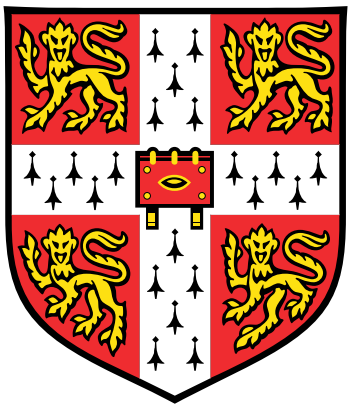
Estimating telomere length from whole genome sequencing data



J. H. R. Farmery

CRUK: CI
University of Cambridge

This dissertation is submitted for the degree of
Doctor of Philosophy

Christ's College May 2018

For my Parents who put me on the right track...

...and for Alexandra who kept me on it

Declaration

This dissertation is the result of my own work and includes nothing which is the outcome of work done in collaboration except where specifically indicated in the text.

It is not substantially the same as any that I have submitted, or, is being concurrently submitted for a degree or diploma or other qualification at the University of Cambridge or any other University or similar institution except as declared in the Preface and specified in the text. I further state that no substantial part of my dissertation has already been submitted, or, is being concurrently submitted for any such degree, diploma or other qualification at the University of Cambridge or any other University of similar institution except as declared in the Preface and specified in the text.

This dissertation does not exceed the 60,000 word limit prescribed by the Clinical Medicine and Clinical Veterinary Medicine Degree Committee.

J. H. R. Farmery
May 2018

Acknowledgements

I would first like to acknowledge my parents, Moira and David, as well as my sisters, Olivia and Sophie, for their limitless support and encouragement. Also Libor Juranek and Alexandra Weissova Snr. for their hospitality and kindness throughout my time at Cambridge. Thanks are also due to my friends, in Cambridge and elsewhere, for being such great chums.

This work would not have been possible without the generosity and patience of so many of my expert colleagues. Thanks particularly to James Thaventhiran, Charlie Massie, Hana Lango-Allen, Ed Williams, Mike Smith and also the Tavaré group, for all their help over the years. Thanks also to Ann Kaminski and Barbara Houlihan who have been great sources of kindness within the institute.

I am grateful to Andy Lynch for taking me on as his first PhD Student. His eye for detail and seemingly inexhaustible statistical knowledge have been invaluable. Thanks also to Simon Tavaré for his support and worldly advice.

Lastly, I wish to acknowledge my fiancé, Alexandra Weissova, without whom I could not have come this far. She has picked me up on so many occasions and, in these four years of hard work, I have only found more to love about her.

Summary

Estimating telomere length from whole genome sequencing data

J. H. R. Farmery

This thesis details the development of two computational tools, Telomerecat and Parabam, as well as their applications to whole genome sequencing (WGS) data.

Telomerecat is a tool for estimating telomere length from WGS data. The strength of Telomerecat lies in its applicability. This applicability is due to a number of advantages over previous attempts to estimate telomere length from WGS. Chief amongst these advantages is that it makes no assumption about the underlying chromosome count or size of the genome within input samples. This means that Telomerecat lends itself well to analysing cancer samples where such assumptions are unfounded. This also means it is applicable to non-human samples, a first for tools of its kind. Furthermore, a novel method for filtering reads derived from interstitial telomere sequences means that it does not rely on previously applied analyses, a source of bias.

The other tool described in this thesis is Parabam. Parabam is the first tool of its kind to allow users to apply a function to all of the reads in sequence alignment files, in parallel. Furthermore, Parabam includes a novel method for iterating over index sorted sequence files as if they were name sorted. We provide evidence that Parabam is a quicker way to create complex subsets and statistics from sequence alignment files.

In the latter half of the thesis we detail two applications of Telomerecat to large scale WGS projects. The first application, to the Prostate ICGC UK cohort, unveils hitherto uncovered associations between telomere length and previously identified molecular subtypes as well as cancer stage. In the second application, to the NIHR BioResource - Rare Disease cohort, we discover a previously unidentified variant in DKC1 that we propose is directly linked to short telomeres and an immunodeficient phenotype.

Table of Contents

List of Figures

List of Algorithms

List of Tables

Chapter 1

Introduction

By now it is a cliché to use the opening sentence of a thesis to proclaim how whole genome sequencing has transformed biology. It is abundantly clear that biology has been transformed. The challenge to computational biologists is now to tame this brave new data and to devise ever more precise and efficient methods to improve our science.

Since the inception of whole genome sequencing (WGS), perhaps best embodied by assembly of the human genome in 2001, a great many computational methods have been developed to interrogate the genome and provided ever more insight into the workings of the cell and the origins of disease. Relatively few of these methods have focused on the extremely repetitive regions of the genome.

This thesis presents two new tools in the form of computer programs. The first is Parabam. Parabam allows the user to apply functions to all reads in a WGS file in parallel, useful when dealing with repetitive sequences which are otherwise hard to summon from the file. The second tool is Telomerecat. Telomerecat provides estimates of the length of telomeres, a highly repetitive region of the genome, from WGS data. Both tools work in synergy throughout this thesis, as Telomerecat uses Parabam as an interface for processing WGS data.

As well as these tools, we present two applications of the Telomerecat method to cohorts of WGS and associated genomic data. In both applications we use Telomerecat to uncover novel associations between disease characteristics and telomere length. The first of these applications is to a set of 192 prostate cancer and associated blood normal samples from the UK wing of the International Cancer Genomics Consortium (ICGC) prostate study. The second is an application to over five thousand samples from the NIHR BioResource - Rare Disease cohort.

In the remainder of this chapter we will review the core concepts underlying the thesis. Where better place to start in this endeavour than at the ends?

1.1 Telomeres

Deoxyribonucleic acid (DNA) is the instruction manual of the cell. DNA is comprised of a chain of base-pairs, individual molecules that can be thought of as letters in the code. These base-pairs are Adenine, Thymine, Cytosine and Guanine but are more commonly referred to by their initials: A, T, C and G. The beginning of a DNA molecule is referred to as the "5 prime end" and the end is referred to as the "3 prime end".

The primary unit of hereditary information within DNA is the gene and these genes are organised into stretches of DNA called chromosomes. In some species, particularly in bacteria, chromosomes are circular structures. However, all eukaryotes (a branch of the tree of life that includes all plants and animals) have linear chromosomes. Analogous to an exceedingly tightly coiled loose piece of string, linear chromosomes are unattached at either end.

Telomere is a highly repetitive stretch of DNA found at the ends of linear chromosomes. Here we describe how telomeres were discovered, their function within the cell and their role in cancer.

1.1.1 A brief history

The first person to use the word telomere[1] in print was Herman Muller. His work in bombarding the DNA of *Drosophila* with X-rays and observing the resultant damage led him to the reason that the ends of chromosomes must possess some protective structure. In his article in The Collecting Net he refers to this "terminal gene" as the telomere (MULLER, 1938).

Simultaneously to Muller's work, Barbara McClintock observed that maize chromosomes that had suffered damage would bind to one another to form a breakage-fusion-bridge (MCCLINTOCK, 1941). From these observations she reasoned that linear chromosomes must have some form of protective structure to ensure the ends of intact chromosomes did not suffer the same fate. She drew upon her previous work to define these ends of chromosomes as functionally and characteristically separate to that of the rest of the chromosome.

Over the succeeding decades, the idea that telomere may have some role beyond the protection of chromosome ends, came into focus. In the 1960s Leonard Hayflick observed that human cells in culture have limited replicative potential (HAYFLICK and MOORHEAD, 1961). A cell is said to have become senescent[2] once it ceases to divide and replicate. Today,

[1] From the Greek words *telo* meaning end and *mere* meaning part

[2] Latin for "to grow old"

the number of cell doublings that a population can endure before entering senescence is referred to as the "Hayflick limit".

It was some time before telomeres were directly implicated as part of this process. In the early 1970s Alexy Olovnikov drew a link between the shortening of telomeres (or as he called them: "telogenes") as a result of replication (a process he refers to as "marginotomy") and the signalling of senescence. Olovnikov even goes as far as to speculate that this process was a cause in disorders of ageing (OLOVNIKOV, 1973).

In 2009 the Nobel prize was awarded jointly to Elizabeth Blackburn, Carol Grieder and Jack Szostak for their contributions to the understanding of telomere. From the late 1980s, individually and in collaboration with one another, they shed considerable light on the role and function of telomere. Perhaps most notably they uncovered the sequence of telomere (SHAMPAY *et al.*, 1984) and telomerase, a protein that appends telomere to the ends of chromosomes encoded by the gene TERT (GREIDER and BLACKBURN, 1985).

Telomeres are still an area of active research. Since the discovery of telomerase, other structures have been identified as interacting directly with telomere. This includes shelterin, a complex of proteins that sheath mammalian telomeres and are thought to assist in their maintenance (LIU *et al.*, 2004). Furthermore, a transcriptional role for the telomere has been described with the discovery of TERRA, an RNA derived from within the telomere itself that can inhibit the function of telomerase (AZZALIN *et al.*, 2007). Similarly the G-quadraplex, a secondary structure within DNA, has been shown to preferentially inhabit the telomeres and act to down regulate telomeric lengthening by telomerase (PARKINSON *et al.*, 2002).

1.1.2 The structure of telomeres

Human telomeres (the focus of this thesis) are comprised of the nucleotide sequence TTAGGG repeated for a stretch within the region of thousands of bases (kilobases or KB for short). Their exact length can vary between chromosomes, individuals and populations, but typically telomere length in humans is thought to be within the region of 2 KB to 12 KB with an average of approximately 6 KB (although different methods for estimating length tend to attain different distributions). At the very end of the telomere the DNA becomes single stranded and forms a looped structure referred to as a T-loop. This T-loop is thought to protect telomere from recognition by DNA damage response proteins (O'SULLIVAN and KARLSEDER, 2010).

When a cell divides, all of the DNA within the cell is replicated and the telomeres are no exception. The cellular machinery that enacts this process is known as a polymerase. Because of the way that the polymerase functions, a 100-200 base pair fragment at the end of each chromosome is not copied. If it were not for telomeres then this failure to copy could

mean the loss of vital gene coding DNA. By sacrificing a small section of the telomere gene coding DNA is spared from loss.

Telomerase is a protein complex that is able to append new telomeric repeats onto the ends of chromosomes. Telomerase possess a small RNA subcomponent (known as TERC) that first identifies the telomere hexamer and then is used as a template for further repeats. Once telomerase has finished adding telomere repeats to the 3 prime ends of a DNA molecule a conventional polymerase completes the complement strand, leaving a small single strand overhang on the three prime end. Telomerase is usually only active within stem cells and gametes but cancerous cells may reactivate telomerase in order to forgo senescence.

Telomerase always appends telomeric repeats to the 3 prime end of a DNA strand. Furthermore, we can be certain that the appended sequence will always be the G rich configurations of the telomere hexamer (TTAGGG). We can be sure of this because the complementary C rich configuration (CCCTAA) is both used as the identifier and scaffold for new repeats and so the G rich configuration must be the one synthesised. This facet of telomerase biology will become important later in the thesis as it is exploited by the Telomerecat method to infer additional information regarding telomere sequencing reads.

1.1.3 The role of telomeres

As our knowledge surrounding telomeres has grown, it has become increasingly clear that telomeres are a multi-functional part of the chromosome. At once protecting DNA and providing a "molecular clock" by which the life span of a cell is determined.

In its protective capacity, the presence of telomeres serves to shield the ends of chromosomes from cellular machinery dedicated to repairing DNA damage. Without telomeres, the ends of chromosomes would be liable to treatment as a double strand breakage. Indeed, when DNA suffers damage or when a chromosome has critically short telomeres, the ends of individual chromatids can become fused by this repair machinery. This fusion is ripped apart when the cell comes to replicate, causing what is known as a breakage fusion bridge. Just as McClintock observed almost 75 years ago in her maize experiments.

This shortening ties in to the molecular clock like activity of telomeres. Exactly how telomeres contribute to the enforcement of the Hayflick limit, at least in humans, is uncertain. The most recent studies suggest that numerous factors may play a part, including chromatin changes and accumulated damage to the DNA itself (O'SULLIVAN and KARLSEDER, 2010). However, it is clear that telomere shortening plays a substantial role in this process (BODNAR *et al.*, 1998).

Cancer is a disease characterised by proliferative cell replication (HANAHAN and WEINBERG, 2000). To allow for this proliferation cancerous tumours must forgo the Hayflick

limit by moderating their telomeres. The majority of cancers (perhaps as many as 90%) do this by reactivating telomerase, often via mutations to the TERT gene (SHAY and WRIGHT, 2011). Another way that telomeres lengthen in cancer cells is via the enigmatic alternative lengthening of telomeres (ALT) pathway. The precise details surrounding ALT are still unclear. It has been hypothesised that ALT is a process of homologous recombination within and amongst telomeres in order to lengthen the sequence at terminal ends of chromosomes. ALT is thought to occur in around 10-15%, however, this rate varies according to the specific cancer type. Prostate cancer, the focus of Chapter 4 is thought to display very little ALT lengthening (HEAPHY *et al.*, 2011b).

Subtelomere and interstitial telomere The subtelomere is the region of the genome situated directly adjacent to the telomere. Historically, there has been little consensus as to where exactly the subtelomere begins and ends. Adding to the complexity is the subtelomere's seeming plasticity and proclivity for recombination (RIETHMAN *et al.*, 2005). What is clear is that at least some portion of the subtelomere bears strong resemblance to telomere, being comprised of a mixture between telomere and psuedo-telomeric repeats. To an extent the subtelomere has been mapped, however, the region most proximal to the telomere proper remain mostly uncharacterised (RIETHMAN *et al.*, 2004). As we shall see, this is the area of greatest interest to Telomerecat. Identifying the difference between subtelomere and telomere plays a central part in Chapter 3.

Interstitial telomere sequences (ITSs) are tracts of the telomere repeat found dispersed throughout the genome. They are thought to occur as a result of historic breakage and repair throughout the evolutionary process (LIN and YAN, 2008). They are of particular interest to methods estimating telomere length from WGS because of their capacity to generate information that *looks* a great deal like telomere. However, a precise method should filter out these regions as they cannot contribute to the actual telomere length. In Chapter 3 we detail a novel method for dealing with these regions.

1.1.4 Methods for estimating the length of telomeres

The first method used to estimate the length of telomeres was terminal restriction fragment Southern blot (ALLSHIRE *et al.*, 1989). This method is referred to as mTRF throughout the thesis. mTRF uses a set of restriction enzymes to cleave DNA at the subtelomere and then uses Southern blotting to ascertain the average molecular weight of the cleaved molecules. Once the average weight of the molecules is known, a length can be derived. mTRF has received wide application and is still used today. One drawback of the method is that the

restriction used to shear the telomere from the chromosome can often include substantial tracts of subtelomere.

Quantitative Polymerase Chain Reaction (qPCR) is another method used to query telomere. However, rather than report telomere length, qPCR reports the ratio of observed telomere copy number, to the copy number of a single reference gene (CAWTHON, 2002). The advantage of qPCR is that it lends itself well to high throughput analysis, indeed the two largest GWAS of telomere use this approach (POOLEY *et al.*, 2013; CODD *et al.*, 2013). However, qPCR is regarded as a less accurate measurement than mTRF (ELBERS *et al.*, 2014).

A method called qFISH has also been used to estimate telomere length. qFISH is an florescence imaging based technique that can provide a high degree of accuracy and can be used to measure the length of telomeres on individual chromosomes. However, qFISH is not well suited to high throughput analysis and requires relatively large quantities of DNA (LANSDORP *et al.*, 1996).

In the last chapter of this thesis we make use of telomere length estimates produced by flowFISH. FlowFISH leverages flow cytometry (a method of cell sorting) and fluorescence to estimate telomere length. FlowFISH has the key advantage of being a more high throughput method than either qFISH or TRF (BAERLOCHER and LANSDORP, 2003).

Not covered in this section are methods for estimating telomere length from whole genome sequencing. Section 1.3 is dedicated to this topic, which is found directly after the following introduction to WGS technology.

1.2 Whole Genome Sequencing

The aim of WGS is to determine the sequence of DNA for a given organism and then to output this in a comprehensible format, this output takes the form of a computer file. The collective name for all of the DNA in a cell is the genome. It is the aim of WGS to sequence *all* of the DNA, rather than other methods that only aim to sequence gene coding sections of DNA. Thus the name, whole genome sequencing.

Several methods for sequencing the genome are available. Amongst these methods the Illumina sequencing method is the most widely used. Indeed, Illumina has become the sequencing platform of choice for many large scale cohort studies (HUDSON *et al.*, 2010; MARX, 2015). The investigations detailed in the later chapters of this thesis are undertaken using various versions of the Illumina sequencing platform.

The resultant output of the Illumina sequencing platform is a computer file containing a multitude of sequencing reads. Each read within the experiment represents the sequence of one small fragment of DNA. Altogether, the human genome is three billion base pairs

long, however the current generation of hi-throughput sequencing technologies are capable of only producing sequences of around 100 to 150 base pairs long. In order to sequence the entire genome with enough redundancy to account for error within the reads it is necessary to create a great many of these reads. Accordingly, modern sequencing routinely produces on the order of billions of reads in one experiment. In Section 1.2.2 we will see how we can derive meaning from all these small fragments.

1.2.1 The Illumina sequencing method

The Illumina sequencing process can be broadly divided into four separate steps: library preparation, cluster generation, sequencing and data analysis. In particular, two processes are fundamental to the Illumina sequencing method, bridge-amplification and synthesis-by-sequencing. Bridge-amplification is applied during the cluster generation stage whereas synthesis-by-sequencing takes place in the sequencing stage. In the following section we shall give an overview of the first three steps before reviewing data analysis in Section 1.2.2. Further details regarding the method can be found in the online resources provided by Illumina (see ILLUMINA INC. 2017 & 2010).

Library preparation Illumina sequencing starts with the library preparation stage. The DNA is fragmented and platform specific adapters are ligated to each end of each fragment (the purpose of the adapters is made clear in the following steps). After the ligation stage the DNA can be exposed to PCR, a technique for increasing the volume of DNA by making copies of each strand. PCR is a source of considerable sequencing bias and some more recent library preparation techniques forgo the PCR step and as such are referred to as "PCR free sequencing". Perhaps the most widely used library preparation, Illumina's own TruSeq platform is available in PCR free variants. The last step in library preparation is gel purification where fragments between a particular size are selected for sequencing. The ideal size of selected fragments varies by platform.

When conducting paired sequencing, for each fragment we will eventually receive two separate sequences each describing opposite ends of the same fragment. This paired-end approach is useful in resolving ambiguous alignments and also allows for greater sequencing depth at the same cost. The fragment size is usually aimed for such that the fragment size is longer than the two read lengths combined. This prevents read pairs from overlapping and sequencing the same portion of DNA. If the goal is to sequence as much of the genome as possible, then avoiding overlap is ideal.

Cluster generation Once a library has been generated from a sample, fragments are loaded on to a device called a flow cell. The flow cell contains millions of "oligos" (a short DNA sequence) that are complementary to the adapters ligated to the fragments in the previous step. The library is washed over the flow cell in an attempt to bind fragments to the oligos. After this binding stage, bridge-amplification takes place in order to generate a dense cluster of poly-nucleotides derived from each fragment.

Sequencing The next step is to read the DNA sequence contained within the multitude of clusters on the flow cell. Illumina uses a process called sequencing-by-synthesis to achieve this goal. At each iteration (the number of iterations is defined by the read length) nucleotides are washed across the flow cell and preferentially bind to the next available nucleotide in each cluster. Once bound, the nucleotides are excited and flash a colour according to their bound base. This flash is recorded and interpreted by the sequencer.

It is at this step that each of the bases are associated with a quality score. The sequencing machine assigns a "Phred score" to each base in the read that represents the likelihood the base has been accurately sequenced.

1.2.2 Post-sequencing

Once sequencing has finished, the individual reads are stored to a computer file in FASTQ format. Each entry in the FASTQ file contains the name of the read, a sequence of base-pairs and an associated Phred quality score. In paired-end sequencing two complementary FASTQ files are produced. The first file contains the first read in the pair and the second file all of the second reads in the pair.

The next crucial step in processing these reads is alignment. Sequence alignment takes each of the read-pairs and attempts to detect where the sequence originated from on the genome. Due to the size of the human genome and the number of reads involved, this is a computationally complex task. Programs dedicated to this task are some of the most widely used and discussed in the entire field of computational biology.

The amount of reads over a specific locus within the genome is referred to as the "coverage". Greater coverage confers greater redundancy amongst the reads, making errors and genuine polymorphisms easier to detect. The draw back of higher coverage is the expense of materials required to generate higher amounts of sequencing reads. Within sequencing experiments coverage is not uniform across the genome. Factors such as the relative amount of G and C in the target sequence are known to have an effect, although progress has been made in recent years to mitigate these biases (SIMS *et al.*, 2014).

The SAM and BAM format The output of these alignment algorithms is called a sequence alignment map or SAM file. Each entry in a SAM file represents a single read from the sequencing experiment. As well as recording the sequence base-pairs and associated quality scores, if a read is mapped the entry will contain information regarding its location on the genome and the quality with which it was mapped. Often reads in SAM files are sorted by the location of alignments.

When the aligner cannot find the location of a read within the genome it marks the read as unmapped and the chromosome and position fields are filled with place holder values. The causes of unmapped reads are numerous. It may be that a read has suffered from errors and so the location of origin on the genome cannot be found. Often repetitive reads do not map to the genome because the corresponding region has been expunged from the reference genome[3]. Alternatively it might be that the read of origin stems from a location that is not within the reference genome. This may occur because of some specific variation to the organism being sequenced or because of a DNA rearrangement in a cancer cell. When the SAM file is sorted by location, the unmapped reads are stored at the end of the SAM file.

SAM files are a plain text format and, as such, are not an efficient way of storing the vast amounts of data produced by modern sequencing technologies. As a result, SAM has a sister format called BAM (binary alignment maps). A BAM is a compressed and binary translated version of a SAM file. Despite the compression, BAM files produced from WGS data can often exceed 100 giga-bytes (GB). In the prostate cancer cohort examined later, the larget BAM file exceeded 250GB.

BAM file compression is handled by the BGZF algorithm. Crucially, the BGZF algorithm allows sequential, random access to the BAM files with the help of specially generated index files. This means that regions of the BAM file can be retrieved without the need to iterate through all of the preceding reads. This random access is facilitated by the fact that the BGZF compression algorithm segments the BAM file into blocks. Blocks are simple binary structures that have a header, detailing the size of the following compressed payload, followed by the compressed data. We will see later in this thesis how this block structure is exploited by parts of the Parabam methods to enable the fast merging of BAM files.

More description of the SAM and BAM file formats, as well as information regarding samtools, the software interface for manipulation the files, is given in Li *et al.* (2009).

[3]This can be the case for telomeric reads, however, due to the presence of ITS in the genome we find that many true telomere reads are mapped erroneously to these locations

1.3 Estimating telomere length from WGS data

As part of this thesis we detail Telomerecat, a tool for estimating telomere length from WGS data. On the outset of the investigation that led to this thesis there were no programs dedicated to this process. However, the field has experienced considerable growth and there are by now at least three fully fledged methods in the form of computer programs, other than Telomerecat. In this section we will detail the work preceding the development of these methods and the methods themselves.

1.3.1 Beginnings

The first study to indicate that sequencing reads could be used to shed light on the length of telomeres was CASTLE *et al.* (2010). CASTLE *et al.* observed the number of sequencing reads containing at least four copies of the telomere hexamer and then adjusted this observation according to the amount of reads within the sequencing experiment. They applied this method to three samples, two of the samples were from healthy adult whole blood donors and one was derived from a lung cancer cell line. CASTLE *et al.* that their measurement showed differential amounts of telomere lengths between the blood and cancer samples. Thus a new field was born.

PARKER *et al.* were the first study to use the Castle method. In PARKER *et al.* (2012) they apply the method to 235 WGS samples from paediatric cancers and observe differential gain and loss of telomere across the different cancer types. These findings were confirmed separately by qPCR, mTRF and qFISH.

1.3.2 Tools

The first dedicated tool to estimate telomere length from WGS, TelSeq, was introduced in DING *et al.* (2012). TelSeq introduced GC correction to the telomere length estimate and performed the first analysis into the optimum threshold for observations of the telomere hexamer with a read. They also provided the first large scale validation of a WGS estimation method by comparing TelSeq estimates to those of mTRF.

TelSeq is an iteration of the Castle method in that it works by counting reads that display a number of telomere hexamer and normalising this by the coverage of the sample as a whole. However, the TelSeq method attempts to implement a system that can account for the GC bias inherent in modern day sequencing.

TelSeq works by observing the amount of sequencing reads that contain the telomere hexamer above a threshold (their analysis suggested a default of 7 for 100bp reads). They

then observe the amount of reads that contain 48-52% G or C nucleotides (we refer to this concept as GC content throughout this thesis). Given an estimate of the human genome within the range of 48-52% GC content and given they know the amount of reads that this amount of the genome provided to the experiment, they infer the length of telomere that produced these reads as

$$length = \frac{t}{g} \cdot \frac{C}{E \cdot 1000}$$

Where $length$ is the telomere expressed in KB, t is the number of reads over the telomere hexamer threshold, g is the number of reads within the GC threshold, C is a constant that represents the amount of the human genome at the GC threshold, and E is the number of telomeres thought to inhabit the sample (unless modified by the user this defaults to 46).

TelomereHunter is another tool designed to analyse telomere in WGS data (FEUERBACH *et al.* (2016), *preprint*). Unlike TelSeq, TelomereHunter does not produce an estimate of length, rather it reports an estimate of relative telomere content. In other ways TelomereHunter approaches the estimation process similarly to TelSeq in that it considers telomere reads by the amount of telomere hexamers contained with the sequence. However, it also refines the threshold for inclusion of telomere reads and filters for reads which are already mapped to non-telomeric regions of the genome; an attempt to filter for ITS reads. Furthermore, TelomereHunter makes considerations for alterations to telomere by the process of ALT and has demonstrated an ability to report the presence of an ALT phenotype amongst input samples.

CompuTel takes a different approach to identifying telomere reads. They align each of the relevant reads in a sequencing file to a hypothetical telomere reference (NERSISYAN and ARAKELYAN, 2015). Reads that align to the reference are assumed to be telomere reads. This addresses some of the frailty of using a hexamer threshold as the sole criterion of whether or not a read is from the telomere. Using an aligner can more accurately detect sequencing errors. However, during our investigation surrounding the development of Telomerecat we uncovered drawbacks to using an aligner that we discuss in Section 3.4.3 of this thesis.

1.3.3 Summary

The methods detailed above were the subject of a review publication which found them each to be comparable in performance (LEE *et al.*, 2017) on a validation data set. In their review, LEE *et al.* also make mention of the cumbersome nature of CompuTel in requiring unaligned FASTQ files as input.

Amongst the tools mentioned above, TelSeq has perhaps received the widest usage. Notable applications of TelSeq include an application to samples from 31 different cancer types (BARTHEL *et al.*, 2017) and a study detailing how loss of function of POT1 can impact risk of melanoma (ROBLES-ESPINOZA *et al.*, 2014). TelomereHunter has also been applied to cancer samples in the form of an application to the PCAWG data set (SIEVERLING *et al.* (2017), *preprint*).

The tools mentioned above have been shown to correlate with existing experimental approaches to measuring telomere length. However, there are considerable shortcomings.

TelSeq and Computel make a direct assumption regarding the amount of chromosomes in the input sample. These assumptions are unfounded when applying the method to cancer samples due to the prevalence of extreme chromosomal aberrations that may increase or decrease the amount of DNA content within a cell.

To avoid making an inference of chromosome count TelomereHunter does not directly report telomere length, rather it reports the telomere content of a sample. This is not unreasonable and there is value in this metric alongside telomere length. However, telomere length itself is an important metric and there is a need for a method that can estimate telomere length agnostic to the amount of chromosomes in the sample.

Furthermore, Computel and TelomereHunter attempt to reduce noise caused by reads from ITSs. To do this they each rely on the mapping locations of these reads. This is another potential source of bias as commonly used aligners are not optimised to handle these repetitive structures and repetitive reads are not reliably mapped to the genome. By relying on the mapping location of these reads, the methods are exposing themselves to bias as a result of the variation of alignment or even reference genome used to pre-process the sample.

In Chapter 3 we detail our method, Telomerecat, which attempts to address the shortcomings of previous approaches.

1.4 Fundamental concepts in computer science

Parts of this thesis, particularly Chapter 2, are focused on the development of software and algorithms. In this section we give an overview of these topics.

Classes and objects In computer science, classes are a set of definitions for mutable structures within programs, known as objects. As an example, if we were writing a program that needed to record information about cars we could define a class called *Car*. When we needed to add information about a certain car, we could create a new *Car* object using the

Car class. The resultant object would store the brand name, miles per gallon and the year of manufacture as variables.

Objects can also have functions assigned to them that we define in the class. For instance, using the *Car* object defined above as an example, we could write a function that returned the age of a car. We might call this function *getAge*. We we call the *getAge* function on a given *Car* object it would simply subtract the year of manufacture from the current day and return the result.

Big O notation Big O notation is a way of discussing the running time of algorithms by comparing how the run time changes as a result of the size of input. While benchmarking is useful for comparing entire pieces of software in controlled settings, it is often not useful or practical to benchmark individual pieces of code. Indeed, what use is it to talk of an algorithm taking taking five seconds to run, when this number is so heavily influenced by external factors? Amongst these external factors is the hardware the program is running on or even the computational load that the hardware is experiencing at any one moment in time.

To provide comparison between different algorithms, big O notation contemplates the number of operations carried out by an algorithm. For instance, imagine we have a list of numbers that contains n entries. We wish to write an algorithm that iterates through this list and records the cumulative sum of the numbers. To do this our algorithm must visit each of the entries of the list in turn. Thus the running time of an algorithm is constrained by the size of the input, in this case the longer the list, the more operations we will need to carry out. We say that this algorithm performs in "linear time". Linear time is expressed as "O(n)" in big O notation.

Aside from algorithms which display O(n) time we will also discuss constant time algorithms, expressed as "O(1)". A constant time algorithm is an algorithm where the run time is not constrained by the number of elements in the input. For instance, using the list from the previous example, if we wished to check whether the first two elements in the list were greater than zero we would write an O(1) algorithm. The amount of operations carried out by the algorithm does not depend on the size of the input list because we always only check the first two elements.

There are many other run time complexities, however, we will not refer to these as part of this thesis. Additionally, big O notation can also be used to discuss the space an algorithm takes up, but again we shall not use big O notation as such in this thesis.

Data structures Data structures enable computer programs to store efficiently and sort through data. Parabam and Telomerecat make use of numerous data structures as part of their operation.

One of these structures is an array list or list for short. A list is a linear collection of objects stored in order. Lists are convenient in that they allow objects to be accessed in O(1) time as long as the index of the object within the list is known beforehand.

In Chapter 2 we will make several references to hash maps. Hash maps are an unsorted data structure that allow objects to be stored via association with a key. In other words, when we make a call to a hash map to retrieve an object, we first provide a key and the hash map will return the object associated to that key. The hash map gets its name from the "hash" function used to create an index from the provided key. A well designed hash map implementation should have a average case retrieval of O(1) making them a highly efficient way of managing unsorted data.

First-in-first-out (FIFO) queues are used extensively for inter process communication within Parabam. A great clue as to how the queue data structure functions is in the name. The FIFO queue object functions just like a queue of people waiting for a service. Once an object is added to the back of the queue (via a "put" command) it will remain in the queue until all the objects that were in placed in the queue before it have been accessed (via a "get") command.

1.5 Chapter Overview

In Chapter 2 we describe in detail Parabam, our new tool for interrogating WGS sequencing files in parallel. We demonstrate that Parabam is able to iterate more quickly over sequencing files than non-parallel approaches.

In Chapter 3 we set out the Telomerecat method and describe the validation experiments and comparisons we have made to existing experimental and computational methods. Towards the end of the chapter we will discuss the complexities surrounding telomere length estimation and how we have attempted to solve these issues.

In Chapter 4 we apply Telomerecat to a cohort of 192 prostate cancer samples. In this chapter we describe associations between cancer stage, the number of somatic mutations and previously identified molecular subtypes.

In Chapter 5 we apply the Telomerecat method to a cohort of 8950 rare blood disease samples. We analyse the results of a telomere length GWAS conducted on these samples and describe how Telomerecat assisted in identifying a new phenotype associated with a rare

germline mutation in the DCK1 gene. The size of this cohort also allows us to add value to the existing knowledge about the relationship between age and gender on telomere length.

In Chapter 6 we conclude by summarising the salient results contained with the thesis as well as reflecting on the current state of the methods and where they perhaps might be improved. We also discuss future work including the potential for estimating telomere length in samples generated by the TENX and minION, which are relatively new sequencing technologies.

1.6 Original Contribution

In this thesis we present several original contributions:

- A new tool for applying functions to reads in a BAM file, Parabam. Included within this tool is a method for applying the functions to pairs of reads in parallel, even when the file is sorted by mapping location.
- A new tool, Telomerecat, that estimates telomere length from whole genome sequencing files without making an assumption of chromosome or telomere count.
- An application of Telomerecat to a cohort of Prostate cancer samples. The largest and most comprehensive study of telomere length within Prostate cancer. This study is the first to suggest a link to the SPOP and TMPRSS2-ERG fusion subtypes and telomere length, amongst other significant trends in Prostate cancer.
- An application of Telomerecat to a cohort of rare blood diseases. Telomerecat helps to identify an extremely rare, hitherto unidentified, point mutation in the DCK1 gene that causes a late onset immunodeficiency caused by short telomeres.

Chapter 2

Parabam: processing BAM files in parallel

As we saw in the introduction to this thesis, the output of WGS experiments is often large, in excess of 100GB per file. Furthermore, the proliferation of WGS data means that there is a rich vein of information waiting to be tapped. In this chapter we detail our method Parabam that aims to analyse these data quickly and efficiently.

Parabam[1] is a program that allows users to apply a function to each read in a BAM file. Users may use this function to create subsets of BAM files or extract statistics from the individual reads. Parabam also allows users to interact with location sorted BAM files as if they were pair sorted.

Furthermore, Parabam conducts these operations in parallel by way of its multiprocessing architecture. The advantage of parallel processing is that operations can be carried out simultaneously on separate processing cores within a computer. This makes the analysis quicker. Parabam also has a strong focus on usability, with a minimal amount of programming knowledge allowing users to create complex subsets and extract statistics from BAM files.

In this chapter we set forth our method for the parallel processing of BAM files.

2.1 Motivations

Processing increasingly large datasets is one of the many challenges faced by contemporary computational biologists. From our vantage point on the precipice of the so called "Big Data" era this problem is a greater challenge than ever. To these ends we propose a method

[1]The name Parabam is a fusion of the words BAM and parallel. Parabam is also the first-person singular imperfect active form of the Latin verb "paro" which means "to present or bring forward".

of processing BAM files in parallel. Parabam aims to provide an easy to use interface for creating complex subsets and statistics from the BAM and SAM format[2].

Parabam provides two modes of operation: *subset* and *stat*. The *subset* mode allows the user to specify a rule in order to select only a subset of reads. Whereas *stat* allows the user to record various statistics regarding reads or read-pairs. The user specifies a rule using the Python programming language, either by a command line interface or programmatically. This rule takes the form of a Python function.

Parabam is the framework around which many of the methods and processes for the development of Telomerecat (detailed in Chapter 3) were built. Indeed, the initial motivation for the creation of Parabam stemmed from the need to search through the entire BAM file for reads containing the specific telomere hexamer: "TTAGGG". The following paragraphs illustrate the motivating example for iterating through the BAM file in a pairwise manner searching for telomere reads. But Parabam is an adaptable program and the user could just as well make a desired subset based on any characteristic of either read in the pair.

