Dissertation

submitted to the

Combined Faculties for the Natural Sciences and for Mathematics

of the Ruperto-Carola University of Heidelberg, Germany

for the degree of

Doctor of Natural Sciences

presented by

Paul-Theodor Pyl, MSc.

born in Berlin

Oral-examination:

Wednesday, 26th of November 2014 – 09:00

Method development for comparative cancer genomics

Referees:

Prof. Dr. Paul Bertone

and

Prof. Dr. Robert Bruce Russell

1 Introduction

1.1 Abstract -- English

With the cost of sequencing continuously dropping and the increased availability of sequencing technologies the coming years will bring a wealth of sequencing data that will be tremendously interesting and challenging to analyse and interpret. It is becoming more and more clear that both the algorithmic approaches as well as the handling of the data itself will prove challenging and some of the legacy approaches and file formats that were developed in the wake of the first large-scale sequencing projects (e.g. the human genome sequencing project) will prove unsuitable or at least inconvenient to use once projects that aim to analyse sequencing data from thousands of samples become the norm rather than the exception.

In this thesis I will discuss my work in the field of sequencing analysis during my PhD studies. I will touch upon some of the challenges and problems researchers are facing in the field today and will present the approaches and solutions I have developed to deal with those issues. To this end I will present my work in methods development for sequencing analysis in general and specifically in the context of cancer genomics and give examples of the application of those methods in projects that I have been involved in. My two main contributions to available methods for sequencing analysis are the **`HTSeq`** Python Library and the **`h5vc`** `R/Bioconductor` package (Anders et al., 2014; Pyl et al., 2014). I co-developed the former with Simon Anders and am the lead developer of the latter. Both pieces of software are available through public repositories, and are well-documented and -maintained.

The projects in which those methods have found application are the HeLa Kyoto sequencing project (Landry et al., 2013) and a set of three cancer genomics projects involving cohorts of up to 18 whole genome sequencing (WGS) samples and up to 21 whole exome sequencing (WES) samples, respectively. I will discuss my methodological contributions to these projects as well as relevant biological results in Section 5.

1.2 Abstract -- German

Sowohl die Reduktion der Kosten als auch die immens gesteigerte Verfügbarkeit von Sequenziertechnik führt zu einem immer größer werdenden Fundus von Sequenzier-Datensätzen. Dies ist ein Trend, welcher sich in den kommenden Jahren noch verstärken wird. Die konsequente Anwendung der verfügbaren Sequenziertechnologien auf vorhandene biologische Proben wird in naher Zukunft eine Menge an Daten produzieren, die in der Geschichte der Bioinformatik ihresgleichen sucht. Die Handhabung und Analyse dieser Datenmengen stellt die Bioinformatik vor eine neue Klasse technischer und algorthmischer Probleme, welche mit den aktuell verfügbaren Werkzeugen und Dateiformaten schwierig zu bewältigen sein wird. Es ist schon jetzt offensichtlich, dass einige der algorithmischen Herangehensweisen und traditionellen Dateiformate dieser Aufgabe nicht gewachsen sind.

In einer Zeit, in der die gleichzeitige Analyse tausender sequenzierter Proben mehr und mehr zur Norm wird, ist es wichtig die verfügbaren Infrastruktur auf diese herannahende Datenflut vorzubereiten, bevor große, ambitionierte Projekte, wie zum Beispiel die Re-Analyse sämtlicher Krebs-Proben aus den ICGC und TCGA Konsortien (mehrere 1000 Datensätze), auf Grund von technischen Problemen ins Stocken geraten.

In meiner Dissertation werde ich meine Tätigkeit im Feld der Sequenzanalyse im Rahmen meiner Zeit als Doktorand in der Grupper von Dr. Wolfgang Huber diskutieren. Ich werde einige der Probleme und Herausforderungen besprechen, denen Wissenschaftler in der Analyse von Sequenzierdaten begegnen und meine eigenen Lösungsansätze präsentieren. Ich werde meine Arbeit auf dem Feld der Sequenzanalyse im Allgemeinen diskutieren und im Speziellen die Entwicklung meiner methodischen Ansätze im Feld der Krebsforschung mittels Sequenzanalyse besprechen.

Meine zwei hauptsächlichen Beiträge verfügbarer Methoden zur Sequenzanalyse sind das **`HTSeq`** Modul für die Programmiersprache Python und das `R/Bioconductor` -Paket **`h5vc`** (Anders et al., 2014; Pyl et al., 2014). An **`HTSeq`** habe ich zusammen mit meinem Kollegen Dr. Simon Anders gearbeitet, während ich der alleinige Hauptentwickler von **`h5vc`** bin. Beide Tools sind öffentlich als wohl-dokumentierte und regelmässig aktualisierte open-source Software verfügbar.

Ich habe sowohl **`HTSeq`** als auch **`h5vc`** in diversen Sequenzierprojekten zur Anwendung gebracht, darunter ein Projekt zur genomischen Charakterisierung der weit verbreiteten HeLa-Kyoto Zellline (Landry et al., 2013). Des Weiteren habe ich an drei Krebssequenzierprojekten mitgewirkt, welche sich mit der genomischen Analyse von Kohorten von bis zu 21 sequenzierten Proben beschäftigen. Ich werde meine methodologischen Beiträge zu diesen Projekten sowie relevante biologische Resultate in Abschnitt 5 diskutieren.

Contents

Acronyms

AML acute myeloid leukaemia. 58

CN copy number. 20

COSMIC Catalogue Of Somatic Mutations In Cancer. 111

dbSNP The Single Nucleotide Polymorphism Database. 26

EVP Ensembl Variant Effect Predictor. 76

GATK Genome Analysis Toolkit. 31

GMAF global minor allele frequency. 52

HLA human leukocyte antigen. 57

HPV human papillomavirus. 23

HSC haematopoietic stem cell. 57

HTS high-throughput sequencing. 10

InDel small insertion or deletion. 19

LoH loss of heterozygousity. 20

NIH National Institutes of Health. 30

pre-LSC pre-leukaemic stem cell. 118

SINE short interspersed element. 64

SNP single nucleotide polymorphism. 32

SNV single nucleotide variant. 19

SV structural variant. 28

T-ALL T-Cell acute lymphoblastic leukaemia. 58

TSS transcription start site. 10

WAS Wiskott-Aldrich syndrome. 57

WES whole exome sequencing. 5

WGS whole genome sequencing. 5

2 Processing sequencing data with HTSeq

HTSeq is a module for processing sequencing data, written for the Python[1] programming language. I co-developed HTSeq with my colleague Simon Anders and our manuscript has been accepted for publication and is available online as a pre-print (Anders et al., 2014). Furthermore, we provide extensive online documentation[2]. In the following sections I will give a brief overview of HTSeq and its functionality and touch upon some of the fundamental concepts that make this library a valuable tool for genomics researchers. Since the initial release of HTSeq in 2010 the package has found considerable use in the research community with ∼5000 downloads per month as reported at `https://pypi.python.org/pypi/HTSeq`. We have an application note under review in *Bioinformatics* and the preprint (available through biorxiv.org[3]) has already been cited 15 times according to Google Scholar (reported on 02.09.2014).

2.1 Introduction

Over the last years high-throughput sequencing (HTS) has seen a rapid technological advance that lead to the development of a multitude of assays based on short-read HTS technologies. The large quantities and different kinds of data resulting from the application of those assays necessitated the development of bioinformatics pipelines suitable to tackling the tasks at hand. For many of the recurring steps in HTS data analysis, like alignment and assembly of sequencing reads, a wide range of tools have been developed (e.g. Li and Durbin (2009), Zerbino and Birney (2008), McKenna et al. (2010)). While most of the standard tasks in HTS analysis can be achieved using those tools, there is still demand for custom scripts in non-standard pipelines that explore new types of analysis or diverge from the typical approach to a task. Usually the need for custom scripts arises after the big common steps in an analysis pipeline have been performed and a set of project-specific questions need to be answered. Other smaller tasks that need to be tackled in a bioinformatics analysis typically are preprocessing steps such as trimming low-quality ends, demultiplexing and removing adapter sequence as well as downstream analyses. Typical downstream analyses are the estimation of transcript abundance in RNA-Seq experiments by counting read alignments overlapping annotated features in gene models, the generation of aggregate coverage profiles around features of interest (e.g. transcription start site (TSS)) from ChIP-Seq data and a multitude of other tasks. Although there exists stand-alone solutions for some of those small tasks they are usually still focused on answering a very specific question and lack flexibility when questions

[1] `https://www.python.org/`
[2] `http://www-huber.embl.de/HTSeq/doc/overview.html`
[3] `http://biorxiv.org/content/early/2014/08/19/002824`

are different from the standard work-flow. With the current state of the art in bioinformatics analysis of HTS data only the most common and widely used workflows can be performed without the need to write custom scripts and a lot of the work of bioinformaticians is to write scripts to glue together the more commonly established steps of an analysis pipeline. HTSeq facilitates the writing of those scripts and allows for the implementation of analysis pipelines dealing with HTS data in a variety of formats. It provides parsers for importing a variety of types of input data in their most common formats (e.g. SAM and BAM import for read alignments). HTSeq comes with extensive documentation, including a tutorial that demonstrates the use of HTSeq's core classes and discusses several important use cases in detail. The documentation, as well as HTSeq's design, is geared towards allowing users with only moderate Python knowledge to create their own scripts, while shielding more advanced internals, such as the use of Cython (Behnel et al., 2011) to enhance performance, from the user.

2.2 Streaming processing of sequencing data

A central design paradigm of HTSeq is the processing of data in a streaming fashion which does not load a whole dataset into memory in order to process it, but rather iterates through the raw data (usually read alignments) and collects some form of summary information from them (e.g. the read coverage overlapping a region of interest in the genome). Using this approach one can handle the vastly increased size of the raw data generated with HTS assays when compared to previous technologies, e.g. microarrays, which are typically analysed by loading the raw data (probe intensity values) into memory and performing the processing and analysis on the fully loaded raw data. The size of datasets is a problem that is more apparent in sequencing assays targeting organisms with comparatively large genomes; e.g. sequencing data from human samples can easily use hundreds of Gigabytes just for storing read alignments. Although the ever increasing affordability and availability of compute resources (CPU power and available RAM) makes it more feasible to work with HTS data in a similar way as is typically done in microarrays, currently servers with a Terabyte of RAM are the exception rather than the rule which makes the in-memory analysis of HTS data from larger genomes challenging. Furthermore, the increases in available compute resources are offset by the increased availability of ever larger sequencing data-sets through increased depth of coverage and bigger read-lengths.

As mentioned previously most analyses are likely to focus on some form of summary information of the raw read alignment data, therefore processing usually happens on a per-alignment or per-read level. HTSeq provides an implementation of iterators for typical HTS file types (FASTA, BAM, SAM, GTF, VCF, etc.). In this way even large datasets can be processed using relatively small amounts of memory, since the results of the processing steps are commonly much smaller than the raw data. An example is the per-gene count matrix of a human RNA-Seq sample, which will fit comfortably into the RAM of a modern computer. The underlying raw data, i.e. the read alignments in BAM format, can be many times larger depending on sequencing depth. This touches upon another important factor, which is that the depth to which we can feasibly sequence samples is increasing and compensates for increases in available compute resources, while most types of summary information will not scale with sequencing depth; e.g. the coverage of a specific position in the genome will, for the foreseeable future, be a number that can be stored as a 32-bit integer (read depths between 0 and ~ 4 billion reads per position). The nucleotide tally approach to HTS analysis which I introduce in Section 4.1 is an example of using summary information that is (almost) independent of sequencing depth to reduce the memory footprint of the analysis that was necessitated by the inconvenience of working with the raw data in a larger cohort of sequencing samples. Within the HTSeq framework the iteration through HTS data files is implemented through a set of two classes for each file type, a data record class (e.g. the `VariantCall` class for entries to a VCF file) and a parser class (e.g. the `VCF_Reader` class, which streams through a VCF file, returning `VariantCall` objects for each of the entries). The parser class handles the file and performs the iteration which yields an object of the associated data record class for each entry (e.g. alignment, variant calls, etc.). For BAM files the parser additionally supports random access by retrieving all alignments overlapping a specified genomic range.

2.3 Storing position specific genomics data – the GenomicArray

An important task in the analysis of HTS data is the collection of data about read alignments that overlap specific genomic positions, i.e. loci of interest like gene annotations, variant calls or viewports in HiC experiments (see Zhao et al. (2006) and Simonis et al. (2006)). The ability to efficiently store, manipulate and query data associated with genomic position is one of the central requirements when developing HTS analysis tools. The genomic positions provide the coordinate system in which most data is expressed and will be interacted with in HTS analyses, since most of those analyses focus on read alignments and their locations on the genome or in relation to other loci of interest. HTSeq provides the `GenomicArray` class, which implements an associative container that stores values associated with positions in the genome. A `GenomicArray` stores these values in a memory-efficient way and implements fast random read and write access routines, which allow the user to get and set values associated with genomic positions. A `GenomicArray` stores a `StepVector` for each chromosome and strand of the genome, where a `StepVector` is a representation of the vector of values associated with positions in the chromosome encoded through their two properties *position* and *value*. In this way the `StepVector` stores the data as a set of steps that are defined by their respective starting position and value. This approach of storing a vector of values as a run-length encoding (i.e. a series of steps) performs lossless data compression. The efficiency of the compression depends on the sparseness and variability of the data; e.g. a `StepVector` of values that are different at each of the n bases of a chromosome will be very inefficient to store since a total of n steps are needed and will actually use more memory than a simple array of values would have (n values and n positions compared to only n values). On the other hand, the coverage vector of a typical RNA-Seq or whole exome sequencing experiment on a human sample will be 0 for most of the genome (i.e. in the introns and intergenic regions) and only the $\sim 3\%$ of the genome that are actually coding will have values different from 0. Therefore the step representation of e.g. an RNA-Seq experiment will store each intergenic region or intron as a single pair of numbers, i.e. the starting position and value (0 in this case), irrespective of the actual length of the intron or intergenic region. For analyses that do not rely on coverage but rather on mismatches (such as calling single nucleotide variants) the read-length does not influence step size but rather the relative sparsity of SNVs and the expected error rate of the sequencing machine will guarantee that in most samples the mismatches will be sparsely distributed along the genome. Many of the standard file formats for storing data associated with genomic position use some form of run-length encoding, e.g. WIG or BED (Kent et al., 2010) files usually store data in the form of steps of a given (fixed or variable) length.

In conclusion the usage of run-length encoding to store data associated with genomic position (as implemented in the `GenomicArray` and `StepVector` classes provided by HTSeq) can be expected to yield favourable compression in many use cases and will allow the user to interact with and analyse such data efficiently and fast within the memory of typical modern desktop computer. In this way the hard to manage raw data (i.e. potentially hundreds of gigabytes of read alignments in BAM format) can be processed to a level of summary information and stored in a `GenomicArray` which will then be the basis of further analysis.

An example of a task that can be easily accomplished using this technology (streaming processing of files and storage of the data in a `GenomicArray` object) is the loading of a set of gene models from a GTF file to associate some identifying information for each exon of each gene to the genomic positions overlapping that exon by using a `GenomicArray`. This `GenomicArray` can then be queried with the genomic positons overlapping read alignments loaded from a BAM file generated in e.g. an RNA-Seq experiment in order to generate a per-exon count table (see Section 2.5 and Anders and Huber (2010); Anders et al. (2012) for details).

2.4 Implementation and Availability

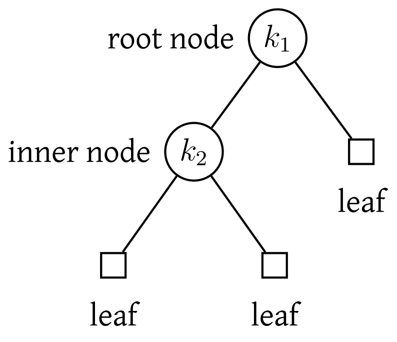
HTSeq is implemented in Python and uses SWIG and Cython (Beazley (1996), Behnel et al. (2011)) to interface with high-performance C++ code used to implement the `StepVector`. I implemented the functionality for reading and writing to BAM files by using the pysam[4] Python module. HTSeq is released as open-source software under the GNU General Public Licence and available from the Huber group's web-page[5] or from the Python Package Index[6].

[4]https://github.com/pysam-developers/pysam
[5]http://www-huber.embl.de/HTSeq
[6]https://pypi.python.org/pypi/HTSeq

2.4.1 The `StepVector` data-structure

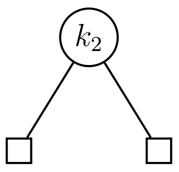
The `StepVector` is a class implemented in HTSeq that stores a vector of values encoded as a series of steps which are defined by their starting position and value. Internally the `StepVector` is implemented through the use of the `map` container that is provided by the C++ standard template library (STL – Becker (2011)). The `map` container holds key–value pairs and is typically implemented as a red-black-tree (Bayer, 1972) which is a form of balanced binary tree that guarantees $O(\log(n))$ access time for random lookups in a tree that holds n elements. Such a self-balancing binary tree has routines that make sure that the height of the tree always stays at approximately $\log(n)$ when operations that potentially change the tree topology occur, e.g. insertion or removal of key-value pairs.

A binary tree is a data-structure that is composed of nodes and leafs. While nodes contain keys, leafs do not contain keys and are essentially place-holders. For a more mathematically precise definition of a binary trees, have a look at Garnier and Taylor (2009). Every node must be connected to exactly two child nodes: one left and one right child. The children of a node can themselves be nodes or leafs whereas leafs do not have children. An example tree is shown here:



Nodes are symbolised as circles and leafs are symbolised as squares and the keys associated with the nodes (k_1 and k_2 in this case) are shown inside the nodes. The root node is special, since it does not have a parent node and all tree operations start there.

A sub-tree is defined as all nodes that lie below a specified node, for example in the tree shown above the sub-tree rooted in the inner node with the key k_2 looks like this:



This sub-tree is called the left sub-tree of the root node, since it is rooted in the left child node of the root. In order for a binary tree to function as a search tree the nodes in the tree must fit the following criteria:

1. Each node i is associated with a key k_i

2. The keys of all nodes in its left sub-tree must be smaller or equal to k_i

3. The keys of all nodes in its right sub-tree must be larger than k_i

To find a given key s in a binary search tree we can implement a simple algorithm called binary search. For this we set the current node c to the root of the tree and then apply the following steps:

1. If the current node is a leaf the search ends and the key was not found in the tree. If we want to insert s into the tree we have to replace the leaf we found with a node that has s as its key.

2. Compare s to the key of the current node c:

   $s < k_c$ The key must be in the left subtree of the current node, so we set the current node to the left child of node c

   $s = k_c$ We have found the key, it is in node c

   $s > k_c$ The key must be in the right subtree of the current node, so we set the current node to the right child of node c

3. Go back to step 1

It should be immediately apparent that this means that the lookup of any key s in a binary search tree is implemented as a traversal of the tree, starting at the root node.

Figure 2.1 illustrates the effects of balancing a binary search tree. The two trees that are shown are both binary search trees containing the keys from 1 to 9. While the tree in panel **(a)** is unbalanced, the tree in panel **(b)** is balanced. In both trees the path that needs to be traversed to find the key 7 is marked in colour. From this example it should be immediately obvious, that balancing a tree reduces the worst-case run-time needed for lookup of a key. The function of balanced binary search trees in the implementation of the `StepVector` is illustrated in Figure 2.2.

For those cases where the data is very variable from position to positon, i.e. the average step-size is approaching 1, we implement plain vector storage in RAM or in memory-mapped files, i.e. an on-disk representation of the vector that is accessible in the same way as a vector stored in RAM would be. Note that using memory mapped files can reduce the speed of a script drastically, depending on the performance of the storage solution. The choice of back-end (balanced binary tree, linear array or memory mapped file) does not influence the interface for programmatic access to the data which allows for reusability of algorithms between different types of HTS data that each require different storage mechanisms.

Using this type of run-length encoding to store a vector as a series of steps is memory efficient and offers lossless compression if the vector itself has long stretches of identical values. An example where this compression is very efficient is the storage of a coverage vector

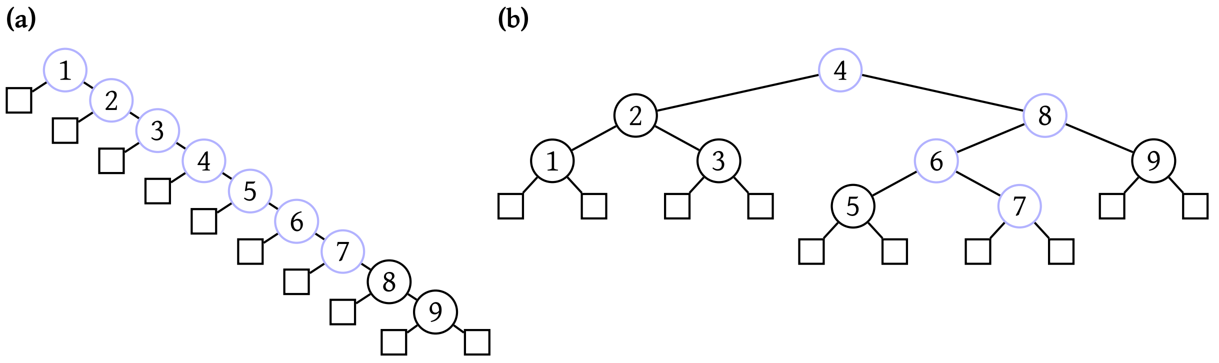


Figure 2.1: Examples of binary search trees. **(a)** unbalanced binary search tree. **(b)** balanced binary search tree. The search path to key 7 is marked in blue. Note the reduced number of steps needed in the balanced tree.

of an RNA-Seq experiment. Such a coverage vector will have long stretches of value 0 that are caused by introns and intergenic regions where no reads map. Assuming a human sample with $\sim$ 3% of the genome being expressed, this means that 97% of the genome will have a coverage of zero (bar some mis-mapped reads) and will essentially be represented by a couple of thousand steps, i.e. the introns and intergenic regions.

2.5 Examples of usage

Probably the most widely used application that utilises HTSeq is the `htseq-count` script that is part of HTSeq. This script is heavily relied upon by the DESeq and DEXSeq (Anders and Huber (2010); Anders et al. (2012)) R/Bioconductor packages which are extensively used for differential expression analysis of RNA-Seq data. The `htseq-count` script creates a table of alignment counts per gene or exon from an input SAM or BAM file and a GTF file (gene-models) by finding overlaps between the read alignments stored in the BAM file and the gene-models defined in the GTF file. The script uses the two important coding strategies of HTSeq described earlier, which are the `GenomicArray` for storing the gene-models in a memory-efficient way that can be easily queried with the genomic intervals associated with the read alignments and the iterative processing of the BAM file, since the read alignments can be read in one by one and their overlaps with gene models can be determined independently from each other. The HTSeq and Python functions used to implement such types of HTS data analysis pipelines are extensively documented online[7]. Listings 1 and 2 show code snippets that are typically used in creating custom HTS data analysis pipelines (e.g. the htseq-count script) and illustrate that those typical processing steps (reading alignments from a BAM file etc.) can be implemented with HTSeq in a few lines of relatively easy to read code.

[7] `http://www-huber.embl.de/HTSeq/doc/overview.html`

Key	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Value	0	0	0	0	3	3	3	17	17	19	19	19	19	5	5	5	9	9

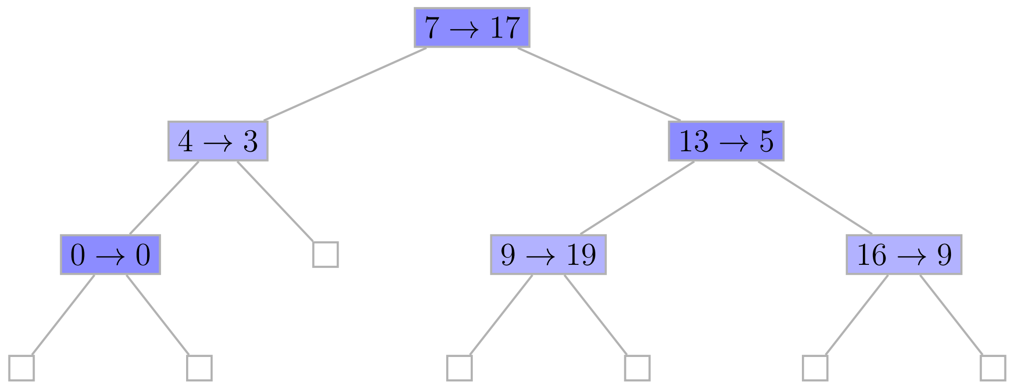


Figure 2.2: Example of the storage of a vector of values as a run-length encoding in a binary search tree. The table shows the value at each position and the binary search tree shown below illustrates the translation of the vector of values into a set of steps with keys and associated values. Each node in the tree is shown as a box and the association between the key k and value v is given as $k \to v$. Note that the search tree is defined on the keys only whereas the associated values do not have to comply with the ordering required of a search tree. Through this layout of the tree we enable fast ($O(\log n)$) access to the value corresponding to any position via binary search. The steps are colour-coded in the table and the nodes in the tree have colours matching the steps they correspond to.

```python
import HTSeq
codingSequence = HTSeq.GenomicArray('auto') #Create an empty GenomicArray
for record in HTSeq.GFF_Reader("example.gtf"): #go through 'example.gtf'
   if record.type == "CDS":
      codingSequence[record.iv] = record.attr["gene_name"]
```

Listing 1: Example code demonstrating the iterative processing of a GTF file. This code reads entries from a GTF file and stores gene names in the intervals associated with coding sequence of the respective gene using a `GenomicArray`.

```python
import collections
geneCounts = collections.defaultdict(int)
for read in HTSeq.BAM_Reader("example.bam"):
   if read.aligned:
      matchingGenes = set(list(codingSequence[read.iv].steps()))
      if len(matchingGenes) == 1: #uniquely mapped to one gene
         geneCounts[matchingGenes[0]] += 1
```

Listing 2: Example code demonstrating the iterative processing of a BAM file for the purpose of matching alignments to the coding intervals generated in Listing 1.

The algorithm used for virus integration site detection in the HeLa genome project (described in Section 3.2.3) and the similar algorithm used to detect vector integration sites in samples from a study into retroviral gene therapy (Section 5.1.2), are both examples of software that was easy to develop when using HTSeq to implement the algorithms and might otherwise have taken considerably more time and effort to complete.

2.6 Summary

HTSeq provides a framework for the rapid prototyping of HTS analysis software and can considerably ease the interfacing between different monolithic tools in a pipeline since it provides a wider selection of parsers for many of the commonly used data formats. HTSeq is designed to be useable by researchers with only moderate knowledge of Python programming and implements concepts that are especially useful in the context of HTS data analysis, like the streaming processing of data and the ability to efficiently store, look-up and modify data associated with genomic locations implemented through the `GenomicArray` and `StepVector` class.

There are other approaches with the goal of achieving functionality comparable to what HTSeq provides, e.g. the `*Ranges` infrastructure included in the `Bioconductor` framework. This is implemented in set of packages that provide classes that are designed specifically to represent data associated with (genomic) intervals. One example is the `GenomicRanges` class that holds a set of genomic intervals and associated data (Lawrence et al., 2013). Functions for manipulation and analysis of data represented as `*Ranges` is provided through a collection of `Bioconductor` packages.

To me HTSeq was tremendously helpful in the implementation of analysis pipelines for the biological projects I describe in Sections 3 and 5, where I had to implement custom analysis software for specific tasks, e.g. the detection of virus integration sites in the HeLa genome analysis project, which I will describe in the next section.

3 The HeLa Genome Project

In this chapter I will describe my work on the HeLa-Kyoto genome project which was published as a shared co-first-authorship between Jonathan Landry and myself (Landry et al., 2013). Some of the figures and tables are taken from our publication (Landry et al., 2013) or its supplementary material.

3.1 Introduction

The HeLa cell line is one of the most widely used model cell lines for studying human cellular and molecular biology and prior to our publication (Landry et al., 2013) no genomic reference for this cell line had been available to the research community. As a consequence of this lack of a high-resolution reference, the experimental design and interpretation of results from assays using HeLa cells was instead performed in the context of the human reference genome (e.g. GRCh37[1]).

However accurate genomic information is essential to the design of effective molecular genetic studies and the ability to interpret the results of such studies. The aim of our study was to provide such information for a HeLa Kyoto cell line. We performed DNA and RNA sequencing of a HeLa Kyoto cell line and analysed the data for different types of genomic mutations, e.g. single nucleotide variants (SNVs), small insertions or deletions (InDels) as well as large chromosomal aberrations (i.e. deletion, duplications, inversion and translocation events). We also characterised the expression profile of the cell line including the analysis for allelic expression in coding mutations (specifically SNVs).

Our results provided the first detailed account of genomic variants in the HeLa genome, yielding insight into their impact on gene expression and cellular function as well as their origins. In addition our analysis piloted workflows that can be used by the research community to regularly characterise the cell lines they are using.

In the following sections I will highlight major findings from our publication as well as specific methodological contributions that I made to the project. I will focus on the genomics analyses, since my contribution to the project was mainly concerned with those. For a complete description including all methods, results and discussions, please refer to the publication (Landry et al., 2013) and the supporting material.

[1]http://www.ncbi.nlm.nih.gov/projects/genome/assembly/grc/human/

3.2 Methods

In the methods section I will focus on analysis steps for which I was the main researcher. The two concrete examples are the estimation of copy number states and the virus integration site detection. I implemented the pipeline using stand alone tools and custom scripts using shell scripting, Python scripting and HTSeq as well as R/Bioconductor for the downstream analysis.

3.2.1 Estimating copy number states

After fully preprocessing the alignments, I computed the depth of coverage in 10-kb bins along the genome and applied corrections for mappability and GC-dependent biases. The coverage calculation was implemented using Python/HTSeq and R/Bioconductor. The mappability correction was based on a custom mappability track in which a position in the genome was called mappable if and only if a read of length 101nt (the same length of reads as our library had generated) originating at that position could be uniquely mapped back to its original position using our alignment pipeline with the exact same parameters as we used to align the DNA-Seq library from our HeLa cell line. This means that there exists no alignments with the same score at other positons for a mappable read. For each 10-kb bin I divided the observed read count by the proportion of mappable positions and discarded all bins where the mappability was less than 0.5 (i.e. more than 50% of the positions in the bin were not mappable). Furthermore I performed GC-bias correction to correct for GC-dependent changes in sequencing depth. To this end I calculated for each 10-kb bin the GC-content (the proportion of reference bases that are G or C) and used a local robust fit (as implemented in the `locfit.robust` function within R) of GC-content and read-count per bin. I picked a desired median coverage of 60 bp and then used the ratio of the fit and that desired coverage as a correction factor by which to multiply observed coverages per bin based on the respective bins GC-content. The effect of the GC correction is illustrated in Figure 3.1. This type of adjustment for GC-content biases in the coverage is based on the assumption, that the copy number state of a region is independent of its GC-content. In both panels of Figure 3.1 we can observe a banding in the plots, representing regions of copy number (CN) 2 and 3, respectively. The majority of the data in those bands covers the full range of observed values for the GC-content (30% – 60%). We can conclude that indeed the copy number state of a region does not seem to depend on its GC-content.

To describe the extent of CN aberrations in our HeLa cell line I created a segmentation of the genome according to integer CN. This track was generated with the R/Bioconductor package DNACopy (Seshan and Olshen (2006)) followed by mixture model fitting. Since DNACopy is designed for working on $\log_2$-ratios of samples (e.g. $\log_2 \frac{\text{cancer}}{\text{control}}$) I transformed our data by using the following formula: $f(x) = \log_2 \frac{x}{x_0}$ on the GC-adjusted depth-of-coverage data x, where x_0 is the median GC-adjusted coverage of a manually curated region of CN 2 on chro-

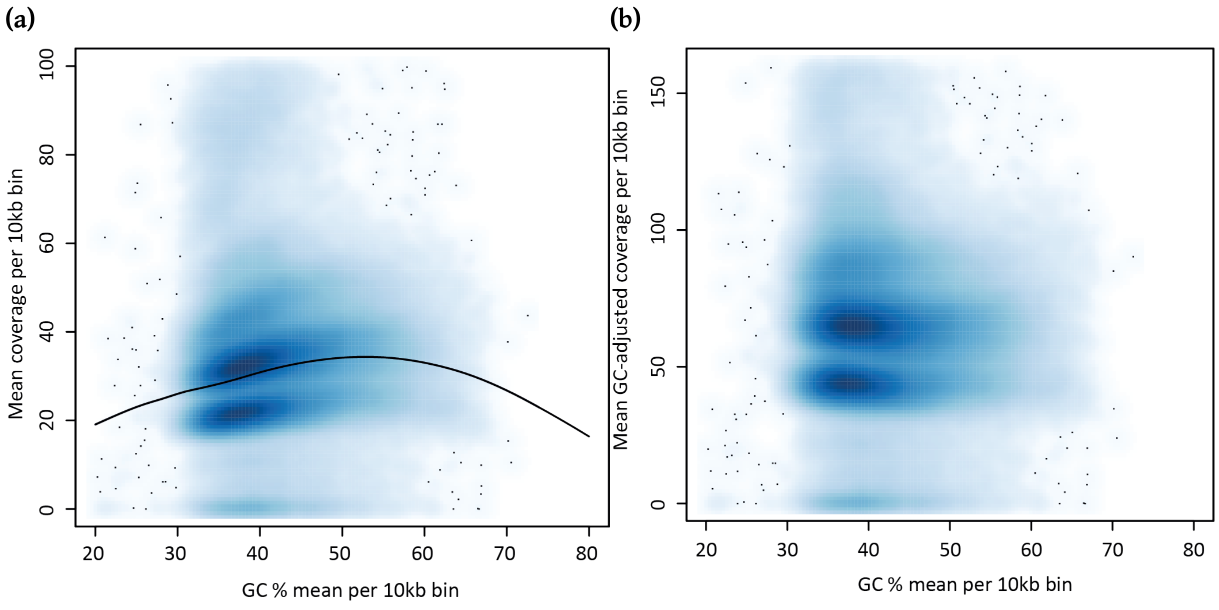


Figure 3.1: GC-bias correction. The panels show the dependency between GC-content and coverage in 10-kb bins in panel **(a)** (the fit is shown as a black line) and the effect of GC bias correction in panel **(b)**.

mosome 4 (see Figure 3.2). On those log2-ratio values I ran the normal DNACopy workflow for segmentation with the parameters `undo.splits = "sdundo"` (remove break points that split segments which are less than `undo.SD` standard deviations apart) and `undo.SD = 2` (setting the cut-off for removal of break points to 2 standard deviations). The result was a segmentation with segments defined by their start, end and average log2-ratio s, from which I calculated the respective average copy number as 2^{s+1}. Finally I fitted a mixture model of $m = 8$ normal distributions with fixed integer means $1, \dots, m$ and weights and standard deviations estimated from the data. The integer copy number of a segment was then determined by a Bayes classifier that used the mixture component weights as prior probabilities and required a posterior probability ≥ 0.95 for an integer copy number to be assigned.

3.2.2 Mixture model copy number classifier

The mixture model I used to fit integer copy number states to the observed copy number in the data consists of $m = 8$ normal distributions with integer means $1, \dots, m$, standard deviations $\sigma_1, \dots, \sigma_m$ and weights $w_1, \dots, w_m$. These distributions were fitted to the copy number estimates c (where c_i is the copy number estimate of the i-th segment) which I calculated from the segmented $\log_2$-ratios s (where s_i corresponds to the $\log_2$-ratio of the i-th segment in the segmentation). I defined c_i as a function of s_i ($c_i = 2 \cdot 2^{s_i}$) and the mixture model was then given as:

$$X \sim \Sigma_{j=1}^{m} w_j \cdot N(j, \sigma_j)$$

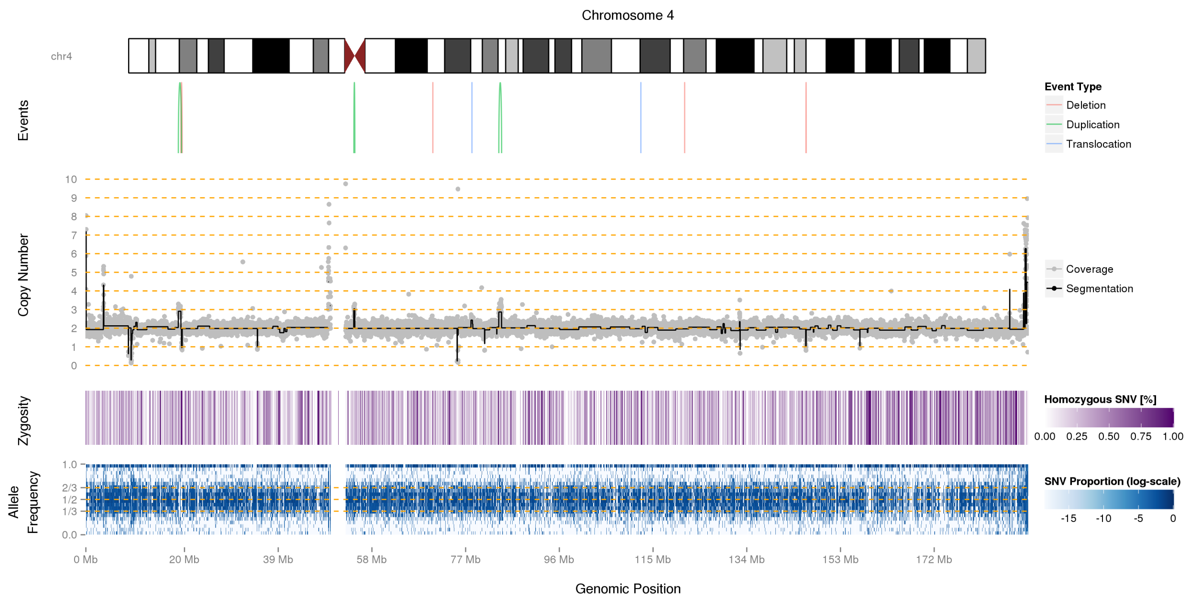


Figure 3.2: Overview plot of chromosome 4 of the HeLa Kyoto cell line. Panels are from top to bottom: Ideogram with cytogenic bands; Large structural variants; 10-kb binned coverage track normalised to CN scale (segmentation in black; integer copy number shown as yellow dashed lines); Homozygousity track, dark purple areas indicate loss of heterozygousity (LoH); SNV allele frequency distribution plot, darker areas contain more SNVs. Note that along the whole chromosome the SNVs are mostly observed at AFs around 50% and 100% indicating a diploid heterozygous region.

and each normal distribution corresponds to a copy number state with a mean that equals the integer copy number I expect for it (e.g. the second normal distribution corresponds to copy number 2 and has mean 2). The following parameters define the mixture model:

- m – number of components
- σ_j – standard deviation of the j-th mixture component
- w_j – weight of the j-th mixture component (proportion of the genome that is in the corresponding state) – note that $\sum_{j=1}^{m} w_j = 1$

I assigned each segment an integer copy number state based on the probability of observing the copy number estimate c_i in each of the m distributions by calculating the log-odds for each possible state for each segment and assigning the CN state with the highest probability.

Using $\phi_{\mu,\sigma}(x)$ as the probability density function of a normal distribution with mean μ and standard deviation σ and using k to represent integer CN states, I defined the probability of a segment with mean coverage value c_i in each of the m normal distributions in the following way:

$$Pr(c_i|k) = \phi_{k,\sigma_k}(c_i)$$

The probability of being in state k given the value c_i can be calculates using the Bayes-

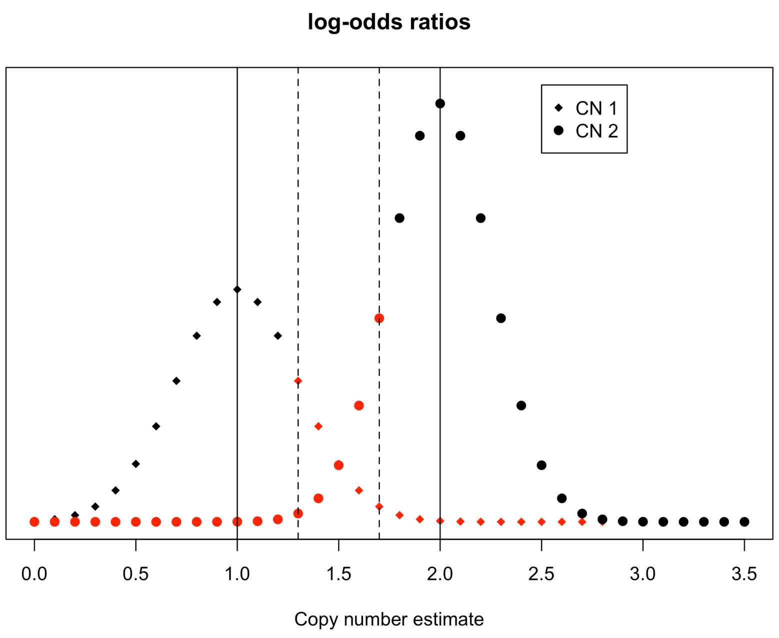


Figure 3.3: Example illustration of the log-odds based CN assignment process. In this plot I show the log-odds ratios of two distributions (centered around 1 and 2 respectively – marked by vertical solid lines). The color encodes whether a copy number estimate (x-axis) would be assigned a specific CN state (black) or not (red). There is a region of overlap between the two distributions where both are marked red and therefore datapoints falling into that region ($[1.3, 1.7]$ – marked by vertical dashed lines) can not be assigned an integer copy number value because there is not enough confidence in either of the two possible states. Segments with estimated copy number in $[1.3, 1.7]$ would be assigned the value NA.

Theorem:

$$Pr(k|c_i) = \frac{Pr(c_i|k) \cdot Pr(k)}{Pr(c_i)}$$

Here $Pr(c_i)$ can be omitted since it is constant. Using the weights w of the mixture model fit as the prior-probability, I arrived at $Pr(k|c_i) = w_k \cdot \phi_{k,\sigma_k}(c_i)$. I assessed the confidence of the highest probability call by its log-odds ratio with the sum of probabilities in all other CN states, which I used as an estimate of the probability of segment i not to be in state k:

$$\begin{aligned} Pr(\neg k|c_i) &= -Pr(k|c_i) + \Sigma_{j=1}^{m} Pr(j|c_i) \\ \text{logodds}(k|c_i) &= \log_{10}\left(\frac{Pr(k|c_i)}{Pr(\neg k|c_i)}\right) \end{aligned}$$

I accepted the highest probability CN as the state of segment i if the log-odds ratio was higher than 1.27, which corresponds to a confidence in the call of at least 0.95; if the log-odds were lower than this cut-off I assigned `NA` as the CN state. The process is illustrated in Figure 3.3.

