Simulation of Related, MDA-Amplified Single Cells

Leonor Schubert Santana

Bachelor Thesis

Prof. Dr. Niko Beerenwinkel

Dr. Francesco Marass

Dr. Jochen Singer

Department of Biosystems Science and Engineering

Spring semester, 2019

ETH

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Abstract

In recent years, single cell sequencing has become increasingly popular thanks to the improvements of technologies and the falling costs of next-generation sequencing. This has been especially useful in tumour research, where heterogeneity between single tumour cells is of interest. However, with single cell sequencing new challenges emerge. In order to sequence the small amount of starting material, whole-genome amplification is needed. A popular choice is MDA amplification because it produces a lot of template DNA, which is useful when one wants to call mutations. This can lead to additional noise, such as uneven coverage, missing information and allelic imbalance. The ‘Single Cell Read Simulator’ software, presented in this thesis, aims to simulate the process of amplification and sequencing of a chromosome. Additionally, it simulates the phylogenetic relationship between cells arising from cell division. This enables the software to simulate tumour data sets, which can be helpful in testing of single-cell mutation identification algorithms.

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Declaration of originality

The signed declaration of originality is a component of every semester paper, Bachelor's thesis, Master's thesis and any other degree paper undertaken during the course of studies, including the respective electronic versions.

Lecturers may also require a declaration of originality for other written papers compiled for their courses.

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor.

Title of work (in block letters):

Simulation of Related, MDA-Amplified Single Cells

Authored by (in block letters):

For papers written by groups the names of all authors are required.

Name(s):	**First name(s):**
Schubert Santana	Leonor Patricia

With my signature I confirm that

- I have committed none of the forms of plagiarism described in the 'Citation etiquette' information sheet.
- I have documented all methods, data and processes truthfully.
- I have not manipulated any data.
- I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for plagiarism.

Place, date	**Signature(s)**
24.08.2019	L. Schubert Santana

For papers written by groups the names of all authors are required. Their signatures collectively guarantee the entire content of the written paper.

Table of Contents

1 Introduction

Single-cell sequencing (SCS) methods have gained popularity in recent years and promise new insights into genetic mosaicism, the genetic diversity, in both healthy and diseased organisms. Particularly in cancer research, where the intra-tumour heterogeneity plays a big role in the development of the tumour (Wang & Navin, 2015), identifying the differences between single tumour cells and recognizing lead actors in disease progression is of great importance (Gawad, Koh, & Quake, 2016).

The challenge with SCS is that there is not a lot of starting material. There are only approximately two copies of the genome, the two alleles. To be able to sequence them as fully as possible one needs to amplify that DNA. Different methods have been developed to do this, each tailored to different applications. Methods leading to uniform coverages are useful for calling copy-number changes, while methods leading to high coverages are useful for calling mutations. The sequencing part is more or less standard.

In this thesis we present our simulator, the ‘Single Cell Read Simulator’ to model reads obtained by MDA amplification and Illumina sequencing of related single cells. The simulation is useful to investigate the impact of MDA artefacts on single-cell data. In order to simulate reads for single-cell sequencing of tumour cells, the way the cells develop and how they are processed has to be included. For that reason, the ‘Single Cell Read Simulator’ firstly defines a binary tree that relates the cells. Then it introduces some somatic mutations in the cells according to this tree structure, creating simulated tumour cells. In the next step, MDA amplification and the sequencing of these cells are simulated. The resulting reads aim to approximate the SCS reads.

This paragraph quickly outlines the structure of this thesis. In the second chapter, we provide some background information on whole-genome amplification, MDA, Illumina sequencing and phylogenetic trees. The next two chapters provide the approach and implementation of the ‘Single Cell Read Simulator’. In the fifth chapter we discuss

some results obtained with our simulation. At the end we provide some conclusions and an outlook.

2 Background

2.1 Whole Genome Amplification

While SCS of DNA has been greatly improved over the last years, the small amount of starting material still poses a problem. A single cell only contains two copies of DNA, which means amplification is necessary before sequencing. To this end, whole-genome amplification (WGA) is used, of which several variants have been developed (Wang & Navin, 2015).

	PCR	MDA	Hybrid
Coverage Uniformity	Quite Uniform	Highly non-uniform	Uniform
Physical coverage	Low	High	Intermediate
Amplification error rate	High	Low	Intermediate

Table 2. 1: Comparison of WGA methods: The three main groups are shown, PCR-based, Isothermal and Hybrid. **(Wang & Navin, 2015)**

The three main groups of WGA are: Pure PCR-based amplification, isothermal amplification and hybrid methods. PCR-based amplification starts the amplification process with priming, using either random oligonucleotides or universal adapters that bind to the DNA strands. These can then be amplified in a regular PCR reaction. While in PCR thermal cycling is employed, isothermal methods, such as MDA, amplify DNA at a constant temperature. MDA will be discussed in detail in Chapter 2.2 Hybrid methods use a combination of the methods above. This combines the advantages of both other groups of WGA. For more details refer to the paper 'Design and Analysis of Single-Cell Sequencing Experiments' (Grün & van Oudenaarden, 2015).

The mentioned methods of WGA have different technical errors occurring and therefore, the appropriate method for a certain application has to be chosen carefully. As shown in Table 2.1, MDA and hybrid methods have a medium to high coverage, useful for mutation calling. PCR and hybrid on the other hand have more uniform coverage, which is important for copy-number calling.

2.2 MDA

MDA is a popular isothermal method because it provides great amplification and low error rates. Unfortunately, it also exhibits a high non-uniformity in genome coverage, which can greatly impact analysis of the data.

After the extraction of DNA from an isolated single cell, the first step of MDA is the denaturation of the two strands. Random priming is then used for the polymerase to bind to the DNA strands and start replication. MDA uses a Φ29 polymerase, which has high processivity and proofreading capacities, leading to mentioned high coverage and low error rates of MDA. In addition, Φ29's strand-displacement ability allows it to release an already copied strand ahead of itself, making it available to new priming and amplification (see Figure 2.1). This leads to an exponential amplification and a high coverage even when starting with a single genome. As amplified DNA can serve as a template for further amplification, parts of the genome that are amplified early tend to be overrepresented, explaining the uneven coverage.

Another problem that arises from the exponential amplification is that errors introduced at the beginning can be overrepresented. Normally real mutations can be distinguished from errors, because real mutations are observed in larger quantities. Overrepresentation of errors makes these MDA errors hard to distinguish from real mutations because they may appear just as often. Dropout events are non-amplifications of one of the two alleles and can lead to overestimating the importance of mutations happening in the remaining allele. (Dean, et al., 2002).

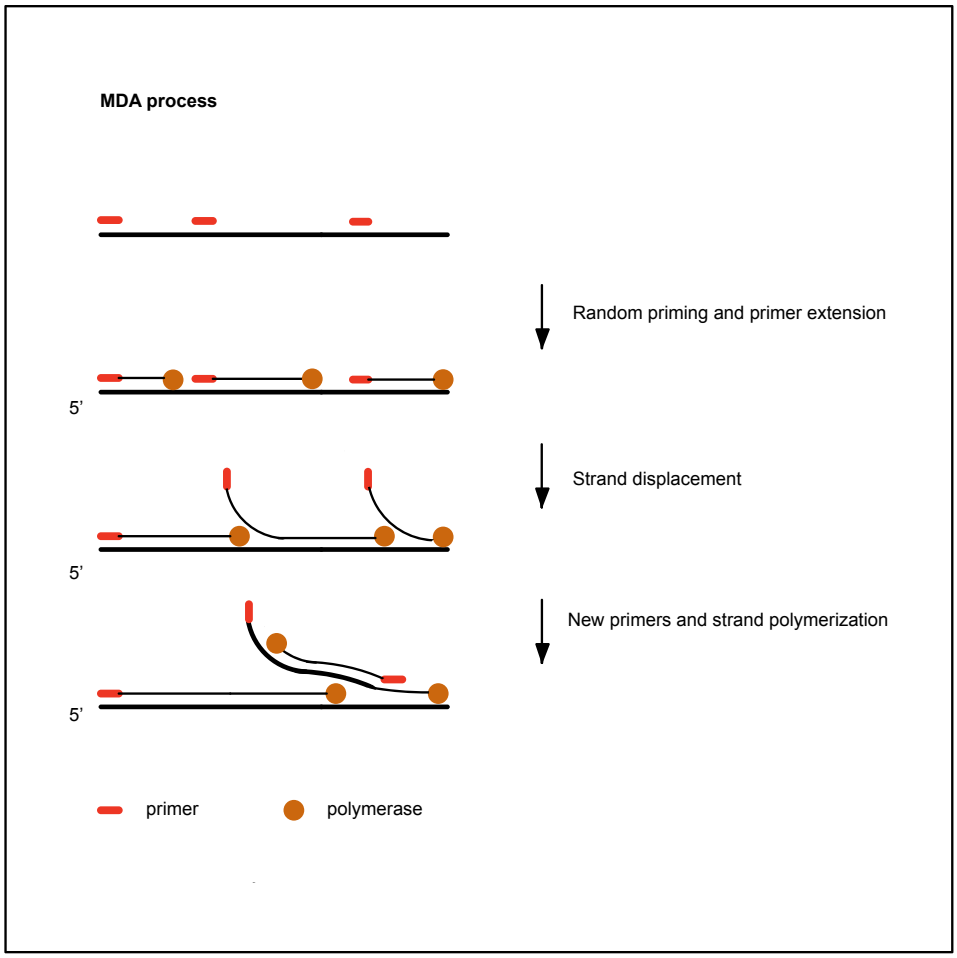

Figure 2. 1: MDA process: 1. Denaturation and random priming of the strands. 2. Copying of the strands by the polymerase Φ29. 3. Φ29 releases already copied strands ahead of itself making them available to new priming and copying.

2.3 Illumina Sequencing and ART

A lot of new sequencing techniques have been developed, which are faster and more efficient than before. One of the most used techniques is Illlumina next-generation sequencing (NGS). It consists of three basic steps, which are outlined in the following paragraphs.

The first step is the preparation of the DNA library. The DNA strands are separated and broken into random fragments of approximately the same size. Adapters are ligated to the ends of these strands. In the next step, the strands are bound to a flow

cell and are amplified into clusters by bridge amplification. Now the actual sequencing cycle can begin. Nucleotides with attached reversible terminators and fluorescent tags are added to the flow cell. All four dNTPs are present, each with a different colour. Polymerases attach the nucleotides complementary to the strands bound to the flow cell. Because the nucleotides have terminators attached, only one dNTP is attached per strand. Fluorescence is elicited with a laser and a camera records the colours of the clusters. Afterwards, the unused dNTPs are washed off and the attached terminators get chemically removed. This cycle of adding dNTPs, taking a picture and washing is repeated until the whole DNA strands are sequenced. Because multiple fragments on the same flow cell can be sequenced at the same time, this process is highly parallel.

One issue which can occur in Illumina sequencing is duplicates. Duplicates occur when the same fragment is bound to multiple different positions on the flow cell and therefore are treated as different fragments. This leads to overrepresentation of these fragments during alignment proceeding into overconfidence of these parts.

