or microsatellites are not

able to skew the estimates. We have now covered how we intend to use logistic regression and estimated slippage rates to identify both errors causing stutter noise and undefined errors. Next we cover how these are used to determine the genotypes.

5.2.2 Determining the genotype

Having trained a logistic classifier and estimated the slippage rates at all microsatellites and for all individuals we now have all that we need to determine the genotypes. To do this for individual j at microsatellite i we start by looking at all reads for the given individual aligned to the given microsatellite and determine the possible genotypes they offer. We then compute the likelihood of the reads given each genotype and pick the genotype that maximizes this likelihood. The expression used to compute this likelihood for a set of reads $R = \{r_1, \ldots, r_n\}$ and a genotype gt is

$$P(\mathrm{R}|\mathrm{gt}) = \prod_s p(r_s|gt) \tag{5.9}$$

Like the expressions for the slippage rate estimates, the formula used to compute $p(r_s|gt)$ was updated several times during the creation of the model. Our first version assumes a Poisson distribution of the stutter noise error events with $\lambda = S_{ij}$ and includes the probability of reporting a true allele given to every read by the logistic regression classifier of the microsatellite it is aligned to. We let $p(r_s^t)$ denote the probability that read s reports a true allele and x stand for number of slippage events. The number of slippage events is defined as the difference between the number of repeats reported by the read and the number of repeats in the genotype. This gives the following expression for a homozygous genotype A.

$$p(r_s|A) = p(r_s^t) \cdot pois(x; S_{ij}) \tag{5.10}$$

and for a heterozygous genotype (A, B) we compute the number of slippage events relative to both alleles in the genotype.

$$p(r_s|A, B) = p(r_s^t) \cdot (\frac{1}{2} \cdot pois(x(A); S_{ij}) + \frac{1}{2} \cdot pois(x(B); S_{ij})) \tag{5.11}$$

The problem with this expression is however that it becomes very small for reads that are error reads not due to slippage events. For non-slippage errors our model assumes that each of the other reported alleles is equally likely and so we set $n^i_{alleles}$ as the number of alleles present in the population for microsatellite i and update the expression for $p(r_s|gt)$ in the following way

$$p(r_s|A) = p(r_s^t) \cdot pois(x; S_{ij}) + \frac{1 - p(r_s^t)}{n^i_{alleles}} \tag{5.12}$$

and in the same way for a heterozygote genotype

$$p(r_s|A, B) = p(r_s^t) \cdot (\frac{1}{2} \cdot pois(x(A); S_{ij}) + \frac{1}{2} \cdot pois(x(B); S_{ij})) + \frac{1 - p(r_s^t)}{n^i_{alleles}} \tag{5.13}$$

Using this expression to determine the probability of a read given a genotype we assume that stutter noise is equally likely to result in reads with fewer repeat units and reads with extra repeat units. This is however not the case as stutter noise is more likely to delete repeat units and to account for this we add a parameter, a, with the same purpose as q in the lobSTR model. The probability of deleting a repeat unit is set as 0.85 and of adding a repeat unit as 0.15 which means we set a as 0.85 if allele$_i$ $< A$ and 0.15 if allele$_i$ $> A$ The final expression derived from this thus becomes

$$p(r_s|A) = p(r_s^t) \cdot pois(x; S_{ij}) \cdot a + \frac{1 - p(r_s^t)}{n^i_{alleles}} \tag{5.14}$$

and in the same way for a heterozygote genotype but there we need a_1 and a_2 to compare allele$_i$ with both alleles of the genotype, A and B

$$p(r_s|A, B) = p(r_s^t) \cdot (\frac{1}{2} \cdot pois(x(A); S_{ij}) \cdot a_1 + \frac{1}{2} \cdot pois(x(B); S_{ij}) \cdot a_2) + \frac{1 - p(r_s^t)}{n^i_{alleles}} \tag{5.15}$$

This final expression combines the power obtained both through the logistic regression classifier and the estimation of slippage rates at microsatellites and individuals. By adding the probability that a read is an error to the number it contributes to the likelihood we raise the tolerance for error reads. This makes it possible to call a homozygous genotype despite having perhaps a number of reads supporting another allele if these other reads get poor results from the logistic regression classifier.

5.2.3 Implementation

Expectation maximization. We use expectation maximization to simultaneously train a logistic regression classifier and estimate the rate of slippage errors for each microsatellite. In the E-step we use the current genotypes to train a classifier and estimate slippage rates. In the M-step we use the probabilities of reporting a true read, given to each read by the logistic regression classifier of the microsatellite it is aligned to, and the estimated slippage rates to determine and update the genotypes. This is iterated until convergence has been reached. Then we should have both an optimally trained classifier and optimal estimates of slippage rates for each microsatellite. The condition chosen for convergence at each microsatellite is that when one or no individuals get an updated genotype convergence has been reached. We then use the estimated slippage rates for each marker to re-estimate slippage rates for each individual and then we repeat the training of logistic regression classifiers and estimations of slippage rates per marker using these new individual slippage rates. This estimation of slippage rates for each individual is performed every time after the training of logistic regression classifiers and estimations of microsatellite slippage rates. If the sum of squared differences between the individual slippage rates in the previous estimation and the current estimation divided by the number of individuals is less than $1 \cdot 10^{-6}$ then we consider convergence to be reached.

We initialize the probability of reporting a true allele as 95% for all reads and the individual slippage rates were initialized as

$$S_j^p(initial) = \frac{\sum p_j^e}{2 \cdot \sum p_j} \tag{5.16}$$

where p_j^e stands for the probability that a read from individual j marked as an error read reports a true allele and p_j denotes the probability that a read from individual j reports a true allele, regardless of how it is marked.

Feature selection for logistic regression. We compute a vector of 9 attributes for every read and use a subset of it, chosen using feature selection, for classification by logistic regression. The first attribute is the quality score given to the alignment of the read to the reference. This gives an idea of how well the flanking areas of the microsatellite sequence match the reference flanking areas of the microsatellite it was aligned to and is therefore highly informative. Alignment scores are deduced from the scoring scheme of the alignment program used and can be set to take into account the base quality in the read. Since the quality score considered here only refers to the alignment quality of the microsatellite's flanking regions then extra (or missing) repeat motifs do not influence it. This means that it does not matter whether the individual being profiled has a different

allele from the reference. Next we consider how clean the microsatellite sequence extracted from the read is, i.e. how purely repetitive the extracted sequence is. This is computed by dividing the actual number of repeats in the sequence by the expected number of repeats so for a completely clean sequence this value would be one. For example a microsatellite sequence where the repeat motif is 'AC' and is 12 basepairs long is expected to have six repeats of 'AC'. However the sequence 'ACACATACAC' has only a cleanness of $5/6 \approx 0.833$.

The distance between the alignment location of the read and the alignment location of its mate is also considered as an influential factor. The distance between paired-end reads follows a tightly bound distribution and if the distance between the alignment of read pairs deviates significantly from this distribution it indicates that the alignment is wrong.

The next three attributes are the number of bases with PHRED values, defined in Section 2.2, above 20 in the part of the read coming before the microsatellite sequence, in the microsatellite sequence itself and in the part of the read coming after the microsatellite sequence. The portion of bases with a PHRED value above 20 in the mate of the read is also computed. The limit of 20 was chosen since since a PHRED score of 20 means that the probability of the base being a sequencing error is 1%.