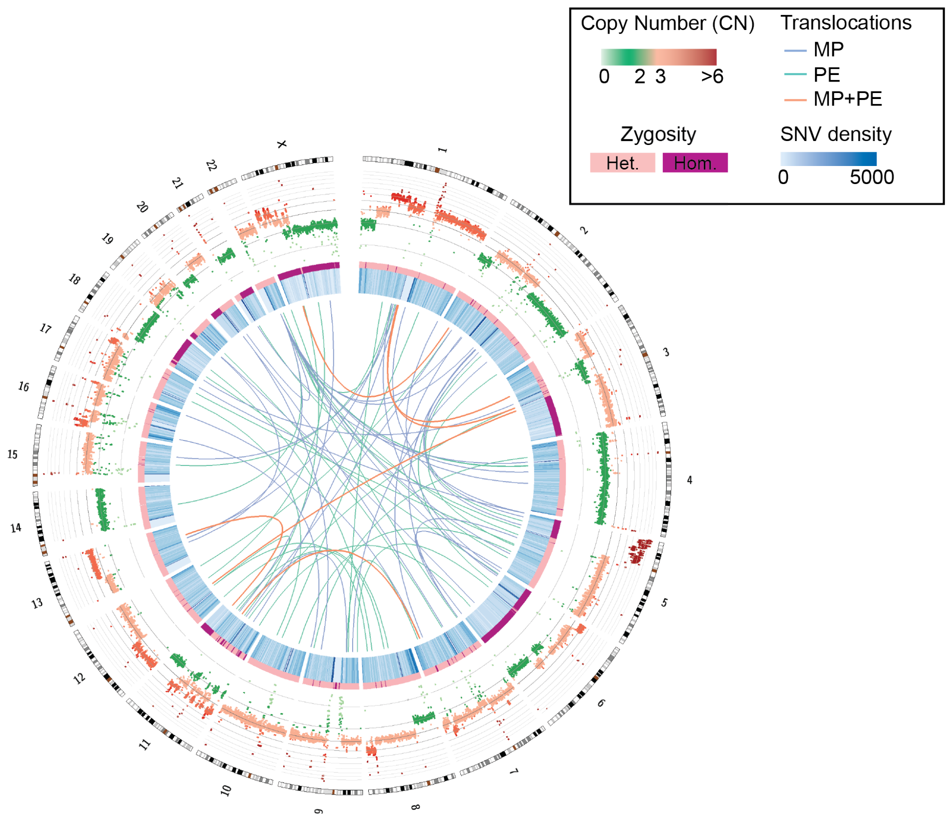


Figure 3.4: Circos (Krzywinski et al., 2009) plot showing an overview of the genome of a HeLa-Kyoto cell line. CN, SNV-density, LoH regions and structural variants are shown. Note the chromothripsis-like pattern on chromosome 11 (see Figure 3.6 for a detailed view of the region). This is the same as panel A of Figure 1 in our paper – Landry et al. (2013)

3.2.3 Finding virus integration sites

Since the HeLa cell line is based on an human papillomavirus (HPV)-induced ovarian cancer (Macville et al., 1999) I mapped potential virus integration sites in the HeLa genome by aligning the reads of our HeLa library against a composite genome containing the chromsomes 1 to 22, X, Y and MT of the human reference (GRCh37) as well as a set of known virus genomes obtained from the NCBI Entrez viral genomes project[2] (Bao et al. (2004)). The algorithm used to find the virus integration sites is described in Section 5.1.2 since I used the same software that I developed for finding vector integration sites in gene therapy samples. The resulting list of viral integration sites is shown in Table 3.1.

3.3 Results

3.3.1 Genomic Landscape

We used short-tandem repeat genotyping to confirm the identity of the analysed cell line, which had a correspondence of more than 80% of the tested markers. This confirmed the

[2] https://www.ncbi.nlm.nih.gov/genomes/GenomesHome.cgi?taxid=10239

Chromosome	Start	End	Strand	Read support	Viral species
1	10002	10118	+	12	Human herpesvirus 6B Equid herpesvirus 2 Human herpesvirus 6A Human herpesvirus 7 Gallid herpesvirus 2 Gallid herpesvirus 3 Meleagrid herpesvirus 1 Ovine herpesvirus 2 Cyprinid herpesvirus 3 Saimiriine herpesvirus 1
8	128189764	128190032	-	552	Human papillomavirus 18 and 32 (Alphapapillomavirus 7 and 1)
8	128192272	128192499	+	23	Human papillomavirus 18 (Alphapapillomavirus 7)
8	128193167	128193435	+	174	
8	128200419	128200690	+	163	
12	49659073	49659136	-	6	Human herpesvirs 5
12	95467	95567	+	9	Human herpesvirus 6B Human herpesvirus 6A Human herpesvirus 7 Gallid herpesvirus 2 Gallid herpesvirus 3 Meleagrid herpesvirus 1 Cyprinid herpesvirus 3
13	19648667	19648780	+	7	Taterapox virus
13	19649043	19649209	-	8	
X	155185203	155185362	-	8	Human herpesvirus 6B Equid herpesvirus 2 Human herpesvirus 6A Human herpesvirus 7 Gallid herpesvirus 2 Gallid herpesvirus 3 Meleagrid herpesvirus 1 Ovine herpesvirus 2

Table 3.1: Table showing an overview of potential virus integration sites that were extracted from the read alignment to a compound genome of human and virus references.

identity of the cell line as HeLa and we went on to characterise the genome through read alignment and subsequent analysis. Of the ~ 1 billion reads of length 101 nt that were produced in the DNA sequencing assay 86% were successfully aligned to the human reference genome (GRCh37). Our characterisation of the genomic landscape of the HeLa-Kyoto cell line was focussed on defining tracks for CN, SNVs, InDels as well as large structural aberrations (Figure 3.4). These data tracks were intended to provide the research community with tools to improve their experimental designs and interpretations of results from studies using HeLa-Kyoto cells. We demonstrated the further interpretation of the data by defining regions of likely chromothripsis as well as regions of likely LoH which represent important facts that can inform the design and interpretation of experiments. I will give details on some of those genomic characterisations in the following sections.

3.3.1.1 Copy Number

The copy number track was defined in 10 kilobase bins along the genome. It revealed a mostly triploid genome exhibiting a topology that is markedly different from the normal state of a human genome (i.e. diploid autosomes plus sex chromosomes). The distribution of copy number calls is shown in Figure 3.5. Additionally, the pattern of copy number calls contained regions that exhibit patterns commonly associated with catastrophic chromosome shattering events (so called Chromothripsis events – Stephens et al. (2011)) on chromosome 11, a phenomenon that had previously not been described in HeLa cells. More specifically, we observed a pattern of many copy number changes between few copy number states concentrated in a relatively small region on chromosome 11 (illustrated in Figure 3.6), which is indicative of a Chromothripsis event. In total we identified potential Chromothripsis regions on chromosomes 5, 11, 19 and X, with 11 being the most obvious candiate.

3.3.1.2 Variant Calls

We produced calls of SNVs and InDels as well as large structural aberrations. In total we called 1750535 SNVs and 18411 InDels as homozygous in HeLa. 97.3% of the SNV calls and 95.3% of the InDel calls had been previously reported in The Single Nucleotide Polymorphism Database (dbSNP) (release 137). A similar overlap was observed with the data from the 1000 Genomes Project (Abecasis et al. (2012)) at 96.9% and 82.6%, respectively (see also Figure 3.7). For the purpose of generating a HeLa reference sequence we focused on high-confidence variant calls since we wanted to have good confidence that those will be present in all cells and are therefore likely to substantially influence genomic assays overlapping their positions. A total of 53121 previously unreported variant calls remained, which were either specific to the donor from whose cervical cancer the HeLa cell line is derived, somatic mutations of the tumour, or they were acquired during the transformation and propagation of the cell line in culture. Since there are no samples from the original cancerous and healthy tissue available we could not distinguish those variants further into the three possible classes of origin.

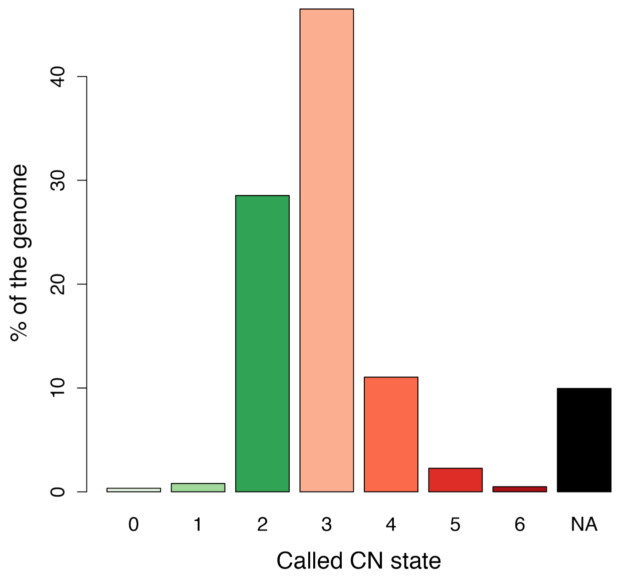


Figure 3.5: Histogram of the distribution of copy number calls amongst the 10-kb bins along the genome of a HeLa-Kyoto cell line. Colour encodes copy number and matches the color coding used in Figure 3.4. Note that the majority of bins are called in the triploid state.

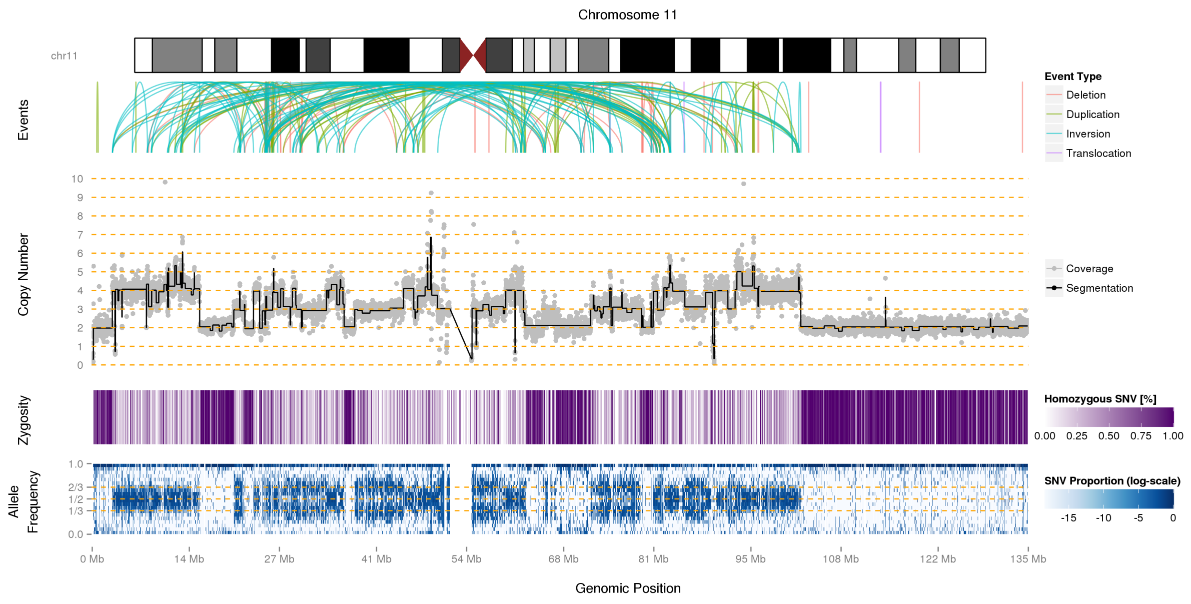


Figure 3.6: Overview plot of chromosome 11 of the HeLa Kyoto cell line. See Figure 3.2 for explanations of the tracks shown. The copy number profile shows patterns typical for chromothripsis events, i.e. many changes of copy number between relatively few states (CN 2, 3, and 4 in this case).

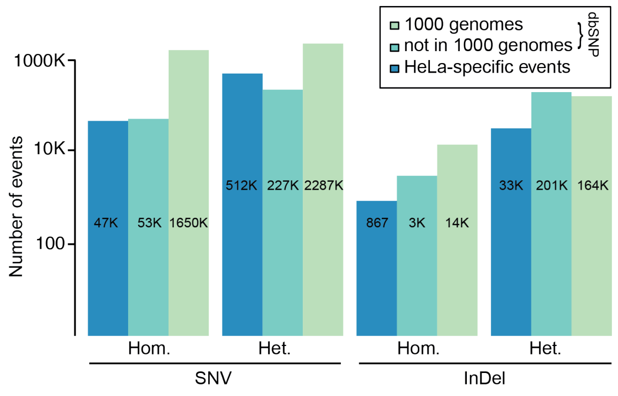


Figure 3.7: Overview of the distribution of variant calls (SNVs and small InDels) in HeLa. Stratified by overlap with previously reported variants in dbSNP and the 1000 Genomes Project data.

In addition to those homozygous variants we called a substantial amount of heterozygous SNVs and InDels as well. A total of 3026053 heterozygous SNVs and 397969 heterozygous InDels were called with high overlaps with dbSNP and the 1000 Genomes Project data. There were 83.1% of heterozygous SNV and 91.8% of heterozygous InDel calls in dbSNP and among those 91% of SNVs and 44.9% of the InDels were also present in the 1000 Genomes Project data.

The set of HeLa-specific SNVs which were selected to be integrated as part of the HeLa genome was subjected to an additional quality control step to reduce the number of false-positives. For this purpose I performed an analysis of the mutation signatures (see Alexandrov et al. (2013)) stratified by the local coverage of the regions. After deciding on the subset of variant calls with local coverage between 10 and 60 as the high-confidence set we had about 60% (336006 out of 559384) of HeLa specific SNV calls left. Note that this number includes both homozygous as well as heterozygous SNV calls.

3.3.1.3 Structural Variant Calling – Large Structural Variants (> 50 bp)

We called large structural variants (SVs) (more than 50 base-pairs in length) using the two tools DELLY (Rausch et al. (2012)) and PINDEL (Ye et al. (2009)). Both tools use aberrantly mapping read pairs and split reads to detect variants and we used DELLY to call deletions, inversion, tandem duplications and translocations and PINDEL to call inversions and tandem duplications. Two sequencing libraries were used: one paired-end library with a target insert size of 300 bp and one mate-pair library with a target insert size of 4 kb. Both libraries produced pairs of reads from the ends of a fragment of DNA and differ in the average size of sequenced fragment. In this way the paired-end library with target insert size 300 bp can be used to detect structural variants as small as 100 bp and the mate-pair library can be used to detect larger events ($>$ 1 kb) because of the larger target insert size. Of the 2891 breakpoint calls for SVs a total of 2277 had support by split-read alignments (a read spanning the breakpoint with one half of the read mapping upstream of the event and one half mapping downstream). Those 2277 SV breakpoints could therefore be described at single nucleotide resolution. Table 3.2 gives an overview of large structural variant calls obtained with DELLY

Type of SV	DELLY Calls on PE data		DELLY Calls on MP data		PINDEL Calls on PE data	
	Number of Calls	Overlap, %	Number of Calls	Overlap, %	Number of Calls	Overlap, %
Deletions	1881	14.89	234	67.09	-	-
Duplications	312	38.78	191	69.63	591	22
Inversions	33	60.61	139	56.12	101	24.75
Translocations	51	9.8	50	10	-	-

Table 3.2: Overview of the number of SV calls stratified by the type of SV (deletion, duplication, inversion and translocation) as well as the two methods (DELLY and PINDEL) and types of DNA sequencing library (paired-end (PE) and mate-pair (MP)). The column *Overlap, %* is calculated as the number of calls that overlap at least one call from one of the other categories, e.g. 14.89% of the 1881 deletions called with DELLY on the paired-end library overlap with at least one deletion call from DELLY on mate-pair data or from PINDEL on paired-end data.

and PINDEL in the paired-end and mate-pair libraries.

3.3.1.4 Loss of heterozygousity

I created a classifier from the allele frequencies observed in our SNV calls to identify mostly homozygous regions and applied it to 100-kb bins along the genome. The classifier was based on a simple cut-off on the proportion of homozygous SNVs (number of homozygous SNVs divided by total number of SNVs) in each bin and the cut-off was informed by the overall distribution of the proportion of homozygous SNVs over all bins. I applied the classifier to the autosomes and the X chromosome (since the tissue that HeLa originates from stems from a female patient), which allowed us to pinpoint regions that were likely to have undergone LoH, which means that all copies of a region carry the same allele. In this way I classified 23% of the genome as homozygous (see also the homozygousity tracks in Figures 3.2, 3.6 and 3.4). The majority of homozygous regions were present at copy numbers larger than 1 (less than 1% of the HeLa genome was called at CN 1), which indicates LoH that could have possibly arisen from the deletion of one allele and subsequent amplification of the remaining copy. For comparison I performed the same analysis on individuals from the HapMap[3] project, which did not reveal any homozygous regions larger than 100 kb in their autosomes. Figure 3.6 shows a large LoH region near the end of chromosome 11 where the copy number is 2. Many of the homozygous segments in the HeLa genome correspond to regions previously reported as LoH hotspots in cervical cancer, i.e. chromosomes 3p, 6p, 11q, and 18q (see Rader et al. (1998), Mullokandov et al. (1996) and Mitra et al. (1994)), which suggests that these LoH events might have arisen in the cervical cancer prior to the cultivation of the HeLa cell line. They could in fact be events that characterised the tumour genome itself.

3.3.1.5 Building a HeLa genome draft

To generate a first draft of a HeLa genome sequence I selected the set of SNVs, InDels and large deletions and insertions which were called homozygous and "applied" them to the

[3]http://hapmap.ncbi.nlm.nih.gov

Variant Type	Number of homozygous calls
SNVs	$1,733,577$
Large Deletions (>50 bp)	748
Short Deletions	$15,034$
Short Insertions	3446

Table 3.3: Overview of the number of homozygous variant calls used in the HeLa genome draft.

human reference genome that the alignments upon which the calls were based were made against (GRCh37). An overview of the number of events integrated into the HeLa genome draft can be found in Table 3.3. The overall chromosome structure was kept identical to the human reference genome (22 autosomes plus sex chromosomes) and we encoded copy number aberrations in a separate file. Both from the DNA-Seq data and mFISH experiments it was clear that HeLa contains compound chromosomes which might merit a separate nomenclature in which they would become stand-alone contigs. We decided to stick with the "traditional" chromosome layout both for ease of usage and because we did not have sufficient data to build a HeLa genome topology *de novo* (i.e. running an assembly software like, e.g. Velvet – Zerbino and Birney (2008)). I implemented the tool to create a HeLa genome sequence draft from an input FASTA file and a set of variant calls as well as a tools to translate genomic coordinate between the two genomes (since we integrated deletions and insertions there are shifts and loci have different coordinates in the two genomes). Both tools were implemented in Python with the functionality provided by HTSeq (described in Section 2), which facilitated the rapid development of those scripts.

3.3.2 Transcriptomic Landscape

The gene expression profile of the HeLa-Kyoto cell-line was characterised based on the sequencing of polyadenylated RNAs. This RNASeq experiment yielded ~ 450 million reads of lengths 76 and 105 nt (two sequencing runs were performed). Of those reads 56% (~ 253 million) were aligned to the HeLa genome sequence draft (see previous section). We analysed the dependency between copy number and expression level (see Figure 4 of Landry et al. (2013)) and compared the transcriptomic data with reference data from human tissues and cell lines (Illumina Human BodyMap 2.0 – Schroth (2011); ENCODE – Rosenbloom et al. (2012)).

3.3.3 Data Availability

The National Institutes of Health (NIH) mediated discussions with the family of the patient whose tumour sample eventually became the HeLa cell line (Henrietta Lacks), and a conclusion was reached, to make our data available through dbGAP in an application-based restricted access model[4].

[4]http://www.ncbi.nlm.nih.gov/projects/gap/cgi-bin/study.cgi?study_id=phs000640.v1.p1

3.4 Discussion

3.4.1 Challenges in variant calling

Dealing with a strikingly aberrant genome such as the one of our HeLa cell line poses challenges in the application of tools that are primarily designed to work well on "normal" diploid genomes. The tools I used were gsnap (Wu and Nacu (2010)) for alignment and samtools (Li et al. (2009)), Picard[5] and the Genome Analysis Toolkit (GATK) (McKenna et al., 2010) as well as PINDEL (Ye et al. (2009)) for postprocessing and variant calling. When I processed the DNA-Seq data from the HeLa sequencing project I followed the best practices at the time, which was alignment followed by a standard GATK workflow of re-aligning around indels, recalibrating base-call quality scores and removing probable PCR (Saiki et al., 1988) duplicates. The final step was calling SNVs and InDels with the `UnifiedGenotyper` walker of the GATK. It is important to note that the `UnifiedGenotyper` is a tool primarily designed for calling variants in diploid genomes. The fact that the HeLa-Kyoto genome is so markedly different from diploid, with large triploid stretches and even higher copy number in some regions, is likely to have caused false-negatives in high ploidy regions, e.g. a variant that is present in 1/5th of the reads in a 5-ploid region looks unlikely to a variant caller that assumes diploidy and therefore models variants as a two-state model (heterozygous and homozygous). Based on the assumption that the HeLa cell line we used is a genetically stable homogeneous population we could assume that the observed allelic frequencies of variant calls represent the ploidy state of the variant (e.g. an allelic frequency of 0.6 would correspond to a location with copy number 5 and 3 out of 5 copies carrying the allele). This approach was likely too optimistic, since HeLa cells are cancer cells and there is no guarantee that sub-populations will not arise (be that by internal genomic instability or external influence in culture).

3.4.2 Conclusion

While the HeLa cell line has been widely used in research ever since it was established in 1951, the substantial amount of studies involving HeLa cells that were conducted prior to our publication were executed with limited knowledge of the genomic properties of the cell line. There were efforts to describe the HeLa genome on a lower resolution, e.g. Macville et al. (1999), but prior to our work no single-nucleotide resolution characterisation of the HeLa genome existed. Our publication provided the first detailed characterisation of the genomic landscape of a HeLa cell line with respect to differences to the human reference genome. We provided the research community with data tracks describing SNVs, InDels, large SVs, CN, LoH as well as expression levels for genes. We furthermore integrated a high-confidence set of homozygous SNVs, InDels and large deletions and insertions into the human reference genome that the calls were made against, thereby creating a first draft for a genome sequence of the

[5] `http://picard.sourceforge.net`

HeLa-Kyoto cell line that we used. Not only did we provide a useful resource to the community, which can help to improve experimental designs and interpretation of results (e.g. in RNA interference screens), but we also described an approach for the characterisation of model organisms (including tools to implement it) that can be easily transferred to other cell lines that are in use as a model system.

One of the key findings from combining the copy number data with expression level estimates is that there is little to no dosage compensation as the expression levels can be shown to correlate with copy number. Given the strikingly aberrant copy number profile of the HeLa cell line we analysed, this could lead to misregulation of protein complexes where the stochiometry of the different parts that constitute the complex is modified by copy numbers changes overlapping the genes encoding some of the parts.

In the light of the relatively recent initial description of the phenomenon known as Chromothripsis (Stephens et al., 2011) it is interesting to note that we could identify regions in the HeLa genome that exhibit the hallmarks of such a catastrophic chromosome shattering event (many CN changes between few CN states). Chromothripsis has since been described in a multitude of different cancer types and has been estimated to be present within the genomes of $2-3\%$ of all cancers. In HeLa it is possible that the original tumour was generated through a Chromothripsis event, but it is also possible that the Chromothripsis occurred sometime after the cultivation of the cell line. In the same way in which we cannot distinguish between HeLa private SNVs stemming from the patient, the tumour or processes during the culture of the cell line for lack of data, we also cannot discern when the chromothripsis event occurred.

We used shotgun sequencing-based assays for data generation and attained a moderate sequencing depth (less than 50x). The usage of this type of data imposes specific limitation on the possible analyses that can be performed. Based on the coverage we obtained with the 101nt paired-end reads, single nucleotide polymorphism (SNP) phasing by common read pairs (i.e. phasing adjacent SNPs based on whether they are supported by the same / overlapping read pairs) was infeasible. Furthermore repetitive regions pose problems if the reads cannot be uniquely aligned to those regions. This can be alleviated by moving to a sequencing technology that produces significantly longer reads (e.g. 10 kilobase reads produced by a PacBio[6] platform), which could eliminate most problems that stem from non-uniquely mappable repetitive regions. A move towards single cell or even single chromosome sequencing as has been proposed to be implemented on a microfluidics platform (Fan et al., 2011) would be a tremendous improvement as well, since that would allow for a very precise characterisation of the topology of the genome (i.e. which sequence is where) up to a point where a reliance on the default chromosome layout of the reference (as we used) would not be needed anymore.

In summary, our approach generated a very detailed and sometimes surprising picture of the genomic landscape of the HeLa cell line that we investigated. Our analyses were published including source codes and tools to repeat essential steps and we expect that similar analyses

[6]http://www.pacificbiosciences.com/

will be performed by researchers working on other cell lines as well. This kind of genomic characterisation can be especially useful in genomically unstable cell lines, i.e. cancer-derived cell lines like HeLa, in which case a regular check-up might be a good idea to keep track of eventual changes in the genomic landscape of those cell-lines and adapt the experimental designs accordingly. We envisage that approaches similar to ours might be useful to ensure the integrity of cell lines and the quality of the biological insights derived from them.

Our study served to emphasise the importance of accounting for the genomic characteristics of model cell lines when designing and interpreting experiments. Certainly the genome of HeLa cells is amongst the more aberrant ones amongst cell lines in use as model organisms, nevertheless it seems a good practice for any researcher to make sure that the genomic peculiarities of the cell line used are taken into account when analysing their results.

Shortly after our publication another publication characterising the haplotype-resolved genome and epi-genome of the HeLa CLL-2 strain was published by Adey et al.. Their genomic characterisation also revealed a mostly triploid genome with a large number of aberrant regions including LoH regions. Adey et al. also detected the HPV integration on chromosome 8 and characterised the affected region with a very detailed structural model. In the analysis of transcriptomic data Adey et al. find the same lack of dosage compensation that we also reported, which is indicated by a clear correlation between expression levels and copy number state. The authors do not mention detecting patterns indicative of chromothripsis in their data.

4 Using HDF5-based nucleotide tallies with 'h5vc'

The challenges researchers face with the advent of massive sequencing efforts aimed at sequencing RNA and DNA of thousands of samples (e.g. in the context of cancer research: the TCGA project[1]) will need to be addressed now, before the flood of data becomes a real problem. The effects of the infeasibility of handling the sequencing data of large cohorts with the current standards (BAM, VCF, BCF, GTF, etc. – Danecek et al. (2011); Li et al. (2009)) have become apparent in recent publications that performed population level analyses of mutations in many cancer samples and work exclusively on the level of preprocessed variant calls stored in VCF/MAF files simply because there is no way to look at the data itself with reasonable resource usage. This can be seen for example in Kandoth et al. (2013). This challenge can be adressed by augmenting the available legacy formats that are typically used for sequencing analyses (SAM/BAM files) with an intermediate file format that stores only the most essential information and provides efficient access to the cohort level data whilst reducing the information loss relative to the raw alignments. This file format will store nucleotide tallies rather than alignments and allow for easy and efficient real-time random access to the data of a whole cohort of samples.

In this section I will introduce the **`h5vc`** `R/Bioconductor` package. I designed **`h5vc`** as a tool to provide researchers with a more intuitive and effective way of interacting with data from large cohorts of samples that have been sequenced with current generation sequencing technologies. It is available for download as part of the `Bioconductor` package repository and I published an accompanying application note in *Bioinformatics* earlier this year (Pyl et al., 2014). I have given a lecture and accompanying workshop introducing the package at the CSAMA 2014 conference[2]. Currently **`h5vc`** is being downloaded around ~ 200 times every month, and an overview of the download statistics can be found in Table 4.5.

4.1 Nucleotide tallies with an HDF5 back-end

The fundamental concept that **`h5vc`** is based on, is to use nucleotide tallies as an intermediate data format between raw alignments stored in BAM files and filtered variant calls stored in VCF files. A nucleotide tally is a table of nucleotide-resolution summary statistics (e.g. counts of mismatches) at genomic positions augmented by additional dimensions like strand and sample. In my current implementation I focus on the problem of calling SNVs as well as small deletions, therefore I tally the mismatch counts, coverages and deletion counts per nucleotide in the datasets described in Table 4.1. The mismatch count dataset is essentially

[1] `http://cancergenome.nih.gov`
[2] `http://www-huber.embl.de/csama/index.html`

a table containing the number of overlapping reads supporting a mismatch for each combination of nucleotide, genomic position, strand and sample in a cohort. In this way all mismatches from aligned reads in a cohort of samples can be expressed in a 4-dimensional array of nucleotide x sample x strand x position (Table 4.1). Additional data tables contain information about the coverage (total number of overlapping reads per position, strand and sample) and number of deleted bases per position. Possible extension of this concept include the storage of the number of reads supporting an insertion following each position or some type of phasing-related data, e.g. the mutual information content of one position and its immediate neighbours with respect to the probability of observing mismatches within read alignments that overlap both positions.

In my implementation, available in the **`h5vc`** package, I use the HDF5 file format (The HDF Group, 2010) as a back-end to facilitate data storage and access of nucleotide tallies. HDF5 is a fast and mature file format that has already seen production use in fields as diverse as NASA space missions and big movie special effects (e.g. The Lord Of The Rings[3]). The fact that this file format has been used in production environments where a lot of resources and money rely on its faultless functionality indicates that this is indeed a reliable format that can be easily adapted in the research community without fear of catastrophic failure. HDF5 is maintained by the HDFGroup (The HDF Group, 2010), which is a non-profit organisation that provides updates and fixes and maintains documentation about the format. This adresses another essential requirement for a file format to find usage with a broad audience, which is continued and reliable support that is secured for the foreseeable future.

The HDF5 data format and library are designed specifically for the efficient storage of large numerical datasets, while allowing for fast and efficient access and providing transparent compression support, i.e. one can activate compression in HDF5 without having to change the way that scripts interact with the HDF5 files. HDF5 is available on many platforms in the form of libraries for different programming languages including C/C++, Java, Python, Matlab and R. My implementation relies on the **`rhdf5`** `R/Bioconductor` package (Fischer and Pau, 2012) for low-level access functions to HDF5 files. Mismatch tallies are stored in a dataset called **Counts** and further quantities are stored in the datasets **Coverages** (total number of overlapping reads per position, sample and strand), **Deletions** (number of reads supporting a deletion event for a base at a given position, sample and strand) and **Reference** (an integer encoding of the reference sequence, storing for each genomic position the code corresponding to the nucleotide as defined in the reference sequence that the alignments were made against). These four datasets represent a complete tally and I store a set of 4 datasets for each chromosome in the reference genome within a tally file. All datasets are essentially large arrays of integers (i.e. numerical data, which HDF5 is especially well equipped to handle) and are defined in Table 4.1.

Within an HDF5 file the data are stored in a hierarchical structure consisting of groups

[3] `http://www.hdfgroup.org/about/history.html`

Dataset	Definition	Dimensions
Counts	observed mismatches relative to the reference	`[bases x samples x strands x positions]`
Coverages	number of overlapping reads	`[samples x strands x positions]`
Deletions	observed deletions of bases	`[samples x strands x positions]`
Reference	vector containing the reference bases	`[positions]`

Table 4.1: Overview of the datasets present in an HDF5 tally file giving definitions of their content and a representation of the dimensions of the datasets.

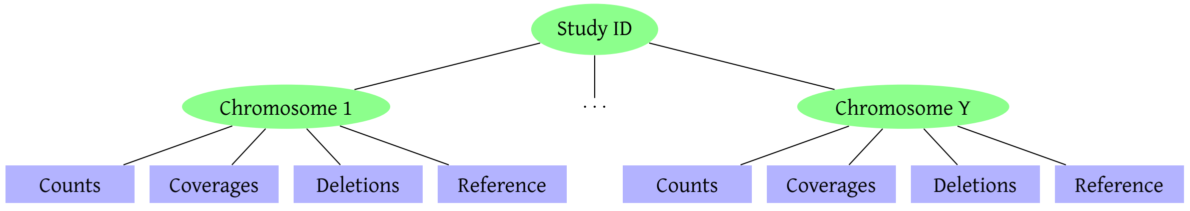


Figure 4.1: Overview of the internal structure of a tally HDF5 file showing the tree structure with groups representing studies and chromosomes (contigs) as internal nodes (visualised as green elliptical nodes) and the 4 datasets defined in Table 4.1 as leafs (visualised as light blue rectangular nodes).

and datasets. This layout is analogous to a file system where groups represent folders and datasets represent files. Within an HDF5 file the location of an object is represented with a path notation similar to what a normal file system uses, e.g. `/Group/Subgroup/Dataset`. In my implementation I use groups to represent the organisatorial units *cohort* and *chromosome*, which translates to a layout in which a root folder (group) represents a cohort or experiment and contains subfolders (sub-group) for each of the chromosomes in the reference, which in turn contain the 4 datasets per chromosome as described above (Table 4.1 and Figure 4.1). In the filesystem analogy the **Counts** dataset of e.g. chromosome *chr7* of a cohort named *ExampleCohort* will be stored at the location `/ExampleCohort/chr7/Counts` within the HDF5 file.

4.1.1 Sample Meta Data

Since HDF5 files are highly optimised towards storing large numerical data-sets we use multi-dimensional arrays to store the tally data in them. The nucleotide tally representation of a cohort of samples must also include some sample meta-data, which describes the sample IDs, which patient IDs they are associated with, whether they are case or control samples and any other important information that characterises the data-set. One particularly important piece of information is the location of samples in the sample dimension of the data-sets, i.e. which index in that dimension corresponds to which sample. The arrays of numerical data stored in HDF5 files do not provide a formalised way of attaching names to the dimensions (e.g. the actual genomic positions and the names of the samples, bases or strands). For the case of the genomic position we avoid having to store the corresponding genomic position for each entry in the data-set by having the index itself represent the position, which is also the reason why the tally is stored per chromosome and not as one continuous block of e.g. ~ 3

billion values for a human sample. The indexes of the nucleotides (“A”, “C”, “G” and “T”) and strands (“+” and “−”) are well defined and always the same in all nucleotide tally data-sets. The mapping of samples to their respective indices in the sample dimension of the datasets on the other hand, is by its very nature specific to a given cohort and not fixed.

To facilitate this mapping and at the same time allow for the storage of other important data describing the cohort on a per-sample level I implemented a sample meta-data framework that stores a table of sample meta-information alongside each group within an HDF5 tally file. The storage of meta-information can be implemented using either HDF5 attributes, which are a part of the HDF5 specification and serve the purpose of annotating groups and data-sets with additional information, or a separate set of data-sets. I implemented an interface that can read the sample meta-data from both the HDF5 attributes as well as a separate set of data-sets. The default behaviour is now to use separate data-sets (which was not the case in previous versions of the **`h5vc`** package) since HDF5 attributes have a size-limit of 64 kilobytes. This limit is usually not reached in cohorts with less than 100 samples, i.e. the cohorts I worked on most of the time – Section 5. To be safe for future applications which are likely to involve larger cohorts I implemented a version without size limitations. The sample meta-data itself is simply a table containing one row per sample in the cohort. The following 4 columns have to be defined and any amount of additional information can be added in further columns:

Sample	A unique sample identifier
Patient	A unique patient identifier (multiple samples can be from the same patient)
Column	The index of the sample in the sample dimension of data-sets
Type	The type of the sample (e.g. “Case” or “Control”)

An example of what the sample meta-data looks like for one of the cohorts I have been working on is shown in Table 4.2 (see also Table 5.1). This table contains multiple samples per patient, some of them re-sequencings of previous runs of unsatisfactory quality as well as relapse and pre-leukaemia samples. Apart from the 4 fixed columns that every sample meta-data table must contain, one may add any number of additional columns containing for example descriptions of the samples or patients (e.g. age, sex, time of the sample extraction) or other relevant clinical variables that can be measured on a per-sample level. The combination of nucleotide tallies stored as multi-dimensional numerical data-sets and sample meta-data within the same HDF5 file allows the researcher to perform their analyses without the need for external sources of information with regard to the cohort itself. Of course there are a lot of useful external resources that can feed into an analysis, e.g. gene models or annotations of repetitive regions to only name a few.

Sample	Patient	Column	Type
WAS1.AML	WAS1	1	Case
WAS1.PreGT	WAS1	2	Control
WAS8.AML	WAS8	3	Case
WAS8.AMLRelapse	WAS8	4	Case
WAS8.PreGT	WAS8	5	Control
WAS9.AML	WAS9	6	Case
WAS9.PreGT	WAS9	7	Control
WAS1PrimaryDNAFirst	WAS1First	8	Case
WAS1ControlDNAFirst	WAS1First	9	Control
WAS1PrimaryDNASecond	WAS1Second	10	Case
WAS1ControlDNASecond	WAS1Second	11	Control
WAS5PrimaryDNA	WAS5	12	Case
WAS5RelapseDNA	WAS5	13	Case
WAS5ControlDNA	WAS5	14	Control
WAS6PrimaryDNA	WAS6	15	Case
WAS6ControlDNA	WAS6	16	Control
WAS7PrimaryDNA	WAS7	17	Case
WAS7ControlDNA	WAS7	18	Control
WAS7PrimaryDNA_ReSeq	WAS7_ReSeq	19	Case
WAS7ControlDNA_ReSeq	WAS7_ReSeq	20	Control
WAS8PreLeukemiaDNA	WAS8	21	Case
WAS8PrimaryDNA	WAS8	22	Case
WAS8ControlDNA	WAS8	23	Control
WAS9PrimaryDNA	WAS9	24	Case
WAS9ControlDNA	WAS9	25	Control

Table 4.2: Example of what sample meta-data is stored in an HDF5 tally file. This data describes the tallies for two cohorts of paired cancer sample that I have been working on (Section 5).

4.2 Creating tally files

HDF5-based nucleotide tally files provide an intermediate level of access to high throughput sequencing data that is more abstract than read alignments stored in BAM files but at the same time more “raw” than a set of variant calls in a VCF file would be. Essentially the creation of a nucleotide tally file in HDF5 format is an additional post-processing step that should be applied to the raw sequencing alignment data. It is important to note that nucleotide tallies offer a lossy compression of read alignment data and they are not a replacement for BAM files. The transformation of read alignments into tables of nucleotide-level summary statistics removes any information about the relationship between observations at neighbouring positions and the tallies of position i are independent of the tallies of position $i+1$. The advantage of this is that nucleotide tallies can be created and analysed in parallel since no dependency between neighbouring positions exist. One negative aspect is the loss of phasing information, i.e. we can make no assumptions about whether two SNV calls are coming from the same or different alleles.

Creating nucleotide tally files is a computationally intensive task and for larger cohorts of e.g. human samples it requires substantial amounts of compute resources. The time and resources used to create tally files can be seen as an investment which will pay off through the way in which they allow the user to interact with the data afterwards. The actual amount of time that is needed depends on the size of the cohort and whether the data is from a WES or WGS approach as well as the size and power of the computers used for the tally creation, e.g. the creation of nucleotide tallies of expressed regions for 188 human WES samples in a highly parallelised fashion takes ~ 2 days to complete. Since nucleotide tallies are storing only position-specific data and do not contain information about the relation between neighbouring positions their creation can be parallelised. An obvious level of parallelisation is the creation of tallies per chromosome and it is advisable to create them in blocks along the chromosome (e.g. of size 1 mega-base). Here we have to make a trade-off between speed and file-size, since HDF5 files can only be written to in parallel if they are not compressed. Since the compression works in blocks we need to be sure about the content of the block at the time of compression and with multiple processes writing to the same file at the same time this is not a guarantee that can be made. Given this limitation of the technology we can choose among two possible approaches in order to arrive at a highly compressed nucleotide tally file (if we do not care about hard-disk usage we can simply switch off compression and write in parallel):

1. Create the tally file uncompressed in parallel using a large amount of disk space and compress it afterwards

2. Create the tallies in parallel in blocks along the genome and write the resulting blocks into the tally file serially (e.g. one after the other)

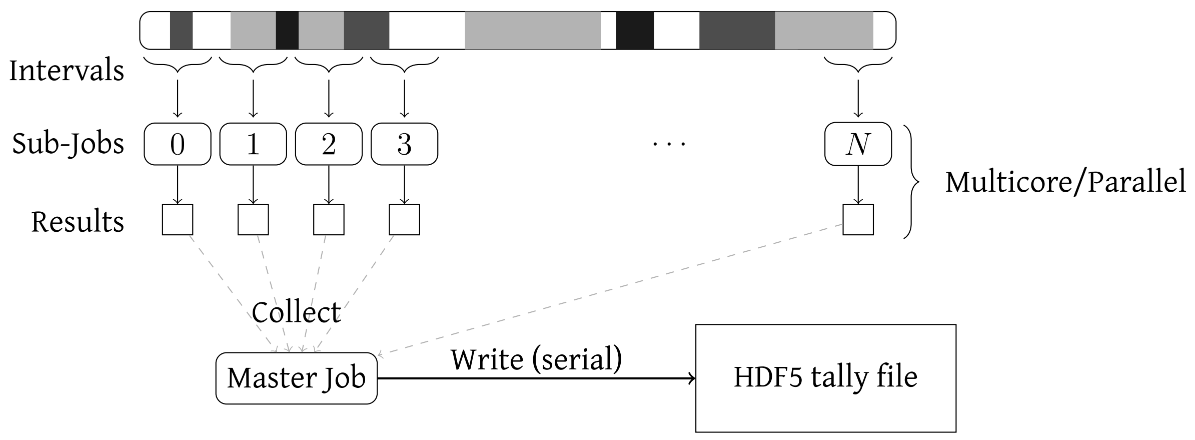


Figure 4.2: Overview of the process layout of partially parallelised nucleotide tally creation. The chromosome is split into intervals (top) which are each assigned to one sub-job of the master job for processing. The sub-jobs can run in parallel on a multicore machine or on a compute cluster composed of multiple machines. The master job waits for sub-jobs to finish and writes the results of finished sub-jobs to the HDF5 tally file. This writing happens serially, since the usage of compression prohibits parallel writing into HDF5 files.

In my implementation I have decided to use the second option and **`h5vc`** contains functionality to create a nucleotide tally file from a set of input BAM files in this fashion. Essentially a master-process is running, which creates sub-processes that each handle the creation of a nucleotide tally of the provided BAM files in a given genomic region (e.g. a bin of size 1 mega-base or a single exon or gene). The sub-processes run in parallel and the master-process constantly checks for sub-processes that have finished and puts any finished processes into a queue. The contents of the queue are then serially written to the nucleotide tally file in (compressed) HDF5 format. In this way we can already start writing finished blocks of data into the nucleotide tally file while other blocks are still being calculated in parallel. The usage of the **`BiocParallel`** and **`BatchJobs`** packages allows us to run the sub-processes either on a multicore machine or on a compute cluster (Morgan et al., 2013; Bischl et al., 2011). An outline of the process can be found in Figure 4.2.