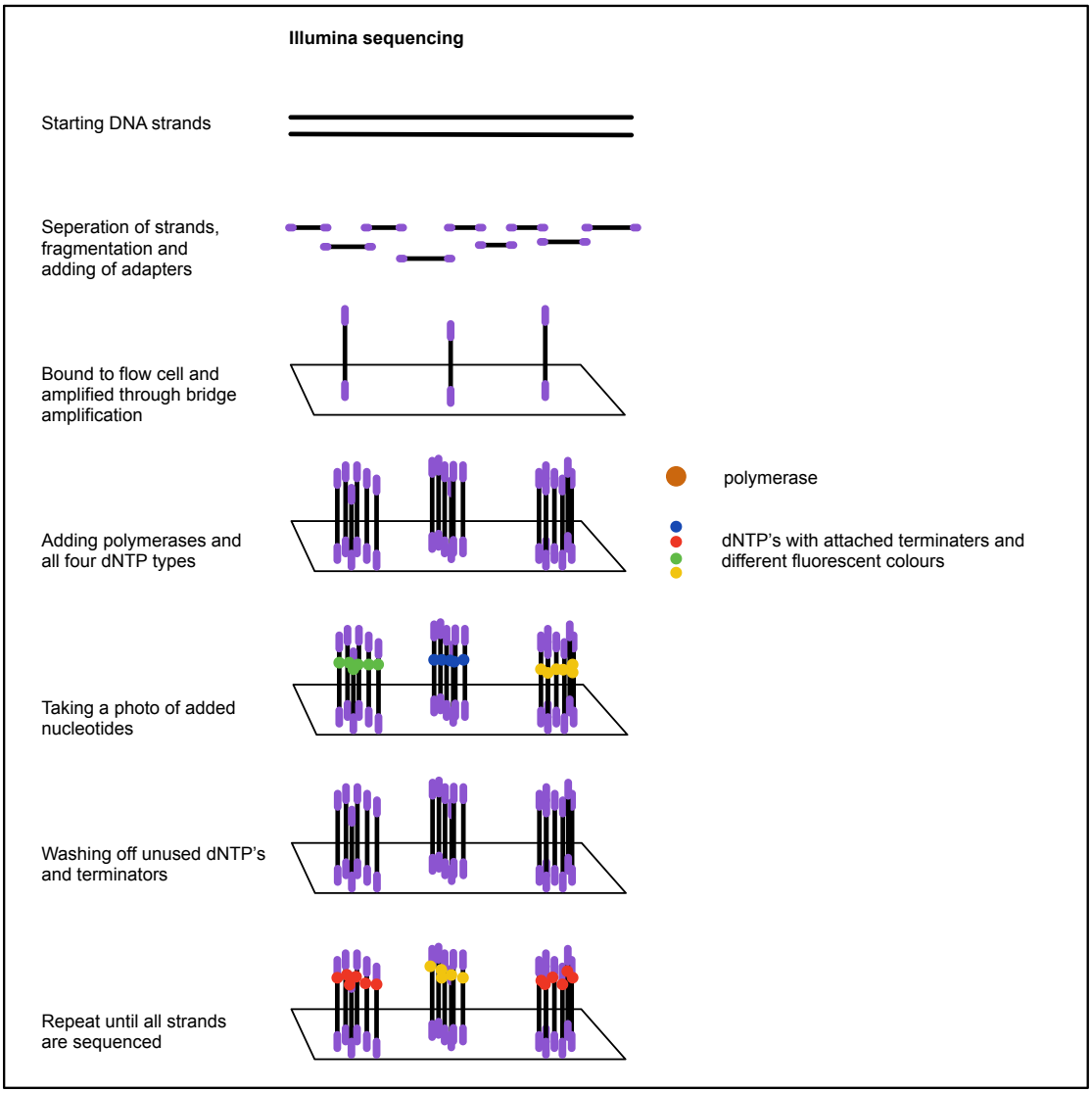

Figure 2. 2: Illumina sequencing: The basic steps of Illumina sequencing are shown.

If the strands are only sequenced from one end, the reads are called single-end reads. It is also possible to sequence a strand from both ends, creating a forward and a reverse read. This means that twice as many reads can be produced with the same preparation. If these two reads together are smaller than the whole strand, they leave an unsequenced part in the middle. Because the fragment lengths, as well as the read lengths are known, the length of the unsequenced part can be estimated. This helps to determine indels and duplicates.

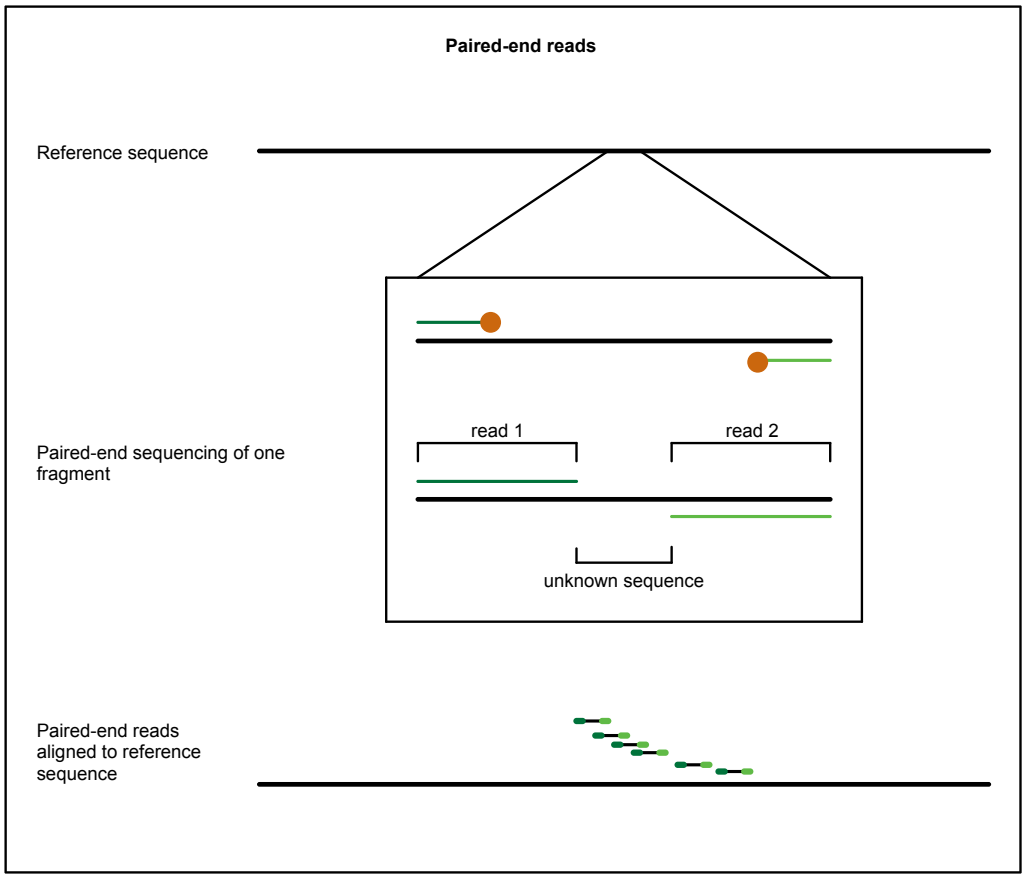

Figure 2. 3: Paired-end reads: Fragments of a strand are sequenced from both ends, leading to two reads. The fragments are then aligned to a reference sequence. Overlapping fragments can help determine the non-sequenced parts in the middle of other fragments.

ART is a next-generation sequencing read simulator, which can mimic reads generated by the three main commercial NGS platforms. It can be used to simulate an Illumina sequencing process of both single-end and paired-end reads. With built-in error profiles estimated from training datasets, as well as some user-specified input, e.g., read length and coverage, it outputs FASTQ files containing simulated reads (Huang, Li, Myers, & Marth, 2012).

2.4 Phylogenetic Trees for Single-cell Tumour Data

Tumour cells are highly heterogeneous, which is one of the major problems in cancer treatment. Their growth corresponds to cell division, where at each division, cells may acquire new mutations. This is what creates genetic diversity. By following the patterns of mutations found in single cells one can learn how the tumour evolved. In practice, this reconstruction is complicated by noise in the data, due both to the DNA amplification process and DNA sequencing.

Tumour cells and their divisions can be represented with a binary tree encoding their relationships, where the nodes symbolize the cells. All children of a node contain the same mutations as their parent (Jahn, Kuipers, & Beerenwinkel, 2016).

2.5 Related Work

Some parts of the ‘Single Cell Read Simulator’ have already been studied, like tumour data relations and Illumina sequencing. Less work has gone into MDA simulations. One notable contribution to model MDA is the MDAsim presented in the paper ‘MDAsim: A multiple displacement amplification simulator’ (Tagliavi & Draghici, 2012). It is based on a biochemical/mechanistic model for the MDA simulation.

Our work is unique in the sense that it combines these components into a simulation of the full process. It also takes a different approach to model MDA than MDAsim.

3 Approach

3.1 Overview

The goal of the ‘Single Cell Read Simulator’ is to simulate the entire process of amplifying related cells with MDA and sequencing them with Illumina. The individual steps of the simulator, necessary for the description of that process, are depicted in Figure 3.1. Note that the simulation works chromosome by chromosome. So, without loss of generality, we will refer to a chromosome as a cell in the following. Firstly, a binary tree representing cell relations is generated. A starting cell is then generated and associated with the tree’s root. In the following step, mutations occurring from cell division according to the structure of the generated binary tree are added to the cells. The MDA process of these cells is then simulated and finally the reads from Illumina sequencing are generated. In the following subchapters, those steps are discussed in more detail.

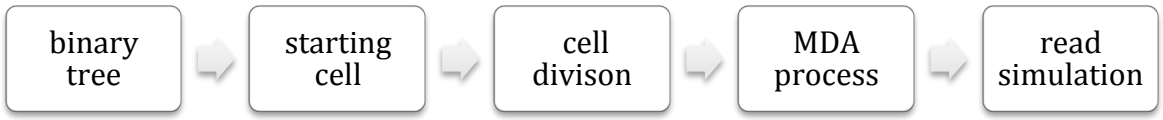

Figure 3. 1: Simulation procedure: 1. A binary tree simulating cell division of tumour cells is created. 2. A starting cell is created. 3. Mutations occurring from cell division are added to the cells according to the created binary tree. 4. The simulation of an MDA process amplifies the alleles created by the binary tree. 5. The amplified segments are fed to ART.

3.2 Tree Simulation

The development from a single starting cell to *n* tumour cells is represented with a binary tree of *n* leaves and *n-1* inner nodes. Each node is a cell consisting of two alleles. A node with two children represents a cell division event. All mutations of a node are passed to its children. The children can introduce additional mutations simulating mutations occurring during cell division. Mutations have to occur in at least two cells. Otherwise, there is insufficient evidence supporting these mutations and they are not considered in the analysis.

The root of this tree is associated with a starting cell. For the starting cell, we begin with a copy of the chromosome from the reference genome. To include germline mutations (e.g. SNPs), homozygous and heterozygous mutations are added to two copies of the reference chromosome representing the two alleles present in a cell. Homozygous mutations are mutations appearing in both alleles, while heterozygous only happen in one of the two alleles.