When reads are aligned to a reference genome they don't always match completely, this could be due to genetic variations, misplacement of reads or sequencing errors for example. The edit distance from the read to the reference is the smallest number of substitutions, insertions, and deletions of bases that can be used to transform the read to the reference. This edit distance of the reads mate is included in the attribute vector since a high edit distance might indicate a bad alignment. The last attribute is the total length of the sequence in the read, all reads in the data used here should be either 101 or 120 bases long and if a read is much shorter than this it indicates that something might have gone wrong during sequencing.

It is possible that some attributes don't have power to predict the quality of a read. These attributes could harm the classification by slowing it down and adding noise to attributes with predictive power. We apply a form of feature selection to select attributes to include in the classifier. First, we train a logistic regression classifier for all microsatellites with all nine attributes. Next, we count for each attribute the number of microsatellites where it was relevant, i.e. its p-value in the logistic regression model was significant. We train a logistic regression classifier with only the most frequently relevant attribute and perform genotyping. The second most frequently relevant attribute is then added to the training

Attribute	Definition
Quality score	Map quality score of the aligned read.
Cleanness	# of repeats in the microsatellite sequence/ # of expected repeats in the microsatellite sequence
Distance to mate	Distance from a reads alignment location to the alignment location of the mate read.
Number over 20 before	The number of bases in the read with a PHRED quality over 20 coming before the microsatellite sequence.
Number over 20 in	The number of bases in the read with a PHRED quality over 20 in the microsatellite sequence.
Number over 20 after	The number of bases in the read with a PHRED quality over 20 coming after the microsatellite sequence.
Portion of mate over 20	The portion of bases in the sequence of the mate read with a PHRED quality over 20.
Edit distance of mate	Edit distance to the reference, including ambiguous bases but excluding clipping.
Sequence length	Total length of the read-sequence.

Table 5.1: The attributes used as control variables in the Logistic regression classification.

process and genotyping is performed again. This is repeated until the genotyping accuracy stops increasing between rounds of adding attributes. when the accuracy stops increasing an optimum subset of attributes for classification has been found.

5.3 Results

We present the results by a pair-wise comparison between the lobSTR error model, updated error model and the benchmarking data. The genotypes compared with the benchmarking data are conditioned to be inferred from at least 12 reads for both the lobSTR error model and our error model. This is done to ensure a level of confidence in the genotypes compared. There is a considerable probability of allelic dropout or severe under-representation of one allele in a heterozygous genotype for decisions made using fewer reads than this.

5.3.1 Bechmark data vs. lobSTR error model

The genotypes returned by the error model provided by lobSTR agreed with the benchmarking data in 87.5% of the cases when considering only genotypes determined from 12 or more reads. The accuracy per location is shown in Figure 5.1.

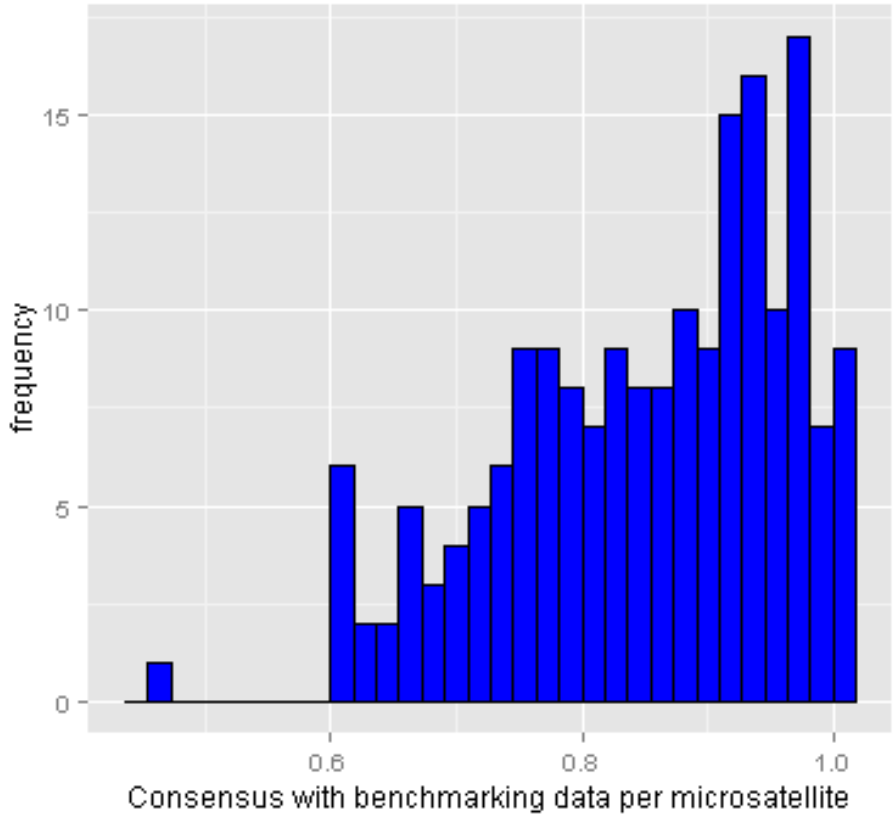

Figure 5.1: Histogram of lobSTR genotyping accuracy per location.

This low consensus could be explained a number of things. The fact that the error model was trained on different data is a very likely cause. The model does not account for any other type of error than stutter noise and this might decrease its predictive power. It's generality, i.e. using the same model for all microsatellites, might also hurt the results and no attempts were made to tune the runtime parameters of the microsatellite profiler and thus it is possible that the accuracy might increase for parameters other than the default ones. Also, these numbers do not tell the entire story since the accuracy of genotyping of course depends heavily on how many reads we infer the genotype from. Figure 5.2 shows clearly how the consensus with the benchmarking data increases as the number of reads available increases. As the number of available reads reaches ≈ 29 the curve levels off at 100% accuracy.

Figure 5.2: The consensus with benchmarking data as a function of the number of available reads.

5.3.2 Benchmark data vs. Updated error model

Here we cover the results of each improvement in the order they were added to show how the development increased the accuracy of genotyping.

The first version determines the genotype using microsatellite specific slippage rate. Next a probability of reporting a true allele is computed for each read using a microsatellite specific logistic regression classifier. To account for the possibility that a read is an error caused by something else than slippage and should not influence the genotype we add to the likelihood contribution of each read the probability that it reports a false allele divided by the total number of alleles in the population. Because the slippage rate in each case depends not only on the microsatellite but on the individual as well we update the slippage estimate to include this. To avoid overestimating the slippage rates we weigh the contributions of each microsatellite and each individual with the inverse of their variance to minimize the effect of noisy individuals and microsatellites. Last we add a parameter expressing the fact that slippage events during replication are more likely to delete repeat motifs than add them.

Estimating slippage rates for each microsatellite. Figure 5.3 shows the distribution of the slippage rates estimated using the initial expression for slippage rates shown in Equation 5.2. Figure 5.4 shows a histogram of the per microsatellite consensus with the benchmarking data and Table 5.2 summarizes the improvement made.