One challenge we faced early on in the development of the Telomerecat method is illustrative of the need to process BAM files in some pair-centric way. Reads containing the telomere hexamer were often unmapped and so could not be summoned from the file using a known mapping location on the genome. Instead we needed to inspect the entire file; every single read had to be interrogated for the sequence.

Early attempts to select these read-pairs, by processing each read individually, revealed that the subset would be missing reads important to the analysis. To illustrate, consider a read-pair where both reads in the pair are unmapped. Read one contains the telomere hexamer and the read two does not. For the purposes of our analysis we require both reads. When iterating over each read individually we would capture read one as it contains the desired sequence. However, read two would not be included in the subset. Furthermore due to read two being unmapped, any attempts to retrieve the read would be impossible.

To solve this problem, in pair processing mode, Parabam provides an interface where both reads in the pair are presented to the user simultaneously. In this way, if a read's pair is relevant to the subset the user can decide to include the both of the reads in the pair.

Although initial motivation for Parabam was as a framework for the processing of sequencing files in search of telomere read-pairs, we realised quickly the broader applicability of such a method. We wished to design a program that took as input simple instructions for inclusions of reads within different subsets. Parabam would take these instructions, collect the reads or read-pairs that met the specification and return a BAM with the desired reads.

[2]Parabam works on both SAM and BAM format, however for convenience we will only refer to BAM files throughout this chapter

Later, we would add functionality to Parabam that would allow the user to simply record information about each read or read-pair and have the information presented as a simple CSV file.

Pre-existing tools There is no tool quite like Parabam; this makes direct comparison difficult. Parabam defines a new way for users to interact with sequencing reads by presenting reads to a function written in Python. While no one tool has the complete feature set of Parabam, other tools implement some similar functionality.

For instance, Biobambam (TISCHLER and LEONARD, 2014) allows users to iterate through an index-sorted BAM file as if it were named sorted. However, this process is done using a single processing thread and Biobambam does not provide an interface for querying each of these reads or making subsets or deriving statistics from them. Any analysis conducted on the reads once they have been returned by the tool must be piped into a further script, should the user wish to interrogate the reads further.

Another tool, Sambamba (TARASOV *et al.*, 2015), attempts to replace the core architecture of the original samtools with a parallelised implementation. The result is a tool that iterates through a BAM file faster than any other. What is more, Sambamba provides a command line interface for creating subsets based on the criteria specified by regular expression. Unlike Parabam however, more complex subsets are not possible (for instance those using higher order language statements like "if statements"). Furthermore, Sambamba is not capable of iterating through an index sorted BAM file as if it were name sorted.

2.2 Program Architecture

Parabam comprises approximately 4000 lines of code across 50 classes. Great care has been taken to conform to the principles of object orientated design and of Clean Code (MARTIN, 2008). Whilst the code base is relatively large, these principles ensure that code re-usage is kept to a minimum. In this Section we detail the algorithms and systems that constitute the Parabam program.

2.2.1 An overview

Parabam is built with a modular structure as depicted in Figure 2.1. The *FileReader*, *Tasks* and *Handlers* all run concurrently and pass information to one another using a "First in First Out" (FIFO) queue system.

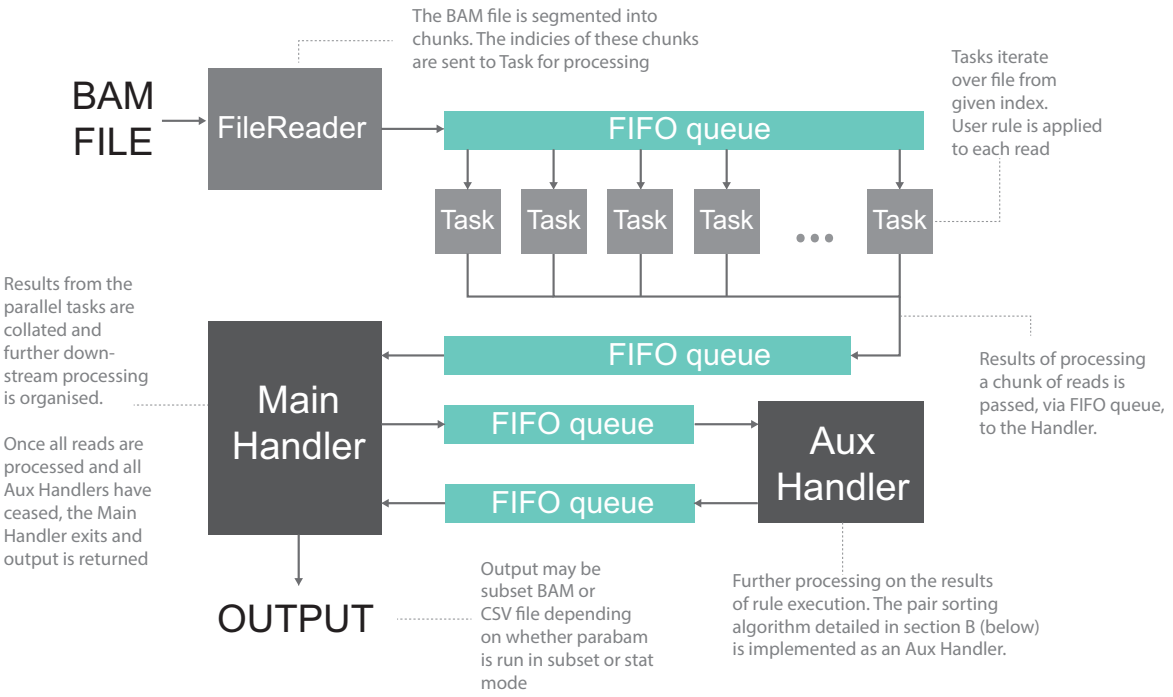


Fig. 2.1 An overview of Parabam at runtime

Applying the user defined rule Parabam's primary goal is to apply a user defined rule to each read in any BAM file. This rule is defined by the user as a python function and supplied to the program at runtime. The core components involved in this process are the *FileReader*, *Task* and *Handler*. In the following paragraphs we will briefly define the main functions of each component. The technical details of how these components function is described later in Section 2.2.3.

Each grey block in Figure 2.1 is defined within the Parabam program as a separate Class. At runtime, each class is initialised and assigned pointers to the various queues that serve as input and output to the object. After initialisation is complete each class runs in its own process for the lifetime of the program and processes data as it appears on the relevant FIFO queues.

The processing of BAM files starts with the *FileReader*. The *FileReader* is initialised and then iterates over the input BAM and outputs file indices to *Task* files via the relevant FIFO queue. Once the *Task* receives the file index it begins processing the BAM file from the specified file index and stops once it processes a pre-specified amount of reads. Within the *Task*, the user specified rule is applied to each read. The exact output of the user-rule is highly dependent on the operation mode. This output is stored and the result of the rule is sent to the *Handler* via FIFO queue. The *Handler* collates and stores the output of the

user-rule. The *Handler* is also responsible for the triggering of various *AuxiliaryHandlers*. *AuxiliaryHandlers* represent flexible classes that can be used to conduct downstream operations in parallel. For example, the pair matching process (Section 2.3) and file merging operations detailed later (Section 2.4.2) are implemented as *AuxiliaryHandlers*. Not shown in 2.1 is the fact that there can be multiple instances of the *FileReader* process.

Operating modes Parabam functions in two operating modes: *subset* and *stat*. The *subset* mode allows users to create subsets of reads according to characteristics (i.e. duplicate status, sequence content or flag). The *stat* mode allows the user to record statistics about different characteristics of reads in the read file. For instance, the user may iterate through the BAM file recording the amount of reads that contain an "N" base-pair. The exact content of the rule is dependent on the user's creativity and objectives. In creating their rule the user may refer to any aspect or characteristic of the read, including its sequence, Phred quality score and mapping characteristics. The read object is of type *AlignedSegment* provided by *pysam* (the python implementation of samtools).

Once Parabam has iterated over the entire BAM file and the user-rule has been applied to each read or read-pair, results of the analysis are presented to the user. The format of these results is predicated on the mode of operation. In the case of the *subset* mode a path to a BAM file containing the relevant subset is produced. In the case of the *stat* mode one or more comma separated value (CSV) files are produced.

Programmatic structure While Figure 2.1 shows a representation of Parabam at runtime, Figure 2.2 shows the class structure of Parabam. Parabam uses multiple inheritance, a core concept of object orientated programming. For instance, we see that the *Task* class is first defined in *parabam.core* and receives a final implementation in both *parabam.command.stat* and *parabam.command.subset*. This object orientated design ensures that code recycling is minimised and also enables the dynamic initialisation described in Section 2.4.1.

Parabam is written in Python 2.7 and then optimised using the Cython framework. Parabam is enabled by extending the *multiprocessing* package distributed as part of the standard Python distribution. Each of the classes mentioned in the previous few paragraphs functions by extending the *Process* class defined in the *multiprocessing* package. Additionally, Parabam makes heavy usage of the FIFO queue as provided by the *multiprocessing* package. In order to interface with BAM files, Parabam uses the python implementation of the samtools library, *pysam*.

2.2.2 Interacting with Parabam

The command-line There are two ways of interacting with Parabam. The first way is via the command-line. Once installed, Parabam is accessible directly from the command line and can be invoked like any other UNIX compatible program. Care has been taken to ensure that a minimal amount of parameters are needed to use Parabam. As such the user needs only to specify the mode of operation, an instruction file and input files. Should the user wish, they may also adjust the amount of *Task* objects Parabam uses (analogous to increasing processing power) or other more low level parametrisations. The style of command-line interface used by Parabam is influenced by the interface employed by samtools (Li *et al.*, 2009), an almost ubiquitous tool in modern bioinformatics. A typical command line invocation of Parabam is as follows:

```
parabam subset --rule instructions.py /path/to/file.bam
```

The command above is understood as follows:

- `parabam` The name of the program
- `subset` The mode of operation. This can be either subset (as shown) or stat. The operating modes are covered in Section 2.2.4
- `--rule instructions.py` This part of the command specifies the location of the instruction file. The contents of the instruction file are dependent on the operating mode and the nature of the user defined rule. Examples of an instruction file can be seen in Appendix A
- `/path/to/file.bam` The path to files that the user wishes to analyse. In the event that more than one file is parsed, each file will be analysed in alphabetical order.

The python API The second way in which a user can interface with Parabam is via the API (Application Programming Interface). This functionality is achieved by interfacing with Parabam within Python programs. Parabam is distributed as a Python library and so, once installed is directly accessible within the python environment with the simple command:

```
import parabam
```

Once imported, the user can make use of the interface classes `parabam.Subset` or `parabam.Stat` to run analyses. The user-rule, constants and input files are passed to these classes as standard python objects: the user-rule is a function, constants a python *dict* and the input files as a list

of strings. Interacting with Parabam in this way makes it easy for the user to use Parabam as a framework for further analysis. Telomerecat, the subject of Chapter 3, interfaces with Parabam in this way.

The user-rule It is the goal of Parabam to apply a user defined rule to each read in the read file. As we have seen in this section there are multiple ways to interface with the program. In each of these ways the actual content of the rule is identical.

The user must define a rule that takes three arguments: *read*, *constants* and *parent*. The read is an instance of *AlignedSegment* from the *pysam* package. The *constants* variable is a Python *dict* object loaded with constants specified by the user, and the parent is an object of type *ParentAlignmentFile*.

Any variables or functions defined in the same scope as the user-rule are accessible from within the user-rule. This is especially useful should the user wish to describe helper functions to assist in the analysis. The functions and variable in the same scope as the user-rule vary slightly based on the way Parabam is used, however when using the command line interface, any variable or function defined in the instruction file is in the same scope as the user-rule.

This rule based paradigm grants a large amount of freedom to the user. The user may access any variable stored in the *read*, *constants* or *parent*. This allows the user to form subsets or record statistics based on a wide range of read characteristics. The user may also pass the *read* object to other functions as long as they are defined in the same scope as the user-rule.

The exact content of the rule will differ according to the aims of the user. Additionally, the operating mode and whether the program is running in pair processing rule require slightly different syntax. These intricacies are covered in detail in Section 2.2.4.

2.2.3 A description of each component

This section describes each of the components shown in Figure 2.1 and how the component contributes to the overall function of Parabam. All of the objects described here exist as a separate process on the users computer. At initialisation, the process is started and the main loop of the class activated. This loop is run for the duration of the program's execution. The loop is only stopped once the entire BAM file has been read, all reads have been analysed by the user-rule and the results of the rule collated into the final output of the program. This section will look at the main function of each of these components. We detail the exact method of how Parabam is initialised in Section 2.4.1 and ended in Section 2.4.3.

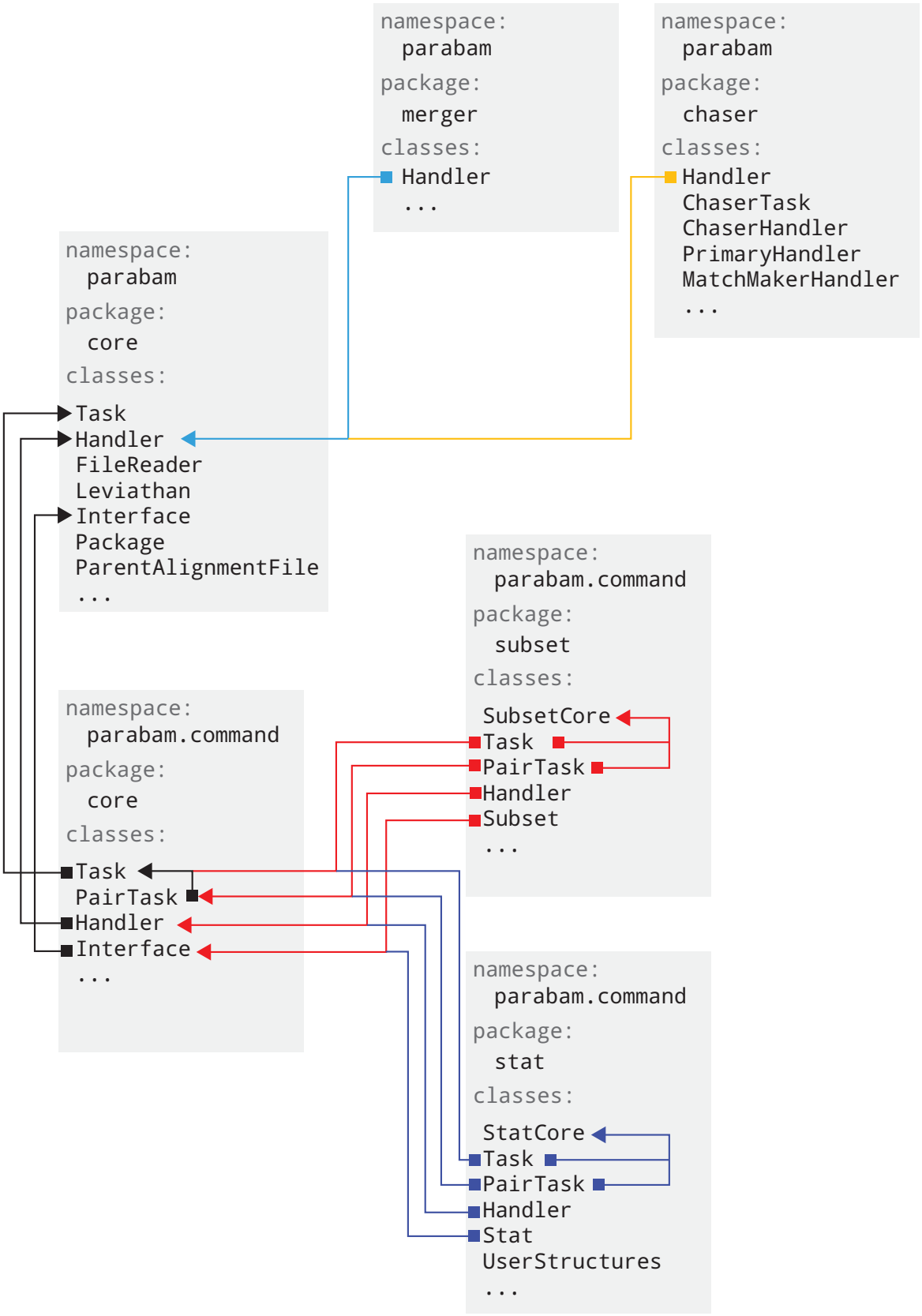


Fig. 2.2 A class diagram for the main packages and class in the Parabam software package. The arrows represent inheritance relationships amongst the classes

FileReader The processing of reads begins in the *FileReader*. The principal job of the *FileReader* is to apportion the reads into batches which in turn are then processed by parallel *Task* processes.

The main loop of the *FileReader* class is detailed in Algorithm 2.1. A file index is passed via FIFO queue to the child tasks for further processing. The *FileReader* then spools to the start of the next batch.

In the event that more than one *FileReader* is in operation a check is made via a modulus operation. If a match is not made, the index is not passed. This enables Parabam to function with more than one *FileReader*. We have found that when using traditional spinning hard disks Parabam is most efficient using one *FileReader*, however, more modern solid state hard disks have been observed to perform better with multiple *FileReader* processes. In the event that Parabam is run with more than one *FileReader*, the total amount of *Tasks* is split evenly between the instances. For example, if the user specifies eight *Tasks* then each *FileReader* will be able to delegate batches to four *Tasks*.

Rather than pass reads as objects in memory, the *FileReader* passes a numerical pointer to a specific location in the file. This passing of file indices cuts memory burden and side steps potential problems with passing potentially large objects in the FIFO queue. Processing speed is maintained by the fact that this process ensures a high rate of disk cache calls when the reads are subsequently retrieved by the *Task* class.

An early prototype version of the Parabam program spawned a separate *FileReader* for each chromosome in the file. We found that this put too great a strain on the physical read head of the hard drive and processing would fail. It seemed that the size of BAM files meant that the actual physical distance between reads stored on the disk was too great to allow efficient reading. The current version of the BAM file ensures that the read head is kept to roughly the same location on the disk. This further increases the likelihood of disk cache calls.

Unless computational complexity within the user-rule is sufficiently high, the bottleneck in an instance of Parabam is in reading from the hard disk. The way in which this operation is carried out has large implications for the speed and efficiency with which Parabam runs.

During the development of Parabam the way in which we implemented the *FileReader* changed several times. For instance, early versions of Parabam would simply hold a copy of each batch of reads in memory. An instance of the *Task* process would then be created with these reads as an input variable [3]. However, this comes with a substantial memory overhead, especially since when a new process is created in Python it receives a copy of its parent's processes memory. Thus, for each *Task* we would temporarily need to store a copy of each

[3] It is not possible to pass objects of type AlignedSegment via the FIFO queue.

read in a batch in memory *twice*. We have found the current method of passing file indices to be the most efficient in terms of memory usage and speed.

Algorithm 2.1 Apportion reads into batches and sends a file index for processing by *Tasks*. Z is the total number of *FileReaders* in the current initialisation. z is the unique identifier of the *FileReader* (an integer). c is the batch size as specified by the user. n is the required number of parallel *Tasks*

```
function FILEREADERMAINLOOP(Z, z, x, n)
    q ← A FIFO queue
    STARTTASKS(q, n)

    B ← BAM File Iterator
    i ← 0
    for all a in B do
        if i % Z == z then
            index ← B.getFileIndex()
            q.send(index)
        i ← i + 1
        READALIGNMENTS(B, x)
    q.send(DestroySignal)
```

Task The *Task* class is responsible for applying the user-rule to each read and capturing the output of the rule. The number of distributed Tasks is decided at runtime by the user. As the part of Parabam which applies the user-rule, the *Task* is often the component with the largest computational burden. The operations carried out within the *Task* will hitherto be referred to as In-Task. For example, the portion of pair sorting carried out within the *Task* is referred to as In-Task pair processing (detailed in Section 2.3).

Pseudo-code for the *Task* class is given in Algorithm 2.2 (for single processing) and Algorithm 2.3 (for pair processing). In both single and pair processing modes the *Task* receives a file index on a FIFO queue, sets the file pointer to the index and begins processing an allotted amount of reads. In Algorithm 2.3 we see that additional steps are required to handle pair sorting. The complexities of conducting pair processing are covered separately in Section 2.3.

Once the *Task* has applied the user-rule to each read or read-pair in the assigned batch it must handle the output and pass this output on the the main *Handler* via the output FIFO queue. The exact nature of this output depends on the operating mode, as we shall explore in Section 2.2.4. To realise the different procedures carried out in *stat* and *subset* modes, two separate versions of the *Task* class are included in Parabam. These different implementations

can be found in `parabam.command.stat.Task` and `parabam.command.subset.Task` as shown in Figure 2.2.

Algorithm 2.2 The main loop of the *Task* process, where p is a file index and b is the task size, or number of reads to be processed in this batch. p is received via a FIFO queue from the *FileReader* instance

```
function TASKMAINLOOP
    H ← HashMap
    A ← FILEITERATOR(p,b)
    for all a in A do
        R ← USERRULE(a)
    return TaskResults(R)
function FILEITERATOR(p,b)
    i ← 0
    F ← a connection to the input BAM file
    F.seek(p)
    while i < b do
        return F.nextAlignment()
        i ← i + 1
```

Handler It is the job of the *Handler* process to collate all of the data output by the concurrent *Task* objects. All operations that take place downstream from the user-rule are implemented as *Handlers*. Thus, the system wide architecture and arrangement of handlers varies a great deal depending on the operating mode Parabam is run in. In any instance of Parabam there may only be one *MainHandler* (as defined in the `parabam.command.subset` and `parabam.command.stat` packages) however, there is no limit on the amount of *AuxiliaryHandlers*.

The *MainHandler*, depicted in Figure 2.1 in the bottom left hand corner of the diagram, is the first stop for all data output by *Task* instances. Algorithm 2.4 gives a pseudo-code representation of the *MainHandler* processes operation. Throughout the duration of the program the *MainHandler* receives results packages on its incoming FIFO queue. The succeeding processing is highly dependent on the operating mode and whether pair processing is enabled. However, broadly speaking, the *MainHandler* may either dispatch jobs to outgoing FIFO queues for further processing by *AuxiliaryHandlers* or may choose to process the new results package immediately. Results packages being processed immediately are dealt with by the `NewPackageAction` whereas packages being stored for further processing are processed via the `PeriodicAction`.

Algorithm 2.3 The main loop of the *Task* process in pair process mode, where p is a file index and b is the task size, or number of reads to be processed in this batch. p is received via a FIFO queue from the *FileReader* instance. Due to pair sorting additional steps are taken to sort and store unpaired reads

```
function TASKMAINLOOP(p,b)
    H ← HashMap
    A ← FILEITERATOR(p,b)
    for all a in A do
        n ← read name of a
        if a in H then
            m ← H.get(n)
            R ← USERRULE(a,m)
            H.remove(n)
        else
            H.insert(n,a)
    T ← STASHUNPAIREDREADS(H)
    return TaskResults(R,T)

function STASHUNPAIREDREADS(H)
    T ← An empty BAM file
    ▷ Write the leftover reads into a temporary file
    for all n in H do
        a ← H.get(n)
        T.write(a)
    return T
```

The main loop of the *AuxiliaryHandler* bears close resemblance to that of the *MainHandler*. As such, the primary method for interacting with incoming packages is via the `NewPackageAction` and `PeriodicAction` functions. However, the actual implementation of *AuxiliaryHandlers* varies based on the intended function of the process. Implementations of AuxiliaryHandlers are discussed in Section 2.3 and Section 2.4.2.

Algorithm 2.4 The main loop of the *MainHandler* process. Where *inqu* is a FIFO queue the other end of which is held by the concurrent *Task* processes. Also where Q is a HashMap with process IDs are mapped to FIFO queues. The other end of each FIFO queue in Q is held by an *AuxiliaryHandler*.

```
function HANDLERMAINLOOP(inqu,Q)
    destroy ← FALSE
    finish ← FALSE
    i ← 0
    while not finish do
        resultPackage ← inqu.get()
        NEWPACKAGEACTION(resultsPackage)
        i = i + 1
        if i % 10 == 0 then
            PERIODICACTION(Q)
        if resultPacakge == DestorySignal then
            destroy = TRUE

        if destroy then
            if some set of finishing conditions then
                finish = TRUE

▷ The contents of the following functions' are
▷ dependent upon the operating mode
function PERIODICACTION(Q)
    ...
function NEWPACKAGEACTION(resultsPackage)
    ...
```

2.2.4 The operating modes

Parabam functions in two separate modes of operation: *subset* and *stat*. The operating mode informs several aspects of the program's operation. First, the mode of operation defines the way in which the user interacts with the program. The format of the user-rule will differ according to the operating mode. Secondly, the mode of operation alters the way in which

classes are instantiated at runtime. Depending on whether the user specifies the *subset* or *stat* modes, Parabam will load different versions of the *Task* class to handle the output of the user-rule.

The subset operating mode The *subset* mode allows the user to create subsets of BAM files. In order to define which reads are included in the subset the user passes Parabam a rule, written in Python code. An example of this rule, formatted for the *subset* operating mode, is shown as pseudo-code in Algorithm 2.5. Actual examples of the user-rules, written in Python code, for the subset mode can be found in Appendix A.

The structure of Parabam at runtime must be prepared to handle the output of the user-rule. When Parabam is running in *subset* mode, the output of the user-rule is always of type *AlignedSegment*; essentially a representation of a sequencing read in memory. Parabam must then correctly allocate the read into the relevant subset. To do this Parabam uses the version of the *Task* class described in the `parabam.command.subset` package in combination with an *AuxiliaryHandler* called the *Merger*.

When Parabam is running in *subset* mode, the *Task* process applies the user-rule to each read. If a read is found to belong to a subset, the *Task* writes the corresponding read to a temporary file. Once the *Task* has finished processing through the entire batch, a path to this temporary file is written to the FIFO output queue as part of a results package. The results package is received by the *MainHandler*, at which point it is added to a staging area (implemented as a python `dict`) by the `NewPackageAction` function. At regular intervals the *MainHandler* polls the staging area using the `PeriodicFunction`. Once a certain limit of temporary files are accrued within the staging area, the *MainHandler* sends these files to the Merger (an *AuxiliaryHandler*) where they are written into a file to be presented to the user when execution has finished.

Algorithm 2.5 An example of a rule passed to Parabam in subset when the program is not in pair processing mode. In this example the read is added if the flag is set to 4. However the user may specify any characteristic of the read for inclusion in the subset

```
function RULE(read, constants, par)
    results = new List
    if read.flag == 4 then
        results.add(read)
    return results
```

The stat operating mode The *stat* operating mode allows the user to record data and statistics concerning the reads in a read file and to return these data as a CSV file. While

the *subset* mode is useful for creating small subsets of reads for which repeated analysis is expected, the *stat* mode can be used to record information about a very large number of reads quickly and efficiently. The two modes are useful when combined sequentially. Often in the analyses detailed later in this thesis, a subset of relevant reads is found first and then the stat operation used to glean information regarding these reads.

In order to effectively store and manage the data types, the *stat* operation uses the *UserStructures* class. *UserStructures* are compact classes defined in the `parabam.core.stat` package that allow the *MainHandler* of the *stat* interface to store incoming data from the user-rule. The *UserStructures* are defined by the user at runtime. All the user needs to do in order to define a *UserStructure* is to define the type of data and the store-method associated with the *UserStructure*. The data types expected by Parabam are as follows:

- `int` An integer. A primitive type within the Python.
- `numpy.array` An N dimensional array defined as part of the *numpy* Python package
- `Counter` A hashmap data structure for storing integers. Part of the *collections* Python package.

The user may choose from four store methods: `max`, `min`, `cumu` and `vstack`. Table 2.1 shows how each store method is implemented in each of the *UserStructures*. *UserStructures* are defined as part of the instruction file in a function called `getBlueprints`. `getBlueprints` is run at initialisation and the output of `getBlueprints` is transformed into a set of *UserStructures*. These *UserStructures* are passed to the *Task* and *Handler* so that they are prepared to handle the output of the user-rule.

In *stat* mode the user-rule outputs a hash-map where each key is the name of a user structure and the corresponding value is the data that we wish to store in the relevant user structure.

In *stat* mode, once the user-rule has been applied to all of the reads in the batch, the *Task* now holds local copies of all the user-structures. These local user structures are simply bundled into a result package and sent via FIFO queue to the *Handler* where the local copies are incorporated into global user structures. Once processing of the file has finished these global structures are written to CSV files and a path is returned to the user.

An example of an instruction file for the *stat* operation mode can be found in Appendix A. Pseudo-code for a user-rule and blueprint definition for the *stat* operation mode can be found in Algorithm 2.6.

Table 2.1 User structures and how the store method is interpreted for each type

	Store Methods			
User Structure	max	min	cumu	vstack
int	Store the maximum of all values	Store the minimum of all values	Cumulative sum of all values	NA
Counter	Store the maximum of all values for each key	Store the minimum of all values for each key	Cumulative sum of all values for each key	NA
numpy.array	Store the maximum observed value at each index in the array	Store the minimum observed value at each index in the array	Cumulative sum of all values in the array	Row bind each array to all previous arrays

Algorithm 2.6 An example of a rule passed to Parabam in *stat* mode when the program is not in pair processing mode

```
function RULE(read, constants, par)
    results ← new HashMap
    results["cCount"] ← read.seq.count("C")
    results["minQual"] ← min(read.qual)
    return results

function GETBLUEPRINTS(blueprints)
    blueprints["cCount"] ← new HashMap{data=int, storeMethod="sum"}
    blueprints["minQual"] ← new HashMap{data=int, storeMethod="min"}
```

2.3 Pair processing in parallel

As we have seen, the aim of Parabam is to apply a user defined function to each read in a BAM file. BAM files are often sorted by the location of individual reads on the genome rather than by fragment name (i.e pair sorted). However, due to the mechanics of pair sequencing and subsequent sequence alignment, a majority of read-pairs fall closely on the genome. The pair processing algorithm exploits general proximity of pairs to allow Parabam to process index sorted BAM files in a pairwise manner. No user intervention is required other than to specify the `-pair` flag at runtime. The user must then provide user-rules that expect to handle pairs of reads.

The implementation of the pair processing algorithm is distributed across several modules of the Parabam processing architecture. The design of Parabam necessitates that reads are processed in batches. Accordingly, pair processing starts In-Task where reads that happen to be in the same batch are paired automatically. Reads that do not fall in the same batch are processed after the fact by a process implemented in the *PairFinder* package.

2.3.1 In-task pair sorting

Pair sorting within the *Task* Pair sorting starts within the distributed *Task* itself. As we saw previously, instances of the *Task* class are provided with file pointers denoting batches of the BAM file to process. In pair processing mode reads are still passed to the user-rule and data is collected as a result of applying the user-rule. However, in pair processing mode, reads are passed to the user-rule in pairs. To achieve this the *Task* conducts pair sorting prior to passing reads to the user-rule. This process is detailed in Algorithm 2.3. The design of this algorithm, based around a series of HashMap searches, ensures that this process performs in $O(n)$ runtimes with $O(n)$ space complexity.

The result of this sorting by the *Task* class means that reads that happen to occur in the same batch are paired and processed by the user defined function within the *Task* itself.

Reads that are *not* paired are saved to a temporary file for downstream processing. There are two scenarios that may lead to a read-pair not occurring within the same *Task* batch. The first reason is that, by chance, reads are separated into different *Task* batches and are therefore processed by separate multiprocessing Tasks. Parabam divides reads in the order which they appear in the BAM file so by chance we can expect a proportion of reads to be separated between batches as a result of this process. Read-pairs of this type are described as a “batch separated read” (BSR).

The other reason for reads occurring in separate *Task* batches is that two ends of a fragment have been aligned to two disparate parts of the genome. This may be the result of a

mistake by the aligner or genuine genome corruption (this type of structural rearrangement of the genome is a common feature of many cancers). Read-pairs of this type are described as a "split read" (SR).

Preparing unmatched pairs for further sorting Once a distributed *Task* has processed a batch (as demonstrated in Algorithm 2.3) some quantity of the reads have been processed by the user specified rule. In a normal BAM file we observe that approximately 90% of reads are paired within the task (this figure is heavily dependent on factors such as the quality of the mapping, species, insert-size, coverage and ploidy). We see in Algorithm 2.3 that reads that are not paired within the task are stored in a temporary file (shown in the algorithm as T). This temporary file is then passed to the *PairFinder* process described in the next section.

2.3.2 The PairFinder: An Auxiliary Handler

The *PairFinder* is tasked with pairing reads that were not paired during the in-task sorting described above. The *PairFinder* class is implemented as an "AuxiliaryHandler" and as such runs continuously throughout the life time of the Parabam instance as it simultaneously sorts and stores the results of Algorithm 2.3. To enable faster searching, *PairFinder* also maintains its own set of child processes to which it distributes the task of searching for pairs amongst its collection of unpaired reads.

In any given input BAM file a proportion of the reads will arrive at the PairFinder. Parabam uses several techniques to reduce the search space of paired reads. The first of these techniques is by conducting searches within chromosome categories.

Searching for reads within "chromosome relationships" The *PairFinder* uses known characteristics of the reads to minimise the number of comparisons necessary to locate pairs. Each entry in a BAM file details both the mapping location of the read itself and the mapping location of the read's mate (pairs of reads are referred to as mates). The mapped chromosome of the read and its pair form a category in which all further comparisons take place. This category is referred to as c.

The first stage of the *PairFinder* algorithm is sorting reads that were not matched with the Task, into the relevant c. When an instance of T (reads that were not paired in-task) is received on the input FIFO queue, the *PairFinder* needs to sort each read within the temporary file into the relevant category, c. The categorisation process takes place in its own parallel process (see pseudo-code in Algorithm 2.7).

The grouping within c is used to minimise the amount of read to read comparisons. From now on, all attempts to find a read's pair will be conducted amongst reads of the same

c. Indeed, comparing two reads which we know do not conform to the correct mapping relationship is a wasted operation. For instance, reads that map to Chromosome 1 with a mate mapping to Chromosome 2 are only compared to reads fitting that description.

Algorithm 2.7 Where L is a list of T. H is a hash map where the keys are of type c and the value is a BAM file, represented as t_c.

```
procedure CATEGORISE(L)
    H ← a HashMap
    for all T in L do
        for all r in T do
            c ← concatenate(a.chromosome, a.mate_chromosome)
            t_c ← H.get(c)
            t_c.write(r)
    return H
```

Algorithm 2.8 Choosing instances of t_c to submit to the MatchMaker distributed process, where P is the data structure detailed in Figure 2.3

```
procedure STARTMATCHMAKERTASK(P)
    for all c in P do
        L ← 0
        while task not started do
            j ← GETTASKSIZE(L)
            if length of P[c][level] >= j then
                p ← subset of P[c][level] of length taskSize
                MATHMAKERTASK(p)
            L ← L + 1

procedure GETTASKSIZE(L)
    if L%2 == 0 then                          ▷ If level is even
        return 2
    else                                      ▷ If level is odd
        return 3
```

The "Pyramid" storage structure Once the reads have been categorised as per Algorithm 2.7, they are ready for further pair sorting. The result of Algorithm 2.7 is a set of BAM files containing reads of only a single c. We shall refer to these files as t_c. Once returned to the *PairFinder* process, the unpaired reads will remain within the *PairFinder* until the mate has been found.