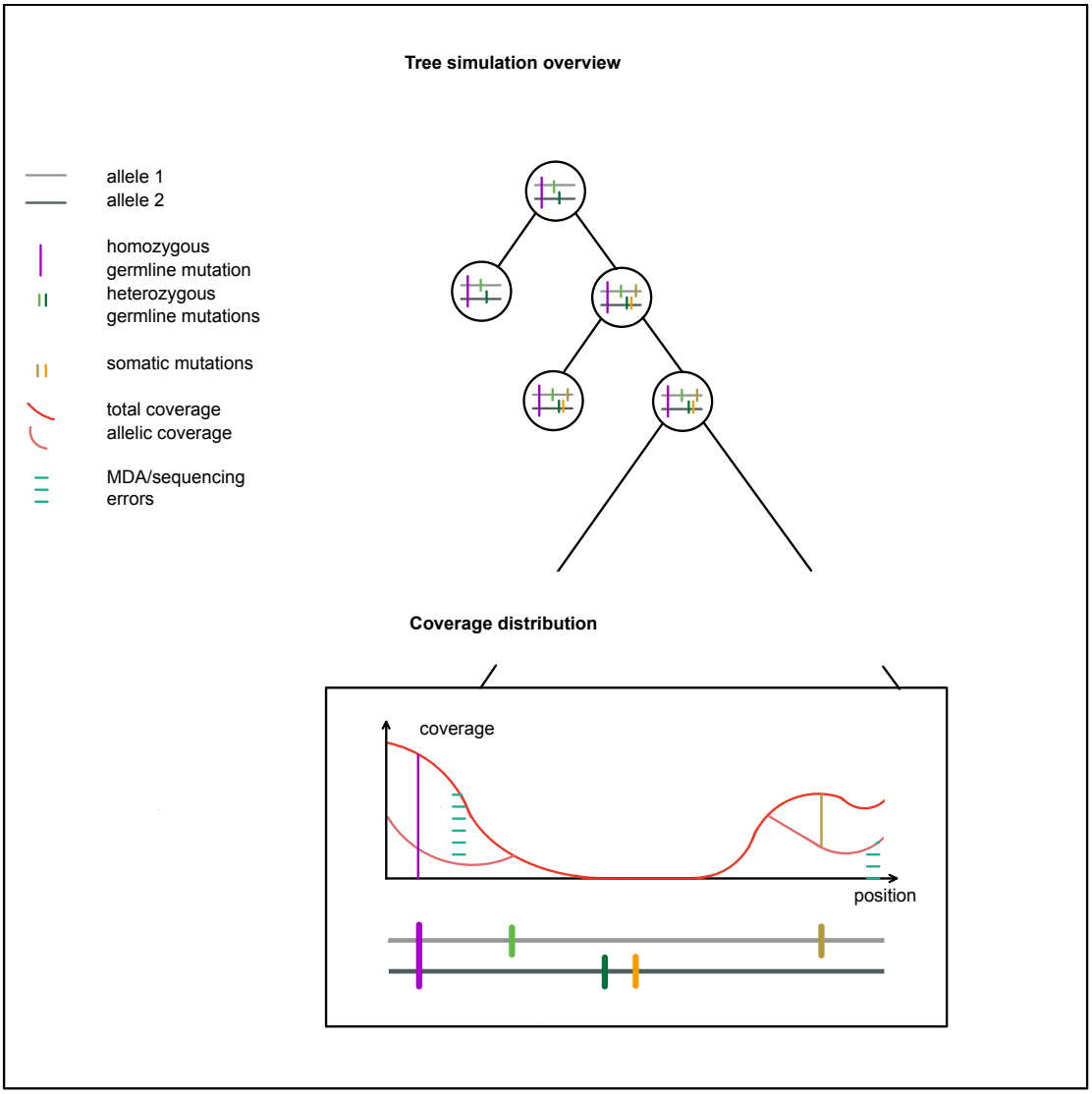

Figure 3. 2: Top: Tree simulation overview: The binary tree represents cell relations, where the nodes symbolize cells and a node having two children, symbolizes a cell division. The cells contain germline and somatic mutations.

Bottom: Coverage per nucleotide of a single cell: The coverage distribution for both alleles and for a single allele after MDA is displayed. The graph also shows the real mutations and MDA errors.

3.3 MDA

For the MDA procedure, each allele is handled independently. To model the uneven coverage, each allele is divided into segments with different coverages according to a distribution. The segment lengths are chosen according to a negative binomial distribution. The parameters are user defined through mean μ_s and standard deviation σ_s. These are then converted to the parameters of the negative binomial distribution, overdispersion r and success probability p (see Appendix A.1). The coverage of each segment is also drawn from a negative binomial distribution with user defined parameters mean μ_c and standard deviation σ_c. The rational behind it is that every segment i in the chromosome has a particular propensity to be sequenced, which we denote with λ_i. Therefore, the number of reads covering the segment i follows a poisson distribution with mean $\mu_c\lambda_i$. If λ_i is assumed to follow a gamma distribution, then the coverage follows a negative binomial distribution (Sampson, Jacobs, Yeager, Chanock, & Chatterjee, 2011).

To handle the big amount of zero coverages observed in real data, the distribution was extended to a zero inflated negative binomial distribution. By setting the zero inflation probability to zero, the original negative binomial distribution is recovered. The method described in this paragraph leads to uneven coverage and allelic imbalance as observed in real data.

To include errors arising during MDA, the copying of a segment is implemented as a Pólya urn model. Figure 3.3 shows this process. In the first iteration, the original data segment is copied and with some probability p_{MDA} an error is introduced into this segment. In the next iteration, one of the already copied segments is randomly chosen as a reference to be duplicated and again with probability p_{MDA} a mutation is added. This process is repeated until the chosen number of copies is achieved. Effectively, this represents the overamplification of errors introduced by MDA.

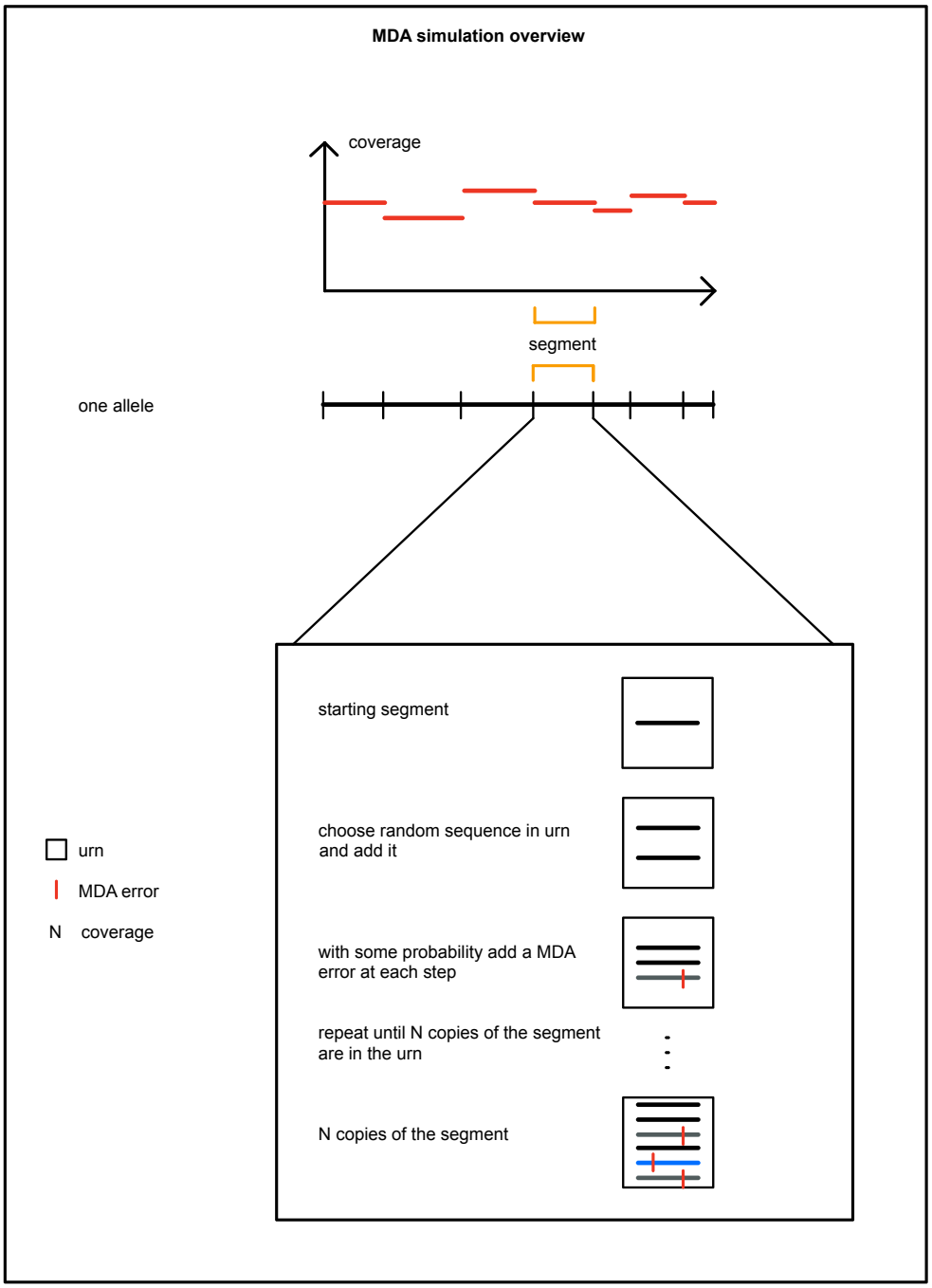

Figure 3. 3: MDA simulation overview: Each allele is split into segments of different lengths. For each segment a coverage is chosen. The segment is then copied according to a Pólya urn model until the chosen coverage is reached.

3.4 Read Simulation

As in actual single-cell sequencing, the next step is to sequence the amplified segments. For that, each segment is fed to the sequencing read simulator ART (Huang, Li, Myers, & Marth, 2012) with the command for paired-end reads and using the Illumina sequencer implementation. Also the desired read length, the mean μ_f as well as the standard deviation σ_f of the segments and the fold coverage of the Illumina bridge amplification are specified. After running ART, some of the reads are dropped with a probability $p_{dropped}$, mimicking the fact that not all fragments are sequenced.

4 Implementation

4.1 Overview

The single cell read simulator's core structure is implemented with snakemake (Köster & Rahmann, 2012). Snakemake is a python based workflow management system and is used here to call the necessary scripts and command line directives. The scripts called by snakemake are all implemented in python. Graphiz, which is a graph visualization software, is used to plot the binary tree created by the simulation. To work with VCF und FASTA files, pyVCF (a python parser for VCF files) and Biopython (a set of tools for biological computation including FASTA file handling) are used (Cock, et al., 2009). Bcftools handles the file conversion from VCF to FASTA (Li, et al., 2009). As mentioned before, simulating the Illumina sequencing is done with ART (Huang, Li, Myers, & Marth, 2012).

The snakemake workflow consists of five main parts, in line with the components of the simulator described in the previous section (see Figure 4.1).

- Generating a random binary tree and plotting it (uses graphiz)
- Creating the starting alleles (uses pyVCF and Biopython)
- Introducing additional mutations according to the generated binary tree (uses pyVCF and Biopython)
- Adding the mutations from the VCF files into FASTA files (uses bcftools)
- Simulating MDA and sequencing (uses pyVCF, Biopython and ART)

User interaction is handled over a yaml config file, where parameters of the simulation can be set. In order to guarantee reproducibility and to change outcomes of different random processes each random process has a seed that can be changed by the user.

Since the implementation uses a lot of existing tools, the necessary environment for each snakemake rule has been created and is automatically called by snakemake. The conda package manager is used to handle all needed packages.

The outputs of the simulation are two FASTQ files for each allele containing the generated paired-end reads. The VCF files for the individual alleles as well as the MDA errors added to each allele are also provided.

The tree structure of the ‘Single Cell Read Simulator’ can also be ignored, which means only the process of MDA and Illumina sequencing of a chromosome is simulated. To this end, the number of germline mutations and somatic mutations have to be set to zero and the number of cells in the tree (leaves) has to be set to one.

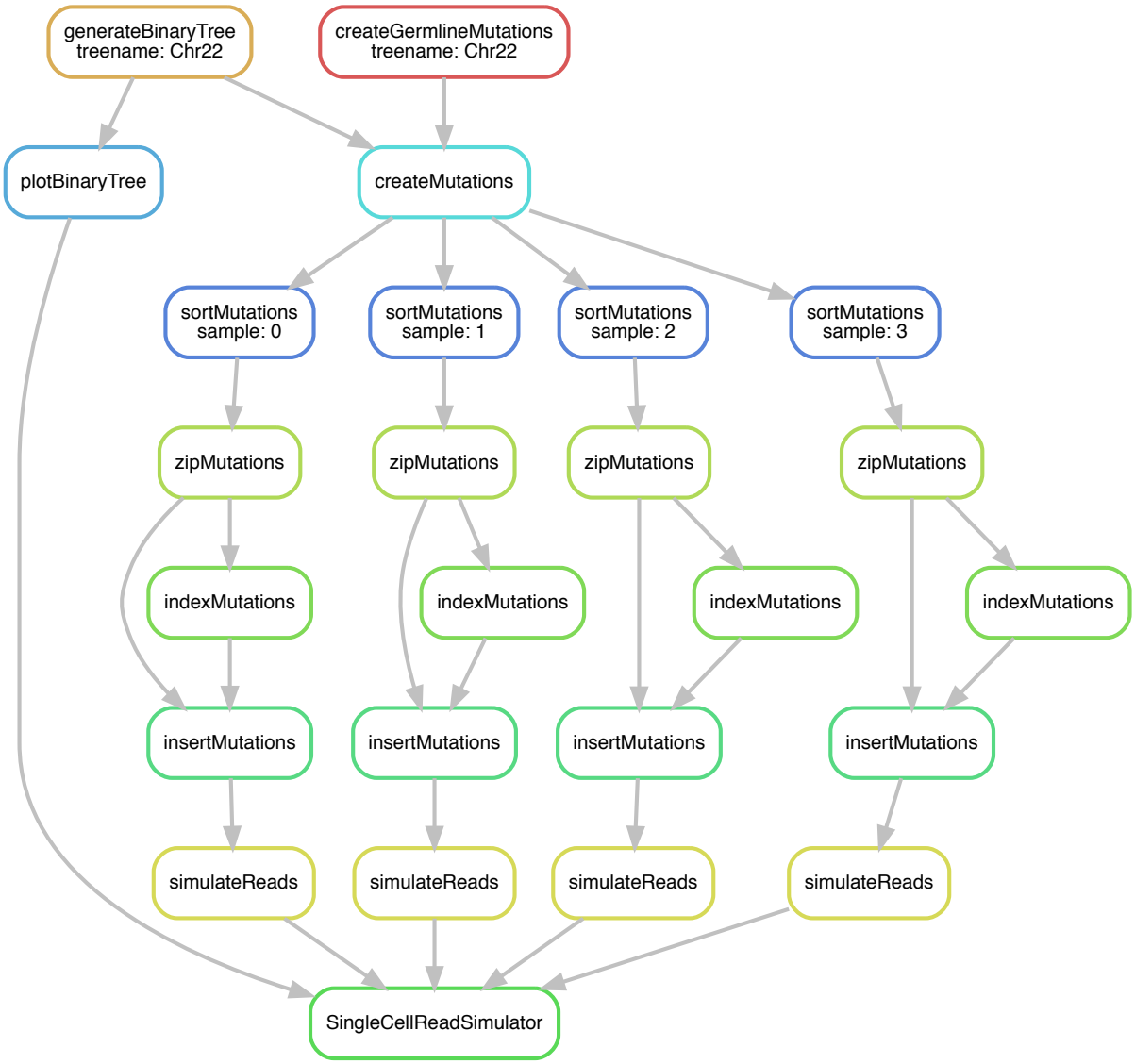

Figure 4. 1: Directed acyclic graph of workflow: Shown is the snakemake workflow, simulating chromosome 22 of two cells.

