Bielefeld University

Master Thesis

Informatics in the Natural Sciences

Unraveling Overlapping Insertions with Agglomerative Clustering

Author:
Maureen Smith

Supervisors:
Dr. Roland Wittler
Prof. Dr. Jens Stoye

A thesis submitted in fulfilment of the requirements
for the degree of Master of Science

in the

AG Genome Informatics
Faculty of Technology

November 2013

Erklärung

Hiermit versichere ich, Maureen Smith, diese Arbeit selbstständig und lediglich unter Benutzung der angegebenen Quellen und Hilfsmittel verfasst zu haben. Ich erkläre weiterhin, dass die vorliegende Arbeit noch nicht im Rahmen eines anderen Prüfungsverfahrens eingereicht wurde.

Unterschrift: _____

Datum: _____

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

It is known that structural variants in the human genome are not only playing an important role for the genome diversity. They can also be involved in the development of genetic diseases, for example cancer. Several approaches exist to detect such structural variants with Next-Generation Sequencing data, such as paired-end mapping. The simultaneous analysis of different samples, in order to compare structural variants and distinguish those associated with genetic diseases from individual mutations, is an important challenge in bioinformatic research.

Wittler [1] introduces aggloDel, a method to detect and unravel overlapping deletions within different samples. With a paired-end mapping approach, mappings are determined which potentially span deletions. These mappings are clustered via agglomerative clustering, regarding their similarity of position and size of the represented deletion. A cluster can be modelled geometrically (in a coordinate system as a triangle), representing these criteria based on the geometric approach from Sindi et al. [2], as shown exemplarily in Figure 1.1. The probability for the range of possible deletion sizes in the third dimension allows to create a certain volume for each cluster. A similarity score is determined by the normalised intersecting volume of two clusters. Iteratively, the most similar clusters are merged until only one cluster is left or a certain score threshold cannot be outreached. If two clusters disagree too much, they form two separate clusters.

In this work aggloIns is presented, an extension of aggloDel with the purpose of finding insertions with the same clustering approach, adapted for insertion clustering. As insertion clustering shows a behaviour different from that of deletion clustering, several parameters and filter settings have to be adjusted and evaluated.

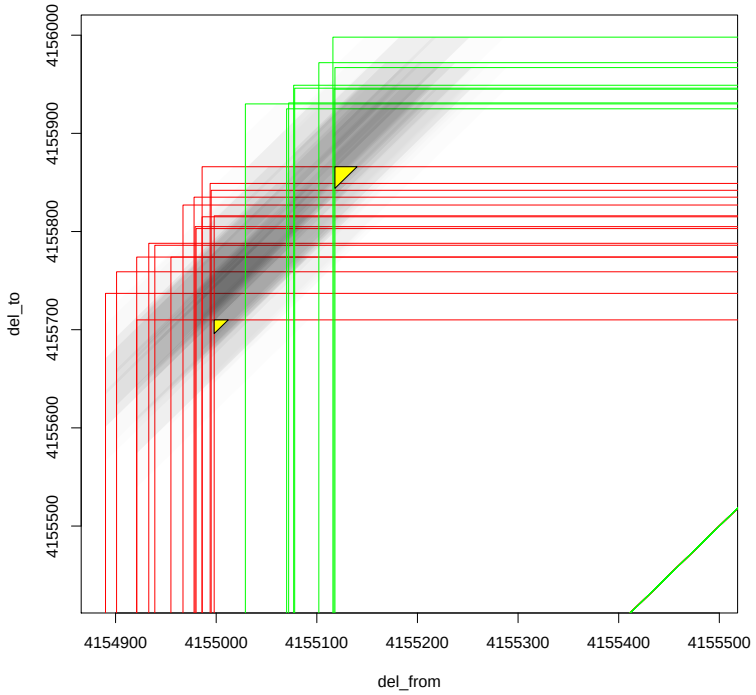


Figure 1.1: Deletion visualisation with aggloDel. The x-axis and y-axis indicate the breakpoint positions. The grey shades imply the possibilities for the deletion sizes in the third dimension. The yellow triangles show predicted deletions.

1.2 Structure of the Thesis

This work consists of four parts. The second chapter, containing background knowledge, starts with an explanation of structural variants in association with diseases. It continues with the presentation of several methods to detect such structural variants and goes into detail about detecting insertions with paired-end mapping. Furthermore, tools and formats used in aggloIns are introduced. The third chapter deals with the modelling of the algorithm and gives an understanding of the clustering process. After that, the implementation of aggloIns is described. In the fourth chapter, results with simulated data are evaluated and benchmarked. Afterwards, aggloIns is applied to a real data set and the results are presented. Conclusions are given in the fifth and last chapter.

Chapter 2

Background

This chapter gives first a short biological background about structural variants in connection with genetic diseases, especially cancer, which is one main motivational aspect for detecting overlapping insertions. The second section gives an overview of different computational methods to detect structural variants without assembling the whole genome. A more precise explanation for finding insertions with the so called paired-end mapping approach is found in the third section. The last section contains information of formats and tools used for aggloIns.

2.1 Structural Variants in Cancer Cells

Genomic variants in the human genome occur in different ways and are not only relevant for genome variation and the diversity of individuals, but also cause genetic diseases like cancer. These variants can be single nucleotide variants (SNVs), length polymorphisms of microsatellite sequences and bigger structural variants (henceforth referred to as SVs) [3]. SVs include duplications, insertions and deletions and more complicated rearrangements like inversions or interchromosomal recombination [1, 3, 4]. Insertions and deletions bigger than 1 kb are called copy number variants (CNVs). Smaller variants are known as indels.

SNVs are the most frequent genetic variants, but more and more larger SVs are detected and brought into relation with human diseases [3–5]. Some of these diseases, like diabetes or heart diseases, are induced by DNA variants in the *germline*, which means they are consequently inherited. Other genetic ailments evolve through lifetime and emerge individually in different tissues. These *somatic* modifications can be evoked by various factors like UV radiation or mutagenic substances. Cancer emerges because of somatic

mutations, which are passed to daughter cells during mitosis. Raphael [4] refers to cancer as a "microevolutionary process within a population of cells because of selection for advantageous mutations".
There are different types of cancer, which vary in their structural variations. These variations lead to activated oncogenes, inactivated tumor-suppressor genes or their altered expression. Mechanisms for the cancer development are, for example, mutations in the promotor region (and increasing or decreasing the expression), mutations in the gene (and inactivating it) or fusion genes, created by translocation. In healthy cells, these oncogenes are responsible for cell growth, so called proto oncogenes. Tumor suppressor genes are responsible for the regulation of the cell growth. If this regulation balance is disturbed, uncontrollable cell growth, the tumor formation, occurs [6].

Recurrent chromosomal rearrangements have been found in many patients with the same disease pattern, for example leukaemia or lymphoma [4]. Fusion genes, which arise because of translocation of different genes, can lead to cancer, as discovered among others in chronic myelogenous leukaemia (CML) [4, 7]. In this example, one of the affected genes, the ABL gene on chromosome 9, encodes tyrosine kinase and plays an important role in the regulation of cellular growth. The other gene, BCR (break point cluster region) is located on chromosome 22. If the fusion gene ABL-BCR is formed, the ABL gene functionality is disturbed and tyrosine kinase is permanently activated. The combined gene acts as an oncogene, thus results in increased and uncontrolled cell growth.

However, the structural variations vary very often, although the disease patterns are the same. One example for this is the somatic mutation of the epidermal growth factor receptor (EGFR) in non-small-cell lung cancer as shown in Figure 2.1. This protein is a member of the ErbB family, a subfamily of four closely related receptor-tyrosine kinases [8]. The classical variations which activate the EGFR are in-frame deletions in exon 19 or a point mutation in exon 21, which are sensitive to reversible EGFR tyrosine kinase inhibitors (TKIs) and so the tumor growth can be treated. Nevertheless, about 4% of the EGFR mutations contain insertions in exon 20 and preclinical and clinical data has shown a resistance to EGFR TKIs for the most of the concerned proteins [9, 10]. Thus, it is not possible to determine the right treatment without knowing the exact EGFR mutation as the reason for this lung cancer.
The last example points out that there is a need for comparative analysis of SVs of different tissues, samples or treatment stages.

The next sections present methods to determine SVs and put a finer point on the detection of insertions with paired-end mapping.

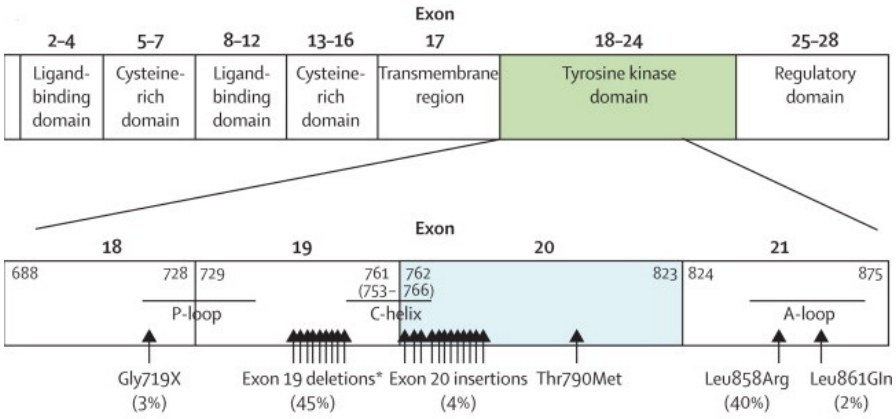


Figure 2.1: Epidermal growth factor receptor (EGFR) mutations. The arrows show the somatic mutations within the tyrosine kinase domain, which belongs to the EGFR exon region. The most prevalent SVs are deletions in exon 19 (45%) and the point mutation *Leu858Arg* in exon 21 (40%). About 4% are insertions in exon 20. The remaining SVs are other point mutations. (Graphic obtained from [10])

2.2 Detection of Structural Variants

There exist many different possibilities for SV identification, where the main idea is to compare a given *donor genome* containing putative mutations with a *reference genome*. On the one hand, there are several experimental approaches, like aCGH (**a**rray-based **C**omparative **G**enomic **H**ybridization) [2, 11]. On the other hand, it is common to use computational methods, which will be discussed more detailed now.

The plainest approach would be to align both genomes after assembling the whole donor genome. In order to save time and cost caused by the genome assembly and its costly finishing steps, a better option is to consider only those reads for an assembly of genome regions, where structural variants are suspected [12]. Another way to find differences is to completely spare the assembly and only align the reads of the donor to the reference genome, a so called *mapping*, and make conclusions by the behaviour of the alignments [1, 13].

There are three common types of algorithms finding SVs: Some solutions, like MrFast [14] or CNV-seq [15], consider regions with an abnormal *coverage*. This means the number of aligned reads in this region is significantly higher or lower than the average of the genome. Algorithms that interpret the configuration of *paired-end mappings* to find SVs (explained shortly below and in detail in Section 2.3), are used, for example, in aggloDel [1], BreakDancer [16], VariationHunter [17] or Pindel [12]. A third method takes *split reads* into account, where reads can not be aligned completely, e.g. in SRiC [18].

All methods have their advantages and disadvantages and it makes sense to combine them and use the strong points of each. Sindi et al. [2], for instance, use both approaches of paired-end mapping and the coverage density for the detection tool GASV to find SVs.

Below follows a short introduction of the three techniques.

Coverage

The main idea is to find regions in the reference, where the read coverage significantly differs from the expected coverage after mapping the reads. The assumption of the expected coverage can be made on the average distribution of the reads.
These read density-based algorithms can detect only dosage-altering SVs like insertions or deletions and are not able to find copy-number neutral variants like inversions or translocations. They work best for big copy number changes (CNVs), so if parts of the genome are lost or duplicated. The bigger the CNV, the more sensitivity and specificity increase [2, 3].

Paired-End Mapping

For paired-end mapping (PEM) or end sequence profiling (ESP), DNA fragments of the donor genome are sequenced from both sides and aligned to the reference sequence. The read length, the approximate distance between the reads and their relative orientation is known. SVs are determined by comparing orientation and distance of the paired reads after mapping them to the reference sequence. A SV is considered, if a read orientation is opposite or the distance of the paired reads differs significantly. A more precise explanation follows in the next section.
With this technique it is possible to find inversions, translocations, transpositions, insertions and deletions.

Split Reads

An indication for a SV are reads which did not completely map to the reference genome. If two parts of one read map on the reference with a certain distance between these parts, there might be a possible deletion in the donor genome. In case a read spans a small insertion on the donor genome, parts of the read without the insertion sequence will map consecutively. Interchromosomal rearrangements could have occurred if reads are found which map in parts on different chromosomes.

The approach of aggloIns only uses PEM. So this method will be described more precisely for the detection of insertions below.

2.3 Detecting Insertions with PEM

Korbel et al. [19] introduced the method of paired-end mapping for detecting SVs. A sample genome (donor) is multiplied and fragmented. These fragments are sequenced from both sides of the double strand with a short-read sequencer like 454 or Illumina, resulting in a massive amount of paired reads. Figure 2.2 illustrates the sequencing process.