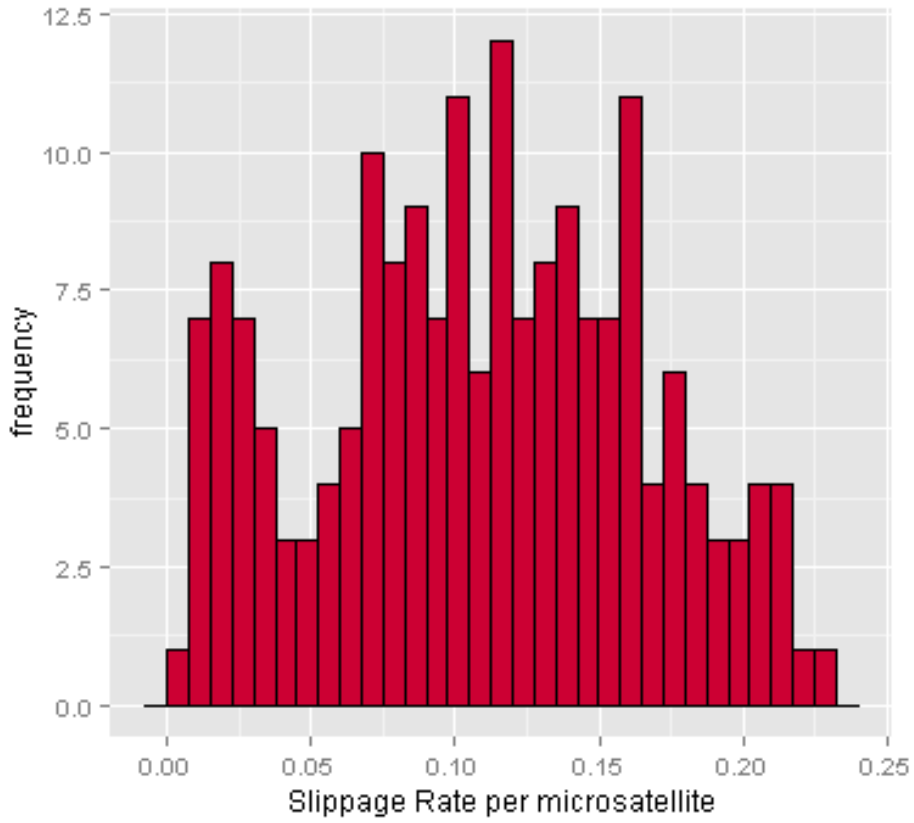

Figure 5.3: The distribution of slippage rates per microsatellite using initial estimate.

Improvement	**Genotyping accuracy**
Estimating microsatellite specific slippage rates	91.8%

Table 5.2: Total genotyping accuracy when estimating a slippage rate for each microsatellite.

Adding logistic regression. Figure 5.5 shows a histogram of the per microsatellite consensus with the benchmarking data and Table 5.3 summarizes the improvement made.

Improvement	**Genotyping accuracy**
Applying logistic regression	91.8%

Table 5.3: Total genotyping accuracy when adding logistic regression to identify undefined errors.

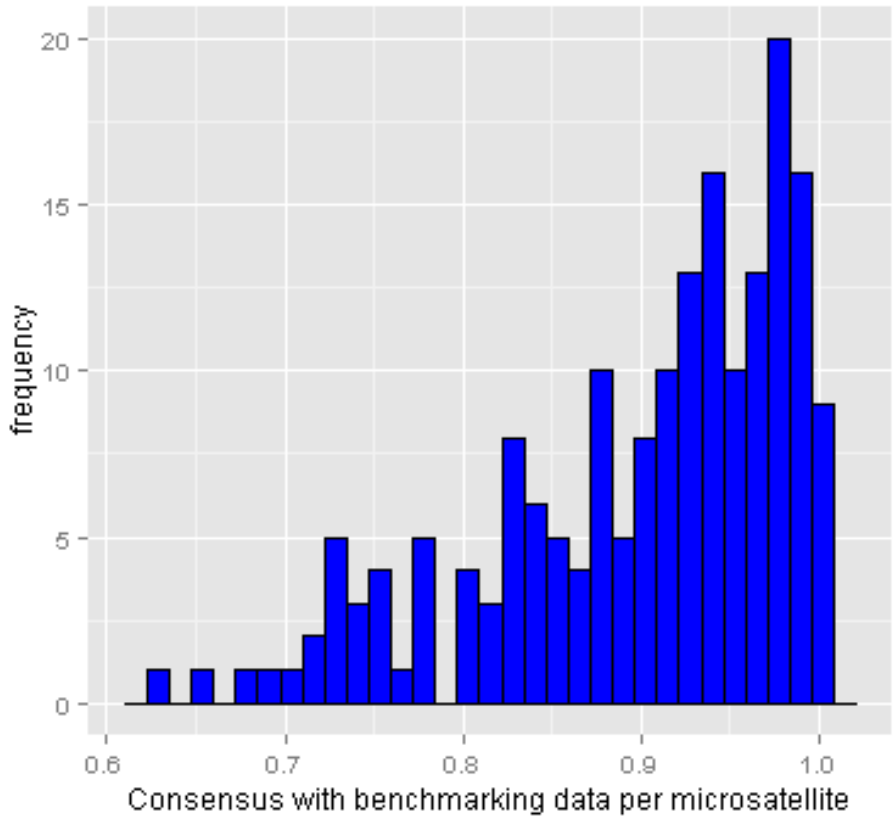

Figure 5.4: Accuracy of genotyping per location when using initial slippage rate estimates.

Adding probability of being an error read divided by number of present alleles to likelihood contribution. Figure 5.6 shows a histogram of the per microsatellite consensus with the benchmarking data per microsatellite and Table 5.4 summarizes the improvement made.

Improvement	Genotyping accuracy
Adding the probability of being an error read divided by the number of alleles in the population to likelihood contribution	93%

Table 5.4: Total genotyping accuracy when adding the probability of being an error read divided by the number of alleles in the population to likelihood contribution.

Estimating slippage rates by considering interaction between microsatellite and individual. Figures 5.7 and 5.8 show the distributions of the slippage rates estimated using the expressions given in Equations 5.4 and 5.5. They consider the

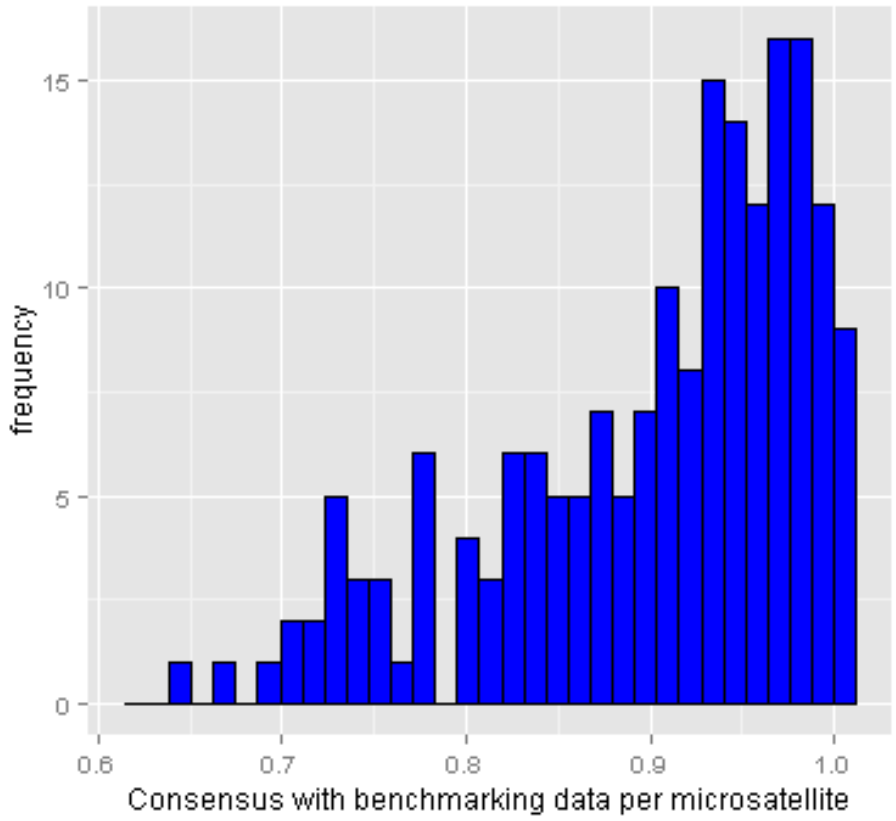

Figure 5.5: Accuracy of genotyping per location when adding logistic regression.

interaction between the microsatellite slippage and the individual slippage. Figure 5.9 shows a shows a histogram of the per microsatellite consensus with the benchmarking data and Table 5.5 summarizes the improvement made.