4.2 Binary Tree

Random binary trees are created recursively. Starting at the root node, the total number of nodes of a tree with *N* leaves (without root) *2×N−2* is randomly distributed to the two children of the root. Then for each child, the given number of nodes is again randomly distributed to its children. This process is repeated until all *2×N−2* nodes are allocated.

The generated tree is saved into a matrix of size *3×N*, where a random number between *1* and *2×N−2* identifies each node. The i^{th} column represents the node identified by the number *i*. The first and second rows indicate the indices of a node's children. The third row numbers the leaves consecutively from *0* to *N* for later analysis. Having the values of its children in the table set to *−1* marks a node being a leaf, e.g. not having any children. This slightly complicated setup simplifies later computations.

The plotted tree in Figure 4.2 shows the generated tree, where *-1* now indicates an inner node. The numbers inside the leaves signify the two alleles of the according cell and are subsequently used as a reference for all files generated by the simulation.

Parent	0	1	2	3	4	5	6
Right child	5	−1	−1	1	−1	3	−1
Left child	6	−1	−1	2	−1	4	−1
Leaf number	−1	0	1	−1	2	−1	3

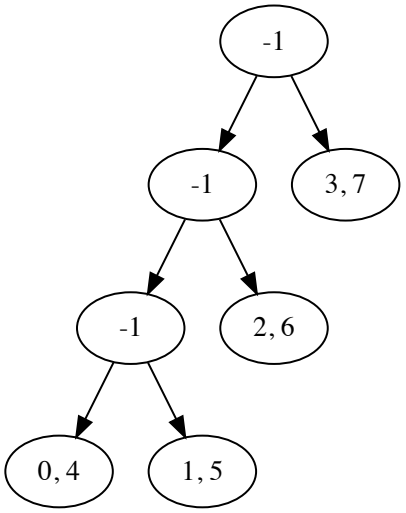

Figure 4. 2: Binary tree representation. Left: Table representation of a binary tree with four leaves i.e. seven nodes in total. Right: Directed graph representation of binary tree, showing the allele indices corresponding to each leaf.

4.3 Germline Mutations

The germline mutations establishing the root's reference chromosome are created in two parts generating two VCF files, one for each allele. The number of mutations and the ratio of homozygous to heterozygous mutations are specified in the config file.

The first part of the simulation consists of creating and adding the homozygous mutations to both VCF files. To create a random mutation, a number from one to four is drawn from a uniform distribution representing the four nucleotides. Then an insertion place is drawn from a uniform distribution from zero to the length of the reference chromosome. To make sure this is a valid mutation, the simulation checks that the new nucleotide is actually different from the nucleotide in the reference and that this position has not been mutated in the simulation before. It also tests that the reference does not contain an N at the chosen position, indicating that the position cannot be sequenced.

The second part is about adding the heterozygous mutations, requiring that the created mutations be added only to one of the alleles. For that, the heterozygous mutations are generated in the same manner as the homozygous mutations. The only difference is that for each mutation the allele to which to add the mutation is chosen randomly.

4.4 Somatic Mutations

Generating the specified number of somatic mutations following the built phylogenetic tree happens in two steps. The user-defined number of mutations is created in the same way as the heterozygous germline mutations. The chosen positions, alleles and changed nucleotides are stored in arrays. In the subsequent step all the mutations are added to the chromosomes following the phylogenetic tree. For each mutation an inner node from the tree is chosen according to a uniform distribution and then the mutation is added to this node and all its descendants. The leaves cannot acquire additional mutations leading to each mutation being contained in at least two cells. At the end, all mutations are written to a VCF file.

4.5 MDA and ART Read Simulation

For every allele, the MDA and ART read simulation is called separately, since these processes happen independently of each other.

The MDA simulation starts with partitioning the chromosome into bins of length l_i chosen from a negative binomial distribution with mean μ_{Bin} and standard deviation σ_{Bin}. Then for each bin its coverage is selected. The coverage is either zero with probability p_{zero} or is drawn from a negative binomial distribution with mean μ_{cov} and standard deviation σ_{cov} with probability $1 - p_{zero}$. This introduces additional zero coverages to the zero coverages created by the negative binomial distribution, making the distribution zero-inflated. The method mentioned in this paragraph simulates the exponential and non-uniform coverage of the MDA process. All the errors introduced during MDA (see Chapter 2.3), are saved in a VCF file for analysis purposes.

After the MDA simulation for a bin has finished, all copied sequences of that bin are combined in a FASTA file. These are supplied to ART Illumina. The reads generated by all the bins of one allele are combined into two FASTQ files, one for each read orientation.

5 Results

5.1 Data Preparation

To compare the simulation to real data, the simulation was applied to two chromosomes of the human reference genome hg38. Then, the coverage distribution was compared to single cell data obtained from the NCBI (National Center for Biotechnology Information) database.

The data set SRR617391 includes the whole genome sequencing of a single human cell. It was amplified with MDA and sequenced with Illumina, creating paired-end reads. The FASTQ files were checked for adapter contamination and if present, removed with Trimmomatic (Bolger, Lohse, & Usadel, 2014). Then, they were aligned to the human reference genome hg38 with bwa (Li, 2013) and duplicates were removed with samtools (Li, et al., 2009). Chromosomes 22 and 11 were extracted and the coverage per nucleotide was calculated with samtools depth (Li, et al., 2009).

For the simulation, a tree of one leaf (one cell) with zero germline and somatic mutations was generated. Afterwards they were aligned to the human reference genome hg38 with bwa (Li, 2013) and the coverage per nucleotide was calculated with samtools depth (Li, et al., 2009).

Because the coverage of the real data comes from single cell data, the extracted coverage per chromosome is the combined coverage of both alleles. Therefore, the coverages of the simulated alleles are added together before they are compared to the real data.

To show the impact of mutations and the software's ablities, the 'Single Cell Read Simulator' is used to simulate chromosome 22 in five cells with a number of somatic mutations. Then, an examination of the real mutations, MDA artefacts and Illumina sequencing is provided. Coverage statistics were computed for both real and simulated data and plotted.

5.2 Coverage

To compare chromosome-wide coverage of the simulated data to the real data, the parameters used for the 'Single Cell Read Simulator' were estimated with the R function optim and Stan (Carpenter, et al., 2017) from the real coverage of single cell read data. This analysis was done for chromosome 22 and for chromosome 11.

The parameters used in all three simulations are:

```
"meanBinSize": 97,
"standartDeviationOfBinSize": 1173,
"readLength": 100,
"meanFragmentSize": 300,
"standartDeviationOfFragmentSize": 30,
"MDAamplificationErrorProbability": 0.000001,
"ARTDropReadProbability": 0.05
```

For Figures 5.1, 5.2 and 5.3 the parameters were estimated with optim and a negative binomial distribution was chosen for the coverage.

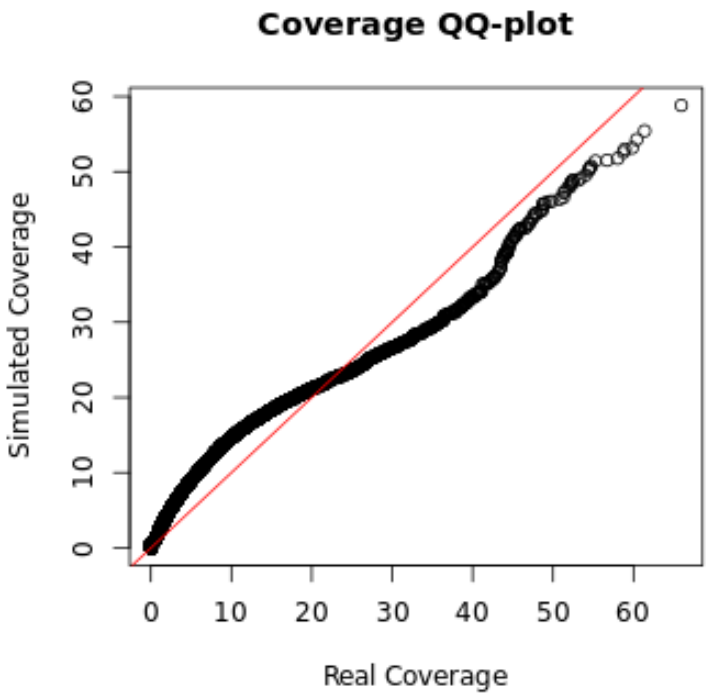

Figure 5. 1: QQ-plot of the real coverage compared to the simulated coverage with a negative binomial distribution. (Parameters estimated with optim)

Parameters:

"meanMDACoverage": 0.5080,

"standartDeviationOfMDACoverage": 1.8277,

"zeroInflatationProbability": 0,

"ARTcoverage": 1

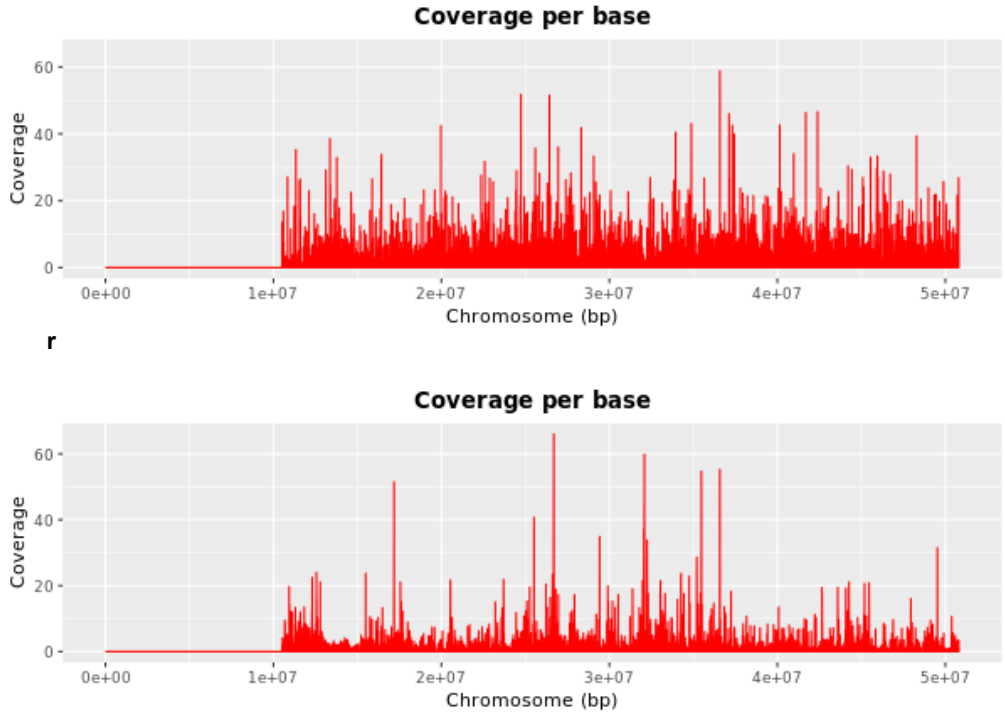

Figure 5. 2: Top: Simulated data. Bottom: Real data. (Same parameters as Figure 5.1)