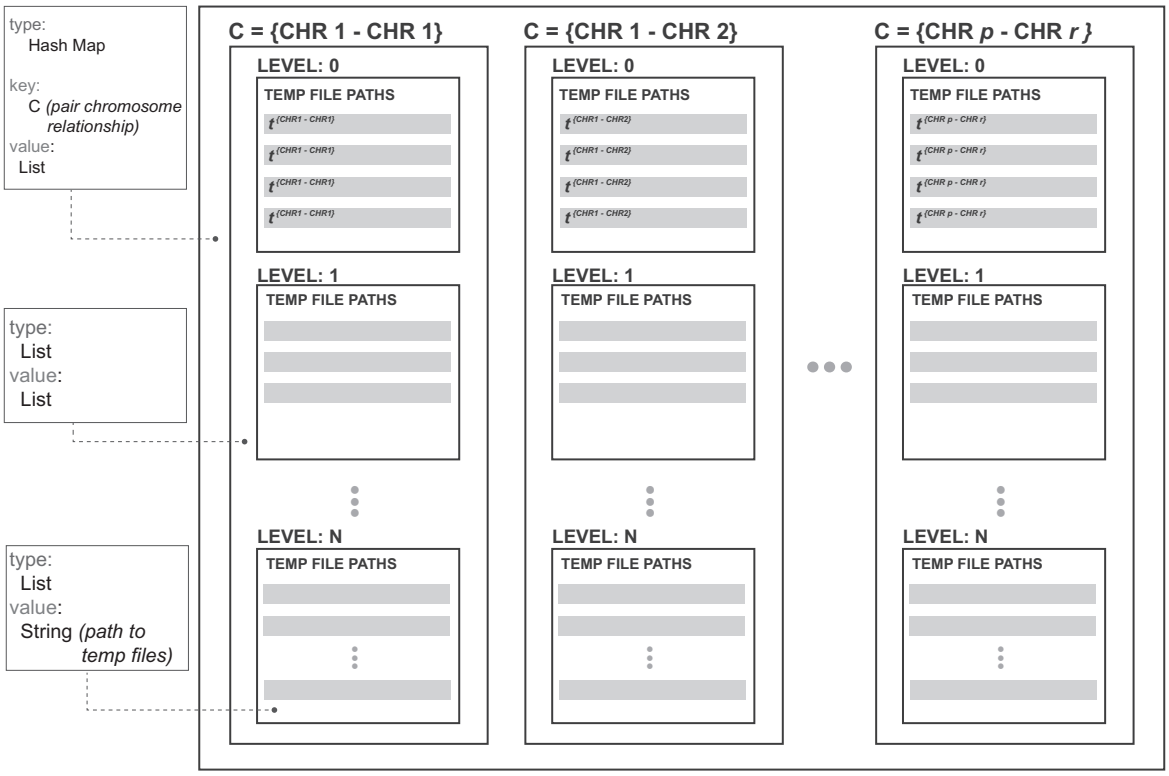


Fig. 2.3 The data structure used to store temporary files containing unpaired reads.

To enable the search of pairs within the relevant subcategories, c, the *PairFinder* makes use of the data structure depicted in Figure 2.3. This structure is referred to throughout the text as the "Pyramid". This name is meant to illustrate the pyramid like nature of the structure in that bottom levels have many files whereas upper levels have few. This level based structure is crucial to the way in which Parabam minimises read comparisons as will be made clear in the following sections.

The Pyramid is a nested data structure containing three layers. The outermost layer is a HashMap. Each key in this HashMap is a string representing a category of c. The value of this outermost HashMap is a list of lists. The first element of this list of lists can be thought of as the first level of the pyramid. All of the newly created incoming instances of t_c, created by the categorisation process, are inserted at this point of the data structure.

Using "levels" to reduce redundant comparisons So far we have seen that reads which were unpaired in the distributed Tasks are transferred to the *PairFinder* where they are sorted into different files (t_c) where all the reads in this file are from the same chromosome category,

c. Once these reads have been categorised into the correct t_c they are stored at the bottom layer of the Pyramid file structure.

From here, the *PairFinder* must coordinate the search for pairs amongst these files whilst continually adding new instances of t_c to the pyramid as they are received. To do this the *PairFinder* chooses a set of files from a single level within the Pyramid and sends them for comparison to one another in a distributed process. This distributed process is referred to as the MatchMaker task and is detailed once this search has completed, the reads that remain unpaired are then re-entered into the Pyramid at the next level. In this manner read-pairs make their way up the Pyramid.

Level based searching is another way in which Parabam reduces unnecessary comparisons. This method is a convenient way to compare instances of t_c which were generated from regions of the genome in close proximity, thus accounting for BSRs. For BSRs the missing pair will be located in a batch that was generated from an adjacent region of the genome. Therefore, we wish to concentrate our search on instances of t_c which were generated in close proximity to one another. In this level based system, instances of t_c that were created from proximal regions of the genome will naturally find themselves at the same level of the pyramid and so are more likely to be compared sooner.

There is another benefit to level based searching. Comparisons are only made once new instances of t_c arrive at a relevant level. This is an automated way of avoiding comparing the same reads over and over again, without having to keep track of which reads have already been compared. To enable this functionality the Pyramid chooses a different number of t_c for searching depending on whether the level number is odd or even.

For instance, consider a situation where the Pyramid currently contains three files at Level 3. As this level number is odd, three files are all selected for processing and a search for paired reads is conducted by the MatchMaker task. Three files containing reads that were not paired during the analysis are reinserted into the pyramid at Level 4. Level 4 is an even numbered level and so it requires that two files are present to conduct searching. Two of these files are chosen and sent for analysis. After this process two files with unpaired reads are inserted into the pyramid at Level 5. To recap, we now have 1 file at Level 4 and 2 files stored at Level 5. As things stand the *PairFinder* will not make any new comparisons until a new instance of t_c arrives at Level 4. Pseudo-code for this process is given in Algorithm 2.8.

The distributed MatchMaker task We have already made mention of comparisons being made by a distributed task. The workings of this distributed task, referred to by the name "MatchMaker", are detailed in this section.

The "MatchMaker" process searches for pairs within instances of t_c. MatchMaker runs as a constant child process of PairFinder. This relationship is similar to the child parent relationship of *Task* and *FileReader*.

As with all of the other processes detailed in this chapter, the MatchMaker waits for input on an incoming FIFO queue and then processes these inputs accordingly. In this case the *MatchMaker* process receives paths to specific instances of t_c. Once received, the process iterates through the provided t_c files and identifies read-pairs, as shown in Algorithm 2.9. By the end of the MatchMaker process all newly discovered read-pairs are processed by the user-rule and all unpaired reads are returned to the PairFinder. Crucially, as the last line of Algorithm 2.9 shows, the level is iterated. The resulting subset of unpaired reads t_c is stored at the next level of the pyramid.

Algorithm 2.9 Comparing the contents for N instances of t_c to find paired reads therein. X is a list of instances of t_c. L corresponds to the level from which the relevant t_c originate. Passing over each file twice ensures that burden on system memory is reduced.

function MATCHMAKERTASK(L,X)

▷ H is a record of whether a pairs occurs within all instances of t_c
$H \leftarrow HashMap$
for all t_c **in** X **do**
  for all a **in** t_c **do**
    $H[a.readName] = H[a.readName] + 1$

▷ R is a hash map containing tuples of newly paired reads
$R \leftarrow HashMap$
$t_c^S \leftarrow$ *New instance of* t_c
for all t_c **in** X **do**
  for all a **in** t_c **do**
    if $H[a.readName] == 2$ **then**
      $R[a.readName].insert(a)$
    else$H[a.readName] < 2$
      ▷ a remains unpaired. Store to send back to PairFinder
      $t_c^S.write(a)$

USERRULE(R)
return t_c^S, L+1

2.3.3 Ending the *PairFinder* process

In an ideal case, all of the reads in the BAM file are paired (meaning that no reads remain in the pyramid) and the *PairFinder* can simply terminate once it has received a signal from the MainHandler that processing of the BAM file has stopped.

However, in some cases BAM files contain reads where no pair is present in the BAM file. This causes a problem as these reads will never be removed from the *PairFinder* and the process cannot terminate. In order to account for this behaviour we introduced several key pieces of functionality to the method. Both of these methods occur once the *PairFinder* has received the signal that guarantees no more reads will enter the PairFinder.

If the amount of unpaired reads is small enough that all of the remaining t_c can be loaded into a single MatchMaker process, then we can simply send all of these files to a single instance of the MatchMaker. If any reads are remaining then we know that reads lack a pair in the BAM file and can be discarded from the analysis.

Where a great many reads are left, across many different groups of c, then it is not possible to simply compare all of them in a single MatchMaker task. In this case we used an approximate method such that the *PairFinder* begins to keep a record of how many times it has made a comparison between a set of files with the result that no pairs were found (this counter is referred to in the code as a "stale counter"). Once a certain amount of these comparisons are conducted we end the analysis with a warning that some reads were unpaired.

This method is only used in extreme cases and so far we have not observed a case where a BAM file in which all reads are paired has had to terminate in this manner. Given that Parabam has been run widely this is a good indication of the robustness of this approach.

2.4 Runtime

In this chapter we have described the core components of Parabam and how they interact with one another. Now we will detail the life-cycle of the program at runtime, focusing on several parts of the implementation which required specialised solutions to overcome implementation challenges.

2.4.1 Starting the analysis: flexible initialisation

The flexible nature of the Parabam architecture presents several challenges at initialisation. How can the system be initialised to cater for the user's specifications (i.e. in the correct op-

erating mode), allocated the correct resources (i.e. number of *Task* processes) and connected such that all of the individual processes are correctly connected by FIFO queues?

The Leviathan To meet these challenges we developed the *Leviathan* class, implemented as part of the `parabam.core` package. The *Leviathan* is completely agnostic to the desired architecture of the program. It exploits a feature of the Python programming language, dynamic class loading, to build Parabam from a set of variables which inform the *Leviathan* the way in which to initialise Parabam.

The primary benefits of this approach are that it reduces code duplication; separate code is not required for initialising the program in different modes with different combinations of *Handlers* or *Tasks*. This approach lends itself to the expansion of the program. For instance if we wished to add a new operating mode to the program we could simply hand the *Leviathan* the relevant variables, rather than having to separately define the initialisation sequence. Furthermore, this method decouples the initialisation sequence of the program from internal changes to the code of the various processes.

Instructing the Leviathan The *Leviathan* is called by an implementation of the Interface class, defined in the `parabam.command.core` package. In turn, the *stat* and *subset* class each have their own inherited implementations of this class. This allows the separate operating modes to define their own class to the Leviathan.

The *Leviathan* expects the following variables in order to initialise the analysis:

- `fileReaderClass` A class that inherits `parabam.core.FileReader`. Both the *subset* and *stat* interfaces pass the default class implemented in `parabam.core.FileReader`.
- `queueNames` A list of strings that represent different FIFO queues to be used in this analysis.
- `handlerBundles` A doubly nested hash-map where each key is a class of type hash-map and the value is a hash-map with values “inqu” and “outqus”. The value of inqu is a name of a queue to which this handler will listen for incomming packages. The value of “outqus” is a dict containing queues that this Handler will output to.
- `handlerOrder` A list of type Class. The list should be in the order that the Handlers will be destroyed in.

From these four variables the *Leviathan* may then construct the desired program architecture. First, the `fileReaderClass` is initialised and the process is started. As part of the

initialisation, the *FileReader* is passed the desired number of child *Task* processes and is responsible for their initialisation.

Next, each *Handler* listed in the `handlerOrder` variable is initialised using the relevant variables described in the `handlerBundles` hash-map. Queues are constructed and assigned by the *Leviathan* according the the `queueNames` list. The resultant queues are supplied to the relevant handlers (as specified by the inqu and outqus variables in the `handlerBundles`).

After this process is complete the *Leviathan* starts each of the handler processes and the analysis begins.

2.4.2 During the analysis

Once the analysis has begun the *FileReader* starts to process reads from the specified BAM file stored at a location on the user's disk. This process will continue until all of the reads in the file have been read and processed.

We have already seen how Tasks apply the user-rule to each read and this constitutes the main computational workload for Parabam during the analysis. In this section we detail processes, carried out during the analysis, that required non-trivial solutions.

Merging subset output As part of the *subset* operation mode it is vital to merge BAM files in an efficient manner. When Parabam is operating in *subset* mode the *Task* processes output temporary files containing all of the reads that met the criteria for inclusion in the output subsets. These files must be merged into a single unified output file to be returned to the user. This merging process is implemented as an *AuxiliaryHandler* referred to in the code as the *Merger*.

To accomplish the task of merging all of the temporary files into a single output file the *Merger* implements the algorithm shown in Algorithm 2.10. The *Merger* maintains an open connection to the output file for the duration of the analysis. As temporary files are received from the input FIFO queue they are gradually incorporated into the main output file.

One crucial optimisation of the *Merger* process is that it does not decompress the temporary files. Instead, we read the file in its compressed binary format straight into the output file as shown in the `dumpToFile` function in Algorithm 2.10. This means that there is no need to decompress and interpret each read in the file. This confers a substantial saving. This method allows Parabam to subset a file as quickly as the disk will accept write requests.

We see in Algorithm 2.10 that files with less than 1000 reads are combined into a holder BAM file first before being dumped into the main output. Here we simply extract reads from the BAM file in the usual way and write them directly into the holder file. As we saw in the introduction, BAM files are comprised of many constituent BGZF blocks.

Incoming temporary files with few reads naturally contain small blocks, far from capacity. By combining these small files we can write full BGZF blocks to the output file.

Algorithm 2.10 An overview of the Merger process. Where S is a BAM file containing reads to be merged into the subset.

```
function MAINLOOP
    O ← new empty BAM file
    T ← new empty BAM file                          ▷ File to be output to the user
    while processing do
        ▷ Get new package from input queue
        S ← inqu.get()
        if S.count > 1000 then
            DUMPTOFILE(O, S)
        else
            MERGESMALLFILE(T, S)
            if T.count > 1000 then
                DUMPTOFILE(O, T)
                T ← empty BAM file
                                                    ▷ BAM file to merge into output
function DUMPTOFILE(O, S)
    h ← location of first byte after header
    S.seek(h)
    f ← True
    while f do
        b ← 1MB of uncompressed binary from S
        ▷ eofByteSignature is a 28Byte sequence that all valid BAM files must end with
        if b.endsWith(eofByteSignature) then
            ▷ End of file found
            f ← False
            S.write(b) − 28                         ▷ Do not write the EOF to output file
        else
            S.write(b)
```

Pause signalling in Parabam In the process of developing Parabam we encountered a problem where early prototypes of Parabam would hang unexpectedly and then fail to complete when in pair processing mode. We noticed that the problem occurred when the *PairFinder* process became particularly busy and began to receive pairs for matching faster than it could process them. We diagnosed the problem as a deadlock occurring when two or more of the FIFO queues used for communication between processes become full. Such a scenario can easily occur when many requests are being made of the PairFinder

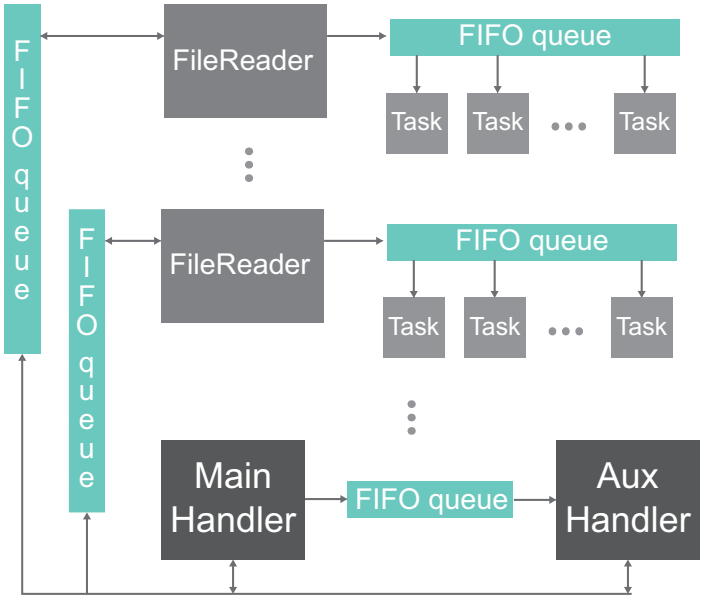


Fig. 2.4 A diagram showing the structure of the pause queue system

To overcome these problems we developed a method for the *PairFinder* to communicate to the *FileReaders* that processing should pause temporarily. This pausing system is implemented as a series of FIFO queues. At one end of each FIFO queue is a *FileReader* object. Attached to the other end of the FIFO queue is the PairFinder. Figure 2.4 shows this arrangement. When the *PairFinder* requires that processing should pause, it sends a pause request to each pause queue, before continuing processing it waits until it has received confirmation that each *FileReader* has received the pause instruction.

2.4.3 Ending the analysis: controlled destruction

Once all reads have been processed and allocated to *Tasks* by the *FileReader*, each instance of the *FileReader* sends a destroy signal to all of it's assigned *Tasks*. This signal proliferates through the system, signalling to downstream *Tasks* and *Handlers* that all reads have been processed and that the procedures to end processing may begin.

These procedures differ depending on the process and the operating mode of the analysis. For instance, in *subset* mode the *MainHandler* must ensure that all temporary files have been sent to the *Merger* before ending processing.

2.5 Benchmarking

In this section we test Parabam under a varied set of operating circumstances and, where possible, compare its performance to other tools. All of the benchmarking tests in this section were run on a conventional desktop computer with 2.5Ghz Intel Xeon Processor (providing 8 separate cores), 32GBs of 667Mhz DDR2 RAM, running a standard distribution of Ubuntu Linux version 14.04.

The tests in this section were run on a subset of reads from a Prostate ICGC BAM file. We used a subset in order to make the benchmarking process more timely. We randomly selected one quater of the read-pairs from within the original file. As a result the BAM file that benchmarking was run on contained 635,805,818 reads. [4]

2.5.1 Single end processing

First, we compared Parabam and Sambamba when creating a simple subset using single processing. In this test a read was included in the subset if the first 8 bases were equal to "A" (representing the Adenine nucleobase). This check can be conducted in constant time and as such is not computationally burdensome. Figure 2.5 shows when using Sambamba's inbuilt filtering system, it is much faster to create simple subsets than with Parabam.

These observations indicate that when the computation in the user rule is minimal, file IO (i.e. the reading of information from the BAM file) is the bottleneck in analysis. Thus, for cases where the user rule conducts computation where complexity is constant adding Tasks to the analysis has little affect on the duration of the analysis (Figure 2.5 A). What's more, most of these use cases are better served by Sambamba as it is capable of reading files from BAM files faster than Parabam and has an interface which allows a user to generate simple subsets.

However, when we consider a case where the computation required to create the subset is more complex, the parallel design of Parabam starts to confer benefits in reducing the duration of the analysis.

To test this we devised an analysis whereby a read was included in the subset only if the GC ratio of the read's sequence was exactly 50% and the first letter in the read's quality score was C. This subset requires inspecting the nucleotide at each locus in the read's sequence and so is of $O(n)$ complexity. We included the check on the first base so that the resultant

[4] Parabam was actually used in the creation of this subset BAM file. We applied a user-rule that used a random number generator to only include reads in the subset 25% of the time. This code is included in Appendix A

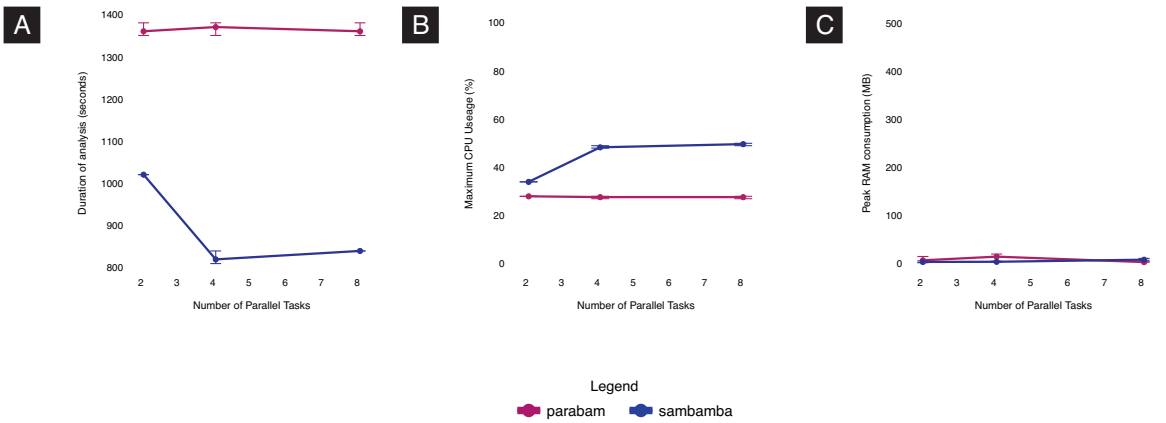


Fig. 2.5 Benchmark results for single processing mode with O(1) read operation complexity ***(A)****: Duration of analysis* ***(B)****: Peak CPU usage* ***(C)****: Peak RAM usage*

subsets were smaller. This meant that we could run many benchmarking tests sequentially without the risk of the hard disk becoming full.

A direct comparison to any other program is difficult in this scenario. No other program allows users to conduct computation on individual reads in parallel in the same way as Parabam. Indeed, the results that follow make reference to subsets created by Sambamba, but in order to achieve this functionality we had to pipe the output of Sambamba into a short Python script which we devised, thus negating a direct comparison between the programs. However, this does represent a viable alternative to using Parabam for this analysis and is perhaps the next best way of generating this subset. In this way it is a useful comparison, even if it uses Sambamba in a way that was not intended by Sambamba's authors.

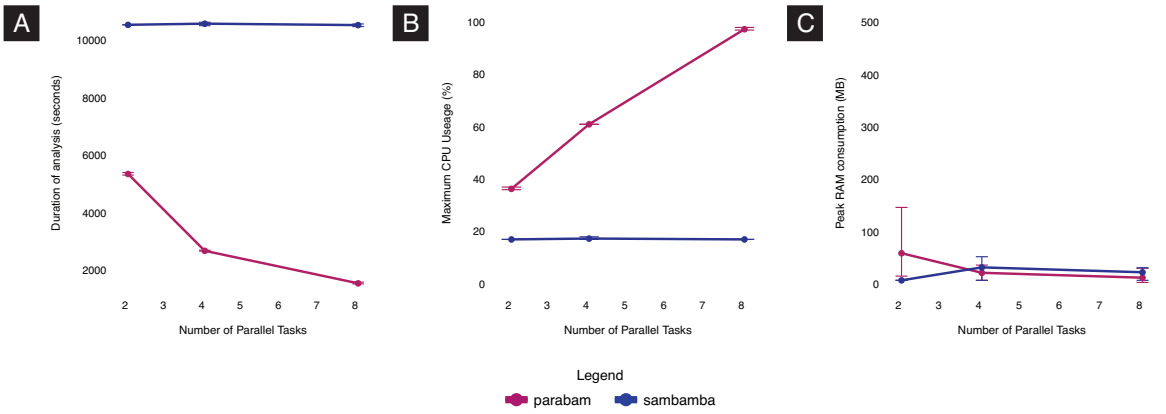


Fig. 2.6 Benchmark results for single processing mode with O(n) read operation complexity ***(A)****: Duration of analysis* ***(B)****: Peak CPU usage* ***(C)****: Peak RAM usage*

Figure 2.6 shows the results for generating subsets of reads where GC content is actually 50% in single end analysis mode. We see that as we increase the number of Tasks available

to Parabam the duration of the analysis decreases. The same cannot be said for our attempt to construct the subset using Sambamba. Whilst Sambamba is able to more quickly process reads from the BAM file when provided with more processes, the bottleneck is now located in the secondary step conducted on the output of the program and so analysis duration is not decreased. There is no way to internalise this process with Sambamba and so piping the output of Sambamba to a secondary process is the only way of generating this subset with the program. We also see that, as shown in Figure 2.6C, these analyses do not consume large amounts of RAM and what's more, increasing computational power does not increase memory usage.

2.5.2 Pair processing

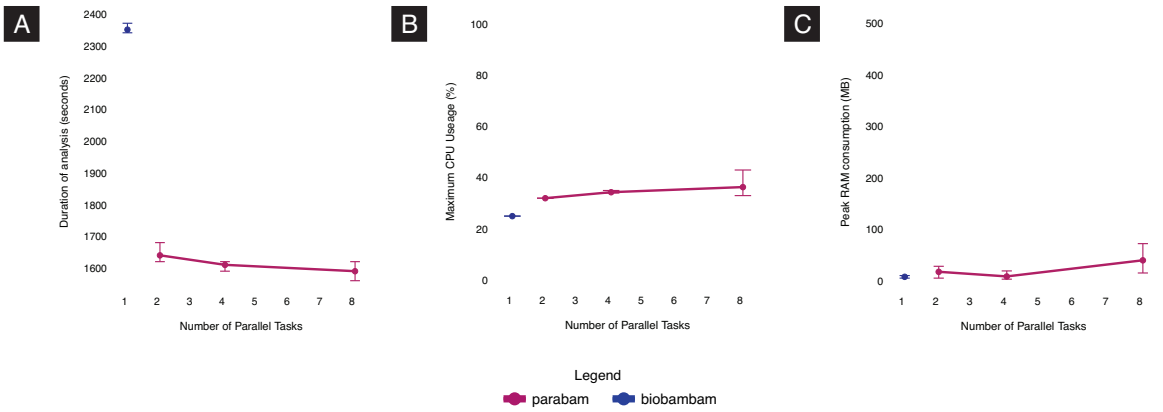


Fig. 2.7 Benchmark results for pair processing mode with O(1) read operation complexity ***(A)****: Duration of analysis* ***(B)****: Peak CPU usage* ***(C)****: Peak RAM usage*

No other tool allows the user to process an index sorted BAM file as if it were pair sorted *in parallel*. However, the Biobambam program implements similar functionality without parallel processing. Even though there is some overlap in the pair processing functionality it is still not possible to make a direct comparison between Biobambam and Parabam. As with Sambamba in the previous section, to compare Biobambam to Parabam we wrote our own intermediary Python script that interpreted the output of Biobambam.

To compare the runtimes of Biobambam and Parabam in pair processing mode we ran two separate tests. In the first test, we generated a subset that required computation of constant complexity on each read by checking whether the first 8 bases of the read's sequence are comprised only of A. If either read in the pair displays this sequence, then both reads are included in the subset.

In contrast to single end processing, we observe that for most analyses conducted in pair processing mode, the bottleneck is no longer file IO and rests instead in the pair matching procedure. This is demonstrated by the fact that even for cases where the user-rule complexity is $O(1)$, the duration of the analysis is slightly decreased when we provide additional tasks. This is shown in Figure 2.7A where the time taken to assemble a subset requiring $O(1)$ complexity is reduced by providing more distributed Tasks to the analysis. We also see in Figure 2.7 that Parabam is substantially faster than Biobambam in this scenario.

Next we compared the performance of Biobambam and Parabam when linear complexity was required to determine inclusion in the subset for each read. We applied an adapted version of the rule used in the single processing benchmark. Reads were included in the subset if their sequence content was exactly 50% GC. If either read in the pair displayed this feature, both reads were included in the subset.

As with single end processing, when more complex computation is required to create the subset Parabam comes into its own against other methods. We see in Figure 2.8A that the duration of the analysis is decreased substantially as parallel tasks are added to the analysis.

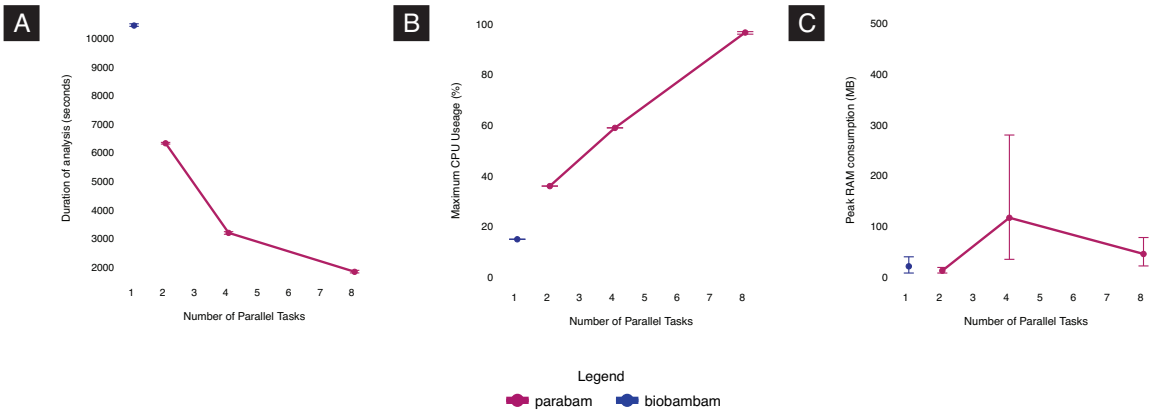


Fig. 2.8 Benchmark results for pair processing mode with O(n) read operation complexity ***(A)****: Duration of analysis* ***(B)****: Peak CPU usage* ***(C)****: Peak RAM usage*

2.6 Summary

In this chapter we have described in detail the inner workings of Parabam and benchmarked its performance. Parabam is a flexible program which lends itself to use by people who are not experienced programmers and whom do not wish to write multiprocessing code.

Contrasting functionality Where possible we have made comparisons to other programs which provide functionality for subsetting BAM files. However as we saw in the bench-

marking carried out in Section 2.5 these comparisons are limited because for any subsetting functionality more complicated than simple logical operators, we need to pipe output of these programs into a separate single process thus negating any inherited speed advantage, most noticeably for Sambamba.

This incomparability speaks to the unique nature of the Parabam platform. There are clearly non-overlapping use cases between Sambamba, Biobambam and Parabam. While we have shown benchmarking for the *subset* operating mode in this chapter, we have not given an application of the *stat* operating mode. It would be harder still to draw a comparison for this operating mode as it is a novel way of processing BAM files, however the *stat* mode has come into its own when processing data for Telomerecat. While the *stat* mode has not been fully explored in this chapter the Appendix A does list user-rules for the *stat* mode which might give some flavour of its usefulness.

Perhaps an area for future work is to combine the speed with which Sambamba can read BAM files with Parabam's interface for creating complex subsets and pair processing in parallel. Such a tool would make for a terrific blend of usability, efficiency and speed.

Use of Parabam in further analysis Parabam has been used extensivly throughout this thesis as the framework around which Telomerecat was developed. Parabam is used at several points in the Telomerecat analysis. Firstly, for the main computational overhead of the program, the collecting of telomere reads from the input BAM files. Additionally, Parabam is also used to conveniently process these resultant subsets. As we shall see, Telomerecat exploits the paired-end nature of modern sequencing data, so Parabam is in its element in processing these files.

This use of Parabam, as the framework for other dedicated tools, demonstrates the benefit of providing the Python API. Code duplication has been greatly reduced, to the extent that the "bam2telbam" command in the Telomerecat program is essentially just a wrapper for a call to Parabam.

The way in which Parabam is implemented encourages the user to create smaller subsets of relevant reads and conduct more complex analysis on these narrowed subsets. We have found this methodology for program development as highly beneficial. Inevitably throughout the development of Telomerecat the length estimation portion of the tool has undergone changes that required us to run the tool on the same sample multiple times. This task of refinement would have been made substantially more difficult had we not separated the analysis into subsetting and downstream analysis steps.

Furthermore, this design pattern of using Parabam to create specialised subsets has implications on reproducibility. In the case of Telomerecat the subset BAM files containing

telomere reads are substantially smaller than the original BAM files. We envisage that these subsets could be made available in place of the larger BAM files making it easier for researchers to test and verify each others results using the tool.

Using Python as the primary development language At several points in the development of Parabam we considered moving away from the Python programming language. Python has many advantages as a language, however, it lacks the processing speeds of compiled languages like C or Java.

Despite not being the fastest language Python remained as the primary language for the development of Parabam. The excellent multiprocessing framework coupled with quick development time due to the uncluttered syntax made a good choice for development. Furthermore we were able to increase the processing speeds of Python but optimising the code with the Cython package. Cython converts Python to C and compiles the language. This removes reliance on the Python interpreter for execution and thus provide a substantial processing speed increase.

Improving pause signalling Currently, Parabam implements pause signalling as detailed in Section 2.4.2. This system must be improved upon in future iterations of the program. The current implementation of the pause signalling system means that despite each auxiliary handler having access to the pause queues, in practise only one of them, the PairFinder, may use the system. As it is currently implemented if two AuxiliaryHandlers were to submit pause requests simultaneously, the system would crash.

One solution to this problem is to introduce another component to the Parabam architecture who's sole function was to coordinate the pause logic. In this scenario, all of the pause logic would be contained in a single process and Handlers could make a single request, via the FIFO queue to this object which would then handle the pausing of the *FileReaders*.

Chapter 3

Telomerecat: a novel method for estimating telomere length

The original concept for a telomere length estimation algorithm that could account for aneuploidy using boundary reads was conceived by Dr A. G. Lynch. The method for interstitial telomere filtering was developed in collaboration with Dr M. L. Smith. All other method creation, development and analysis was undertaken by the author.

This chapter incorporates material that was published previously in FARMERY et al. *(2018). Particularly, Sections 3.2 and 3.3 that make use of figures, formulae and text from the aforementioned paper.*

In the introduction to this thesis we described the ontology of the telomere, its role in cell and its role in cancer. We also saw how previous methods have attempted to estimate TL from WGS data. In this chapter we outline our *de novo* computational method for estimating mean telomere length from WGS paired end data: Telomerecat.

First, we outline the motivations behind the creation of Telomerecat. We then explain the method in its current form, and detail the validation we carried out to test and confirm the method's accuracy. Later, in the discussion section of this chapter we will reflect on the novel aspects of the Telomerecat algorithm and consider the advantages and disadvantages of Telomerecat over other methods.

3.1 Motivations

The motivation for the development of Telomerecat stems from the need to account for disrupted genomes commonly found in cancer sequencing experiments. Existing attempts to profile telomere content from WGS experiments rely on an assumption regarding the number chromosomes in a sample. These assumptions would not hold for samples from cancerous tissue.

In cancer, assumptions of chromosome count are unfounded. Many cancers display an atypical amount of chromosomes. Atypical chromosome counts can manifest as aneuploidy (an abnormal number of complete chromosomes in the cell) or euploidy (an atypical amount of copies of a single chromosome). It follows that if a cancer cell displays aneuploidy or euploidy then it can not be assumed that the number of telomeres in the cell remains normal. Indeed, a sequencing experiment from a cancer sample may have more telomere reads because it has *more* instances of telomere, not because it has *longer* telomere.

All methods for estimating TL must find a way of adjusting the observed counts of telomere read by the depth of sequencing coverage. To illustrate, consider a case where a method identifies two samples as having 10,000 telomere reads each. However Sample A is a 30$\times$ coverage experiment and Sample B is a 15$\times$ coverage experiment. In this straightforward example we would expect the method to report that Sample A has shorter telomeres than Sample B because they have the same observed reads despite Sample A having double the coverage.

Now let us consider that Sample A has 33 chromosome pairs whereas Sample B has the conventional 23 pairs. In Sample A there is now more genomic content than in Sample B. In Sample A, the direct relationships between estimates of sequencing coverage, ploidy and TL are broken. This affects TL estimation in two ways.

Firstly, any estimate of coverage for Sample A must take into account the increased genomic content; previous methods do not.

Secondly, the aneuploidy in Sample A means we are no longer certain that each chromosome will still possess two telomeres; previous methods assume all samples have the normal 92 individual telomeres[1]. Clearly, the observed 10,000 telomere reads in Sample A and Sample B are no longer directly comparable even when we adjust for sequencing coverage.

We hypothesised that telomere coverage could be determined vicariously by observing the ratio of telomere reads to reads on the boundary between telomere and subtelomere.