4.3 Compression and Chunking in HDF5 files

4.3.1 Compression in HDF5 files

The choice for favouring serial writing with compression over parallel writing without compression in the creation of the HDF5 tally files is based on the impracticality of storing nucleotide tallies of large cohorts uncompressed. When working on the EBI compute cluster, this is actually not really a problem since disk space is abundantly available there. In a more realistic application scenario where a smaller research group with limited compute resources attempts to create such a file, this will become a problem. The increased runtime of the tally creation process when writing serial is feasible and a cohort can usually be processed within a reasonable timeframe. The exact time it takes to write depends on the size of the cohort and the type of experiment, since in a WES sample the tallies actually only need to be created and written within the coding regions. This approach reduces the amount of data to write by e.g. $\sim 97\%$ in human samples where $\sim 3\%$ of nucleotides are coding. The largest cohort I have worked with so far is a set of 188 human WES samples which can be tallied in less than a week when covering the whole genome. Tallying only genomic intervals that overlap genes (including introns) reduces this time consumption to a bit less than two days and a further reduction to covering only exons (i.e. not covering the intronic regions) is likely to speed this up even more. In this way whole exome samples are faster to analyse than whole genome samples.

The way I designed the layouts of the data sets stored in the HDF5 files implies a strong need for compression, since the genomic position is stored implicitly in the genomic position dimension. Therefore the position that a value corresponds to is not explicitly stored but rather reflected in the location of the value within the data set. The problem with this is very apparent in the case where a whole exome sample is stored as a nucleotide tally, which will contain non-trivial values for only about 3% of the genomic positions and therefore the large majority of the data sets (i.e. the 97% of positions that correspond to introns and intergenic regions) will contain zeroes. Storing all those zeroes uncompressed creates an HDF5 file that is equally as big as a corresponding tally file from a whole genome sample. Activating compression will remove the memory impact of the large blocks of zeroes. This compression effect is very similar to the effect of the run-length encoding approach used in HTSeq (Section 2.3) where one can imagine the compression of large blocks of zeroes through storing only the information of the size of the block instead of the whole block itself. A dataset containing 100 million zeroes can be stored in a compressed HDF5 dataset using less than 1 mega-byte on the disk.

4.3.2 Chunking in HDF5 files

Within HDF5 files the data sets are stored in chunks, which are the atomic unit for compression, access and caching of the data. The need for chunking arises from the fact that any representation of multi-dimensional data (e.g. 4D-arrays of nucleotide counts) on the disk will have to be serial. Let us consider for example a two-dimensional array of n rows and m columns that needs to be stored in a file on the hard disk as a serial vector of numbers. There are two natural approaches that can be taken: one can either concatenate the rows or concatenate the columns to form one long vector of length $l = m \cdot n$ and store this vector on disk. This serial storage of multi-dimensional arrays poses some challenges when we are trying to retrieve sub-sets of the data (e.g. the k-th row where $0 < k < n$). If we store the array by concatenating the rows we know that the k-th row will start at position $(k-1) \cdot m + 1$ (all preceding rows concatenated, which each contain m numbers) and the next m values correspond to the k-th row. If we however want to extract the k-th column, the corresponding values will be distributed evenly along the serial representation of the data. This difference can influence

1	2	3	$\dots$	m
$m+1$	$m+2$	$m+3$	$\dots$	$2\cdot m$
$\dots$				$\dots$
$l-m+1$	$l-m+2$	$l-m+3$	$\dots$	l

Table 4.3: This table represents a two-dimensional array of n rows and m columns stored in a row-concatenated fashion. The numbers in the cells correspond to the positions of the values within the serialised representation which is a vector of length $l = m \cdot n$. The second column and second row are marked in red. Note how the second row can be read from the vector representation as one continuous block of data (spanning the interval $[m+1, 2\cdot m]$), while the second column must be read as a series of single values spaced m positions apart within the vector representation.

the time needed to read data from the file significantly. The example is illustrated in Table 4.3.

To alleviate this effect and allow us to efficiently read from a data-set it is serialised in chunks within the HDF5 file. This means that a chunk shape is defined, which can be multi-dimensional and the data is split into chunks of that shape before each chunk separately is serialised into the HDF5 file. This will reduce the time spent seeking the correct values within the files since the effects illustrated in Table 4.3 are now localised to the chunk we want to read from, instead of affecting the whole data-set. Without such behaviour, extracting the data corresponding to a single position in the human genome could easily lead to the algorithm having to jump around the HDF5 file by as much as 3 billion positions, just to collect the correct values. If we for example chunk the *Counts* dataset (which has 4 dimensions) in chunks that span all nucleotides, samples and strands but are limited to 10000 positions along the genome, then the function for the retrieval of data will at most have to move by 10000 positions. This is to compare with moves of up to 250 million positions when we are reading from an un-chunked data-set that corresponds to chromosome 1 in a human sample. Because of this layout it can be helpful to choose a chunk size that suits the way in which one wants to access the data. When the main way of interaction with the data is in blocks in the range of hundreds of genomic positions (e.g. SNV positions and a window up- and downstream of them) then using chunks of a size in the range of mega-bases will create a lot of overhead.

At the moment **`h5vc`** implements chunking on the level of genomic position and sample, therefore the data sets are stored in chunks that correspond to a certain number of samples (the `sampleChunkSize`) and a certain number of genomic positions (the `chunkSize`; named so for backwards compatibility reasons) as well as both strands and all nucleotides (where applicable). The default behaviour is for chunks to span all samples and have a length of 50000 in the genomic position dimension. Those values are based on experience with cohorts of about 20 samples (i.e. the kind of cohorts I usually work on; see also Section 5). Since both values can be configured by the user through function parameters it should be easy to adapt them to studies of different dimensions, e.g. containing thousands of samples. The underlying rationale for choosing chunk sizes should always be informed by the way in which a user plans to interact with the data-sets in their downstream analyses. For example, a block of data containing the counts and coverages of 20 samples in a genomic region of width 50000 bases stored as an array of 32-bit integers uses less than 100 Megabytes of space in memory, making handling the data in chunks of this size easy even on low-end laptop computers. When the cohort contains e.g. 200 samples the memory usage of a block of 50000 genomic positions is already almost 1 giga-byte. In the following section I will explain the principles behind the interface to HDF5 tally files, and it will become more clear that the size of the blocks in which one interacts with the data is an important thing to consider.

4.4 Interacting with HDF5 nucleotide tallies

The concept of using nucleotide tallies is geared towards reducing the size of the data-set and making the data more easy to interact with than if one had to work on e.g. a set of alignments stored in a BAM file. The tallies themselves are large matrices of numerical data and in fact a lot of the savings in data-set size comes from the compression that the HDF5 library offers (i.e. making the compressed HDF5 file small). If one were to load the mismatch nucleotide tally representation for the whole genome of a cohort of human samples, e.g. 21 samples, into memory one would need to store a total of about $21 \cdot 2 \cdot 3 \cdot 10^9 \cdot 4 \approx 5 \cdot 10^{11}$ values in memory (21 samples times 2 strands times 3 billion positions times 4 bases). If we assume that we can represent each number by a 32-bit integer (i.e. 4 bytes per number) the total size of this dataset is somewhere in the range of 2 terabyte. Most people do not have access to machines with this much memory and therefore **`h5vc`** implements a similar streaming approach to data processing as I already introduced in HTSeq (see Section 2).

4.4.0.1 The h5dapply and h5readBlock functions

To facilitate the analysis of nucleotide tally data whilst maintaining a reasonable memory footprint I implemented two functions in **`h5vc`** (`h5dapply` and `h5readBlock`) which allow for the retrieval of data corresponding to a particular genomic region of interest and the application of a function (e.g. a variant caller) in blocks along the genome. Nucleotide tallies, while using comparatively little amounts of disk space when stored in compressed HDF5 files, can be quite large and usually the nucleotide tally of a set of human samples will not fit into a computer's RAM. This is a problem that is typical for HTS data and the reason behind the streaming approach to processing read alignments used in HTSeq, introduced in Section 2.2. In the most common use-case we want to do an analysis on a nucleotide tally that works on the genomic position level, where we want to compute some sort of summary statistic (e.g. a binned coverage) or we want to find genomic positions that have certain properties (e.g. the positions of somatic variants in cancer samples). The nucleotide tallies of neighbouring genomic positions are usually independent of each other and although some correlations might exist we cannot make any inference about the relations between nucleotide counts at neighbouring positions once the read alignments have been transformed into tables of mismatches. In other words, during the translation of read alignments into nucleotide tallies all phasing information is lost and we can make no assumptions about whether two neighbouring mismatches were caused by the same read or two different reads (i.e. whether they are from the same allele or not).

It is immediately clear that any calculation of the aforementioned type (binned summary statistics or position specific properties) can be calculated in arbitrarily chosen bins along the genome and each bin can be calculated without knowledge of the data in any other bin. This leads to the conclusion that almost any function a researcher might want to apply to a nucleotide tally can be applied in bins along the genome and for each bin only the tally data corresponding to it has to be available. In some cases it might be necessary to use overlapping bins in order to deal with positions at bin boundaries, e.g. when calculating for each position in the genome the number of mismatches observed in a window of size k centered on the position (with k being odd), one should choose bins that overlap by k positions. The `h5dapply` function implements exactly this behaviour and can be used to apply functions to tallies in a block-wise fashion (e.g. apply a coverage estimation function in 10 kilobase bins along the genome as a first step of a segmentation according to copy number). The fact that we can usually calculate any result of any bin or block independent of the result of any of the other bins / blocks allows us to process large nucleotide tallies in chunks that fit comfortably into the memory of a normal laptop. Furthermore the fact that results are independent between chunks allows for easy parallelisation of any such calculation and **h5vc** implements such behaviour in the `h5dapply` function, which can use multicore or even multi-machine setups to parallelise calculations.

The `h5dapply` function exposes an interface for applying any R function to nucleotide tally data in blocks along the genome (see Listings 3 and 4 for example code). The concept is to split the genome (or a genomic region) into bins of a user-chosen size and successively retrieving the nucleotide tally data corresponding to each bin, applying the user-provided function to it and returning the result. This process internally uses the `bplapply` function from the **BiocParallel** package, which can be switched from serial processing to multi-core or multi-machine parallel processing via a simple configuration interface. As a convenience to the user, **h5vc** provides the `h5readBlock` function, which implements essentially the same behaviour as `h5dapply` does, if there is only one block to be processed and the function to be applied is the identity (i.e. `function(x) {return(x)}` when implemented in R). It basically reads a block of data from the tally file and returns it directly. In addition to the ability to apply a function to blocks along the genome or extract specific blocks of data, a further subsetting by sample is implemented. This subsetting by sample is implemented on the level of the HDF5 library and therefore performed before the data is loaded into memory, this behaviour is crucial when dealing with particularly large cohorts, where even a block of data corresponding to a region of only 10 kilobases might be too big to fit into the computer's memory if there are e.g. 10000 samples in the cohort.

In summary, the interface that **h5vc** provides to the user through those two functions (`h5dapply` and `h5readBlock`) allows for the easy retrieval of data corresponding to arbitrary subsets of genomic positions and samples as well as the direct application of analysis functions to this data (e.g. variant calling functionality).

4.5 Implementation and Availability

I implemented the **h5vc** package using R and C/C++ and made it available as a package in the `Bioconductor` project. It is well documented both in a reference of available functions and a set of vignettes that illustrate common workflows and examples of application. The Bioconductor repository is regularly tested and well maintained and provides a reliable path for updates and time-critical bug fixes through its versioning structure. While most of the functionality is implemented as pure R code some essential parts are implemented in C/C++. All interactions with HDF5 files are facilitated by exposing the HDF5 C API to R (this was implemented by Bernd Fischer in the **rhdf5** package). I implemented the function for creating nucleotide tallies from BAM files (`tallyBAM`) in C for increased performance of this most time-consuming step in the process. I based the function definition and interface on the `bam2R` function provided by the **deepSNV** package, which was written by Moritz Gerstung (Gerstung et al., 2012).

4.6 Examples of usage

h5vc is a project that was primarily motivated by my disappointment and frustration over using the available tools at the time of its inception. As such I designed the tool with flexibility in mind and it is more of a framework for analysis of HTS data than just a monolithic tool that performs one specialised task. This design principle is apparent in the way that the interface of the `h5dapply` function is designed to allow the researcher to plug in arbitrary functions as long as they are callable from within R. This includes both pure R as well as C functions exposed through an R interface. Instead of writing a package that calculates coverages or SNV positions I wrote a package that enables multiple researchers to rapidly prototype ideas and implement and try them out quickly. In addition **h5vc** provides functions both for coverage analysis as well as calling of variants in single samples or pairs of samples. These implementations are intended as illustrative examples to the users of **h5vc** and while the functionality for coverage analysis might be called as good as it can get (since the task is so simple), the functions for calling variants (SNVs and InDels) are not intended as competition to the established variant calling tools. While I believe the variant calling functionality provided by **h5vc** to be adequate for its intended tasks I will not perform a detailed analysis of the sensitivity, specificity and runtime of the implementations, since the main point of **h5vc** is not to implement one specific variant calling function, but rather to enable the research community to easily experiment with different approaches to solving the problems at hand. With that in mind I show examples of the implementation of functions to be used with `h5dapply` in Listing 3 as well as their actual application to a specified genomic region of interest in Listing 4.

```
binnedCoverage <- function( data, sampledata ){
  ret <- rowSums( data$Coverages )
  names(ret) <- sampledata$Sample[order(sampledata$Column)]
  ret <- as.data.frame(t(ret))
  ret$Start <- data$h5dapplyInfo$Blockstart
  ret
}
```

Listing 3: Example code of the `binnedCoverage` function provided by the **`h5vc`** package.

```
data <- h5dapply(
  filename = tallyFile,
  group = "/ExampleStudy/22",
  names = c( "Coverages" ),
  blocksize = 500,
  range = c(38750000,39250000),
  FUN = binnedCoverage,
  sampledata = sampleData
  )
data <- do.call(rbind, data)
rownames(data) = NULL
head(data)
```

Listing 4: Example code of applying the `binnedCoverage` function provided by the **`h5vc`** package to region `22:38750000-39250000` of the human genome in bins of size 500bp.

4.6.1 Challenges in comparative variant calling

One of the most fundamental challenges in cancer sequencing analyses is the problem of comparative variant calling. In most cases the question we are trying to answer is: What are the differences between two or more samples in terms of the presence and penetrance (allelic frequency / zygosity state) of variants in those samples. In the setting of comparative cancer sequencing the most common comparison will be between a cancer sample and its matched control sample in order to find somatic variants. For population studies one might also be interested in $1 : N$ as well as $N : M$ comparisons, e.g. one cancer sample compared to a set of control samples ($1 : N$) or a set of cancer samples compared to a set of control samples ($N : M$).

Up until recently (around 2012) this was a hard problem with few dedicated softwares available to perform such comparisons. Prior to the development of **`h5vc`** I was using the

GATK (McKenna et al. (2010)) both for processing the BAM files as well as for variant calling. The GATK provides a variant caller that can operate on multiple samples at once (the `UnifiedGenotyper`) but the algorithm is based on an assumed diploidy of the samples and will likely not perform well when confronted with polyploidy (as can occur in cancer samples).

There are a number of stand-alone solutions for somatic variant calling available to the research community, e.g. MuTect (Cibulskis et al., 2013), SomaticSniper (Larson et al., 2012) and Strelka (Saunders et al., 2012). When I was first trying to perform comparative multi-sample variant calling in one of my cancer sequencing projects, described in Section 5, I realised that the `UnifiedGenotyper` was not able to provide calls of sufficient quality for our needs and using MuTect seemed more promising since the tool was specifically designed to address this question.

Ultimately I decided to rely on my own implementation using the framework provided by **`h5vc`** which allowed me to have full control over the algorithm and tremendously improved the way in which I was able to interact with the data. Furthermore it was of great importance to me, to be able to quickly visualise variant calls in an automated fashion (Section 4.6.5) and the available tools (e.g. batch programming IGV to generate screenshots – Thorvaldsdottir et al. (2013)) were too cumbersome and inflexible for my needs.

4.6.2 Variant calling implementations

`h5vc` contains example implementations of variant calling functions which I developed during my work on cancer sequencing analysis (Section 5). It should be noted that I do not intend to compete with any of the stand-alone solutions mentioned in Section 4.6.1, however most somatic SNV calling algorithms can likely be implemented as a series of operations on matrixes of mismatch counts and coverages and can therefore be implemented in R (or C made available as an R function through e.g. Rcpp Eddelbuettel and François (2011)). Since **`h5vc`** provides a framework for the quick implementation of algorithms based on nucleotide tallies one can therefore implement most somatic SNV calling algorithms as R functions that can be applied to HDF5-based nucleotide tallies through the `h5dapply` function provided by **`h5vc`**.

4.6.2.1 Principles – Error Profiles

A very useful metric for determining the reliability of a given variant call is an error profile of the position and the surrounding region. This can help to spot error prone regions and define statistical tests that use a more accurate prior probability for mismatches than the error rate as specified by the manufacturer of the sequencing machine. That the error rate observed in sequencing experiments is indeed position dependent is apparent from just looking at the data and can be nicely visualised using **`h5vc`** (see Figure 4.5 for an example). I will explain the usage of background error frequency data in variant calling in cohorts of samples in the following sections.

`backgroundFrequencyFwd`	The averaged frequency of mismatches/deletions at the position of all samples of type Control on the forward strand
`backgroundFrequencyRev`	The averaged frequency of mismatches/deletions at the position of all samples of type Control on the reverse strand
`pValueFwd`	The p.value of the Binomial test of the support and coverage in the case sample versus the estimated background error frequency `backgroundFrequencyFwd` (forward strand)
`pValueRev`	The p.value of the Binomial test of the support and coverage in the case sample versus the estimated background error frequency `backgroundFrequencyRev` (reverse strand)

Table 4.4: Explanation of the values used to annotate variant calls with information about the estimated background mismatch frequencies in the cohort.

An illustration of how the availability of the tally data of a whole cohort in an easily accessible matrix-like format can be helpful for variant calling approaches can be seen in Figure 4.3.

4.6.3 Paired SNV and deletion calling with h5vc

`h5vc` provides functions to perform comparative variant calling (`CallVariantsPaired`) which calls SNVs and small deletions (i.e. deletions that can be detected form single-read alignments). The implementation is very basic and relies on absolute cut-offs for the read support and total coverage of a variant position (the number of reads supporting the variant and the total number of reads overlapping the position). In addition to the cut-off based decision to call a position a potential variant a comparison to the background frequency of mismatches in the whole cohort may be performed. Table 4.4 contains explanations of 4 values that this comparison generates for each variant call.

In this way the information about the background mismatch frequencies that is available from the cohort is integrated as a set of scores for each variant, wich we can apply filters to in the downstream analysis. For example when calling somatic variants we might want to exclude positions that show a high allelic frequency over all of the controls in the cohort, even if the matched control of the cancer sample shows no support for the variant at all.

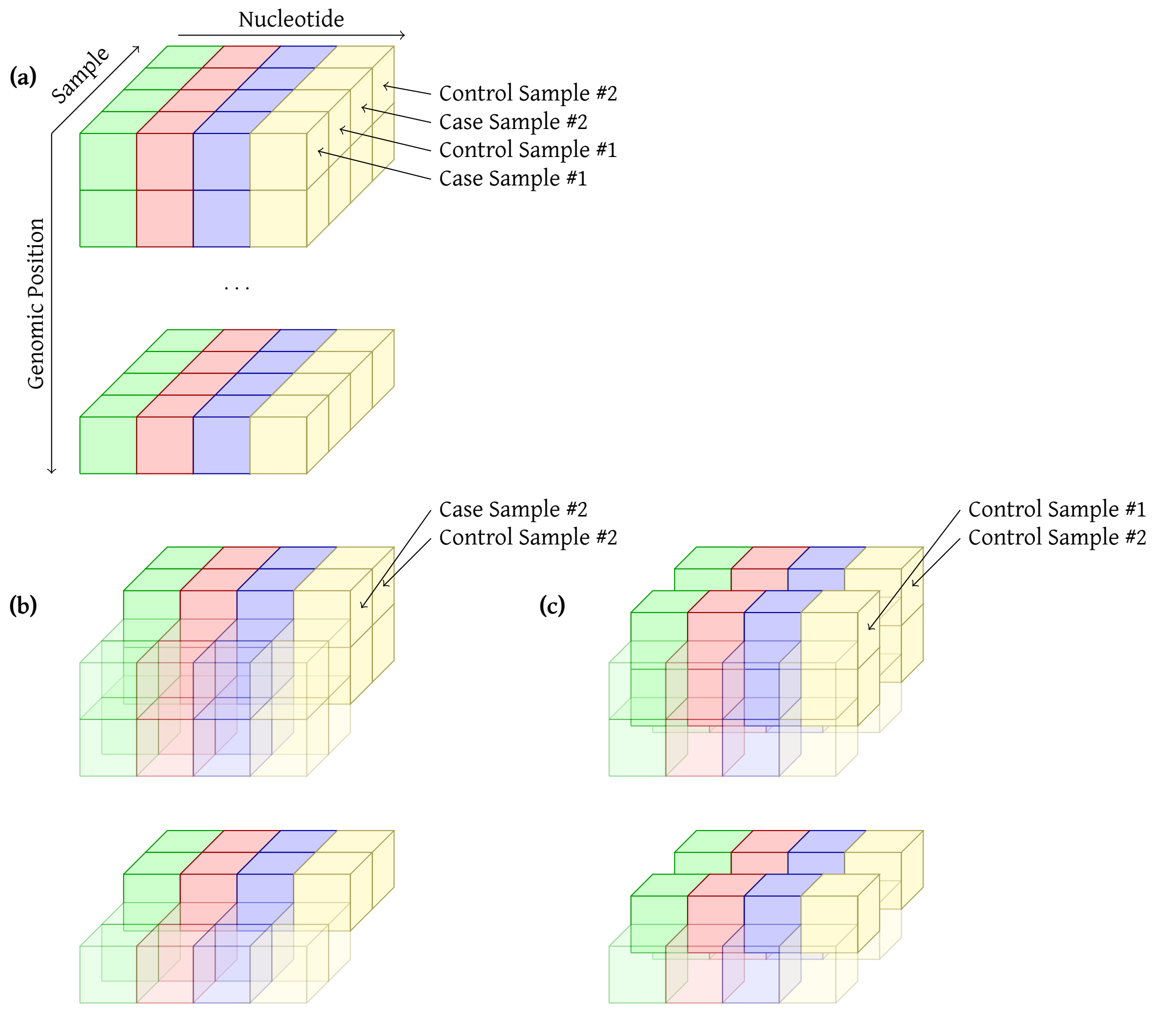


Figure 4.3: Illustration of the layout of mismatch tallies for a cohort of two pairs of samples (Case and Control sample of patients 1 and 2). Tallies are shown for only one strand to reduce the dimensionality of the data that has to be plotted (i.e. limiting to three dimensions) and any algorithm designed to work on nucleotide tallies will likely be implemented to handle the actual 4-dimensional data that includes information for each strand. **(a)** General layout of the mismatch tallies of the four samples on one strand. Colours encode the alternative allele along the horizontal axis and the vertical axis represents the genomic position. **(b)** Example of how the data would be subsetted to perform comparative variant calling in the two samples of patient 2. **(c)** Example of usage for calculating background mismatch frequencies from the control samples of all patients. By making the data available as a matrix we can implement many different algorithms by expressing them through matrix operations, e.g. the comparative variant calling is implemented through subsetting and comparing two slices (the case and control sample respectively). The background mismatch frequency estimate is implemented through subsetting on only the control samples followed by summarising over all controls and dividing by the sum of coverages in them. The coverage can be obtained as a matrix from the HDF5 file as well.

In the paired variant calling algorithms implemented in **h5vc** the basic workflow is using the following steps:

1. extracting the nucleotide tally data for each pair of samples from the cohort data (that was provided by a call to `h5daply` or `h5readBlock`), this step uses the sample meta-data that describes the cohort to find the appropriate indices in the sample dimension of the datasets.

2. applying the filters defined in the configuration list `cl` which is generated by a call to the `vcConfParams` function. The filters that are applied are hard cut-offs defining minimum and maximum allowable values for coverage and support per strand for both the case and matching control sample, e.g. a minimum support per strand in the case sample is usually implemented alongside a maximum support per strand in the control sample in order to call somatic variants. Such filters can be expressed as simple matrix operations on the nucleotide tally data. An example of this is given in Listing 5.

3. if the `returnDataPoints` flag is not set we return the list of positions satisfying the applied filters.

4. if the `returnDataPoints` flag is set we collect relevant variant specific data for all positions that passed the filters; this includes the support and coverage values in both the case and control sample as well as the chromosome, positions and alleles involved.

5. if the `annotateWithBackground` flag is set we additionally calculate the mismatch frequency that is observed in all control samples of the cohort combined for each of the strands, i.e. the forward and reverse strand-specific mismatch frequency of all controls in the cohort; furthermore a p-value is calculated for each strand by applying a Binomial test against the hypothesis that the true probability of success (i.e. the probability of sequencing a read with the alternative allele) is smaller or equal to the observed mismatch frequency in the controls of the cohort. Essentially the question is whether it is probable to observe the support we observed for the alternative allele in the case sample if the underlying probability of seeing a mismatch is equal to that of the control samples. If we can reject this hypothesis then this position could be a viable variant call and not just some artefact that could happen in any of the samples (also the controls).

This workflow is essentially identical for both SNVs and small deletions and for the purpose of the calculation the deletion of a single base is actually just treated like another kind of base, effectively using the following set of possible alternative alleles: `A,C,G,T,-`, where `-` represents a deletion of a single nucleotide.

```
caseCoverage = data$Coverages[caseColumn,,] #Coverage values of case sample
controlCoverage = data$Coverages[controlColumn,,] #Coverage values of control
cov_filter = (caseCoverage[1,] >= cl$minStrandCov &
caseCoverage[1,] <= cl$maxStrandCov & #Maximum case coverage forward strand
caseCoverage[2,] >= cl$minStrandCov & #Minimum case coverage reverse strand
caseCoverage[2,] <= cl$maxStrandCov &
controlCoverage[1,] >= cl$minStrandCov &
controlCoverage[1,] <= cl$maxStrandCov &
controlCoverage[2,] >= cl$minStrandCov &
controlCoverage[2,] <= cl$maxStrandCov)
```

Listing 5: Example code showing how to implement a coverage filter of tallies extracted from an HDF5 file (i.e. the `data` object) based on a configuration list provided to the variant calling function (`cl`). This code works on a pair of samples that are defined by `caseColumn` and `controlColumn`, respectively. The mapping of sample pairs to columns (i.e. indices in the sample dimension of tally datasets) is stored as sample meta-data within the HDF5 files (exemplified Table 4.2 for an example).

4.6.4 Calling single sample variants

For the case where we want to call variants in a single sample without comparing to a matched sample from the same patient **h5vc** implements the `callVariantsSingle` function. It is important to know that single sample variant calling is significantly more susceptible to false positives caused by artefacts which would be filtered out in the comparative variant calling setting (i.e. if mismatches are a consequence of sequencing chemistry and local sequence context they will appear in both the case and control sample and will therefore be discarded). This is more problematic when the single sample that is to be analysed is potentially polyclonal and carries copy number aberrations, since we can make no assumptions about the expected frequency with which variants should appear. In a monoclonal diploid sample a strong filter against allelic frequencies that differ from 50% can be applied, since we assume variants to either be heterozygous at 50% allelic frequency or homozygous at 100%.

In the implementation that I included in **h5vc**, variant calls are based on simple cut-offs for minimum support (total as well as per strand) and allelic frequency. Additionally an error probability is calculated for each variant call based on an expected error rate (e.g. $1/1000$ for Illumina machines) and the observed percentage of reads carrying mismatches. Furthermore a statistical test is applied to check for bias in the distribution of reads that carry mismatches amongst the two strand when compared to the distribution of all reads, i.e. the coverage. This can be informative when looking for artefacts that are strand specific (see Figure 4.5) since in those cases the distribution of support for the variant between the two strands is different from the distribution in the total number of reads and we can test for that. In my implementation I use Fisher's exact test as implemented in the R function `fisher.test`. The selection of likely variant positions is therefore based on them having at least a certain number of reads supporting the variant on each strand (the minimum per strand support) as well as at least a certain total number of supporting reads and an allelic frequency (as estimated from total support divided by coverage at the position) larger than a certain minimum. The provided annotations give the probability of observing the number of supporting reads in the background error model (e.g. the $1/1000$ rate of error that Illumina claims for their machines) as well as the p-value of the Fisher's exact test comparing the distribution of support amongst the two strands with the distribution of all reads amongst the two strands (strongly skewed distributions of support are indicative of artefacts).

4.6.5 Visualising variant calls

When calling single nucleotide variants or small deletions (but also more generally when calling any type of variant) it is important to implement reasonable measures for quality control. Usually the filters that are applied to a set of raw variant calls will rely on properties of the sample at the variant position itself, e.g. the number of supporting reads or the coverage at that position. Furthermore, properties of other samples in the cohort or the population in general (e.g. data from the 1000 genomes project - Abecasis et al. (2012)), such as mismatch frequency distributions amongst the samples and global minor allele frequency (GMAF) estimates should be used.

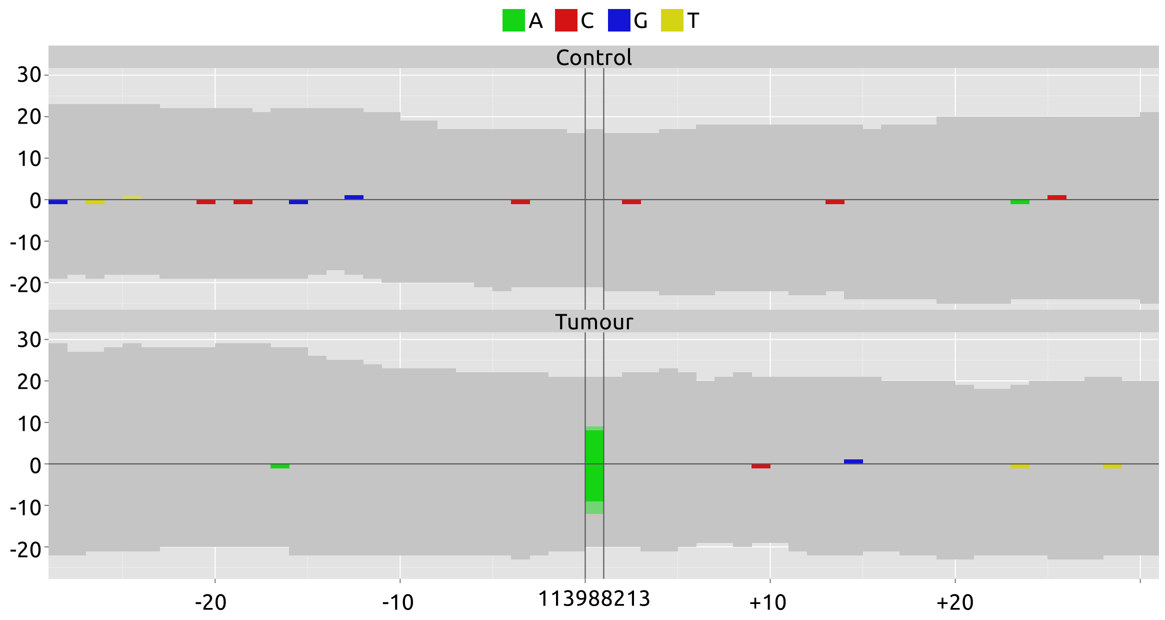
In addition to those position specific properties it can also be enlightening to investigate the immediately surrounding genomic region for unusual patterns of mismatches and likely artefacts. An example of this is given in Figure 4.4, where two variants are visualised which do not differ significantly in their position-specific properties but when seen in their genomic context one might arrive at different levels of confidence in the respective variant calls. In **`h5vc`** the `mismatchPlot` function is implemented, which generates this type of plots from the nucleotide tallies.

When processing variant calls from a cohort of sequencing samples I will usually apply some position specific filters to the variant calls, e.g. on the support, coverage, allelic frequency, GMAF and SIFT conservation score (Kumar et al., 2009) for coding variants to reduce the number of calls to a more manageable size, e.g. thousands or hundreds of calls. From this set of calls I will typically create HTML reports in the form of tables of position specific properties that contain links to additional information (e.g. overlapping genes and existing variants) as well as mismatch plots of the variant regions in all samples of the associated patient, i.e. when observing a variant in the primary tumour sample of a patient I will plot the mismatch plots for all of the samples from that patient, e.g. the control, the primary and potentially a relapse tumour. I typically use the **`ReportingTools`** (Huntley et al., 2013) `R/Bioconductor` package to create those reports.

4.6.6 Sequence and strand specific artefacts

During my work on analysing cohorts of sequencing samples I have encountered a variety of artefacts that can be mistaken for variants when they are more likely to be explained by specific effects of the library preparation and sequencing technology used. One example is

(a)



(b)

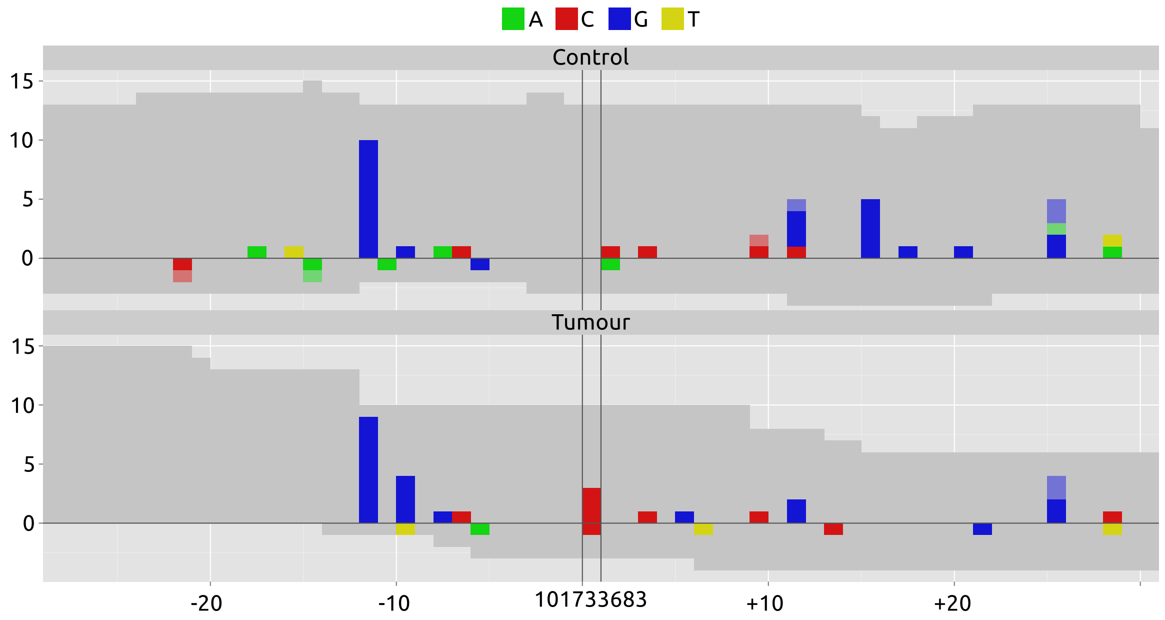


Figure 4.4: `mismatchPlots` of two candidate variant sites. Each sample (Control, Tumour) is shown in a separate panel, with the genomic position as a common x-axis centered around the position of the variant. Along the y-axis, alignment statistics of the forward and reverse strand are shown as positive and negative values, respectively. Grey areas represent coverage by sequences matching the reference, coloured areas represent mismatches and deletions. **(a)** Variant present in the tumour sample, but not in the control. **(b)** Variant of comparable position-specific statistics as the one shown in **(a)**. Note the noisyness of the region, which is not immediately obvious from the position-specific values alone.

the strongly strand-biased accumulation of mismatches that seem to be dependent on the local sequence and can therefore be observed with comparable intensity in unrelated sample.

Figure 4.5 shows mismatch plots of a region of the genome where such artefacts can be observed in multiple control samples of a cohort of paired sequencing samples. I did not further investigate those artefacts for common sequence context around their positions but think such analyses could help to better understand artefacts of this sort and help spot regions of the genome that might be more prone to their occurrence. The existence of highly strand-specific pile-ups of mismatches in all samples points to a technical rather than a biological reason. When calling variants in a comparative setting those artefacts are easily removed simply because they appear everywhere and one of the most basic filters applied in comparative sequencing analysis is for the absence of a tumour variant in the matching control. When performing single sample variant calling those artefacts are harder to spot. They can be filtered out by requiring support for variants to occur on both strands and annotating potential variant positions with the observed mismatch frequency of the variant in the rest of the cohort if there are more samples available. In certain situations, e.g. when one is looking for commonalities between a number of samples, the analysis is particularly vulnerable to such artefacts and it is always advisable to visualise the variants in order to spot problems.

4.7 Summary

In this section I have introduced the **h5vc** R/Bioconductor package, outlined the challenges researchers face when handling and analysing large amounts of HTS data and illustrated how the usage of **h5vc** and nucleotide tallies stored in HDF5 files can be used to adress those challenges. It is important to note that **h5vc** is essentially a framework that provides infrastructure to allow researchers to handle, analyse and (visually) explore large HTS data sets, which are challenging to handle with the current default suite of file types. It is not designed as a variant caller, a copy number analysis tool or a solution for any other specialised task one might encounter during the analysis of HTS data, instead **h5vc** provides access to a highly portable, accessible and compressed representation of the HTS data (i.e. the nucleotide tallies stored in HDF5 files) and comes with functionality that enables the research community to quickly implement and test new ideas and algorithms on the data. An example of this is the implemented variant calling functionality, which is simply a collection of pure R functions which could be written by researchers with limited programming knowledge. In a standalone application one needs to implement a substantial amount of bureaucracy code, e.g. parsing read alignments from BAM files, collating the information into a nucleotide tally representation and subsequent calling of likely variant positions, using **h5vc** and nucleotide tallies stored in HDF5 files we can omit the first two steps completely and instead focus on the scientifically interesting question of how to determine the positions of potential variants.

In the cancer sequencing projects that I am involved in (see next Section) this infrastruc-

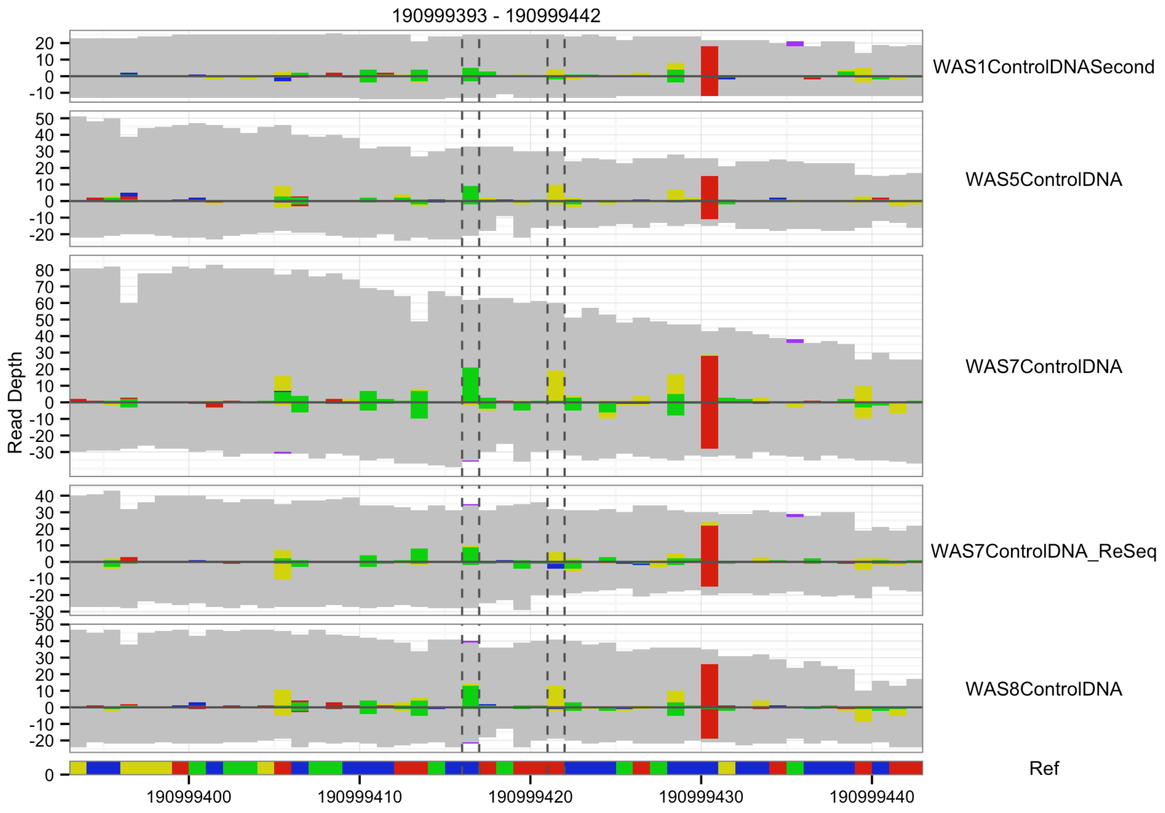


Figure 4.5: Example mismatch plots of a region of the genome with strong strand specific artefacts that can be found in a set of unrelated control samples (marked by vertical dashed lines). Note also the general noisiness of the region with a multitude of pileups of mismatches constituting low-frequency variants or artefacts.

ture has greatly improved my ability to provide solid analyses and meaningful reports when dealing with up to 21 samples at the same time. Colleagues in the group use the software on cohorts of up to 188 samples and I have heard reports from users of **`h5vc`** that work on cohorts with more than 300 samples.

Usage statistics for my package are shown in the table below, which contains information about the monthly number of downloads of **`h5vc`** as collected by the `Bioconductor` webpage. Since it was first introduced in the October release of `Bioconductor` in 2013 the number of downloads has stabilised around ~ 200 per month.

Month	Year	Number of distinct IPs	Number of downloads
Sep	2013	0	0
Oct	2013	100	157
Nov	2013	129	172
Dec	2013	126	262
Jan	2014	115	174
Feb	2014	121	168
Mar	2014	186	297
Apr	2014	198	378
May	2014	201	422
Jun	2014	269	401
Jul	2014	230	326
Aug	2014	224	346
Total		1376	2930

Table 4.5: Table of the total number of downloads as well as total number of distinct IP address from which **`h5vc`** was downloaded stratified by month. Note that the number of distinct IP address is the better estimate for the size of the user-base, since it compensates for people that re-download the package multiple times, e.g. because of an update. This data was taken from `http://bioconductor.org/packages/stats/bioc/h5vc.html` on the 3rd of September 2014.

5 Comparative cancer sequencing projects

In this section of my thesis I will provide an overview of the comparative cancer sequencing projects I am involved in. All of the projects in this section are dealing with cohorts of paired samples from cancers and matched control whose DNA has been sequenced using whole exome or WGS technology on Illumina machines. The challenges I faced when I started to work on those projects prompted me to implement the HDF5 based nucleotide tally framework provided by **h5vc** which I described in the previous section.

I will focus on three cancer sequencing projects, of which two projects are related to cancer as a side effect of gene therapy and use samples from the same set of patients. Both of these projects are collaborations with the group of Manfred Schmidt at the National Center for Tumour Disease in Heidelberg. The third project aims to investigate the genomic makeup of acute myeloid leukaemia samples sequenced with WES technology and is a collaboration with the group of Prof. Dr. Anthony Ho from the Universitätsklinikum Heidelberg.