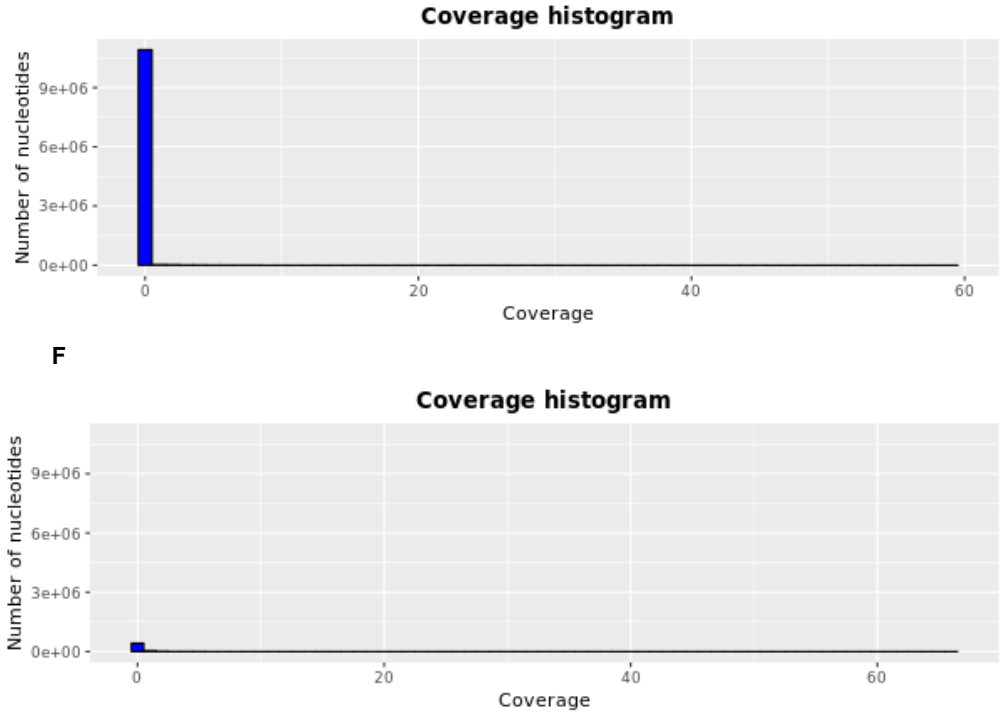

Figure 5. 3: Top: Simulated data. Bottom: Real data. (Same parameters as Figure 5.1)

In the first plot (Figure 5.1), we can see the QQ-plot of the real data compared to our simulated data. If we managed to get the same number of each coverage depth as the real data, the data points would lie on the diagonal (red line). The plot shows that we have a slightly higher number of small coverages and a slightly lower number of high coverages. This can be seen on the coverage histograms as well (Figure 5.3). There we also observe that we estimate a lot more zero coverages than the real data. The negative binomial distribution seems to be able to account for the large variance, but has more values that have coverages between zero and five.

To investigate the effect of different parameters, they were also estimated with STAN. The plot below shows chromosome 22 with the same negative binomial distribution for the coverage with the parameters estimated from STAN.

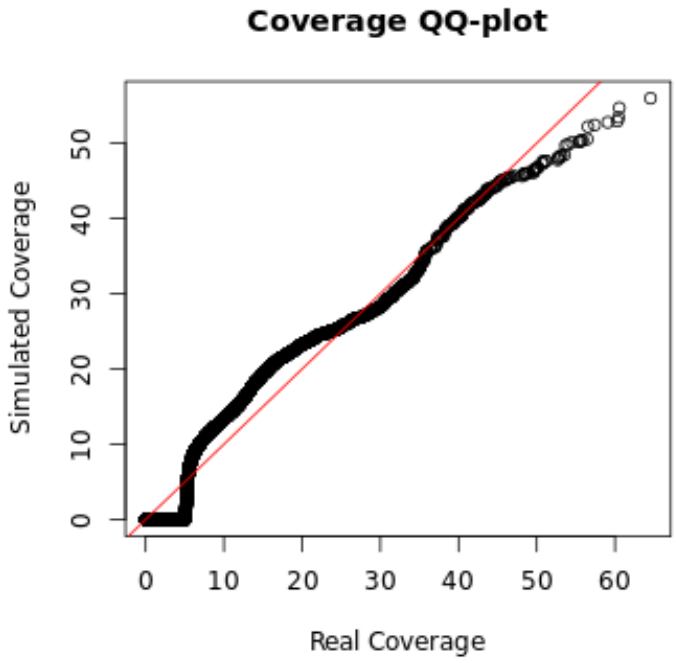

Figure 5. 4: QQ-plot of the real coverage compared to the simulated coverage with a negative binomial distribution. (Parameters estimated with STAN)

Parameters:

"meanMDACoverage": 0.011904762,

"standartDeviationOfMDACoverage": 0.121405227,

"zeroInflatationProbability": 0,

"ARTcoverage": 11

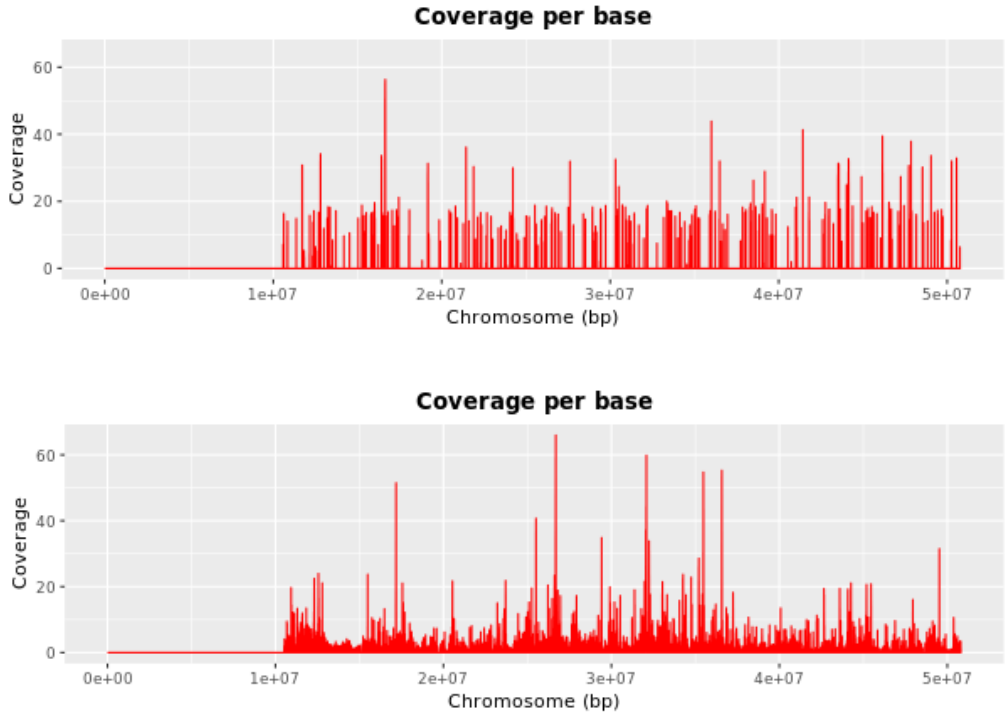

Figure 5. 5: Top: Simulated data. Bottom: Real data. (Same parameters as Figure 5.4)

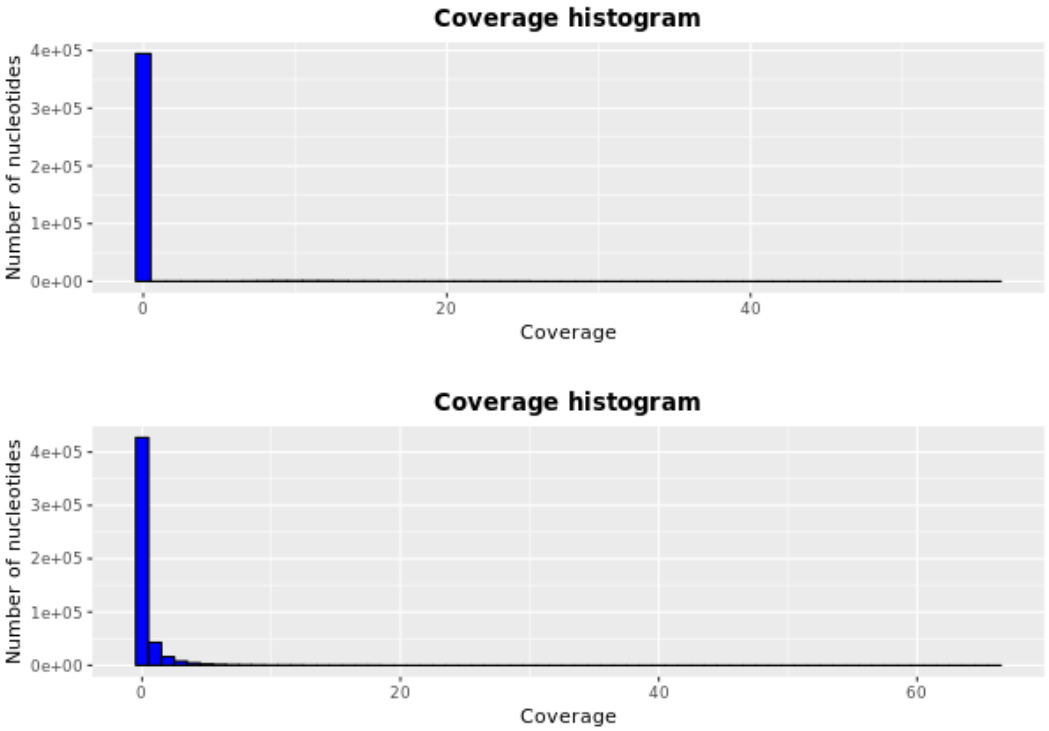

Figure 5. 6: Top: Simulated data. Bottom: Real data. (Same parameters as Figure 5.4)

The QQ-plot in Figure 5.4 shows that with the STAN parameters we get a better fit than before. But we see a lot less small values in the simulated data than in the real data. This could be explained with the fact that we estimate the coverage of each allele separately and add them together for the overall coverage. That means that even though one allele could have a small coverage, it is unlikely that the other allele will also have a small coverage exactly at the same place. This could explain why we observe less small values for the simulated data than in the real data. The negative binomial cannot quite account for the high number of zero coverages and the shape of the distribution (which has a large variance) at the same time. This is also visible in the coverage histogram, where the simulated data only achieves coverage depths up to 40, while the real data goes up to 60.

Since the negative binomial did not seem to fit perfectly, the zero-inflated negative binomial (ZINB) distribution was implemented. The idea is, that by decoupling the model for the zeros and the model for the non-zero values, we may get a better fit. Below are the plots showing the coverage of the simulated chromosome 22 with a ZINB distributed coverage and parameters estimated with optim.