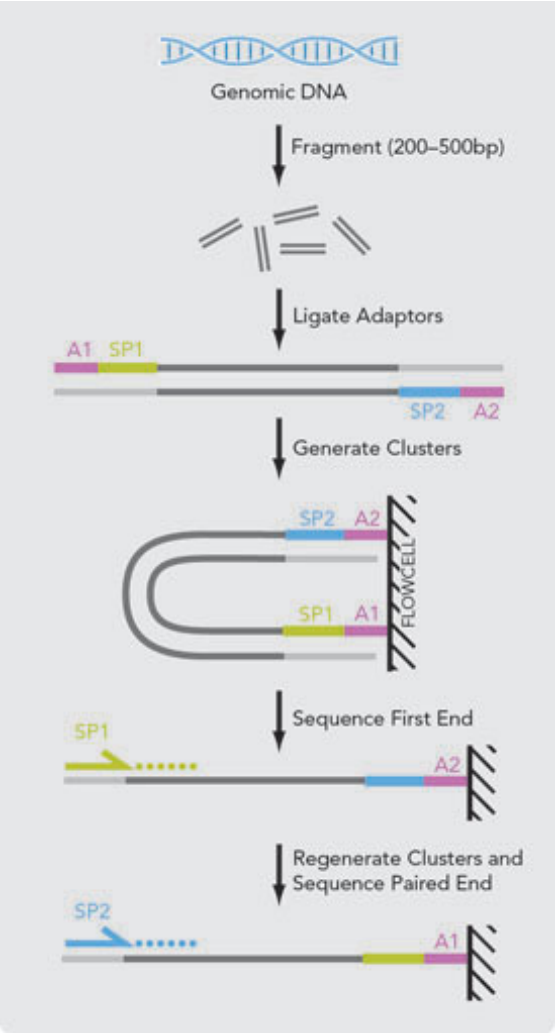


Figure 2.2: Paired end sequencing. First the genome is fragmented, adaptors are ligated and the fragments can be sequenced from both ends. (Graphic is used from [20].)

The read length and its relative direction on the double strand of the donor genome are known. The *fragment size* (Figure 2.3), spanned by the paired reads, is distributed around a certain length.

The *paired ends* are mapped to the reference sequence. A mapping is *concordant*, if the initial chromosome and direction on the donor sequence are retained on the reference and the distance between the read ends is still associated with the fragment size. If the reads are mapped on different chromosomes, their relative orientation is incorrect or the distance differs significantly from the initial fragment size, the mapping is called *discordant*. This case indicates a given SV in the donor genome (or errors during the mapping or assembling process).

The task of aggloIns is the detection of insertions. This means a sequence is inserted into the donor genome with regard to the reference, thus is only found in the donor sequence. The insertion position is also called *insertion breakpoint*. Those paired reads which span the whole insertion within the donor will be mapped to the reference with

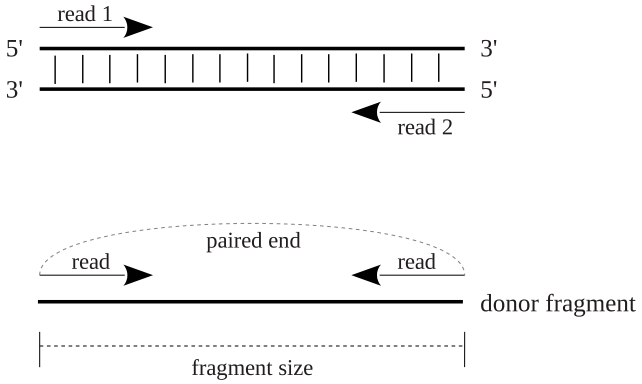


Figure 2.3: The two arrows represent the paired-end reads, which have a fixed size and direction within the donor genome. The fragment size is approximately known and spans the space between the outer ends of both reads.

the correct direction but with discordance referring to the fragment size: the paired-end mapping will span the insertion breakpoint with a distance shortened by the insertion size (Figure 2.4). We call such a mapping *abridged mapping*.

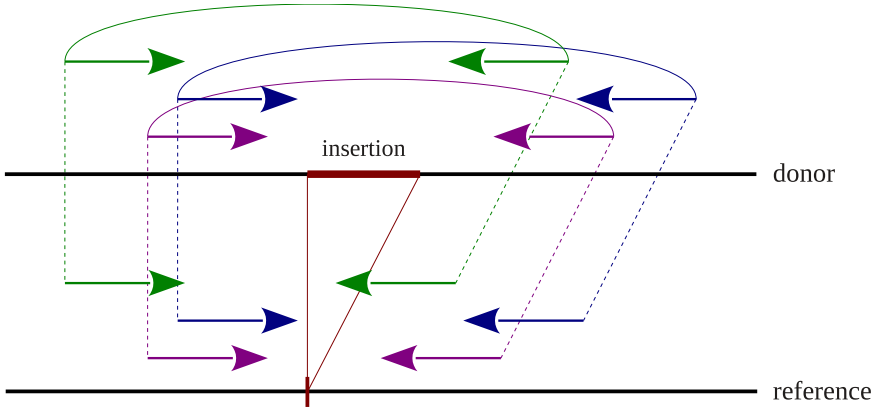
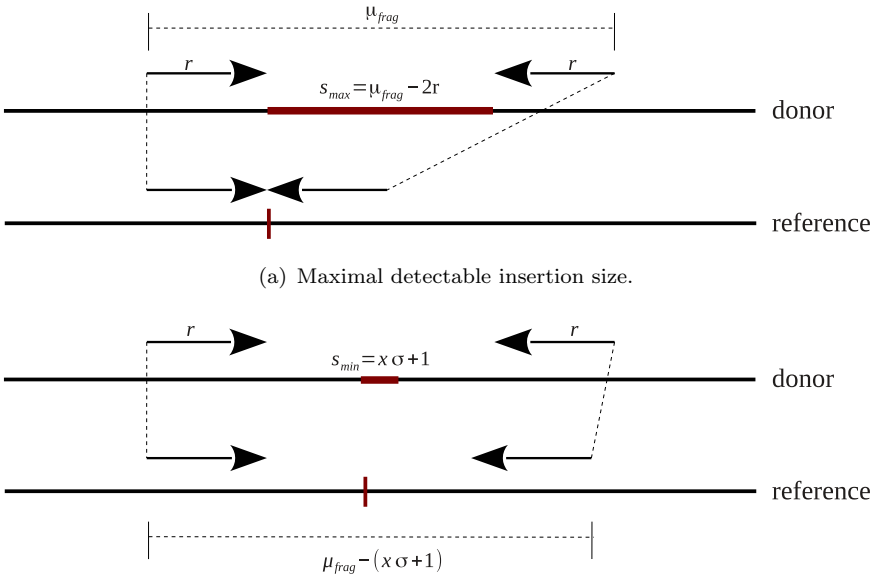


Figure 2.4: Paired-end mappings spanning an insertion. The distance of the paired reads in the reference mapping is shortened by the size of the insertion.

Because the original fragment size and hence the distance between the mapped reads is not exactly known, it is not possible to determine the exact insertion size. However, since the fragment size is assumed to follow the Gaussian distribution around a certain mean μ_{frag} (as seen in the distribution of fragment sizes in Figure 4.14 in Section 4.2) and the read length r is known, the expected mapping distance can be settled as normally distributed around mean $\mu_{\text{dist}} = \mu_{\text{frag}} - 2r$ with the standard deviation σ. A reference mapping can be considered as abridged, if the actual mapping distance $d < \mu_{\text{dist}} - x\sigma$, where $x > 0$ can be chosen freely. The insertion size s can therefore be approximated by

$$s = \mu_{\text{dist}} - d.$$

The size of detectable insertions is limited in both ways, shown in Figure 2.5.



(a) Maximal detectable insertion size.

(b) Minimal detectable insertion size.

Figure 2.5: The detectable insertion size via PEM is limited by the fragment size and the accuracy of the sequencing process.

The fragment size is a border for the maximal detectable insertion size, because both reads have to be mapped to the reference, which requires $\mu_{\text{frag}} - s \geq 2r$ (see Figure 2.5(a)). The maximal detectable insertion size is therefore

$$s_{\max} = \mu_{\text{frag}} - 2r = \mu_{\text{dist}}.$$

The minimal detectable insertion size depends on the accuracy of the sequencing process, i.e. the standard deviation. As a mapping with distance $d \geq \mu_{\text{dist}} - x\sigma$ is considered as concordant, the minimal detectable insertion size has to be

$$s_{\min} = x\sigma + 1,$$

shown in Figure 2.5(b).

This means, the PEM approach is only capable for the detection of medium sized insertions and gets more limited to the detectable insertion sizes with higher standard deviations.

2.4 Used Tools and Formats

2.4.1 SAM/BAM format and SAMTools

SAM (**S**equence **A**lignment/**M**ap) is a format standard for storing large DNA sequence alignments or mappings. It is not intended for the human reader, but for the simple processing with various tools. It can store all the information about the alignments from different programs and is easy to generate or converted from/to other alignment formats. It is possible to avoid loading the whole alignment into memory by streaming the information for most of the operations. To retrieve only reads that are mapped at a certain locus, it allows to index the file with the genomic position.
The format is also available in a binary version, the BAM format, to save disc space. It compresses the information in the Blocked GNU Zip Format (BGZF), which makes it still possible to access the indexed queries despite the packed format [21]. SAMTools provides several opportunities to view the data in a more comfortable format, filter and sort the output, do realignments or merge data sets. In this context the utility *view* is called while reading the BAM files in aggloIns to get the mapped and filtered read information. For the simulation of the data in the evaluation step, the utilities *view*, *sort* and *index* are called to convert the SAM files to BAM and keep the indexed loci [22].

2.4.2 Picard

Picard provides a JAVA API (SAM-JDK) to read, manipulate and write SAM and BAM files in own programs. AggloIns uses the API to read and filter the BAM files.
Furthermore, Picard offers a package with JAVA based command-line utilities. The contained jar files afford many opportunities to manipulate and convert SAM files. The tool SAMToFastq.jar is used for the conversion from BAM to FASTQ format in the data simulation workflow (Section 4.1.2). The SAM-JDK as well as the Picard tools can be downloaded on http://picard.sourceforge.net/.

Chapter 3

Methods

This chapter illustrates the modelling and implementation of the clustering process in aggloIns. The previously described abridged mappings are required as input and are aggregated to clusters, representing the insertions. The method is based on the model for the deletion clustering in aggloDel [1] with adjustments for the purpose of finding insertions. The modelling section gives an overview of the definition of clusters, their similarity scores, and the agglomerative clustering procedure. The implementation section describes the practical realisation and ends with the analysis of the time complexity.

3.1 Modelling

The model of the clustering process for overlapping insertions is built on the approach of Wittler [1] to find overlapping deletions with aggloDel.
A *region* within a genome may show up diverse insertions on different alleles or samples. If the insertion breakpoints are located very close to each other with respect to the reference, it is possible that the paired mappings span more than one insertion breakpoint. It gets difficult to compart the mappings and distinguish the insertions (Figure 3.1).
Insertion breakpoints are located in one region if they are so close, that their respective mappings might overlap. This means for both of two mappings, that the right end of the left read is located to the left of the inner end of the other mapping's right read. The aim of aggloIns is to approximate a solution by pooling similar mappings to one cluster and assign them to one insertion.
The algorithm used in aggloIns for this purpose is called *agglomerative clustering*. It is a bottom up approach which starts with each element of a set as a singleton cluster. These are merged successively following a score hierarchy until only one cluster is left or a certain score threshold is reached and no more clusters can be joined. The score

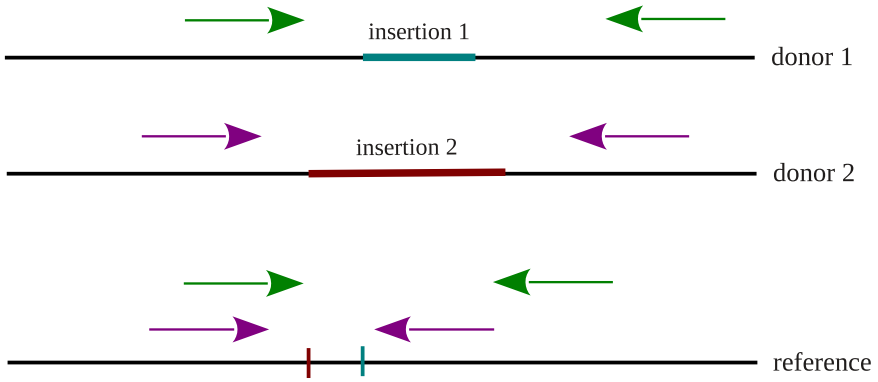


Figure 3.1: Two different alleles contain different insertions in the same region. The insertion breakpoints are so close that the mappings span both of them on the reference sequence. The challenge is to assign the mapped reads to the correct insertion.

defined here indicates the similarity with regard to location and size of the putative concerned insertion.
Below is explained how a cluster can be represented combining both criteria and making the required similarity score definition for the agglomerative clustering process possible.