5.1 Cancer as a side effect of retroviral gene therapy of Wiskott-Aldrich Syndrome

5.1.1 Introduction

Wiskott-Aldrich syndrome (WAS) is an X-linked, complex primary immunodeficiency disorder that is caused by a lack of expression of functional product of the *WAS* gene located at `X:48,534,985-48,549,818` on the forward strand. This disease leads to multiple dysfunctions in parts of the blood system, including defective T and B cell function, disturbed formation of the natural killer cell immunological synapse and impaired migratory responses of all leukocyte subsets. The symptoms experienced by affected patients include recurrent infections, thrombocytopenia, eczema, autoimmunity and an increased risk of lymphoma. In severe cases infections and haemorrhaging can lead to an early death and for those cases the standard therapy is allogeneic haematopoietic stem cell (HSC) transplantation. This kind of treatment carries a substantial morbidity and mortality risk, especially in cases where no human leukocyte antigen (HLA)-matched HSC donor is available.

An alternative therapeutic strategy for primary immunodeficiency diseases such as WAS, is HSC gene therapy (Kay, 2011). With this approach, clinical benefits and corrected function of immune cells have been shown in several studies, e.g. on patients with X-linked severe combined immunodeficiency disease (γc-SCID), adenosine deaminase (ADA)-deficient SCID and chronic granulomatous disease. In this context there is always a risk that the location of the vector integrations in the genome can influence gene expression. Both silencing as well as induced over-expression occur. If tumour-suppressors or oncogenes are affected the development of leukaemia is a possible side-effect of the treatment. For each separate treated cell the risk of creating such effects is relatively low, but the risk for the patient is substantially increased by the creation of many millions of modified cells in the course of the treatment.

Our collaborators have performed a study into the feasibility of HSC gene therapy for treatment of WAS on a total of 10 patients (aged between 2 and 14 years). Preliminary results for 2 of those patients have been published (Boztug et al., 2010). A substantial number of patients developed some form of leukaemia at a later time.

The aim of my work in this project was the genomic characterisation of pairs of matched tumour and control samples from the patients of this study that had developed leukemia as a side-effect of the HSC gene therapy applied to treat their WAS. Initially there were 3 patients that developed T-Cell acute lymphoblastic leukaemias (T-ALLs) and this number has increased to 6 (out of 10) during the time that I worked on this project. The common factor in those T-ALLss was that the dominant clone could be determined to carry a vector integration in or close to the *LMO2* oncogene. Furthermore, three patients developed acute myeloid leukaemia (AML), two of which did so after their initial T-ALL had been treated (the third patient did not develop a T-ALL previously). Those AMLs were induced by vector integration in or near *MDS1* and *MN1*, respectively.

All patients in this study were closely monitored and therefore control samples and early time-points of leukaemia are available and have been sequenced (see Table 5.1 for an overview). All samples were sequenced from peripheral blood. In Tables 5.2 I give an overview of the 10 patients involved in the study. This includes the patient ID, their age at the time of gene therapy as well as the causative mutation to their development of WAS. The columns listing number of infused cells as well as the estimated average number of vector integrations per cell are important, since a higher vector-load leads to more integrations and therefore also more possibilities for adverse effects. Note the difference in the number of days post gene therapy that the leukaemias were diagnosed (last column). This ranged from 488 to 1812 days post gene therapy. One of the key questions we were trying to answer was whether there exists a set of second hits in those leukemias, i.e. if there are specific genetic events necessary that have to occur in addition to the vector integration, for the cell to become cancerous. This question arose from the observation of the markedly different timespans until diagnosis of leukaemia after gene therapy, hinting at a possible second event that has to occur.

For both patient 1 and 7 we performed additional sequencing runs because of insufficient data quality in the sequencing data of the initial runs. In further analyses I disregarded those initial samples, with labels *WAS1PrimaryDNAFirst*, *WAS1ControlDNAFirst*, *WAS7PrimaryDNA* and *WAS7ControlDNA*. Unless I specify otherwise all statements regarding samples from those patients are made with respect to the second set of sequencing runs we performed on them. I explain the reasoning behind requesting additional sequencing runs for patient 7 in Section 5.3.1.2.

Patient ID	**Sample ID**	**Library**	**Disease**
WAS1	primary tumour	WGS	T-ALL (LMO2 induced)
WAS1	primary tumour (re-sequencing)	WGS	T-ALL (LMO2 induced)
WAS1	pre-gene-therapy sample	WGS	-
WAS1	pre-gene-therapy sample (re-sequencing)	WGS	-
WAS1	primary tumour	WES	AML (MN1 induced)
WAS1	pre-gene-therapy samples	WES	-
WAS5	primary tumour	WGS	T-ALL (LMO2 induced)
WAS5	relapse tumour	WGS	T-ALL (LMO2 induced)
WAS5	pre-gene-therapy sample	WGS	-
WAS6	primary tumour	WGS	T-ALL (LMO2 induced)
WAS6	pre-gene-therapy sample	WGS	-
WAS7	primary tumour	WGS	T-ALL (LMO2 induced)
WAS7	primary tumour (re-sequencing)	WGS	T-ALL (LMO2 induced)
WAS7	pre-gene-therapy sample	WGS	-
WAS7	pre-gene-therapy sample (re-sequencing)	WGS	-
WAS8	primary tumour	WGS	T-ALL (LMO2 induced)
WAS8	pre-gene-therapy sample	WGS	-
WAS8	pre-leukaemia / post-gene-therapy sample	WGS	-
WAS8	primary tumour	WES	AML (MDS1 induced)
WAS8	pre-gene-therapy samples	WES	-
WAS9	primary tumour	WGS	AML (MDS1 induced)
WAS9	pre-gene-therapy samples	WGS	-
WAS9	primary tumour	WES	AML (MDS1 induced)
WAS9	pre-gene-therapy samples	WES	-

Table 5.1: Overview of the sequenced samples in the cancer cohorts relating to the WAS HSC gene therapy study. The **Library** column contains abbreviations for WGS and WES. Some samples had to be re-sequenced due to quality issues.

Patient Nr	Age (at GT)	WAS mutation	Infused CD34+ cells per kg ($\times 10^6$)	Vector copies per cell	Clinical Status post gene therapy
1	3	IVS6+1 G>T	18.6	2.4	T-ALL (day 1813), AML during maintenance therapy
2	3	Arg86His	13.5	2.7	Well
3	3	Glu133Lys	2.9	3.0	Well, status post haploid-identical haematopoietic stem cell transplantation (HSCT)
4	4	Arg34X	24.9	3.2	Well
5	12	Glu31Lys	17.9	5.2	T-ALL (day 1073), relapse shortly after allogeneic HSCT and death
6	4	IVS8+1_4 delGAGT	20.6	3.0	T-ALL (day 488), now well and in remission after allogeneic HSCT, but lost to further follow up
7	3	IVS3+1 G>T	13.5	2.7	T-ALL (day 1105), in remission after allogeneic HSCT
8	3	Ala134Thr	20.9	1.7	T-ALL (day 792), AML during maintenance therapy
9	2	Val303fsX4	21.1	2.3	AML (day 1165), in remission after allogeneic HSCT
10	14	His30del	9.7	3.5	Weak vasculitis and residual proctitis

Table 5.2: Overview of the 10 patients that were part of the original study into gene therapy as a treatment for WAS. The original (WAS causative) mutations are shown as well as the number of cells used in for the gene therapy, the average number of vector copies per cell and the clinical status after gene therapy. Note that the number of vector copies per cell differ quite a lot between the different patients.

Patient ID	Sample time (days post gene therapy)	Peripheral blasts at diagnosis (%)
WAS1	1813	56.5
WAS5	1073	63
WAS6	488	74
WAS7	1105	98
WAS8	792	91
WAS9	1165	20 – 30

Table 5.3: Table showing for each patient the percentage of leukaemic blasts in the peripheral blood at the time of diagnosis of the initial leukaemia (given in the second column). This percentage can be used as an estimate of the percentage of tumour cells in the sequenced samples and is therefore our first approximation of the tumour purity in the sequencing samples.

Table 5.3 lists all samples that were sequenced with WGS technology and gives both the time post gene therapy that the leukaemia was diagnosed as well as the percentage of leukaemic blasts in the peripheral blood at the time of diagnosis.

In totaI I analysed the two cohorts corresponding to the 6 T-ALLs with matched control and additional samples (relapses as well as pre-leukaemic samples) and the 3 AMLs with matched controls, relapse and pre-leukaemic samples.

5.1.2 Vector integration detection

5.1.2.1 Method

Detecting and mapping vector integration sites is an essential step towards understanding the biology of the leukemias presented in this cohort. We used LAM-PCR (Schmidt et al., 2007) to identify vector integration sites in the samples. Furthermore I developed an algorithm for the identification of vector integration sites from WGS data at single nucleotide resolution and implemented the solution using the HTSeq framework described in Section 2.

The detection of vector integrations relies on an alignment of the sequencing reads to a compound genome containing the chromosomes of the human reference genome and an additional contig that contains the vector sequence itself. Using this approach, vector integration site detection becomes a specialised case of translocation detection. Given that our dataset was sequenced with a paired-end library, I could exploit the pair information to map the vector integration sites by searching for pairs of mapped reads where one read mapped to the vector contig and the other to one of the reference chromosomes (Figure 5.1).

After identification of the set of read pairs that indicate potential integration sites I performed clustering of those pairs and annotated the clusters of read pairs with their mapping interval (the spanning genomic interval that covers all reads assigned to a cluster) and support (number of reads in the cluster). Figure 5.2 shows a typical cluster.

I implemented the algorithm for vector integration detection as a Python script that can be downloaded from the Huber Group EMBL Servers[1]. The algorithm performs the following

[1]http://www-huber.embl.de/users/pyl/Tools/VectorIntegrationSiteDetection/vector.

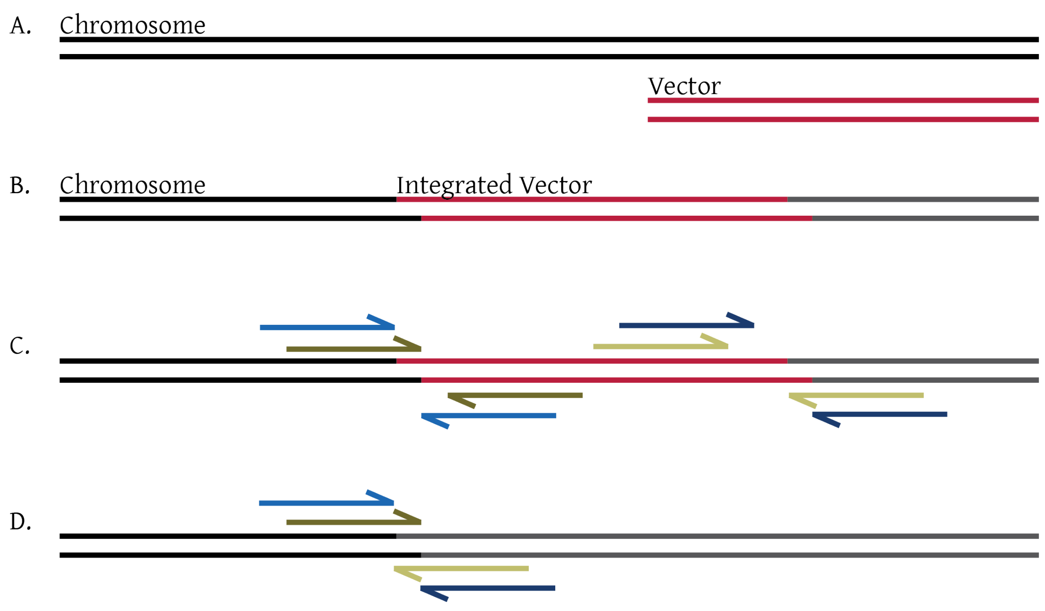


Figure 5.1: Overview of the typical pattern of read alignments around a vector integration site. (A) Shows the target chromosome and the vector. (B) Shows the vector as it is integrated into the chromosome. Note the sticky ends at the integrations site. (C) Shows the way reads are sequenced from the construct (colours of reads encode the respective pair) (D) Shows how the reads on the chromosome side will align to the human reference genome at the integration site. Note how the clusters of reads from each side of the vector overlap slightly because of the sticky end type cut that is used to facilitate the integration.

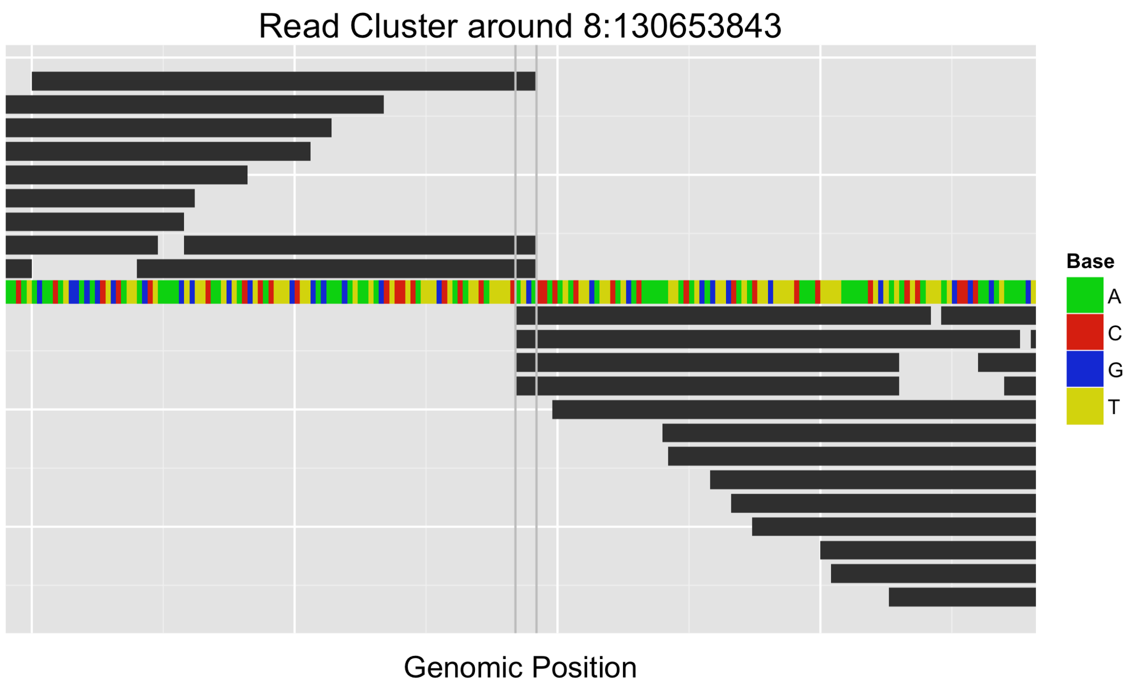


Figure 5.2: Overview plot showing the reads of a typical cluster that indicates a vector integration site at position 8:130653843. Reads are shown as dark grey blocks with a reference genome track in between the reads mapping to the forward strand (reads plotted above the reference track) and the reads mapping to the reverse strand (reads plotted below the reference track). The vertical grey lines mark the overlap between the two clusters, which is the typical 4 base pairs caused by the usage of sticky ends for the vector integration.

steps on an input `BAM` file:

1. Iterate through the provided input `BAM` file and store all records that represent aligned reads with an aligned mate where exactly one of the reads in the pair is mapped to the vector (the extra contig we added to the genome)

2. for each pair of reads define a pair of genomic intervals representing the regions of the genome where the reads are mapped

3. cluster the interval pairs by overlap, where two pairs of intervals are clustered together if both intervals of each of the pairs overlap, i.e. they are likely to originate from the same event. Note that we relax this requirement on the vector contig since it is repetitive and the reads mapping there will have two equally good alignments at either end of the vector. Therefore overlap is only stringently required on the non-vector side of the cluster.

4. find pairs of clusters that overlap on opposing strands which are indicative of a vector integration event (see Figure 5.1) and report them

The two clusters that form around a vector integration site overlap by a number of bases, which was typically 4 bases in our data-set. The reason behind this overlap is the way in which the double strand break is created when the vector integrates since it uses sticky ends. The actual number of observed overlapping bases can vary depending on sequencing depth, i.e. it depends on whether we have enough reads to cover the breakpoint extensively. Such variations can also be the effects of local sequence similarity, where the vector sequence that is inserted and the normal progression of the chromosome (i.e. the sequence that is stored in the reference genome) are identical for a couple of bases and therefore the reads can align over the breakpoint.

I defined the properties that a pair of read-pair clusters have to exhibit in order to be a good indicator of an integration event in the following way:

- each cluster has to be anchored on one end in the vector sequence and on the other end in the reference genome

- the reference anchors of both clusters must overlap by a few basepairs (with the expected number being 4 bp) when disregarding the strand that they are mapped to

- the strands of the reference anchors the clusters must be different from each other (i.e. one maps on the $+$ and one on the $-$ strand)

`integration.latest.py`

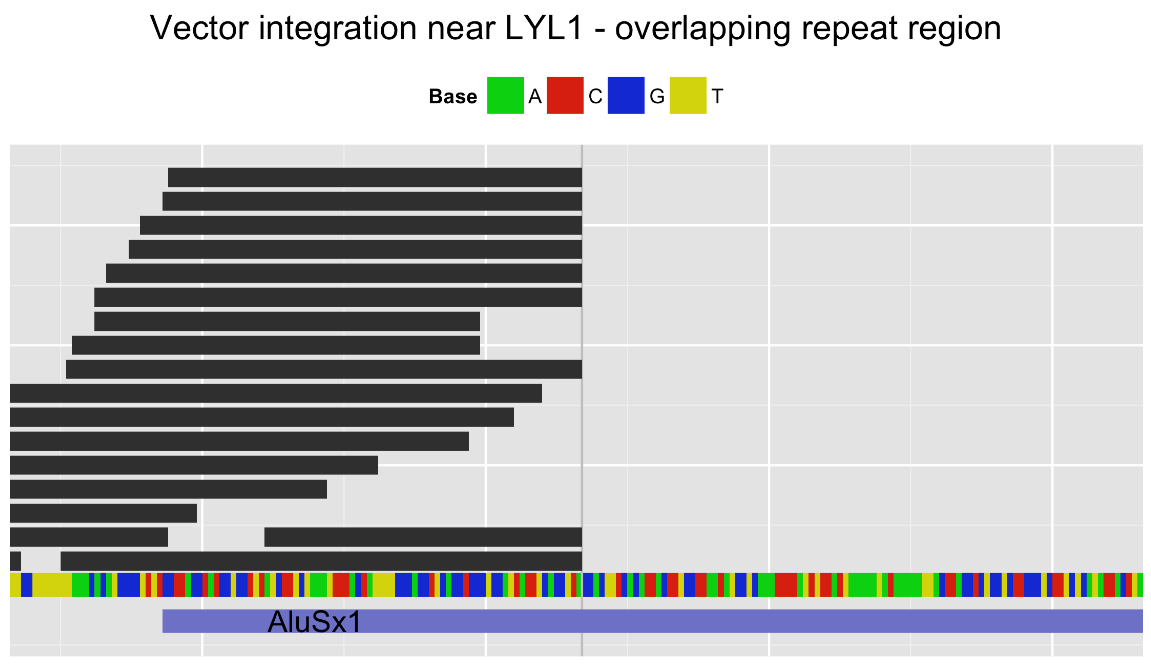


Figure 5.3: Overview plot showing the reads of an unmatched cluster that indicates a vector integration site overlapping an Alu element (AluSx1 - shown in light blue). Note the complete absence of reads mapping to the reverse strand.

5.1.2.2 Results

The resulting list of identified vector integration events is shown in Table 5.4. In this table we can see that all of the T-ALL samples carry a vector integration event close to or overlapping with the *LMO2* oncogene (highlighted in red in the table). Those events had been previously detected by my collaborators by LAM-PCR. Furthermore it is apparent that the number of vector integrations per clone varies between patients. This is likely the effect of different virus loads in the initial treatment, i.e. the retroviral gene therapy. Interestingly for some patients (1, 5, 6 and 7) the number of detected integration sites in the leukaemic clones as shown in Table 5.4 does correlate with the estimated vector load during gene therapy shown in Table 5.2 with a correlation of 0.89. Patients 8 and 9 do not show such correlation and have an estimated vector load of 1.7 and 2.3 copies per cell at gene therapy, respectively, while their leukaemic clones exhibit 5 and 8 different vector integrations. This is not unexpected though, since the distribution of vectors amongst treated cells can fluctuate.

Note that there are specific cases where not all of the requirements for a reliable read-pair cluster defined in the previous section can be fulfilled. An example is the cluster mapping near the *LYL1* gene (lymphoblastic leukemia derived sequence 1; located on the reverse strand at `19:13,209,847-13,213,975`) in the leukaemia sample of patient 8, which lacks support on one side, however the very good signal on the other side prompted further investigation. To be able to find such cases as well as cases where due to low coverage the support on one side of the cluster is weak or non-existent the tool I implemented can be configured to relax the requirements for read-pair clusters. In the case of the vector integration site close to *`LYL1`* I observed a single cluster of very good support on the forward strand but I could not find a cluster to pair it with. The reason for this can be found by investigating annotated repetitive regions in the reference genome which tells us that the insertion point lies close to a repetitive region in such a way that all reads originating upstream of the insertion site can be mapped, whereas the reads originating from the insertion site and downstream of it fall fully into the repetitive element and can therefore not be mapped properly. In this specific case we are dealing with an Alu (i.e. repetitive) element of the short interspersed element (SINE) class which lies in the interval `19:13,216,593-13,216,894` and is named AluSx1 as classified by RepeatMasker (Smit et al., 2007). The forward-strand cluster ends at position `13,216,666` and therefore overlaps slightly with AluSx1, but since the overlap is 73 bases and the reads are 101 bases long we still have those 38 bases as the front of the reads to anchor their alignments. The corresponding cluster on the reverse strand would be starting at position `13,216,662` (accounting for the characteristic 4 bp overlap) and reads originating from this cluster will fall completely into AluSx1 since it has length 302. As a consequence of this, the reads of the reverse-strand cluster can not be reliably mapped to this region. Either they are not mapped at all if the aligner was required to only report reads with unique alignments, or they can be mapped on other copies of the repetitive element, thereby diluting or completely removing the signal we were looking for. In our dataset multiple alignments were allowed and a post-processing step removed alignments that were not called primary (the primary alignment of a read has the best mapping score amongst all reported alignments). Note that the alignment strategy I employed would not yield a primary alignment for reads that have more than one optimally scoring alignment available to them, thus removing all reads whose best scoring alignments map equally well in more than one region of the genome. Figure 5.3 illustrates the break-point near *`LYL1`* and shows the mapping of reads in that region.

In conclusion, I was able to faithfully detect vector integrations based on Illumina shotgun sequencing data at single base-pair resolution and corroborate the findings of my collaborators (LAM-PCR based vector integration detection).

5.1.2.3 Vector Integration detection from whole exome sequencing data

When attempting to run my tool on the AML samples that have only been sequenced with whole exome libraries it became clear that in many cases we lack detection power simply because most vector integrations do not in fact overlap with coding regions and we simply do not get reads from those regions. For the 4 samples processed this way (primary AMLs from patients 1, 8 and 9 as well as a relapse sample from patient 8) only the primary AML sample of patient 8 contained clusters with at least two reads supports, though all of those were unpaired (only one side was found). From those results I conclude that whole exome sequencing data is not a good data source for calling vector integrations, since such data only

Forward-strand cluster				Reverse-strand cluster				Common Properties	
Chrom	From	To	Support	Chrom	From	To	Support	Distance	Sample
8	80813647	80813794	7	8	80813791	80813972	11	-3	WAS1PrimaryDNAFirst
11	33934228	33934414	12	11	33934411	33934579	15	-3	WAS1PrimaryDNAFirst
X	53311089	53311255	9	X	53311250	53311414	14	-5	WAS1PrimaryDNAFirst
8	80813629	80813794	10	8	80813791	80813948	5	-3	WAS1PrimaryDNASecond
11	33934202	33934406	8	11	33934411	33934599	9	5	WAS1PrimaryDNASecond
X	53311029	53311255	8	X	53311250	53311396	3	-5	WAS1PrimaryDNASecond
1	47703242	47703445	12	1	47703442	47703632	14	-3	WAS5PrimaryDNA
1	208005333	208005617	12	1	208005613	208005809	9	-4	WAS5PrimaryDNA
6	37173972	37174143	14	6	37174140	37174343	11	-3	WAS5PrimaryDNA
10	17496754	17496929	11	10	17496937	17497128	10	8	WAS5PrimaryDNA
11	33945916	33946105	17	11	33946101	33946320	12	-4	WAS5PrimaryDNA
11	38177580	38177776	13	11	38177773	38177973	14	-3	WAS5PrimaryDNA
14	92997399	92997587	12	14	92997578	92997800	15	-9	WAS5PrimaryDNA
17	16299791	16299979	9	17	16299976	16300168	8	-3	WAS5PrimaryDNA
11	33946778	33946851	2	11	33946847	33946931	3	-4	WAS6PrimaryDNA
1	47695816	47696058	38	1	47696055	47696283	38	-3	WAS7PrimaryDNA
11	33915411	33915680	35	11	33915677	33915947	40	-3	WAS7PrimaryDNA
1	47695786	47696058	18	1	47696055	47696300	16	-3	WAS7PrimaryDNAReSeq
11	33915419	33915680	18	11	33915677	33915943	14	-3	WAS7PrimaryDNAReSeq
2	158288046	158288297	16	2	158288294	158288477	14	-3	WAS8PrimaryDNA
7	110779563	110779765	15	7	110779760	110779933	16	-5	WAS8PrimaryDNA
8	130653635	130653845	13	8	130653841	130654040	18	-4	WAS8PrimaryDNA
11	33904947	33905149	16	11	33905144	33905337	15	-5	WAS8PrimaryDNA
19	13216467	13216666	19	NA	NA	NA	NA	NA	WAS8PrimaryDNA
2	8679211	8679493	12	2	8679489	8679748	8	-4	WAS9PrimaryDNA
2	16618975	16619237	12	2	16619233	16619488	20	-4	WAS9PrimaryDNA
3	13456935	13457178	16	3	13457174	13457384	13	-4	WAS9PrimaryDNA
3	169081857	169082080	13	3	169082075	169082297	13	-5	WAS9PrimaryDNA
7	21395410	21395598	10	7	21395594	21395776	4	-4	WAS9PrimaryDNA
12	46878057	46878328	11	12	46878325	46878547	15	-3	WAS9PrimaryDNA
22	28064024	28064248	17	22	28064244	28064437	11	-4	WAS9PrimaryDNA
22	32534505	32534737	17	22	32534750	32534974	7	13	WAS9PrimaryDNA

Table 5.4: Table listing the detected vector integration in all whole genome sequencing samples from the WAS T-ALL and AML datasets. Note that WAS9PrimaryDNA is the only AML sample that was sequenced with WGS and therefore shows sufficient detection power for vector integrations. Each row represents one cluster pair indicating a potential vector integration site. The columns shown for each cluster are the chromosome (Chrom), the starting position (From), the end position (To) and the number of read pairs supporting the cluster (Support). Additionally the column 'Distance' indicates the number of base-pairs that lie in between the end of the forward-strand cluster and the beginning of the reverse-strand cluster (negative values indicate overlap, which is the expected behaviour). Note that there are clusters with positive distance, which means that those clusters do not actually overlap. This can be the case if the local coverage is too low and we lack power to detect the exact breakpoint at single base-pair resolution, i.e. we simply didn't sequence a read overlapping the break-point exactly. The last column indicates the sample that the vector integration was detected in. Cluster pairs located close to or overlapping with the *LMO2* oncogene (LIM domain only 2 – located on the reverse strand of `11:33,880,122-33,913,836`) are highlighted in light red and cluster pairs close to or overlapping with the *MDS1* oncogene (MECOM MDS1 and EVI1 complex locus – located in the reverse strand of `3:168,801,287-169,381,406`) are highlighted in light blue. The cluster on chromosome 19 in sample *WAS8PrimaryDNA* is lacking a partner on the reverse strand because of a repetitive element nearby, see Section 5.1.2.2 and Figure 5.3 for details.

has power to detect anything in the $\sim$ 3% of the genome that are sequenced.

5.2 Data Processing Steps for Cancer Sequencing Data

Unless specified in the methods sections for the different projects all samples from all sequencing projects were processed in the same way up to the generation of the `HDF5` nucleotide tally files. Specifically I treated all WAS related samples as one cohort up until the point of analysis, therefore both the T-ALL and AML samples were aligned, preprocessed and tallied with the same commands. The data processing pipeline from raw reads in `FASTQ` format to fully processed `BAM` file and HDF5-based nucleotide tally representation is outlined below.

5.2.1 Read alignment

The reads were aligned using the GSNAP read alignment software (Wu and Nacu, 2010) version `2013-09-11` using the parameters described in Table 5.5. I left all other options at their respective default settings.

Parameter	Meaning
`--nthreads=12`	Threaded processing using 12 processes at the same time
`--batch=4`	Using batch mode 4 which instructs the aligner to load the genome index into the computers memory completely to improve speed; smaller values lead to the genome index being loaded from disk during usage which reduces memory consumption at the cost of increased runtime
`--npaths=4`	Maximum number of alignments to report for each read
`--quiet-if-excessive`	Do nor report any alignments for reads which have more than `--npaths` valid alignments of equal quality (in our case this restricts reporting to reads with no more than 4 equally well mapping alignments)
`-A sam`	Output the alignments in SAM format (Li et al., 2009) instead of the default GSNAP format

Table 5.5: Table listing the parameters used for the alignment of paired-end reads in the cancer sequencing projects I have worked on.

The alignment was heavily parallelised using the ability of GSNAP to process FASTQ files in parts to split each sequencing lane into ten parts which in turn used 12 sub-processes each as described in Table 5.5. In this way the generation of raw read alignments is sped up significantly (using a sufficiently powerful compute cluster; I used the EBI cluster for my calculations) at the expense of having to add additional post-processing steps for merging of the output files. Since GSNAP only outputs SAM file format, which is text based and therefore unsuited for large data I used samtools to directly convert the outputs to the compressed binary BAM file format.

5.2.2 Alignment post-processing

After the raw read alignments had been generated several post-processing steps were necessary to prepare the alignments for analysis. Firstly the alignments of each sample (we had generated 10 parts per lane in the raw alignment step) were merged into a single file using the `MergeAndSort` tool from the Picard[2] toolbox for manipulation of SAM/BAM files, followed by marking of duplicate alignments using the `MarkDuplicates` tool from the same toolbox. This step is necessary since read alignment pairs with the exact same starting and end position are likely to be PCR duplicates of the same fragment of DNA and we want to count each fragment only once.

The next step is the realignment of reads around InDels which serves to reduce artefacts when performing SNV calling. Those artefacts occur with reads that span InDels which have more than one valid and equally scoring alignment. When the alignment program chooses which possible alignment to report it is not aware of the alignments of any of the other reads spanning the same InDel and the distribution of reads to those different possible alignments is essentially random. Figure 5.4 illustrates the kind of artefact that can arise from read alignments like this. The InDel-realignment was performed using the Genome Analysis Toolkit (GATK – McKenna et al. (2010); DePristo et al. (2011)) released by the Broad Institute (version `2.5-2-gf57256b`). Specifically two tools were used to perform the InDel-realignment; the `RealignerTargetCreator` and the `IndelRealigner`, both are part of the GATK. In a first step I defined a set of genomic intervals in which realignment has to occur. This was done using the `RealignerTargetCreator` tool, which takes a set of BAM files as input and generates a text file containing genomic intervals for realignment. Here it was important to use all BAM files of the whole cohort as input to ensure that the realignment happens in all intervals and samples. This allows for all samples to be compared to each other in the later analysis, i.e. I wanted to make sure that all samples will behave in the same way around those InDels that the `RealignerTargetCreator` has identified for realignment. If each sample is realigned separately a situation can occur where there are two valid alignment states around an InDel and the samples end up in the two different states, which can create artefacts that look like SNVs as well.

In the second step I used the `IndelRealigner` tool to perform the actual realignment. This tool iterates through the previously identified genomic intervals, loads all reads overlapping each of them and aligns all reads together to make sure that eventual ambiguities are resolved in the same way in all read alignments. To ensure comparability between all samples I applied this tool to the whole cohort simultaneously so that all reads in all samples of the cohort will be ensured to behave in the same way when spanning InDels that allow for multi-

[2] `http://picard.sourceforge.net`

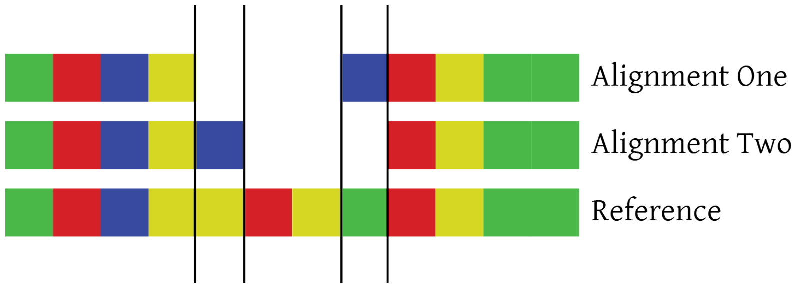


Figure 5.4: Example of a small deletion overlapping a SNV where there are two possible alignment states over the region which each have the same score (1 mismatch and 3 deleted bases). When the aligner assigns the state randomly for each read we will end up with what looks like two SNVs but can actually be explained by using just one SNV.

ple equally well scoring alignment states. Note that this requirement could be relaxed to the patient level (i.e. running the `IndelRealigner` on all samples from the same patient simultaneously) since the most likely scenario of sample comparisons in downstream analyses is that of comparing different samples from the same patient with each other (e.g. a tumour and a matched control sample). In larger cohorts of WGS samples (i.e. the WAS-ALL dataset) I used the patient-level for indel realignment to reduce runtime. At this point the alignments were fully preprocessed and ready to be used for genomics analysis.

5.2.3 Creating HDF5-based nucleotide tallies

I used the framework provided by **`h5vc`** for most of the subsequent analyses. I created nucleotide tallies stored in HDF5 files using the functionality provided by **`h5vc`**, specifically the `batchTallies` function which I implemented as part of **`h5vc`** to allow me to generate HDF5 tally files quickly for large cohorts using the EBI compute cluster. This function splits the genome into bins (1 megabase default size) and calculates the nucleotide tallies and auxiliary datasets (i.e. the datasets `Counts`, `Coverages`, `Deletions` and `Reference`) for each sample in each bin. The calculations were performed in parallel by using the functionality for interacting with high performance compute clusters that is provided by the **`BatchJobs`** R/Bioconductor package. A master process handles the collection of computation results per bin and sample and writes them to the respective HDF5 tally file. See Section 4.2 and Figure 4.2 for details about this process. It is important to note that the nucleotide tallies for the WAS data were created with a version of **`h5vc`** that used slightly different functions than the most recent version does. This only influences the time consumption of this step and the resulting tallies are not different from the set created with the most current implementation.

After this introduction to the common steps in the analysis of cancer sequencing cohort, I will now discuss methods and results for each project separately.

5.3 Genomic characterization of childhood T-ALLs induced by LMO2 over-expression

I will now discuss my methods and findings in the analysis of the WAS T-ALL dataset, i.e. the samples from patients that had developed *LMO2*-induced T-ALLs as a side effect of the HSC gene therapy used as a treatment of their WAS. The samples are pairs of tumour and matched controls that were sequenced with a whole genome approach. The aim of this study was to characterise genomic events in the cancer samples as potential second hits (in addition the *LMO2* over-expression) in the progression towards leukaemia.

5.3.1 Methods

5.3.1.1 Copy number analysis

I performed a copy number analysis of the WGS data of the T-ALL samples. I used the `binnedCoverage` function that I implemented as part of the **`h5vc`** package in conjunction with the `h5dapply` function (see Section 4.4 for details) for the parallel computation of the coverage in bins of size 10 kilobases. The `binnedCoverage` function returns a table that contains for each sample the total number of aligned bases per bin and also calculates the number of G or C bases in the reference sequence in that bin. This data I used to fit a dependency between coverage and GC-count of each bin, which is used to adjust for the influence of GC-count on the coverage. It is important to perform this step to make the samples more comparable for the downstream analysis to identify copy number changes. The algorithm I used here is essentially identical to the approach I originally developed for use in the HeLa sequencing project, see also Section 3.2.1 and Figure 3.1. I did not perform a mappability analysis for the WAS cohort, since I did not expect such a step to improve the overall quality of the CN analysis. The crucial points of the analysis are the calculation of coverage and GC-content in 10-kb bins and the fitting of the relationship between coverage and GC-content. The fit is used to estimate a correction factor to adjust for this bias in coverage which can be different from sample to sample. The adjusted coverages of the samples should all exhibit the same basic relationship between coverage and GC-count in the bins.

For a given sample s I define the coverage c^s (where c^s_i represents the coverage of the i-th bin in sample s) and the GC-content gc (gc_i for the i-th bin). A local regression is fit using the `locfit.robust` function proved by R, which implements a robust local regression similar to the lowess method (Cleveland, 1981). Here a low-degree polynomial is fitted to a neighbourhood of each point in the data where the influence of the neighbours is weighted by their distance to the point the fit is being calculated for. Additionally the fit is made robust by down-weighting neighbours of the point based on their distance to the fitted polynomial in an iterative fashion. The result is the value of the fitted polynomial at each of the datapoints.

The fit f is performed on the relationship between c^s and gc. I used the default parameters for the R function to perform the fit. The adjustment factor a_i for the i-th bin is defined as the ratio between the predicted coverage and a desired coverage dc^s chosen by the user. Since all downstream analysis of adjusted coverage values is always done by comparing the values of different regions or different samples (when searching for gains and losses) it is in fact not necessary to base the choice of desired coverage on the input data. I chose a desired coverage of 50 since that roughly reflects the coverage that the sequencing experiments were aiming for, although arbitrary other choices would be equally as valid. Another effect of choosing the same fixed desired coverage for all samples is that the values become directly comparable and library size effects are eliminated. This works well if the samples largely have the same copy number, i.e. if they are mostly diploid. If one of the samples were largely triploid then the fit (which detects the biggest population of datapoints) will still make sure the median adjusted coverage is 50 but that would then correspond to the triploid state in the sample. I assumed that the samples I analysed here were still mostly diploid and there should be no problems with the approach.

The adjusted coverage ac is then given as $ac_i^s = c_i^s \cdot a_i$ and the effect of this adjustment is illustrated in Figure 5.5 which shows the relationship between coverage and GC-count (number of G or C bases within each 10KB bin) using a smoothed scatter plot.

A similar approach to GC-content correction is also implemented in the **`HMMCopy`** package, which is available through Bioconductor (Lai et al., 2012).

5.3.1.2 GC-bias as quality control

Creating smoothed scatter plots of the relation between coverage and GC-content for all samples can help with spotting problems that might not have been apparent in the preceding quality control steps. A good example of this is the behaviour of the primary tumour sample of patient 7 (labelled *WAS7PrimaryDNA*). In the smoothed scatter plot (shown in Panel **(a)** of Figure 5.6) we can see a very strong correlation between coverage and GC-count of the bins which is coupled with a noisiness of the data that leads to a complete lack of the features typically associated with haploid regions in those plots, i.e. two parallel clouds of points corresponding to diploid and haploid regions, respectively. In this severe case the fitting and the adjustment algorithm do not work and are unable to remove the GC-bias in the data. Based on my observation of the noisiness and generally bad quality of the data, this specific sample was sequenced a second time. The smoothed scatter plots of that sequencing run are shown in Panel **(b)** of Figure 5.6. The re-sequencing of the sample fixed the problem to some extent and did not exhibit such an extreme GC-bias as the initial sequencing run. However, it is still a bit more noisy than other samples in the cohort.

Similar observations of noisiness in the initial sequencing runs of the samples from patient 1 lead to the decision to re-sequence those samples as well.

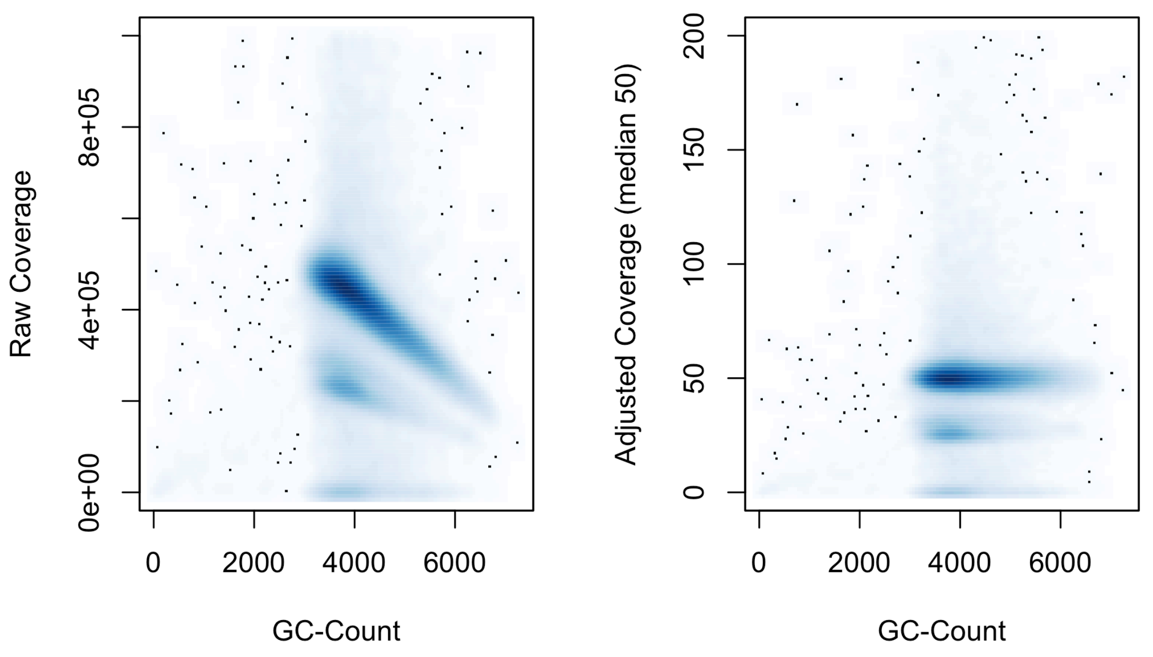
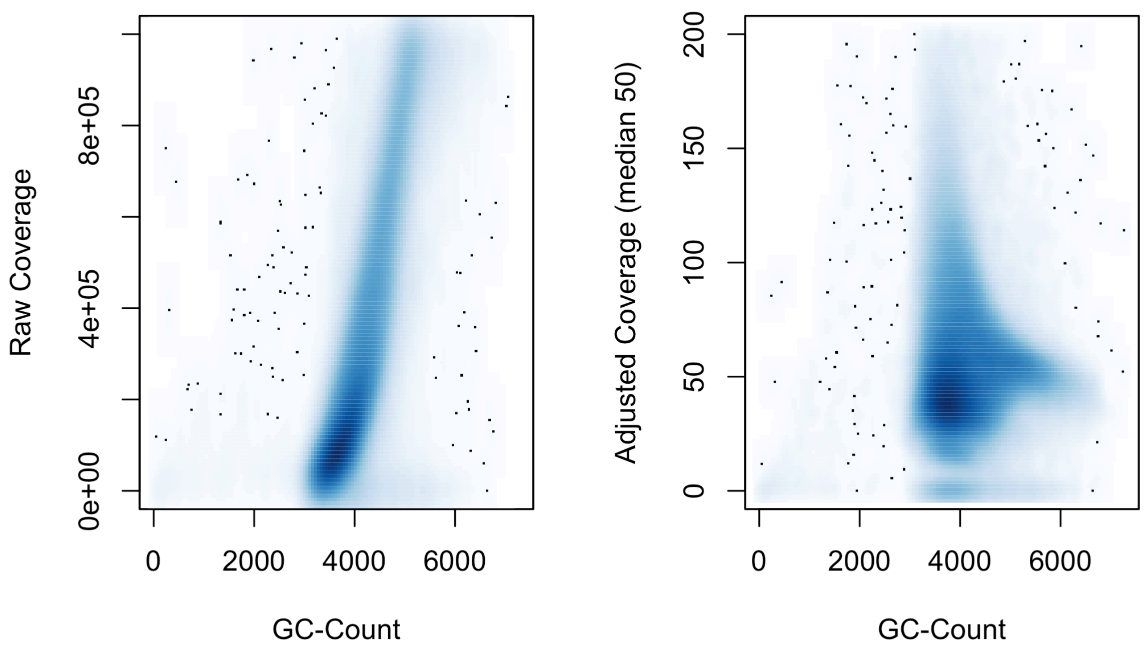


Figure 5.5: Smoothed scatter-plots illustrating the effect of adjusting the coverage for GC-bias. The input data is shown on the left, the result of the adjustment on the right. Note that in the input data the coverage is defined as total number of aligned bases in a bin of size 10KB leading to significantly larger numbers. The adjustment is made with a target value of 50 which can essentially be chosen freely. If it is chosen too low there might be issues with discretisation when comparing values between samples or regions. Note that each of the plots shows two clouds of points, corresponding to the diploid (autosomes) and haploid (allosome) regions of the genome. All samples in this cohort exhibit such patterns since all patients are male.