Improvement	**Genotyping accuracy**
Considering interaction between microsatellite and individual when estimating slippage rates	92.8%

Table 5.5: Total genotyping accuracy when adding the interaction between microsatellite and individual to slippage rate estimates.

Weighing contributions of individuals and microsatellites with their variance. Figures 5.10 and 5.11 show the distributions of the slippage rates estimated when using the expressions in Equations 5.7 and 5.8. They consider the interaction between the microsatellite slippage and the individual slippage and weigh each contribution with the

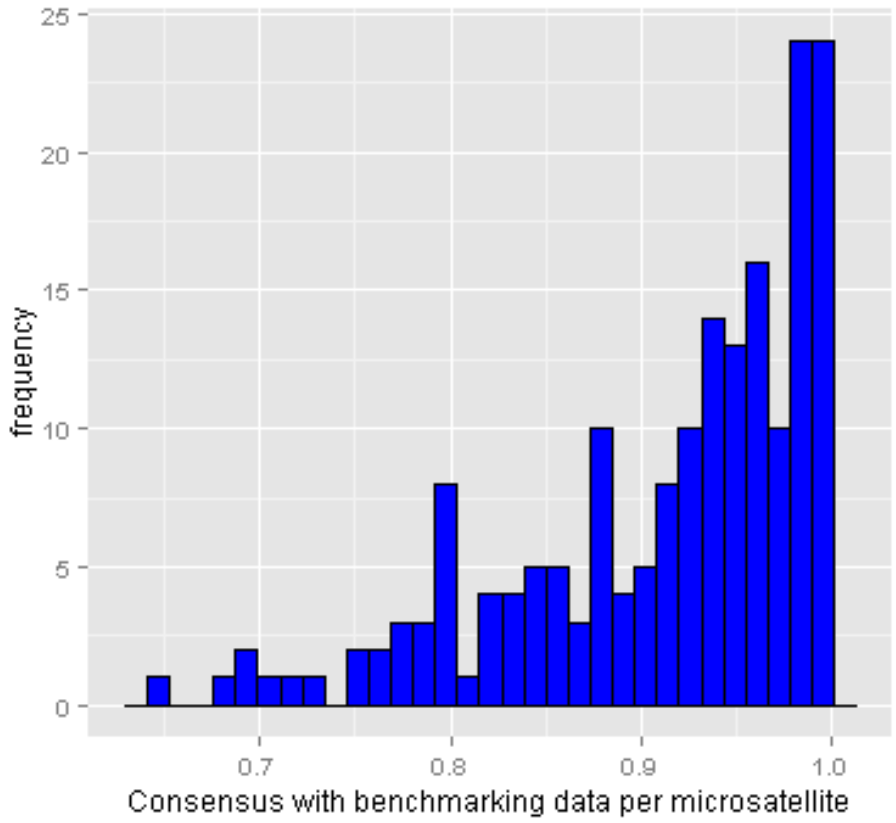

Figure 5.6: Accuracy of genotyping per location when adding probability of being a non-slippage error read.

inverse of its variance. Figure 5.12 shows a shows a histogram of the per microsatellite consensus with the benchmarking data and Table 5.6 summarizes the improvement made.

Improvement	**Genotyping accuracy**
Considering the contribution of each individual and each microsatellite and weighing it with the inverse of its variance	93%

Table 5.6: Total genotyping accuracy when adding the interaction between microsatellite and individual to slippage rate estimates and weighing contributions with the inverse of their variance.

Correcting for the fact that stutter noise tends to delete repeats Figure 5.13 shows a histogram of the consensus with the benchmarking data per microsatellite when adding a parameter expressing that stutter noise is more likely to delete repeat units than add them. Table 5.7 summarizes the improvement made.

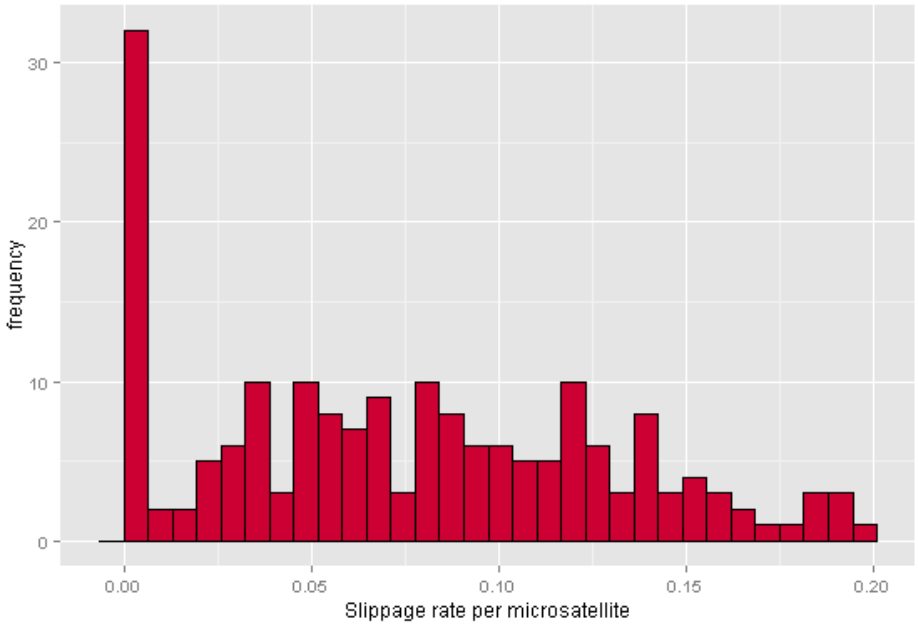

Figure 5.7: Distribution of slippage rates per microsatellite when considering the contribution of each individual.

Improvement	**Genotyping accuracy**
Adding parameter for probability of deleting or adding repeat units	96.3%

Table 5.7: Total genotyping accuracy when adding parameter for probability of deleting or adding repeat units.

As mentioned before the number of reads aligned to a microsatellite heavily influences the accuracy of the genotype assigned to it. Figure 5.14 shows how the consensus with the benchmarking data increases as the number of reads available to perform genotyping increases. As the number of reads reaches ≈ 20 the accuracy reaches 100% and levels off.