3.1.1 Clusters

In the beginning of the clustering process, each discordant mapping suspecting an insertion is a singleton cluster. The attributes of a singleton cluster are determined by the characteristics of the mapping.
The position of the represented insertion lies between the paired reads of the mapping. A cluster has therefore a minimal and maximal position for the breakpoint, x_{min} and x_{max}.
The insertion size s is approximated by the expected fragment size less the actual mapping size: $s = \mu_{\text{frag}} - 2r - d$.
As the insertion size is not exact, just approximated by the mean fragment size, we assume the standard deviation of the fragment size σ as deviation for the insertion size σ_{ins} and by association the variation σ^2_{ins}. With this modelling, the insertion size follows a discrete normal distribution around s and each possible insertion size has a certain probability.

These attributes can be projected in a coordinate system, as seen in Figure 3.2: The x-axis indicates the position within the reference genome. x_{min} and x_{max} are the bounds for the possible insertion breakpoint site. The insertion size is specified on the y-axis. The probability for each insertion size is presented as a bellcurve in the third dimension (in Figure 3.2 indicated with the grey shaded area).

This geometric presentation allows to define another attribute for a cluster: the volume

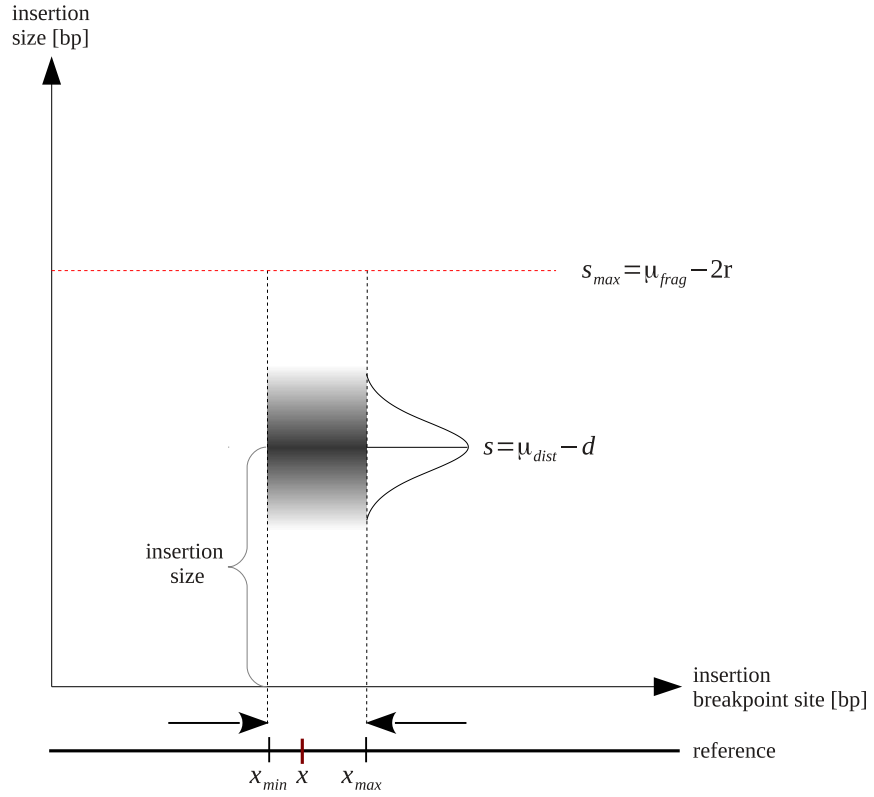


Figure 3.2: Mapping cluster presentation in a coordinate system.

V, defined by the interval for the possible breakpoint positions of length $x_{\max} - x_{\min} + 1$ and the area under the Gaussian bellcurve. The boundaries for the insertion size are limited by zero (as an insertion must have a size of at least 1) and the maximal detectable insertion size $s_{\max} = \mu_{\text{frag}} - 2r$, discussed in Section 2.3. The volume combines both criteria of position and length of an insertion and can be used for the measuring of the similarity between two clusters.

3.1.2 Score

The score for the clustering process has to assess the similarity of two clusters referring to position and size of the predicted insertion. These criteria are represented by the volume of the geometric model of an insertion, as seen in the previous section. The more two cluster volumes overlap, the more they have in common. Therefore, the similarity score S of two clusters A and B is given by the intersection of the two cluster volumes V, normalised by the bigger volume:

$$S(A,B) = \frac{V(A) \cap V(B)}{\max(V(A), V(B))}$$

This score is between zero and one and increases the more they agree. The intersection volume is given by the overlapping area of both normal distributions and the intersecting breakpoint position range, given by the distance between the inner most insertion position extrema, $\min(x_{\max_A}, x_{\max_B}) - \max(x_{\min_A}, x_{\min_B})$, as seen in Figure 3.3. The

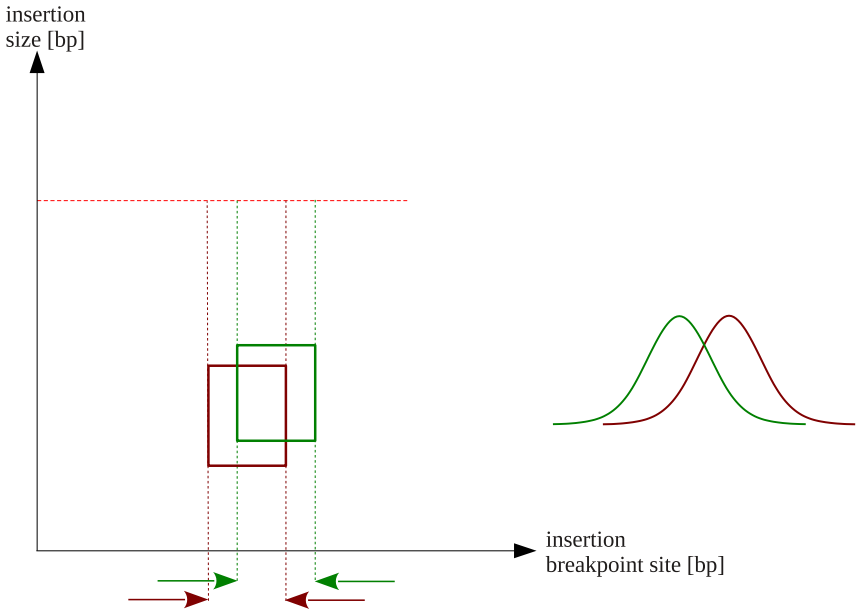


Figure 3.3: The intersection of two cluster volumes. The diagram shows the intersection of two clusters with an overlapping insertion position range. Next to the coordinate system the intersection is shown from a different angle, the overlapping normal distribution in the third dimension.

intersection volume is more affected by differences in the size than by differences in the position of an insertion. This is a desired feature of the score, because one insertion breakpoint can be covered by several mappings which are located at different positions. Moreover, the exact position of the insertion breakpoint can be approached better if there are several mappings spanning it. It is unlikely that they map all at the same position. They are rather spread around the breakpoint with a small shift in the mapping position. So the range of the potential insertion position can be limited with each shift in the mapping position.

The volume is more sensitive to a shift of the normal distribution peaks, meaning the insertion size. This harder punishment for size differences is wanted, because it means that the mappings are inconsistent and should be clustered into different clusters for different insertions at the same position.

The score is exactly one if and only if the attributes of both clusters are the same. The score is zero if there is no intersection of the possible insertion positions. If the clusters agree with the position, the score is theoretically never zero as the normal distribution only converges to zero and never reaches it. In practice a positive probability is only considered for possible insertion sizes, between zero and $s_{\max}$, though.

Nevertheless, if two clusters have insertions with a size within the same allowed range,

there is always an intersection point, no matter how far apart the mean insertion sizes are. For this case a minimum score threshold is defined. If the score of two clusters does not reach this threshold, the clusters will not be merged and the clustering process stops. Wittler [1] introduces a method to find an appropriate threshold for the deletion clustering by separating significant similarity values from noise. The median of all minimal scores of each singleton cluster is determined in a preprocessing step, when there is no threshold parameter given by the user.
In Section 4.1.3, the results with different score thresholds, including the median, for aggloIns are compared. The experiments showed that no limitations for the threshold during the insertion clustering yielded better results (Figure 4.5).

3.1.3 Clustering

During the agglomerative clustering process the two clusters A and B with the highest similarity score are merged to one new cluster C. All attributes of the new emerged cluster will be determined by combining the attributes of the fused clusters:

- $n_C = n_A + n_B$
- $x_{\min_C} = \max(x_{\min_A}, x_{\min_B})$
- $x_{\max_C} = \min(x_{\max_A}, x_{\max_B})$
- $s_C = \dfrac{n_A s_A + n_B s_B}{n_A + n_B}$
- $\sigma^2_{\text{ins}_C} = \dfrac{n_A \sigma^2_{\text{ins}_A} + n_B \sigma^2_{\text{ins}_B}}{n_A + n_B} + \dfrac{n_A n_B (s_A - s_B)^2}{(n_A + n_B)^2}$

These computations are carried over from the merging process of clusters with potential deletions in aggloDel [1]. By choosing $x_{\min}$ and $x_{\max}$ as the margin for the location of the insertion, the congruent area for an insertion of the original clusters is still compatible for both of the clusters. It will be determined more precisely with an increasing number of similar mappings restricting the boundaries.
The merged mean of the insertion size is weighted by the number of mappings in the respective cluster. This means an insertion size in a certain region is more emphasised the more mappings in a cluster are supporting it.
Like in aggloDel, we use an equation from population based statistics for aggregating non-overlapping sub-populations to create the joint variance/standard deviation for the mean insertion size in aggloIns. In the case that the insertion sizes of the two merged clusters are the same and thus $s_A - s_B = 0$, the new variance is simply the mean variance. However, the standard deviation increases with the disagreement of the clusters in the

mean insertion size. This effect is reduced by the number of mappings supporting both clusters. The more mappings each cluster contains, the more consolidated and accurate is the joint insertion size.
The next step is to recompute all similarity scores concerning the merged clusters with the attributes of the new cluster. The merging step is repeated with the next cluster pair with the actual highest score. This happens successively in a bottom-up approach until only one cluster is left or, if given, the score threshold cannot be outreached.

3.2 Implementation

The method for detecting overlapping insertions in aggloIns is an extension of the existing tool aggloDel [1], which is implemented in JAVA. Although aggloIns is based on aggloDel, some of the existing methods and structures have been modified and will be explained below.
To start the program, at least two parameters are needed:

1. The tab separated **configuration file**, which contains a list of all BAM files to import, followed by the mean fragment length and the standard deviation of the paired mappings contained in each file. Optionally, a colour can be declared for each BAM file to distinguish them in the visualisation.
2. The **output prefix**, with the path and the prefix name for the output files.

```
java -jar agglo.jar [OPTIONS] <BAMLIST> <OUTPREFIX>
```

In addition, several optional parameters can be set. The relevant arguments for finding overlapping insertions are

indels	a flag to define whether insertions (1), deletions (2) or both (0) have to be found, default: 0
insSimThreshold	minimum similarity score used in clustering of insertions, default: 0
regions	either a file containing regions or a string with a given region to consider for the import of the data
stdThreshold	read only mappings longer than mean+stdThreshold*std, default: read all
outputR	R file output for each region to produce graphics
clusterMin	only output clusters of at least this size, default: 2
quali	only consider mappings with a quality value higher than quali, default: 20
numMM	number of allowed mismatches in the mappings, default: read all
unique	a flag, if mappings with alternative hits, suboptimal hits and more than one best hit are filtered out

The program can be divided into several steps:
After checking the parameters, the procedure starts with reading the config file containing all BAM files and if defined, the regions. After that the mappings are imported by reading the BAM files. If no regions are given, all mappings are considered for the import and regions are assembled where enough mappings were found. The next step is the clustering process itself, followed by the output of the results.
The steps will be described in particular in the following subsections.

3.2.1 Mapping Import

The read-in process has been changed from parsing the SAM format and interpreting the containing flags and tags to the usage of the Picard SAM JDK package [23], introduced in Section 2.4.2. This is an API for the import of SAM or BAM files by providing each read as a so called SAMRecord object, which can be filtered with several filter settings. For the purpose of finding insertions (or deletions), only those mappings are read, which are aligned to the reference, have no duplicate reads (several copies of the same read), are the primary alignment and match the Vendor read quality filter.
Furthermore, the mapping quality value should be higher than 30 and the number of open gaps has to be limited to 0 or 1, as tests have shown that these restrictions support the best results for insertions (Figure 4.10 in Section 4.1.3). Allowing non-unique mappings also increases the evaluation result, as insertions are often in genome regions with repetitive sequences and filtering out all multiple mappings makes it hard to

find insertions. Also the minimum cluster size, the number of mappings in a cluster, takes an important role for the result and choice of filter criteria. The detailed experiment is presented in Section 4.1.3.
If both reads of a paired mapping pass the filter criteria, this gets classified as concordant or discordant, according to the fragment length of the mapping (Section 2.3). A mapping classified as discordant is assigned as insertion or deletion.

3.2.2 Determining Regions

Regions containing clusters can optionally be defined beforehand and only those mappings are considered for the clustering which are within these regions. If no regions are given, they are determined by overlapping mappings. This means, a mapping is added to a region if it is at the same chromosome and overlaps with another mapping of this region. If a mapping cannot be assigned to an existing region, or in other words, a region contains only one mapping, it is filtered out. The process of assembling the regions is used for both deletions and insertions.