(a)



(b)

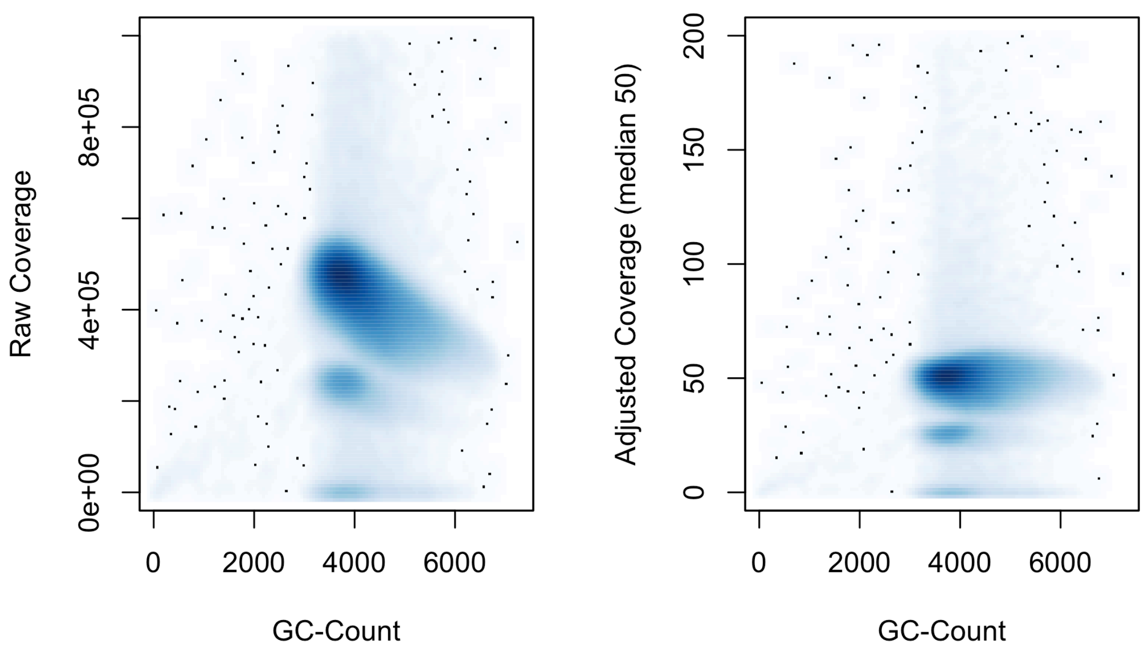


Figure 5.6: Smoothed scatter plots of the initial and re-sequencing run for the tumour sample of patient 7. **(a)** Initial sequencing run with the left plot showing the raw data and the right plot showing the adjusted data. **(b)** Plots for the re-sequencing of the sample. Note the extreme GC-bias in the raw data of the initial sequencing run, which can not be adjusted for through the applied algorithm. The re-sequencing alleviates the problems slightly, allowing for a better removal of GC-bias while still remaining slightly more noisy than other samples in the cohort (compare Figure 5.5)

5.3.1.3 Calculation of $\log_2$-ratios between paired samples

The next step was the calculation of the $\log_2$-ratios of adjusted coverages in the 10kb bins for each pair of samples. This was performed for all pairs of case and control samples, e.g. for patient 5 the two pairs primary vs. control and relapse vs. control were analysed (see Table 5.1). The $\log_2$-ratios were used to estimate gains and losses in the case samples relative to their matched controls. For this purpose the $\log_2$-ratio (r^{ct}) of adjusted coverages of two samples c (the control sample) and t (the tumour sample) is calculated according to the following formula:

$$r_i^{ct} = \log_2(\frac{ac_i^t}{ac_i^c})$$

where r_i^{ct} is the $\log_2$-ratio of adjusted coverages of samples c and t in the i-th bin and ac_i^t is the adjusted coverage of sample t in the i-th bin. To compensate for differences in library size r^{ct} is centered on 0 by subtraction of the median of r^{ct} from the ratios of all the bins. When using a fixed desired coverage for all samples as I did, this step is not necessary, since the median of the $\log_2$-ratios should be 0 anyway.

$$r^{ct} = r^{ct} - \tilde{r}^{ct}$$

The resulting track of GC-adjusted $\log_2$-ratios was segmented using the **`DNAcopy`** package in the same way as I implemented for the HeLa genome project (see Section 3.2.1). The adjusted coverage data was first smoothed using the `smooth.CNA` function provided by the **`DNAcopy`** package, which implements outlier detection and smoothing using circular binary segmentation (CBS; Olshen et al. (2004)). The **`DNAcopy`** package was originally developed in the microarray era and works on probe intensities from e.g. tiling arrays (Mockler et al. (2005)). The algorithmic approach itself is not dependent on the data source and therefore coverage estimates from aligned reads per bin can be easily substituted as inputs. Figure 5.7 shows the $\log_2$-ratios for chromosome 6 of three sample-pairs from the cohort. We can see a large (potentially polyclonal) deletion in the tumour sample of patient 6 and the difference between the initial and re-sequencing of the tumour sample of patient 7 which does not carry large structural aberrations on this chromosome.

When performing a copy number analysis it is important to consider potential contamination of the tumour sample with control material as well as the possibility of a polyclonal population of cells in the cancer sample. Table 5.3 lists the tumour purity for the samples in this cohort as determined prior to sequencing.

5.3.1.4 Comparative SNV calling

I called somatic SNVs in the samples from the WAS T-ALL cohort on the HDF5-based nucleotide tallies I created for this cohort. I used the `callVariantsPaired` function to perform

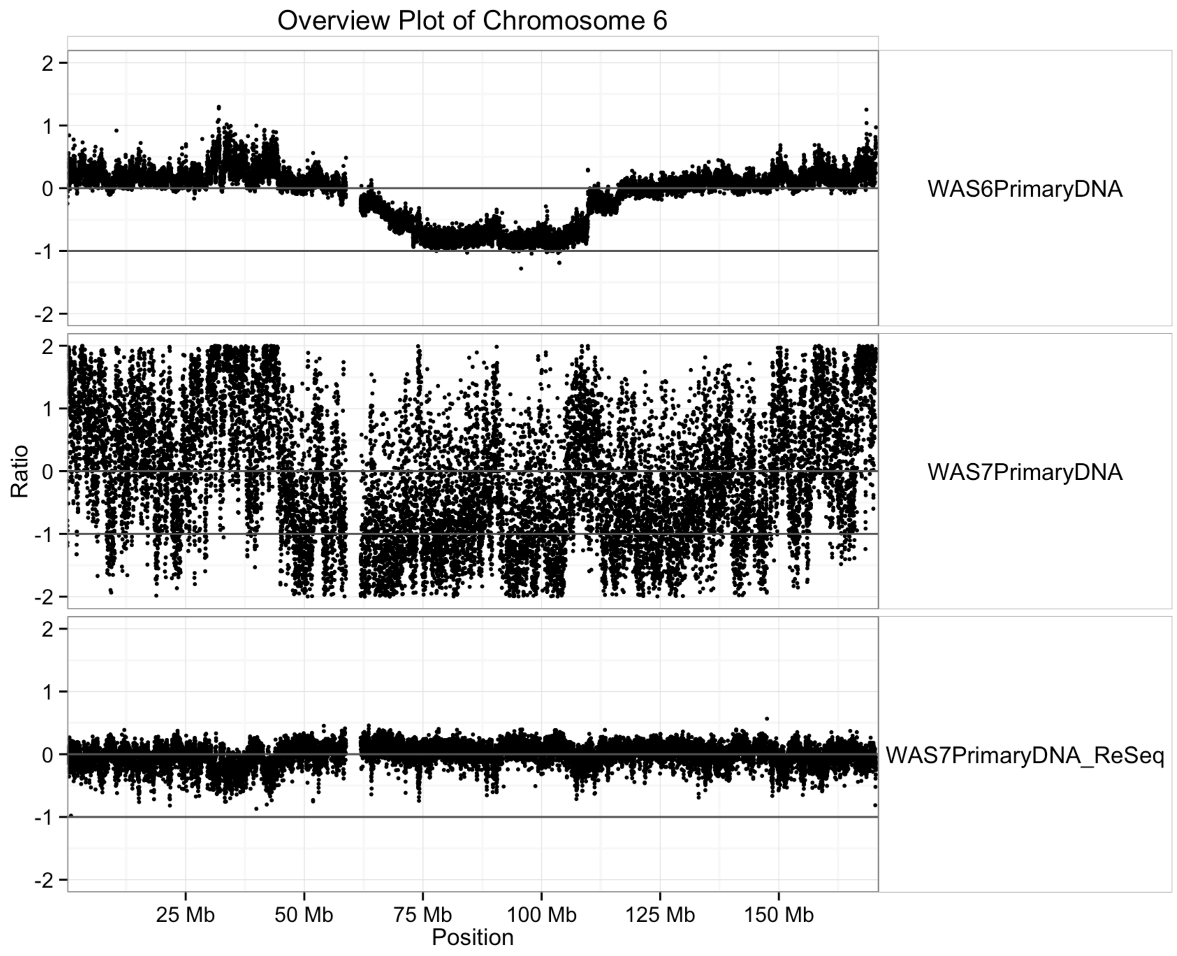


Figure 5.7: Overview plot of chromosome 6 showing the $\log_2$-ratios of GC-adjusted coverages in 10 kilobase bins for 3 sample pairs from the WAS T-ALL cohort. The x-axis shows the genomic position and the y-axis the $\log_2$-ratio where a value of e.g. -1 corresponds to a reduction of coverage by 50%, i.e a loss of one copy of the respective chromosomal region. The three panels are labeled with the sample identifiers of their respective tumour sample. The top panel (labeled *WAS6PrimaryDNA*) shows the $\log_2$-ratio of tumour and control of patient 6, where a large deletion is present at a frequency of $\sim$ 50% in the tumour sample. The two lower panels serve to illustrate the effect of the extreme GC-bias observed in the initial sequencing run of patient 7 (see also Figure 5.6). The results of the copy number analysis of this cohort are discussed in detail in Section 5.3.2.1.

somatic variant calling on tumour-normal pairs. See Section 4.6.3 for a description of the variant calling algorithm. I used the default settings for the variant caller, which require the following properties for a position to be called a variants:

Sample	Property	Limits	Description
Case	Coverage (c)	$5 \leq c \leq 200$	Total number of reads overlapping the position
Case	Support (s)	$s \geq 2$	Number of reads supporting the alternative allele
Control	Coverage (c)	$5 \leq c \leq 200$	Total number of reads overlapping the position
Control	Support (s)	$s = 0$	Number of reads supporting the alternative allele

Note that these requirements need to be fulfilled on both strands separately. The default values make sure that a variant has at least 2 reads supporting it on each strand of the tumour sample (the *Case* sample) and no supporting reads in the matched control sample. Furthermore I limit the range of coverages in which to attempt calling to have at least 5 reads on each strand so we have enough power to even detect a variant and no more than 200 reads per strand to exclude repetitive regions such as pseudogenes or transposons. The upper limit on the coverage serves to reduce the number of artefacts caused by reads from a similar but not exactly equal genomic region being falsely mapped (with a few mismatches) to a homologous region of their origin. This can happen if the alignment program is ill-equipped to deal with sequence homology or if the reference sequence actually does not contain the homologous region.

I generated a set of *raw* somatic variant calls for each pair of samples using the function and parameters described above. I implemented the calling with the use of the **`BatchJobs`** R package to heavily parallelise the computations on a compute cluster, using one compute node per mega-base and effectively running ~ 3000 processes at the same time. This parallelisation allows for the genome-wide calling of somatic variants in the whole cohort in only a few hours.

The raw variant calls were then filtered more stringently to extract a set of high-confidence calls and further limited to such variants that were predicted to overlap with coding regions and therefore constitute non-synonymous coding SNVs. I used the Ensembl Variant Effect Predictor (EVP) tool (McLaren et al., 2010) to make those predictions.

5.3.1.5 Single Sample Variant Calling

I used the `callVariantsSingle` function from **`h5vc`** on the HDF5-based nucleotide tallies of the WAS T-ALL cohort to create a set of single sample variant calls. I configured the variant calling function to require at least 6 reads supporting a variant (at least 3 on each strand) as well as a minimum allelic frequency of more than 15% to generate a first (*raw*) set of single sample variant calls. From this *raw* set of SNV calls I created a filtered set of variant calls, requiring an allelic frequency of at least 20% and a coverage of at least 20 reads. This filtered call-set was used in further analyses of the single sample variant calls.

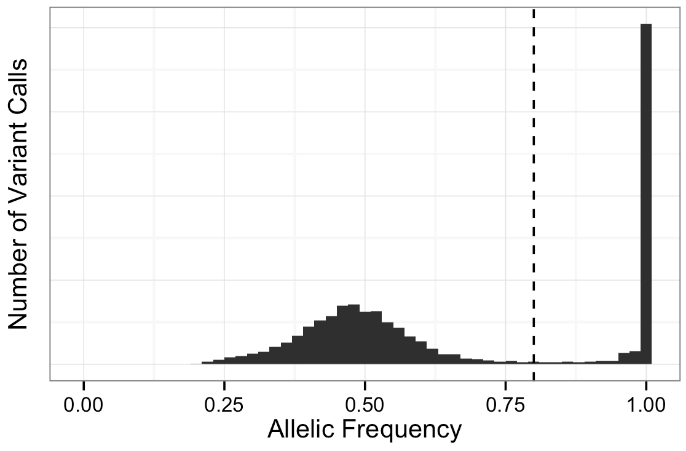


Figure 5.8: Histogram of the observed allelic frequencies in single sample variant calls from all samples in the cohort. The cut-off for defining a variant as homozygous was chosen at 80% (vertical black line). Note that one cannot simply define homozygous variants by their allelic frequency being 100% since the purity of the tumour sample can not be guaranteed (see also Table 5.3).

5.3.1.6 Allele frequency profile analysis

In order to detect regions of abnormal copy number as well as CN-neutral structural variants that cause LoH I profiled the distributions of observable allelic frequencies in bins along the genome. I performed single sample variant calling on all the samples in the cohort using relatively stringent filters to avoid noise (i.e. low-frequency artefacts and sequencing errors) to influence the signal. I required an allelic frequency of at least 20% and a coverage of at least 20 reads for a position to be considered a variant (see Section 5.3.1.5). I visualised the variant calls by plotting the observed allele frequencies along the genome by subsampling the variant calls down to ~ 1 call per ten kilo-bases to reduce plot density (i.e to avoid over-plotting) and improve performance of the plotting function. This allowed for visual inspection for large regions of interest. I also developed a classifier analogous to the LoH classifier used in the HeLa genome sequencing project, by first defining all variants with allelic frequencies larger than 80% as potentially homozygous (informed by the distribution of allelic frequencies, see Figure 5.8) and then calculating the proportion of potentially homozygous variant calls over all variant calls per mega-base along the genome. Since all patients are male I focused this analysis on the autosomes only. I used the segmentation algorithm for LoH data provided by the `DNAcopy` package to define a segmentation of those 1 mega-base bins according to the percentage of homozygous calls. The histogram of the mean percentages of homozygous calls per segment is shown in Figure 5.9 and a cut-off for defining a segment as mostly homozygous is derived from this histogram. I defined any segment with a mean percentage of homozygous calls larger than 60% to be a homozygous segment. Based in this classifier I could detect homozygous regions at a resolution of 1 mega-base in the cohort. This helped to identify copy-number neutral LoH events.

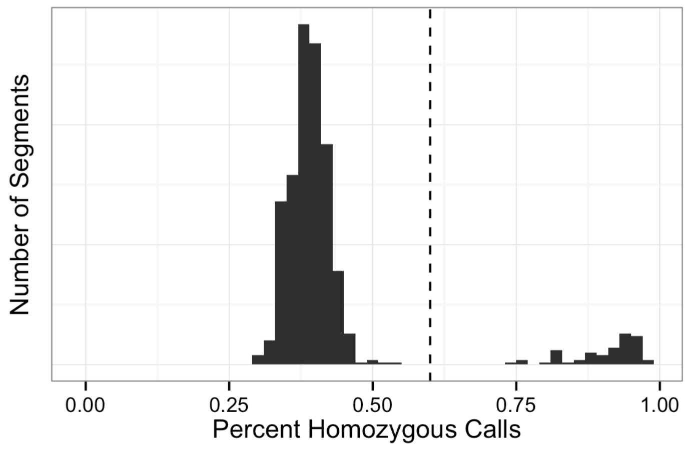


Figure 5.9: Histogram of the percentage of homozygous variant calls per 1 mega-base bin along the genome. Data is shown pooled from all samples in the cohort. Guided by this histogram a cut-off for defining mostly homozygous regions was chosen at 60% (dashed black vertical line).

5.3.2 Results

5.3.2.1 Results of copy number analysis

The copy number analysis (Section 5.3.1.1) was performed on bins of size 10 kilobases along the genome for all samples that were analysed with whole genome sequencing. After adjusting the coverage for GC-bias I calculated pairwise $\log_2$-ratios for all pairs of samples (tumour and matched control) and segmented this signal using **`DNAcopy`** (`R/Bioconductor` package). The sample pairs will always be identified by the sample identifier of the tumour sample, since for some patients multiple tumour samples were compared to the same control sample, e.g. in patient 5 I compared primary tumour and relapse tumour each with the same control sample.

Figure 5.10 shows histogram of the mean $\log_2$-ratios of the segments for all tumour samples. We can see for both cases where re-sequencing was done, how the signal was more noisy in the initial sequencing run, much more so in the initial sample of patient 7 than patient 1 (labeled as *WAS1PrimaryDNAFirst*).

I observed large structural variants in this cohort which were present in samples from multiple patients. One of those is a gain on the long arm of chromosome 17 in the samples of patient 1 and 5, which is shown in Figure 5.11. This copy number gain on chromosome 17 has been previously described, e.g. in Strefford et al. (2007) and can therefore be considered a typical event in T-ALLs and not a specific symptom of the vector-integrations caused by the retroviral gene therapy that the patients were treated with.

5.3.2.2 Comparative SNV calling – Results

In total I called 270829 *raw* SNV calls with the somatic variant calling approach described Section 5.3.1.4. After filtering out variant calls with allelic frequencies smaller than 10% a total of 89608 variants remained, that were distributed among the tumour samples. An overview

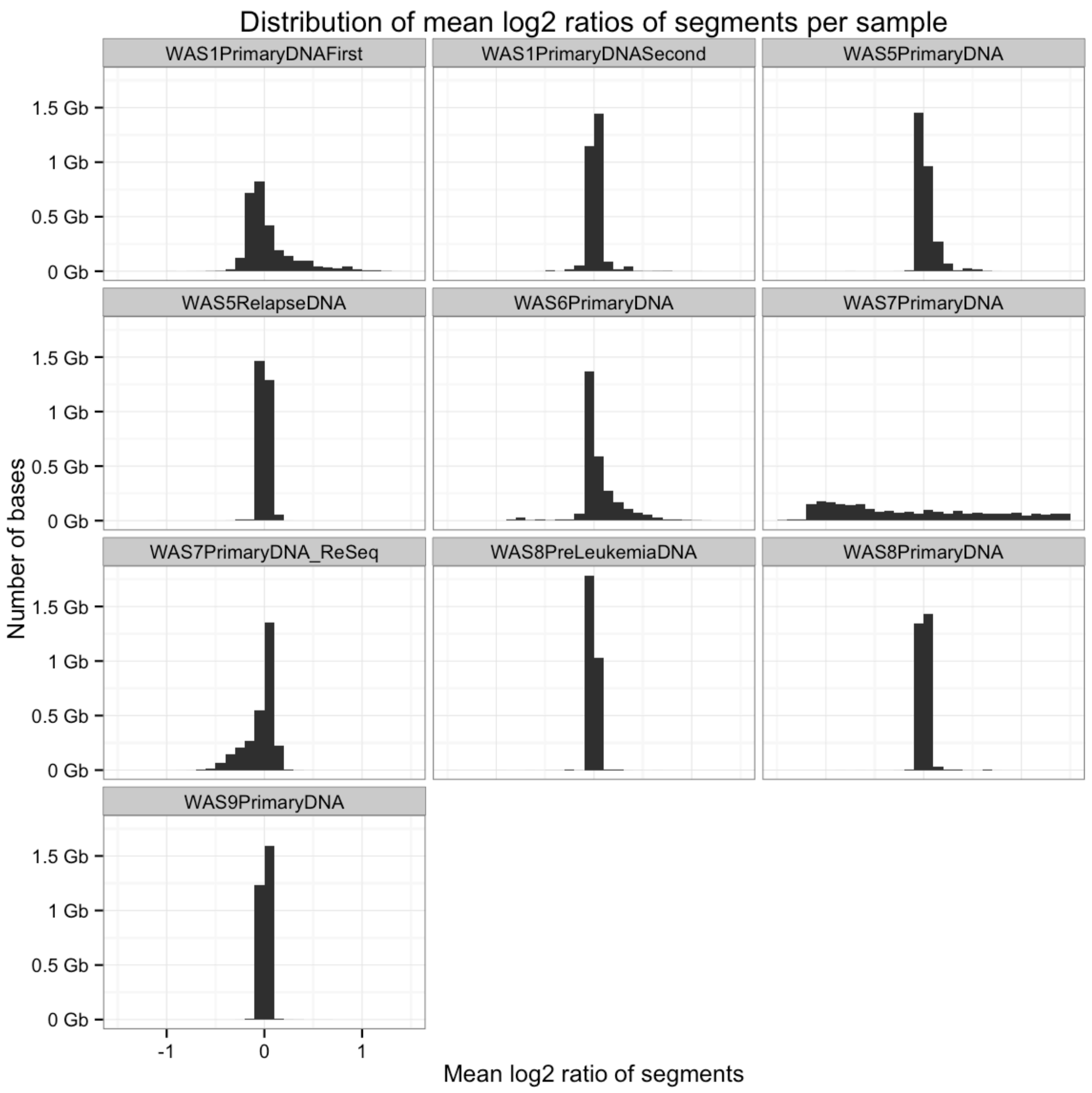


Figure 5.10: Histograms of the segmentations of $\log_2$-ratios in all samples from the WAS cohort that were sequenced with WGS. (Note that *WAS9PrimaryDNA* is an AML sample, while all others are T-ALL samples). The x-axis shows the mean $\log_2$-ratios of the segments and the y-axis the total amount of bases on the genome with that $\log_2$-ratio. It is apparent that the initial sequencing runs of both patient 7 (*WAS7PrimaryDNA*) and patient 1 (*WAS1PrimaryDNAFirst*) are much more noisy than their respective re-sequencings (*WAS7PrimaryDNA_ReSeq* and *WAS1PrimaryDNASecond*). Apart from this noisiness in the initial sequencing runs of low-quality most samples seem rather quiet and do not show extreme copy number aberrations (compared to e.g. the HeLa data shown in Figure 3.5)

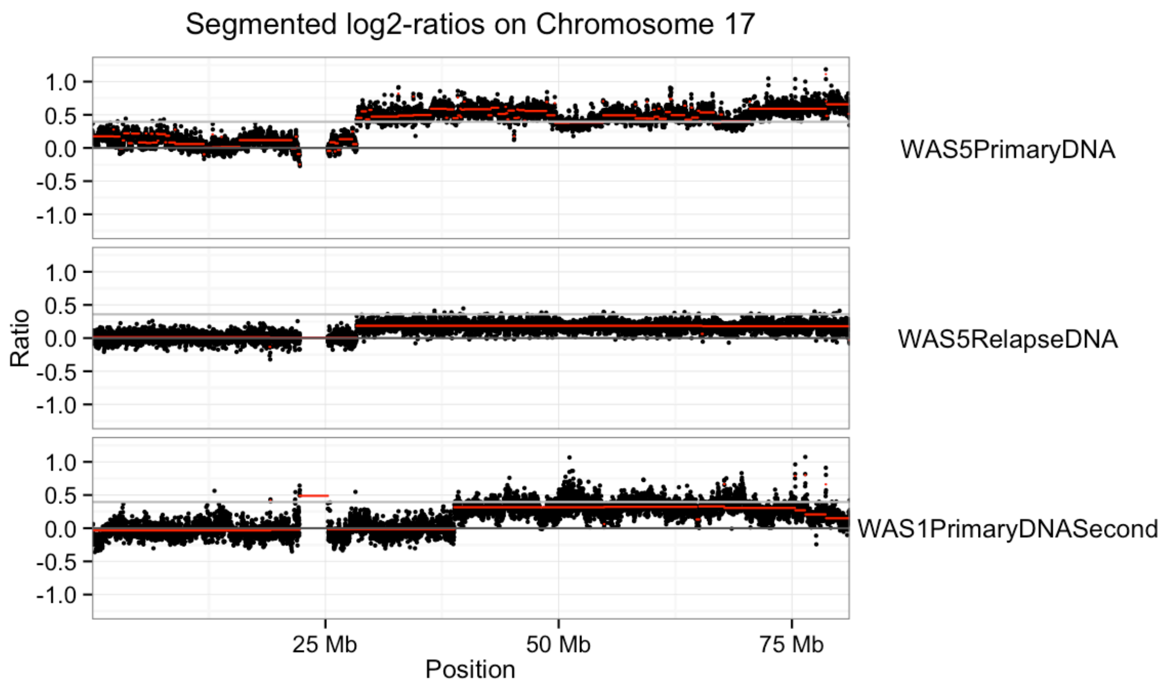


Figure 5.11: Overview plot of chromosome 17 showing the segmented $\log_2$-ratios of the tumour samples of patient 1 and 5 (primary tumour and relapse sample). Genomic position is shown on the x-axis and $\log_2$-ratios on the y-axis. Black dots represent 10kb bins and red lines show the segmentation of those bins according to their $\log_2$-ratios. The primary and relapse tumour sample of patient 5 are shown in the two top panels and the primary tumour sample of patient 1 is shown below. Note the common gain of one copy of the long arm of chromosome 17. The different purities of the tumour samples lead to different values for the segmentation which are slightly smaller than the expected $\log_2(3/2) = 0.58$. Given the observed purity of the tumour samples (56% for patient 1 and 63% for patient 5 – see Table 5.3) the expected values for heterozygous events can be calculated and are shown as grey horizontal lines in the plots. In patient 1 the observed segment very closely follows this expected value whereas in the primary tumour of patient 5 the signal is more noisy and generally a bit higher. This can be the result of imprecise measurements of peripheral blast percentage or an effect of the noise. Additionally it seems that the gain is polyclonal in the relapse sample of patient 5 which could be explained by the gain being present in the primary tumour sample and the relapse sample being composed of a small fraction of resistant primary tumour cells and a larger fraction of a new clone.

Sample	Raw	Filtered	Coding
WAS1PrimaryDNAFirst	89807	21363	126
WAS1PrimaryDNASecond	45655	14456	58
WAS5PrimaryDNA	23677	7108	30
WAS5RelapseDNA	5289	3726	9
WAS6PrimaryDNA	34628	17091	48
WAS7PrimaryDNA	17174	7285	25
WAS7PrimaryDNA_ReSeq	23523	6462	24
WAS8PreLeukemiaDNA	5896	4205	20
WAS8PrimaryDNA	25180	7912	42
	270829	89608	382

Table 5.6: Table showing the number of somatic SNVs called in the T-ALL samples from the WAS cohort. The *raw*, filtered (AF $\geq$ 10%) as well as coding SNV call counts are shown. Note that this table includes the initial (low quality) sequencing runs of patients 1 and 7.

of the distribution of somatic SNV calls among the samples is shown in Table 5.6. For further analysis I excluded the initial sequencing runs of the tumour samples from patients 1 and 7, further reducing the total number of variant calls to 163848 *raw* and 60960 filtered variant calls, respectively. After filtering and exclusion of initial (low quality) sequencing runs a total of 231 coding somatic variants were left.

To answer the question of possible second hits after the initial vector integration and subsequent over-expression of *LMO2* I focused on early events in the analysis of the SNV calls. To this end I filtered the variant calls further to exclude events with less than 30% allelic frequency. A total of 59 coding somatic variants with sufficiently high allelic frequency to be likely early events and therefore likely driver mutations of the respective cancer samples remained after this filtering step. Amongst the genes affected by those 59 variant calls where *FBXW7* (Park et al., 2009), *PTEN* and *LEF1*, which have been implicated in the context of cancer development or progression. I will discuss these findings in more detail in Section 5.3.3.1.

5.3.2.3 Single Sample Variant Calling – Results

After the filtering of the *raw* variant calls a total of $\sim$ 40 million positions remained (excluding the initial sequencing runs of samples from patients 1 and 7).

This set of variant calls was generated to enable the analysis of profiles of allelic frequencies along the genome. I did not investigate the single variant calls separately, but rather used them to make high-level predictions about shifts in the distributions of observable allelic frequencies in genomic regions. Such shifts occur as a side effect of structural variants and can help with the identification of structural variants, especially when they are copy-number neutral and would not be detected by analysis of e.g. the $\log_2$-ratios of coverages between tumour and control samples.

Sample	Homozygous Regions (Mb)
WAS1PrimaryDNAFirst	27
WAS1PrimaryDNASecond	27
WAS5ControlDNA	180
WAS5PrimaryDNA	218
WAS5RelapseDNA	172
WAS6ControlDNA	7
WAS6PrimaryDNA	7
WAS8PrimaryDNA	35

Table 5.7: Table showing for those samples that have at least one bin (of size 1 mega-base) classified as homozygous the total length of homozygous regions in mega-bases. Note that for the tumour sample of patient 1 two entries exist because this sample was sequenced twice. The common LoH event on chromosome 9 has an approximate length between 27 (patient 1) and 35 (patient 8) mega-bases. Note that all samples of patient 5 show a remarkably high number of homozygous regions, indicating that this patients germ-line genome is homozygous in regions of about 180 mega-bases length.

5.3.2.4 Allele frequency profile analysis – Results

Analysis of the patterns of observable allelic frequencies revealed a common copy-number neutral LoH event on the short arm of chromosome 9 in the primary tumour samples of patients 1, 5 and 8. The segmented $\log_2$-ratios of Chromosome 9 do not show indications of large structural aberrations on that chromosome and are shown in Figure 5.14. The common LoH event is visualised in Figure 5.15 where I plot the observed allelic frequencies of single nucleotide variant calls in the tumour samples of patients 1, 5, 6, 7, 8 and 9. The results of the binary classifier applied to bins of size 1 mega-base is shown in Figure 5.12, where I plot the percentage of homozygous calls for each of the bins on chromosome 9 as well as an overlay of the segmentation into homozygous and heterozygous regions. Table 5.7 shows the total length of genomic regions that were classified as homozygous for sample where at least one bin was classified as homozygous (samples not present in this table have no large homozygous regions). It is apparent that patient 5 has large homozygous regions present on chromosome 9 in his germ-line sample; whether this fact influenced the progression of his cancer is unknown (Figures 5.15 and 5.12). The copy-number neutral LoH event on the short arm of chromosome 9 however has been described previously (Takeuchi et al., 1997), as well as the homozygous deletion at *9p21* in the tumour sample of patient 8, visualised in panel **(b)** of Figure 5.14). The fact that Takeuchi et al. described those events at 57% and 43% prevalence, respectively, indicates that those features of our cohort are typical aberrations in childhood acute lymphoblastic leukaemia and likely not linked to the fact that the T-ALLs we analysed were caused by vector integrations from retroviral gene therapy. Although a number of large homozygous regions are apparent throughout the genome of the control sample of patient 5, the LoH on the short arm of chromosome 9 is specific to the tumour sample of this patient and likely unrelated to the presence of those germ-line homozygous regions.

Through visual inspection of the observed allelic frequencies of called variants I could also

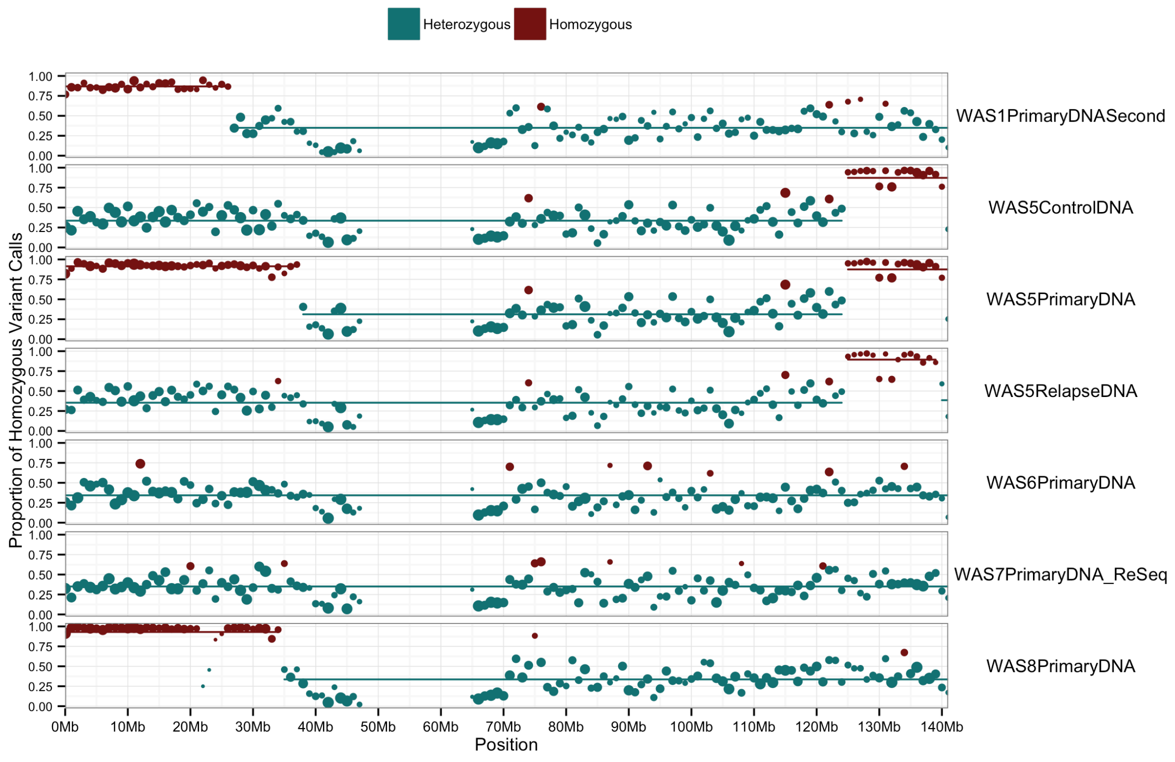


Figure 5.12: Overview of percentage of homozygous variant calls on chromosome 9 for cancer samples from patients 1, 5, 6, 7 and 8 as well as the control sample of patient 5. The x-axis is the genomic position and the y-axis is the percentage of homozygous calls. The dots represent the 1Mb bins along the chromosome and the lines show their segmentation. The colour encodes the classification into heterozygous and homozygous regions and the size of dots shows the number of variant calls in that bin and serves as a measure of the confidence we can have in the classifier. Note the common LoH on the short arm of chromosome 9 shared between the primary tumour samples of patients 1, 5 and 8. Three samples are shown from patient 5 and it is clear that the LoH is only present in the primary tumour sample but not in the relapse tumour or the control. Furthermore, large regions of homozygousity are apparent near the end of the chromosome in all samples from patient 5, including the control sample.

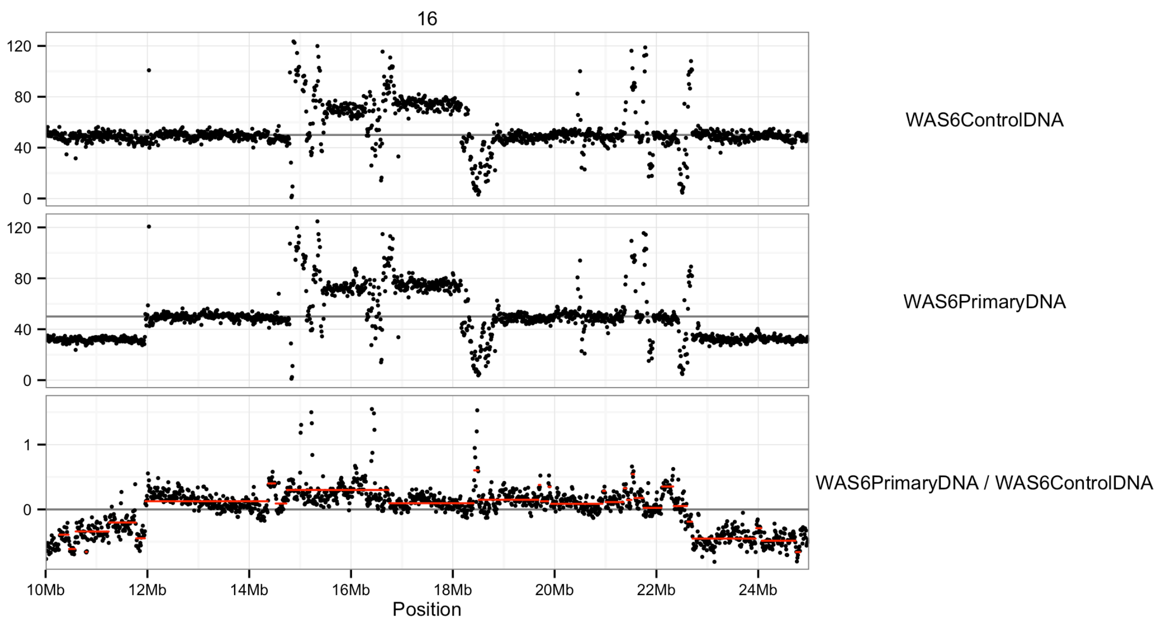


Figure 5.13: Overview plot showing the GC-adjusted coverages of the control and tumour sample of patient 6 in the top two panels. The third panel shows the $\log_2$-ratio and segmentation of those two samples. The region shown lies on the short arm of chromosome 16. A germ-line duplication event spanning from around 15 up to 18 mega-bases is apparent in the top two panels. In the $\log_2$-ratio plot as well as the coverage of the tumour sample two somatic deletion events located some mega-bases up- and down-stream of the duplicated region can be seen. The $\log_2$-ratio overlapping the duplication event does not show the event, however. Note that the ratios and segmentation are a bit offset from 0 (the expected value). This is likely due to noise in the data, since the samples of patient 6 were amongst the more noisy ones in this cohort (see also Figure 5.10).

detect a germ-line duplication of about 5 kilo-base size on chromosome 16 in patient 6. This event was also not apparent from the $\log_2$-ratio plots, since the duplication is already present in the control sample. Figure 5.13 shows the $\log_2$-ratio as well as the GC-count adjusted coverage vectors for the control and tumour sample of that patient which indicate the duplication event.

The set of aberrations I could detect by analysis of allelic frequency distributions illustrates well the usefulness of this approach, since some of those events are silent in the coverage tracks, e.g. the copy number neutral LoH could not be detected by absolute coverage or $\log_2$-ratio of coverages. Furthermore the essentially comparative approach to analysis of cancer genomes fails to detect germ-line events. The analysis of $\log_2$-ratios of coverages between tumour samples and their matched controls as well as the analysis of SNVs based on evidence for their presence in the tumour and absence in the control samples can not spot such copy-number neutral events as the LoH of chromosome *9p* or the duplication on chromosome 16 of patient 6 (shown in Figure 5.13). Therefore the genome-level analysis of the distributions of allelic frequencies and the GC-adjusted coverage for each sample can help to understand large-scale events of this type. For the calling of SNVs however the comparative approach is vastly superior, since the control samples provide a filter against biologically uninteresting

SNVs as well as technical artefacts that occur in all samples independent of their biology.

5.3.3 Discussion

5.3.3.1 Comparative SNV calling – Discussion

When investigating the 59 filtered coding somatic variant calls I could not determine any genes that were affected within more than one of the cancer samples that are relevant to the cancer progression. The only gene mutated in at least two cancer samples is *DUX4L5* (double homeobox 4 like 5 [Homo sapiens (human)] – GeneID: 653545). This gene has not been previously implicated in childhood leukaemia. Despite this I am convinced that this hit is not relevant to the cancer since I have seen hits in multiple other genes of the same family (double homeobox 4) in another set of cancer samples I analysed in a project which deals with the analysis of multiple myeloma sequencing samples (not reported on in this thesis). Furthermore, those two variants have relatively low support and coverage and are present in a region of severe background noise (i.e. the *DUX4L5* gene). A mismatch plot of the region in question is shown in Figure 5.16, which makes it apparent that the region itself does not yield reliable variant calls since the noise levels are very high (probably as a consequence of the large number of homologous genes in the *DUX* family).