To examine the effect of the number of reads used for genotyping on a per microsatellite basis we plot the genotyping accuracy for all microsatellites as a function of the median of the available reads at that microsatellite. This is shown in Figure 5.15. The trend here is not as clear as in 5.14 but it is evident that as the median of available reads at a microsatellite increases, so does the genotyping accuracy. The slippage rate varies between microsatellites. We examine how the genotyping accuracy develops as a function of the slippage rate at the microsatellite. This is shown in Figure 5.16 and like one would expect the accuracy decreases as the slippage rate increases since more error reads make

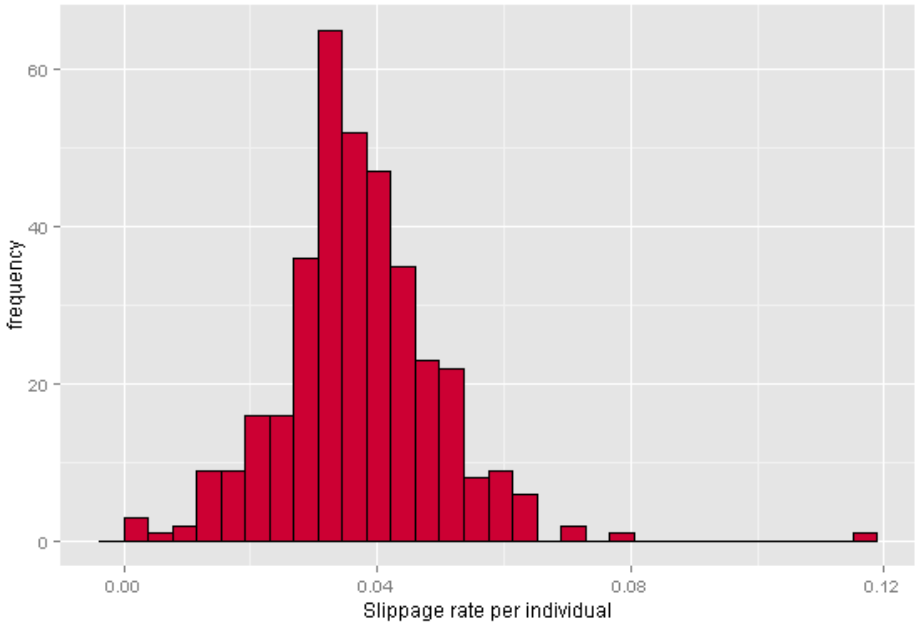

Figure 5.8: Distribution of slippage rates per individual when considering the contribution of each microsatellite.

it harder to infer the right genotype.

Allelic dropout occurs when a DNA sample is sequenced and one or more alleles are not present in the resulting sequencing data. Assuming that each read has equal probability of coming from either allele and that we need at least two reads representing an allele to have a chance of identifying it then the probability of allelic dropout in a set of n reads can be computed by

$$p(\text{allelic dropout}) = \frac{1}{2^n} \cdot \left(\binom{n}{0} + \binom{n}{1} + \binom{n}{(n-1)} + \binom{n}{n} \right) = \frac{1}{2^n} \cdot \left(\frac{n!}{0!(n-0)!} + \frac{n!}{1!(n-1)!} + \frac{n!}{(n-1)!(n-(n-1))!} + \frac{n!}{n!(n-n)!} \right) = \frac{1}{2^n} \cdot (2n+2) \quad (5.17)$$

Because we condition the genotypes we compare to the benchmarking data to be determined from at least 12 reads the possibility of allelic dropout is barely high enough to be considered. Using the expression in Equation 5.17 we can compute the expected total number of allelic dropout cases by

$$\mathbb{E}(\#ad) = \sum_{i=1}^{i=maxCoverage} \#cases_n \cdot \frac{1}{2^n} \cdot (2n+2) \quad (5.18)$$

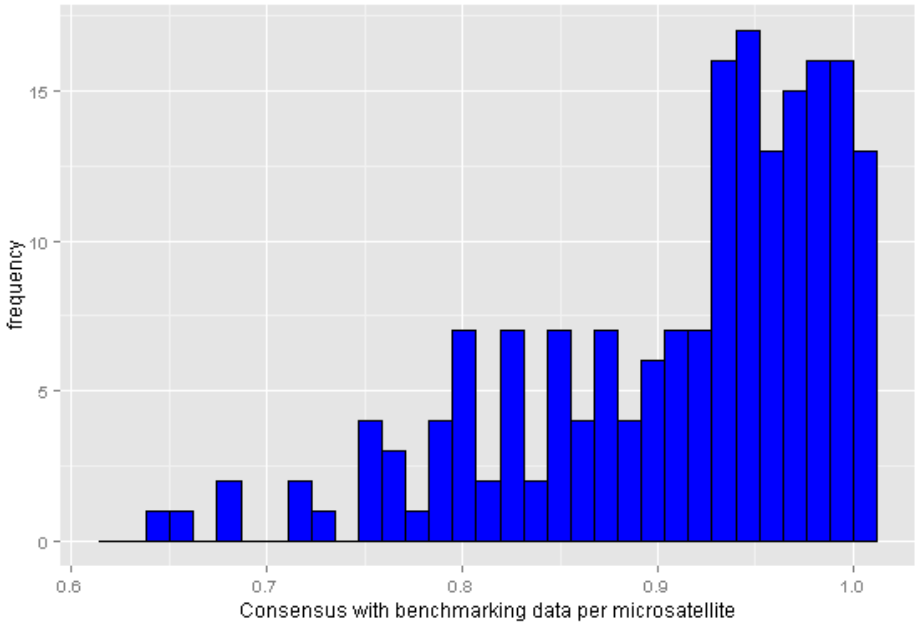

Figure 5.9: Accuracy of genotyping per microsatellite when adding the interaction between microsatellite and individual.

where $\#ad$ stands for number of allelic dropout cases and $\#cases_n$ represents the number of cases where n reads are available to determine a genotype. The effect that this could have on our results in however negligible and will not be considered here. Table 5.8 summarizes all improvements and corrections made and shows the contribution of each to the genotyping accuracy. It is worth mentioning that despite the decrease in accuracy caused by adding the per individual estimates of slippage rates, removing it from the final result decreases the accuracy. This indicates that in combination with other factors of the model the contribution of these slippage estimates is positive.

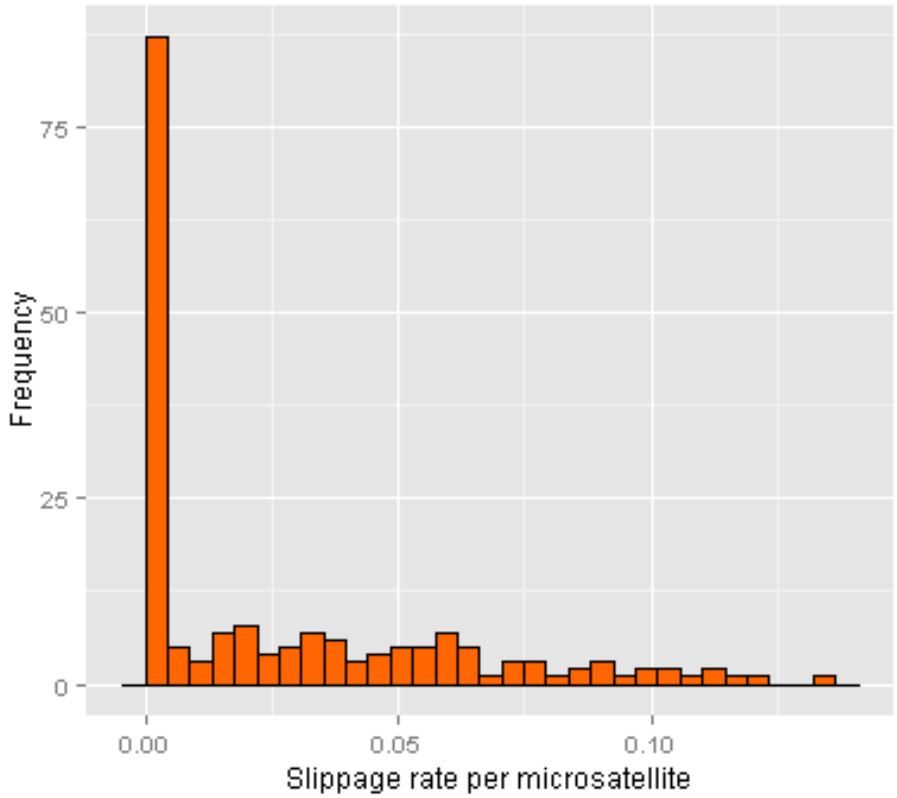

Figure 5.10: Distribution of slippage rates per microsatellite when considering the contribution of each individual and weighing it with the inverse of its variance.