By considering this count of boundary reads we can indirectly observe the amount of telomeres in the sample. Put simply, if there are more telomeres in the sample then there

[1] One telomere appended to each end of each chromosome ($23 \times 2 \times 2$)

should be more reads at the boundary. The boundary count also captures information about the sequencing coverage in the sample. If sequencing coverage is higher then there will be more reads at the boundary. Observing boundary counts negates the need to quantify the exact number of telomeres and also allows Telomerecat to estimate coverage in a way that considers the challenges imposed by aneuploidy.

Further, we also noted that reads stemming from interstitial telomere repeats (ITS) could obscure estimates, as ITSs produce reads that appear telomeric, but are not telomeric, and so should not be counted when attempting to estimate telomere length. Pre-existing methods failed to consider noise caused by sequencing reads from ITSs.

With these considerations in mind we aimed to produce a tool that could be applied easily to a wide variety of paired end sequencing data. We wished the tool to account for aneuploidy and filter interstitial reads from the analysis. While we hoped the tool would have special relevance to cancer we also wished to create a tool that was versatile, quick to run, easy to install and easy to use.

3.2 A description of the method

3.2.1 An overview

Telomerecat [2] comprises three main steps, as outlined in Figure 3.1.

In the first step we iterate through all of the read-pairs in a given BAM file and collect all the read-pairs that have at least two occurrences of the telomere hexamer. This collection of read-pairs is referred to as a TELBAM throughout this thesis. Further detail about the generation of the TELBAM is given in Section 3.2.2

Once we have collected these candidate telomere reads in a TELBAM, we proceed to the second step of sorting each read in categories defined by whether or not they are telomeric. Before we can categorise a read we conduct several procedures on the read to determine its characteristics. Firstly, we check whether any of the loci in the read's sequence do not match the expected telomere sequence. Secondly, we interrogate these loci to determine their sequencing quality. We use this information to determine whether the read is suffering from sequencing error. This process of ascribing error in telomere reads is described in Section 3.2.4.

[2] Short for Telomere Computational Analysis Tool

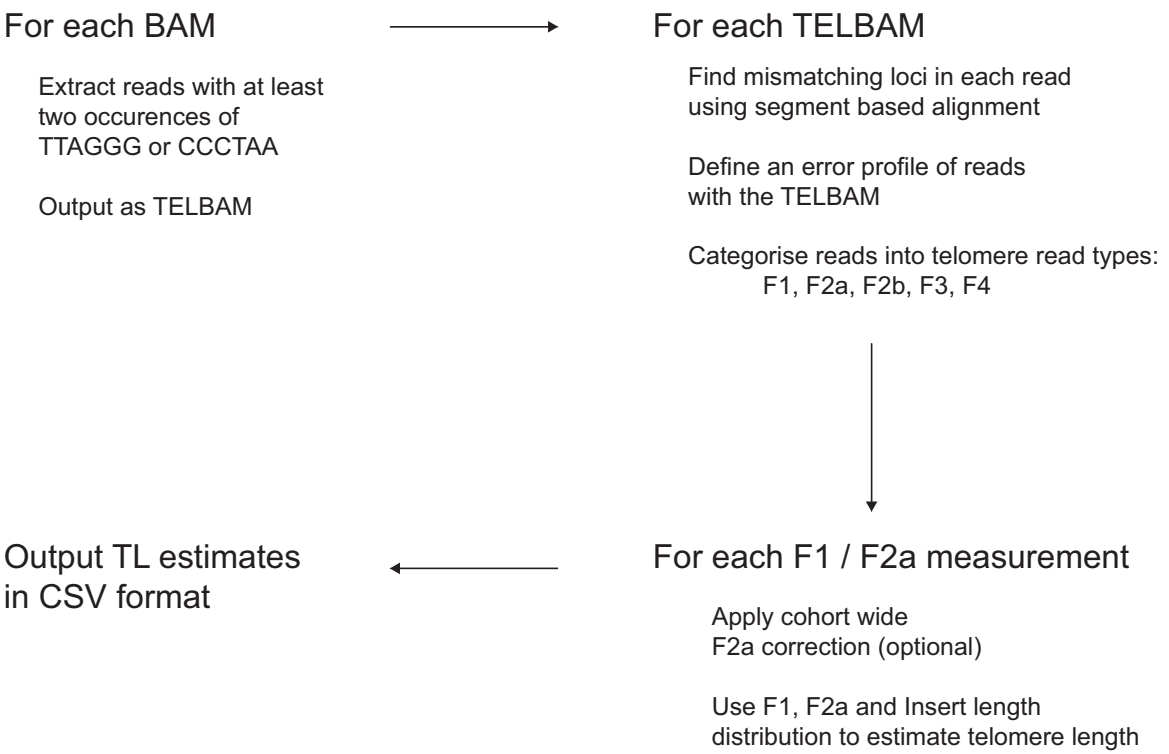


Fig. 3.1 An overview of the Telomerecat method

Once we have determined which reads are suffering from sequencing error we categorise the read-pairs according to the corresponding characteristics:

- Both reads in the pair are fully telomeric
- One of the reads in a pair is fully telomeric, the other is not
- Neither read in the pair is telomeric

Using these categories we determine two vital statistics for the sample: the number of reads stemming from the telomere and the number of reads on the boundary between telomere and subtelomere. Read categorisation is covered extensively in Section 3.2.6.

As we shall see in the following sections, the process of differentiating telomere and subtelomere reads is complicated by the similarity of telomere and subtelomere repeats, as well as the rate of sequencing error in telomere reads. Herein we detail the computational and statistical methods that we have developed in order to differentiate these read types.

Finally, we conduct the third step by using a simulation based approach to convert our counts of complete and boundary reads to generate an estimate of telomere length. This process is detailed in Section 3.2.8.

3.2.2 Collecting reads for analysis

Before we can estimate TL from a WGS experiment we must extract all of the reads that contain two occurrences of the telomere hexamer and their pairs. We refer to this subset of sequencing reads as a TELBAM.

This is a non-trivial problem. The output from sequencing experiments can often be on the order of a billions of reads. As we saw in the Introduction, reads are usually stored in compressed files called BAM files. Usually reads are stored in BAM files according to where they mapped on the genome. However, the telomere is not included in the reference genome and accordingly many telomere reads are unmapped. For the TELBAM to be generated swiftly, we wished for the processing to be undertaken in parallel.

Generating the TELBAM was the main motivation for the creation of Parabam (see Chapter 2). Parabam started as the processing framework for Telomerecat and became its own software once the code became increasingly complex. Accordingly, the generation of the TELBAM is simply invocation of Parabam's subset operation using the rule shown in Algorithm 3.1

Algorithm 3.1 The rule passed to the Parabam subset operation to create the TELBAM, where read1 and read2 are paired reads

```
function TELBAMRULE(read1, read2)
    read1Status ← CONTAINSTELOMERE(read1.sequence)
    read2Status ← CONTAINSTELOMERE(read2.sequence)
    if read1Status or read2Status then
        return read1 and read2
    else
        return None
function CONTAINSTELOMERE(sequence)
    return sequence.count(TTAGGG) >= 2  or  sequence.count(CCCTAA) >= 2
```

By orientating Telomerecat around the TELBAM, the algorithm is made more adaptable and reproducible. Telomerecat can be invoked from the command line in such a way that TELBAM generation is a separate process from length estimation. This enables the user to conduct the computationally expensive process of TELBAM generation on a separate computer to length estimation. Then, once a TELBAM is generated, the user can run multiple length estimation procedures without having to iterate through the entire BAM file again. Furthermore, should other researchers wish to reproduce an analysis, they may do so with access only to the TELBAM rather than the much larger BAM files.

This design paradigm has proven to be extremely useful throughout the development of Telomerecat and our investigations. While developing the method, we were able to generate

TELBAMS from a set of validation data only once. Once we had generated TELBAMS for each sample, we could then run multiple versions of the algorithm and compare results. This would simply not have been possible without the TELBAM as each run of Telomerecat would have taken a prohibitively long time to run.

A TELBAM will usually contain approximately one ten thousandth of all the reads from a WGS BAM file. This reduction in size ensures that the files are more easily stored and exchanged.

3.2.3 Identifying telomere sequencing reads

When we generate the TELBAM we use the requirement of two occurrences of the telomere hexamer within the sequence of either read in a pair. This specification is broad enough that it includes many reads which are not telomere. We can divide each read-pair in the TELBAM between the following categories:

- Read pairs originating from the telomere
- Read pairs from boundary between the telomere and subtelomere
- Subtelomeric and ITS read-pairs
- Read pairs that contain two occurrences of the hexamer by chance

Telomerecat uses only the first two categories to estimate telomere length: complete telomere read-pairs and read-pairs on the boundary between telomere and subtelomere.

In this section we detail how telomere reads are identified and categorised. Further, we explain how this classification inherently filters interstitial reads.

Before we can categorise read-pairs we first must determine what constitutes a telomere read. Sections 3.2.4 & 3.2.5 detail a method for determining whether individual *reads* are comprised of complete telomere. We then detail how we categorise each *read-pair* in Section 3.2.6.

3.2.4 Mismatching loci

Pre-existing methods use the count of telomere hexamers in a read to differentiate nontelomere and telomere reads. In contrast, Telomerecat considers whether a read is telomere by the amount of loci in the read's sequence that differ from the expected telomere sequence.

We developed an alignment method that allows us to interrogate reads for loci where the observed sequence is mismatched to an idealised telomere sequence. We refer to these as "mismatching loci" and later in the mathematic formulation as $\mathbf{g}$.

The alignment method is described in Figure 3.2. This method allows for insertion and deletions in the read's sequence that would otherwise obscure the true extent to which the read's sequence matches the telomere sequence.

Justification for this method, as well as its various advantages in opposition to existing methods, is discussed later in the chapter in Section 3.4.1.

3.2.5 Capturing sample-wide error with the error profile

Once we have collected the mismatching loci for all reads in a TELBAM we use this information to build a profile of the error within the TELBAM. We then use this error profile to define a subset of reads that are full telomere reads suffering from sequencing error.

We define P_{max} and P_{min} as the global maximum and minimum observed Phred score across all reads, and (L) as the read length used.

We let N represent the total number of reads in the TELBAM such that $\{0,1,n,...,N-1\}$ are indices representing each read. Values associated with the n^{th} read are denoted with a superscript (n). For example, the vector of Phred scores associated with the L locations in read n is denoted $\mathbf{p}^{(n)} = \{p_0^{(n)}, p_1^{(n)}, ..., p_{L-1}^{(n)}\}$. For the n^{th} read, let $m^{(n)}$ be a random vector in the space $\{0,1\}^L$ such that a 1 is found at each loci in the read that does not agree with the telomere sequence. In the case that the sequence is comprised of perfect telomere sequence then the vector should sum to zero. The method for obtaining $m^{(n)}$ via a fragmentary alignment method is shown in Figure 3.2.

Then define z^n (the number of mismatches for read n), and λ^n (the average Phred score at mismatches in read n) as:

$$z^n = \sum_{i=0}^{L-1} \mathbf{m}_i^{(n)}$$

$$\lambda^n = \left\lfloor \frac{\sum_{i=0}^{L-1} \mathbf{m}_i^{(n)} \mathbf{p}_i^{(n)}}{z^{(n)}} \right\rfloor - P_{min}$$

We then define an indicator function

$$\mathbb{1}(\lambda, z, i, j) := \begin{cases} 1 & \text{if } \lambda = i \wedge z = j, \\ 0 & \text{if } \lambda \neq i \vee z \neq j. \end{cases}$$

Algorithm 3.2 Algorithms to reduce noise on the error profile mask matrix E. Where $P_o = P_{max} - P_{min}$

function MASKCORRECTION(E)

$N \leftarrow$ NEIGHBOURCOUNT(E)
$C \leftarrow$ CONTINUOUSCOUNT(E)
$C^T \leftarrow$ CONTINUOUSCOUNT$(T(E))$
$E' \leftarrow$ *An* $L \times (P_o)$ *matrix where* $E'_{ij} = 0$
for i **in**$\{0, 1, ..., P_o\}$ **do**
  for j **in**$\{0, 1, ..., L\}$ **do**
    if $N_{ij} \geq 4$ **or** $C_{ij} \geq 4$ **or** $C^T_{ij} \geq 4$ **then**
      $E'_{ij} = 1$
    else
      $E'_{ij} = 0$
return E'

function NEIGHBOURCOUNT(E)
$N \leftarrow$ *An* $L \times (P_{max} - P_{min})$ *matrix where* $N_{ij} = 0$
for i **in**$\{0, 1, ..., P_o\}$ **do**
  for j **in**$\{0, 1, ..., L\}$ **do**
    $N_{ij} \leftarrow \sum_{x=-1}^{1} \sum_{y=-1}^{1} E_{i+x,j+y}$
    $N_{ij} \leftarrow N_{ij} - E_{ij}$
return N

function CONTINUOUSCOUNT(E)
$C \leftarrow$ *An* $L \times (P_o)$ *matrix where* $C_{ij} = 0$
for i **in**$\{0, 1, ..., P_o\}$ **do**
  $start \leftarrow 0$; $count \leftarrow 0$;
  for j **in**$\{0, 1, ..., L\}$ **do**
    if $E_{ij} == 1$ **then**
      $count \leftarrow count + 1$
    else if $E_{ij} == 0$ **then**
      $C_{i,start:j} \leftarrow count$
      $start \leftarrow j + 1$;$count \leftarrow 0$
return C

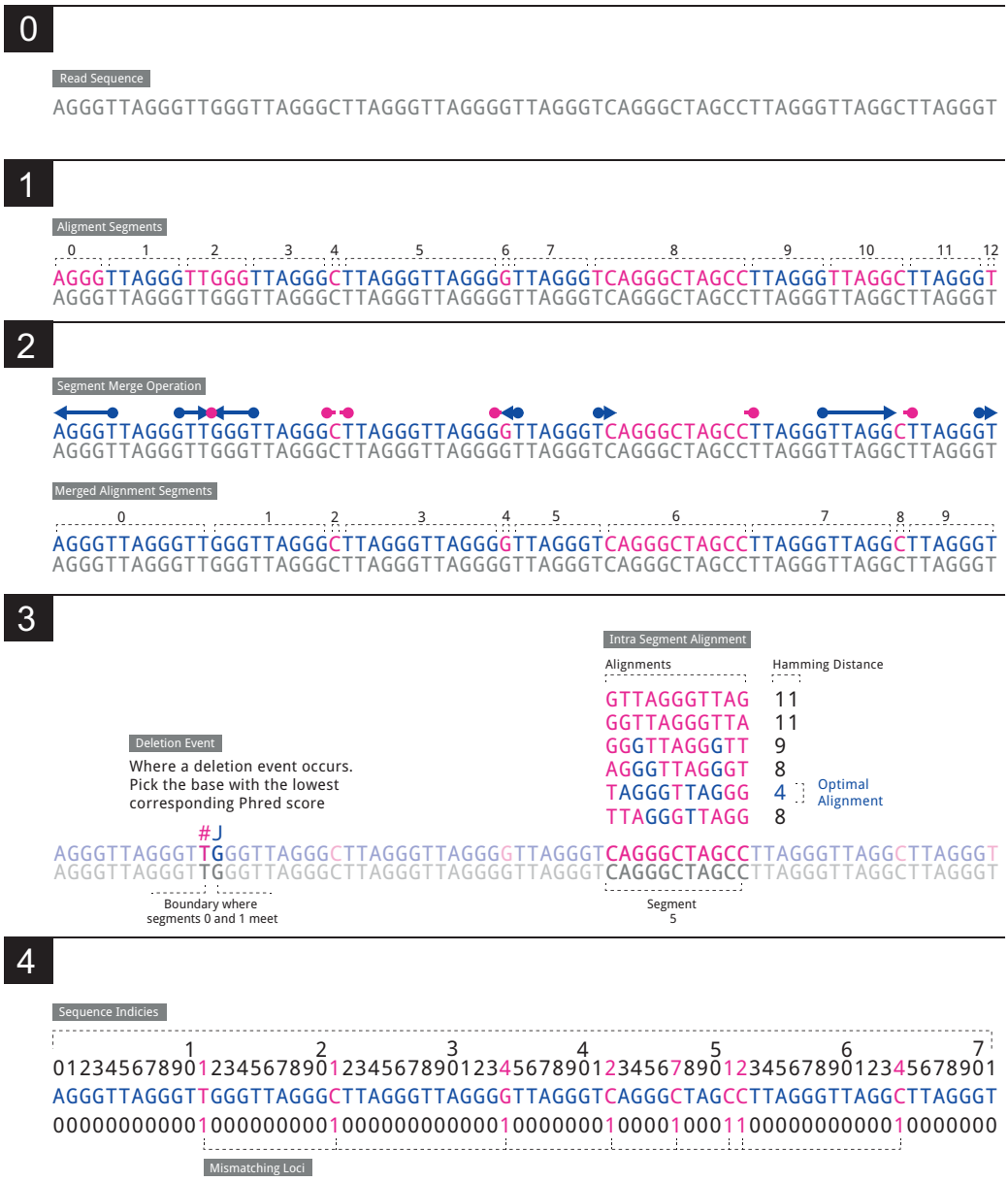


Fig. 3.2 The algorithm that determines the indices of divergence from the telomere sequence. ***0****: We observe a sequencing read* ***1****: We split the read into 'segments' (11 in total in our example) such that each segment is a substring of the original sequence and that every other segment consists of unbroken telomere sequence. In our example we see that segments 1,3,5,7,9,11 contain unbroken telomere sequence.* ***2****: Each segment containing a telomere hexamer is "expanded" to capture the full extent of the surrounding telomere sequence. The number of segments is reduced by 2.* ***3****: When two segments both containing the telomere hexamer are adjacent after step 2 this indicates a frame shift. We take the loci with the lowest corresponding Phred score. For any segment that does not contain a telomere hexamer and where the length of the segment is greater or equal to 4 apply we conduct a basic alignment of all possible telomere offset telomere sequences. The telomere sequence with the lowest Hamming distance is taken as a local alignment for that segment. Where two alignments are equal the one with the lowest average Phred score is preferred.* ***4****: Sequence loci that are not in a complete hexamer or were mismatched in the Hamming alignment step are taken as mismatching loci.* **m** *for this example is given in the final line of the diagram*

So that a matrix $\mathbf{X}$ takes the form,

$$x_{ij} = \sum_{n=0}^{N-1} \mathbb{1}(\lambda^{(n)}, z^{(n)}, i, j)$$

Where $i \in \{0, ..., P_{max} - P_{min}\}$ and $j \in \{0, ..., L-1\}$. Thus each x_{ij} in $\mathbf{X}$ records the number of reads with the relevant λ and z contained within the TELBAM and is depicted in Figure 3.3A.

Where $\mathbf{X}$ captures information about the average Phred score ($\lambda^{(n)}$) at $z^{(n)}$ mismatching loci, we seek to create an equivalent matrix $\mathbf{Y}$ about the average Phred score at $z^{(n)}$ *random* loci in the n^{th} read.

For the n^{th} read, let $r^{(n)}$ be a random vector in the space $\{0,1\}^L$ such that $\sum_{k=1}^{L} r_k^{(n)} = z^{(n)}$. That is, a vector for which the non-zero entries identify $z^{(n)}$ random loci within the read.

So that,

$$\mu^{(n)} = \left\lfloor \frac{\sum_{i=1}^{L} \mathbf{r}_i^{(n)} \mathbf{p}_i}{z^{(n)}} \right\rfloor - P_{min}$$

Thus,

$$\mathbb{1}(\mu, z, i, j) := \begin{cases} 1 & \text{if } \mu = i \wedge z = j, \\ 0 & \text{if } \mu \neq i \vee z \neq j. \end{cases}$$

$$y_{ij} = \sum_{n=0}^{N-1} \mathbb{1}(\mu^{(n)}, z^{(n)}, i, j)$$

As before, $i \in \{0, ..., P_{max} - P_{min}\}$ and j $\in \{0, ..., L-1\}$.

When we plot the matrices $\mathbf{X}$ (Figure 3.3A) and $\mathbf{Y}$ (Figure 3.3B) as heat maps we typically see that there is a striking difference in their composition. The heatmap for $\mathbf{X}$ shows an intensity in the upper left hand corner pertaining to reads with low Phred scores at mismatching loci. This hotspot is missing from the $\mathbf{Y}$ heatmap. We interpret this region as representing telomere reads affected by sequencing error that we wish to capture in our length estimation process.

We find the difference between the two matrices:

$$\mathbf{D} = \mathbf{X} - \mathbf{Y}$$

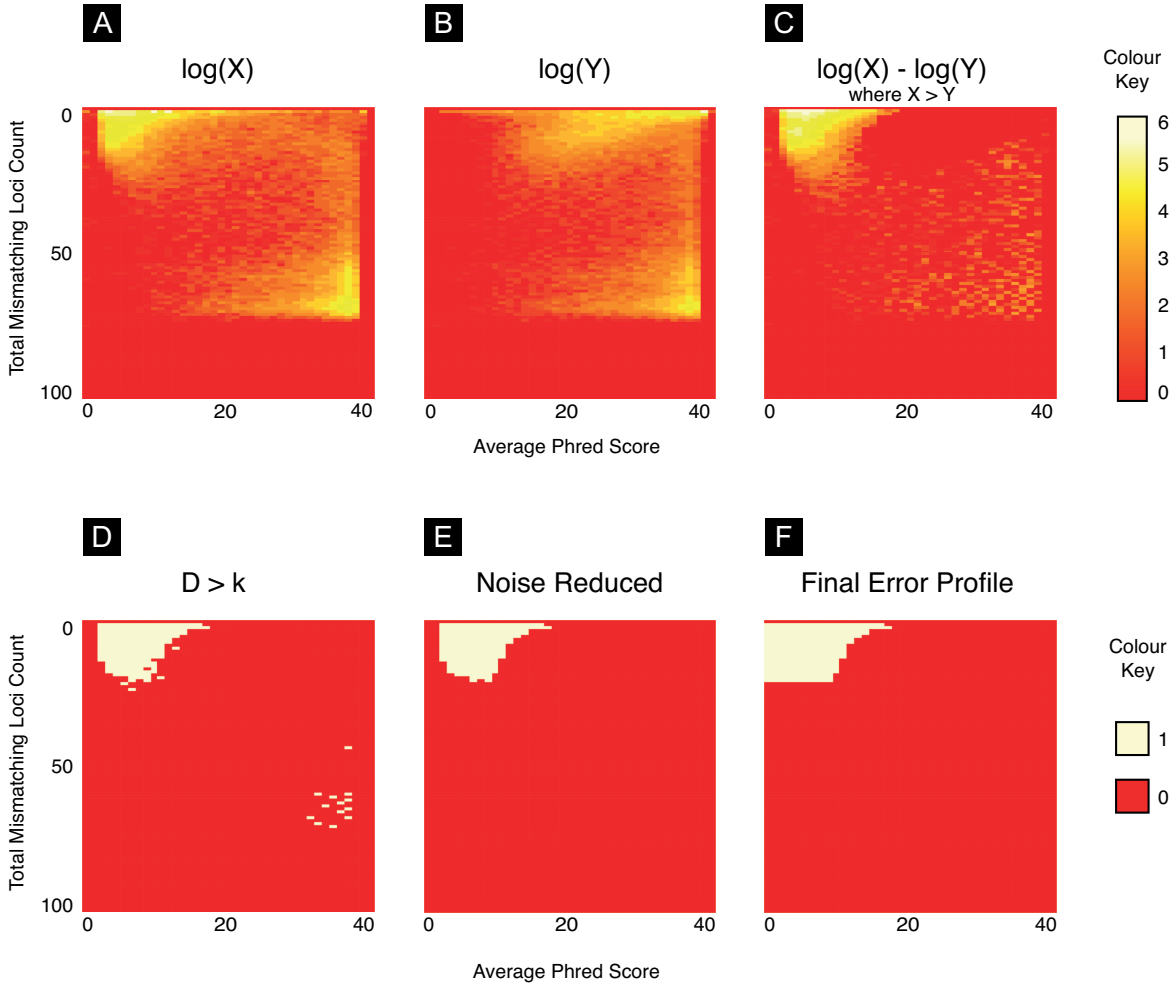


Fig. 3.3 ***(A):*** *A heatmap of the joint distribution of Phred scores a mismatching loci and the number of mismatching loci (***X***). The intensities in the top left corner of the heatmap indicate an association between fewer mismatches and lower Phred scores. We observe that the maximum mismatching loci is commonly ∼75% of the read length. This effect is caused by non-telomere reads match a the telomere sequence simply by chance* ***(B)****: A heatmap of the joint distribution of random loci in reads and the associated Phred score (***Y***). We note that the joint distribution of reads in the upper half of the matrix is different from that in* **X** *while the lower portion is identical.* ***(C):*** *The difference between* **X** *and* **Y**. *Referred to as* **D** *in the text.* ***(D):*** *A binary heatmap showing all cells in* **D** *that are greater than the threshold k. We note the preponderance of cells in the upper right hand corner of the figure* ***(E):*** *We remove noise from the figure using the methods detailed in Algorithm 3.2* ***(F):*** *We apply a final rule to ensure cells associated with low Phred scores are captured in the error profile, see Algorithm 3.3*

We plot values of $\mathbf{D} > 0$ as a heatmap in Figure 3.3C. To capture cells that contain more reads than we would expect at random we define a mask **E**. **E** is defined such that:

$$e_{ij} = \begin{cases} 1 & \text{if } d_{ij} > k, \\ 0 & \text{if } d_{ij} \leq k. \end{cases}$$

Where k is $\max\{\mathbf{D}_{ij}\}$ for all values where $\frac{1}{2}p < i \leq p$ and $\frac{1}{2}L < j \leq L$. This matrix is depicted as a heatmap in Figure 3.3D.

We note that the mask depicted in Figure 3.3D has gaps that appear as a result of using k as a threshold. We apply the procedure detailed in Algorithm 3.2 in order to remove noise from the error profile. The results of applying this procedure are shown in Figure 3.3E. We conclude by applying the operation described in Algorithm 3.3 and shown in Figure 3.3F. This is the final matrix and is provided to the read-pair classification procedure shown in Algorithm 3.4 as **E**. All read-pairs falling within the area by the error profile are counted as fully telomeric suffering from sequencing error.

Algorithm 3.3 Final step in producing the error profile

```
function INCLUSIVEMASK(E)

    maxIndicies ← an empty list
    for j in{0,1,...,L} do
        rowMaxima ← 0
        for i in{0,1,...,P_o} do
            if E_ij == 1 then
                rowMaxima ← j
        maxIndicies append rowMaxima
    E' ← An L × P_o matrix where E'_ij = 0
    for j in{0,1,...,L} do
        for i in{0,1,...,P_o} do
            if i ≤ maxIndicies[j] then
                E'_ij ← 1
    return E'
```

3.2.6 Categorising telomere read-pairs

Our definitive definition of a fully telomeric read is a read where 90% of the sequence is telomere or the read falls into the error profile (see Algorithm 3.4). In practice we observe

that using a threshold above 90% leads to decreased accuracy (see Section 3.4.2). It is possible that this is indicative of genuine telomere heterogeneity.

The next stage in the Telomerecat estimation process is the categorisation of read-pairs. Now that we have defined what constitutes a telomere read, we can categorise read-pairs into four distinct types as shown in Figure 3.4. Pseudo-code for read categorisation is given in Algorithm 3.4.

Read-pairs are categorised according to how many reads in the pair are deemed complete. Additionally, in the case of the boundary, the read-pair's directionality is also considered. In other words, we assess whether the complete read was identified in the $5'$ or $3'$ section of the read. The consideration of directionality and categorisation is crucial to Telomerecat's ability to filter for interstitial reads and account for aneuploidy.

Below is the list from Section 3.2.1, annotated by the read types from Figure 3.4.

- **F1**: Read pairs originating from the telomere
- **F2a**: Read pairs from boundary between the telomere and subtelomere
- **F2b+F4**: Subtelomeric and ITS read-pairs
- **F3**: Read pairs that contain occurrences of the hexamer by chance

We observe that read-pairs where one read in the pair is completely telomeric and the other is not, can come in two configurations. In F2 read-pairs, the complete read is comprised of the "CCCTAA". In F4 read-pairs the complete read is comprised of "TTAGGG".

This information is vitally important as the F4 configuration is only possible on the boundary of an interstitial or subtelomeric repeat stretch. As we saw in the Introduction, telomeres are appended to the chromosome in such a way that the CCCTAA sequence is always to the $5'$ of the DNA strand. Thus all reads from the genuine boundary between telomere and subtelomere should be in the F2 configuration.

We now see that within our classification of F2 reads we must distinguish between F2 reads that fall on the genuine telomere boundary and reads that fall on the boundary of ITSs. Accordingly, we specify that read-pairs on the genuine telomere boundary are F2a reads and read-pairs from ITS are referred to as F2b.

F2a and F2b read-pairs are identical. However, by observing the amount of F4 reads in a sample we are able to infer the amount of F2b. We assume that, on average, each ITS or subtelomere region that contributed an F4 would also contribute an F2b. Thus we are able to estimate F2a.

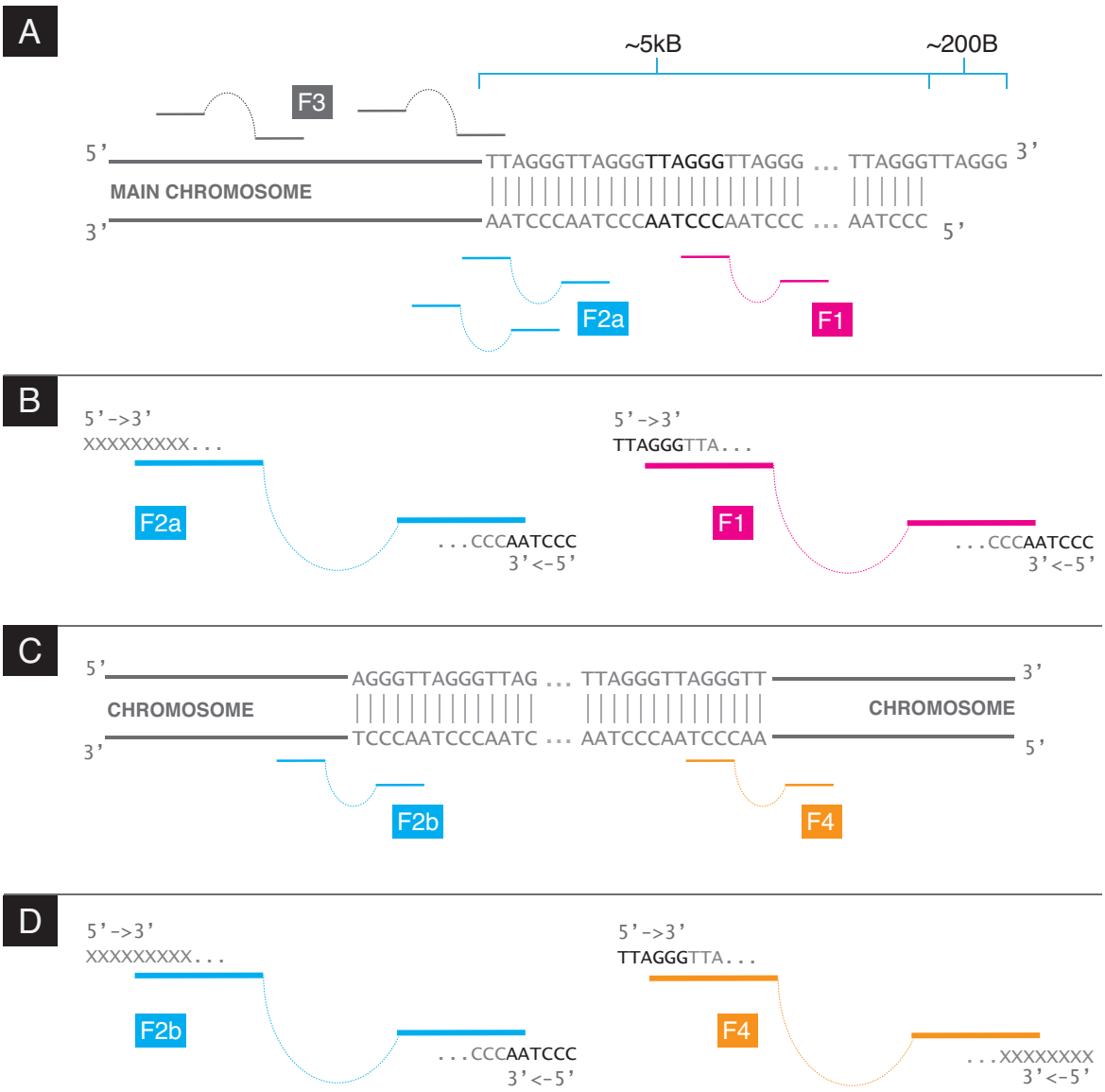


Fig. 3.4 ***(A)****: The read-pair types at the boundary between telomere and subtelomere. F2a reads stem from the boundary whereas F1 reads stem from anywhere within the telomere proper. F3 are reads where neither read in the pair is complete telomere* ***(B)****: Detail of the F1 and F2a read types. F1 read-pairs are comprised of two complete telomere reads. F2a read-pairs are comprised of a read-pair where one read is complete telomere and the other is not. Crucially, the complete telomere read is comprised of CCCTAA* ***(C)****: The read-pair types at an ITS.* ***(D)*** *Detail of the F2b and F4 read types. Note that the F2b is physically indistinguishable from an F2a read. An F4 read is a read-pair where one read is complete telomere and the other is not. The complete end is comprised of TTAGGG*

$$F2b \equiv F4$$

$$F2a = F2 - F2b$$

In this way, we are able to observe an estimate of reads over the actual telomere boundary. Therefore, we see that this process filters for interstitial reads and estimates coverage at the boundary of the telomere and subtelomere. The distorting effects of aneuploidy are removed, as at no point do we assume a fixed number of boundaries. We simply observe the amount of reads.

Along with F2a reads, telomere length is influenced by the amount of F1 reads in a sample. F1 reads come from the telomere proper and are defined as any read-pair where both reads in the pair are telomeric.

Once the reads have been categorised we can proceed to length estimation.

Algorithm 3.4 Sort read-pairs into the read types shown in Figure 3.4. We assume that the variables z, λ and L were calculated previously for each of the reads.

```
function GETREADTYPE(read1, read2)
    if ISTELOMERE(read1) and ISTELOMERE(read2) then
        ▷ Both reads in the pair are telomere
        return F1
    else if ISTELOMERE(read1) or ISTELOMERE(read2) then
        ▷ Exactly one of the reads in the pair is complete
        teloRead ← read1 if ISTELOMERE(read1) else read2
        if CCCTAA in teloRead.seq then
            return F2
        else
            return F4
    else
        ▷ Neither read is complete
        return F3

function ISTELOMERE(read)
    z ← z_read
    λ ← λ_read
    L ← L_read
    if z < (1/10) · L then
        return TRUE
    else if E_{λ,z} == 1 then
        return TRUE
    else
        return FALSE
```

3.2.7 Correcting for cohort-wide error

We observe that in some cases it is useful to normalise a cohort's $F2a$ count based on information from other samples in the batch. What follows is a method for adjusting F2a using a weighted average.