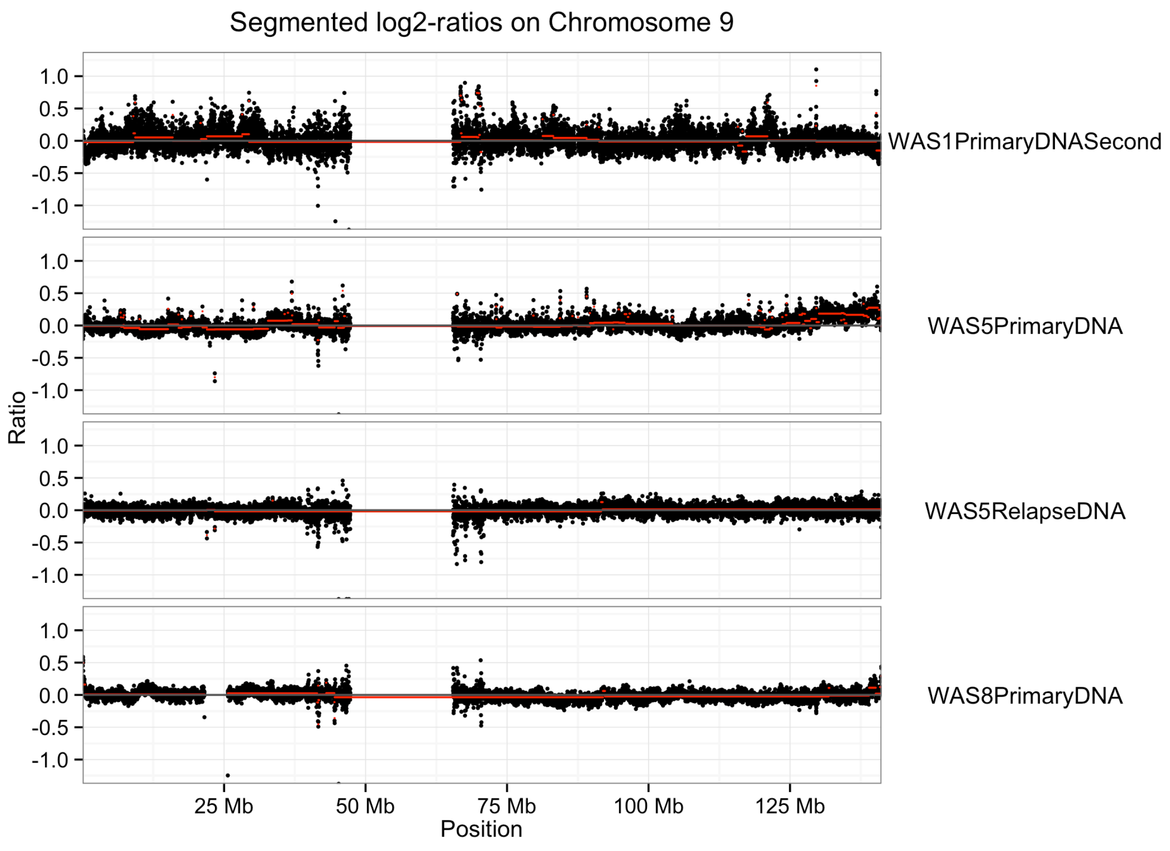
Amongst the remaining somatic variants that are likely early events there are some that affect genes that have been strongly implicated in cancer, such as *FBXW7* (Park et al., 2009), *PTEN* and *LEF1*. When relaxing the requirement for allelic frequency to allow for lower frequency variants (i.e. at least 10%) we can furthermore pick up *TMP53*, *TP53RK*, *PTPN11* and *NOTCH1* (Park et al., 2009). However, those variants are not likely drivers due to their low allelic frequency.

5.3.3.2 Conclusion

The general picture in this cohort is one of an unspecific cancer-like behaviour of the samples without any clear hints at a common second event that occurred after the initial hit (i.e. the vector integration). This second hit was speculated since the time-frames are so vastly different between the patients (between 488 and 1813 days post gene therapy – Table 5.3) and no patient developed their leukaemia straight after gene therapy.

The only common events I could identify were the LoH on the short arm of chromosome 9 (Figure 5.14) and the duplication of the long arm of chromosome 17 (Figure 5.11), while no commonly mutated positions, commonly mutated genes or even commonly mutated pathways could be detected with the set of high-frequency coding somatic variants. While the common large structural aberrations are typical of childhood leukaemia and have been previously described (Strefford et al., 2007; Takeuchi et al., 1997), I do not believe them to be related to the induced *LMO2* over-expression that is a common factor between the T-ALLs analysed in this cohort.

(a)



(b)

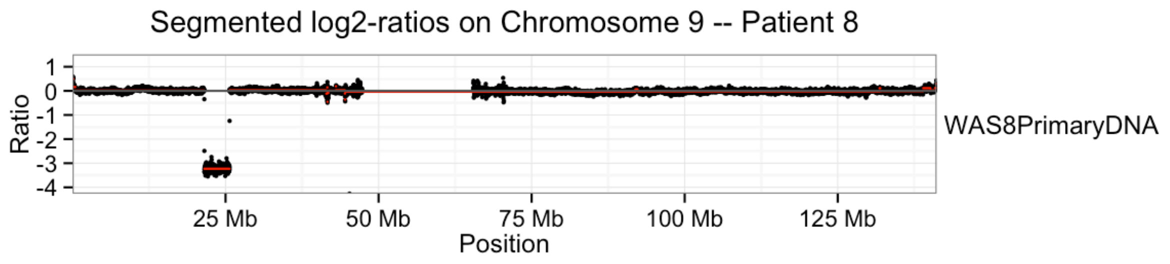


Figure 5.14: Overview of $\log_2$-ratios for tumours with LoH events on chromosome 9. **(a)** Overview of $\log_2$-ratios for all samples affected by the LoH event on chromosome 9. Note that none of the samples show indications of copy number changes (except for a small region in the tumour sample of patient 8). The sample from patient 1 is more noisy than e.g. the samples of patients 5 and 8. **(b)** Tumour sample of patient 8, showing a wider range of $\log_2$-ratios to illustrate a small deletion around 20 to 25 mega-bases. Note that a $\log_2$-ratio of -3.5 (tumour / control) would indicate that the control sample had about 10 times as many copies of that region as the tumour. Considering that this is an extremely unlikely scenario it is clear that this segment is actually completely lost in the tumour sample and the reads that we observe stem from contamination of the tumour sample with the control. This lines up well with the expected tumour purity of 91% (see Table 5.3), since a mixture of 91% tumour with a homozygous deletion and 9% diploid control sample would lead to a theoretical $\log_2$-ratio of -3.47, which is exactly what we observe here ($\log_2(\frac{0.09}{1.00}) = -3.47$).

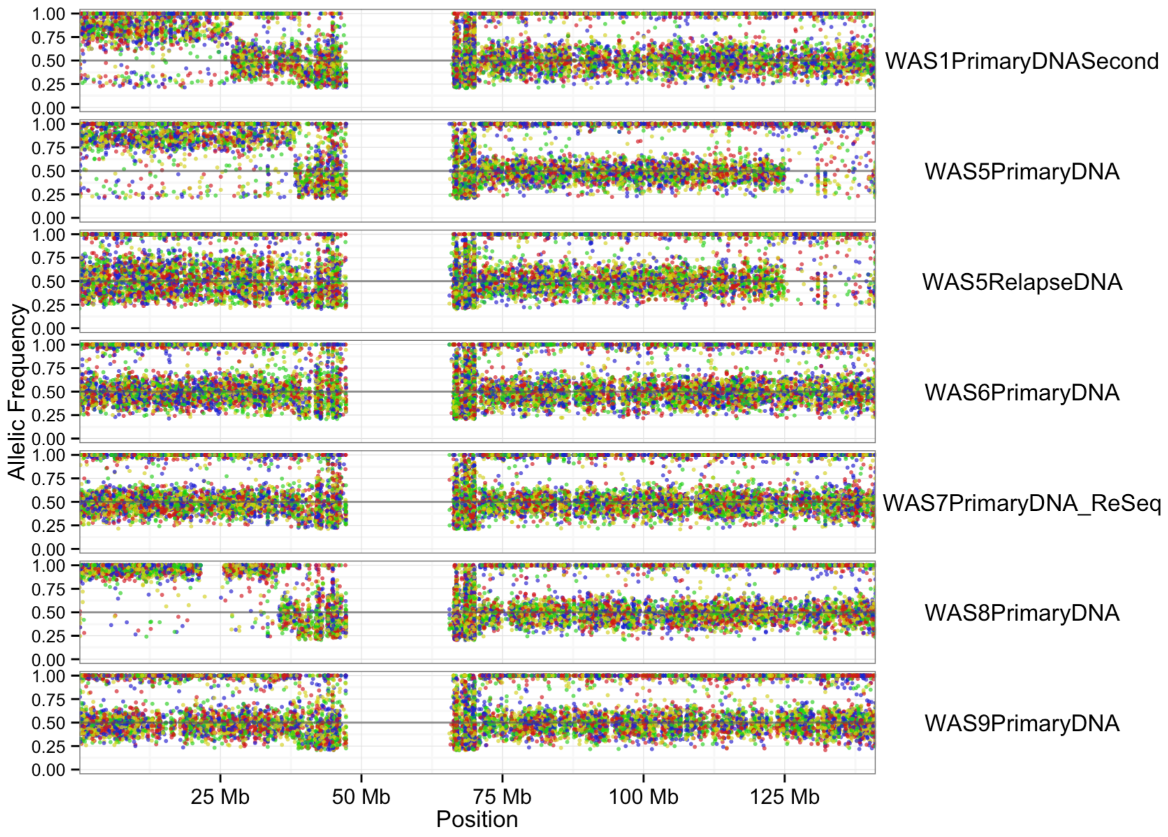


Figure 5.15: Plots showing observed allele frequencies in the tumour samples of patients from the WAS T-ALL cohort. Chromosome 9 is shown with the genomic position on the x-axis and the observed allelic frequency of single nucleotide variant calls on the y-axis. Each dot represents a variant call and is colour coded by the alternative allele (A, C, G and T). In normal diploid human samples most variants will fall either into a band around an allelic frequency of 0.5 or 1.0, representing the heterozygous and homozygous case, respectively. The tumour sample of patient 7 (*WAS7PrimaryDNA_ReSeq*) is a good example where the whole chromosome exhibits the typical pattern of a mixture of homozygous and heterozygous variants. The tumour samples of patients 1, 5 and 8 show a common pattern of LoH on the short arm of chromosome 9, where the middle band (the one around 0.5 representing heterozygous events) is absent. Note that the relapse sample of patient 5 does not show the common LoH, indicating that this relapse is indeed from a different clone than the original tumour. However, we can identify another large region that shows very few homozygous events in samples from this patient (downstream of 125 mega-bases). There are in fact multiple regions like this spread around the genome of patient 5 that can be detected in the germ-line sample as well.

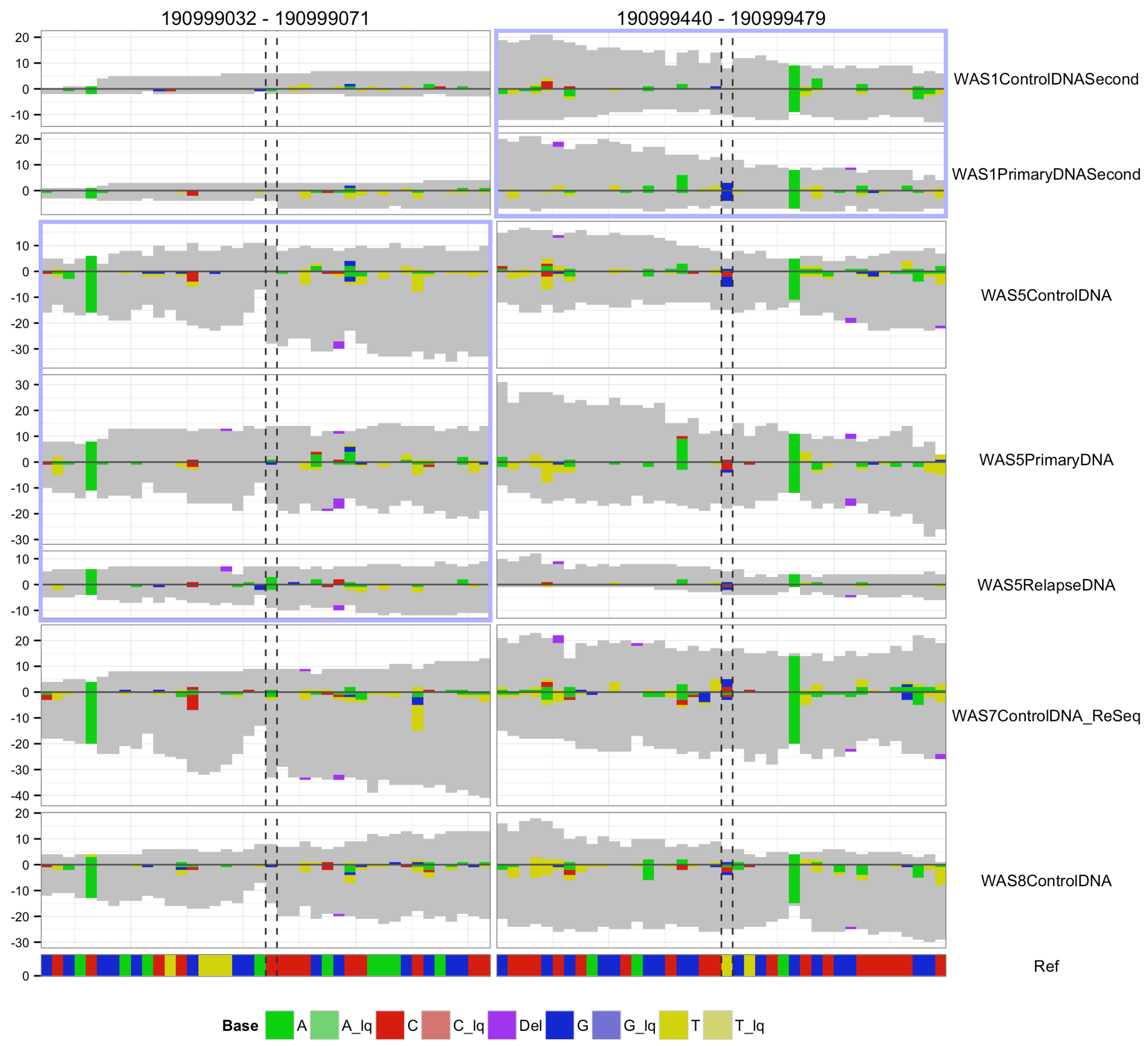


Figure 5.16: Mismatch plot of the region containing the two somatic variant calls overlapping the *DUX4L5* gene in the relapse sample of patient 5 and the primary tumour sample of patient 1, respectively. The left panel shows the region around a somatic variant in the relapse sample of patient 5 located at position `190,999,052` on chromosome 4 and the right panel shows the region around a variant in the tumour sample of patient 1 that is located at position `190,999,460` on the same chromosome. All samples from both patients 1 and 5 are shown and the somatic variant calls are marked with vertical dashed lines. In each panel the samples that are involved in the variant call are marked with a light blue border. To illustrate the general nature of the regions in question a set of unrelated control samples (from patients 7 and 8) are shown. It is apparent that the whole region is very active in terms of mismatches. Furthermore the positions in which somatic variants are detected in the cancer samples from patients 1 and 5 carry a lot of mismatched in the unrelated control samples of patients 7 and 8. Taking all those properties of the regions and variant positions into account leads me to the conclusion that they are biologically not relevant and likely artefacts.

5.4 Genomic characterization of childhood AMLs induced by MDS1 / MN1 over-expression

I will now discuss my methods and findings in the analysis of the WAS AML dataset, i.e. the samples from three patients that had developed *MDS1* or *MN1*-induced AMLs as a side effect of the HSC gene therapy used as a treatment of their WAS. Two of those AMLs were developed after an initial *LMO2*-induced T-ALL had been treated. For one patient the AML was the first leukaemia. The samples are pairs of tumour and matched controls that were sequenced with a whole exome approach. Analogous to the analysis of the T-ALLs, the aim of this study was to characterise genomic events in the cancer samples as potential second hits (in addition the *MN1* / *MDS1* over-expression) in the progression towards leukaemia.

5.4.1 Methods

5.4.1.1 Copy number analysis

Since all AML samples except for the one from patient 9 were sequenced with WES I could not use the same pipeline as I implemented for the copy number analysis in the T-ALL samples of the WAS cohort. When dealing with WES data it is not possible to define fixed-size bins along the genome and determine the coverage and GC-bias in this way (like the 10Kb bins I used for the T-ALL samples; Section 5.3.1.1). This is because the read alignments will only fall onto the coding regions of the genome. As a consequence of the alignments mapping only to coding regions it is non-trivial to determine and adjust for a possible GC-bias in the coverage. There are tools that attempt this and I have experimented with e.g. the **`exomeCopy`** package (Love, 2013), but in general a relationship between coverage and GC-count is hard to establish if the data-points stem from genomic intervals of varying sizes (i.e. the annotated exons in the human reference genome). It is still possible to determine $\log_2$-ratios of coverage and since all samples in this cohort were sequenced on the same flow-cell at the same time and using the same library preparation it is a reasonable assumption that if a GC-bias exists, it will likely affect all samples equally. For comparative analyses, like the detection of gains and losses through $\log_2$-ratios of coverage, the GC-bias would then have a negligible effect.

Therefore, the copy number analysis of the AML samples sequenced with WES is based on the coverage per annotated exon. I used the `h5dapply` and `binnedCoverage` functions since `h5dapply` can work on a list of specific genomic intervals and the list of annotated exons can be easily retrieved using `R`/`Bioconductor` through annotation packages. An example is the **`TxDb.Hsapiens.UCSC.hg19.knownGene`** annotation package (Carlson, 2011). I computed the $\log_2$-ratios for the tumour samples on the raw (unadjusted) coverages directly, followed by segmentation with **`DNAcopy`** and visualisation as chromosome overview plots.

5.4.1.2 Comparative SNV calling

I called somatic SNVs in the samples from WAS patients that developed AMLs using the HDF5-based nucleotide tallies that I created for this cohort. I used the `callVariantsPaired` function to perform somatic variant calling on tumour-normal pairs and the same set-up as for the WAS T-ALL cohort as described in Section 5.3.1.4 The raw variant calls were then filtered more stringently to extract a set of high-confidence calls and further limited to such variants that were predicted to overlap with coding regions using EVP – (McLaren et al., 2010).

5.4.1.3 Homozygousity Classification

In order to detect abnormalities that could hint at copy-number neutral LoH events or other irregularities, I applied the same classifier that I developed for the analysis of the WAS T-ALL samples (described in Section 5.3.1.6) to the AML samples. In order to identify homozygous regions on a mega-base scale I used the same cut-offs as for the T-ALL cohort.

5.4.2 Results

5.4.2.1 Copy number analysis

Since I could not adjust the WES-based coverages for GC-bias it was important to check whether such a bias was evident in the $\log_2$-ratios. To this end I created smoothed scatter plots of the relationship between the $\log_2$-ratios (per exon) of the AML samples (the tumour samples compared to their respective controls) and the GC-content of the respective exons. The plots are shown in Figure 5.17. It is apparent that indeed a tendency exists for those samples to have higher (positive) $\log_2$-ratios in exons with higher GC-content. The effects do not seem to be the same amongst the different sample pairs. Whether this effect is indeed negligible is hard to estimate. Therefore, as an additional precaution when analysing the $\log_2$-ratios, I chose to visualise for all exons the percentage of GC, in order to be able to detect copy number aberrations that show suspicious patterns of GC-content in their affected exons.

The only samples showing large copy number aberrations were the primary and relapse samples of patient 8. There are obvious large copy number variations on chromosomes 9 and 17 in those samples. Coincidentally those are the same chromosomes that carry recurrent large aberrations in a number of the T-ALL tumour samples. Those recurrent events are described in Sections 5.3.1.6 and 5.3.2.1 and the corresponding Figures 5.15 and 5.11. The large aberrations in the AML samples are shown in Figures 5.18 and 5.19. I coloured the data according to GC-content in the plots of the AML samples to see whether a correlation between high GC and gain or between low GC and loss could be established. Such a correlation could explain the pattern seen in Figure 5.17 that indicates a dependency between GC content and $\log_2$-ratio of the exons.

When comparing the distribution of GC-percentage per exon of the affected regions visualised in Figures 5.18 and 5.19 (i.e. on the short arms of chromosomes 9 and 17) I could not

Sample	Raw	Filtered	Coding
WAS1.AML	8751	1483	26
WAS8.AML	15911	1911	26
WAS8.AMLRelapse	5400	754	37
WAS9.AML	5127	729	33
WAS9PrimaryDNA	28394	11302	30
	63483	16179	152

Table 5.8: Table showing the number of somatic SNVs called in the AML samples from the WAS cohort. The *raw*, filtered (AF $\geq$ 10%) as well as coding SNV call counts are shown. Note that this table contains two sequencing runs of the same biological samples (*WAS9PrimaryDNA* and *WAS9.AML*) and for further analysis the latter one was used exclusively.

detect any biases towards higher GC-percentage in the region affected by a CN gain and lower GC in the region affected by a CN loss. Therefore it appears that the GC-dependency of the $\log_2$-ratios in samples from patient 8 is not just an effect of the large copy number variations overlapping with particularly GC-rich or GC-poor regions of the genome, but rather a more global effect that could be caused by different GC dependencies in the tumour and the control sample.

5.4.2.2 Comparative SNV calling

In total I called 63483 *raw* SNV calls with the somatic variant calling approach described in Section 5.4.1.2. After filtering out variant calls with allelic frequencies smaller than 10% a total of 16179 variants remained. An overview of the distribution of somatic SNV calls among the tumour samples is shown in Table 5.8. Note that the sample *WAS9PrimaryDNA* is a WGS sample of the same biological material as the sample labelled *WAS9.AML*. The number of coding variants in both is very similar – 30 and 33, respectively. Since I am limiting my analysis of somatic variants to coding variants the sample labelled *WAS9PrimaryDNA* was excluded from further analyses and I limited my analysis to the AML samples sequenced with WES. Those samples are all processed on the same flow-cell of the sequencing machine and should therefore have similar artefacts and mismatch profiles, allowing for easier comparative analyses. After filtering and exclusion of initial (low quality) sequencing runs, a total of 122 coding somatic variants were left in the whole exome AML samples.

To detect likely early mutations (i.e. likely driver mutations) I limited the variant calls further to those that were predicted to be non-synonymous and were present at an allelic frequency of at least 30%, leaving 15 variant calls. Each sample contained between 2 and 5 of those 15 driver candidates. No genes were found to be mutated in more than one of the tumour samples (not even between the primary and relapse AMLs of patient 8).

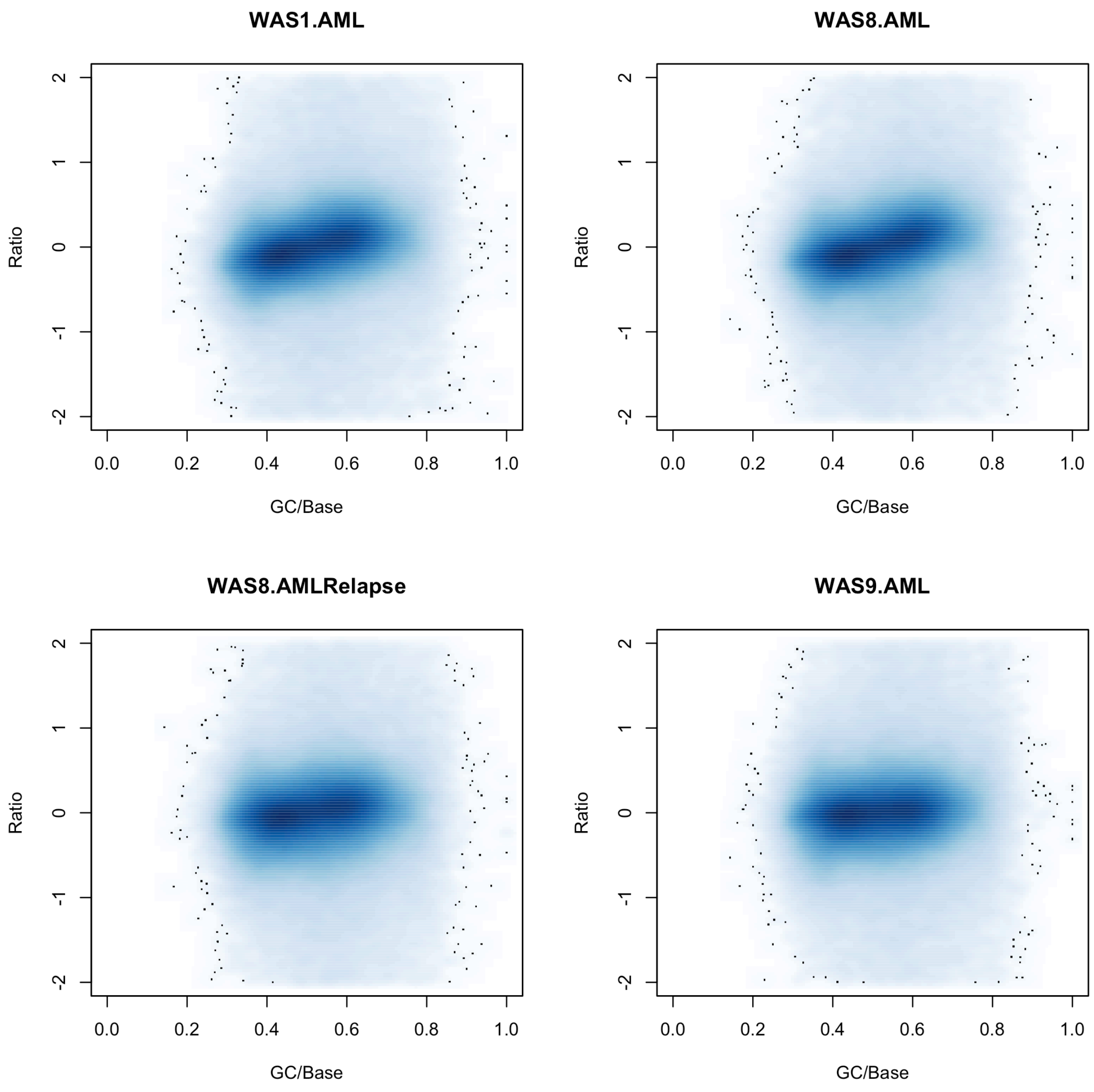


Figure 5.17: Smoothed scatter plots of the relationship between $\log_2$-ratios and GC-content (per exon) of sample pairs in the WAS AML cohort. The x-axis shows the percent GC bases per exon and the y-axis shows the observed $\log_2$-ratios in the exons. The majority of the data lies around a $\log_2$-ratio of 0 indicating that the copy number of most exons is unaffected in the cancer. It is important to note that in some sample pairs (notably the initial AML of patient 8) a clear dependency of the $\log_2$-ratio on the GC-content can be observed.

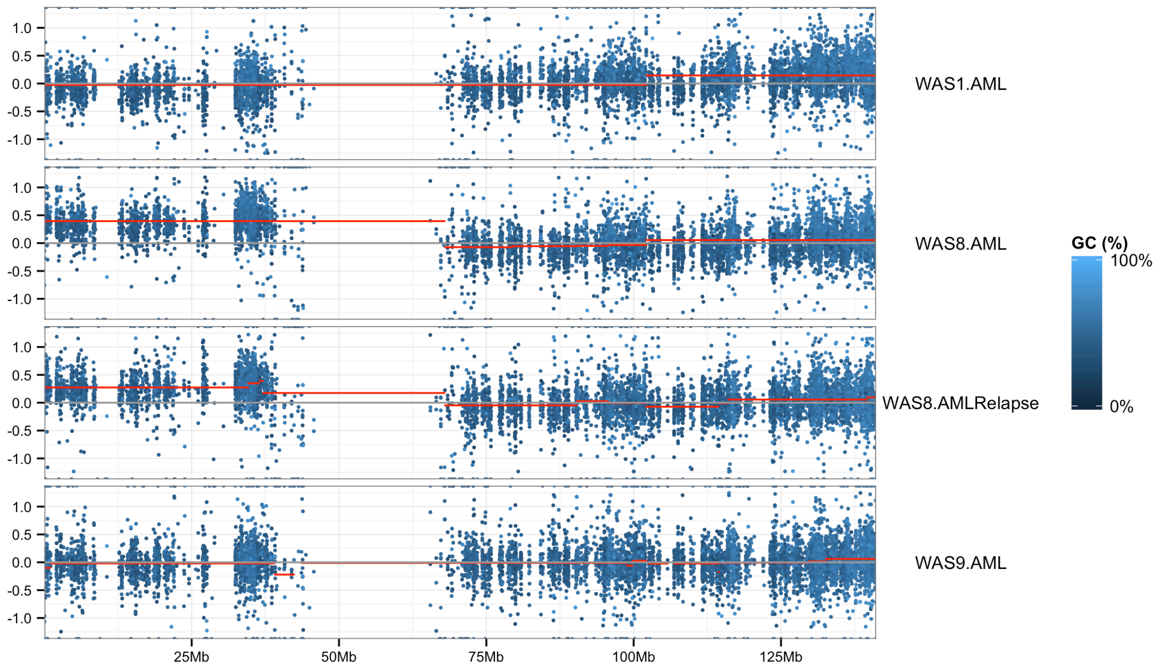


Figure 5.18: Overview plot of chromosome 9 showing the $\log_2$-ratios of AML samples sequenced with whole exome sequencing technologies. The genomic position is shown on the x-axis and the $\log_2$-ratio on the y-axis. Dots represent exons (coloured by GC-content) and red lines show the segmentation of those per-exon $\log_2$-ratios. This plot shows one of the few large-scale copy-number aberrations, which is a gain on the short arm of chromosome 9 in the AML and relapse sample of patient 8. Seeing this in contrast to the structure of this chromosome in the T-ALL samples of the same patient (panel **(b)** of Figure 5.14) we can corroborate the observation that this AML is indeed based in a different clone than the T-ALL, which is entirely expected, given that the location of the vector integrations in the AML samples as determined by LAM-PCR are different than in the T-ALL sample from the same patient.

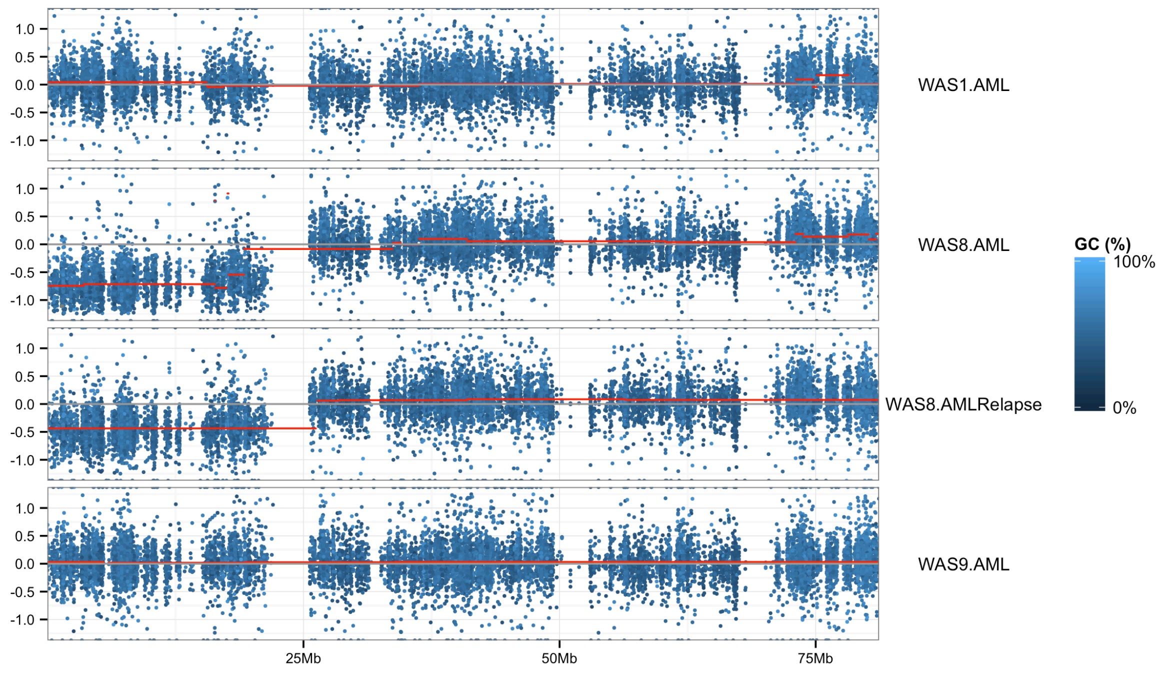


Figure 5.19: Overview plot of chromosome 17 showing the $\log_2$-ratios of AML samples sequenced with whole exome sequencing technologies. The genomic position is shown on the x-axis and the $\log_2$-ratio on the y-axis. Dots represent exons (coloured by GC-content) and red lines show the segmentation of those per-exon $\log_2$-ratios. This plot shows one of the few large-scale copy-number aberrations, which is a loss on the short arm of chromosome 17 in the AML and relapse sample of patient 8. The same chromosome was affected by a recurrent gain on the long arm in the T-ALL cohort (Figure5.11).

5.4.2.3 Homozygousity Classification

There were mega-base bins in the AML samples classified as homozygous, e.g. on the short arm of chromosome 17 where a deletion was apparent in the primary AML sample of patient 8 (Figure 5.19). Apart from those regions overlapping deletion events (i.e. those that are homozygous because there is only one copy left) the segmentation algorithm did not yield any large regions of homozygousity in the AML samples, in contrast to the copy-number-neutral LoH on chromosome 9 and the large stretches of homozygous regions in samples from patient 5 as I have described in the T-ALL samples (Figure 5.15).

5.4.3 Discussion

5.4.3.1 Copy number analysis

The type of deletion affecting the short arm of chromosome 17 as observed in the primary tumour and relapse sample of patient 8 is one of the more severe chromosomal abnormalities that had been reported in AML (Battiwalla, 2014). The important tumour suppressor gene *TP53* is located in this region (17:7.57-–7.59Mb) and the deletion therefore constitutes a loss of one copy of *TP53* in the AML samples of this patient. Such a deletion is known to have severe consequences when it happens in the germ-line (see Li-Fraumeni Syndrome – Li and Fraumeni (1969)). As described by Battiwalla, such a cytogenetic aberration has a generally poor outcome in AML and recurrence is observed in almost 50% of AMLs that were treated with HSC transplantation. In light of this observation it is not surprising that the only AML patient that had a relapse (patient 8) was the one in whose cancer samples a loss of one copy of *TP53* was found.

It should be noted that both CN events found in the tumour samples of patient 8 (Figures 5.18 and 5.19) seem to be present at high, but sub 100% frequency in the cells sequenced in the sample. Whether this stems from subclonality of the tumour population or contamination of the tumour sample with normal cells is hard to establish. If we assume no subclonality but rather contamination, we can estimate the proportion of cells carrying the events in the primary AML sample at 67.5% and 74.5%, in the duplication on chromosome 9 and the deletion on chromosome 17, respectively. For the relapse sample the estimated purities are 46.8% and 43.7% for each of those events. Although we cannot exclude subclonality playing a role, the numbers support the theory that the primary AML sample has a tumour purity of about 70% (67.5% − 74.5%). In the relapse sample about 45% of cells carry the event. Whether or not the remaining 55% of cells represent clonal populations of tumour cells or normal contamination can not be easily tested with the available data. However, the numbers point towards two aberrations being carried by the same population of tumour cells, since the estimated percentages of affected cells are so similar between the two events.

Sample	Affected Genes
WAS1.AML	ZSCAN10
WAS8.AML	NRAS, IKZF1
WAS8.AMLRelapse	NRAS, IKZF1
WAS9.AML	TP53RK, HK2, CSMD3

Table 5.9: Table showing the genes affected by high frequency somatic mutations in AML samples from WAS patients. Those genes can be considered driver candidates and some of them hint at an obvious cancer background, e.g. *NRAS* and *TP53RK*.

5.4.3.2 Comparative SNV calling

After visual inspection of the mismatch plots of the 15 likely driver mutations I could exclude some of them based on noisiness of the region or presence of the variant in other control samples, leaving a total of 11 likely early mutations distributed amongst the 4 AML samples as shown in Table 5.9

Amongst those driver candidate mutations some affect genes with obvious connections to cancer, like *NRAS* and *TP53RK* (TP53 Regulating Kinase), others are implicated in AML susceptibility, like *IKZF1* (Ross et al., 2013). These results verify that my variant calling approach is working well, and I can reliably detect somatic coding mutation that are potential drivers of cancer. In patient 1 the only remaining candidate mutation in the AML sample affects the gene *ZSCAN10* (Zinc Finger And SCAN Domain Containing 10), which encodes for a zink-finger protein. I could not find literature describing mutations in this gene in the context of AML. Whether this mutation is a novel driver of leukaemia or a passenger mutation of a different event that can not be detected using WES, is hard to distinguish.

Similarly to the results of the WAS T-ALL analysis of coding somatic SNVs, I cannot point out a single position, gene or pathway that is commonly affected between all the samples. Therefore the hypothesis that a second hit in addition to the vector integrations in or near *MDS1* and *MN1* occurred, can not be substantiated by the results of my analysis.

5.5 Identification of metastasis drivers and predisposing mutations in AML

5.5.1 Introduction

This project is a collaboration with the group of Prof. Dr. Anthony Ho from the Universitätsklinikum Heidelberg. It is based on the whole exome sequencing analysis of pairs of AML and control samples from 10 patients and one auxiliary set of samples of breast cancer and matched control from a relative of one of the patients. An overview of the samples in this cohort is given in Table 5.10.

My objectives in working with this data were to establish a variant calling approach that provides high-quality variant calls that can be reliably verified using PCR methods. Furthermore, my analysis was geared towards identifying somatic mutations that could be candidate drivers for metastasis as well as pre-leukaemic mutations present in the germ-line samples of patients. A third aspect was the search for likely predisposing mutations in the germ-line samples of some of the AML patients. However, this part of the project is still ongoing and I will not present results on it in this thesis.

Note that the patients with the IDs *KM89*, *KM891* and *KM892* are two brothers who both developed an AML and their sister, who developed a breast cancer.

5.5.2 Methods

5.5.2.1 Sequencing

The sequencing library preparation was done by my collaborators according to the following protocol. The library preparation for capturing of selected DNA regions (Agilent Human All Exon 51Mb v4) was performed according to Agilent's SureSelect protocol for Illumina paired-end sequencing. In brief, 3.0μg of genomic DNA was sheared on a Covaris™ E220 instrument. The fragment size (150-300 bp) and the quantity was confirmed with the Agilent 2100 Bioanalyzer 7500 chip. Fragmented DNA was end-repaired, adenylated and ligated to Agilent indexing-specific paired-end adaptors. The DNA with adaptor-modified ends was PCR amplified (6 cycles, Herculase II fusion DNA polymerase) with SureSelect Primer and SureSelect Pre-capture Reverse PCR primers, quality controlled on the DNA 7500 assay for the library size range of 250 to 450 bp and hybridized for 24 hrs on 65 °C (Applied Biosystems 2720 Thermal Cycler). The hybridization mix was washed in the presence of magnetic beads (Dynabeads MyOne Streptavidin T1, Life Technologies) and the eluate was PCR amplified (16 cycles) in order to add the index tags using SureSelectXT Indexes for Illumina. The final library size and concentration was determined on Agilent 2100 Bioanalyzer 7500 chip. Each library was sequenced on an Illumina HiSeq 2000 platform following the manufacturer's protocol, with a paired end sequencing run of 2×76 bp to at least $70\times$ mean coverage. Images from the instrument were processed using the manufacturer's software to generate FASTQ sequence

Sample	Patient	Type	Notes	Molecular Biology
KM103AML	KM103	Case	-	FLT-3-ITD
KM103Control	KM103	Control	-	-
KM115Control	KM115	Control	The samples from patient *KM115* were mis-labeled (sample swap in sequencing); see Section 5.5.3.2 for details	no screening performed
KM115AML1	KM115	Case		no screening performed
KM115AML2	KM115	Case		no screening performed
KM80AML	KM80	Case	-	FLT-3 ITD, NPM1
KM80Control	KM80	Control	-	-
KM87AML	KM87	Case	-	FLT-3-LM, NPM1
KM87Control	KM87	Control	-	-
KM88AML	KM88	Case	-	CBFb-MYH11 (inv16)
KM88Control	KM88	Control	-	-
KM891AML	KM891	Case	AML sample of older brother	no mutation found
KM891Control	KM891	Control	control sample of older brother	-
KM891PreAML	KM891	Case	pre-leukaemic sample of the older brother	-
KM892Control	KM892	Control	control sample of sister	-
KM892Primary	KM892	Case	breast cancer sample of sister	no screening performed
KM89AML	KM89	Case	AML sample of older brother	no mutation found
KM89Control	KM89	Control	control sample of younger brother	-
KM89PreAML	KM89	Case	donated bone marrow sample from younger to older brother	-
KM90AML	KM90	Case	-	FLT-3 ITD
KM90Control	KM90	Control	-	-
KM93AML	KM93	Case	-	CEBPA
KM93Control	KM93	Control	-	-
KM95AML	KM95	Case	-	no mutation found
KM95Control	KM95	Control	-	-

Table 5.10: Tabular overview of the samples in the AML cohort from the collaboration with Prof. Dr. Anthony Ho. Columns are in order: the sample ID, the patient ID the type of sample (*Case* or *Control*), a short description and a list of mutations known from molecular biology analysis in the in-house analysis lab of our collaborators at the Universitätsklinikum Heidelberg.

Parameter	Meaning
`--nthreads=12`	Threaded processing using 12 processes at the same time
`--batch=4`	Using batch mode 4 which instructs the aligner to load the genome index into the computers memory completely to improve speed; smaller values lead to the genome index being loaded from disk during usage which reduces memory consumption at the cost of increased runtime
`-A sam`	Output the alignments in `SAM` format (Li et al., 2009) instead of the default GSNAP format

Table 5.11: Table listing the parameters used for the alignment of paired-end reads.

files.

5.5.2.2 Common AML Mutation Screening

The samples were subjected to PCR-based screening for typical AML mutations by our collaborators at an external diagnostics lab. The results of this screening are shown in the last column of Table 5.10.

5.5.2.3 Read alignment

The reads were aligned using the GSNAP read alignment software (Wu and Nacu, 2010) version 2013-07-20 with the parameters described in Table 5.11. I left all other options at their respective default settings.

The alignment was heavily parallelised using the ability of GSNAP to process FASTQ files in parts to split each sequencing lane into ten parts which in turn used 12 sub-processes each (Table 5.11). In this way the generation of raw read alignments is sped up significantly (using a sufficiently powerful compute cluster; I used the EBI cluster for my calculations) at the expense of having to add additional post-processing steps for merging of the output files.

5.5.2.4 Alignment post-processing

The alignments were post-processed with the same pipeline as I implemented for the AML and ALL samples from the WAS cohort. The details are described in Section 5.2.2

5.5.2.5 Creating HDF5-based Nucleotide Tallies

I created a set of HDF5-based nucleotide tallies from the alignment data of this cohort using the same approach as for the WAS data (described in Section 5.2.3), with some modifications. In the WAS cohort I was dealing with a mixture of whole exome and whole genome sequencing samples and decided to tally in 1 mega-base bins along the genome to keep the pipeline simple. In this cohort there are only WES samples and therefore I ran the tallying functions on a set of genomic intervals representing the annotated genes in the human reference genome. I extracted those intervals from the **`TxDb.Hsapiens.UCSC.hg19.knownGene`** `R/Bioconductor` annotation package using the `genes` function. This allowed for a faster creation of the tallies, since I only had to process $\sim 3\%$ of the genome. I did not expect significant loss of information since the WES library is designed to only yield reads overlapping coding regions and I would not expect alignments that fall outside of coding regions to be reliable data.