Improvement	**Genotyping accuracy**
lobSTR genotype caller	87.5%
Estimating microsatellite specific slippage rates	91.8%
Applying logistic regression	91.8%
Adding the probability of being an error read divided by the number of alleles in the population to likelihood contribution	93%
Considering the contribution of each individual and each microsatellite when estimating slippage rates	92.8%
Considering the contribution of each individual and each microsatellite when estimating slippage rates and weighing it with the inverse of its variance	93%
Adding parameter for probability of deleting or adding repeat units	96.3%

Table 5.8: Total summary of improvement of genotyping accuracy

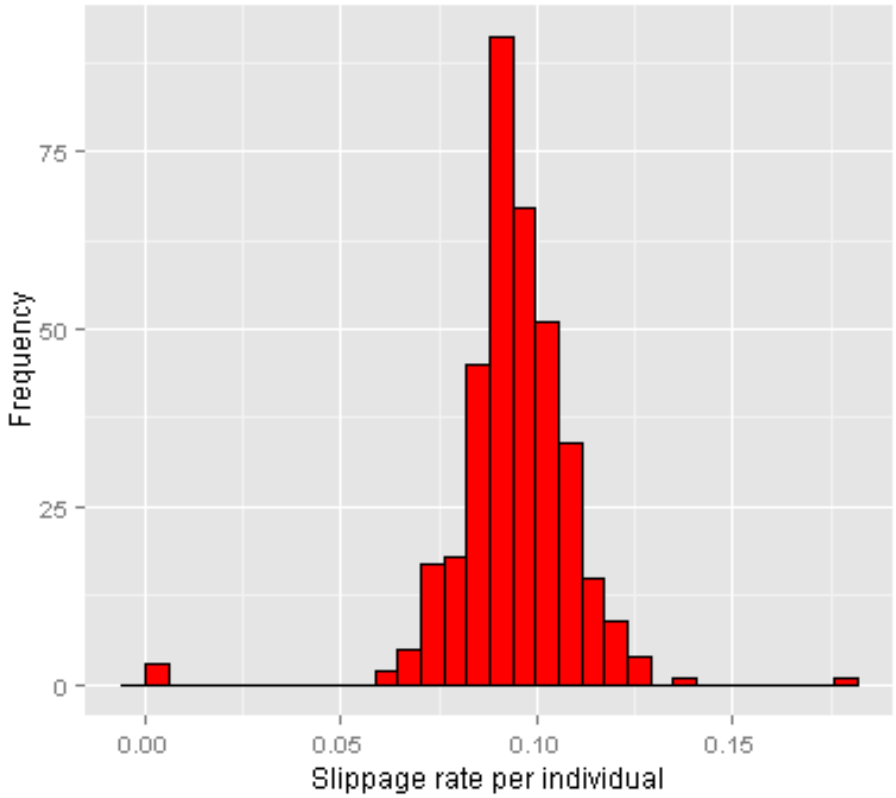

Figure 5.11: Distribution of slippage rates per individual when considering the contribution of each microsatellite and and weighing it with the inverse of its variance.

5.3.3 Updated error model vs. lobSTR error model

Here we compare the result of the final version of our updated error model to the results of the error model from lobSTR. Figure 5.17 shows the consensus between the two models on a per microsatellite basis. There were four microsatellites where the genotyping accuracy of the lobSTR error model genotyping was greater than the one of the updated model genotyping. In all four cases the accuracy was above 95% for both models and the maximum difference was 3.6%.

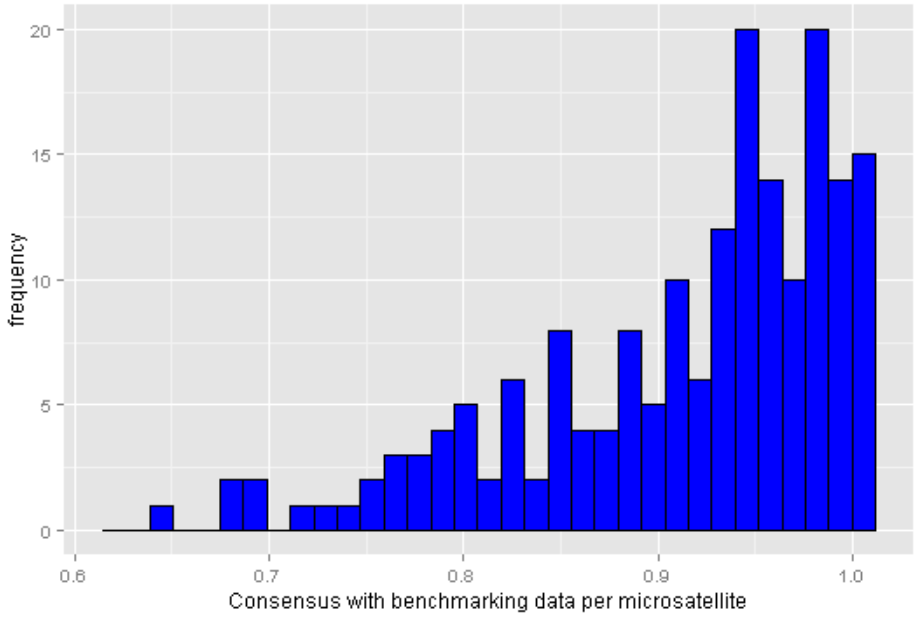

Figure 5.12: Accuracy of genotyping per microsatellite when adding the interaction between microsatellite and individual and weighing it with the inverse of its variance.

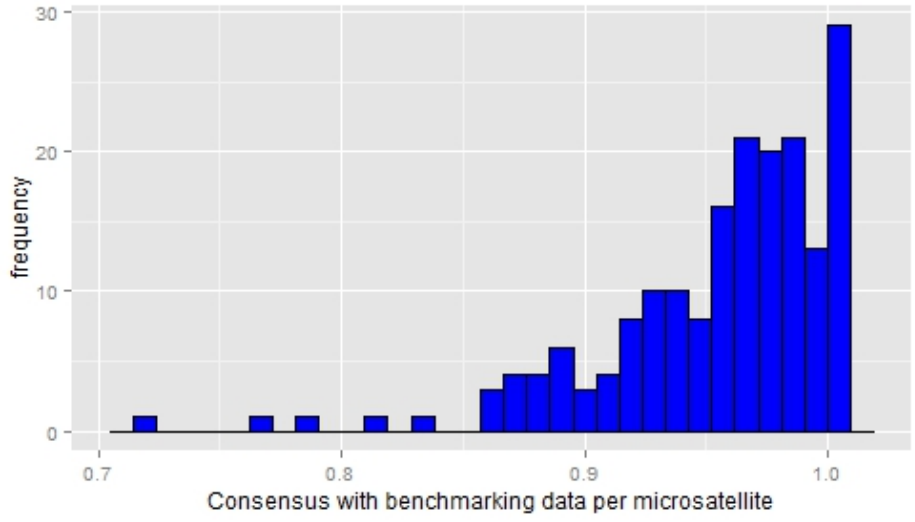

Figure 5.13: Histogram of genotyping accuracy per microsatellite when adding parameter for probability of slippage errors adding vs removing repeat units.