3.2.3 Similarity Score Threshold

Before the clustering process starts, the similarity threshold has to be determined, if it is not given as a parameter. For clustering deletions, the median of all minimal scores from all mappings appears to be the best choice for a threshold in practice [1]. Compared to a range of several set thresholds, the best results are reached with the median score. Higher thresholds show an increased false positive rate and smaller values decrease in the true positive rate.
In contrast, several tests with aggloIns achieved an optimal result without any threshold, as seen in Figure 4.5. Especially for the detection of overlapping insertions, both false positive rate and true positive rate improve with a decreasing threshold and have their optimum with a threshold of zero.
Thus, if no similarity threshold for clustering insertions is provided, it is set to zero and the clustering process runs until no clusters are overlapping anymore.

3.2.4 Clustering Process

The clustering process for insertions is the same as for deletions and is performed for each region separately. Each mapping belonging to this respective region is considered as one singleton cluster. In a preprocessing step, all scores are computed for each cluster to all the others. They are stored in priority queues for each cluster.

Like explained in Section 3.1.3, the clusters with the maximal similarity score are merged to a new cluster. The iteration stops if only one cluster is left or the remaining similarity score cannot exceed the score threshold.

3.2.5 Output

The output format for the found insertion clusters resembles that for detected deletions. Each cluster is listed in a tab separated file with the cluster attributes, for instance chromosome number, minimal and maximal insertion breakpoint position and cluster ID. A region ID is given to see if different clusters are in one region. The insertion position, defined as center between minimal and maximal insertion site, is only given for illustration purposes. It does not represent the found insertion site.
Furthermore the approximate insertion size and its standard deviation are reported as well as the last merging score and the number of mappings included in the cluster. Additionally, the number of mappings per BAM file is given, because this information can be used to determine to which sample a potential insertion belongs. For example, if many mappings count among a sample of tumor tissue and only a few mappings from the healthy sample are among them, the insertion may be involved in tumor emergence. Another output file lists all mappings per cluster to track the cluster formation.
If the parameter is set, output files with R code are created, which give the possibility to visualise the clusters of a region in a PDF file: The theoretical geometric presentation of an insertion, demonstrated in Figure 3.2, is shaped like a rectangle, with the minimal and maximal positions of the insertion breakpoint on the x-axis and the insertion size on the y-axis. In contrast, deletions are modelled as triangles above the diagonal of the coordinate system, because x- and y-axis are showing the left and right deletion breakpoint positions within the genome and the deletion size is the distance to the diagonal. To combine deletions and insertions in one graphic, it is more convenient to add another y-axis, indicating the deletion size, below the x-axis and plot the deletions analogous to the insertions. Figure 3.4 illustrates the visualisation of both geometric modellings. The mappings of each BAM file are coloured as defined in the config file. The normal distribution in the third dimension is implied by grey shades. Clusters that contain at least as many mappings as defined by the parameter (default two) are coloured yellow, with a pink label indicating the number of involved mappings per sample. The smaller clusters are indicated only by dotted lines.

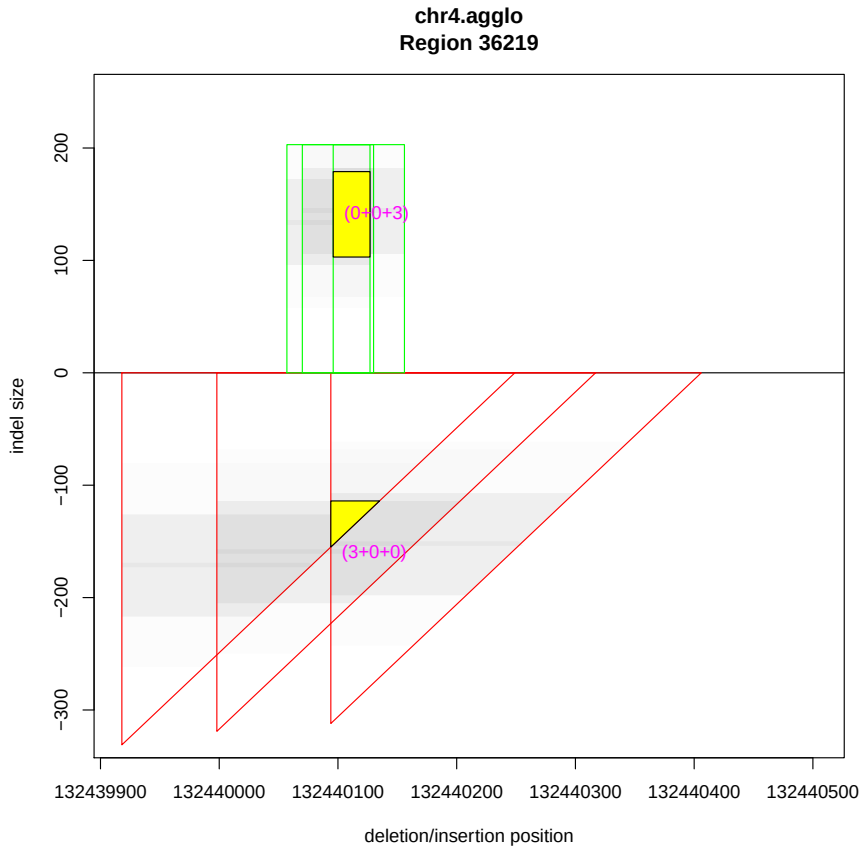


Figure 3.4: Visualisation of insertions and deletions in one graphic. The green cluster indicates an insertion at the same position where the red sample harbours a probable deletion. (Taken from the results in Section 4.2)

3.2.6 Time Complexity

In the first step of the clustering process, all $N(N-1)$ similarity scores are computed and are added to the priority queue. It takes $O(\log N)$ time to add an element to a priority queue. This leads to a run time for the first step of $O(N^2(P+\log N))$, with P for the score computation, which is described below. After that, N clusters are merged iteratively. For this, the currently maximal score has to be determined in $O(N)$ time by looking for the maximal score in the priority queue of each cluster. The two respective clusters are merged to one cluster, which includes the recomputation and updating of the concerned scores in $O(N(P+\log N))$ time.

The time to calculate the score depends on the value of the fragment size. The intersection of two volumes is given by the overlapping normal distributions indicating the probability of an insertion size and the range of putative breakpoint positions, $x_{\max}-x_{\min}+1$. The maximal possible distance between mapped paired reads (and thus the number of possible breakpoint positions) is $\mu_{\text{frag}}-2r$. However, for this case the insertion size is approximately zero.

The width of the normal distribution is yielded by zero and the maximal insertion size

$s_{\max}$, which is also depending on μ_{frag}. A cluster indicating the maximal insertion size implicates a distance of zero between the most inner reads.
Therefore, the score computation P is constant and takes $O(\mu_{\text{frag}})$ time. The overall run time for the clustering process is consequently $O(N^2 \log N)$.
In practice, the most temporal effort arises during the reading of the BAM files and increases linearly with the number of mappings.

Chapter 4

Results

This chapter presents the evaluation of aggloIns. The first section describes the evaluation on simulated data. It starts with the adjustment for optimal mapping filters and parameter settings. After that follows a benchmarking in comparison to another tool. The second section shows the results of aggloIns used with a real data set.

4.1 Simulated Data

This section starts with the explanation for the establishment of the simulated data. It describes which genome serves as reference, which insertions are sampled and which test scenarios are simulated. Then, several filter and option adjustments for different scenarios are performed, followed by the benchmarking, where aggloIns is compared to the tool BreakDancer [16].

4.1.1 Sample Insertions

To assess the parameters, evaluate the results and to benchmark the tool, genomes with insertions were simulated. Levy et al. [24] identified structural variants by comparing J. Craig Venter's genome (HuRef) with hg18. This list of SVs, including insertions along with their sequences, served as source for the insertion simulation (insertion size distribution is seen in Figure 4.1). Thus, hg18 was used as reference. It is a version of the human reference assembly from 2006 by the National Center for Biotechnology Information (NCBI). To have enough data in order to get stable results and at the same time ensure practicability, we took four chromosomes of the reference genome.
Two copies of the reference chromosomes were created and insertions from the venter list of known insertions were integrated, 500 overlapping and 500 non-overlapping (125 per

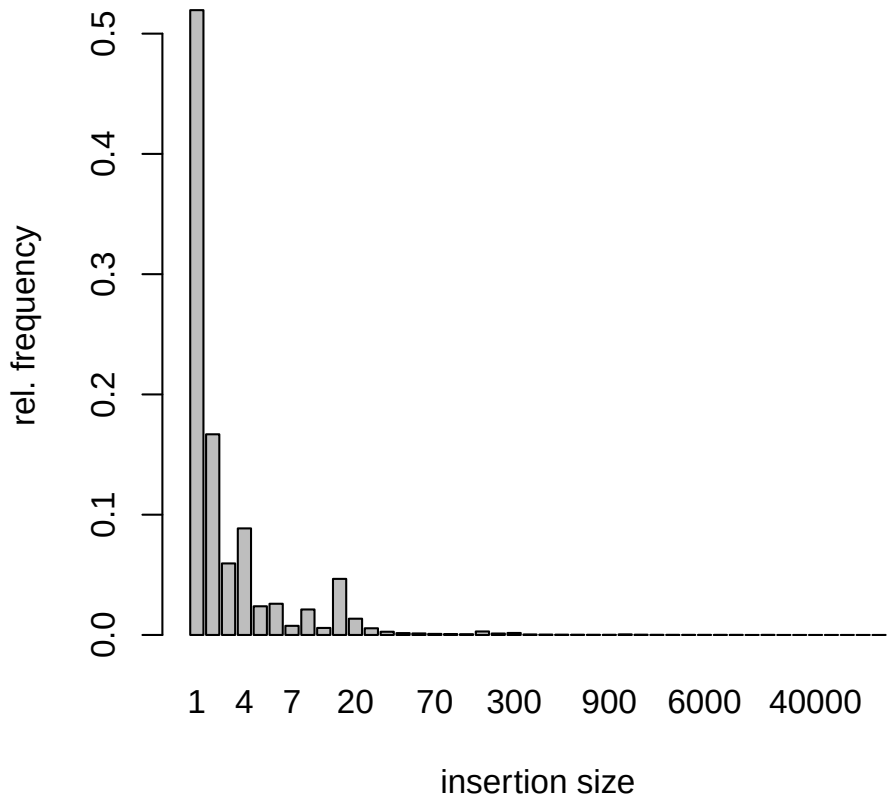


Figure 4.1: Histogram of the insertion size distribution of the venter genome of overall 403623 insertions.

chromosome). The first chromosome copy contains 250 insertions. They were randomly chosen and the sequence was inserted at the actual position. The second copy contains 125 insertions that are overlapping with 125 from the first copy. For this, randomly chosen insertions were inserted close to the position of the corresponding overlapping insertion: in a random distance to the other position between 0 and the expected read distance, so that it is possible that the mapped reads span both of the insertion breakpoints.

The characteristics for the read pair simulation is based on the characteristics of the real data in Section 4.2, to obtain a simulated data set as realistic as possible. Table 4.1 shows that the read pairs of the real data set have a fragment length around 300 bp and a standard deviation around 35. The read length is 51 bp. For the simulation, the read length and the fragment length are adopted. However, about 92% of all insertions in the venter list are shorter than 100 bp. Fragment size and standard deviation are delimiting the detectable insertion size, as explained in Section 2.3. So the standard deviation is

scaled down to 15 to take also smaller insertions into account. We sampled only insertions from the list with a size between 35 and 190. (Although the theoretical minimal detectable insertions size would be 45, we examined also smaller insertions, because of the deviation). The distribution of the sampled insertion sizes is seen in Figure 4.2.

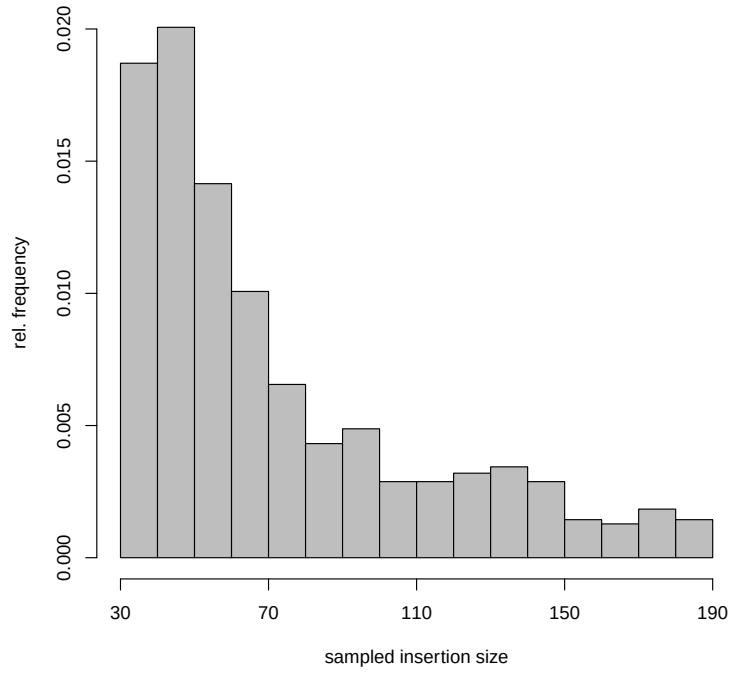
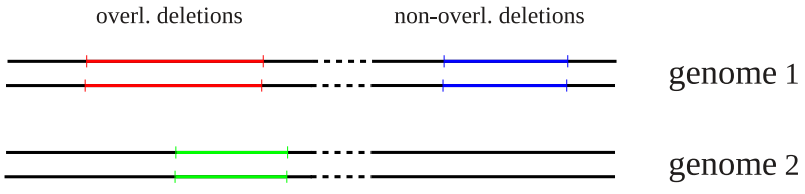


Figure 4.2: Histogram of the sampled insertion size distribution.