Let C be the total number of TELBAMs in a batch provided to Telomerecat. Such that subscript c represents a value relevant to any individual TELBAM. Let $\theta = \frac{F2a}{F2+F4}$ such that θ^{exp} is the average θ observed across all TELBAMs in a cohort and θ_c^{obs} is the observed value of θ with in a particular TELBAM.

$$\theta^{exp} = \frac{\sum_{c=1}^{C} \theta_c^{obs}}{C}$$
$$\theta_c^{cor} = \frac{\theta_c^{obs} \cdot \psi_c + \theta^{exp} \cdot w}{\psi_c \cdot w}$$

Where w is a predetermined weight of 3. ψ for any given TELBAM is obtained as follows:

$$\mu_c = \frac{\sum_{i=1}^{\frac{2}{5}p} \sum_{j=1}^{L} \mathbf{X}ij}{L \cdot (\frac{2}{5}p)}$$
$$\sigma_c = \frac{\sum_{i=1}^{\frac{2}{5}p} \sum_{j=1}^{L} (\mathbf{X}ij - \mu_c)^2}{L \cdot (\frac{2}{5}p)}$$
$$\psi_c = \frac{\sigma_c}{\mu_c}$$

So it follows that the adjusted value of F2a is given as $\theta^{cor} \cdot (F2 + F4)$

3.2.8 From telomere read counts to an estimation of length

The final step of the telomere length estimation process involves converting a ratio of $F1 : F2a$ read counts into an estimation of length. We achieve this by simulating telomere length under the observation of counts for F1, F2a and the fragment size. Pseudocode for the simulation is given in Algorithm 3.5

The fragment size distribution is acquired by observing the average fragment for all properly mapped pairs in the TELBAM where mapping quality is high. We observe that fragment size is normally distributed around a mean and varies considerably according to sequencing chemistry and platform. Figure 3.5 shows the distribution of fragment sizes in a single sample and also fragment sizes across the entire TwinsUK cohort (a validation cohort used later in this chapter).

Algorithm 3.5 Telomerecat length estimation simulation algorithm

function LENGTHESTIMATION(F1, $F2a$)
  $\tau \leftarrow$ Arbitrary starting TL
  $\mu, \sigma \leftarrow$ Sample fragment mean and standard deviation
  while $(F1' \neq F1)\&(F2a' \neq F2a)$ **do**
    $F1', F2a' \leftarrow simulate(\tau, F1 + F2a, \mu, \sigma)$
    if $F1' < F1$ **then**
      $\tau \leftarrow \tau + i$
    else if $F1' > F1$ **then**
      $\tau \leftarrow \tau - i$
  return τ

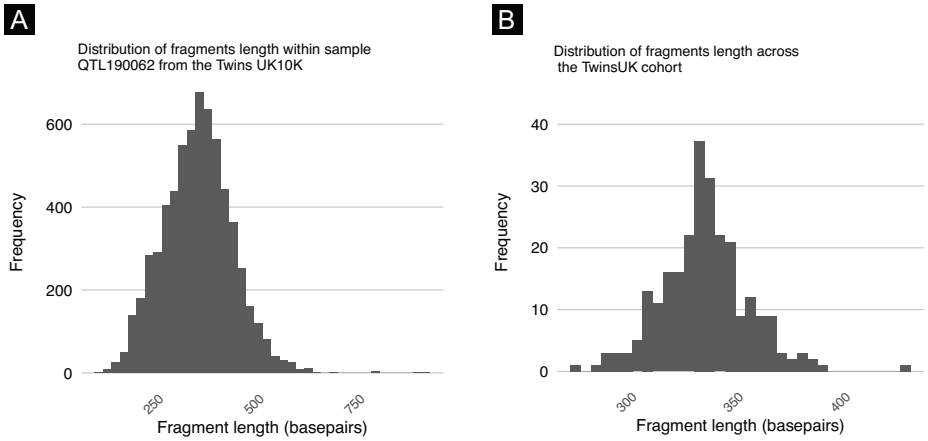


Fig. 3.5 Fragment sizes in the TwinsUK10K cohort. ***(A)****: Fragment size distribution within a sample from the TwinsUK10K cohort.* ***(B)****: The distribution of fragment sizes across the TwinsUK10K cohort*

3.3 Validation

In this section we detail a set of experiments made to assess Telomerecat's ability to estimate telomere length. We have applied the tool widely in order to compare it with existing methods, to examine whether it can identify expected telomere dynamics and to what extent it agrees across duplicated samples.

3.3.1 Testing Telomerecat on simulated data

To test the underlying principles of our method we applied Telomerecat to a set of simulated data. The simulation approach that we developed allowed us to generate reads from a sample with known telomere length. We could then compare the length estimates provided by Telomerecat with the underlying true telomere length. The simulation approach also allowed

us to test whether Telomerecat was influenced by the number of chromosomes in the sample and whether our method of interstitial filtering was well founded.

Pseudocode for the validation simulation is given in Algorithm 3.6. In essence, we use the ART simulator (HUANG *et al.*, 2012) to simulate a set of reads from a hypothetical reference genome comprised of telomere, subtelomere and random DNA sequence. A diagram and explanation for the method used to create the hypothetical sequence is shown in Figure 3.6. The set of parameters used for this investigation is shown in 3.1. For the purposes of this investigation we generated estimations for 5,000 samples with varying telomere length, subtelomere length, coverage and ploidy.

Algorithm 3.6 Estimate telomere length from an artificial DNA sequence. The parameters used for the investigation described in the text are given in Table 3.1. A diagram of the genome as generated by the GetGenome function is given in Figure 3.6

function VALIDATIONSIM(μ^t, σ^t, μ^s, σ^s, μ^c, σ^c, μ^d, σ^d)
- ▷ Draw from the normal distribution to decide the
- ▷ length of telomere (t), length of subtelomere (s),
- ▷ number of chromosomes (c) and depth of sequencing (d)
- $t \leftarrow \mathcal{N}(\mu^t, \sigma^t)$
- $s \leftarrow \mathcal{N}(\mu^s, \sigma^s)$
- $c \leftarrow round(\mathcal{N}(\mu^c, \sigma^c))$
- $d \leftarrow round(\mathcal{N}(\mu^d, \sigma^d))$

- ▷ Use number of t, s, and c to generate a random
- ▷ DNA sequence with interstitial telomere sequence
- $seq \leftarrow$ GETGENOME(t, s, c)

- ▷ Simulate reads from the hypothetical reference
- ▷ sequence using the ART simulator
- $bam \leftarrow$ SIMULATEREADS(seq, d)

- ▷ Estimate telomere length using Telomerecat
- $e \leftarrow$ ESTIMATETELOMERE(bam)

- ▷ Return the estimated and actual telomere length
- **return** e, t

The results of this investigation show strong agreement between estimated telomere length and the true underlying telomere length (Figure 3.7A). We also confirmed that estimated telomere length did not appear to be biased by the amount of chromosomes present in the simulated sequence (Figure 3.7B).

We were also reassured to find that extreme outliers were only seen in samples with low depth of sequencing and few chromosomes (Figure 3.7C).

The simulation approach detailed in this section is extremely limited in the extent to which it can demonstrate Telomerecat's ability to estimate telomere length accurately. It is not clear that our hypothetical genome model is a good likeness of a real chromosome. Particularly the interface between telomere and subtelomere. Furthermore, we suspect that the sequencing reads produced by the ART simulator contain less sequencing error in the telomere reads than actual sequencing experiments. As such, this does not represent a validation of Telomerecat's ability to differentiate telomere reads suffering sequencing error and subtelomere reads.

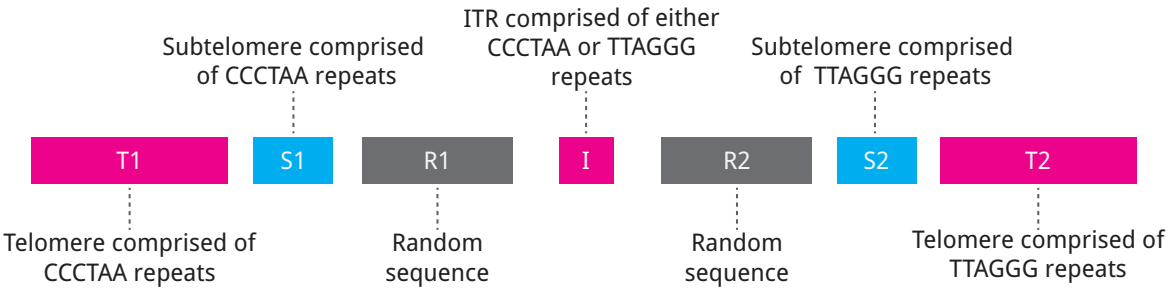


Fig. 3.6 A structural overview of the hypothetical sequence used for the validation simulation. ***T1 & T2:****representation of telomere. These sections are comprised of the canonic telomere sequence repeated.* ***S1 & S2:*** *representation of subtelomere. This is simply the telomere sequence where ĩ0% of bases were mutated at random. Accordingly it bears a strong resemblance to telomere.* ***R1 & R2:*** *random DNA sequence in which the letters T,A,C,G appear uniformly throughout.* ***ITS:*** *A stretch of telomere repeats. A coin is flipped to decide whether the sequence will appear as CCCTAA or TTAGGG in the reference. The lengths of the subtelomere and telomere sections of the reference are sampled from the normal distribution as per Algorithm 3.6. The R1 and R2 sections are always 1.5KB long and the ITS always constitutes 17 instances of the relevant canonic sequence (102bp)*

Despite the drawbacks of this approach, it is still a useful investigation. The results lead us to believe that our method for inferring telomere coverage by observing the boundary between telomere and subtelomere is sound and yields estimates unbiased by ploidy. We also see that the method identifies interstitial reads as per our expectation. It is clear that there is routinely a surplus of F2 reads in comparison to F4. The only reasonable explanation for this observation is that F2 and F4 reads are produced evenly at the site of the ITS, however, *only* F2 reads exist at the boundary between telomere and subtelomere.

This investigation shows us that our interpretation of telomere biology and how it is represented in WGS samples is sound. Using this high quality simulation data, Telomerecat is able to estimate telomere length to a high degree of accuracy.

Name	Value
μ^t	5
σ^t	2.5
μ^s	1.5
σ^s	1
μ^c	23
σ^c	10
μ^d	25
σ^d	20
Read length	100
Insert size	350

Table 3.1 Parameters used in the validation simulation investigation

3.3.2 Comparison to existing computational and experimental methods

To verify that Telomerecat is able to identify telomere length within WGS samples, we compared the algorithm to an established experimental method (mean terminal restriction fragment Southern blot experiment (mTRF)) and the existing computational method (TelSeq). Blood samples were taken from 260 adult females as part of the TwinsUK10K study, WGS and mTRF were conducted on each sample (described previously (VALDES *et al.*, 2005; MOAYYERI *et al.*, 2013)). The donor's age at sample collection is also recorded for each sample. Since absolute agreement is not expected, we consider correlations between the methods. The results of the comparisons are shown in Table 3.2 and in Figure 3.8.

We observe that the best correlation is between the two computational methods at $r = 0.631$. The next best correlation was between mTRF and Telomerecat. These results suggest both Telomerecat and TelSeq correlate well with mTRF indicating that both tools are providing realistic estimates of telomere length. Telomerecat fares slightly better but it is not clear that this is a significant difference or random chance. The extent that Telomerecat correlates with mTRF is in line with correlations previously observed between other experimental methods and mTRF (GUTIERREZ-RODRIGUES *et al.*, 2014).

qPCR measurements were also made for each of the samples in this cohort, however these data are not shown in Table 3.2 or Figure 3.8. We found that qPCR measurements to display a correlation with age that was significantly worse than other methods ($r = 0.09$). Furthermore qPCR showed only weak agreement with TelSeq, Telomerecat and mTRF. In all cases correlation was $r < 0.4$. This result concurs with other literature which indicates qPCR to have a lower fidelity than other commonly used telomere length estimation techniques

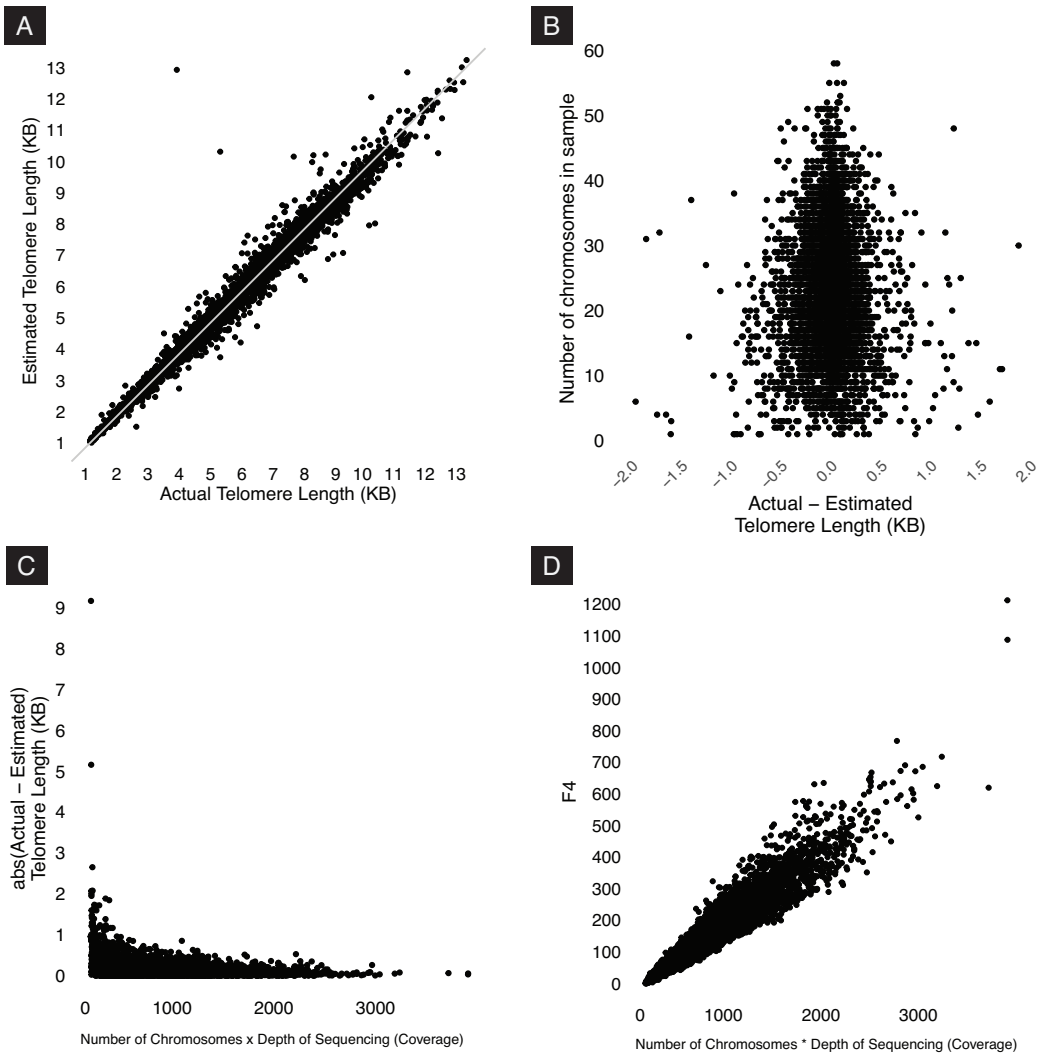


Fig. 3.7 Results of the validation simulation ***(A):*** *A plot of estimated vs actual telomere length. Light grey line is line of agreement. It appears that estimated and actual telomere length strongly agree* ***(B):*** *Difference between estimated and actual telomere length plotted by number of chromosomes in the sample. We see that the difference between estimates is centred around 0 regardless of chromosome count. Extreme outliers have been cropped from this plot.* ***(C):*** *Number of chromosomes multiplied by depth of sequencing (a proxy for reads in the sample) plotted against absolute difference between estimated and actual telomere length. We see that samples with the poorest estimation were produced by samples with low sequencing coverage and chromosome count* ***(D):*** *F4 vs Number of chromosomes multiplied by depth of sequencing*

(AUBERT *et al.*, 2012). Other studies have produced far better correlations between qPCR and mTRF (AVIV *et al.*, 2011), which could call into question the validity of both of the

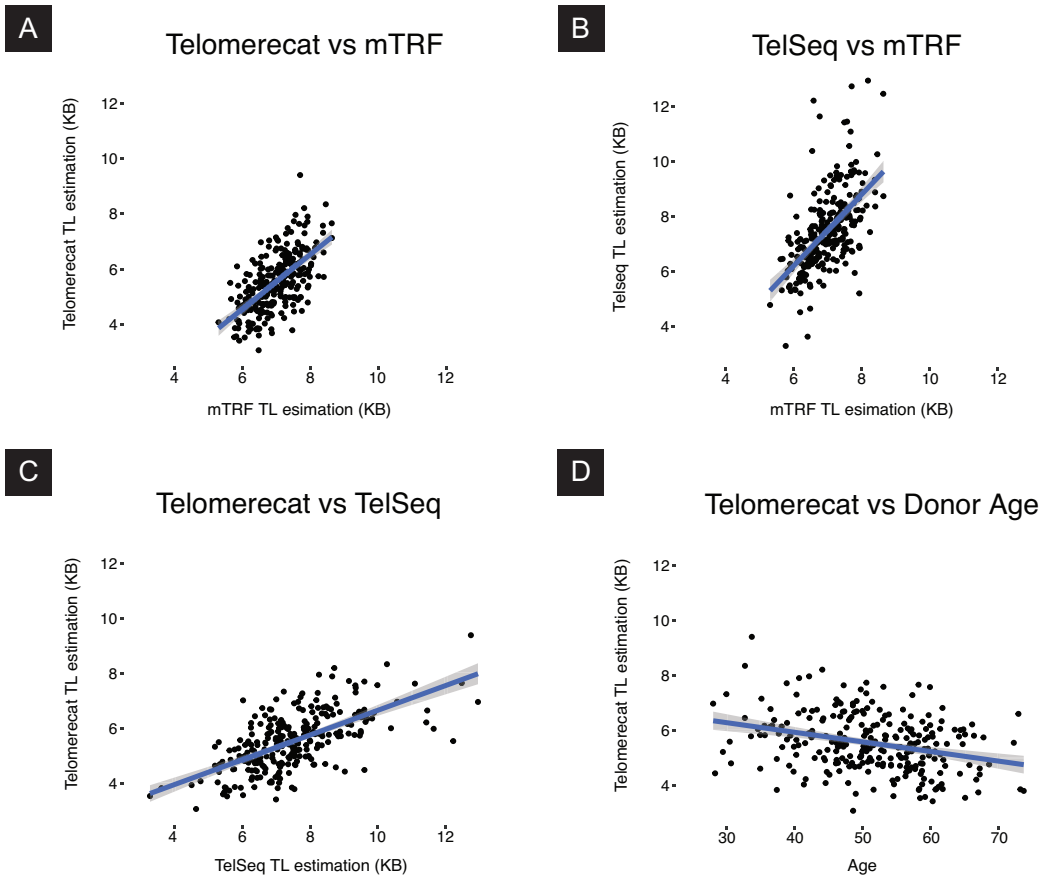


Fig. 3.8 Scatter plots describing the relationship between Telomerecat, mTRF, and TelSeq estimates of telomere length (TL)

measurements used in this analysis. Given that qPCR demonstrates a lower correlation with age in comparison to mTRF, we are more inclined to trust the mTRF estimates.

The results of this analysis show that Telomerecat correlates well with establish experimental methods, even in samples with relatively low coverage. Furthermore, we are reassured to see Telomerecat produced a correlation with age only slightly weaker than that of mTRF; a strong indicator that we are capturing genuine information about telomere lengths.

It is worth remarking that the sequencing depth of these samples is amongst the lowest of all the samples analysed as part of this thesis. Our simulation data, as well as those carried out by TelSeq's authors, shows that the accuracy of both methods improve as sequencing coverage is increased. It would be interesting to see to what extent the correlation between the Telomerecat, TelSeq and mTRF improved as the sequencing coverage was increased.

	Telomerecat	TelSeq	mTRF
TelSeq	$r = 0.631$	-	-
mTRF	$r = 0.618$	$r = 0.583$	-
Donor Age	$r = -0.306$	$r = -0.239$	$r = -0.321$

Table 3.2 Results for the comparisons between Telomerecat, TelSeq, mTRF and Donor Age. Pearson correlation was used for each comparison.

Telomerecat estimates telomere length that is shorter, on average, than TelSeq. At least part of this disparity may be due to Telomerecat's active filtering of reads from ITSs. Telomerecat finds that, on average, 7% of telomeric read-pairs identified are from ITSs.

3.3.3 Application to a longitudinal MSC data set

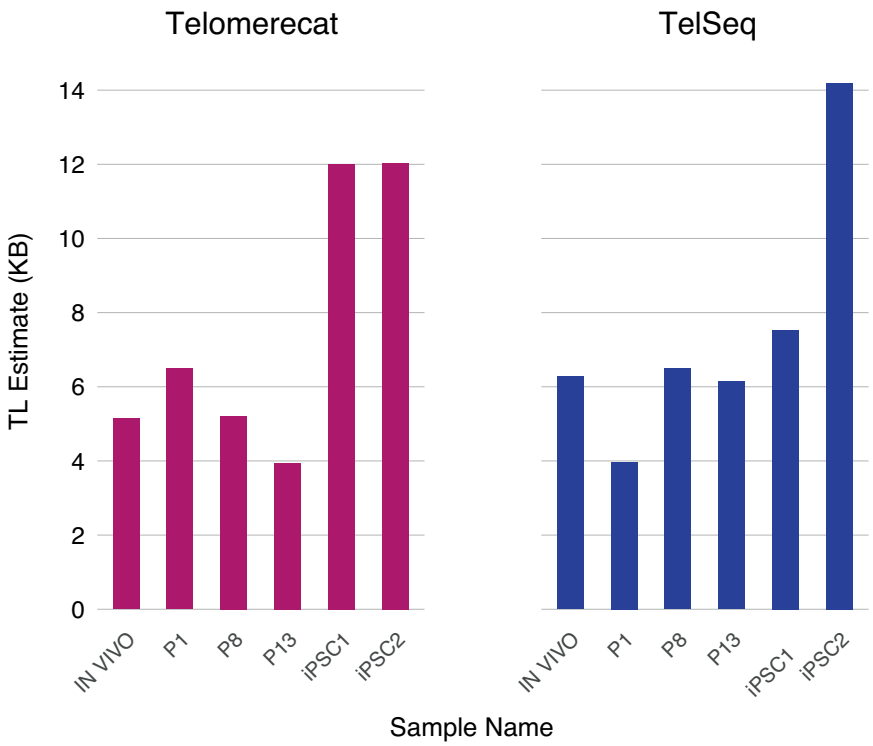


Fig. 3.9 Estimates for the MSC samples produced by Telomerecat (left) and TelSeq (right). We expect to see a decrease in telomere length with additional passaging (P1 to P13), but consistently high telomere lengths in the two iPSC samples (iPSC1 and iPSC2)

We applied Telomerecat to a set of WGS samples from a mesenchymal stem cell (MSC) experiment described previously (Cai *et al.*, 2014). Mesenchymal stem cells are multipotent

stromal cells commonly located in bone marrow (MINGUELL *et al.*, 2001). The experiment constituted six WGS samples: an in vivo MSC sample from a healthy 31 year old male, three passaged MSC samples (P1,P8 and P13) and two induced pluripotent stem cell (iPSC) samples.

MSCs are unusual amongst mature human stem cells as they do not express any measurable amount of telomerase (ZIMMERMANN *et al.*, 2003). Accordingly, telomere length attrition has been described in MSC passage experiments (SAMSONRAJ *et al.*, 2013). Conversely, iPSC cells have been shown to exhibit heightened telomerase expression (MARION *et al.*, 2009). We hypothesised that telomere length would shorten across the passaged MSC samples and lengthen within the iPSC samples.

We applied Telomerecat and TelSeq to the aforementioned MSC WGS data. The results are shown in Figure 3.9. Telomerecat identifies telomere shortening across the passaged samples, as expected. We see that Telomerecat estimates that between P1 and P13 the average telomere length was shortened by 2.5KB, at a rate of approximately 0.2KB per passage. Furthermore, we see that Telomerecat identifies long telomere length in the the two iPSC samples. We also note that TelSeq fails to identify the expected telomere dynamics.

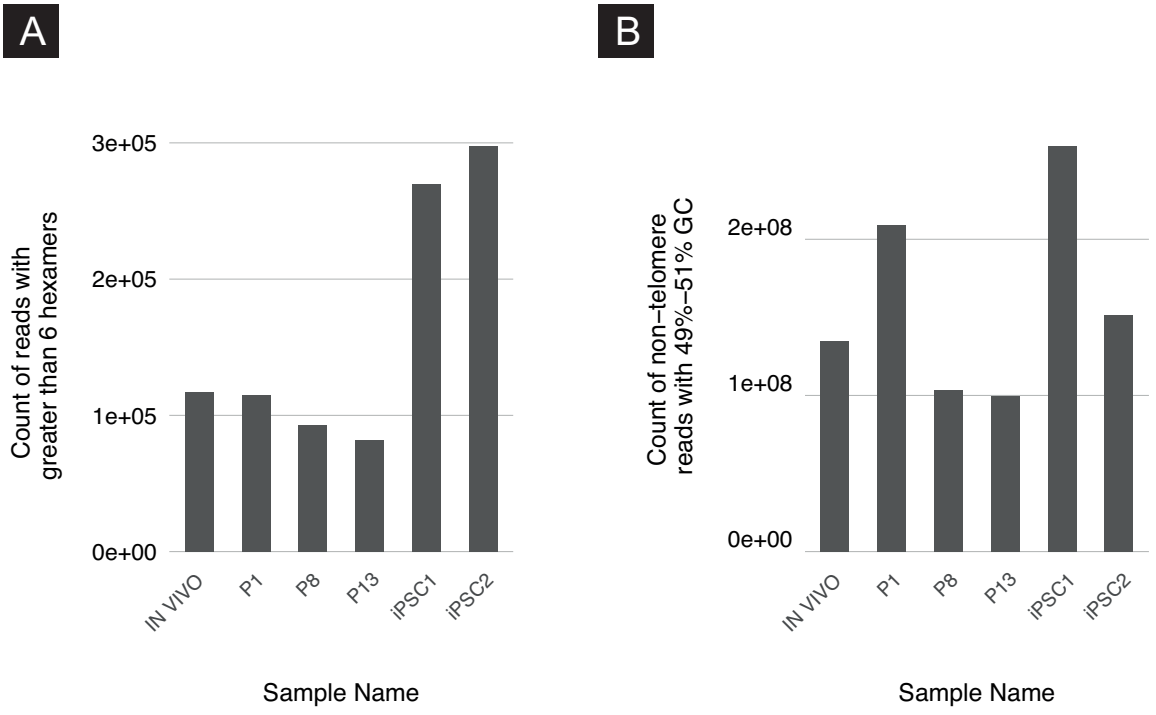


Fig. 3.10 ***(A)****: The amount of reads with 7 or more telomere hexamers for each sample.* ***(B)****: The amount of reads falling within the specified GC boundaries for each sample as defined by TelSeq*

The reason for TelSeq's failure to identify the correct dynamics seems to lie in a disconnect between the telomere content and coverage at GC locations. As we saw in the Introduction, TelSeq relies on estimating coverage at regions of the genome that display GC content of 48% to 52%. We see in Figure 3.10A that if one considers only the amount of telomere reads (as chosen by TelSeq) the correct dynamics are identified. However, in Figure 3.10B we see that the corresponding amount of reads at the same GC content differs greatly between samples. This disconnect between telomere reads and coverage of the telomeres seems to scupper TelSeq's ability to define the correct telomere dynamics.

3.3.4 Application to a set of mouse samples

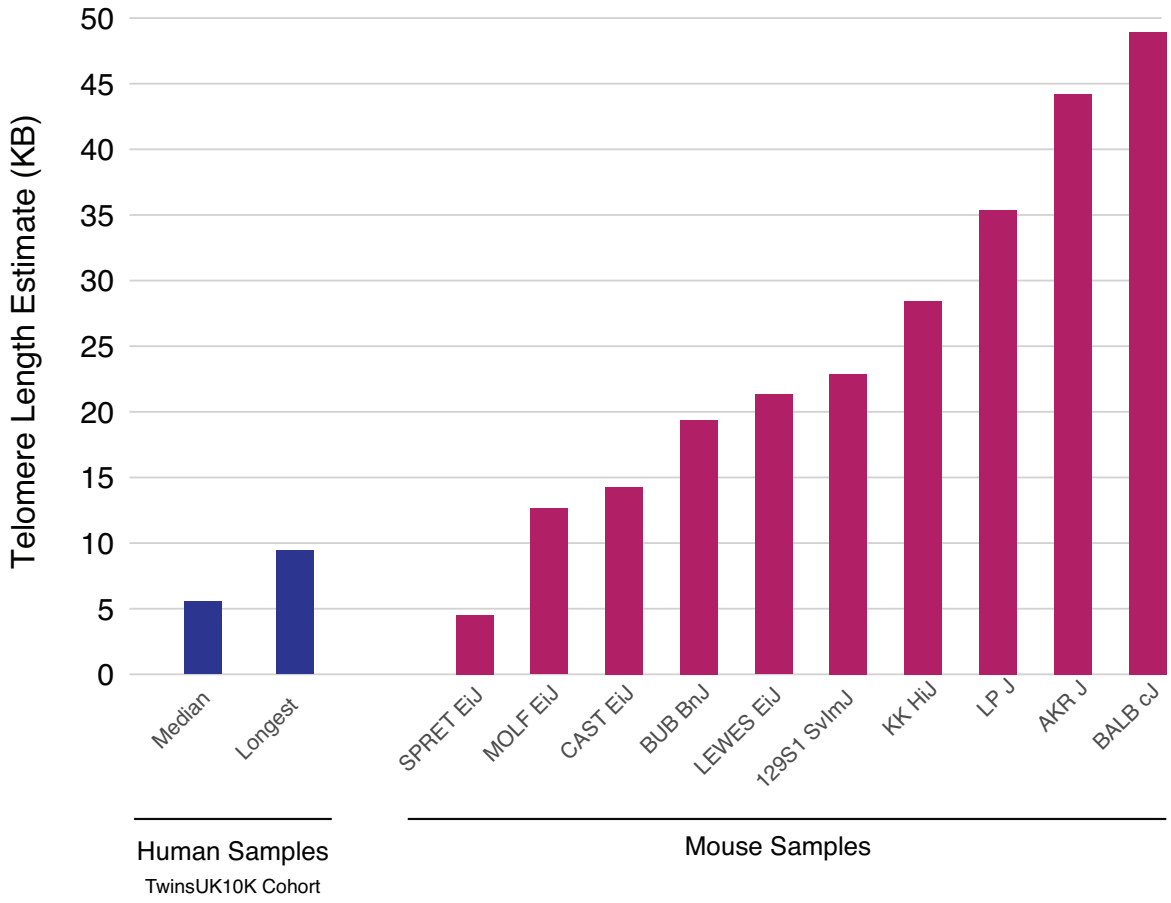


Fig. 3.11 Telomere length estimates by Telomerecat for 10 mouse samples from the Mouse Genomes Project

In general, mouse telomeres are known to be longer than human telomeres (KIPLING and COOKE, 1990), however, telomere length is known to vary across different mouse strains. We applied Telomerecat to 10 samples from the Mouse Genomes Project (KEANE *et al.*,

2011). Telomerecat identifies a range of telomere lengths, most of which are substantially greater than estimates from human samples. The estimates for the mouse samples, as well as two human samples for comparison, are shown in Figure 3.11. TelSeq was not applied as the tool is specifically tailored to the human genome.

Telomerecat identifies a range of telomere lengths for the mice, almost all of the lengths are substantially longer than the longest human telomeres in the TwinsUK10K cohort. Additionally, we note that two of the samples with the shortest estimates - CAST Eij and SPRET Eij - have been identified as having comparatively short telomeres (CALLICOTT and WOMACK, 2006; HEMANN and GREIDER, 2000; ZHU *et al.*, 1998). We also note that previous studies have identified the BALB cJ mouse strain as having long telomeres (ZHU *et al.*, 1998).

3.3.5 Application to a set of repeated measurements

We have also tested Telomerecat on pairs of WGS repeated measurements from the NIHR BioResource - Rare Diseases study. Telomerecat was applied to 93 samples of DNA extracted from whole blood. For each participant two samples were taken. Each sample was sequenced on either the HiSeq2000 or HiSeqX platform. We observe cases in this cohort where samples from the same participant were sequenced on the same technology and where samples were sequenced on different technologies. These blood samples from donor pairs were taken on separate occasions up to 3 years apart.

A sound approach to telomere length estimation will be reproducible across duplicate samples. After accounting for batch effects relating to choice of platform, Telomerecat achieves good agreement between the repeat measurements, as shown in Figure 3.12.

We observe that estimates from the two measurements show a Pearson correlation of $r = 0.8$. We see that in 80% of the duplicate pairs the difference in estimation is less than 1KB. Previous work suggests that the mTRF has a resolution of 1KB (although other methods have higher resolution) (AUBERT *et al.*, 2012). The fact that Telomerecat displays a similar accuracy on a set of repeat measurements is a reassuring sign, especially given that we expect a certain amount of technical noise and true biological difference between the telomere length of these biological duplicates.

In analysing this data set we noticed that samples sequenced on the Illumina HiSeqX platform display noticeably shorter telomere lengths. However, when comparing within duplicate pairs that were both sequenced on the XTEN platform the estimation correlates strongly. Furthermore we notice that HiSeqX samples routinely display lower values of the error measurement, Ψ. Figure 3.13 shows the difference in uncorrected mean telomere length

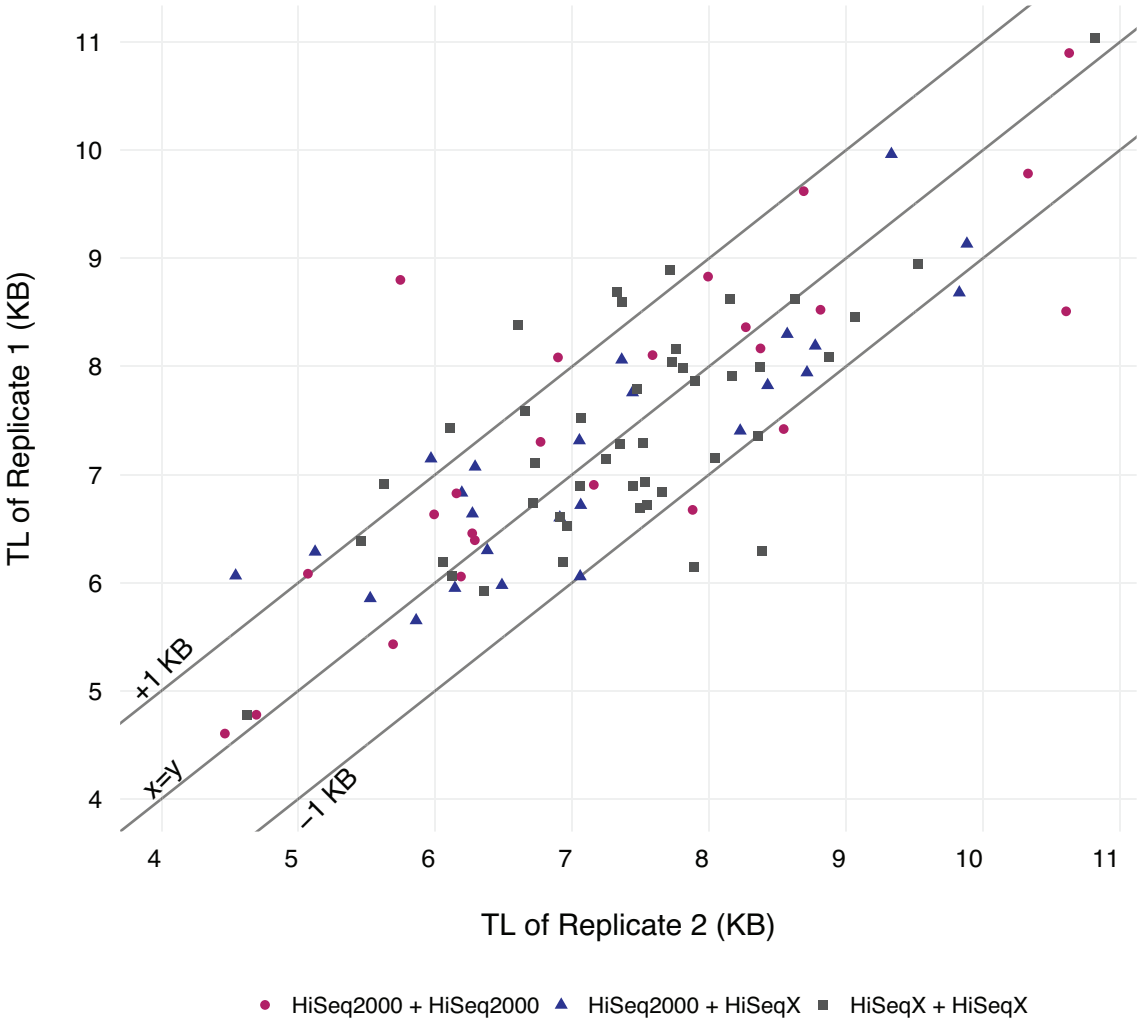


Fig. 3.12 A plot of telomere length (TL) estimates for repeated measurement pairs. Colours correspond to the sequencing platform of each sample in the pair

and Ψ. To account for the shorter estimates we mean corrected samples on the HiSeqX samples.