5.5.2.6 Copy number analysis

As described in Section 5.4.1.1, WES data is challenging to deal with when trying to adjust for GC-bias in the coverage and I therefore treated the AML samples in this cohort in the same way as I treated the AML samples in the WAS cohort, i.e. not adjusting for GC count and analysing copy number aberrations based on the $\log_2$-ratios of coverage per exon. I used the `h5dapply` and `binnedCoverage` functions working on a list of specific genomic intervals I retrieved from the same source as the intervals in which I created the tallies, i.e. the **`TxDb.Hsapiens.UCSC.hg19.knownGene`** annotation package (Carlson, 2011). I computed the $\log_2$-ratios for the tumour samples on the raw (unadjusted) coverages directly, followed by segmentation with **`DNAcopy`** and visualisation as chromosome overview plots. This process was identical to the way I implemented the copy number analysis of the WAS AML samples (Section 5.4.1.1).

5.5.2.7 Calling SNVs

For this cohort I created two sets of SNV calls to find both single sample and somatic variants. Somatic variants were called using the `callVariantsPaired` function provided by **`h5vc`** (see section 4.6.3 for a description of the variant calling algorithm). I used the same parameters to call somatic variants as in the WAS T-ALL cohort (described in Section 5.3.1.4). The single sample variant calls were also obtained in the same way as for the WAS T-ALL cohort and used for the purpose of identifying homozygous regions (see Sections 5.3.1.5 and 5.3.1.6).

5.5.2.8 Allelic frequency distribution analysis

In order to spot homozygous regions, I performed the same analysis as for the two WAS cohorts (T-ALL and AML) where I ran a classifier on the track of percentage of homozygous SNV calls per mega-base bin (Sections 5.3.1.6 and 5.4.2.3). This helped with the identification of copy-number neutral events, like LoH. In this way I could identify the copy number losses which lead to single-copy states, since those regions are homozygous as well.

5.5.2.9 PCR Validation of SNV calls

From the somatic variant calls I selected a high-confidence sub-set for PCR validation. The variants were selected from the top of the list of calls, when sorted by both coverage and allelic frequency. Each variant was manually checked for suspicious mismatch patterns in the immediate surrounding region (i.e. mismatch plots were created and evaluated).

The PCR validations were performed by my collaborators. Forward and reverse primers were designed mapping roughly 250 bp up- and down-stream of the variant to amplify a section of the genome of approximately 500 bp length with the variant centered in it. One third ($8\mu l$) of the obtained $25\mu l$ PCR product was tested in 1.7% agarose gel for the detection of the amplification band. Based on the thickness of the band (which indicated successful amplification) between 6 and $12\mu l$ of PCR product was diluted with water to prepare $13.5\mu l$ of diluted PCR product to which $1.5\mu l$ of forward primer was added. The well mixed $15\mu l$ of diluted PCR product and forward primer was then sent to an external service provider (Eurofins Genomics[3]) for direct PCR product sequencing. The result of the direct PCR product sequencing was sent back by the company as chromatogram files in which the presence of the called SNVs was manually checked.

5.5.2.10 Metastasis-driver detection in series of mouse xenografts

Our collaborators set up a series of mouse xenografts wherein the primary tumour samples of the patients in this cohort are transferred into immunocompromised mice. This is followed by a series of xenografts from the engrafted cells of one mouse into another mouse. Based on the assumption that there is a genetic basis for the ability to metastasise or engraft in an immunocompromised mouse we can expect this process to exert a selective pressure on those mutations that enable cells to engraft or improve their engraftment efficiency. After having established the SNV calling pipeline and the ability to verify the presence of SNVs through PCR the next step was to use targeted sequencing on those genes that carry somatic mutations (as detected by my analysis pipeline) in order to generate profiles of the allelic frequencies of the mutation in the time-series. This is ongoing work and the samples are currently being sequenced. I expect the data to be available by the end of the year.

5.5.3 Results

5.5.3.1 Copy number analysis

As a quality control step before analysing the segmented $\log_2$-ratios of the AML samples in this cohort I created smoothed scatter plots of the dependency between $\log_2$-ratios and GC-content of the exons for each of the sample pairs I analysed. Figure 5.20 shows the plots, that are of the same type as those I created for the WAS AML samples (Figure 5.17). Notably there are a number of samples exhibiting unusual patterns which hint at severe problems

[3] http://www.eurofinsgenomics.eu/

with data quality. Most of those samples are concentrated within the subset of samples that come from the two brothers that both developed AMLs as well as the breast cancer sample from their sister. We can see that we will have to consider the observed $\log_2$-ratios in the sample pairs for *KM891PreAML*, *KM89PreAML* and *KM891AML* to be potentially unreliable and we should take extra precautions when analysing them. This assertion is corroborated by the histograms of observed $\log_2$-ratios shown in Figure 5.21, which show far more noisy and spread-out $\log_2$-ratios for the samples in question, when we compare them to the distributions observed in other samples from the same cohort. Essentially the only samples between the three siblings showing $\log_2$-ratio to GC-content dependencies suitable for further analysis are the AML sample of the younger brother (*KM89AML*) and the breast cancer sample from the sister (*KM892Primary*) and I have therefore focused on those samples and disregarded the other samples from this set of siblings.

It is not surprising that the AML sample of the older brother is unusable, since this sample was stored for 5 years before the sequencing and the noisy data indicates that the sample apparently deteriorated during that time. We were not able to reconstruct a history of the storage facility to check for problems like power outages that might have occurred during the 5 years and could have lead to thawing and accelerated degradation of the sample. Investigating the samples labelled *KM891PreAML* and *KM89PreAML*, which are actually one pre-leukaemic sample and one sample that is taken from an initial bone marrow transplant performed between the younger and the older brother, we can see a clear dependency of $\log_2$-ratio to GC-content. An explanation for this lies in the fact that a subset of the samples was sequenced on a different flow-cell and lane than the rest of the cohort. Specifically both case and control samples from the sister (*KM892*) as well as the control sample from the younger brother (*KM89*) and the pre-leukaemic sample from the older brother (*KM891*) were sequenced together and separate from the rest. This means that in addition to any noise from the sample itself, the comparisons of *KM891AML* and *KM891Control* can be affected by a difference in sequencing chemistry leading to a difference in the way the coverage depends on the GC-content. Such a difference will then of course also affect the $\log_2$-ratio (Figure 5.22).

When we compare samples from the same flow cell, which therefore should have the same dependency of coverage to GC-content, the usage of ratios removes this bias and the dependency between $\log_2$-ratio and GC-content becomes essentially flat, as can be seen for e.g. sample *KM89AML* or *KM87AML* in Figure 5.20. Apart from the problematic samples discussed above, all samples looked reasonable and were used for further analyses.

After segmenting the $\log_2$-ratios of all samples some large chromosomal aberrations became apparent. Among those are such events that have been previously reported to play a role in the occurrence or progression of AMLs. One such example is the occurrence of large deletions or even the total loss of one copy of chromosome 7 (Figure 5.23). Such deletion events have been shown to affect specifically the *MLL3* gene (Histone-lysine N-methyltransferase MLL3), which is a haploinsufficient tumour suppressor located on chromosome 7 in the region

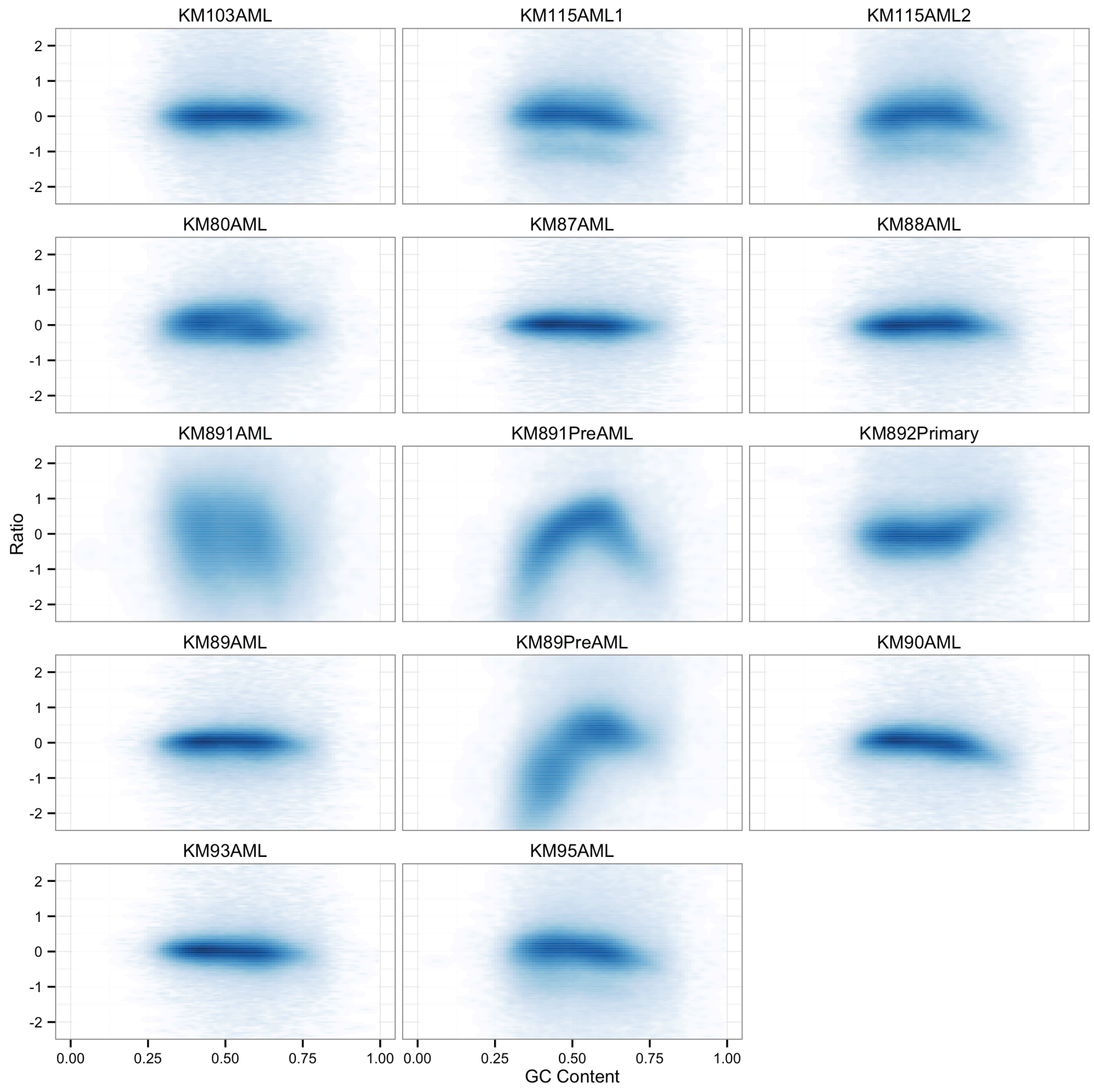


Figure 5.20: Smoothed scatter plots of the relationship between $\log_2$-ratios and GC-content (per exon) of sample pairs in the cohort. The x-axis shows the percent GC bases per exon and the y-axis shows the observed $\log_2$-ratios in the exons. The majority of the data lies around a $\log_2$-ratio of 0 indicating that the copy number of most exons is unaffected. It is important to note that in some samples a clear dependency between $\log_2$-ratio and GC-content can be established. Furthermore, some samples (e.g. *KM115AML1*, *KM115AML2* and *KM95AML*) show a banding in the plot that hints at the presence of large single-copy regions within the genomes of those samples, i.e. they are likely affected by large deletion events. There are samples that show severe effects, such as *KM891AML* (very noisy) and *KM891PreAML* (non-linear GC-dependency).

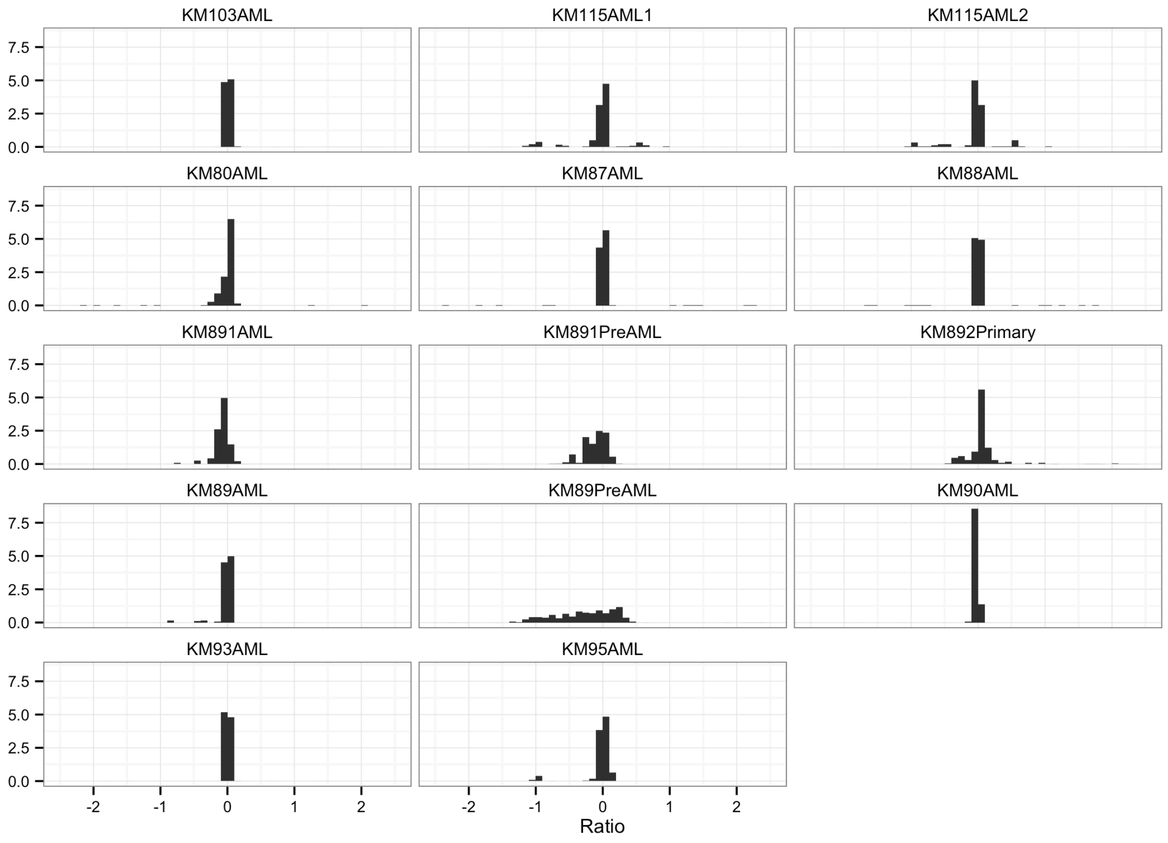


Figure 5.21: Histograms illustrating the distributions of segmented $\log_2$-ratios in the samples from this cohort. Note how we can see indications of large deletions in the samples *KM115AML1* and *KM115AML2* as well as *KM95AML* through peaks near a $\log_2$-ratio of -1, which correspond to losses of one copy in a diploid region. We already saw indications of such events through banding observed in the smoothed scatter plots shown in Figure 5.20. Furthermore a general noisiness is apparent in samples *KM891PreAML* and *KM89PreAML*, as was also apparent from the smoothed scatter plots shown in Figure 5.20

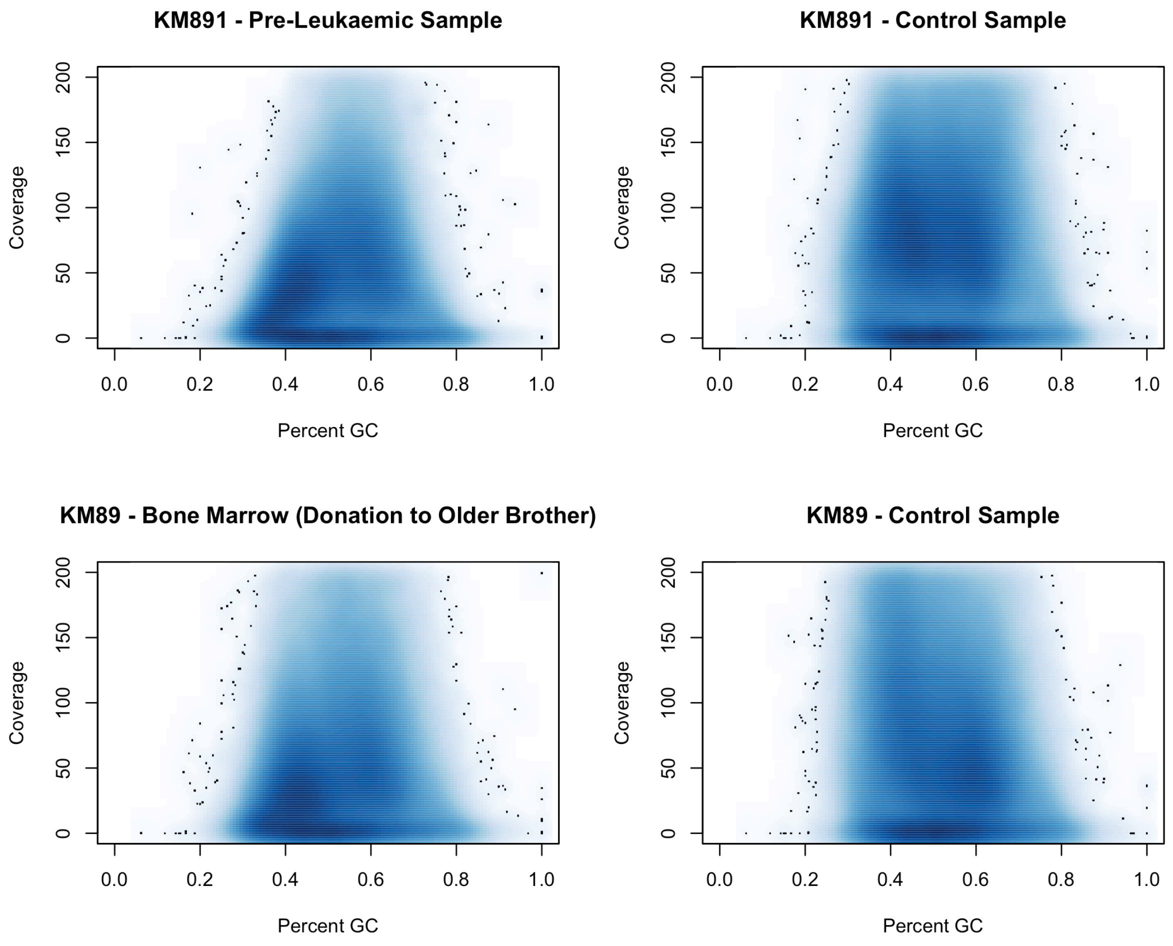


Figure 5.22: Smoothed scatter plots of the relationship between coverage and GC-content for samples from the two brothers (Patients *KM89* and *KM891*). The left plots are showing the case samples and the right plots show their matched controls. Each pair of plots exhibits the same non-linear dependency between $\log_2$-ratio and GC-content (see Figure 5.20, panels *KM89PreAML* and *KM891PreAML*). Looking at the relationships between the coverage and the GC-content as shown here we can immediately see how each case sample differs from their respective matched control sample in a similar way and both controls and both case samples behave similarly. This explains why their respective $\log_2$-ratios show similar non-linear dependency on the GC-content as shown in Figure 5.20.

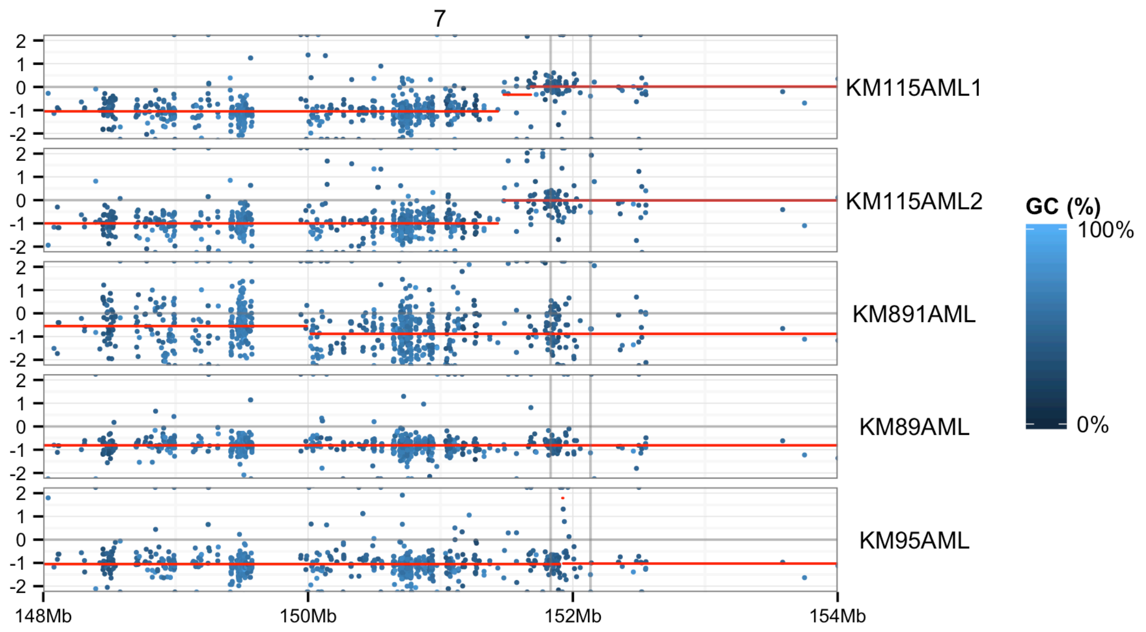


Figure 5.23: Overview plot of chromosome 7 showing the $\log_2$-ratios and their segmentation for a subset of the samples (i.e. those with large deletions on chromosome 7). The genomic position is shown on the x-axis and the $\log_2$-ratios are shown on the y-axis. The dots represent exons and are coloured according to their respective GC-content to visualise eventual biases. Red lines represent the segmentation of the $\log_2$-ratios. The genomic interval corresponding to *MLL3* is marked with vertical grey lines. Note how the deletion events in *KM115AML1* and *KM115AML2* lie upstream of *MLL3*, whereas in the other samples a loss of one copy of *MLL3* can be observed.

7:151832010-152133090 (Chen et al., 2014), frequently leading to AML. Figure 5.23 shows an overview of the region containing *MLL3* in samples with deletions on chromosome 7. Here we can see deletions overlapping *MLL3* in samples *KM95AML*, *KM89AML*, *KM891AML*. The $\log_2$-ratios of those events are exactly at or extremely close to -1, indicating that the events affect all or most of the cells in the respective sample, i.e. they are early events in the tumour progression and therefore potential drivers of the cancer. In the two tumour samples of patient *KM115* there are large deletions upstream of the *MLL3* locus. It seems that *MLL3* is not affected by those deletions directly. Furthermore, a close investigation of other copy number aberrations in those samples revealed that those samples are affected by deletions events overlapping the genes *PURA* and *PURB* (Homo sapiens purine-rich element binding protein A and B). The locations of the genes are shown in the following table:

Gene	Chromosome	Start	End
PURA	5	139493708	139499001
PURB	7	44915892	44924960

Interestingly the two samples in question (*KM115AML1* and *KM115AML2*) show deletion events overlapping the location of both these genes as is illustrated in Figure 5.24. Those events are at $\log_2$-ratios of around -1, indicating presence of the events in almost all cells in the samples, i.e. they are likely early events and therefore potential drivers of the cancer. Simultaneous deletion of both of these genes has been implicated in the development of AML as a progression of myelodysplastic syndrome (MDS) as reported by Lezon-Geyda et al..

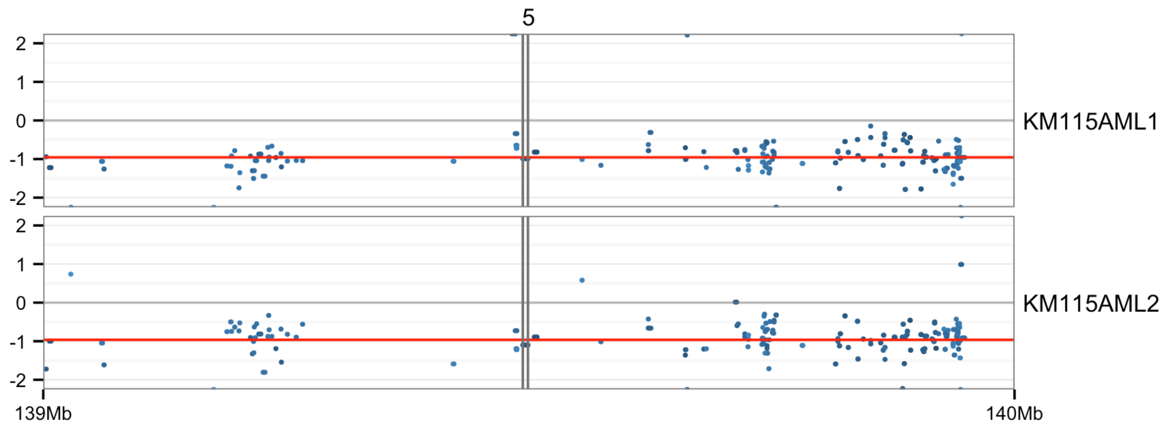
5.5.3.2 Sample Swap in Patient *KM115*

Early in my analysis I discovered a sample swap in the cohort in the samples from patient *KM115*, where there were a surprisingly high number of somatic variant calls (millions) while a typical sample has a couple of hundred somatic variant calls at most. In addition to this SNV-level evidence, the copy number analysis further substantiated the sample swap as illustrated in Figure 5.25. The following table shows the original and fixed sample labels:

Old Label	New Label
KM115Control	KM115AML1
KM115Relapse	KM115AML2
KM115AML	KM115Control

Knowing that the original control and AML samples were swapped lead to my decision to label the two AML samples simply *KM115AML1* and *KM115AML2* since it is not quite clear whether even more mistakes were made and I did not feel confident to make a data-based ordering of the AML samples into primary and relapse, respectively.

(a)



(b)

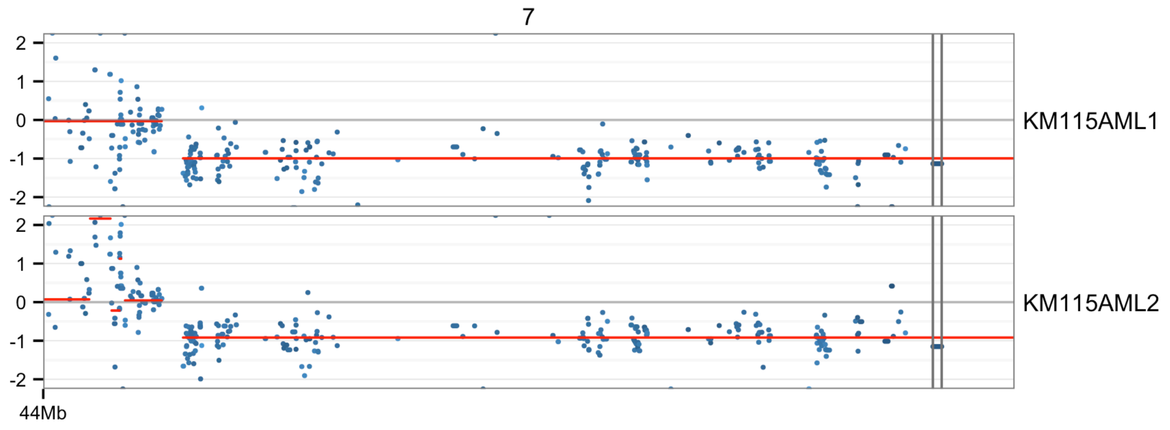


Figure 5.24: Overview plots showing the $\log_2$-ratios of exons (blue) and their segmentations (red lines) for the two samples *KM115AML1* and *KM115AML2* in the regions surrounding the *PURA* and *PURB* genes, respectively. The genomic position is shown on the x-axis and the $\log_2$-ratio on the y-axis. Vertical grey lines mark the respective start and end of the two genes. **(a)** region surrounding *PURA*; **(b)** region surrounding *PURB*. Note how both genes are at a $\log_2$-ratio of ~ -1 indicating loss in all (or almost) all cells in the samples, i.e. they are likely early events.

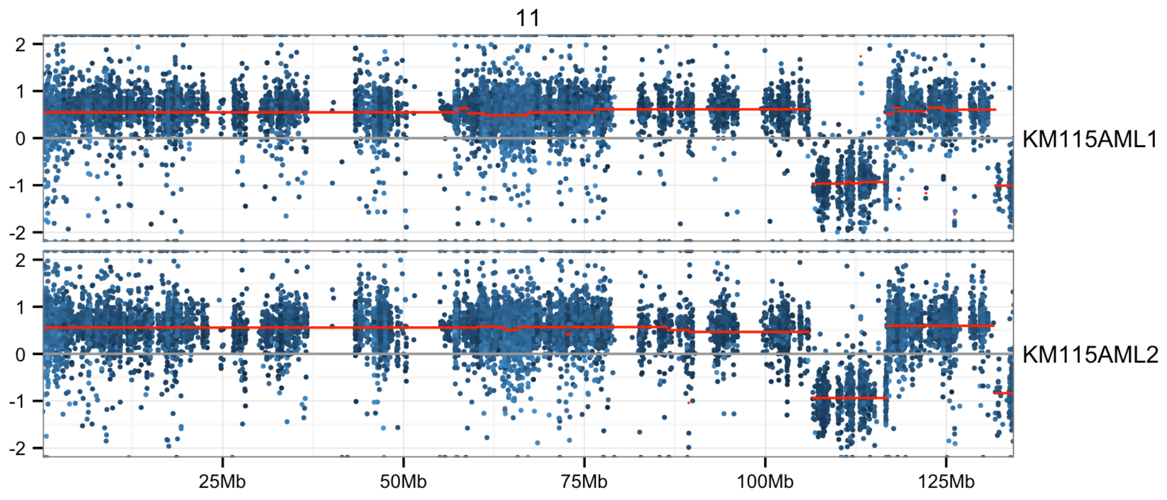


Figure 5.25: Overview plot of chromosome 11 showing the $\log_2$-ratios observed with the fixed (i.e. new) sample labels. Genomic position is shown on the x-axis and $\log_2$-ratio of case and control sample on the y-axis. The dots represent exons and are coloured according to their respective GC-content. Red lines represent the segmentation of the $\log_2$-ratios of exons. This is the same type of plot as shown for the WAS AML cohort in Figures 5.19 and 5.18. Given the obvious similarity between the behaviour of the two AML samples (same pattern of gains and losses) it is apparent, that the labelling used here (i.e. the new/fixed sample labelling) correctly identifies the control and cancer samples.

5.5.3.3 Variant Calling

I created a total of 3901 somatic variant calls of which 1065 had an allelic frequency larger than 10% and 446 had an allelic frequency higher than 20%. After filtering for variants that are coding and non-synonymous, 144 variants with allelic frequency higher than 20% remained. A per-tumour sample overview of the somatic variants is shown in Figure 5.26 and Table 5.12 shows the distribution of variants amongst the samples.

From the 144 somatic coding non-synonymous variant calls with allelic frequencies larger than 20% we selected 33 somatic SNVs and one somatic small deletion to be validated by PCR as well as one variant with only 18.5% allelic frequency. This was done to verify that the variant calling pipeline works and that we could track those variants for the later analysis of xenografts in series. The validation of a further 12 SNVs was delayed due to a lack of available DNA from their corresponding samples and my collaborators are working to obtain the DNA in order to verify those variants as well. Of the 34 variants that were already subjected to PCR validation we could reliably detect 33 that were present at allelic frequencies between 20.5% and 50.5%. One variant could not be confirmed, which was the variant with the lowest allelic frequency of only 18.5%. This variant is located on chromosome 19 (19:55250979) and was detected in the tumour sample of patient KM90. The mismatch plot showing the region for both samples of the patient is given in Figure 5.27. From visual inspection the variant call seems justified and it is likely that the relatively low allelic frequency is the reason for the failed PCR validation, since the PCR protocols for variant detection have lower limits for the

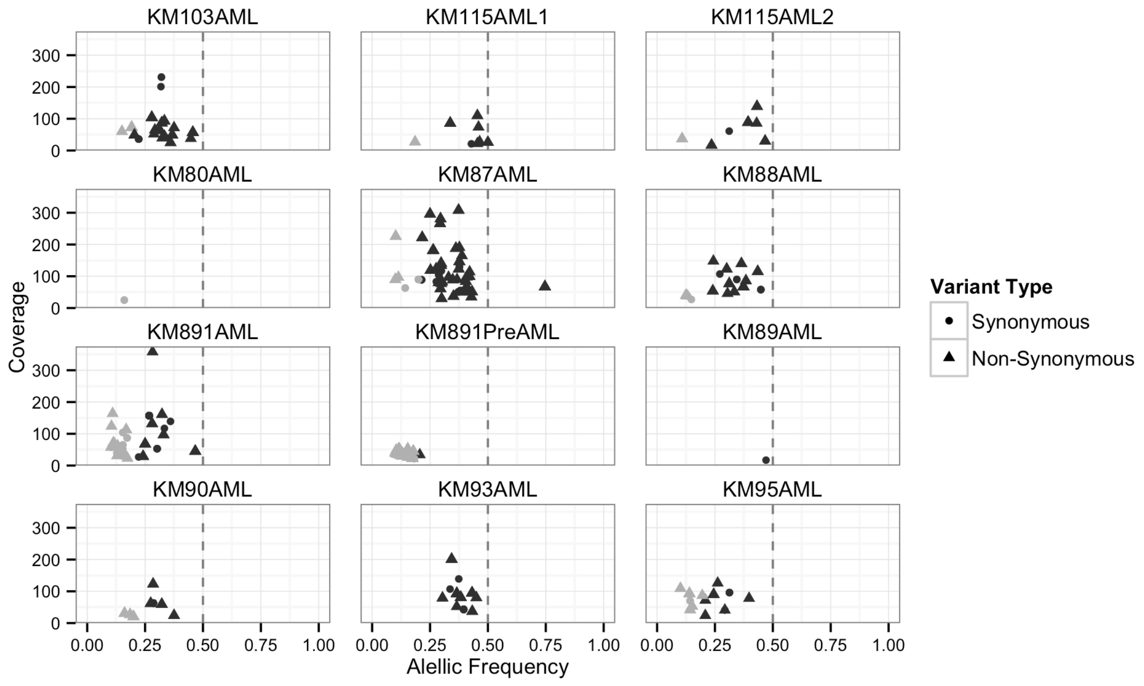


Figure 5.26: Overview plot of somatic coding variants in AML samples. The x-axis shows allelic frequency and the y-axis shows the coverage. Each dot represents a variant call and is coloured black if it passed the basic filters ($AF > 20\%$) and grey otherwise. Circles represent synonymous mutations and triangles represent non-synonymous mutations. The expected allelic frequency for a heterozygous early event (50%) is marked with a dashed vertical line. Note that some samples do not carry any variants that passed the filters (*KM89AML*) or do so at very low frequency (*KM891PreAML*). Sample *KM89AML* carries only one coding mutation and it is a non-synonymous one. In principle we can group the samples into active and non-active ones, based on how many variant calls are present. *KM87AML* and *KM88AML* are examples of active samples with many variant calls, this points to a more instable genome and the potential for sub-clonality, but could also be the effect of noise in the data. The samples *KM90AML*, *KM115AML1* and *KM115AML2* are of the non-active variety and show relatively few variants. Those samples are likely more stable and less likely to contain many sub-clones.

Sample	Raw	Filtered ($AF > 20\%$)	Coding ($AF > 20\%$)
KM80AML	20	4	0
KM87AML	96	73	34
KM88AML	42	27	10
KM89PreAML	92	21	1
KM89AML	21	5	0
KM891PreAML	1440	85	1
KM891AML	302	117	8
KM892Primary	1369	529	47
KM90AML	44	16	4
KM93AML	32	19	8
KM95AML	86	36	6
KM103AML	113	57	14
KM115AML1	41	21	6
KM115AML2	203	55	5
	3901	1065	144

Table 5.12: Tabular overview of the distribution of variant calls among the samples. The second column shows the number of *raw* variant calls, i.e. output of the variant calling function. The column labelled *Filtered* shows how many of those *raw* calls have allelic frequencies larger than 20% and the last column shows how many of the filtered variants are actually coding (and non-synonymous).

detectable allele frequency in the range of $\sim 20\%$. However, my collaborators work on establishing protocols that should allow for the detection of variants with allelic frequencies as low as 10%. This work is still ongoing.

To identify early mutations that might be drivers of the cancer I further filtered the variant calls to have allelic frequencies higher than 30% and be predicted to be deleterious by SIFT, as obtained through the application of the EVP Software on the variant call set (Kumar et al., 2009). This revealed mutations in typical cancer genes such as *DNMT3A* and *NRAS*.

Through annotation of the variant calls with overlapping previously reported variants from dbSNP (Smigielski et al., 2000) and the Catalogue Of Somatic Mutations In Cancer (COSMIC) database (Forbes et al., 2011), I could further narrow down the list of candidate mutations. If a variant was overlapping with a dbSNP variant and had a significant population frequency (more than 1%) I excluded it as a likely cancer driver mutation, since it is described as being present in the general population at frequencies at which I would expect an obvious cancer relation of the variant to have been detected already. Amongst the deleterious coding somatic variant calls that I created for this cohort, I found 13 with associated dbSNP accession numbers, i.e. there is an annotated variant in dbSNP at the same position. Note that the EVP tool only checks for overlap of position, not of alternative allele, when reporting existing variants overlapping with the input set of calls. Therefore each variant overlapping a known site was investigated separately because it might have a different and not previously described alternative allele at this position.

The COSMIC database contains curated variant calls from many different analyses of can-

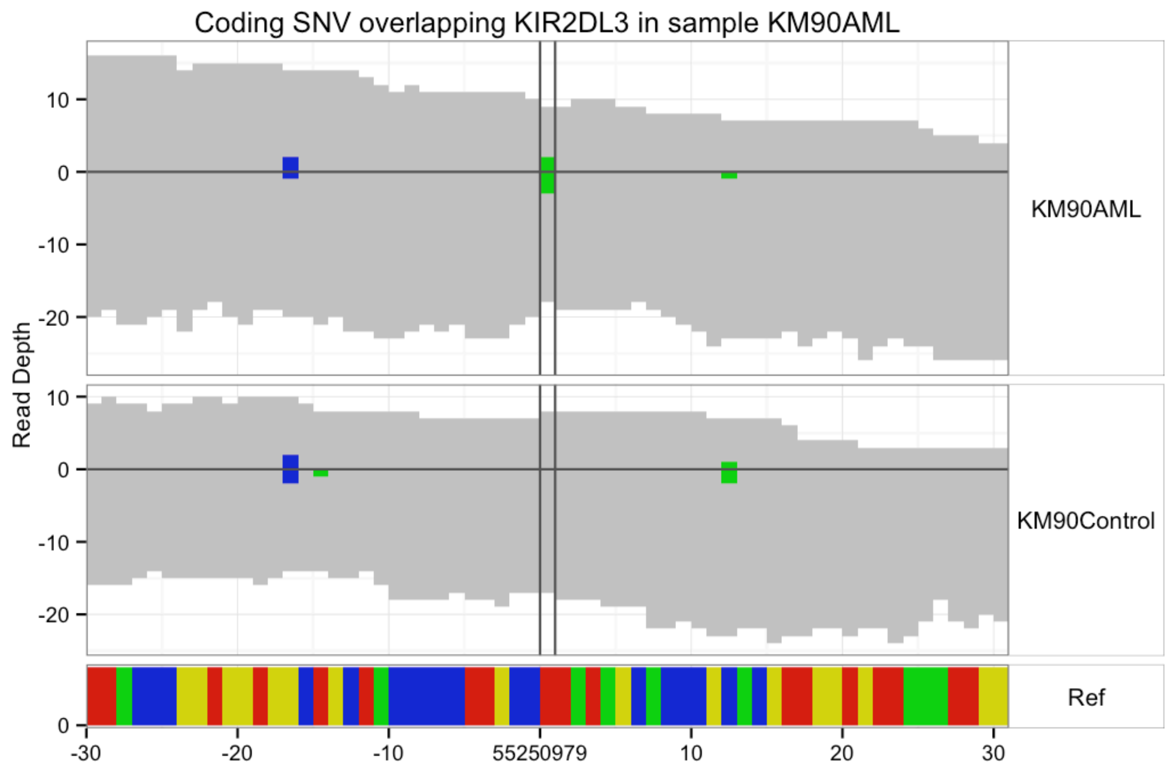


Figure 5.27: Somatic coding variant in the AML sample of patient KM90 overlapping the *KIR2DL3* gene. This SNV is called at an allelic frequency of 18.5% and was not detectable by PCR validation. However, from the mismatch plot and local context it seems to be a good variant call.

cer sequencing data. An overlap in COSMIC is indicative of a true cancer variant and the frequency at which the variant was observed in different tissues and cancer types can be investigated on the COSMIC webpage. It should be noted that sometimes a COSMIC variant overlaps a variant call with matching alternative allele, but the COSMIC entry describes the variant in a completely different tissue/tumour context. Although it can certainly happen, that cancer mutations occur in different cancer types (e.g. in genes like *TP53*), when the variant had only been described in one or two experiments I still investigated the variant stringently for any signs of the presence of an artefact. Looking into the overlap of my likely early events ($AF > 30\%$) with the COSMIC database, I found 4 mutations to overlap with positions that have been previously reported to contain cancer mutations. The following table shows the samples and affected genes:

Sample	Gene	ID
KM87AML	ETV6	COSM158691
KM87AML	DNMT3A	COSM442676
KM103AML	U2AF1	COSM1318797
KM115AML1	KRT77	COSM162063

Notably the mutations affecting *DNMT3A*, *U2AF1* and *ETV6* have been reported in as cancerous mutations in haematopoietic and lymphoid tissues (Ding et al. (2012) and Neumann et al. (2013)) according to COSMIC. Moreover *DNMT3A* mutations are of particular interest because they have been implicated to exist in pre-leukaemic states at lower frequencies which turn into fully developed leukaemias through the acquisition of additional driver mutations (Corces-Zimmerman et al., 2014). In their paper Corces-Zimmerman et al. describe a model for AML progression in which HSCs acquire a subset of the mutations that constitute the mutation profile of the fully developed AML. This acquisition of mutations occurs in a serial fashion and somatic mutations in e.g. *IDH2* or *DNMT3A* can spontaneously occur in single HSCs, which then start to expand and acquire further mutations, before the cell population transfers to a tumour state. Notably it is possible that the pre-leukaemic HSCs remain a population in the patient after the AML is fully developed. The pre-leukaemic HSCs are implicated to form a reservoir for remission post-chemotherapy, in case some of them are able to survive the treatment. In addition to this somatic mutation in *DNMT3A* I specifically searched for evidence of the presence of such pre-leukaemic mutations in the control samples of our patients, extending the search range to both *DNMT3A* as well as *IDH1* and *IDH2* based on the information presented by Corces-Zimmerman et al. (2014). I was able to identify pre-leukaemic mutations of *DNMT3A* in the control samples of patients *KM80*, *KM89* and *KM90* (Table 5.13). Note that there are some mutations in the control samples of patients *KM115* and *KM93*, that look heterozygous in the control samples and are therefore not likely to be pre-leukaemic mutation. They are probably normal population variants. In *IDH2* I identified a variant (`15:90631838 C->T`) that was present at 11% AF in the control sample of patient *KM89* and 45% in the matched AML sample. This profile of allelic frequency development fits the model proposed by Corces-Zimmerman

Position	Ref	Alt	Sample	AF
25457242	C	T	KM80AML	0.38
25457242	C	T	KM80Control	0.21
25457242	C	T	KM87AML	0.51
25457242	C	T	KM89PreAML	0.01
25457242	C	T	KM90AML	0.50
25457242	C	T	KM90Control	0.05
25457242	C	T	KM95Control	0.02
25467448	C	A	KM89AML	0.94
25467448	C	A	KM89Control	0.19
25467448	C	A	KM89PreAML	0.02
25469502	C	T	KM115Control	0.57
25469502	C	T	KM115AML1	0.50
25469502	C	T	KM115AML2	0.41
25469502	C	T	KM90AML	0.01
25469502	C	T	KM93AML	0.49
25469502	C	T	KM93Control	0.48

Table 5.13: Tabular overview of *DNMT3A* mutations in AML and matched control samples. The columns show the genomic position (on chromosome 2), the reference allele, alternative allele, affected sample and observed allelic frequency. Horizontal lines divide the table based on the location of the variant. Mismatch plots of those variants are shown in Figure 5.32.

et al. very well and the variant is therefore a likely pre-leukaemic mutation of the type described above.