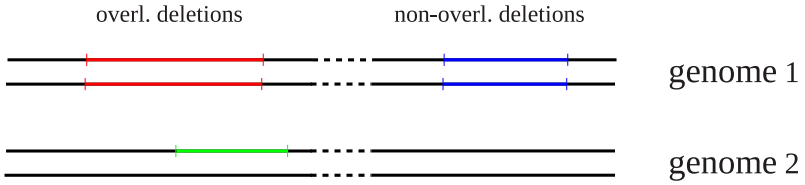
4.1.2 Simulate Mappings

An insertion can either be homozygous, which means the insertion occurs in both alleles or it is heterozygous. In this case, the insertions is only existing in one of the alleles. For the evaluation, two different scenarios are established (Figure 4.3): In the first one, both genomes are homozygous. In the second scenario, only the first copy is homozygous and the second copy is heterozygous. Both scenarios are tested with a coverage of 20× and 60×, which means that each position of the genome is covered by 20, respectively 60, different reads.

The tool *SimSeq* [25] is used to simulate the reads of both modified genomes. It is an Illumina paired-end (and mate-pair) short read simulator, which provides typical sequencing errors occurring during the sequencing process. For this, an error profile is needed. In our case, the example error profile from a 100 bp Illumina GAIIx data set, provided by SimSeq, is applied, because it is practicable for reads $\leq$ 100 bp. The other



(a) Scenario 1: homozygous



(b) Scenario 2: heterozygous

Figure 4.3: The two scenarios for the evaluation of the simulated diploid data set. The first copy contains 500 insertions belonging to the overlapping pairs and 500 non-overlapping insertions. The corresponding 500 overlapping insertions are in the second copy. (a) Both genomes are both homozygous. (b) The first copy is homozygous, but only one allele of the second copy carries the insertions, i.e. it is heterozygous.

parameter values are chosen, as described earlier: a read length of 51 bp, an insert size of 300 and a standard deviation of 15.

The output is in SAM format and has to be converted to a sorted and indexed BAM file. This is done via SAMTools [22] with the commands *view*, *sort* and *index* (see detailed information about SAM/BAM and SAMTools in Section 2.4.1).

To be able to map the reads to the reference, the BAM files are converted to FASTQ files, containing the sequences of the reads along with their quality scores. The reads are stored in two files, each for one mate of the paired reads. The converter used here is contained in the Picard-tools package [23] (Section 2.4.2).

The reads are mapped to the reference sequence with BWA (**B**urrows-**W**heeler **A**lignment) [26], version 0.7.5. BWA is a tool to align short reads (e.g. by Illumina/Solexa sequencing with $32-100$ bp) against a long reference, like in our case several chromosomes of the human genome. It is based on the backwards search and BWT (**B**urrows-**W**heeler **T**ransform), and allows mismatches and gaps. Besides single read mapping, paired-end mapping is also supported, which is an essential option for the purpose in this context. The output is in SAM/BAM format and can be used as input for aggloIns.

4.1.3 Evaluation and Comparison of different Score Thresholds and Mapping Filter Criteria

To evaluate the results, all correct insertion predictions have to be counted. The output for the found insertions provides the interval for the potential insertion between $x_{\min}$ and

$x_{\max}$, as well as the approximate insertion size and its associated standard deviation. A found insertion is only counted as correct hit if exactly one true insertion is found within the interval. Additionally, the difference of the true and found insertion size must not be bigger than three times the standard deviation. The boundaries of the insertion site window are extended by 4 positions. The practice showed that several insertions were not counted as correctly found, even though the true position was only a few positions next to the window. The reason for this is a fairly probability that the insertion sequence has a similar starting/ending sequence as the sequence at the breakpoint position (Figure 4.4). Examining the insertions showed, that the inserted sequence is often similar to the genome sequence around the breakpoint.

Figure 4.4: If the insertions sequence (red sequence) is similar to the sequence around the insertion breakpoint (bold sequence in donor and reference sequence), it might happen that a mapped read (black arrow) overlaps the breakpoint position within the reference (red line).

If the predicted insertion is not accepted as correctly found, it is counted as false positive. Each undetected true insertion is also counted. The evaluation is presented as a ROC plot (**R**eceiver **O**perating **C**haracteristic), visualising the false positive rate (FPR) against the true positive rate (TPR), also called sensitivity. The rates for the single insertions (TPR1, FPR1) and overlapping insertions (TPR2, FPR2) are given by:

$$\text{TPR1} = \frac{\#\text{correctly found single ins.}}{\#\text{simulated single ins.}} \quad \text{FPR1} = \frac{\#\text{falsely found single ins.}}{\#\text{sim. overl. ins.}+\#\text{falsely found ins.}}$$
$$\text{TPR2} = \frac{\#\text{correctly found overl. ins.}}{\#\text{simulated overl. ins.}} \quad \text{FPR2} = \frac{\#\text{falsely found overl. ins.}}{\#\text{sim. single ins.}+\#\text{falsely found ins.}}$$

Before benchmarking aggloIns in comparison to another tool, the optimal filter and parameter settings have to be adjusted.

First of all we checked if the median of all minimal singleton cluster scores is the ideal choice for a score threshold, like for the deletion clustering. A range of score thresholds between 0 and 0.1, and the median are compared and the resulting ROC curve is seen in Figure 4.5. We do not elaborate on the TPR and FPR values in particular at this point, we are only interested in the behaviour of the ROC curve.

The plot shows that the true positive rate for overlapping insertions (solid circles) is increasing with smaller score thresholds. Their false positive rate converges to zero. The optimum is already reached with a threshold of 0.01 (indicated by a blue star), but stagnates for smaller threshold values until 0 (red cross). The true positive rate for single insertions (rings) increases also with smaller thresholds, but so does the false positive rate. However, the slope of the ROC curve illustrates that the true positive

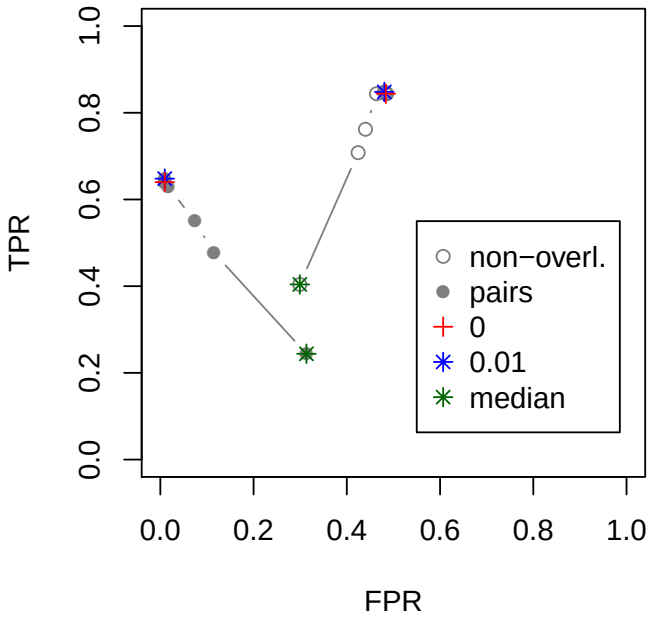


Figure 4.5: ROC plot of different score thresholds.

rate is increasing more than the false positive rate. Again, a thresholds from 0 to 0.01 provides the best results. The median, in contrast, reaches a very high value, in this case 0.29315 (green star), and is therefore not applicable as score threshold.
Taking a closer look at each TPR and FPR per score in Figure 4.6, it is seen that there is a small drop of TPR2 from 0.01 to 0. The reason are four overlapping insertions that are merged to single clusters with smaller thresholds than 0.01. However, the result is stagnating for the rest and no other impairments are found. So we count this mistake as outlier and still assume zero as a good choice for a default threshold.

Conclusion: In contrast to the optimal threshold for deletion clustering, the median cannot be considered as optimal score threshold for the clustering of insertions. The cause for this may be the similarity of insertions indicating discordant mappings in the same region. The scores to all mappings and the resulting median are too high to take this approach for an optimal solution. Moreover, the results converge to the optimal results with smaller thresholds. So, if no threshold is given by the user, the threshold is set to zero.

The next step is to find the best mapping filter options and parameter settings to detect actual insertions. We tested different settings with the scenario of the homozygous genome with 60× coverage and minimum cluster size of 3 (at least 3 mappings within

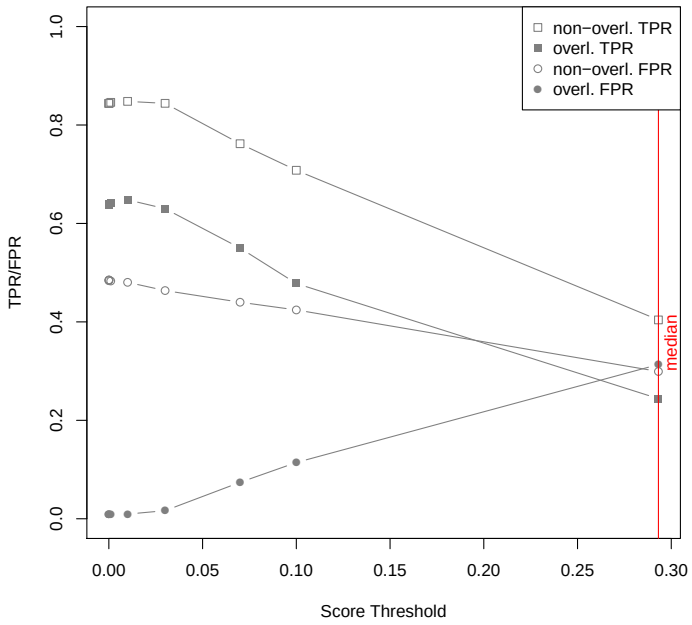


Figure 4.6: Plot of TPR and FPR per score. Solid icons show the rates for overlapping insertions. The contour icons present the rates for single insertions. The true positive rates are illustrated by rectangles and the false positive rates by circles.

the cluster).

A first evaluation of aggloIns with all original filter settings for the BAM reading process, as used for aggloDel, showed an unsatisfying result. Only around 40% of all overlapping insertions were found correctly and 63% of all single insertions. Nearly 790 insertions were mistakenly predicted.

AggloDel filters mappings with a mapping quality < 20 and mappings which have alternative and suboptimal hits. Mismatches are allowed, open gaps are not.

A closer look at the mappings with the help of IGV (integrative genomics viewer) [27, 28] gave a hint, why so many falsely predicted insertions appeared and many of the true insertions were not found at all. Many mappings, that were mistakenly considered as discordant, showed a fairly small quality between 20 and 30 or had mismatches within the read mapping. Often, those read mappings overlapped the insertion breakpoint. They were still counted as mapped read and only contained one insertion at the breakpoint position, as seen in Figure 4.7.

A restriction of the filter for mappings with a quality < 30 and containing mismatches gained an improved result. Figure 4.8 illustrates this, where the original settings are presented by the red plot and the modification with the blue plot. Only a third of the falsely predicted insertions are left and over 50 overlapping and non-overlapping insertions were detected correctly (see tables in the appendix).

Still, some insertions could not be found, because the filter settings were on the other

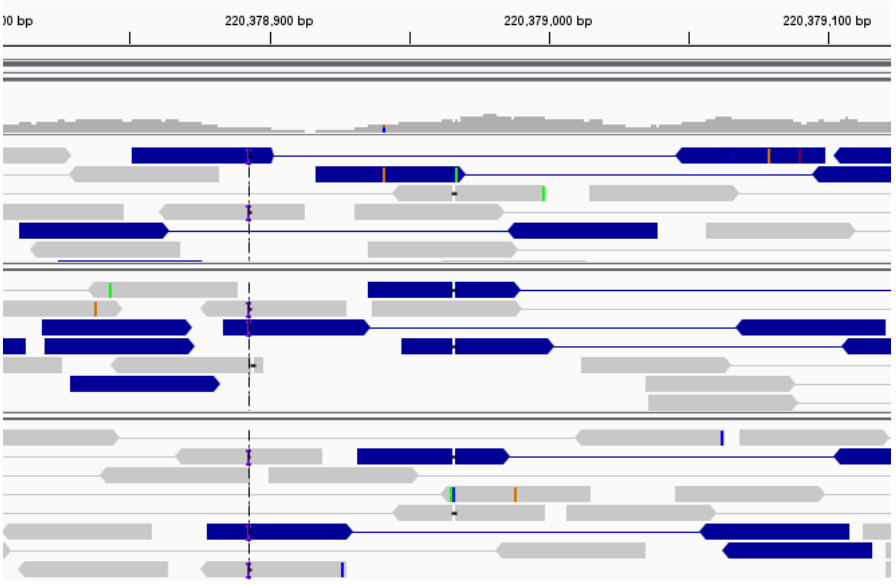


Figure 4.7: Mapped reads, overlapping an insertion breakpoint. The presentation with IGV shows that at the position of the insertion breakpoint (dotted line) few reads are mapped. They contain a single insertion at the breakpoint position (purple I).

side too strict. Many mappings had alternative hits. In some regions with insertions, all mappings were filtered out, so there was no chance to find them. Allowing those non-unique mappings let few more falsely predicted insertions appear, admittedly. Nevertheless, it increased the number of correctly found hits again by 50 single insertions and 50 overlapping insertions. The best solution for the given scenario is therefore considering non-unique mappings without any mismatches.