Figure 5.14: The consensus with benchmarking data as a function of the number of reads used to infer the genotype.

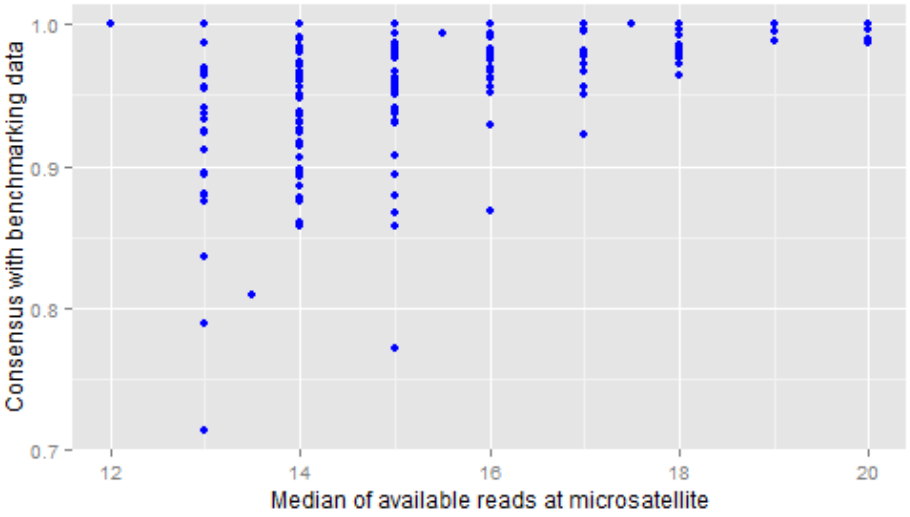

Figure 5.15: The consensus with benchmarking data as a function of the median of the coverage.

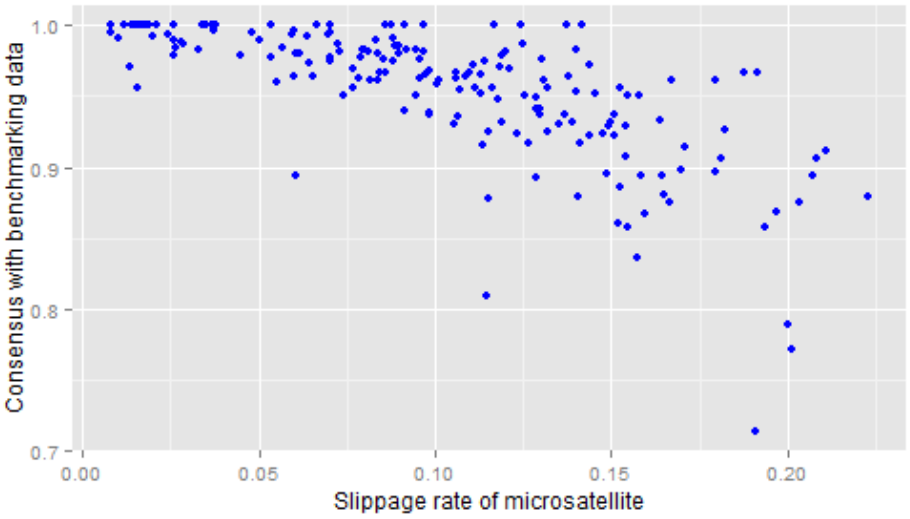

Figure 5.16: The consensus with benchmarking data as a function of the slippage rate at the microsatellite.

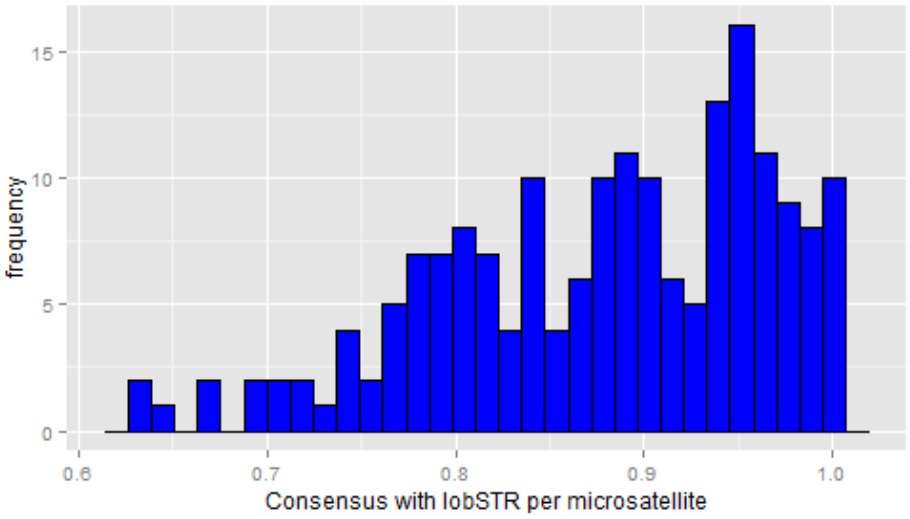

Figure 5.17: Consensus between updated error model and lobSTR error model genotyping results.

Chapter 6

Conclusion

6.1 Summary

Microsatellites are repetitive areas in the genome with a repeat motif of length 2-6 bases and have a high mutation rate. The number of repeats in a microsatellite are known as the alleles of the microsatellite and vary greatly between individuals. This makes determining the alleles of microsatellites in individuals very interesting but their structure makes it hard and time consuming using the sequencing technology available today.

Here we have shown that when creating a microsatellite profile for an individual using previously aligned data it is possible to significantly decrease the running time by considering only reads that are either aligned to a known microsatellite location or not aligned at all. This type of filtering dismisses a large portion of the data immediately without significantly effecting the microsatellite profile. To do this we used the alignment locations of reads along with attributes indicating undesirable qualities. We compared the microsatellite profiles generated using this input to the profiles generated using all reads from a whole genome sequencing BAM-file. The comparison showed a considerable decrease in input size and an average of 74.4% decrease in running time but only a very small difference in the microsatellite profiles returned, an average of less than 2%. This indicates that when using previously aligned data for generating microsatellite profiles it is enough to include only reads aligned to known microsatellite locations and unaligned reads.

An error model was also created using population information to better describe and identify reads supporting false alleles and thereby increase the accuracy of genotyping. The model dealt with stutter noise errors created during PCR amplification, misplaced reads and other sequencing errors. The reads resulting from sequencing errors or were misplaced where identified by a logistic regression classifier using a number of attributes extracted from their alignment in BAM-files. For modeling the stutter noise errors we

took into account both the contribution of each person to the stutter noise and the contribution of each microsatellite to the stutter noise. This created a unique stutter rate for each individual/microsatellite pair which increased the accuracy of the genotyping compared to previous methods. To estimate the accuracy we used genotypes available at deCODE obtained by microsatellite analysis using capillary electrophoresis. The accuracy of the error model provided with the micorsatellite profiler used for comparison was was 87.5% and the accuracy of the improved model was 96.3%.