3.4 Discussion

In this chapter we have set forth our method for the estimation of telomere length from WGS data. We have carried out a number of validation experiments that seek to demonstrate Telomerecat's ability to accurately estimate telomere length in WGS data. In this section we shall discuss elements of the Telomerecat algorithm and their justifications.

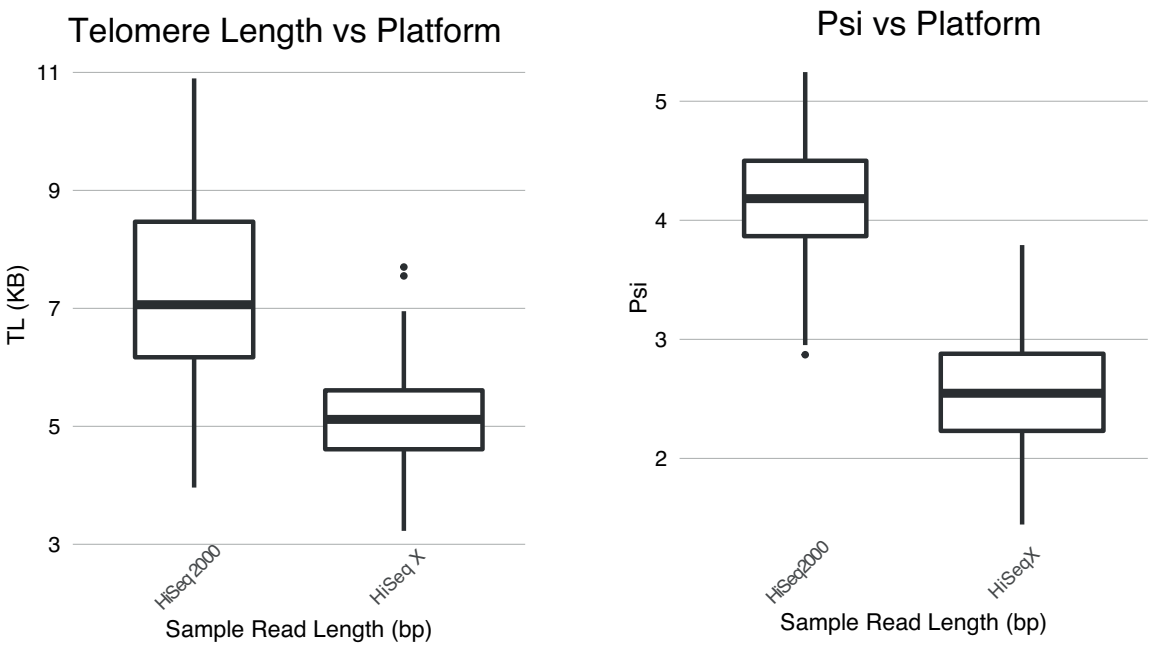


Fig. 3.13 Metrics from the blood technical replicate analysis

3.4.1 Comparing approaches for identifying telomere reads

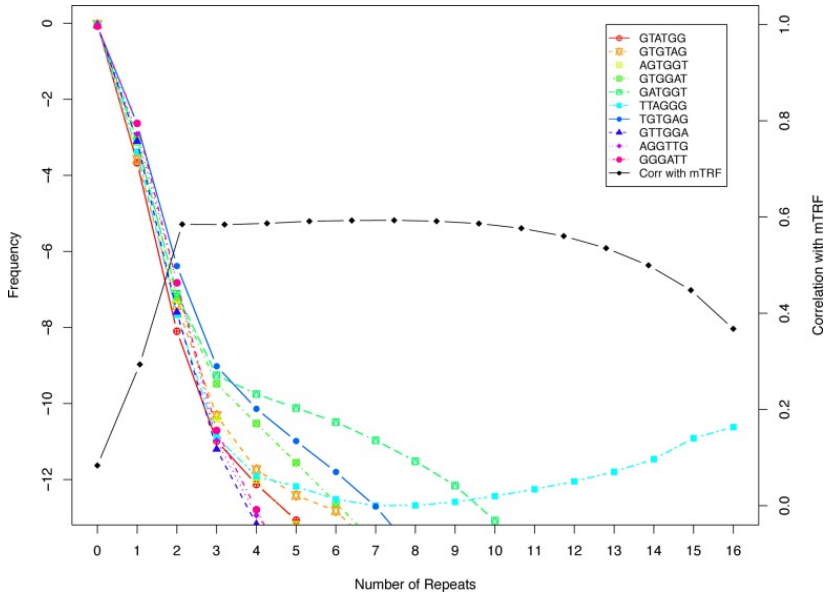


*Fig. 3.14 The frequency of various hexamers (coloured lines) and TelSeq's correlation with mTRF at various thresholds. Figure reproduced from (*DING et al.*, 2014) 2014*

As we saw in the introductory chapter of this thesis, numerous WGS telomere estimation approaches preceded Telomerecat. The first to attempt the estimation of telomeres from

WGS data was by CASTLE *et al.* (2010). CASTLE *et al.* took a sensible approach to defining telomere reads. Any read with four occurrences of the telomere hexamer was considered telomere. This approach was further refined by TelSeq; the first fully fledged WGS telomere length estimation tool.

Two of the methods, TelSeq and TelomereHunter, define telomere reads by the amount of occurrences of the telomere hexamers within a given read. In the case of TelSeq (DING *et al.*, 2014), the default threshold is seven occurrences. The reasoning for this cut-off was sound. DING *et al.* demonstrated that reads containing seven occurrences of other non-telomeric hexamers decreased in frequency. Conversely, the prevalence of reads containing the telomere hexamer increased after seven occurrences. This demonstrated that some other factor beyond random sequence arrangement was responsible for the amount of reads. This other factor is surely a measurable signal of telomere. The relationship between hexamer count and mTRF is shown in Figure 3.14 reproduced from the 2014 publication by DING *et al.*.

However, this threshold poses a contradiction. Telomere in humans is known to be an extremely homogeneous region of the genome. However, a threshold of seven hexamers allows reads where 58% of the sequence does not conform to telomere, if the read is 100 basepairs long. Furthermore, according to the study by DING *et al.*, concurrence with mTRF *decreases* as the threshold is made more stringent. How can we reconcile these observations?

We hypothesise that this relatively low threshold is a method of accounting for sequencing error, albeit an inaccurate one. By using the low threshold of $(TTAGGG)_7$, TelSeq ensures that reads suffering from sequencing error are included in the analysis. However, a price is paid: a substantial amount of false positives are introduced into the analysis. The extent of this problem is realised when we consider the nature of subtelomere. As we saw in the introduction, subtelomere is comprised of microsatellite repeats that bear striking resemblance to the canonic telomere repeat. It follows that the reason TelSeq fails to correlate well with mTRF when the threshold is too high is because genuine telomere reads are being excluded due to sequencing error.

Telomerecat's mismatching loci are associated with poorer Phred scores Telomerecat uses a more principled way to define Telomere reads. In Section 3.2.6 and 3.2.4 of this chapter we outlined our method for identifying a subset of reads that we believe to be true telomere reads suffering from error. This method assumes that sequencing error occurs in the telomere in such a way that the error is reflected in the associated Phred score for the mismatching base pair. We have several key pieces of evidence to back up this assertion.

Figure 3.15A shows a histogram of Phred scores for bases which do not match the sequence at their alignment position on the genome. This plot was created using every read

aligned to Chromosome 14 from a sample in the Prostate cancer cohort analysed in Chapter 4. In Figure 3.15B we see the distribution of Phred scores associated with mismatching loci in telomere reads as identified by Telomerecat. We see that the peak to the right of Figure 3.15B is drastically reduced, indicating that sequencing error accounts for a great deal of variation from expected sequence in telomere reads.

The plots shown in Figure 3.15 lead us to believe that deviation from the canonic Telomere sequence is rare and that where reads have few mismatches, they are caused by sequencing error.

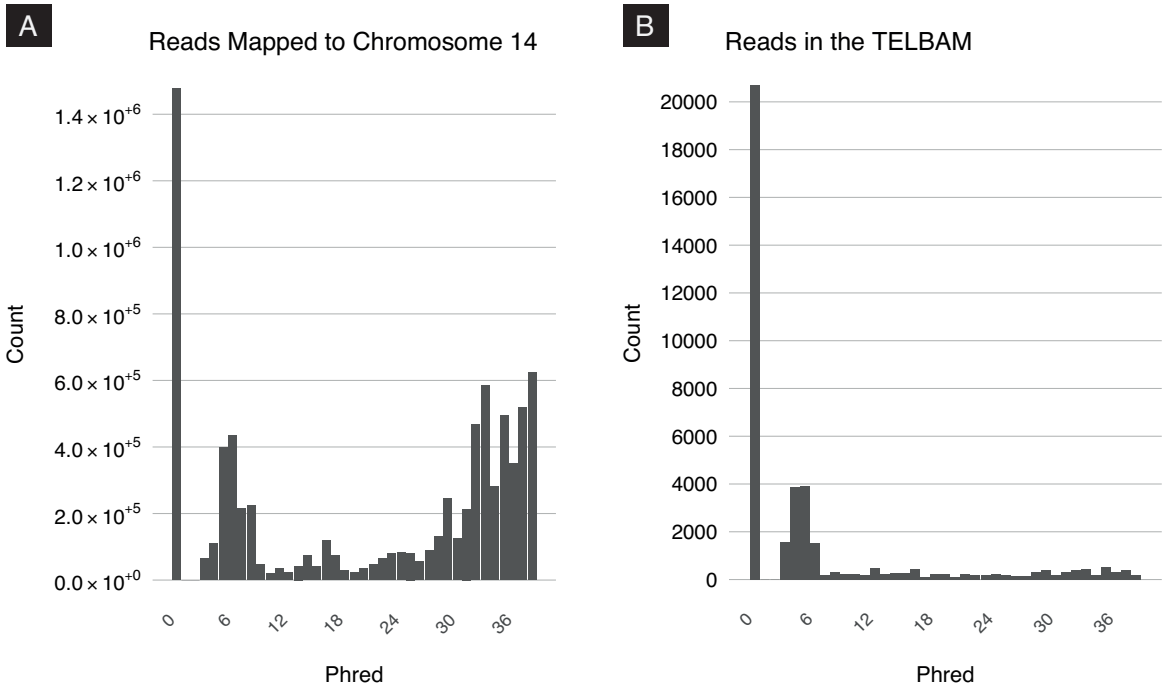


Fig. 3.15 Histograms recording the number of mismatching loci with the relevant Phred score. ***(A)****: Mismatching loci determined by MD tag for every read mapped to Chromosome 14 that had fewer than 4 mismatches. We see two peaks in the distribution. One representing sequencing error and the other genuine biological polymorphisms* ***(B)****: The same plot but for all reads in the corresponding TELBAM and where the mismatching loci were identified by Telomerecat. We see that mismatching loci are predominately associated with low Phred scores. There is no equivalent peak for genuine variation*

Evidence that our method identifies genuine sequencing error is from observation of Phred scores at mismatching loci, contrasted to Phred scores at random loci within the same read. We see in Figure 3.17 the difference in the Phred distribution at relevant loci. We see that the loci which our alignment method captures are associated with significantly poorer Phred scores.

Ambiguity in alignment of repetitive sequences

Early in the development of Telomerecat it became apparent that insertions and deletions pose a non trivial impediment to the understanding of Telomere sequence. The most natural way to quantify the difference between two character strings is by Hamming distance. Hamming distance is defined as the sum of mismatches between a string and some reference.

However, consider the example sequence shown in Figure 3.16 which uses the same sequence depicted in Figure 3.2. In Figure 3.16 we see that when we use a simple Hamming comparison, the number is skewed at the occurrence of the first "frameshift" (position 34).

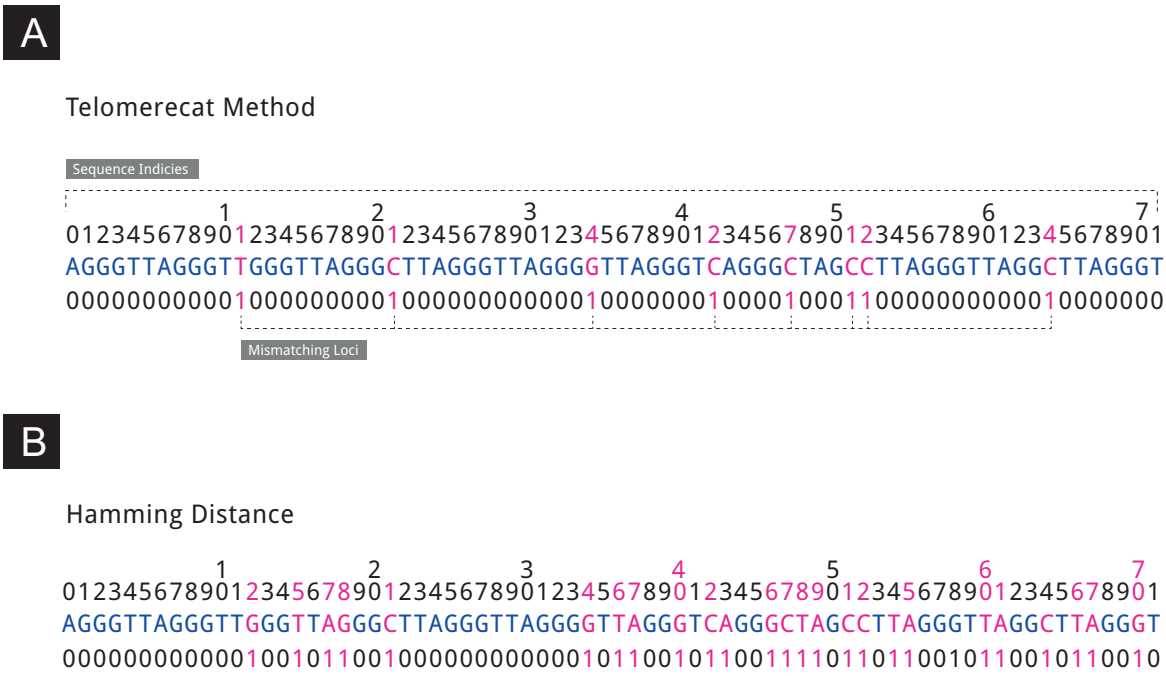


Fig. 3.16 ***(A)****: A diagram showing a set of mismatching loci according to Telomerecat. Telomerecat identifies 8 mismatching loci* ***(B)****: A plot showing a simple Hamming distance comparison on the same basepair sequence. The simple Hamming distance method finds 24 mismatching loci. A considerable difference. This difference is largely caused by the frameshift events starting at the 36th basepair*

TelSeq and TelomereHunter's comprehension of telomere reads on a hexamer counting basis is a simple way of avoiding this problem. The number of occurrences of the telomere hexamer is independent of the sequence in which they occur. However, as previously discussed, the hexamer counting comes with its own disadvantage.

Telomerecat attempts to identify these "frameshift" events so that a truer representation of a reads telomere sequence is ascertained. However, there is a certain ambiguity in interpreting these results. For instance, in Figure 3.16 at locus 34 in the sequence read, it is unclear

whether the resulting frameshift is at the behest of a deletion of “TTAG” or the insertion of an extra “G”.

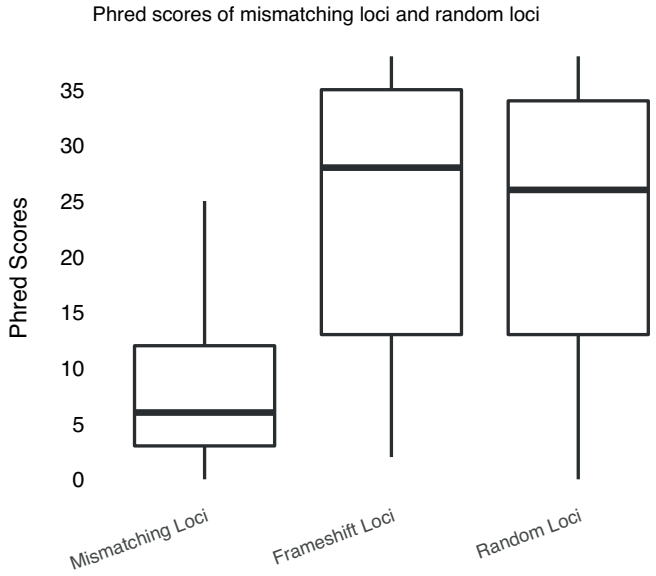


Fig. 3.17 Boxplots comparing Phred scores associated with mismatching, frameshift and random loci for every read in a TELBAM that had fewer than 20 mismatching and frameshift loci combined

Telomerecat attempts to resolve this ambiguity by taking a sample of basepairs directly adjacent to the site of a frameshift. This process is shown and described on the left hand side of Figure 3.2(3). In Figure 3.17 we compare the observed Phred scores at these frameshift loci with random loci in the read.

Figure 3.17 shows that frameshifts share the same distribution of associated Phred scores as random loci in the read. This observation could mean one of two things: either telomere frameshifts as a result of sequencing error are not represented by Phred scores or frameshifts are genuine biological features. In either case, it is important to account for frameshifts in the alignment method due to the distorting nature that their presence has on understanding the telomere read.

3.4.2 Genuine telomere heterogeneity and cohort-wide correction

During the development of Telomerecat, assumptions that underlied the method were discovered to be unfounded. In order to provide the accurate estimates of telomere length presented in the Section 3.3 we had to adapt the algorithm.

The first assumption that proved unfounded was that telomeres are entirely homogeneous. Early investigations into the content of telomere reads identified a preponderance of reads that were comprised of a pristine telomere sequence (i.e every basepair matched the expected telomere sequence). An area plot of all the reads within a TELBAM and relevant numbers of mismatching loci for a TwinsUK10k sample is plotted in Figure 3.18. On observing the association of poor Phred scores at mismatching loci, as explored in Figure 3.15, we proceeded under the assumption that the reads in the highlighted area in Figure 3.18 were actually complete telomere reads suffering from telomere error. However, when we came to apply Telomerecat to the TwinsUK10K sequencing data, it was clear that a conception of completely canonic telomere was too stringent.

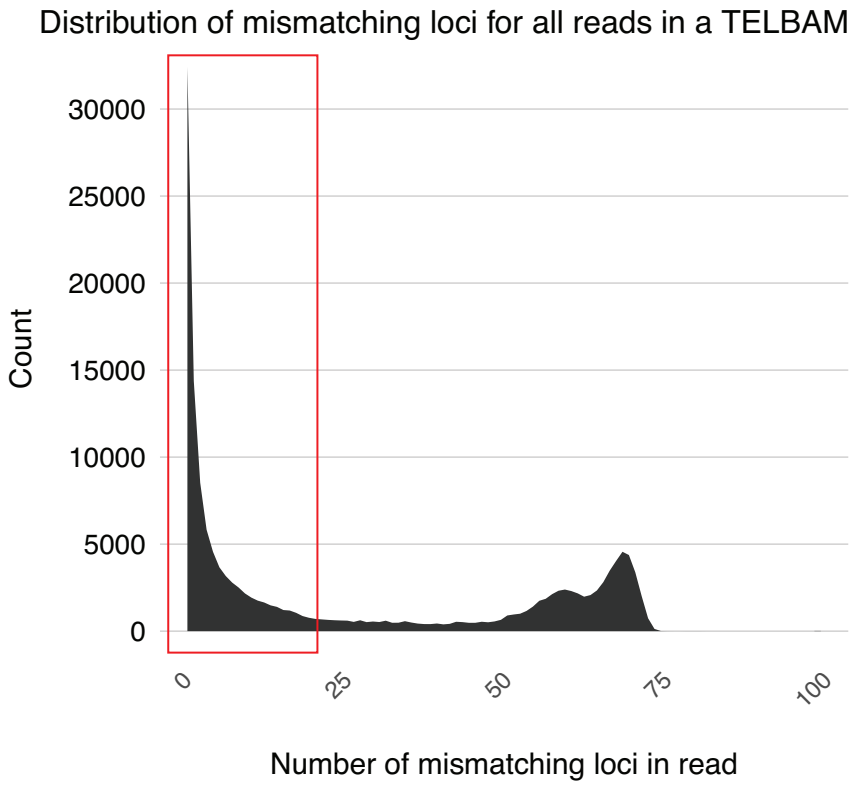


Fig. 3.18 An area plot of the distribution of the number mismatching loci within a single TELBAM from the Prostate cancer cohort explored in Chapter 4. The reads within the segment highlighted by the red rectangle were originally believed to display increased mismatching loci counts purely because of sequencing error

When we run Telomerecat on the TwinsUK10K cohort without allowing for any genuine mismatches in the telomere sequence (i.e a threshold of 100% in Algorithm 3.4) 61% of the samples fail due to a failure to identify telomere length. The failure is caused by extremely

unstable F2a counts, presumably because regions of the telomere proper are now strewn with boundaries as a result of the stringency.

The second assumption was that sequencing depth in relatively low coverage or poor quality data would be sufficient for us to measure F2a accurately.

We are certain of the validity of our logic in using the directionality of reads to filter for interstitial boundaries. Accordingly, the vast majority of samples present counts where F2 is greater than F4. This is no more apparent than in the Rare Blood Disease cohort (explored in Chapter 5) where more than 9,000 samples all report positive F2a counts. We believe that this is on account of the good quality and high depth nature of this data set. Furthermore, our tests with simulated data in Section 3.3.1 assure us that our logic surrounding the orientation of reads in sound.

To overcome this issue we designed the cohort correction method detailed in 3.2.7. This method uses the mean observed F2a count for the whole population to correct outlying values. As values of θ seem to vary across sequencing platforms it makes sense to only use the cohort correction "within batch".

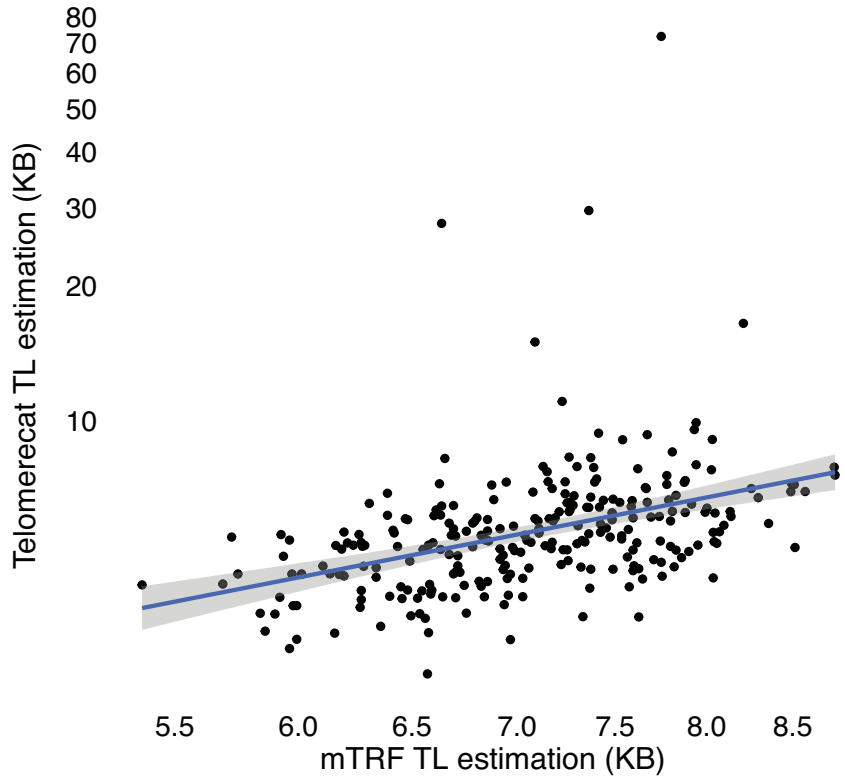


Fig. 3.19 The relationship between Telomerecat and mTRF without cohort correction. Notice that the Y-axis is scaled to log10

Figure 3.19 shows what happens when Telomerecat is run on the TwinsUK10K cohort without cohort correction. Extreme outliers (driven by very small F2a counts) cause the Pearson's correlation between Telomerecat and mTRF to fall to $r = 0.19$. On higher coverage data sets cohort correction is not needed. This is demonstrated by Telomerecat's good performance on HiSeq2000 repeated measurements samples where no cohort-wide correction is required and good agreement between duplicates is still observed.

3.4.3 Advantages of the error profile

The error profile method discussed in Section 3.2.5 is the way in which Telomerecat selects reads which are deemed to be suffering from sequencing error. This method has proven to be superior to all others in that it is extremely adaptable to different Phred and error distributions.

Figure 3.20 shows the distributions of different Phred scores to mismatching loci count from various samples. Clearly the shape of each of the 2D distribution differs greatly. The advantage to the error profile method is that it can fit the requisite shape to all of these distributions without the need for parametrisation. This increases the applicability of the Telomerecat method.

Early versions of Telomerecat used bowtie2 (a popular read aligner) to identify sequencing error in telomere reads. We found that when aligning telomere reads with bowtie2 different samples, with differing Phred score distributions, were treated differently. Furthermore we found it difficult to optimise the parameters of bowtie2 to work on different sequencing chemistries. The error profile method detailed here was found to be more adaptable. This is demonstrated by the way the error profile handles the different 2D distributions shown in Figure 3.20.

One way in which we experimented with the error profile method was in the use of more traditional noise reduction techniques such as those defined by mathematical morphology. However, we find that the methods shown in Algorithm 3.2 have the advantage of not requiring a "structuring element" as is the case for mathematical morphology methods. Choosing a structuring element for the three different distributions shown in Figure 3.20 would be a non-trivial problem that is avoided by using our error-profile method.

3.4.4 Other applications of Telomerecat

Aside from the investigations detailed in the subsequent chapters of this thesis, Telomerecat has been applied as part of the following studies:

- Genomic Evolution of Breast Cancer Metastasis and Relapse (YATES *et al.*, 2017)

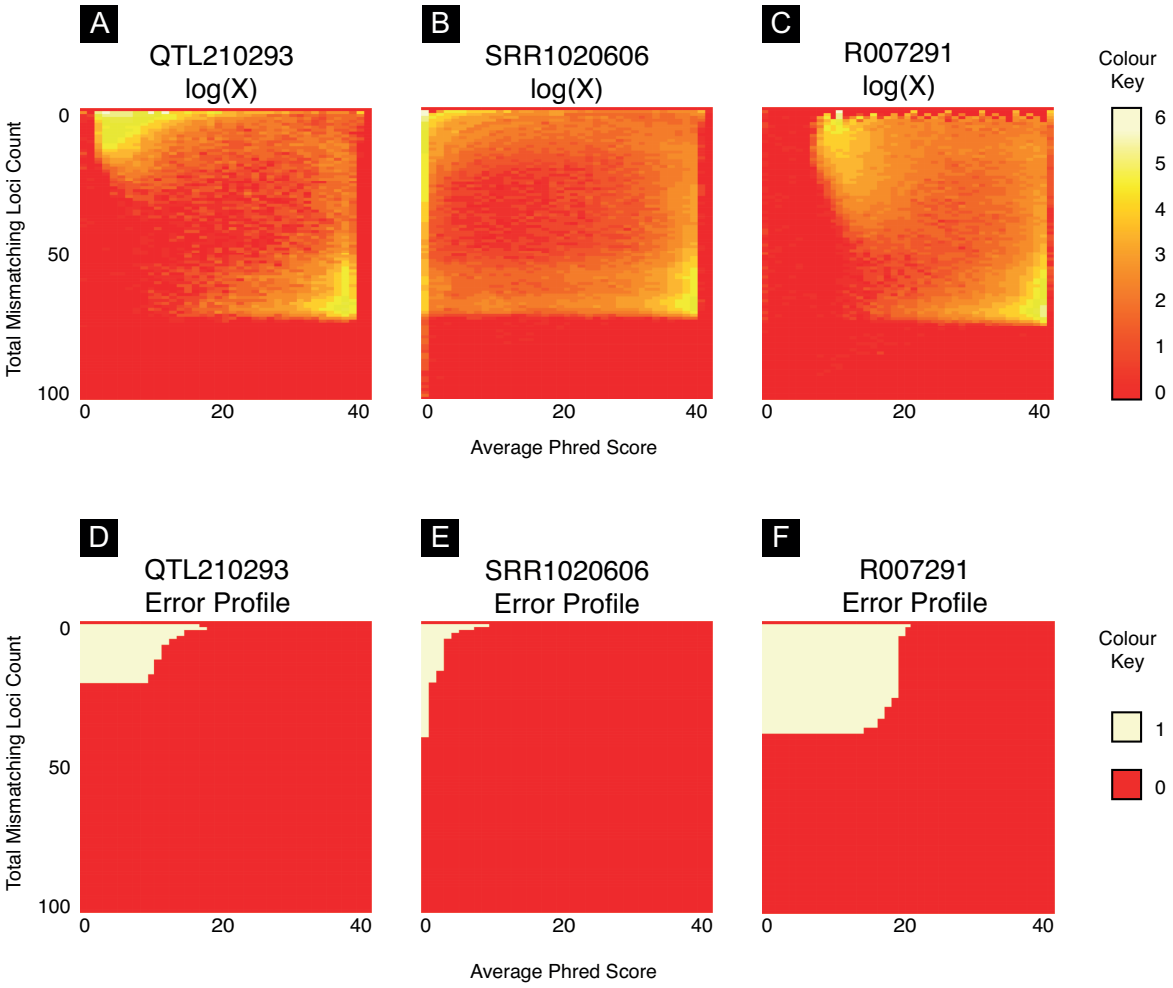


Fig. 3.20 Plots highlighting the different 2D distribution across samples from different sequencing experiments and their resultant error profile ***Top Row****: Heatmaps showing the 2D distribution of mismatching loci and average Phred scores for all reads in a TELBAM.* ***Bottom Row****: Binary heatmaps of the resultant error profiles. QTL210293 is a sample from the TwinsUK10K cohort. SRR1020606 is a sample from the MSC analysis. R007291 is a sample from the blood duplicates analysis*

- Copy-number signatures and mutational processes in ovarian carcinoma (MACINTYRE *et al.*, 2017)

Not included above is a study linking a previously undiscovered variant in the 5 prime UTR region upstream of the TERT gene to telomere length in kidney cancer. This study is in currently under passing through the final round of peer review.

3.5 Summary

Here we have presented our method, Telomerecat, and provided evidence that it can correctly estimate telomere length in WGS samples.

We believe that Telomerecat is an adaptable method. Firstly, because it is not constrained by the number of chromosomes in a sample and secondly, because it has been shown to work on a variety of different sequencing platforms and chemistries. This ability to work on numerous platforms comes from the flexibility of the novel error profile method to morph to differing distributions and the accuracy that using boundary coverage as a proxy for telomere provides.

The ITS filtering method developed as part of Telomerecat is also a breakthrough. Current attempts to filter for ITS simply subtract reads that map to gene-rich or non-telomere genomic regions. But mapping of repetitive reads is notoriously biased. We feel that our method for filtering that draws on knowledge of how telomerase appends telomere repeats, is superior in not relying on the mapping coordinates of reads. This further enhances the applicability of the method.

Chapter 4

Applying Telomerecat to a cohort of Prostate cancer samples

All the WGS samples analysed as part of this chapter were collected, sequenced and pre-processed by the ICGC Prostate working group. SNV data and associated clinical data were also collected, processed and normalised by the ICGC Prostate working group. RNA microarray expression data were collected and pre-processed by the CamCaP Study Group.

In the previous chapter we detailed our method for the estimation of telomere length from WGS, Telomerecat. With the method validated and capable of producing accurate estimates of telomere length our attention turned to an application to a set of cancer WGS samples.

In this chapter we detail the results of our application of Telomerecat to the ICGC UK Prostate Adenocarcinoma WGS cohort, funded by Cancer Research UK. We applied Telomerecat to 192 primary tumour samples all from separate patients. For each tumour sample we also estimated telomere length for a matched blood normal sample. Here we investigate the relationship of telomere length to various clinical and biological factors prevalent in Prostate cancer (PCa).

We also conduct an analysis on a separate set of samples from three donors expressing multi-focal disease, described previously by COOPER *et al.* (2015). Collectively these donors are referred to as the "Complex Men". Our analysis highlights the heterogeneity of telomere length across separate cancerous tumours from the same prostate.

4.1 Background

PCa is a disease predominantly found in older men; 77% of new diagnoses are in men over 60 and only one tenth of a percent are in men under forty (LI *et al.*, 2012). Exact age of onset is difficult to determine due to the symptomless nature of early disease. However, prospective studies on men who died from non-PCa related causes suggest that 44% of men between 71-80 have some measurable form of PCa (ZLOTTA *et al.*, 2013). The same study identified that of men over the age of 81, 66% displayed PCa. This speaks of the prevalence of the disease and also the role of age as a risk factor.

Ethnicity has also been proposed as a risk factor, with black men experiencing an increased risk in developing prostate cancer at some point in their lives, with some studies suggesting as much as a three-fold increase in risk (CHINEGWUNDOH *et al.*, 2006). However, it is not a settled matter as to the extent to which socio-economic factors influence underlying differences in biology between ethnicities (KHEIRANDISH and CHINEGWUNDOH, 2011).

Despite being described as "a disease of very rare occurrence" in the 1853 edition of the Lancet (ADAMS, 1853) PCa is now the second most common cancer amongst men worldwide and the single most prevalent cancer in developed countries (JEMAL *et al.*, 2011). The introduction of the Prostate Specific Antigen test (PSA) led to an increase of diagnoses from the mid 1980s which have led to controversies surrounding over-diagnosis of the disease (GREENLEE *et al.*, 2000; CENTER *et al.*, 2012). Recent estimates suggest that 50% of cases will be described as "low-risk" (KLOTZ and EMBERTON, 2014). However, this is not to say that prostate cancer is a harmless condition. It is the fifth leading cause of all cancer mortality worldwide (JEMAL *et al.*, 2011) and is listed as the cause of death for 258,000 men worldwide, each year (CENTER *et al.*, 2012).