We also ran aggloIns with the described filter settings with the scenario of the homozygous genome and 20× coverage. Here these restrictions seemed to be too strict for the scenario. Figure 4.9 shows that restricting quality value and mismatches (blue) worsens both true positive rates, but at least cuts back the high false positive rate for single insertions in the original setting (red). Taking non-unique mappings into account raises TPR1 and TPR2 slightly (purple) to about the same level as the original setting. Several insertions could not be detected, as some read pairs were sorted out because of single mismatches. Single nucleotide errors can easily happen during the sequencing process. So we allowed one mismatch. The results improved remarkably, for both single insertions and overlapping insertions, illustrated by the green plot. We also tried to improve the result by allowing 2 or 3 mismatches. However, these approaches deteriorated the results again, as too many mappings are considered mistakenly. They prevent correct insertions to be found, because they might shift the window for the possible insertions breakpoint and too many insertions are falsely predicted. Thus, the best choice for filter setting in this scenario is to consider non-unique mappings with at most one mismatch. Allowing one mismatch for a coverage of 60×, however, does not improve the result (dark green plot in Figure 4.8).

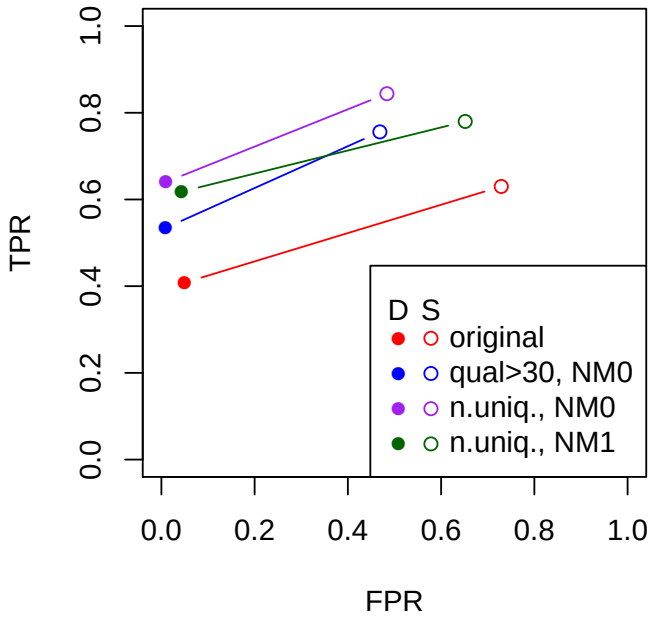


Figure 4.8: ROC plot with different options for the homozygous genome with 60× coverage and cluster size 3. Solid circles indicate overlapping insertions (D=double) and rings represent the rates for single insertions (S=single).

As the optimal solutions for 20× and 60× coverage are different, we had a look at the cluster size, i.e. the number of supporting mappings. Up to now, we counted each cluster with a minimum size of 3 for both coverages. Instead, we adjusted the ratio of coverage to cluster size. For coverage 60×, we used a minimum cluster size of 9 into account. Figure 4.10 shows that the results are now nearly the same as for 20× coverage and a minimum cluster size of 3: The true positive rates are lower in comparison to cluster sizes of 3, especially TPR1. The result hierarchy resembles now the one with 20× coverage. However, the allowing non-unique mappings without any mismatches (purple) gains better results than the original setting (red). The best result is achieved with non-unique mappings and one allowed mismatch (dark green).

Conclusion: The experiments showed that the most correct single and overlapping insertions are found, if high quality non-unique mappings with at most one mismatch are clustered. This applies to low coverage and small cluster sizes as well as for higher coverage with bigger cluster size. Small clusters found with a high coverage can still be accurate, though, but with stricter filter settings. Only perfect mappings without any mismatches may be considered, otherwise too many false predictions occur.

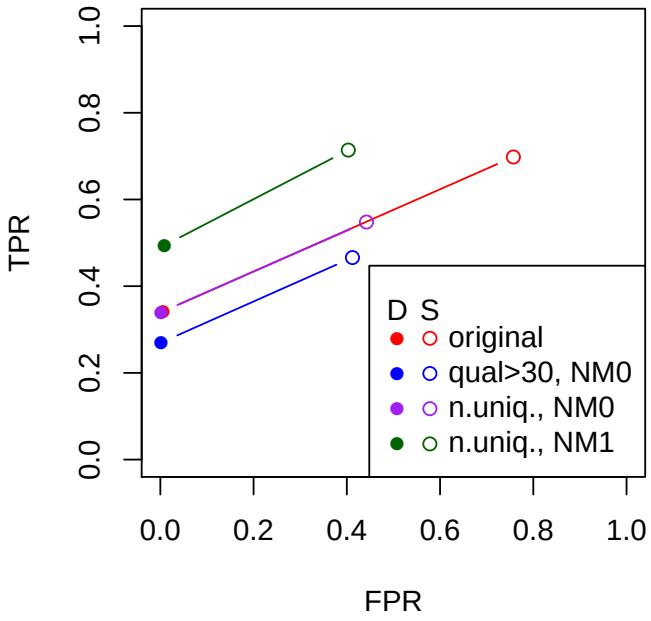


Figure 4.9: ROC plot with different options for the homozygous genome with 20× coverage and cluster size 3.

4.1.4 Comparison with BreakDancer

To benchmark aggloIns, we want to compare the results of a different tool considering the same data set. The detection of deletions with aggloDel was compared to GASV [2] and CLEVER [29], though CLEVER was regarded only marginally in the end. Although it reports overlapping deletions, they are meant as alternative predictions, rather than overlapping predictions. GASV, however, only focuses on deletions, inversions and translocations and does not support the detection of insertions.

Alternatively, we wanted to do the benchmarking with SVM^2 [30], also a paired-end mapping approach to detect SVs. Unfortunately, the server hosting the program is not available.

Lastly, we ran BreakDancer [16]. The algorithm also takes anomalous paired-end mappings into account, based on the read pair distance and the direction of the aligned reads. It considers regions within the genome with a number of anomalous mappings deviating from the one expected on average. Putative SVs are determined by finding regions interconnected by the same mappings and calculating a confidence score for these SVs based on a Poisson model, considering the number of supporting reads, the size of the regions and the coverage of the genome.

We compared the results of detected overlapping and single insertions of all scenarios

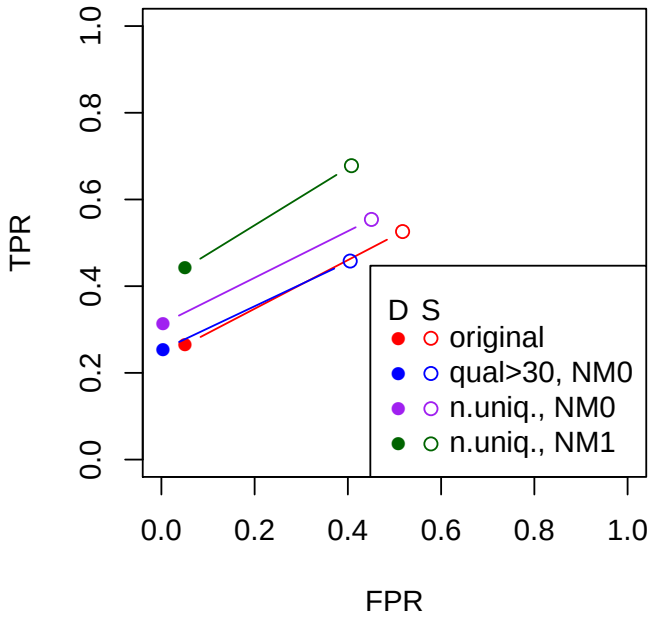
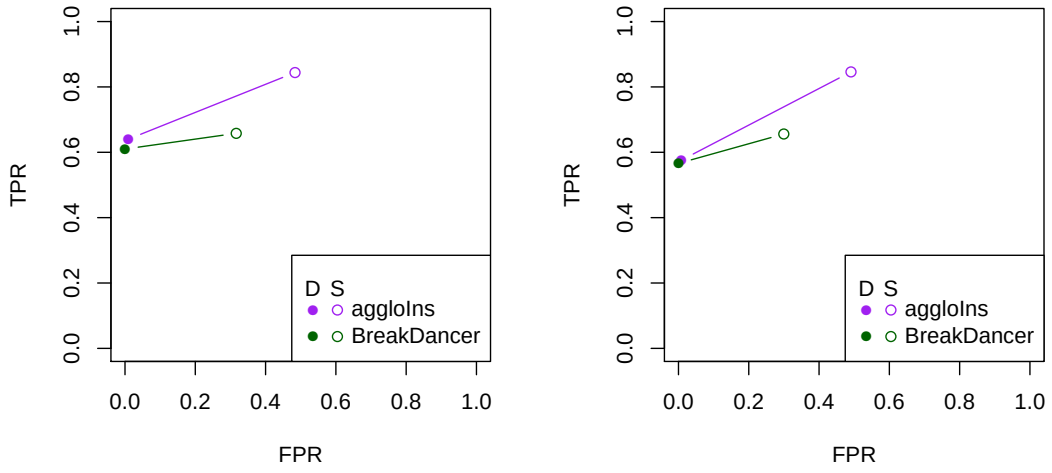


Figure 4.10: ROC plot with different options for the homozygous genome with 60× coverage and cluster size 9.

with homozygous and heterozygous genomes and 20× and 60× coverage, respectively. Like explained in the previous section, we filtered the mappings of the data set with 60× coverage with a mapping quality < 30 and any mismatches, but allowed non-unique mappings. For the data set with 20× coverage we allowed one mismatch.

Figure 4.11(a) shows the results for the homozygous genome with 60× coverage. The true positive rate for overlapping insertions is quite similar, though aggloIns finds a few more overlapping pairs. BreakDancer found 61% and aggloIns was able to detect 64%. However, the results for found single insertions differ drastically. TPR1 and yet FPR1 are much higher for aggloIns. 84.4% of all single insertions could be recognised, whereas BreakDancer could only find 65.8%. Admittedly, aggloIns predicted 238 insertions mistakenly and BreakDancer only 94.
For the heterozygous genome, the tendencies stay the same, see Figure 4.11(b). Only the number of correctly predicted overlapping insertions decreases for both methods.

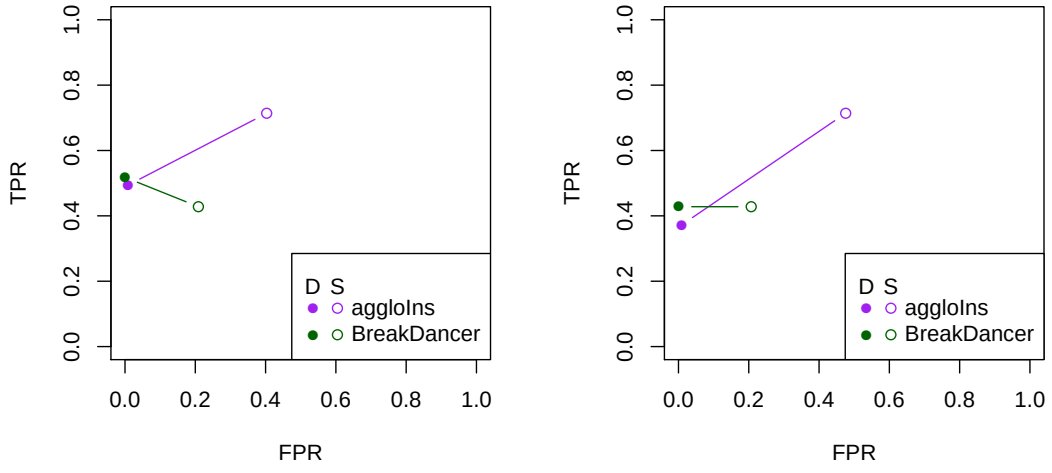
The ROC plot of the results for both scenarios with 20× coverage in Figure 4.12 illustrates that the divergence of both TPR1 and FPR1 values increases even more with the lower coverage. Exemplarily, for the homozygous scenario, aggloIns is able to detect 71.4% of the single insertions correctly, BreakDancer only 42.8%. Nevertheless, aggloIns reports 78 falsely spotted insertions and BreakDancer only 21. In the homozygous scenario, the true positive rates for overlapping insertions from both methods are



(a) 60× coverage, homozygous (b) 60× coverage, heterozygous

Figure 4.11: ROC plot for aggloIns and BreakDancer with 60× coverage

still close. 49.5% of all overlapping insertions could be found by aggloIns and 51.8% by BreakDancer. Figure 4.12(b) shows that aggloIns has its difficulty with the heterozygous scenario and detects only 37.2% of the overlapping insertions. Also BreakDancer achieves a poor result of 42.8%, though.