Combining these two parts of the thesis we can take a previous method which was both slow inaccurate and create a new microsatellite genotyping procedure which is both more accurate and faster.

6.2 Suggestions for Future Research

6.2.1 Aligning reads to microsatellites

Future plans for this project include reimplementation of the sensing and alignment step using the SeqAn library. The new implementation would simply compute and store all k-mers with an entropy below a chosen threshold and then create hashtable of those instead of performing entropy calculations for all reads. This would hopefully decrease the running time and increase consensus with benchmark data results. The removal of low quality reads would also be added as a way to improve result quality.

6.2.2 Further improvements in the error model

Since the results of the error model trained here were less than perfect it would be interesting to try other approaches to training it. One of these would be to use data from individuals known to be homozygote in certain areas of the genome to estimate error and slippage rates. Since these areas are distributed over all of the genome then the estimated error rates should correctly reflect the true ones. Another approach would be to use other attributes to detect reads resulting from undefined sequencing errors or use another form of regression than the logistic regression.

Bibliography

[1] A Brief History of the Human Genome Project, https://www.genome.gov/12011239.

[2] File formats - great documentation - confluence , http://bejerano.stanford.edu/help/display/GREAT/File+Formats.

[3] Human Genetic Variation, http://science.education.nih.gov/supplements/nih1/Genetic/guide/pdfs/nih_genetics.pdf.

[4] An introduction to Variant Call Format, http://faculty.washington.edu/browning/beagle/intro-to-vcf.html.

[5] Modeling PCR stutter noise for accurate calling of STRs from short reads, http://erlichlab.wi.mit.edu/lobSTR/GSAStutterNoise_mgymrek.pdf.

[6] Paired-End Sequencing | Achieve maximum coverage across the genome | Illumina , http://www.illumina.com/technology/paired_end_sequencing_assay.ilmn.

[7] Single End Sequencing | Illumina , http://www.illumina.com/technology/single_read_sequencing_assay.ilmn.

[8] UCSC genome bioinformatics: FAQ, http://genome.ucsc.edu/FAQ/FAQformat#format1.

[9] What is FASTA format?, http://zhanglab.ccmb.med.umich.edu/FASTA/.

[10] What is logistic regression? - University of Strathclyde, http://www.strath.ac.uk/aer/materials/5furtherquantitativeresearchdesignandanalysi unit6/whatislogisticregression/.

[11] J. Adams. Dna sequencing technologies. *Nature Education*, 1(1), 2008.

[12] T. A. Brown. *Genomes*. Wiley-Liss, Oxford, 2nd edition, 2002.

[13] P. J. A. Cock, C. J. Fields, N. Goto, M. L. Heuer, and P. M. Rice. The Sanger FASTQ file format for sequences with quality scores, and the Solexa/Illumina FASTQ variants. *Nucleic Acids Res*, 38(6):1767–1771, 2010.

[14] P. Danecek, A. Auton, G. Abecasis, C. A. Albers, E. Banks, M. A. DePristo, R. E. Handsaker, G. Lunter, G. T. Marth, S. T. Sherry, G. McVean, R. Durbin, and . G. P. A. G. . The variant call format and vcftools. *Bioinformatics*, 27(15):2156–2158, Aug 2011.

[15] A. P. Dempster, N. M. Laird, D. B. Rubin, et al. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal statistical Society*, 39(1):1–38, 1977.

[16] A. Döring, D. Weese, T. Rausch, and K. Reinert. Seqan an efficient, generic c++ library for sequence analysis. *BMC Bioinformatics*, 9:11, 2008.

[17] J. Duitama, A. Zablotskaya, R. Gemayel, A. Jansen, S. Belet, J. R. Vermeesch, K. J. Verstrepen, and G. Froyen. Large-scale analysis of tandem repeat variability in the human genome. *Nucleic Acids Res*, 2014.

[18] H. Ellegren. Microsatellites: simple sequences with complex evolution. *Nat Rev Genet*, 5(6):435–445, 2004.

[19] K. A. Frazer, S. S. Murray, N. J. Schork, and E. J. Topol. Human genetic variation and its contribution to complex traits. *Nature Reviews Genetics*, 10(4):241–251, 2009.

[20] S. D. e. a. Griffiths AJF, Miller JH. *An Introduction to Genetic Analysis. 7th edition.* W. H. Freeman, http://www.ncbi.nlm.nih.gov/books/NBK21894/, 2000.

[21] M. Gymrek, D. Golan, S. Rosset, and Y. Erlich. lobstr: A short tandem repeat profiler for personal genomes. *Genome Res*, 22(6):1154–1162, Jun 2012.

[22] G. Highnam, C. Franck, A. Martin, C. Stephens, A. Puthige, and D. Mittelman. Accurate human microsatellite genotypes from high-throughput resequencing data using informed error profiles. *Nucleic Acids Res*, 41(1):e32, Jan 2013.

[23] H. Li, B. Handsaker, A. Wysoker, T. Fennell, J. Ruan, N. Homer, G. Marth, G. Abecasis, R. Durbin, and . G. P. D. P. S. . The sequence alignment/map format and samtools. *Bioinformatics*, 25(16):2078–2079, Aug 2009.

[24] H. Li and N. Homer. A survey of sequence alignment algorithms for next-generation sequencing. *Brief Bioinform*, 11(5):473–483, Sep 2010.

[25] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to information retrieval*, volume 1. Cambridge university press Cambridge, 2008.

[26] S. M. Mirkin. Expandable dna repeats and human disease. *Nature*, 447(7147):932–940, Jun 2007.

[27] B. Pálsson, F. Pálsson, M. Perlin, H. Gudbjartsson, K. Stefánsson, and J. Gulcher. Using quality measures to facilitate allele calling in high-throughput genotyping. *Genome research*, 9(10):1002–1012, 1999.

[28] W. R. Pearson and D. J. Lipman. Improved tools for biological sequence comparison. *Proc Natl Acad Sci U S A*, 85(8):2444–2448, Apr 1988.

[29] J. S. Reis-Filho. Next-generation sequencing. *Breast Cancer Res*, 11(Suppl 3):S12, 2009.

[30] S. Snider and J. Brimlow. An introduction to population growth. *Nature Education Knowledge*, 4(4):3, 2013.

[31] J. X. Sun, A. Helgason, G. Masson, S. S. Ebenesersdóttir, H. Li, S. Mallick, S. Gnerre, N. Patterson, A. Kong, D. Reich, et al. A direct characterization of human mutation based on microsatellites. *Nature genetics*, 44(10):1161–1165, 2012.

[32] H. Tae, D.-Y. Kim, J. McCormick, R. E. Settlage, and H. R. Garner. Discretized gaussian mixture for genotyping of microsatellite loci containing homopolymer runs. *Bioinformatics*, 30(5):652–659, Mar 2014.

[33] J. C. Venter, M. D. Adams, E. W. Myers, P. W. Li, R. J. Mural, G. G. Sutton, H. O. Smith, M. Yandell, C. A. Evans, R. A. Holt, et al. The sequence of the human genome. *science*, 291(5507):1304–1351, 2001.

[34] X. Yang, S. Aluru, and K. S. Dorman. Repeat-aware modeling and correction of short read errors. *BMC Bioinformatics*, 12 Suppl 1:S52, 2011.

List of Figures

List of Tables