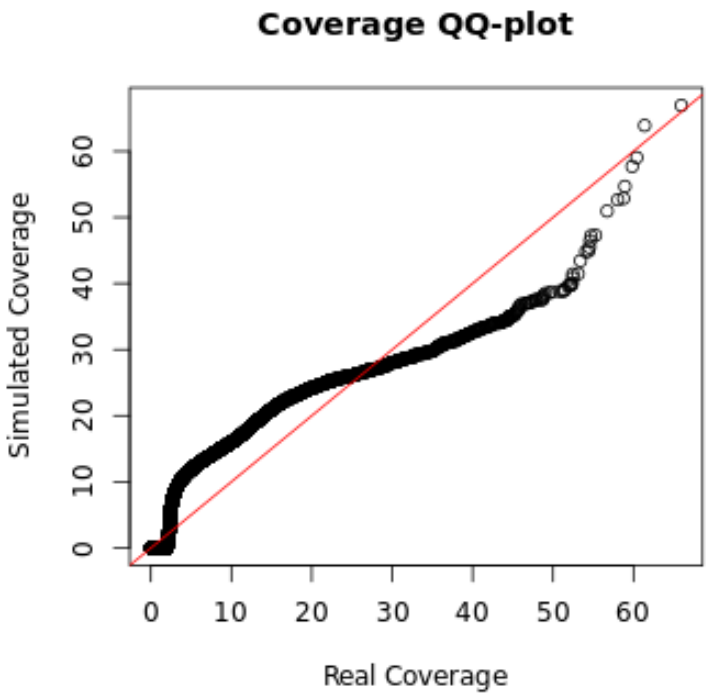

Figure 5. 7: QQ-plot of the real coverage compared to the simulated coverage with a zero inflated negative binomial distribution. (Parameters estimated with optim)

Parameters:

"meanMDACoverage": 0.035772771896924106,

"standartDeviationOfMDACoverage": 0.21144078115678,

"zeroInflatationProbability": 0.0895552,

"ARTcoverage": 11

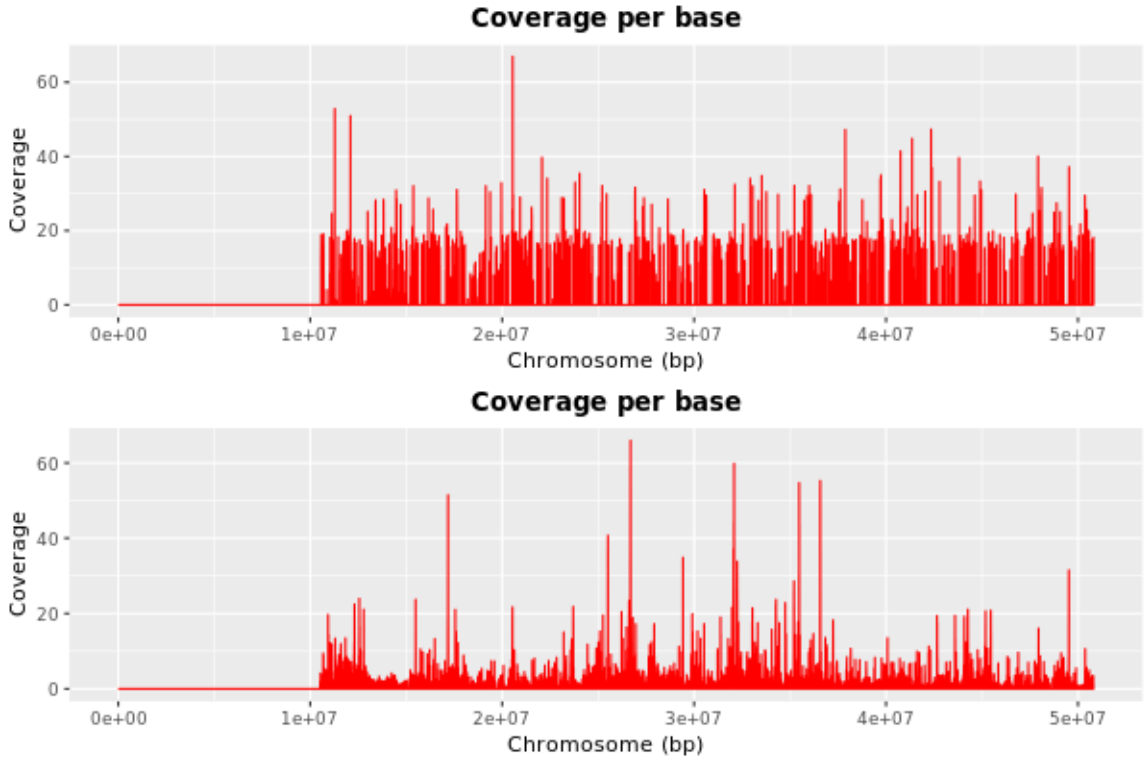

Figure 5. 8: Top: Simulated data. Bottom: Real data. (Same parameters as Figure 5.7)

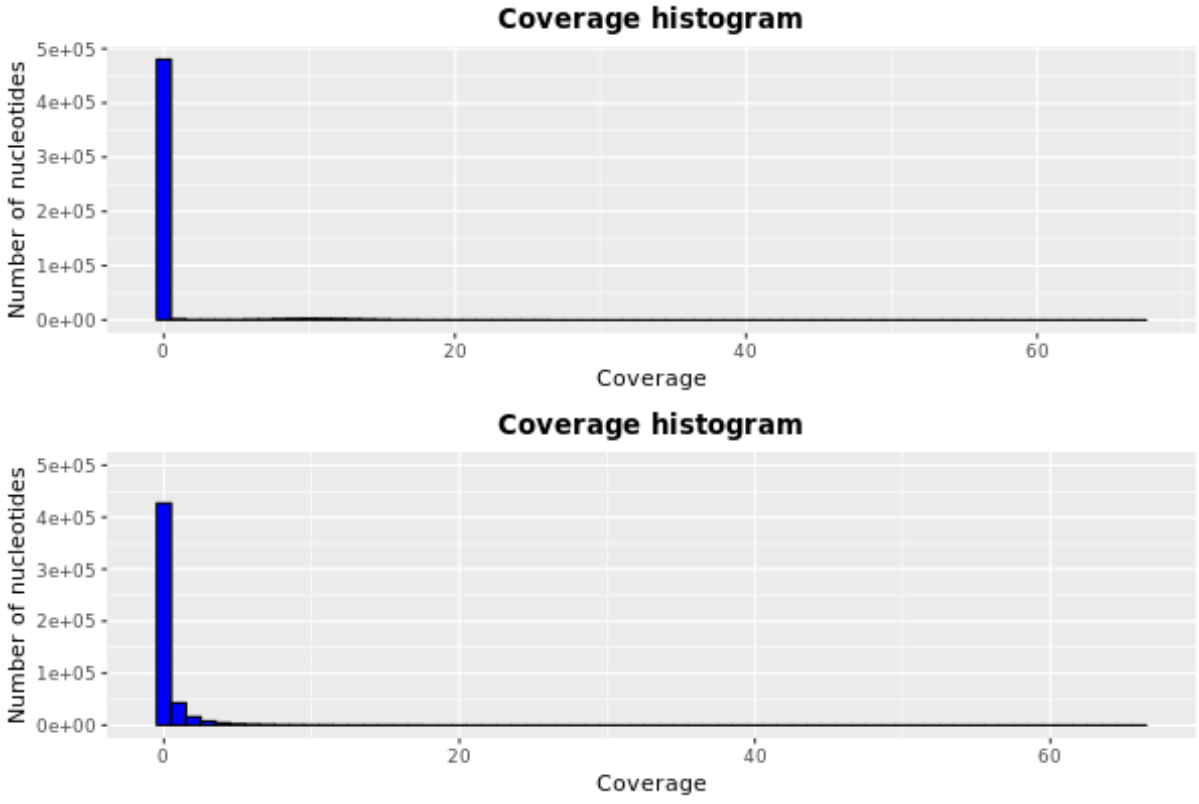

Figure 5. 9: Top: Simulated data. Bottom: Real data. (Same parameters as Figure 5.7)

If we look at the QQ-plot in Figure 5.7 and the coverage histogram in Figure 5.8, we do not observe a better fit with the ZINB distribution than with the negative binomial distribution. We manage to have a similar amount of zero coverages and variance, but in-between we have quite a lot of small values and less big values.

Overall, the best fit, if we look at the QQ-plot, seems to be a negative binomial distribution with the parameters estimated with STAN (as seen in Figure 5.4).

Nevertheless, the ZINB distribution was not removed from the simulation, since for some other data it could still be a better fit and setting the zero inflation probability to zero can still simulate the negative binomial distribution.

The ‘Single Cell Read Simulator’ was also tested for chromosome 11. The same data preparation as for chromosome 22 was done. The parameters used were the same as for chromosome 22 except for the mean bin size, the variance of the bin size and the parameters for the negative binomial distribution. The first two were estimated with R and the latter again with STAN.

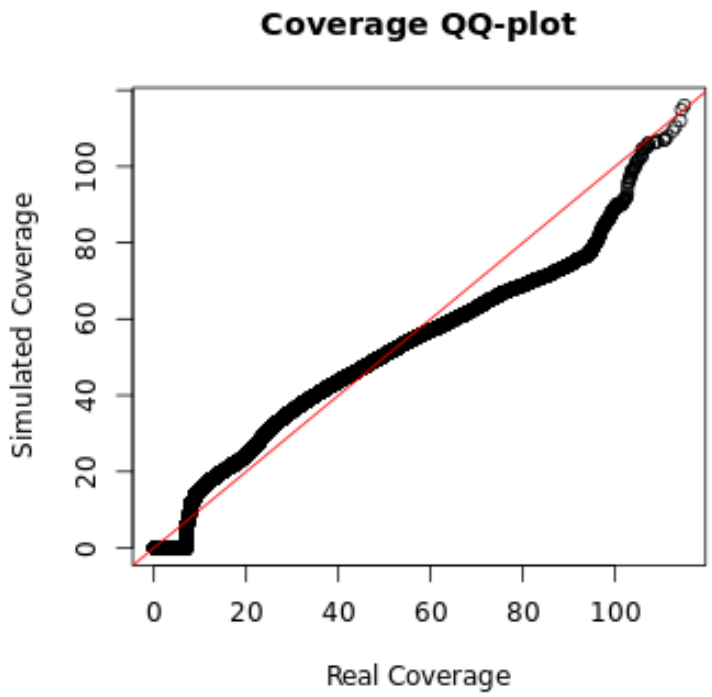

Figure 5. 10: QQ-plot of the real coverage compared to the simulated coverage with a negative binomial distribution. (Parameters estimated with STAN)

Parameters:

“meanBInSize”: 37,

“standartDeviationOfBinSize”: 410,

"standartDeviationOfMDACoverage": 0.181793834,

"zeroInflatationProbability": 0,

"ARTcoverage": 11

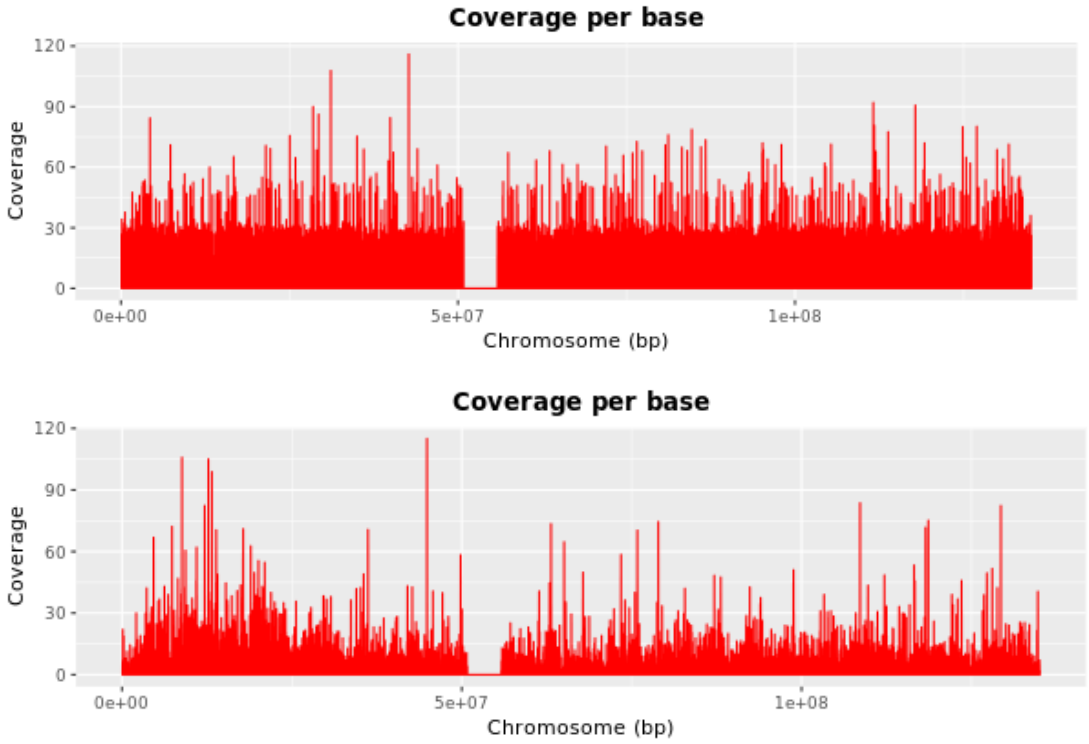

Figure 5. 11: Top: Simulated data. Bottom: Real data. (Same parameters as Figure 5.10)