5.5.3.4 Allelic frequency distribution analysis

The analysis of the distributions of allelic frequencies along the genome using the classifier described in Section 5.3.1.6 yielded evidence of the copy-number losses that were also evident from the analysis of the $\log_2$-ratios. No patients with significant homozygous regions in their control samples were identified, except for a small (less than 10 kilo-bases) region on chromosome 17 in the control sample of patient *KM80*. Figure 5.28 illustrates the samples and genomic regions that have been identified as homozygous in this cohort. There was significant overlap between the regions classified as homozygous and regions in which the $\log_2$-ratios indicated copy-number losses, i.e. the majority of homozygous regions were simple the result of deletion events that remove one of the two alleles in the affected region completely.

In Figures 5.29, 5.30 and 5.31 I show a visualisation of the homozygousity classifier for chromosomes containing homozygous regions alongside their $\log_2$-ratio plots. Here it is apparent that the homozygous regions on chromosomes 5, 7 and 18 in the AML samples of patient *KM115* are the results of deletion events leading to single copy regions which are then obviously homozygous (see Figure 5.29). In Figure 5.30 (showing data from the samples of patient *KM89*) it is apparent that the homozygous region on chromosome 7 is the result of a deletion event with a $\log_2$-ratio of ~ -1, i.e. a deletion that affects almost all cancer cells and is therefore a

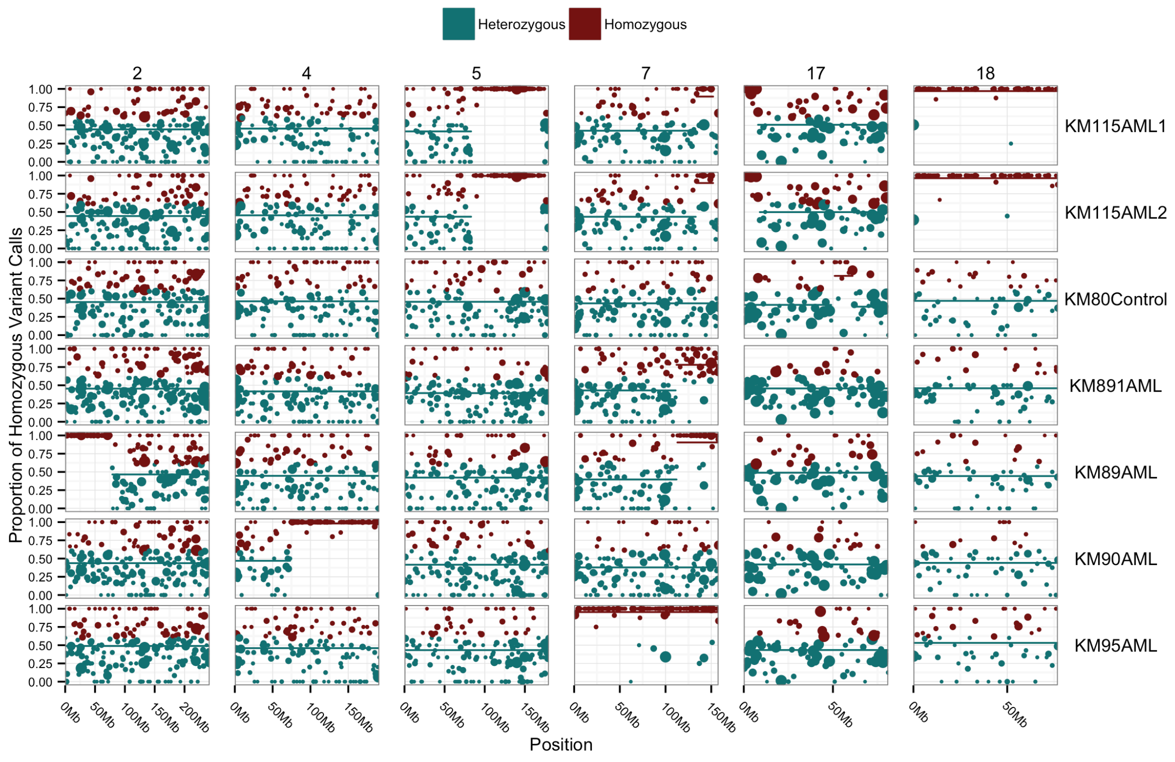


Figure 5.28: Overview plot showing chromosomes 2, 4, 5, 7, 17 and 18 in horizontal panels and a selection of samples that show large homozygous regions on those chromosomes in the vertical panels. The plot is essentially a multi-chromosome version of Figure 5.12, where dots represent 1 mega-base bins. The genomic position is shown on the x-axis and the percentage of homozygous variant calls on the y-axis. Red dots and their respective red segmentations (shown as horizontal lines) indicate homozygous region in the respective sample and chromosome.

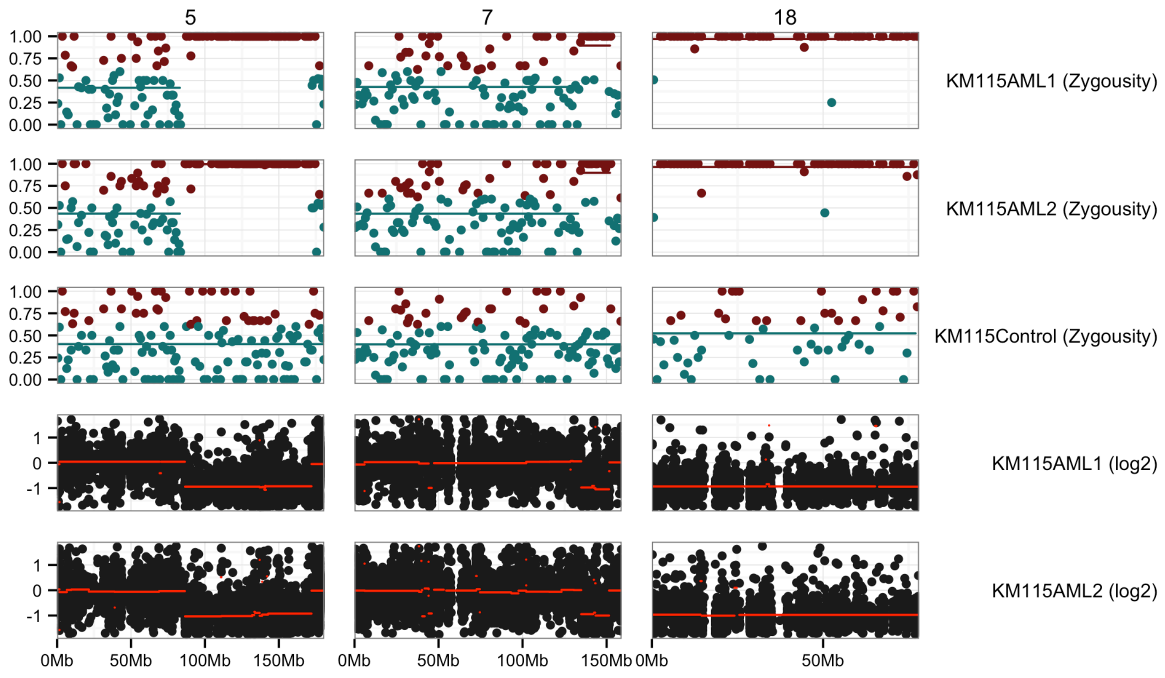


Figure 5.29: Plot showing homozygousity classifier results (top three panels) and $\log_2$-ratios (two lower panels) of samples from patient *KM115*. Genomic position is shown on the x-axis and the percentage of homozygous variants, and the $\log_2$-ratio of the matching tumour-normal comparison is shown on the y-axis in the respective tracks. Note how the homozygous regions (reg coloured segments in the upper tracks), correspond to deletion events as identified by the $\log_2$-ratios dropping to around -1 in the bottom panels. Here we show three chromosomes with large homozygous regions in the samples of patient *KM115*. Those are caused by large deletion events on chromosomes 5 and 7 and a complete loss of one copy of chromosome 18.

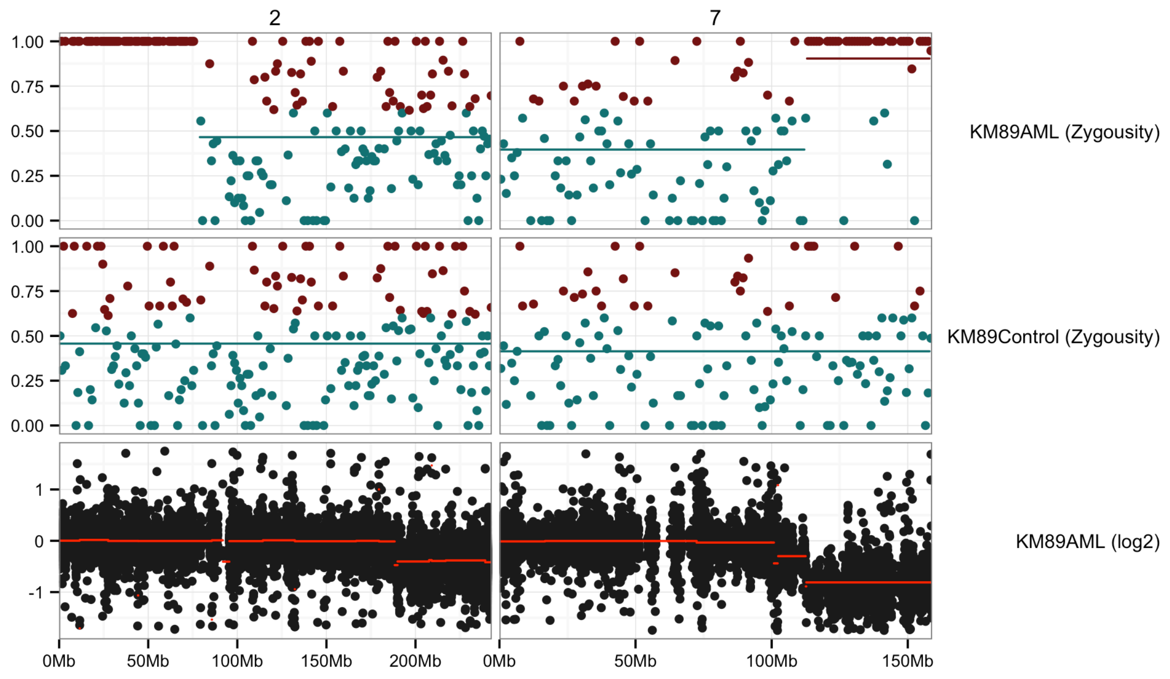


Figure 5.30: Overview plot of a copy-number neutral LoH event on chromosome 2 in sample *KM89AML* as well as a large deletion event on chromosome 7. Note also the large deletion near the end of chromosome 2, which does not cause a homozygous region because it is only present in a subclone of the population. The plot is of the same type as the one shown in Figure 5.29.

likely early event and potential driver of the cancer. The result of this deletion is a single-copy region of the genome which is homoyzgous near the end of chromosome 7. On chromosome 2 a homozygous region corresponding to roughly the first 75 mega-bases of the chromosome, with no corresponding change in $\log_2$-ratio, can be observed. This is the typical pattern of a copy-number neutral LoH event. Also note that the deletion near the end of chromosome 2 has no corresponding homozygous region, which is due to the fact that the event is only present in about half of the sequenced cells ($\log_2$-ratio of ~ -0.5) and the variant calling algorithm (which works on the combined data of the bulk sequencing of all the cells) can not detect homozygous events there. Figure 5.31 shows clear indications of a copy-number neutral LoH event on the long arm of chromosome 4 that affects the AML sample of patient *KM90*.

5.5.4 Discussion

5.5.4.1 Causative mutations

Looking at the molecular biology results shown in the last column of Table 5.10 we can conclude that amongst the AML samples a likely causative mutation has been found in patients *KM103*, *KM80*, *KM87*, *KM88*, *KM90* and *KM93*.

Amongst the AML samples in which a causative mutation could not be detected, deletions on chromosomes 5 and 7 that overlap with the genes *MLL3*, *PURA* and *PURB* were found. Tak-

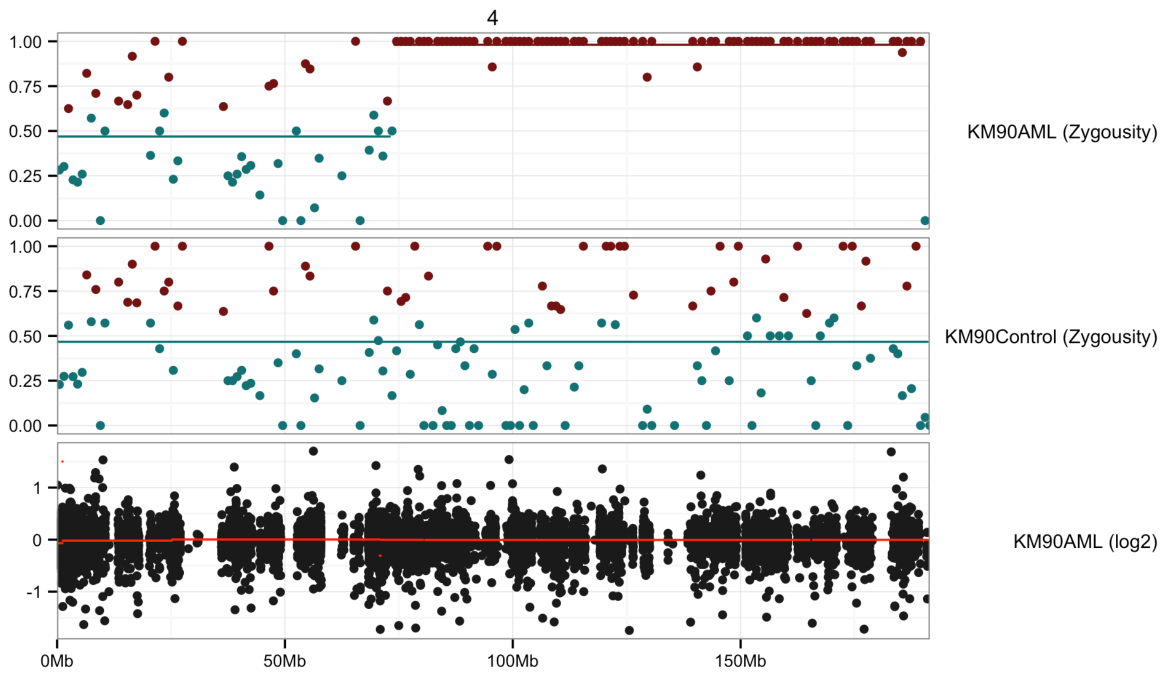


Figure 5.31: Copy-number neutral LoH event on the long arm of chromosome 4 in sample *KM90AML*. The plot is of the same type as the one shown in Figures 5.29 and 5.29.

ing into account the findings of Chen et al. and Lezon-Geyda et al. and the results shown in Figures 5.23 and 5.24 it is apparent that those deletion events constitute likely causative mutation for those AMLs. Furthermore, the events affecting *MLL3*, *PURA* and *PURB* are present in the majority of sequenced cells and are therefore likely early events in the cancer progression. Through this simple analysis of common AML mutations by PCR and the identification of copy number changes through the analysis of $\log_2$-ratios we could identify for each of the AML samples a good candidate causative mutation, providing insights into the drivers of those AMLs. This kind of analysis, if implemented as a clinical assay, can help inform treatment decisions in newly admitted AML patients.

5.5.4.2 Pre-Leukaemic mutations in *DNMT3A* and *IDH1/2*

Prompted by previous reports by Corces-Zimmerman et al. I investigated the presence of pre-leukaemic mutations in *DNMT3A* in the control samples of this cohort. Such mutations can increase the ability for self-renewal and inhibit the differentiation of haematopoietic stem cells. Such properties are beneficial to a prospective cancer cell and the acquisition of a *DNMT3A* mutation can therefore be seen as a first step in the progression of a normal haematopoietic stem cell on the path to a leukaemia. When a cell starts acquiring such mutations this can lead to its clonal expansion which creates a population of pre-leukaemic stem cells (pre-LSCs). The *DNMT3A* mutations in such populations will show up with less than 50% allelic frequency in the control samples of the affected patients and will be present in the majority of cancer cells since they are essential for the development of the cancer. An overview of all *DNMT3A* mutations I found is shown in Table 5.13. Figure 5.32 shows mismatch plots of the mutations I found to be present in the pre-leukaemic state in control samples of patients in the cohort. In some cases one can observe the progression from a low frequency pre-leukaemic state to a high-frequency cancer state representing a fully developed leukaemia with the mutation present in the majority (if not all of the) cells.

There are some patients where the mutations exist in the AML samples but there is no indication at all that they were present in the control sample at significant proportions. This is the case in the samples from patients *KM87* (Figure 5.32 and Table 5.13). Such behaviour is not a problematic result, since it is entirely possible that the cancer cells acquired the *DNMT3A* mutations and the other necessary driver mutations within a very short timeframe, and the time spent in the pre-leukaemic state was therefore too short for the mutations to be observable at significant proportions in the control sample. In principle the same mutations that are beneficial for a cancer cell in the sequential progression model can also beneficial to a cancer cell if they are acquired all at once.

For patient *KM90* a very small amount of support (2 reads only) for a variant can be observed on the AML sample. Given the extremely small allelic frequency of this mutation it stands to reason, that it is not significantly involved in the development and progression of this specific cancer.

In the samples of patient *KM80* one can observe the mutation at about 21% frequency in the control and closer to 40% in the AML, representing a very typical example of the expansion of a pre-leukaemic mutation. Here the 21% of cells in the control sample that carry the mutation are likely pre-LSCs of the leukaemia the patient developed later on.

In the AML sample of patient *KM89* the pre-leukaemic mutation from the control is actually present in a homouzygous state with allele frequency 94%

5.5.4.3 Copy-number neutral loss of heterozygousity events

Through the analysis of the allelic frequency distribution in single sample variant calls combined with the copy-number analysis based on $\log_2$-ratios of coverage I could identify copy-number neutral LoH events on chromosome 4 in sample *KM90AML* and chromosome 2 in sample *KM89AML* (Figures 5.31 and 5.30). Their importance to the development of the cancers remains unclear and one might speculate that if cancer-driving mutations were to exist in the affected regions, then a copy-number neutral LoH event could eliminate the second (healthy) copy of an affected gene and replace it with a mutated copy, making the locus homozygous for the mutated genotype. Such behaviour has been described previously in AMLs with *FLT3* internal tandem duplications, where the region containing *FLT3* undergoes copy-number neutral LoH which yields two mutated copies of *FLT3*, resulting in an unfavourable clinical prognosis (Whitman et al., 2001). In our data we can see such an effect for the *DNMT3A* mutation in patient *KM89*. *DNMT3A* overlaps with the copy-number-neutral LoH event on chro-

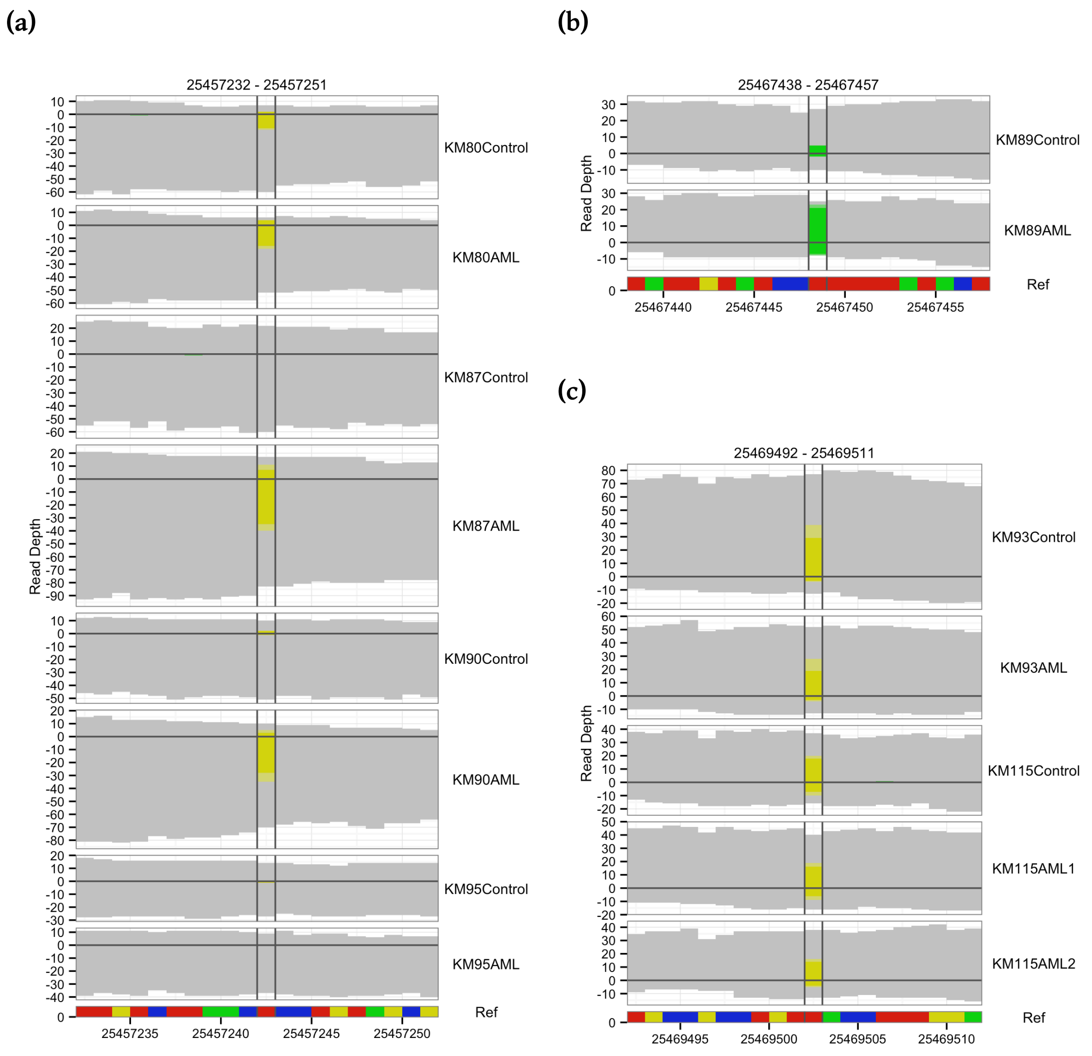


Figure 5.32: The *DNMT3A* mutations found as germ-line as well as somatic mutations in this cohort. Mismatch plots are shown for the three mutated positions listed in 5.13 showing all samples of patients that carry each mutation in at least one of their samples. **(a)** `2:25457242 C->T`; **(b)** `2:25467448 C->A`; **(c)** `2:25469502 C->T`;

mosome 2 in this patient, and the mutation `2:25467448 C -> A` is observed in the control sample at allelic frequency 19% and in the AML sample at 95%. In this case the LoH event has made a pre-leukaemic mutation homozygous in the AML.

5.5.4.4 Tracking drivers of metastasis through mouse xenografts

The part of the project that deals with tracking the progression of allelic frequencies of mutations through a series of xenografts is still ongoing and I expect arrival of the targeted sequencing data for the xenograft series by the end of the year. So far I have established a pipeline for somatic variant calling in the samples from this project and my collaborators have established an accompanying PCR protocol to experimentally validate the somatic variant calls. We are therefore in a position where we can properly call and validate somatic variants once the data arrives.

Furthermore all the parts of the analysis pipeline to quickly generate allele frequency profiles for somatic variants in the time series of xenografts are in place. This will allow us to distinguish variants that convey selective advantages to the cancer cells in the xenograft setting, which is a good model for the formation of metastases.

6 Conclusion

During my time as a PhD student I have worked on various projects related to the analysis of sequencing data. Usually this involved human samples of cancers and their matched controls. I have encountered a number of shortcomings of the available software infrastructure when dealing with larger datasets from sequencing multiple human samples and published some of my attempts at improving the way such data is analysed in two papers describing the HTSeq Python Module (Anders et al., 2014) and the **h5vc** R/Bioconductor package (Pyl et al., 2014). I have also been a shared first author with Jonathan Landry on the HeLa Genome sequencing project (Landry et al., 2013), which was one of the earlier sequencing projects I worked on. I have developed methods for analysing large cohorts of cancer samples in an efficient and intuitive way and made those methods available through public open-source software packages that can be easily used and vetted by the research community. I am committed to keeping those software packages updated (specifically **h5vc**, which is part of the Bioconductor repository) and I was engaged in teaching their usage at the CSAMA[1] conference 2014 in Brixen, Italy. I am working on improving **h5vc** with every release cycle of R/Bioconductor (every 6 months). So far I have used the software for cohorts up to 198 samples of WES in 98 pairs of human cancer samples and their respective controls. This work was a collaboration in a project of my colleague Małgorzata Oleś who is working on the analysis of this dataset. I know of users of the **h5vc** package that are analysing up to 300 human whole exomes with it and in the context of the Huber group's involvement with the ICGC Pan-Cancer project I will test out my software on an initial cohort of 50 tumour-normal sample pairs that have been sequenced with WGS technology. Ultimately I am aiming at generating HDF5-based nucleotide tallies for the complete Pan-Cancer dataset, which will encompass thousands of samples from the TCGA[2] and ICGC[3] projects.

In collaborations with groups at the National Center for Tumour Disease and the Universitätsklinikum Heidelberg I have worked on different cohorts of paired cancer samples, focussing on blood cancers (AMLs and T-ALLs). We have not published any of the results of those projects yet, as some analyses are still ongoing. However, the results of my analysis so far yielded a wealth of interesting biological findings, including, for example, the presence of pre-leukaemic mutations in matched control samples of AMLs.

I am convinced, that my contributions to the set of available computational tools in the field of cancer sequencing analysis will enable more researchers to analyse their data in a more intuitive fashion. The advantages of using HDF5-based nucleotide tallies and **h5vc** will

[1]`http://www-huber.embl.de/csama/index.html`
[2]`http://cancergenome.nih.gov/`
[3]`https://icgc.org/`

become even more apparent as the size of sequencing data-sets increases.

6.1 Acknowledgements

I would like to thank my supervisors and colleagues for providing an inviting and enjoyable work atmosphere and providing insights and ideas in many fruitful discussions both scientific and otherwise. I would also like to thank everyone that supported me on my way, especially my parents. Special thanks go to Jenny for providing helpful insights from a different perspective and proof-reading.

Bibliography

Abecasis GR, Auton A, Brooks LD, DePristo MA, Durbin RM, et al. An integrated map of genetic variation from 1,092 human genomes. *Nature*, 491(7422):56–65 (2012).

Adey A, Burton JN, Kitzman JO, Hiatt JB, Lewis AP, et al. The haplotype-resolved genome and epigenome of the aneuploid HeLa cancer cell line. *Nature*, 500(7461):207–211 (2013).

Alexandrov LB, Nik-Zainal S, Wedge DC, Aparicio SA, Behjati S, et al. Signatures of mutational processes in human cancer. *Nature*, 500(7463):415–421 (2013).

Anders S and Huber W. Differential expression analysis for sequence count data. *Genome Biol.*, 11(10):R106 (2010).

Anders S, Reyes A, and Huber W. Detecting differential usage of exons from RNA-seq data. *Genome Res.*, 22(10):2008–2017 (2012).

Anders S, Pyl PT, and Huber W. HTSeq: A Python framework to work with high-throughput sequencing data. *bioRxiv* (2014). doi:10.1101/002824. Accepted for publication in Bioinformatics.

Bao Y, Federhen S, Leipe D, Pham V, Resenchuk S, et al. National center for biotechnology information viral genomes project. *J. Virol.*, 78(14):7291–7298 (2004).

Battiwalla M. Abnl(17p) in AML: who will guard the guardian? *Blood*, 123(19):2906–2907 (2014).

Bayer R. Symmetric binary b-trees: Data structure and maintenance algorithms. *Acta Informatica*, 1(4):290–306 (1972). ISSN 0001-5903. doi:10.1007/BF00289509.

Beazley DM. Swig: An easy to use tool for integrating scripting languages with c and c++. In *Proceedings of the 4th Conference on USENIX Tcl/Tk Workshop, 1996 - Volume 4*, TCLTK'96, pages 15–15. USENIX Association, Berkeley, CA, USA (1996).

Becker P. Iso/iec 14882:2011 programming language c++ (2011).

Behnel S, Bradshaw R, Citro C, Dalcin L, Seljebotn D, et al. Cython: The best of both worlds. *Computing in Science Engineering*, 13(2):31 –39 (2011). ISSN 1521-9615. doi:10.1109/MCSE.2010.118.

Bischl B, Lang M, Mersmann O, Rahnenfuehrer J, and Weihs C. Computing on high performance clusters with R: Packages batchjobs and batchexperiments. Technical Report 1, TU Dortmund (2011).

Boztug K, Schmidt M, Schwarzer A, Banerjee PP, Diez IA, et al. Stem-cell gene therapy for the Wiskott-Aldrich syndrome. *N. Engl. J. Med.*, 363(20):1918–1927 (2010).

Carlson M. *TxDb.Hsapiens.UCSC.hg19.knownGene: Annotation package for TranscriptDb object(s)* (2011). R package version 2.14.0.

Chen C, Liu Y, Rappaport AR, Kitzing T, Schultz N, et al. MLL3 is a haploinsufficient 7q tumor suppressor in acute myeloid leukemia. *Cancer Cell*, 25(5):652–665 (2014).

Cibulskis K, Lawrence MS, Carter SL, Sivachenko A, Jaffe D, et al. Sensitive detection of somatic point mutations in impure and heterogeneous cancer samples. *Nat Biotech*, 31(3):213–219 (2013). ISSN 1087-0156.

Cleveland WS. Lowess: A program for smoothing scatterplots by robust locally weighted regression. *The American Statistician*, 35(1):54 (1981).

Corces-Zimmerman MR, Hong WJ, Weissman IL, Medeiros BC, and Majeti R. Preleukemic mutations in human acute myeloid leukemia affect epigenetic regulators and persist in remission. *Proc. Natl. Acad. Sci. U.S.A.*, 111(7):2548–2553 (2014).

Danecek P, Auton A, Abecasis G, Albers CA, Banks E, et al. The variant call format and VCFtools. *Bioinformatics*, 27(15):2156–2158 (2011).

DePristo MA, Banks E, Poplin R, Garimella KV, Maguire JR, et al. A framework for variation discovery and genotyping using next-generation DNA sequencing data. *Nat. Genet.*, 43(5):491–498 (2011).

Ding L, Ley TJ, Larson DE, Miller CA, Koboldt DC, et al. Clonal evolution in relapsed acute myeloid leukaemia revealed by whole-genome sequencing. *Nature*, 481(7382):506–510 (2012).

Eddelbuettel D and François R. Rcpp: Seamless R and C++ integration. *Journal of Statistical Software*, 40(8):1–18 (2011).

Fan HC, Wang J, Potanina A, and Quake SR. Whole-genome molecular haplotyping of single cells. *Nat. Biotechnol.*, 29(1):51–57 (2011).

Fischer B and Pau G. HDF5 interface to R (2012).

Forbes SA, Bindal N, Bamford S, Cole C, Kok CY, et al. COSMIC: mining complete cancer genomes in the Catalogue of Somatic Mutations in Cancer. *Nucleic Acids Res.*, 39(Database issue):D945–950 (2011).

Garnier R and Taylor J. *Discrete Mathematics: Proofs, Structures and Applications, Third Edition*. CRC Press (2009). ISBN 978-1-4398-1280-8.

Gerstung M, Beisel C, Rechsteiner M, Wild P, Schraml P, et al. Reliable detection of subclonal single-nucleotide variants in tumour cell populations. *Nat Commun*, 3:811 (2012).

Huntley MA, Larson JL, Chaivorapol C, Becker G, Lawrence M, et al. ReportingTools: an automated result processing and presentation toolkit for high-throughput genomic analyses. *Bioinformatics*, 29(24):3220–3221 (2013).

Kandoth C, McLellan MD, Vandin F, Ye K, Niu B, et al. Mutational landscape and significance across 12 major cancer types. *Nature*, 502(7471):333–339 (2013).

Kay MA. State-of-the-art gene-based therapies: the road ahead. *Nat. Rev. Genet.*, 12(5):316–328 (2011).

Kent WJ, Zweig AS, Barber G, Hinrichs AS, and Karolchik D. BigWig and BigBed: enabling browsing of large distributed datasets. *Bioinformatics*, 26(17):2204–2207 (2010).

Krzywinski M, Schein J, Birol I, Connors J, Gascoyne R, et al. Circos: an information aesthetic for comparative genomics. *Genome Res.*, 19(9):1639–1645 (2009).

Kumar P, Henikoff S, and Ng PC. Predicting the effects of coding non-synonymous variants on protein function using the SIFT algorithm. *Nat Protoc*, 4(7):1073–1081 (2009).

Lai D, Ha G, and Shah S. Hmmcopy: Copy number prediction with correction for gc and mappability bias for hts data (2012).

Landry JJ, Pyl PT, Rausch T, Zichner T, Tekkedil MM, et al. The genomic and transcriptomic landscape of a HeLa cell line. *G3 (Bethesda)*, 3(8):1213–1224 (2013).

Larson DE, Harris CC, Chen K, Koboldt DC, Abbott TE, et al. SomaticSniper: identification of somatic point mutations in whole genome sequencing data. *Bioinformatics*, 28(3):311–317 (2012).

Lawrence M, Huber W, Pages H, Aboyoun P, Carlson M, et al. Software for computing and annotating genomic ranges. *PLoS Comput. Biol.*, 9(8):e1003118 (2013).

Lezon-Geyda K, Najfeld V, and Johnson EM. Deletions of PURA, at 5q31, and PURB, at 7p13, in myelodysplastic syndrome and progression to acute myelogenous leukemia. *Leukemia*, 15(6):954–962 (2001).

Li FP and Fraumeni JF. Soft-tissue sarcomas, breast cancer, and other neoplasms. A familial syndrome? *Ann. Intern. Med.*, 71(4):747–752 (1969).

Li H and Durbin R. Fast and accurate short read alignment with Burrows-Wheeler transform. *Bioinformatics*, 25(14):1754–1760 (2009).

Li H, Handsaker B, Wysoker A, Fennell T, Ruan J, et al. The Sequence Alignment/Map format and SAMtools. *Bioinformatics*, 25(16):2078–2079 (2009).

Love M. exomecopy: Copy number variant detection from exome sequencing read depth. (2013).

Macville M, Schrock E, Padilla-Nash H, Keck C, Ghadimi BM, et al. Comprehensive and definitive molecular cytogenetic characterization of HeLa cells by spectral karyotyping. *Cancer Res.*, 59(1):141–150 (1999).

McKenna A, Hanna M, Banks E, Sivachenko A, Cibulskis K, et al. The Genome Analysis Toolkit: a MapReduce framework for analyzing next-generation DNA sequencing data. *Genome Res.*, 20(9):1297–1303 (2010).

McLaren W, Pritchard B, Rios D, Chen Y, Flicek P, et al. Deriving the consequences of genomic variants with the Ensembl API and SNP Effect Predictor. *Bioinformatics*, 26(16):2069–2070 (2010).

Mitra AB, Murty VV, Li RG, Pratap M, Luthra UK, et al. Allelotype analysis of cervical carcinoma. *Cancer Res.*, 54(16):4481–4487 (1994).

Mockler TC, Chan S, Sundaresan A, Chen H, Jacobsen SE, et al. Applications of DNA tiling arrays for whole-genome analysis. *Genomics*, 85(1):1–15 (2005).

Morgan M, Lang M, and Thompson R. Biocparallel: Bioconductor facilities for parallel evaluation. (2013).

Mullokandov MR, Kholodilov NG, Atkin NB, Burk RD, Johnson AB, et al. Genomic alterations in cervical carcinoma: losses of chromosome heterozygosity and human papilloma virus tumor status. *Cancer Res.*, 56(1):197–205 (1996).

Neumann M, Heesch S, Schlee C, Schwartz S, Gokbuget N, et al. Whole-exome sequencing in adult ETP-ALL reveals a high rate of DNMT3A mutations. *Blood*, 121(23):4749–4752 (2013).

Olshen AB, Venkatraman ES, Lucito R, and Wigler M. Circular binary segmentation for the analysis of array-based DNA copy number data. *Biostatistics*, 5(4):557–572 (2004).

Park MJ, Taki T, Oda M, Watanabe T, Yumura-Yagi K, et al. FBXW7 and NOTCH1 mutations in childhood T cell acute lymphoblastic leukaemia and T cell non-Hodgkin lymphoma. *Br. J. Haematol.*, 145(2):198–206 (2009).

Pyl PT, Gehring J, Fischer B, and Huber W. h5vc: scalable nucleotide tallies with HDF5. *Bioinformatics*, 30(10):1464–1466 (2014).

Rader JS, Gerhard DS, O'Sullivan MJ, Li Y, Li L, et al. Cervical intraepithelial neoplasia III shows frequent allelic loss in 3p and 6p. *Genes Chromosomes Cancer*, 22(1):57–65 (1998).

Rausch T, Zichner T, Schlattl A, Stutz AM, Benes V, et al. DELLY: structural variant discovery by integrated paired-end and split-read analysis. *Bioinformatics*, 28(18):i333–i339 (2012).

Rosenbloom KR, Dreszer TR, Long JC, Malladi VS, Sloan CA, et al. ENCODE whole-genome data in the UCSC Genome Browser: update 2012. *Nucleic Acids Res.*, 40(Database issue):D912–917 (2012).

Ross JA, Linabery AM, Blommer CN, Langer EK, Spector LG, et al. Genetic variants modify susceptibility to leukemia in infants: a Children's Oncology Group report. *Pediatr Blood Cancer*, 60(1):31–34 (2013).

Saiki RK, Gelfand DH, Stoffel S, Scharf SJ, Higuchi R, et al. Primer-directed enzymatic amplification of DNA with a thermostable DNA polymerase. *Science*, 239(4839):487–491 (1988).

Saunders CT, Wong WS, Swamy S, Becq J, Murray LJ, et al. Strelka: accurate somatic small-variant calling from sequenced tumor-normal sample pairs. *Bioinformatics*, 28(14):1811–1817 (2012).

Schmidt M, Schwarzwaelder K, Bartholomae C, Zaoui K, Ball C, et al. High-resolution insertion-site analysis by linear amplification-mediated PCR (LAM-PCR). *Nat. Methods*, 4(12):1051–1057 (2007).

Schroth GP. E-mtab-513 - rna-seq of human individual tissues and mixture of 16 tissues (illumina body map) (2011).

Seshan V and Olshen A. Dnacopy: Dna copy number data analysis (2006).

Simonis M, Klous P, Splinter E, Moshkin Y, Willemsen R, et al. Nuclear organization of active and inactive chromatin domains uncovered by chromosome conformation capture-on-chip (4C). *Nat. Genet.*, 38(11):1348–1354 (2006).

Smigielski EM, Sirotkin K, Ward M, and Sherry ST. dbSNP: a database of single nucleotide polymorphisms. *Nucleic Acids Res.*, 28(1):352–355 (2000).

Smit A, Hubley R, and Green P. Repeatmasker (2007).

Stephens PJ, Greenman CD, Fu B, Yang F, Bignell GR, et al. Massive genomic rearrangement acquired in a single catastrophic event during cancer development. *Cell*, 144(1):27–40 (2011).

Strefford JC, Worley H, Barber K, Wright S, Stewart AR, et al. Genome complexity in acute lymphoblastic leukemia is revealed by array-based comparative genomic hybridization. *Oncogene*, 26(29):4306–4318 (2007).

Takeuchi S, Koike M, Seriu T, Bartram CR, Slater J, et al. Homozygous deletions at 9p21 in childhood acute lymphoblastic leukemia detected by microsatellite analysis. *Leukemia*, 11(10):1636–1640 (1997).

The HDF Group. Hierarchical data format version 5, 2000-2010 (2010).

Thorvaldsdottir H, Robinson JT, and Mesirov JP. Integrative Genomics Viewer (IGV): high-performance genomics data visualization and exploration. *Brief. Bioinformatics*, 14(2):178–192 (2013).

Whitman SP, Archer KJ, Feng L, Baldus C, Becknell B, et al. Absence of the wild-type allele predicts poor prognosis in adult de novo acute myeloid leukemia with normal cytogenetics and the internal tandem duplication of FLT3: a cancer and leukemia group B study. *Cancer Res.*, 61(19):7233–7239 (2001).

Wu TD and Nacu S. Fast and SNP-tolerant detection of complex variants and splicing in short reads. *Bioinformatics*, 26(7):873–881 (2010).

Ye K, Schulz MH, Long Q, Apweiler R, and Ning Z. 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).

Zerbino DR and Birney E. Velvet: algorithms for de novo short read assembly using de Bruijn graphs. *Genome Res.*, 18(5):821–829 (2008).

Zhao Z, Tavoosidana G, Sjolinder M, Gondor A, Mariano P, et al. Circular chromosome conformation capture (4C) uncovers extensive networks of epigenetically regulated intra- and interchromosomal interactions. *Nat. Genet.*, 38(11):1341–1347 (2006).