Classifying prostate cancer PCas are typically classified by two separate clinical measures, Gleason scores and tumour-metastasis-node (TMN) grading. Gleason scoring was developed specifically for PCa in the 1960s and focuses on the histological progress of the disease (HUMPHREY, 2004). Low Gleason scores indicate a carcinoma with compact and well differentiated glands while higher scores indicate poorly differentiated cells with little coherent glandular architecture. Gleason scores are comprised of two measurements which detail the two most prevalent areas of the prostate. For instance, if a majority of the prostate is graded as 3 and a smaller portion as 4, then the resultant Gleason score is given as: "3+4". A lower score denotes more differentiated and thus less severe PCa.

TMN is a combined metric separately detailing the status of the tumour, node and metastasis (SCHRODER *et al.*, 1992; CHENG *et al.*, 2012). In contrast to Gleason scoring, TNM is used to classify a multitude of cancers. The tumour is graded from 1-3 based on its

size and the extent to the tumours invasion into surrounding structures (precise staging for the Prostate is given in Table 4.3). The metastasis and node sections of the metric define whether the disease has spread to the relevant structures.

While histological categorisations have existed for decades, more recent studies have attempted to stratify prostate cancers by molecular subtypes using genomic data. The first of these investigations by TOMLINS *et al.* (2005) identified a fusion event between the genes TMPRSS2 and ERG (referred to as TMPRSS2-ERG). A further study went on to identify this marker as present in roughly 40-50% of all PCas (TOMLINS *et al.*, 2009). According to YU *et al.* (2010) TMPRSS2-ERG fusions are thought to have a complex relationship with one of the key proteins in prostate cancer, the androgen receptor (AR).

In healthy prostate tissue, AR's primary function is to serve as a transcription factor activated by a collection of hormones known as androgens of which testosterone is perhaps the most well known example (HEEMERS and TINDALL, 2007). However, AR malfunction is thought to be a root cause in many cases of PCa. Indeed most PCa cases respond to androgen ablation therapy, demonstrating the dependence of the cancers on AR (FELDMAN and FELDMAN, 2001). Furthermore, PCa is rare amongst eunuchs, who display almost non-existent AR activity on account of their exceedingly low testosterone production (STOCKING *et al.*, 2016).

TMPRSS2-ERG fusions have been shown to allow androgens to bring ERG under AR regulation by fusion with the androgen regulated TMPRSS2 (TOMLINS *et al.*, 2005). Recent work has uncovered more complexity in the relationship between TMPRSS2-ERG fusions and the AR genes. Notably YU *et al.* (2010) who suggest that TMPRSS2-ERG fusion may have a role in disrupting AR's normal function in prostate tissue differentiation, thus providing a platform for oncogenesis.

Other subtypes defined by somatic mutations in the SPOP and FOXA1 genes have also been identified in PCa. In their 2012 study BARBIERI *et al.* identified SPOP and FOXA1 as frequently occurring mutations in PCa. BARBIERI *et al.* (2012) suggested that tumours with an SPOP mutation might herald a new prostate cancer subtype due to SPOP mutations being mutually exclusive with TMPRSS2-ERG fusions.

Recently, ABESHOUSE *et al.* (2015) conducted a comprehensive multi-omic analysis in order to more fully define the extent of these molecular subtypes. Their analysis revealed 7 differentiated subtypes, including subtypes characterised by previously identified TMPRSS2-ERG fusion, SPOP and FOXA1 mutations. ABESHOUSE *et al.* (2015) analysed a cohort of 333 PCa samples, 46% of which were classified owing to an occurrence of the TMPRSS2-ERG fusion, making this the largest subset. Samples with SPOP mutations made up 11% of the samples, samples with FOXA1 mutations contributed 3% whilst 1% of samples

were categorised according to an IDH1 mutation. The remaining samples were either not categorised or were defined by fusions or over-expression in the ETV1, ETV4 or FL1 genes.

Telomeres in prostate cancer There are currently no large scale multi-omic cohort studies pertaining to the variation of telomere length in PCa. However, telomere length has been touted as a possible prognostic marker in PCa (HEAPHY *et al.*, 2013) and reduced telomere content in PCa samples has been associated with increased likelihood of recurrence after radical prostatectomy (FORDYCE *et al.*, 2005). Further to the finding that shortened telomeres were predictive of poorer prognosis, a study showed that "comprehensive lifestyle interventions" could be used to elongate telomere in PCa (ORNISH *et al.*, 2013). Interestingly, although telomere length has been identified as a risk factor for other cancers, it seems there is no association between telomere length variation and increased PCa risk (MIRABELLO *et al.*, 2009; WENTZENSEN *et al.*, 2011; HURWITZ *et al.*, 2014).

Telomerase expression has received considerable attention with PCa. As with many cancers, PCa has been shown to display elevated telomerase expression in comparison to adjacent normal tissue (SOMMERFELD *et al.*, 1996; PFITZENMAIER *et al.*, 2006). Furthermore, at least two studies show an association between heightened telomerase expression and more advanced forms of PCa (LIN *et al.*, 1997; DE KOK *et al.*, 2002). In contrast LATIL *et al.* (2000) did not observe an association between telomerase expression and tumour stage or Gleason grade. Although LATIL *et al.* do report an association between elevated telomerase and MYC (a known oncogene) within prostate cancer. DING *et al.* (2012) identified the expression of telomerase as leading to the formation of more aggressive and invasive PCa tumours in a mouse model missing Pten and p53 and thus predisposed to PCa.

ICGC project and the data The data used in the study detailed in this chapter was collected and preprocessed by the International Cancer Genome Consortium (ICGC, 2017). The following investigation was undertaken on WGS data taken from 192 primary PCa tumours each with a matched blood sample. The tumours were extracted via radical prostatectomy between November 2010 and January 2011 as described previously in (WARREN *et al.*, 2013). Once extracted the samples were processed and sequenced using the HiSeq Illumina 2000 platform. Three different chemistries were used for sequencing (HiSeq4; TruSeqv3; TruSeqv3 Q-bin) across 6 sequencing batches. The mean age of donor at tumour collection was 61, the youngest donor was 41 and the oldest was 88.

4.2 Data Normalisation

Before investigating telomere length in PCa we wished to ensure that the data was unbiased and yielding reasonable estimates of telomere length.

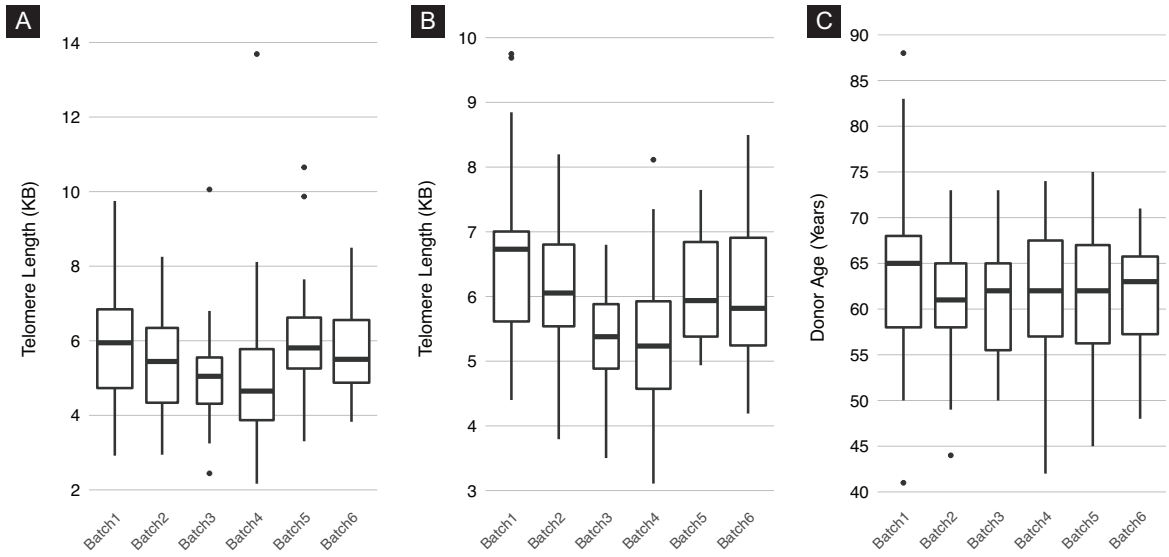


Fig. 4.1 Box-plots of prostate sample estimates pre-normalisation ***(A)****: All samples* ***(B)****: Blood samples only* ***(C)****: Donor age by batch*

We observed a noticeable difference in the telomere lengths across the 6 batches of sequencing. Figure 4.1A shows the difference in telomere across batches for tumour and blood samples combined. Figure 4.1B shows the difference when only blood samples are plotted. We sought to remove this batch effect before proceeding.

As part of our normalisation we also considered the differing distributions of age amongst the sequencing batches (Figure 4.1C). At least some of the variation of length estimates across batches is due to this genuine discrepancy in the distribution of age. We resolved that our method for assessing the effect of different batches would need to be independent of this age effect.

To these ends, we used a linear model to account for the difference caused by batch effects whilst not introducing more bias by removing genuine disparity caused by the differing age distribution.

The following approach uses only the blood normal samples to estimate the effect on telomere length bestowed by batch effects. Using only blood samples to detect batch effects means that changes on telomere length due to the effects of cancer are excluded from the normalisation.

Table 4.1 The coefficient as output by the normalisation linear model

	Coefficient
batch_2	-838.07
batch_3	-1409.87
batch_4	-1576.59
batch_5	-932.55
batch_6	-880.95
age	-41.34

We defined the linear model as

$$\begin{aligned} \text{length} \sim \beta_0 &+ \beta_1 \text{batch}_2 \\ &+ \beta_2 \text{batch}_3 \\ &+ \beta_3 \text{batch}_4 \\ &+ \beta_4 \text{batch}_5 \\ &+ \beta_5 \text{batch}_6 + \beta_6 \text{age} + \varepsilon \end{aligned} \tag{4.1}$$

Where, β are regression coefficients, batch_n represents a dummy variable denoting the batch a sample was sequenced in and age is the age of the donor at tumour collection. For each batch the relevant coefficient was subtracted from all of the observed telomere length within the batch. Batch coefficients are given in Table 4.1.

Some indication that this approach increases the accuracy of the estimation is that correlation with telomere length and age in blood samples increases after normalisation. This correlation with age is also a good indication that Telomerecat is identifying telomere length in the cohort. Before normalisation for batch effects, blood samples were Pearson's correlated $\rho = 0.18$ with age. After normalisation the correlation increased to $\rho = 0.26$ (Figure 4.3A). We also confirm that a correlation with age and blood telomere length was present within batches when measured separately (Figure 4.2).

Figure 4.4 shows telomere length by batch once normalisation has been applied.

All further investigations in this Chapter were conducted using telomere lengths corrected with the aforementioned normalisation technique. We proceeded with estimates for 192 matched tumour and blood normal pairs.

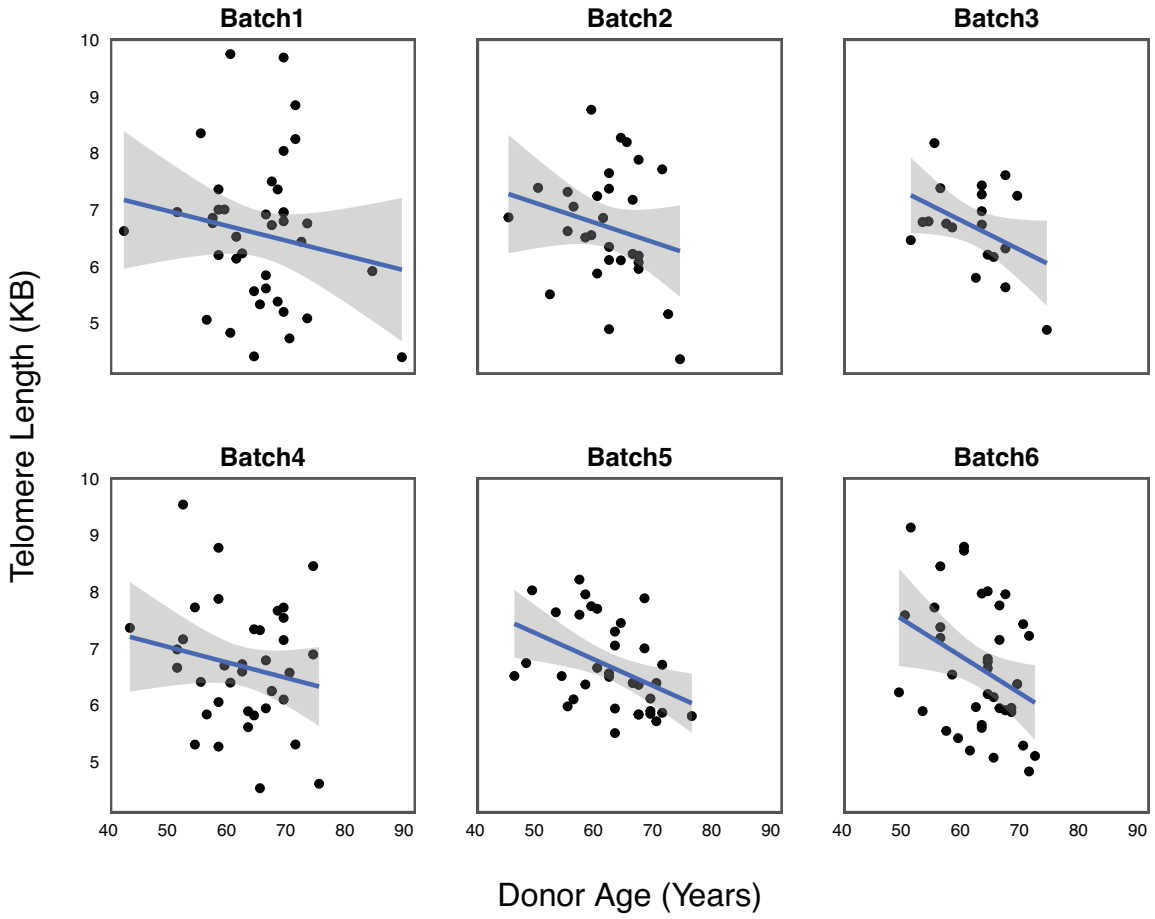


Fig. 4.2 A plot showing telomere against age for each sequencing batch. We see that each batch displays a negative correlation with age. All lines of best fit produced using a linear model.

4.3 Results

Once we had normalised the data so as to minimise bias owing to batch effects, we proceeded to analyse the data for associations with telomere length.

4.3.1 Sample type

Prostate cancer tumours have been shown to display shorter telomere length than matched normal samples (SOMMERFELD *et al.*, 1996; BARTHEL *et al.*, 2017). We observe this trend in our cohort. A two-sided Student T Test finds matched blood sample telomere lengths are significantly longer than tumour sample telomere lengths ($p < 0.001$, Figure 4.5A).

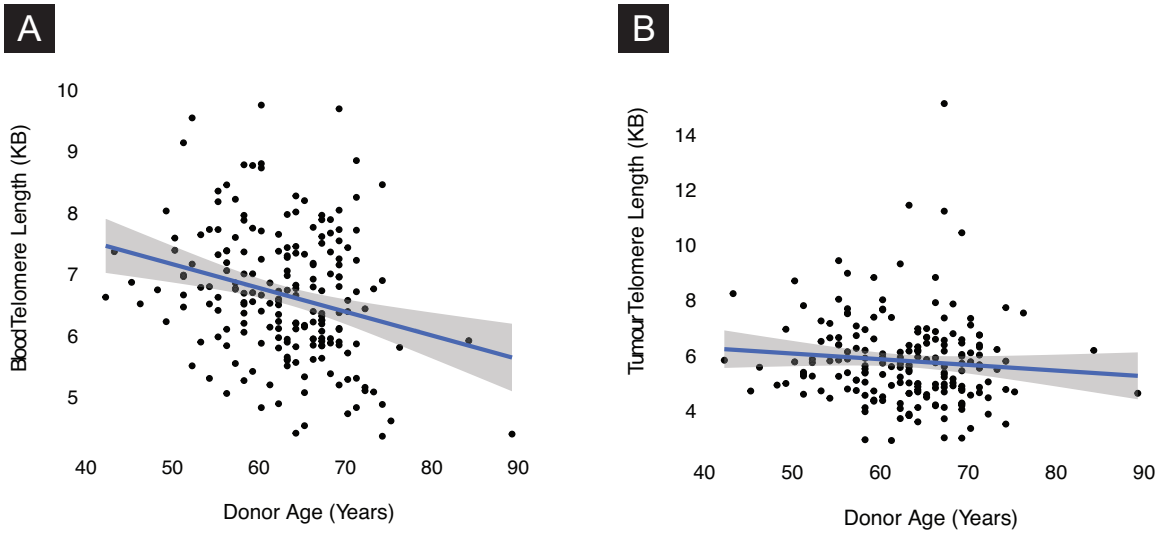


*Fig. 4.3 Telomere plotted against length for in the prostate cohort **(A)**: Normal samples, line of best fit according to a linear model **(B)**: Tumour samples, line of best fit according to a linear model*

We observe that while telomere length in blood samples is correlated with age, the same correlation is weaker in prostate tumour samples. A Pearson's correlation of $\rho = -0.09$ is observed (see Figure 4.3B). When we fit a linear model to the relationship with age and tumour telomere length, we find that the relationship is not deemed significant ($p = 0.2$).

Furthermore, we see that matched blood and tumour sample telomere length do not correlate with each other, a linear model does not find the relationship significant (Figure 4.5B). This is further evidence of tumour specific telomere shortening in prostate cancer.

Poor correlations indicative of tumour related telomere stress These observations are indicative of the active stress to telomere length within the tumour environment. The fact that tumour telomere length does not correlate with age indicates that tumour telomeres have been affected by cancer. These pressures might take the form of either shortening as a result of increased cell proliferation or lengthening in order to forgo senescence.

We have also seen how blood and tumour telomere lengths are not significantly correlated. This could indicate further evidence of stressors acting upon tumour telomere length. A study by DANIALI *et al.* (2013) suggests that we should expect telomere length of blood and other tissues from the same donor to be well correlated, however, the study does not make a direct comparison between prostate and blood.

Eighteen of the samples within this cohort do have associated tissue normal samples. Using this data we can shed some light on the strength of assocation between blood and

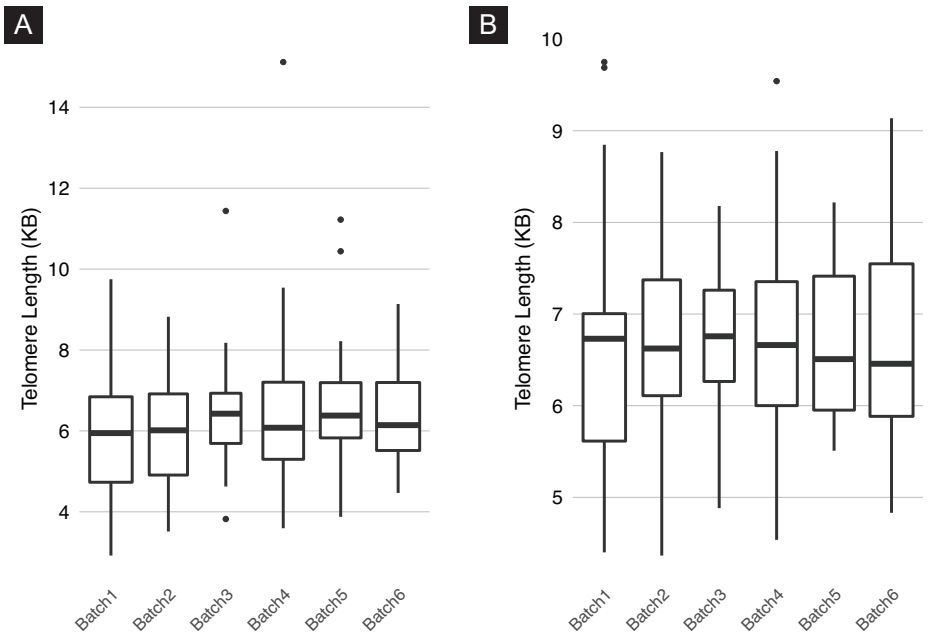


Fig. 4.4 Box-plots of prostate sample estimates post-normalisation ***(A)****: All samples* ***(B)****: Blood samples only*

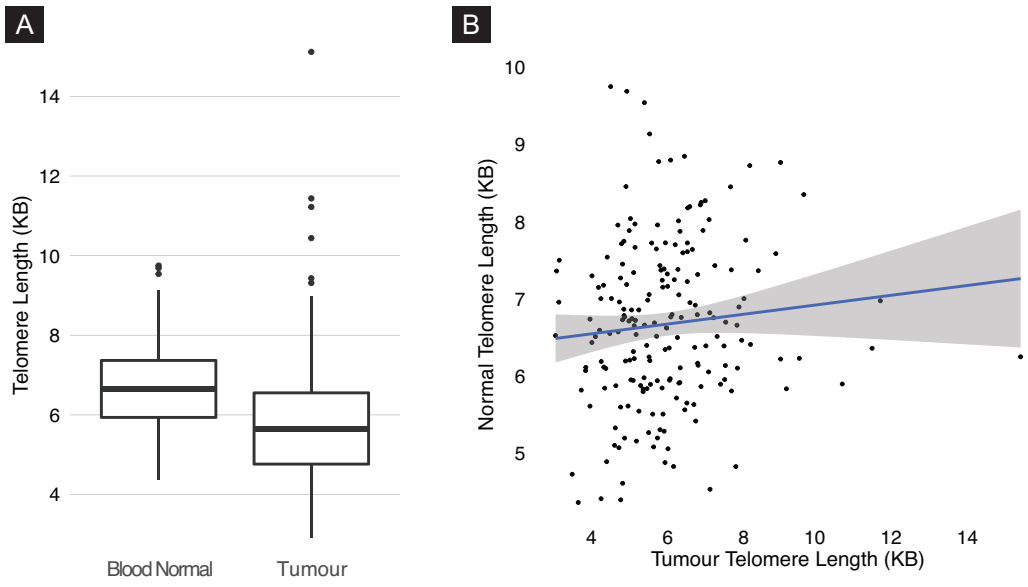


Fig. 4.5 Plots detailing difference telomere length between sample types ***(A)****: Boxplot of telomere length by sample type* ***(B)****: Normal telomere length vs tumour telomere length. Line of best fit given by a linear model*

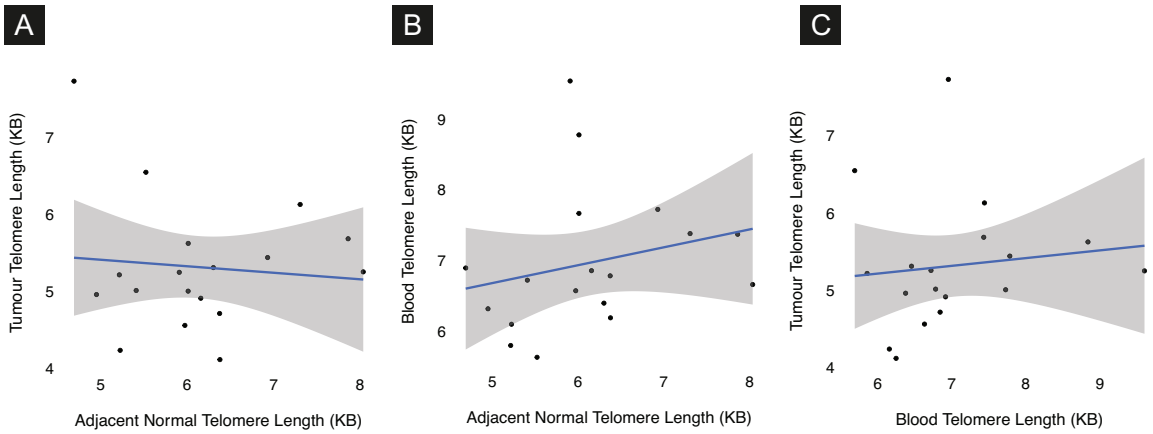


Fig. 4.6 Plots showing the relationship between adjacent normal, blood and tumour telomere lengths for 18 samples where adjacent normal samples exist. ***(A)****: Adjacent normal versus tumour* ***(B)****: Adjacent normal versus blood* ***(C)****: Blood versus tumour*

normal prostate tissue. Figure 4.6 shows the relationship of normal, blood and tumour telomere length across these eighteen samples.

None of these relationships is found to be significant under a linear model. However we note that with so few samples we are possibly under powered in our search. However, blood and adjacent normal show the strongest positive correlation (Pearson's $r = 0.23$). Similarly to the relationship between blood and tumour telomere length, there appears to be little correlation between adjacent normal and tumour telomere length. The relationship to age is maintained across the adjacent normal telomere lengths, with a comparatively high Pearson's correlation of $r = -0.51$.

The results from these 18 samples are not conclusive, but there is still the possibility that we should expect an association between prostate and blood telomere length. This being so, the fact that blood does not correlate with tumour could be further evidence for the effects of cancer on the tumour telomere lengths. This hypothesis is given further credence by the lack of positive correlation between adjacent normal and tumour telomere lengths seen in Figure 4.6A.

Together these results indicate that telomere length in PCa tumours are altered by the effects and stressors associated with cancer.

4.3.2 Tumour stage and disease progression

Next we examined the relationship between Telomere length and the stage of the disease. As we saw in the introduction to this thesis, prostate cancer can be graded by the TNM

Table 4.2 Results for a Tukey's range test applied to the ANOVA model described in Section 4.3.2. Upper and lower estimates describe range of 95% confidence. The p-value for the model as a whole is $4.5 \cdot 10^{-5}$

	p-value	*lower (KB)*	*upper (KB)*
T1 - T2	0.022	0.0745	1.211
T1 - T3	0.00005	0.712	2.373
T2 - T3	0.03	0.052	1.74

staging classification system. The "T" initial in the TNM initialism stands for tumour and accordingly describes the stages of tumour progression. The various stages of the prostate tumour progression are given in Table 4.3.

*Table 4.3 Tumour staging as defined by AJCC (*CHENG et al.*, 2012)*

T1	Clinically inapparent tumour neither palpable nor visible by imaging	
	a	Tumour incidental histological finding in $\leq$5% of tissue resected
	b	Tumour incidental histological finding in >5% of tissue resected
	c	Tumour identified by needle biopsy (e.g. because of elevated PSA)
T2	Tumour confined within prostate	
	a	Tumour involves $\leq$one-half of one lobe
	b	Tumour involves >one-half of one lobe but not both lobes
	c	Tumour involves both lobes
T3	Tumour extends through the prostate capsule	
	a	Extracapsular extension (unilateral or bilateral)
	b	Tumour invades seminal vesicle(s)
T4	Tumour is fixed or invades adjacent structures other than seminal vesicles such as external sphincter, rectum, bladder, levator muscles, and/or pelvic wall	

We tested the relationship between the tumour grade and telomere length. When we plot telomere length by the associated tumour stage there appears to be some relationship. Figure 4.7A shows a boxplot of telomere length by category and Figure 4.7B shows a boxplot using the more granular subcategory definitions. We also observed that donor age and blood telomere length showed no association with tumour stage (see Figures 4.7C & 4.7D).

To test this relationship formally, we describe a linear model

$$\text{length} = \beta_0 + \beta_1 \text{stage}_{\text{T2}} + \beta_2 \text{stage}_{\text{T3}} + \varepsilon \tag{4.2}$$

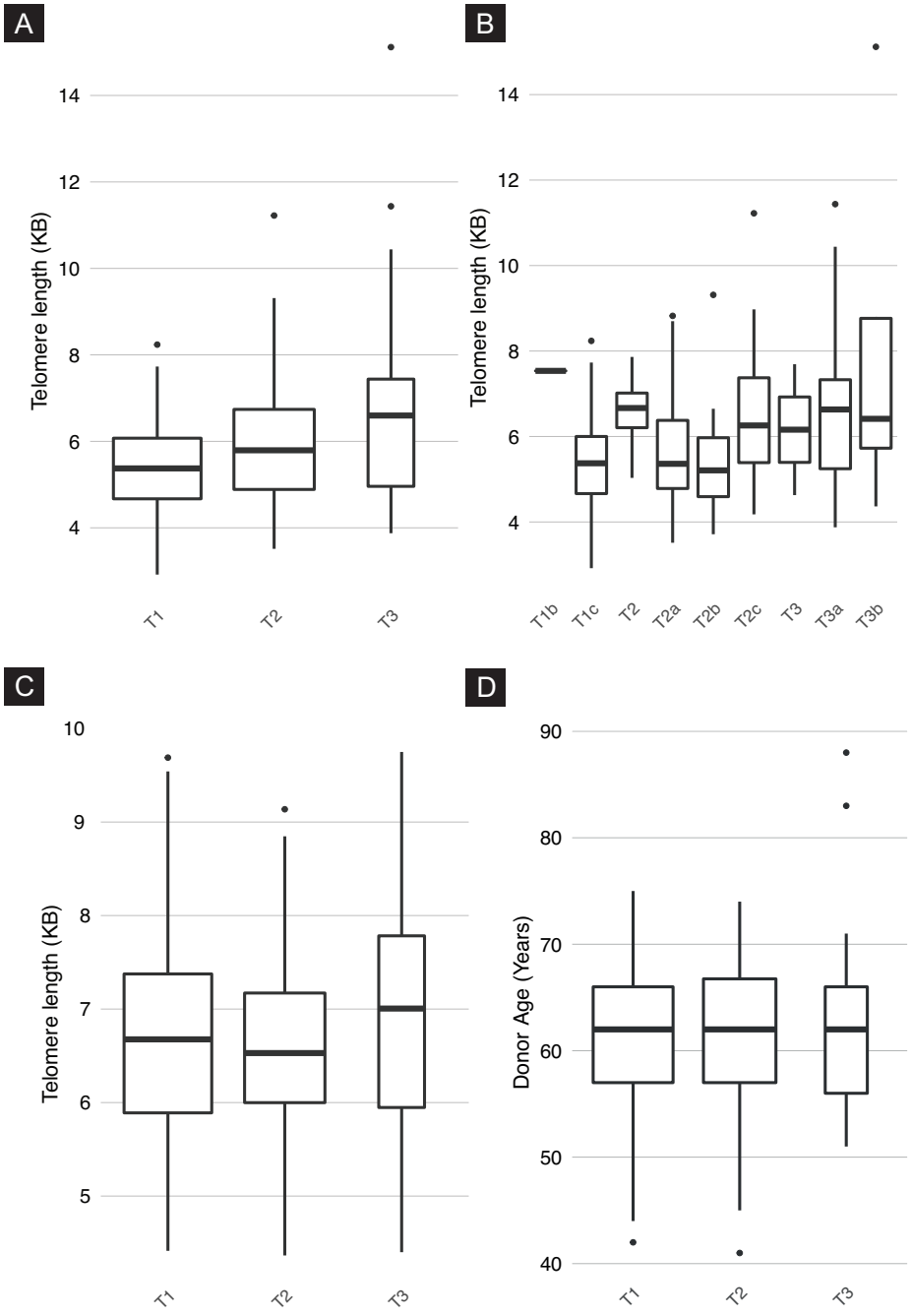


Fig. 4.7 Telomere lengths by tumour stage. ***(A)****: Tumour telomere length by tumour stage* ***(B)****: Tumour telomere length by the more granular definition of cancer stage* ***(C)****: Matched blood telomere length by stage* ***(D)****: Donor age by tumour stage*

Where "length" is tumour telomere length and "stage" is a dummy variable corresponding to which T stage the tumour was graded as.

ANOVA testing finds significant difference between stages We then subjected this linear model to an ANOVA test that was found to be highly significant ($p < 1 \cdot 10^{-4}$). The model describes significantly more difference in telomere length across all the groups than we would expect by chance. To test the difference *between* groups we used Tukey's range test. The results of this test are in Table 4.2. We see that telomere length is significantly differentiated between all three stages. Given these results we can reject the null hypothesis that there is no difference between telomere length when separated by cancer stage.

It is interesting to speculate that this result is an observation of overall telomere dynamics within PCa. Perhaps short telomeres bestow a selective advantage in that genomic instability is more frequent and therefore mutations and genomic aberration may occur in cells displaying shorter telomeres. Whereas long telomeres are preferential for later stage tumours where proliferative potential is prioritised. Previous studies have shown that TERT expression is greater within later stages of PCa (LIN *et al.*, 1997), so perhaps this lengthening is the result of a cumulative extension of telomere length over time.

The relationship between telomere length and stage is less clear when the samples are separated by the more granular subcategory definitions. This effect may be because the increase in categories forces a smaller amount of observations into each category. However, it is not clear that the subcategories are chronological (a tumour may progress through Stages T1c, T2b, T3a without passing through any of the other subcategories) so if this association is driven by the progress of the tumour through time, then we would not necessarily expect the relationship to hold for subcategories.

Investigations into the "M" and "N" portions of the TNM staging metric did not reveal interesting relationships to tumour length. M is a metric that determines whether a disease has metastasised and N is a metric for gauging the extent of spread to lymph nodes. There were not sufficient samples presenting a positive call for either the "M" or "N" categories within this cohort. As a result it was not possible to examine the relationship between telomere length and the "M" and "N" classification categories.

Telomere length in progressed samples Next we observed whether telomere length displayed any difference depending on whether or not the patient had experienced biochemical recurrence. Biochemical recurrence is a clinical measure defined as an increase in Prostate Specific Antigen (PSA). PSA has been shown to increase in the presence of PCa and so is a reliable estimator of disease progress (PALLER and ANTONARAKIS, 2013; STEPHENSON

et al., 2006). We see that telomere length was shorter in samples where the donor has progressed. We found this relationship significant in a two-sided T Student T test ($p = 0.03$, Figure 4.8A). We also observed that telomere length was separated by progression status across the tumour's clinical stage (Figure 4.8B). The largest effect was in samples displaying T1 clinical stage.

These observations may hint at a possible role for telomere length as a prognostic marker for disease progression, especially in early stage PCa. There is a precedent in the literature for telomere length serving as a prognostic marker in other cancers (VALLS *et al.*, 2011). Also, in PCa telomere length has been linked with cancer progression (HEAPHY *et al.*, 2013). However, in contrast to our observations, HEAPHY *et al.* do not see a link between short telomere length and increased chance of progression, but rather increased variability in telomere length.