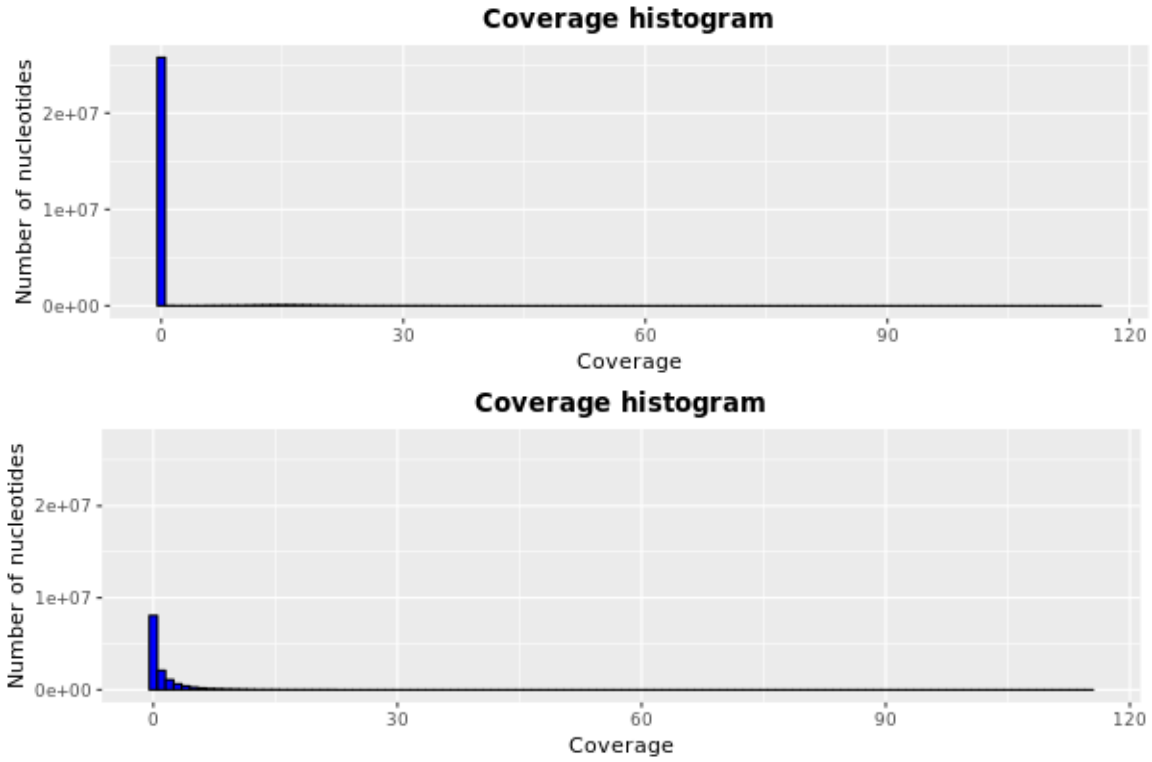

Figure 5. 12: Top: Simulated data. Bottom: Real data. (Same parameters as Figure 5.10)

Similar to chromosome 22, we can see in Figure 5.10 that we do not have many small values. This probably arises from the way we calculate the coverage from the two alleles. We also observe more zero coverages in the simulated data than in the real data (see Figure 5.12). Additionally, we observe fewer large values in our simulated data than in the real one.

In conclusion, it seems that the negative binomial can account for the high number of zero coverages non-amplifications and the big variance of coverage depths. However, the fit for intermediate values requires improvement.

Since we can simulate both alleles of a chromosome, we can also estimate the individual coverage contributions. This can be useful for analyses in which dropout events play an important role. Below are the two plots showing the allelic coverages from the simulated data of chromosome 22 and 11.

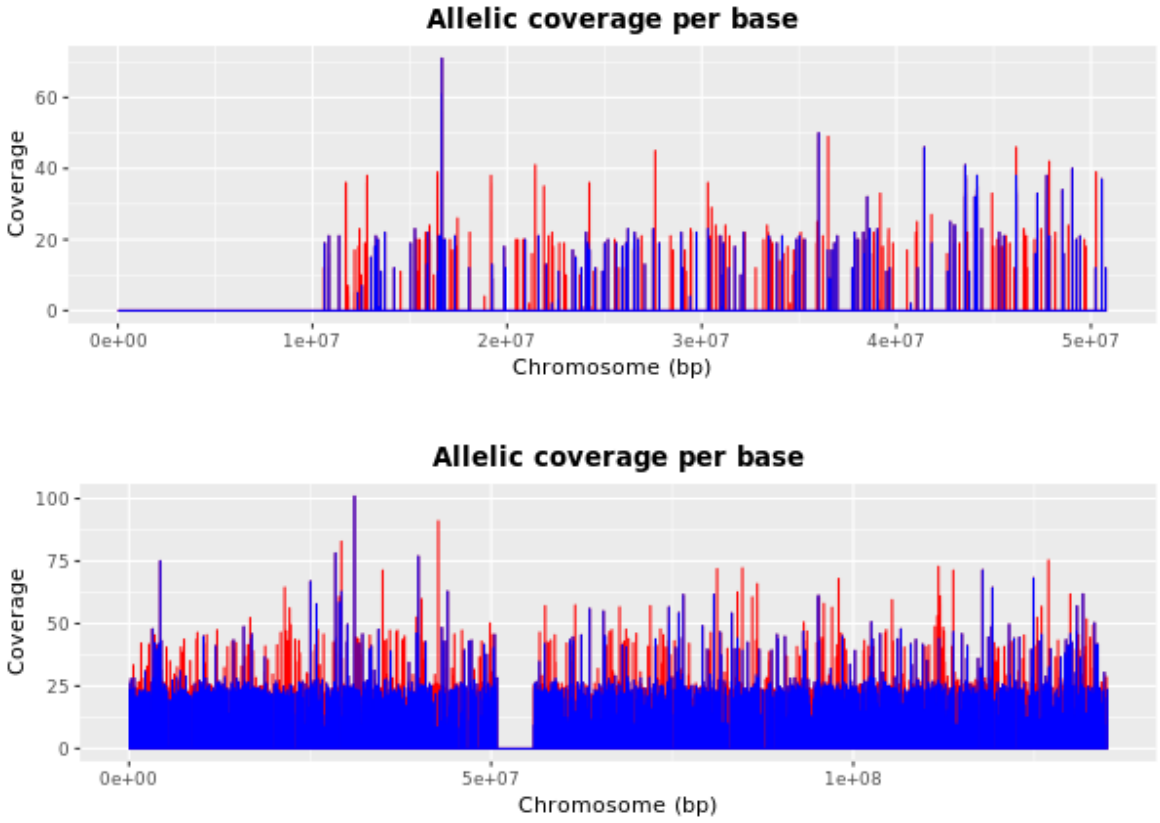

Figure 5. 13: Allelic coverage, where blue indicates allele 1 and red allele 2.
Top: Allelic coverage of the simulated chromosome 22. (Same parameters as Figure 5.4)
Bottom: Allelic coverage of the simulated chromosome eleven (Same parameters as Figure 5.10)

5.3 Errors

The simulation's main purpose is to simulate the process of single cell read generation. But it also takes the history of cells into account to be able to help with testing mutation-detecting algorithms. To this end, a few graphs are presented below, showing the abilities of the simulator.

Simulated is chromosome 22 of five cells (ten alleles) with 100 and 200 somatic mutations. The rest of the parameters are the same as in Figure 5.4.

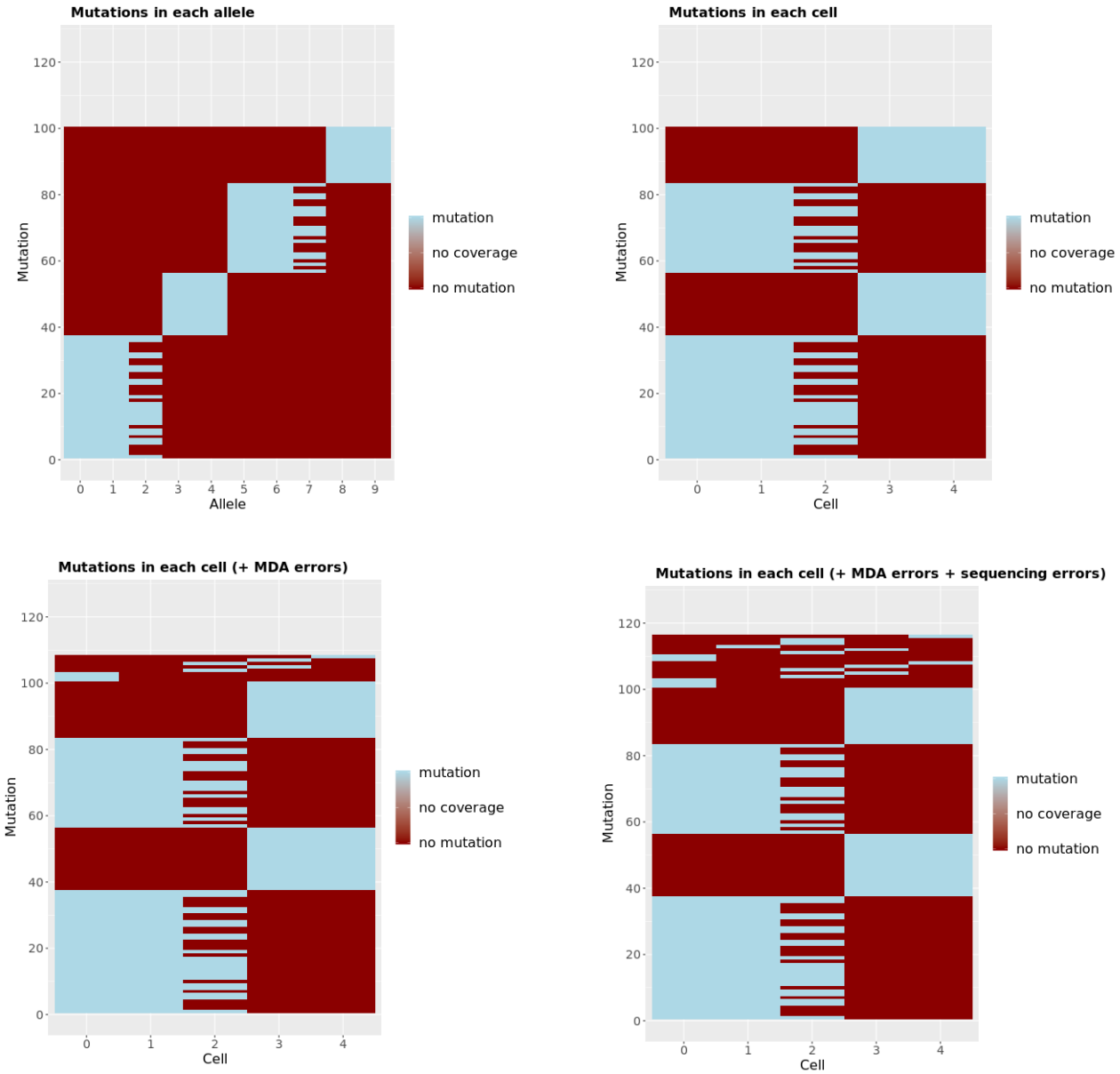

Figure 5. 14: Five cells containing 100 mutations from cell division. Top: On the left, the mutations each allele contains are shown and on the right the mutations a cell contains in total are shown. Bottom: On the left, the mutations and the MDA errors of a cell are shown and on the right the mutations, MSA errors and the sequencing errors are shown.