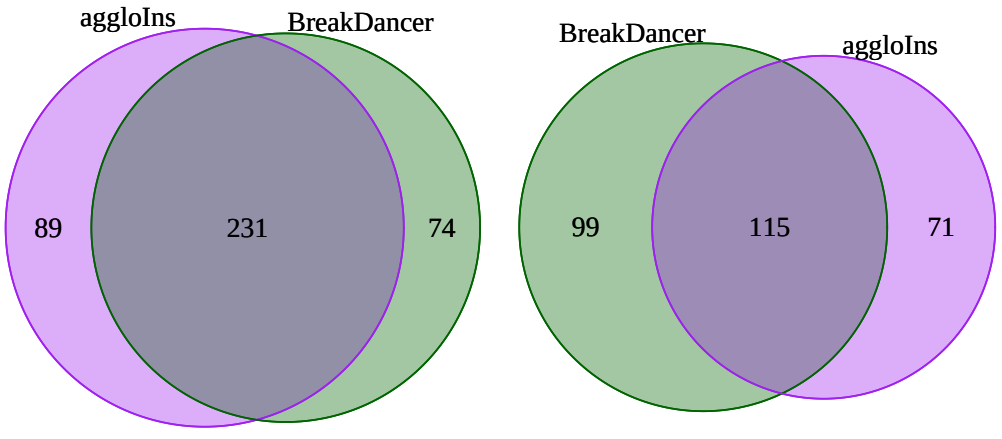


(a) 20× coverage, homozygous (b) 20× coverage, heterozygous

Figure 4.12: ROC plot for aggloIns and BreakDancer with 20× coverage

Conclusion: With a higher coverage, aggloIns has the ability to detect few more overlapping insertions than BreakDancer and even 18.6 more percentage points of all single insertions. However, the specificity for single insertions with aggloIns is lower, which means more erroneous insertion predictions occur. As expected, the ability to find overlapping insertions decreases with a lower coverage for both methods. Here, BreakDancer detects a few more than aggloIns, yet aggloIns is way out in front of predicting single insertions. The detection with aggloIns can achieve 26.6 more percentage points.

We surveyed how many overlapping insertions were correctly found with both tools and how many were detected by only one of the methods. The Venn diagrams in Figure 4.13, for the extreme scenarios of homozygous with 60× coverage and heterozygous with 20× coverage, show that many overlapping insertions were only found by one of the methods. Especially for the case with 20× coverage, less than half of all overlapping insertions were predicted by both aggloIns and BreakDancer. With 60× coverage, about 59% of all insertion pairs were found by both of them. Investigating the characteristics of those insertions found by only one method showed, that this circumstance comes along with the concerning insertion sizes. BreakDancer had problems with overlapping insertions that differ extremely in their insertion size, e.g. if one of the two has a size around 40 and the other one over 100. They are merged and reported as one insertion prediction. On the contrary, overlapping insertions that were not reported by aggloIns mainly had a similar position and size, with a difference up to 30, and were therefore clustered together.



(a) 60× coverage, homozygous (b) 20× coverage, heterozygous

Figure 4.13: Venn diagram of correctly found overlapping insertions with (a) 60× coverage and homozygous scenario (b) 20× coverage and heterozygous scenario.

4.2 Real Data

The data set used in this evaluation has been provided by the Department of Paediatric Oncology, Haematology and Immunology at the Düsseldorf University Hospital, Germany. Three samples of an acute lymphoblastic leukaemia patient have been sequenced: before the treatment, after the treatment and after a relapse. The sequencing was done on an Illumina HiSeq 2000 with a read length 51, a segment length around 300 and coverage of respectively 6×, 8× and 8×. These reads were mapped to the reference sequence hg19 at the Institute of Medical Informatics at University of Münster, Germany. They used BWA [26] as mapping tool with not more than three allowed mismatches and removed duplicates with Picard tools [23].
Table 4.1 shows the mean fragment length and its standard deviation for chromosome 1 of all three stages as example. The assumption that the fragment length is normal distributed is supported by Figure 4.14, which shows the fragment length distribution of chromosome 1 in the relapse sample mappings.

Table 4.1: Mean fragment length and standard deviation of chromosome 1

data set	**mean**	**std**
chr1 initial	261.7477	40.90812
chr1 remission	297.2338	32.82395
chr1 relapse	306.2386	34.71178

Running aggloIns on a Sun machine with AMD 64 bit processors and 256 GB Memory, only detecting insertions, took in total 3:59:02 hours. This time was dominated with 3:57:47 hours by reading and filtering 66 GB data containing about 1.6 billion reads. Thus, only 1:15 minutes were needed to cluster the $2,282,493$ filtered mappings and report the results.
As the total coverage of the initial, remission and relapse sample sum up to 22×, the data was filtered with the optimal filter settings for low coverage (see Section 4.1.3): Non-unique mappings are allowed, clusters with more than one mismatch and a mapping quality < 30 are filtered out. All mappings with a fragment length deviating more than three times the standard deviation from the expected fragment length are considered. These mappings assembled to overall $102,859$ regions, as described in Section 3.2.2. However, $3,624$ regions contained $4,404$ clusters, which are supported by at least three mappings. Table 4.2 gives an overview of the amount of regions with the number of comprising insertions. The majority contains single insertions, only 374 regions could be found with overlapping insertions (example in Figure 4.15). Regions with a high number of overlapping insertions are primarily found in telomeric and centromeric areas

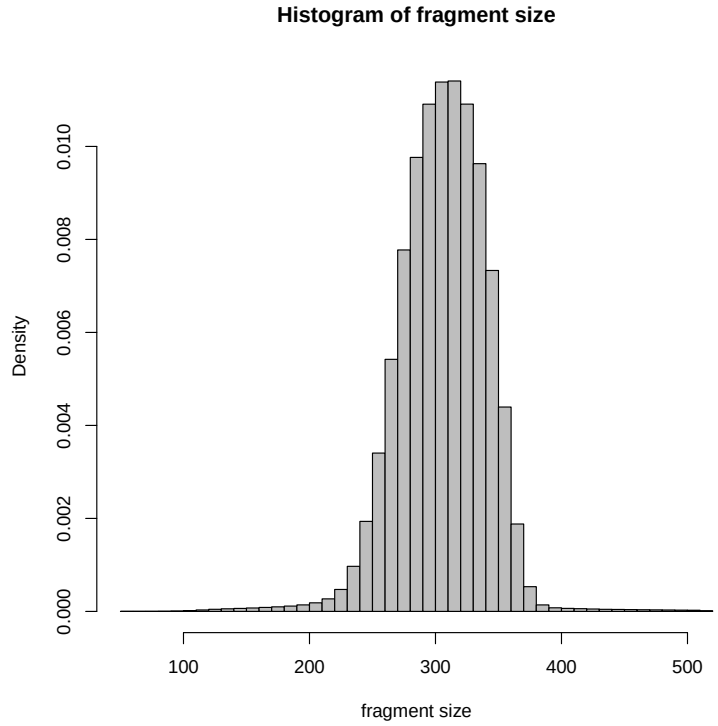


Figure 4.14: Mapped fragment size distribution of chromosome 1 (relapse)

Table 4.2: Number and size of detected regions harbouring insertions.

region size	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**≥10**	**total**
# regions (insertions)	3250	246	72	29	16	3	2	1	0	5	3624
# regions (indels)	2968	449	99	36	24	4	6	3	3	14	3606

of the chromosomes. Mappings in these regions are not very reliable and should be ignored.

Running the program with detecting both insertions and deletions revealed 3,606 regions containing insertions supported by at least three mappings. An example for an overlapping Indel is found in Section 3.2 in Figure 3.4. However, overall over 86,000 regions, harbouring clusters of at least size 3, were found. The reason is that deletions in this size range appear much more often.
Figure 4.16 shows the distribution of the found insertion sizes. Most insertions have a size between 130 bp and 150 bp. The low number of found insertions in total could be explained by the restrictions of detectable insertions with this method, explained in Section 2.3. With a mean fragment size of 300 and a standard deviation of 35, considering those mappings which differ at least three times the standard deviation, the minimal

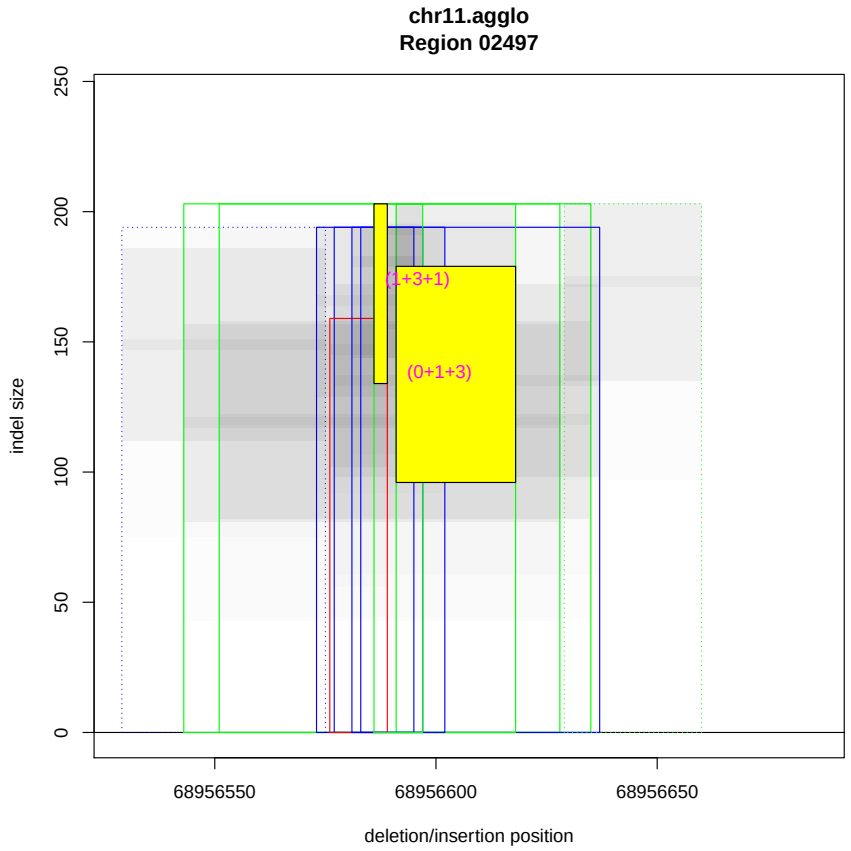


Figure 4.15: Detected overlapping insertion in chromosome 11. One insertion with an approximate length of 138 is found in the relapse sample close to an insertion of length 174 in the remission sample.

detectable size is 106. With a read length of 51, the maximal detectable insertion size is 198 (as a reminder: $s_{\max} = \mu_{\text{frag}} - 2r$). Looking at the list of insertions in HuRef found by Levy et al. [24], only 1035 insertions in this size range are found within the whole genome. Over 99% of all insertions are smaller than 100 and about 92% have a size below 10 (distribution see in Figure 4.1 in Section 4.1.1). A data set with a more precise fragment size would lead to more detectable insertions and results containing more possible overlapping insertions.

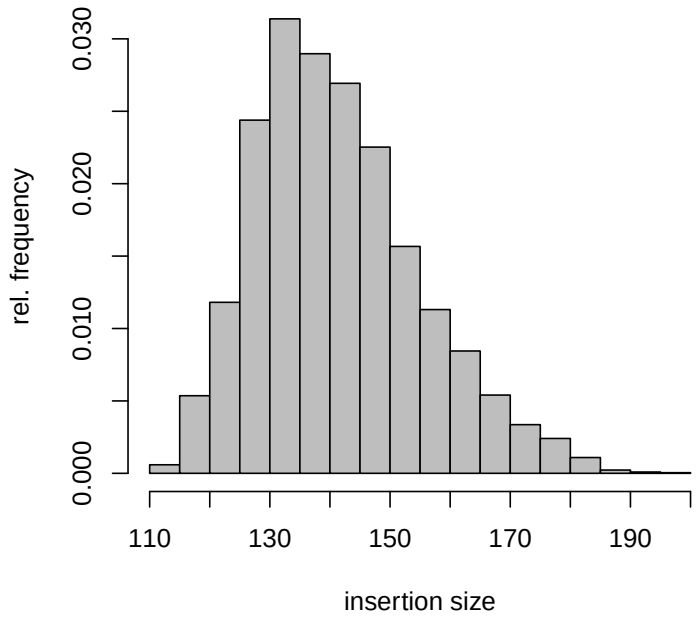


Figure 4.16: Insertion size distribution of all detected insertions.

Chapter 5

Conclusion

Structural variants are not only involved in the diversity of genomes, but also in the development of genetic diseases, especially cancer. Detecting such disease-causing structural variants undergoes an increased attention in the genomic research. Furthermore, the comparative analysis of found structural variants in different samples or tissues is an important issue in order to distinguish and assign, for example, individual mutations and tumor causing structural variants, occurring in the same region.