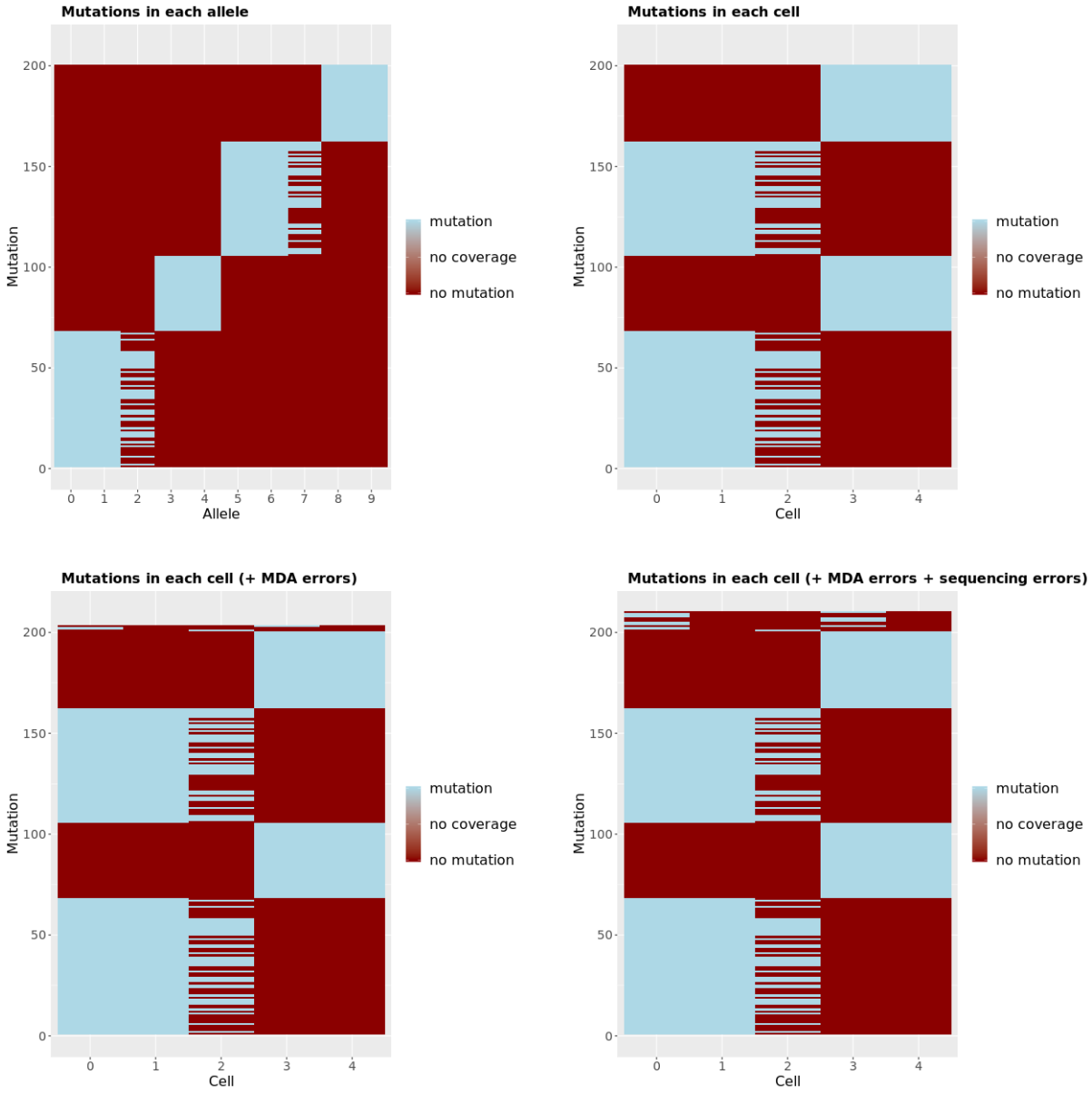

Figure 5. 15: Five cells containing 200 mutations from cell division. Top: On the left, the mutations each allele contains are shown and on the right the mutations a cell contains in total are shown. Bottom: On the left, the mutations and the MDA errors of a cell are shown and on the right the mutations, MSA errors and the sequencing errors are shown.

The four plots in Figures 5.14 and 5.15 show which mutations each cell has. In the first plot, only the somatic mutations are shown for each allele. In the next one, the somatic mutations each cell contains are shown. A cell contains the mutation if at least one of the two alleles contains the mutation. For the last two, the MDA errors and then the sequencing errors are added.

Light blue indicates that a cell or allele contains the mutation indicated on the y-axis. We can see some structures on these plots like the light blue square in Figure 5.14 on the bottom left. This shows that cell zero and one have the same 50 mutations, indicating that they probably have the same parent in the phylogenetic tree.

The plots on the bottom of Figure 5.14 show the errors occurring during the amplification and simulation process. There is considerably less structure to these errors than to the real mutations, since they occur randomly and independent of the different cells.

Figure 5.15 shows similar structures for the somatic mutations as Figure 5.14, but with a bigger number of mutations. Again, we can observe that the MDA and sequencing errors do not have a structure to them.

The simulation is easily adaptable to different amounts of errors and mutations. It can, therefore, be used for analysing different scenarios.

6 Conclusions and Outlook

The main strength of the ‘Single Cell Read Simulator’ is that it can simulate data for a set of related tumour cells, with MDA artefacts affecting both coverage and mutations. Additionally, it provides an interface for simulating tumour data and can, therefore, be useful for testing and benchmarking of algorithms concerning such data.

The ‘Single Cell Read Simulator’ still needs some improvement concerning the coverage distribution. That could be due to factors that are not currently taken into account in this MDA model, like GC content. We also learn the parameters of the coverage distribution of the sum of both alleles, but simulate each allele independently and then compare the coverage sum to the real data. The runtime of the ‘Single Cell Read Simulator’ is still a bit slow and that is why only two chromosomes, one of small length and one of medium length, were simulated so far.

To further improve the simulator, one could implement overlapping bins to produce smoother coverage. It would also be a good idea to change the way parameters are estimated so that we can correctly compare real data with simulations. We could also let the simulation handle the parameter estimations by extending it to firstly estimate the needed parameters from some user defined learning data and then run the simulation with the estimated values. We could also account for additional covariates, like GC content, to get more realistic distributions. For some applications it would make sense not to introduce random somatic mutations, but select them from a list of know mutations present in cancer cells.

The runtime is a bit slow because it has to simulate two whole chromosomes for each cell, which then are amplified even further and have to be fed ART. This can be mitigated to some extend by running the simulation in parallel using the snakemake flag ‘ –j ‘ when running it. Further performance optimizations may make sense.

7 Acknowledgments

I firstly want to thank my two amazing supervisors Dr. Jochen Singer and Dr. Francesco Marass. They were incredibly supportive, always making time for a meeting and helping me out. And special thanks to Francesco for reading through my thesis. I also want to thank Prof. Dr. Niko Beerenwinkel for giving me this opportunity to investigate this interesting topic. I enjoyed working on the simulator immensely and learned a lot.

Secondly, I want to thank my boyfriend for listening to my problems and motivating me throughout. And lastly, I want to thank my family for their great support and for believing in me.

8 References

1. Bolger, A. M., Lohse, M., & Usadel, B. (2014). Trimmomatic: a flexible trimmer for Illumina sequence data. *Bioinformatics* , 30(15): 2114-2120.
2. Carpenter, B., Gelman, A., Hoffman, M. D., Lee, D., Goodrich, B., Betancourt, M., et al. (2017). Stan: A probabilistic programming language. *Journal of Statistical Software* , 76(1): 1-32.
3. Cock, P. J., Antao, T., Chang, J. T., Chapman, B. A., Cox, C. J., Dalke, A., et al. (2009). Biopython: freely available Python tools for computational molecular biology and bioinformatics. *Bioinformatics* , 25(11): 1422-1423.
4. Dean, F. B., Hosono, S., Wu, X., Sun, Z., Bray-Ward, P., Sun, Z., et al. (2002). Comprehensive human genome amplification using multiple displacement amplification. *Proc Natl Acad Sci USA* , 99: 5261-5266.
5. Gawad, C., Koh, W., & Quake, S. R. (2016). Single-cell genome sequencing: current state of the science. *Nature Reviews Genetics* , 17: 175-188.
6. Grün, D., & van Oudenaarden, A. (2015). Design and Analysis of Single-Cell Sequencing Experiments. *Cell* , 163(4): 799-810.
7. Huang, W., Li, L., Myers, J. R., & Marth, G. T. (2012). ART: a next-generation sequencing read simulator. *Bioinformatics* , 28(4): 593-594.
8. Jahn, K., Kuipers, J., & Beerenwinkel, N. (2016). Tree inference for single-cell data. *Genome Biology* , 17(1): 86-103.
9. Köster, J., & Rahmann, S. (2012). Snakemake - a scalable bioinformatics workflow engine. *Bioinformatics* , 28(19): 2520-2522.
10. Li, H. (2013). Aligning sequence reads, clone sequences and assembly contigs with BWA-MEM. *arXiv preprint* , arXiv:1303.3997.
11. Li, H., Handsaker, B., Wysoker, A., Fennell, T., Ruan, J., Homer, N., et al. (2009). The Sequence Alignment/Map format and SAMtools. *Bioinformatics* , 25(16): 2078-2079.
12. Python Software Foundation. (2019, 07 22). *Python Software Foundation [US] : random — Generate pseudo-random numbers*. Retrieved 07 22, 2019 from https://docs.python.org/3/library/random.html

13. Sampson, J., Jacobs, K., Yeager, M., Chanock, S., & Chatterjee, N. (2011). Efficient Study Design for Next Generation Sequencing. *genet Epidemiol* , 35(4): 269-277.
14. Singer, J., Kuipers, J., Jahn, K., & Beerenwinkel, N. (2018). SCIΦ: Single-cell mutation identification via phylogenetic inference. *Nature Communications* , 9(1): 5144.
15. Tagliavi, Z., & Draghici, S. (2012). MDAsim: A multiple displacement amplification simulator. *IEEE International Conference on Bioinformatics and Biomedicine* , 1-4.
16. The SciPy community. (2019, 01 31). *SciPy.org : numpy.random.negative_binomial*. Retrieved 07 22, 2019 from https://docs.scipy.org/doc/numpy/reference/generated/numpy.random.negative_binomial.html
17. Wang, Y., & Navin, N. E. (2015). Advances and Applications of Single Cell Sequencing Technologies. *Mol Cell* , 58(4): 598-609.

9 List of Abbreviations

NGS	Next Generation Sequencing
WGA	Whole-Genome Amplification
MDA	Multiple Displacement Amplification
SCS	Single-Cell Sequencing
dNTP	Deoxynucleotide Triphosphate
ADO	Allelic Dropout
ZINB	Zero Inflated Negative Binomial

10 List of Tables

11 List of Figures

A Appendix

A.1 Probability Distributions

Negative binomial distribution:

$$P(X = n) = \binom{n + r - 1}{n} p^r \, (1 - p)^n$$

n : Number of failures

r : Number of successes or overdispersion

p : Probability of one success

Parameterization of negative binomial distribution in terms of mean μ and variance σ^2:

$$p = \frac{\sigma^2 - \mu}{\sigma^2}$$

$$r = \frac{\mu^2}{\sigma^2 - \mu}$$

σ^2 : Variance

μ : Mean

r : Number of successes or overdispersion

p : Probability of one success

A.2 Random Numbers in Python and Numpy

Uniform distribution:

- `random.randrange(start, end + 1, step)`
- `random.randint(start, end)`

(Python Software Foundation, 2019)

Negative binomial distribution:

- `np.random.negative_binomial(r, (1 - p))`
  (The SciPy community, 2019)

Probability p:

- `random.choices(population = […], weights = [(1 - p), p], k)`
  (Python Software Foundation, 2019)

A.3 Genome Data

The sequence data used can be found in the 'Sequence Read Archive' from NCBI under the accession number SRR617391.

The reference chromosomes used for simulation can be obtained from the UCSC (University of California Santa Cruz) under the name GRCh38/hg38.

A.4 Single Cell Read Simulator

The 'Single Cell Read Simulator' software can be cloned from gitlab:
https://gitlab.ethz.ch/leonors/bachelor-thesis.git

A README guiding through the necessary steps of the installation and the deployment of the software is available.