This work presents an approach to detect overlapping insertions by clustering paired-end mappings with a discordant configuration. The score to assess the similarity of mappings, respectively clusters, is based on a geometrical representation of a putative insertion, derived from the behaviour of the involved mappings. With the agglomerative clustering process, the most similar clusters are merged step by step and possibly overlapping clusters are revealed.

The algorithm of the agglomerative clustering approach is an intuitive, simple and fast solution. Defining the similarity score by the normalised cluster volume showed plausible results and a threshold of zero performed best in the evaluation. Investigating more clustering algorithms, e.g. the top down approach of divisive clustering, or other score definitions might exhibit even better and more accurate results.

Applying the method to a real data set confirmed that the number and accuracy of predicted (overlapping) insertions is depending on the fragment size and its deviation. These criteria are limiting the detectable insertions size. Only few actually overlapping insertions could be found. Running aggloIns on a more accurate data set would possibly yield more predictions of overlapping insertions. There is also potential in combining this approach with other methods, like split read analysis or coverage evaluation, to improve the accuracy of predicted insertions and to be able to detect insertions beyond the given size restrictions.

Yet, it has been shown that our approach provides good results according to the evaluation with simulated data, particularly for high coverage mapping data. It is planned to provide aggloIns together with aggloDel in one tool together with the source code on the Bielefeld University Bioinformatics Server (http://bibiserv.cebitec.uni-bielefeld.de).

Appendix A

Evaluation Results

The following tables show the results for all evaluations with the simulated data from Section 4.1. It is counted how many correct and false predictions were made. The header for each column is interpreted exemplarily as follows:
T0F1 $\rightarrow$ 0 true insertions in this region, but found 1 insertion
T1FX $\rightarrow$ 1 true insertion in this region, but found more than 2 insertions
T2F2 $\rightarrow$ 2 true insertions in this region and found 2 insertion (correct hit)
and so on.

Table A.1: Results for different thresholds, homozygous scenario and 60× coverage in Section 4.1.3.

threshold	T0F1	T0F2	T0FX	T1F0	T1F1	T1F2	T1FX	T2F0	T2F1	T2F2	T2FX
0	238	0	0	71	422	7	0	51	119	320	10
10^{-8}	238	0	0	71	422	7	0	51	120	319	10
10^{-4}	238	0	0	71	422	7	0	51	120	319	10
10^{-3}	239	0	0	70	423	7	0	51	118	321	10
10^{-2}	239	0	0	69	424	7	0	49	116	324	11
$3 \cdot 10^{-2}$	229	0	0	66	422	12	0	49	109	315	27
$7 \cdot 10^{-2}$	216	0	0	66	381	53	0	48	99	275	78
10^{-1}	200	0	0	66	354	80	0	47	97	239	117
0.293151	132	3	0	70	202	196	32	44	58	122	276

Table A.2: Results for different filter and option settings and homozygous scenario in Section 4.1.3. The first columns show coverage (cov), minimal cluster size cs), number of allowed mismatches (nm) and if the mappings are unique (u).

cov	cs	nm	u	T0F1	T0F2	T0FX	T1F0	T1F1	T1F2	T1FX	T2F0	T2F1	T2F2	T2FX
20	3	0	0	36	0	0	225	274	1	0	129	201	170	0
20	3	0	1	29	0	0	266	233	1	0	176	189	135	0
20	3	1	0	78	0	0	138	357	5	0	93	155	247	5
20	3	1	1	68	0	0	185	310	5	0	126	169	201	4
20	3	-	1	645	0	0	144	349	7	0	99	222	171	8
60	3	0	0	238	0	0	71	422	7	0	51	119	320	10
60	3	0	1	202	0	0	116	378	6	0	98	127	267	8
60	3	1	0	634	0	0	61	390	49	0	45	105	309	41
60	3	1	1	558	0	0	103	356	41	0	85	114	267	34
60	3	2	0	763	0	0	76	350	74	0	48	132	254	66
60	3	2	1	678	0	0	117	319	64	0	88	135	219	58
60	3	3	0	807	0	0	100	327	73	0	61	159	232	48
60	3	3	1	711	0	0	128	309	63	0	100	154	201	45
60	3	-	1	788	0	0	121	315	64	0	98	151	204	47
60	9	0	0	26	0	0	221	277	2	0	126	211	157	6
60	9	0	1	21	0	0	269	229	2	0	178	190	127	5
60	9	1	0	47	0	0	133	339	28	0	83	176	221	20
60	9	1	1	35	0	0	186	290	24	0	137	165	178	20
60	9	2	0	74	0	0	144	315	41	0	89	189	183	39
60	9	2	1	63	0	0	197	269	34	0	137	183	146	34
60	9	3	0	112	0	0	171	290	39	0	110	204	162	24
60	9	3	1	90	0	0	210	258	32	0	148	197	130	25
60	9	-	1	134	0	0	205	263	32	0	148	194	132	26

Table A.3: Results for different filter and option settings and heterozygous scenario in Section 4.1.3.

cov	cs	nm	u	T0F1	T0F2	T0FX	T1F0	T1F1	T1F2	T1FX	T2F0	T2F1	T2F2	T2FX
20	3	0	0	31	0	0	225	274	1	0	178	216	106	0
20	3	1	0	78	0	0	138	357	5	0	116	197	186	1
60	3	0	0	209	0	0	71	423	6	0	69	139	288	4
60	3	1	0	638	0	0	61	391	48	0	57	124	293	26

Table A.4: Results with BreakDancer in Section 4.1.4.

cov	scenario	nm	T0F1	T0F2	T0FX	T1F0	T1F1	T1F2	T1FX	T2F0	T2F1	T2F2	T2FX
20	hetero	1	17	0	0	286	214	0	0	196	90	214	0
20	hetero	0	4	0	0	372	128	0	0	286	63	151	0
20	homo	1	21	0	1	286	214	0	0	153	88	259	0
20	homo	0	4	0	1	370	130	0	0	225	69	206	0
60	hetero	1	316	0	0	123	377	0	0	101	95	304	0
60	hetero	0	84	0	0	172	328	0	0	126	91	283	0
60	homo	1	304	0	0	122	378	0	0	83	104	313	0
60	homo	0	94	0	0	171	329	0	0	101	94	305	0

Bibliography

[1] Roland Wittler. Unraveling overlapping deletions by agglomerative clustering. *BMC genomics*, 14(Suppl 1):S12, 2013.

[2] Suzanne Sindi, Elena Helman, Ali Bashir, and Benjamin J Raphael. A geometric approach for classification and comparison of structural variants. *Bioinformatics*, 25(12):i222–i230, 2009.

[3] Ruibin Xi, Tae-Min Kim, and Peter J Park. Detecting structural variations in the human genome using next generation sequencing. *Briefings in functional genomics*, 9(5-6):405–415, 2010.

[4] Benjamin J Raphael. Chapter 6: Structural variation and medical genomics. *PLoS Comput Biol*, 8(12):e1002821, 12 2012.

[5] Erin D Pleasance, R Keira Cheetham, Philip J Stephens, David J McBride, Sean J Humphray, Chris D Greenman, Ignacio Varela, Meng-Lay Lin, Gonzalo R Ordóñez, Graham R Bignell, et al. A comprehensive catalogue of somatic mutations from a human cancer genome. *Nature*, 463(7278):191–196, 2009.

[6] Gabbert. Krebsentstehung - Was ist Krebs? `http://www.krebsgesellschaft.de/krebsentstehung,11266.html`, May 2011.

[7] Charles L Sawyers. Chronic myeloid leukemia. *New England Journal of Medicine*, 340(17):1330–1340, 1999.

[8] Tetsuya Mitsudomi and Yasushi Yatabe. Epidermal growth factor receptor in relation to tumor development: EGFR gene and cancer. *FEBS Journal*, 277(2): 301–308, 2010.

[9] Susumu Kobayashi, Titus J Boggon, Tajhal Dayaram, Pasi A Jänne, Olivier Kocher, Matthew Meyerson, Bruce E Johnson, Michael J Eck, Daniel G Tenen, and Balázs Halmos. EGFR mutation and resistance of non–small-cell lung cancer to gefitinib. *New England Journal of Medicine*, 352(8):786–792, 2005.

[10] Hiroyuki Yasuda, Susumu Kobayashi, and Daniel B Costa. EGFR exon 20 insertion mutations in non-small-cell lung cancer: preclinical data and clinical implications. *The lancet oncology*, 13(1):e23–e31, 2012.

[11] Sabina Solinas-Toldo, Stefan Lampel, Stephan Stilgenbauer, Jeremy Nickolenko, Axel Benner, Hartmut Döhner, Thomas Cremer, and Peter Lichter. Matrix-based comparative genomic hybridization: biochips to screen for genomic imbalances. *Genes, chromosomes and cancer*, 20(4):399–407, 1997.

[12] Kai Ye, Marcel H Schulz, Quan Long, Rolf Apweiler, and Zemin Ning. Pindel: a pattern growth approach to detect break points of large deletions and medium sized insertions from paired-end short reads. *Bioinformatics*, 25(21):2865–2871, 2009.

[13] Iman Hajirasouliha, Fereydoun Hormozdiari, Can Alkan, Jeffrey M Kidd, Inanc Birol, Evan E Eichler, and S Cenk Sahinalp. Detection and characterization of novel sequence insertions using paired-end next-generation sequencing. *Bioinformatics*, 26(10):1277–1283, 2010.

[14] Can Alkan, Jeffrey M Kidd, Tomas Marques-Bonet, Gozde Aksay, Francesca Antonacci, Fereydoun Hormozdiari, Jacob O Kitzman, Carl Baker, Maika Malig, Onur Mutlu, et al. Personalized copy number and segmental duplication maps using next-generation sequencing. *Nature genetics*, 41(10):1061–1067, 2009.

[15] Chao Xie and Martti Tammi. CNV-seq, a new method to detect copy number variation using high-throughput sequencing. *BMC bioinformatics*, 10(1):80, 2009.

[16] Ken Chen, John W Wallis, Michael D McLellan, David E Larson, Joelle M Kalicki, Craig S Pohl, Sean D McGrath, Michael C Wendl, Qunyuan Zhang, Devin P Locke, et al. BreakDancer: an algorithm for high-resolution mapping of genomic structural variation. *Nature methods*, 6(9):677–681, 2009.

[17] Fereydoun Hormozdiari, Can Alkan, Evan E Eichler, and S Cenk Sahinalp. Combinatorial algorithms for structural variation detection in high-throughput sequenced genomes. *Genome research*, 19(7):1270–1278, 2009.

[18] Zhengdong D Zhang, Jiang Du, Hugo Lam, Alex Abyzov, Alexander E Urban, Michael Snyder, and Mark Gerstein. Identification of genomic indels and structural variations using split reads. *BMC genomics*, 12(1):375, 2011.

[19] Jan O Korbel, Alexander Eckehart Urban, Jason P Affourtit, Brian Godwin, Fabian Grubert, Jan Fredrik Simons, Philip M Kim, Dean Palejev, Nicholas J Carriero, Lei Du, et al. Paired-end mapping reveals extensive structural variation in the human genome. *Science*, 318(5849):420–426, 2007.

[20] Inc Illumina. Paired-end Sequencing Assay. `http://www.illumina.com/technology/paired_end_sequencing_assay.ilmn`, November 2013.

[21] *The SAM Format Specification*. The SAM Format Specification Working Group, v1.4 edition, October 2013.

[22] Heng Li, Bob Handsaker, Alec Wysoker, Tim Fennell, Jue Ruan, Nils Homer, Gabor Marth, Goncalo Abecasis, Richard Durbin, et al. The sequence alignment/map format and samtools. *Bioinformatics*, 25(16):2078–2079, 2009.

[23] Picard. `http://picard.sourceforge.net/`, November 2013.

[24] Samuel Levy, Granger Sutton, Pauline C Ng, Lars Feuk, Aaron L Halpern, Brian P Walenz, Nelson Axelrod, Jiaqi Huang, Ewen F Kirkness, Gennady Denisov, et al. The diploid genome sequence of an individual human. *PLoS biology*, 5(10):e254, 2007.

[25] S J John. SimSeq. `https://github.com/jstjohn/SimSeq`, December 2011.

[26] Heng Li and Richard Durbin. Fast and accurate short read alignment with burrows-wheeler transform. *Bioinformatics*, 25(14):1754–1760, 2009.

[27] Helga Thorvaldsdóttir, James T Robinson, and Jill P Mesirov. Integrative genomics viewer (igv): high-performance genomics data visualization and exploration. *Briefings in bioinformatics*, 14(2):178–192, 2013.

[28] James T Robinson, Helga Thorvaldsdóttir, Wendy Winckler, Mitchell Guttman, Eric S Lander, Gad Getz, and Jill P Mesirov. Integrative genomics viewer. *Nature biotechnology*, 29(1):24–26, 2011.

[29] Tobias Marschall, Ivan G Costa, Stefan Canzar, Markus Bauer, Gunnar W Klau, Alexander Schliep, and Alexander Schönhuth. Clever: clique-enumerating variant finder. *Bioinformatics*, 28(22):2875–2882, 2012.

[30] Matteo Chiara, Graziano Pesole, and David S Horner. Svm2: an improved paired-end-based tool for the detection of small genomic structural variations using high-throughput single-genome resequencing data. *Nucleic acids research*, 40(18):e145–e145, 2012.