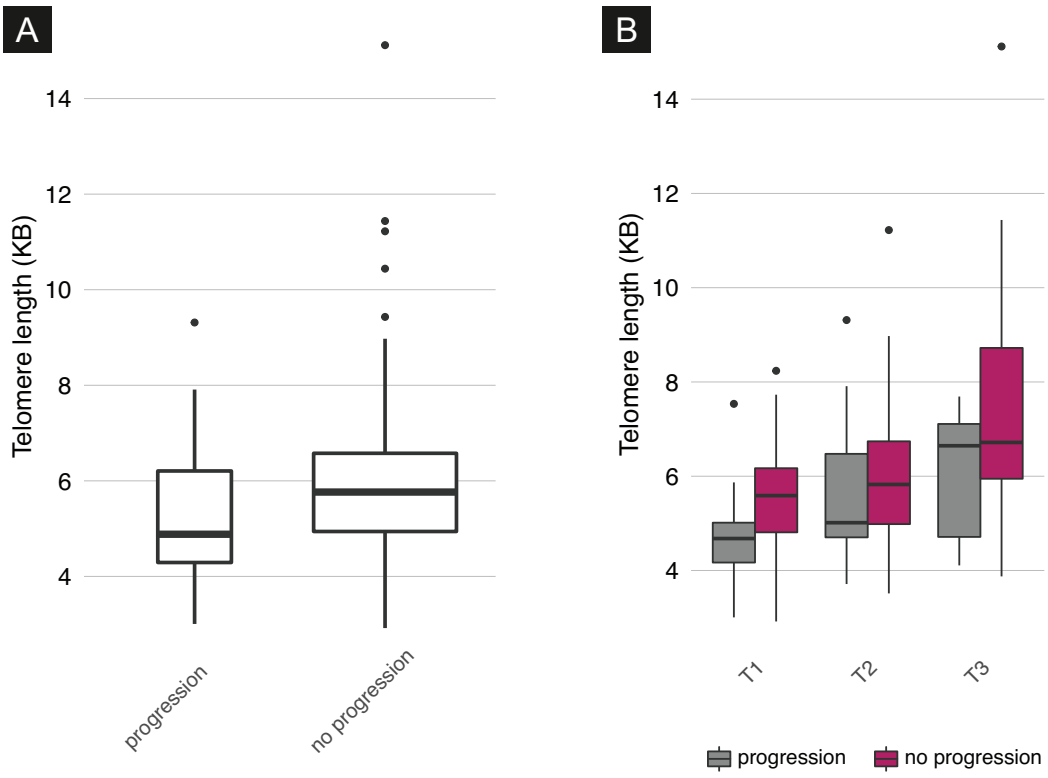


*Fig. 4.8 Disease progression and tumour telomere length **(A)**: Boxplot of tumour telomere length separated by disease progression status **(B)**: Tumour telomere length separated by stage and grouped by progression status*

No clear relationship with Gleason scoring or PSA In the course of this study we also tested whether telomere length was associated with Gleason grade and PSA. In both cases

we did not identify a clear relationship with either of the measures. Figure 4.9 shows plots detailing the relationship between these measures.

In the case of the Gleason score, there is some indication of a trend; samples with more severe scores seem to display longer telomeres. However, the lack of samples within the more severe categories (5+5 is comprised of a single sample and 5+4 is comprised of 4 samples) make this result hard to verify.

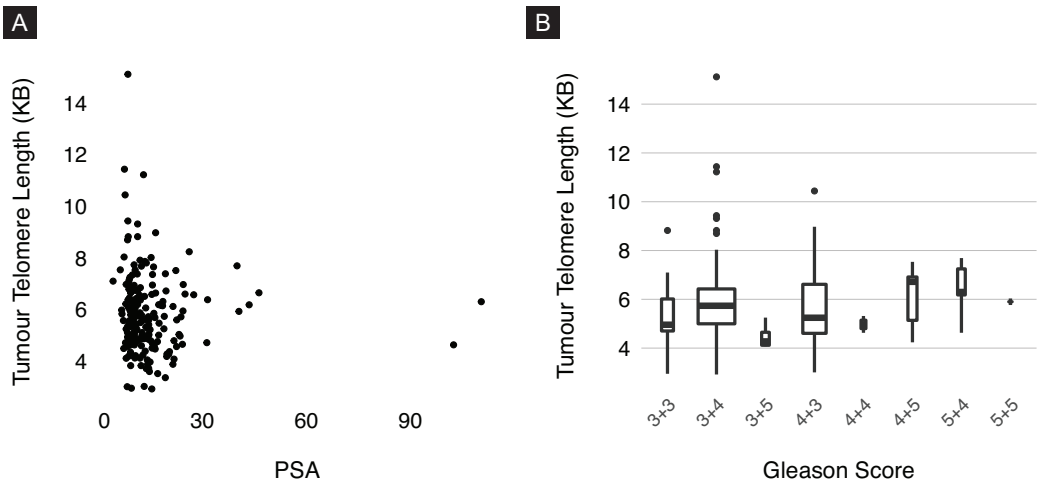


*Fig. 4.9 Plots showing the relationship between Telomere and Prostate **(A)**: PSA versus Telomere Length **(B)**: Gleason score vs Telomere Length*

4.3.3 Molecular subtypes

As we saw in the introduction to this chapter, previous studies have made considerable progress in the division of prostate cancer into molecular subtypes. The most recent and most comprehensive of these attempts is ABESHOUSE *et al.* (2015). ABESHOUSE *et al.* successfully characterised 74% of 333 PCa primary tumours between 7 separate molecular subtypes. The data at our disposal allowed us to test whether a subset of these molecular subtypes showed an association with telomere length.

TMPRSS2-ERG Of the original 192 tumour samples, 49 were profiled for a TMPRSS2-ERG fusion. Of these 49 samples, 23 (46%) were found to be fusion positive. Previous studies have found that 40-50% of prostate cancers display a TMPRSS2-ERG fusion so this observation is in line with expectation.

A two-sided Student T Test showed that the mean estimate for fusion positive samples was significantly higher than that of fusion negative samples ($p = 0.01$). We also observed that matched blood telomere length displayed an association with TMPRSS2-ERG fusion

verging on significant ($p = 0.06$). Figure 4.10 shows boxplots of telomere length plotted by TMPRSS2-ERG fusion status.

It is interesting that the relationship seems to be present in blood and tumour tissues. The situation is nuanced further by the fact that blood and tumour telomere length seem not to be strongly correlated. This observation requires further study to assess whether the relationship is genuine and whether causation can be attributed in either direction. However, these results pose the hypothesis that long telomeres predispose the genome to TMPRSS2-ERG fusions. TMPRSS2-ERG is known to occur early in PCa (PERNER *et al.*, 2007) so it is interesting to speculate that longer telomeres predispose the cell to TMPRSS2-ERG fusions in pre-malignant tissue. Perhaps this would go some way to explaining why telomere length seems to be associated with fusion status in both blood and tumour samples.

The literature poses another possible cause for this observation. Previous studies have suggested that TMPRSS2-ERG fusion is more common in samples from younger donors (STEURER *et al.*, 2014). If younger patients were more likely to exhibit a TMPRSS2-ERG fusion, then this might go some way to explain the association with long telomeres, as we would expect younger donors to have longer telomeres. However, amongst patients in our cohort, there seems to be no association with age and fusion status. This nullifies the chance that our observations of difference in telomere length between fusion positive and negative samples is simply a reflection of the age of patients. However, that is not to say that this effect may not be observable on a cohort where younger patients *were* in abundance in TMPRSS2-ERG fusion positive samples and should be addressed in such analyses.

SPOP Another common molecular subtype is characterised by mutations in the SPOP gene. 13 of the tumour samples in our cohort carry an exonic somatic SNV in the SPOP gene and so can be considered part of this subtype. Of these 13 samples eight are missense mutations.

Using a Student T Test we see that the mean telomere length of samples with an SPOP mutation are significantly shorter. This effect is not seen in associated blood samples. Figure 4.11A shows boxplots for telomere length plotted by presence of SPOP mutation.

We wished to ensure that short telomeres in SPOP samples were a separate phenomenon to long telomeres in TMPRSS2-ERG samples, given that the two subtypes are known to be mutually exclusive. To test this we split the cohort into three groups: *SPOP*, *TMPRSS2-ERG* and *Uncharacterised.* An ANOVA test and Tukey's range test finds all three of these categories to be significantly differentiated (see Figure 4.11B). This approach has one major caveat. Not all of the samples in the cohort were characterised for TMPRSS2-ERG fusion

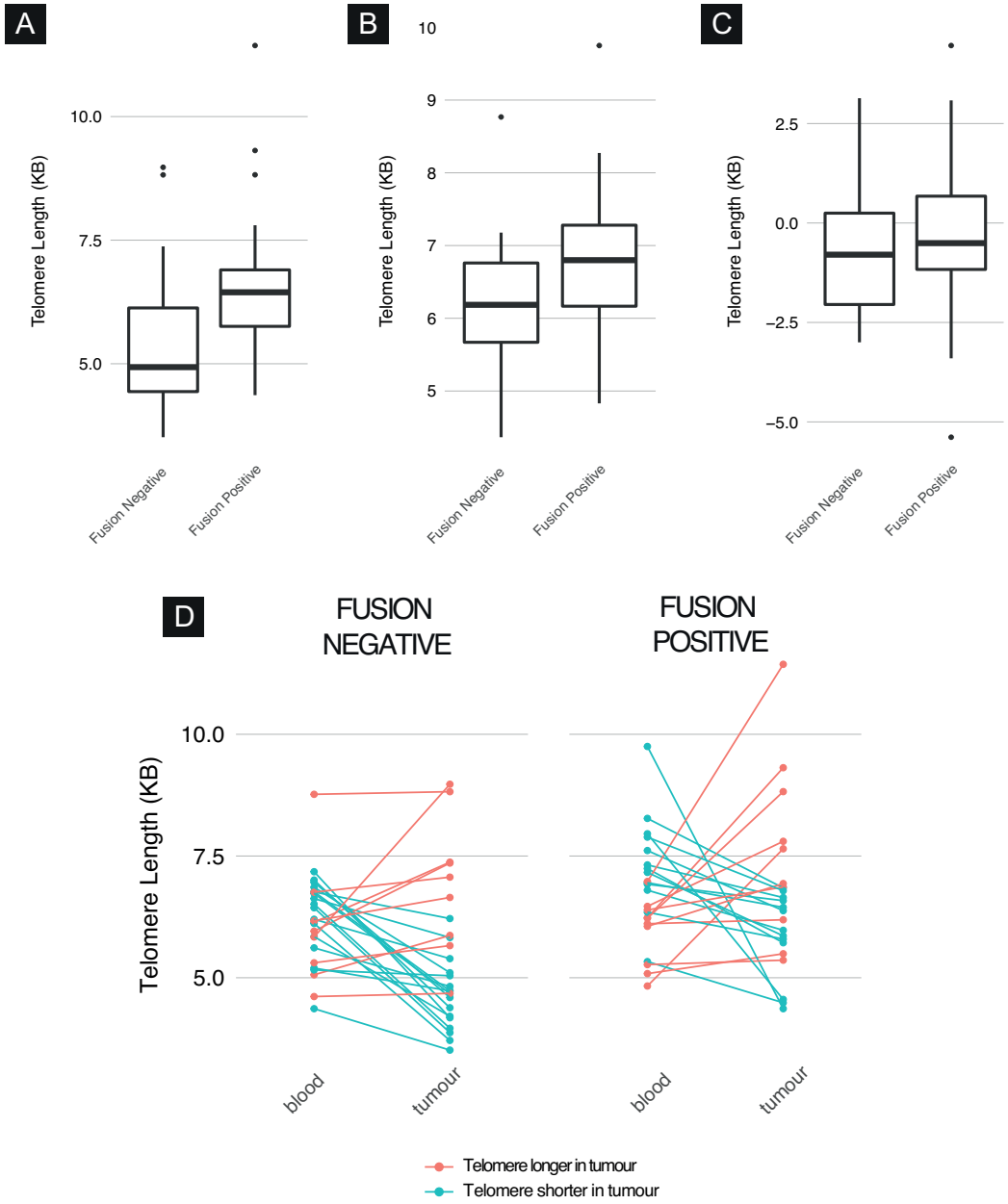


*Fig. 4.10 TMPRSS2-ERG fusion and telomere length. **(A)**: A boxplot showing tumour telomere length vs fusion status. **(B)**: A boxplot showing blood telomere length vs fusion status **(C)**: Boxplot of the difference between tumour and blood telomere length, by fusion status. **(D)**: A line and point plot showing the relationship between tumour and blood telomere length in both fusion negative and positive samples*

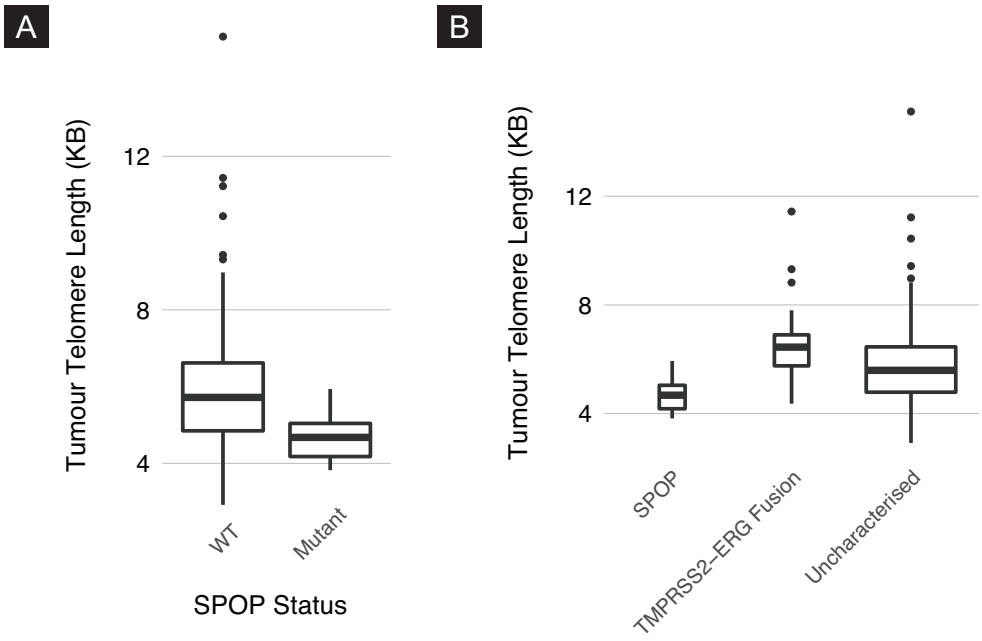


Fig. 4.11 Plots showing the relationship between the SPOP subtype and telomere length. ***(A)****: Samples with a somatic mutation in SPOP display significantly shorter tumour telomere in comparison to samples without a mutation* ***(B)****: When we separate out samples with TMPRSS2-ERG fusions we continue to see good separation between the categories. The separation between all three categories is shown to be significant using Tukey's range test*

and so there are almost certainly still samples with a TMPRSS2-ERG fusion extant in the Uncharacterised category.

Given this association between SPOP mutations and shortened telomere length, we had cause to consider the underlying nature of the relationship.

It seems unlikely that telomere shortening is causing SPOP to accrue mutations. The SPOP gene is located distally from either telomere, and so is unlikely to experience localised damage as a direct result of telomere shortening.

It also seems unlikely that these mutations are causing an increased expression of the SPOP gene leading to shortened telomere. We observe that across a set of ten separate cancer studies, the mutations in SPOP does not lead to higher expression in SPOP (see Figure 4.12).

This leads us to consider a functional change in the nature of SPOP driven by mutation. We note that SPOP is known to interact with DAXX (La *et al.*, 2004). DAXX has well profiled associations with telomere, including as a known facilitator of alternative lengthening of telomeres (commonly referred to as ALT) (Heaphy *et al.*, 2011a). Additionally, DAXX

has been profiled as a key protein in an apoptotic pathway through its relationship to Fas (YANG *et al.*, 1997).

It may be the case that the presence of shortened telomere in samples with SPOP mutations, is tied to some function of the DAXX protein as a result of an altered interaction with SPOP. Perhaps, if apoptosis is accelerated then cells must proliferate in order to remain viable as a tumour, resulting in shortened telomere. This narrative points to SPOP mutated tumours as being encumbered by poorer replicative potential, in that they could potentially surpass the Hayflick limit more quickly. In concordance with this narrative ABESHOUSE *et al.* (2015) show that samples with SPOP mutations were under represented as a proportion of metastatic samples.

FOXA1 and IDH1 Lastly, we observed whether samples with exonic mutations in FOXA1 and IDH1 showed a relationship to telomere.

In the case of FOXA1 we did not see any specific relationship to telomere in either blood or tumour telomere length. In the case of IDH1, only a single sample in our cohort displayed a mutation in IDH1 so we lacked the statistical power to check for an association to telomere length. The lack of samples displaying the IDH1 subtype was to be expected. In the ABESHOUSE *et al.* analysis it was observed in only 1% of cases.

ABESHOUSE *et al.* also postulate that IDH1 is a subtype involved with the early onset of prostate cancer. If this is the case it would be interesting to see whether patients with this subtype presented with shorter telomeres in adjacent normal tissue. The rarity of the IDH1 subtype means that greater sample sizes will be required to gauge the nature of its relationship to telomere. Indeed in this cohort there are not enough samples presenting the IDH1 to conduct a meaningful analysis.

4.3.4 Somatic mutations and telomere length

After confirming a relationship between somatic mutations in the SPOP gene and telomere length, we wished to investigate whether any other genes displayed a relationship to telomere length. Somatic Single Nucleotide Variations (SNV) calls were made using the CaVEMan tool (JONES *et al.*, 2016) for each of the 192 tumour samples. Somatic SNVs are SNVs that are identified in the tumour sample but not the corresponding normal sample. This narrows our search to SNVs that have occurred within the tumour and therefore may be relevant to the progression or development of the cancer.

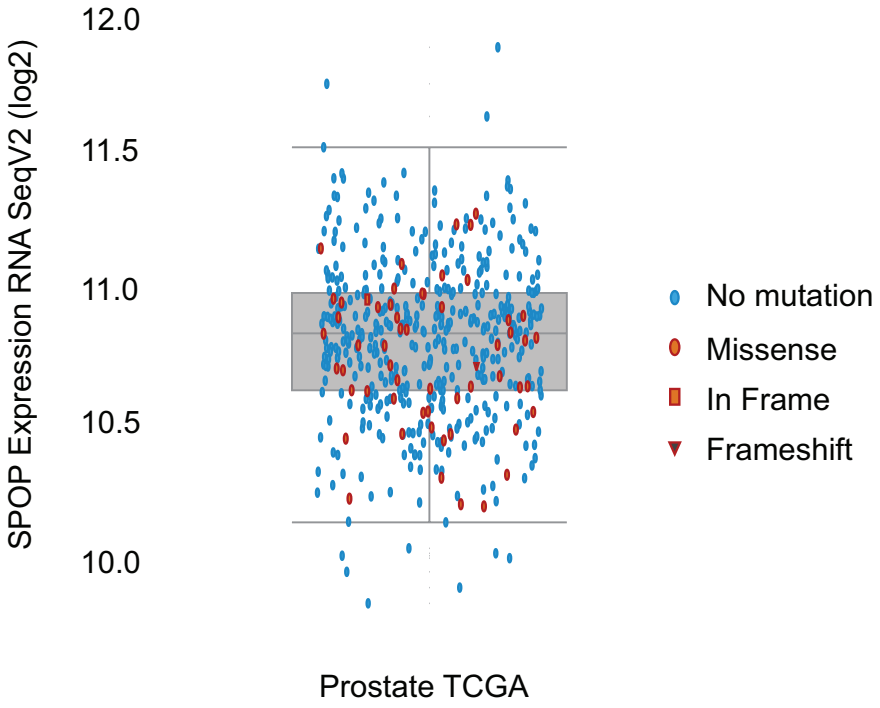


*Fig. 4.12 A figure produced by cBioPortal (*GAO et al.*, 2013) showing samples from all the PCa studies hosted on cBioPortal. Each point is coloured by SPOP mutation status and plotted on axis of RNA expression. Samples with SPOP mutations (red coloured dots) are not associated with either high or low expression*

Comprehensive gene analysis We started by taking a genome wide approach to the analysis. For each of our samples we recorded which genes displayed a missense SNV occurring in an exonic location, across all known genes.

We then ran a two sided Student T Test, comparing the telomere length of those samples with a mutation and those without for each gene where at least 3 samples displayed an exonic mutation. Only genes where more than 3 samples displayed a SNV were considered by this analysis. Only 37 genes in the analysis met these criteria. Of these 37, only two returned p-values less than 0.05. These were SPOP (13 samples, $p < 1 \cdot 10^{-15}$) and PTPRD (4 samples, $p = 0.03$). Once we adjusted for multiple hypothesis testing using FDR, only SPOP remained significant.

The results of this analysis seem to suggest that, other than the previously identified relationship with SPOP, there are no other clear relationships to somatic SNVs and telomere length. According to the online gene information repository GeneCards (REBHAN *et al.*, 1998) PTPRD is known to be an exceptionally long gene at 2MB long. This probably explains the reason for an enrichment of somatic SNVs within the gene. The length of the gene, coupled with the fact that it did not survive multiple hypothesis correction, indicates that this is probably not a genuine relationship to telomeres.

Somatic mutations within the telomere interactome The genome wide analysis investigation was in part hampered by the fact that relatively few genes displayed SNVs in more than a handful of cases. This was not unexpected as PCa is known to display relatively few somatic SNVs in comparison with other cancers (MARTINCORENA and CAMPBELL, 2015).

We investigated whether somatic SNVs in specific genes, known to interact with telomeres, had a relationship with telomeres. Rather than focusing on single genes we used an online tool to compile a list of genes that are reported to share an affiliation with a set of "seed genes" which are known to interact directly with telomeres. We shall refer to network of genes as the "interactome". For this analysis we considered all types of somatic SNVs rather than limiting the search to only missense mutations.

To construct the telomere interactome we identified a small set of genes known to interact directly with telomere (see genes labelled "seed genes" in Table 4.4). We then used the GeneMania (WARDE-FARLEY *et al.*, 2010) gene network database to build a network from the seed gene set. A depiction of the resultant network produced by GeneMania is shown in Figure 4.13. All of the genes in the final interactome are listed in Table 4.4.

GeneMania considers multiple sources of evidence to link genes with one another. The tool aggregates data from co-expression, known pathway affiliation and physical interactions to form gene networks. Additionally, GeneMania allows the users to refine their search by

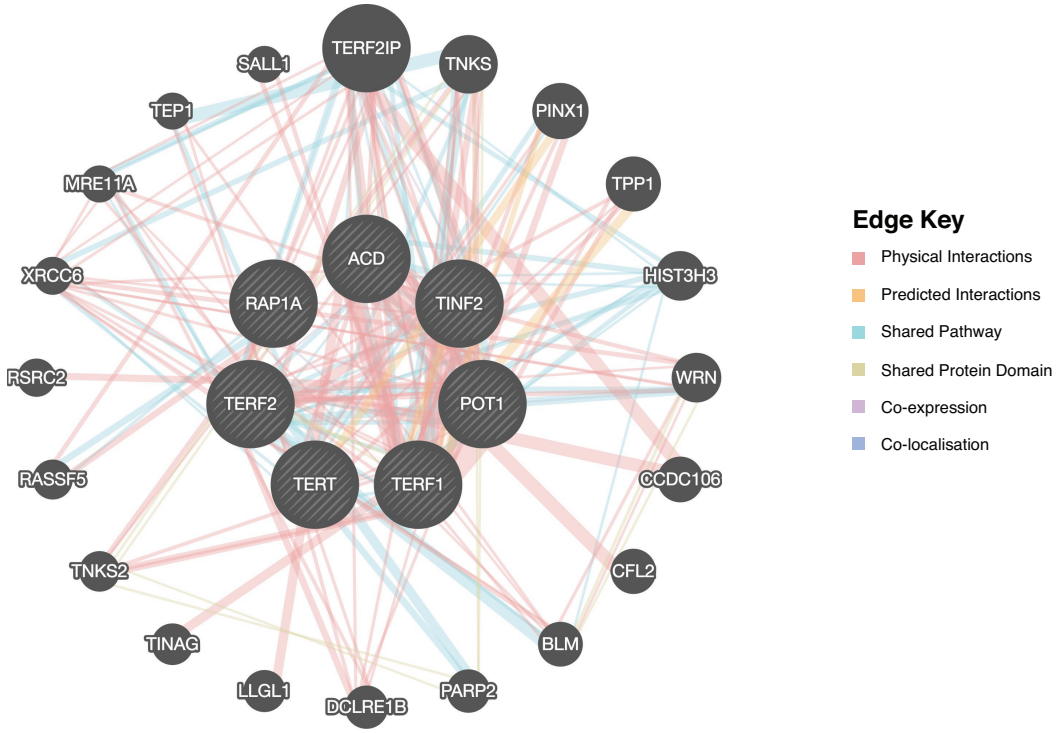


Fig. 4.13 The telomere interactome network as determined by GeneMania

specifying a tissue of interest. For the purposes of this investigation we specified interactions within prostate tissue.

We had hoped that by considering genes within the context of their interactions it would increase the amount of effect mutations within the cohort. We hypothesised that telomere length may be altered by a mutation at some point in the telomere interactome. As PCa is a cancer known to show relatively few somatic mutations inspecting genes individually might not give enough power to detect such an effect. Indeed, we see that of the genes in our telomere interactome BLM is the single gene with the most affected samples, but only four samples display an exonic mutation in BLM. However, by considering the interactome as a whole we see that 23 of the samples have at least one exonic SNV in one of the identified genes.

This analysis did not find any relationship between somatic SNVs in the telomere interactome and telomere length (see Figure 4.14A). When we inspect the genes individually only BLM seems to associate with differentiated telomere length, a boxplot is shown in Figure 4.14B. However, with only four affected samples it is difficult to assess the validity of this finding. Furthermore, of those four samples with a somatic SNV in BLM, only one of them is a missense mutation.

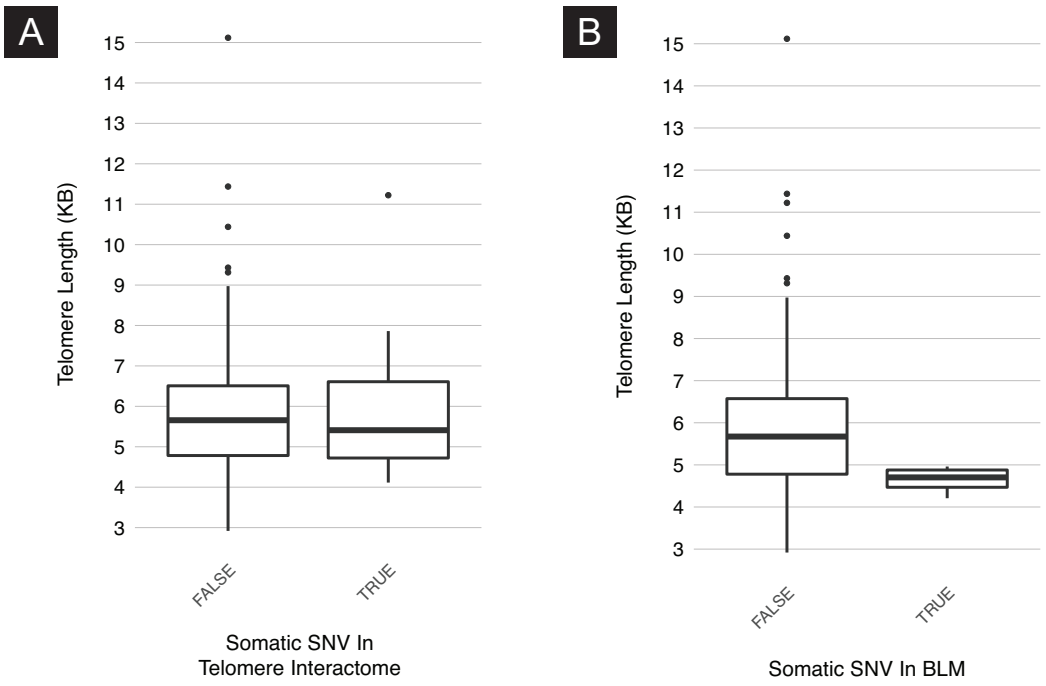


Fig. 4.14 Boxplots showing the relationship between somatic SNVs in the interactome and telomere length ***(A)****: Telomere interactome somatic SNV status versus telomere length* ***(B)****: Samples separated by whether or not they had an exonic BLM mutation versus telomere length*

Table 4.4 Genes in the telomere interactome analysis

	Gene	# of samples with SNV	# missense	*Notes*
1	TERT	1	1	Seed; telomerase
2	POT1	0	0	Seed; shelterin
3	TINF2	0	0	Seed; shelterin
4	TERF1	0	0	Seed; shelterin
5	TERF2	0	0	Seed; shelterin
6	RAP1A	0	0	Seed; shelterin
7	ACD	1	0	Seed; shelterin
8	MRE11A	0	0	
9	PPP1R10	2	0	
10	BLM	4	1	
11	PARP2	1	0	
12	ACADVL	1	0	
13	TEP1	2	1	
14	CHAMP1	2	0	
15	PINX1	0	0	
16	HIST3H3	0	0	
17	TNKS	1	0	
18	RAD50	1	0	
19	NBN	0	0	
20	XRCC6	2	2	
21	CTC1	3	2	
22	STN1	1	0	
23	MCPH1	0	0	
24	DCLRE1B	1	1	
25	CCDC106	0	0	
26	WRN	1	0	

Table 4.5 Variables from the total number of somatic mutations analysis

	p-value	Coefficient
Length	0.034	-0.002
Age	$1.22 \cdot 10^{-8}$	1.46

When this interactome analysis was constrained to only exonic missense mutations we observed fewer samples with mutations. With only 8 samples displaying a missense SNV in any of the interactome genes. We did not observe any relationship to telomere length using this more constrained definition of inclusion. Furthermore the paucity of missense mutations may also suggest that there is little selective pressure towards acquiring them within telomere related genes.

4.3.5 The total number of somatic mutations

After failing to identify a significant relationship between any specific SNVs and telomere length (other than SPOP) we decided to see whether there was a relationship to the number of accrued somatic SNVs within the sample and telomere length.

Shorter telomeres are hypothesised to expose DNA to greater risk of DNA damage (O'SULLIVAN and KARLSEDER, 2010). We reasoned that this predicts that tumour samples in our cohort with shorter telomeres should display more evidence of DNA damage. To test this we sought to define the relationship between telomere length and the number of somatic SNVs within the cohort.

For each donor we observed the amount of SNVs that were in a sample. We restricted our analysis to SNVs occurring within any part of all known genes. We then defined the following linear model

$$\text{snvs} = \beta_0 + \beta_1 \text{length} + \beta_2 \text{age} + \varepsilon \tag{4.3}$$

Where length is the telomere length estimated by Telomerecat and age is the age at tumour collection for each donor. The model indicates that both the length of telomere and the age contribute significantly to explaining the variance in the model (see Table 4.5).

Significant associations with age and telomere Age is strongly associated with the number of somatic mutations. As we saw in the Introduction to this thesis, prostate cancer is often characterised as a disease with slow progression, especially in early stages (PARKER, 2004). Tumours are also liable to have existed for a prolonged period of time before diagnosis due

to the symptomless nature of early disease (ETZIONI *et al.*, 1998). It follows that an older patient is more likely to have an older tumour which in turn has had more time to gather somatic mutations. This would explain the strong relationship between patient age and the amount of somatic SNVs.

The linear model also showed a significant role for telomere length in explaining the amount of somatic SNVs in a sample. In Section 4.3.1 we saw that tumour telomere length had no relationship with the age of the patient. We conclude therefore that while age is certainly a factor in SNV acquisition, it is a separate phenomenon to that of telomere shortening.

An analysis of this nature cannot comment conclusively on the causative nature of shorter telomere and an increased number of somatic SNVs. Indeed it may be that a sample with more somatic SNVs is more likely to accrue damage to genes responsible for the elongation of telomere. However, given the known protective role of telomere on the genome it is plausible that the length of telomere has a direct effect on the number of somatic SNVs.

4.3.6 Telomere length and RNA expression

Next, we examined the relationship between RNA expression, as deduced by Illumina micro array experiment and telomere length in a subset of the prostate cohort. We had expression measurements for 47 of our original 192 tumour and blood sample pairs (one sample was removed as an outlier see the discussion section of this chapter for details).

We observed that a substantial amount of the expression probes in the analysis were not differentially expressed across the samples. Figure 4.15 shows the distribution of the difference between maximum and minimum observed values for each probe. In order to focus on the analysis that showed heightened differential expression across our samples, we chose probes where the difference between the minimum and maximum observed value on the probe was greater than 1.2. This threshold represents the 75th percentile of all probes and is shown by the red line in Figure 4.15. As a result, we excluded 35929 probes leaving 12178 probes as the subject of the analysis.

In order to identify associations between telomere length and RNA expression, we fit a linear model with the following variables for each of the expression probes in our dataset:

$$\text{length} = \beta_0 + \beta_1 \text{expression} \tag{4.4}$$

Where *length* are the tumour telomere length estimates for the cohort and expression is the relevant expression probe.

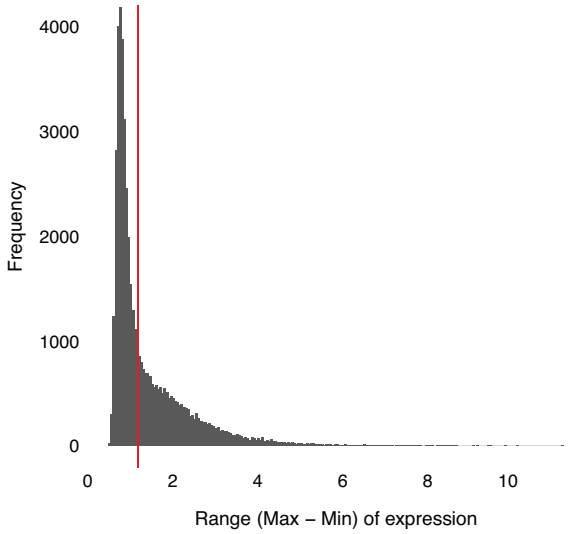


Fig. 4.15 A histogram showing the ranges of all probes in the expression data set. The red line denotes the threshold at which probes were considered for the analysis detailed in Section 4.3.6

The probes with the highest R^2 are given in Table 4.6. This analysis did not identify any probes that were significantly correlated with telomere length once we had corrected for multiple hypothesis testing with FDR.

4.3.7 The Complex Men

Whereas the previous sections of this chapter have focused on the main PCa ICGC cohort, this section describes the telomere lengths across a set of patients known as the "complex men". The complex men are a set of three donors each displaying multi-focal PCa. For each man, samples were collected from several areas of the prostate thought to be cancerous. These separate areas are referred to as "cores" throughout this section. Additionally, adjacent tissue and circulating blood samples were also collected. Each of the samples was subjected to WGS analysis, somatic SNV calling and TMPRSS2-ERG status profiling. The complex men were the subject of an extensive study as detailed in COOPER *et al.* (2015). For historical reasons the complex men are referred to individually as cases 6, 7 and 8.

We wished to explore the telomere dynamics across these models of multi-focal PCa. For each of the complex men, COOPER *et al.* constructed phylogenies detailing the relatedness of each of the tumour cores, based on the distribution of SNVs across each of the cores. Figure 4.16 shows the phylogeny for each man alongside the corresponding telomere lengths, as estimated by Telomerecat, for each sample.

Table 4.6 The 30 probes with the best correlation with tumour telomere length. FDR N = 4376

	Gene	R^2	$p-value$	FDR	Association
1	MPST	0.30374	0.00006	0.24470	-
2	SLC7A8	0.30209	0.00006	0.24470	-
3	SP8	0.29160	0.00009	0.24470	+
4	NSMCE4A	0.29106	0.00009	0.24470	+
5	CBX1	0.28386	0.00012	0.24470	+
6	AGBL5	0.27962	0.00013	0.24470	-
7	AGR3	0.27150	0.00017	0.24470	+
8	DMKN	0.27014	0.00018	0.24470	+
9	BANP	0.26556	0.00021	0.24470	-
10	FOXRED2	0.26077	0.00025	0.24470	-
11	FKBP4	0.25919	0.00026	0.24470	-
12	PIGX	0.25835	0.00026	0.24470	-
13	PRRG4	0.25294	0.00031	0.24470	-
14	ZNF614	0.25133	0.00033	0.24470	+
15	SWI5	0.25021	0.00034	0.24470	-
16	OAT	0.24926	0.00035	0.24470	+
17	PAPL	0.24782	0.00037	0.24470	-
18	ZNF615	0.24252	0.00044	0.25783	+
19	MBOAT2	0.24211	0.00044	0.25783	-
20	LOC100132057	0.24101	0.00046	0.25783	-
21	FAM65B	0.23624	0.00053	0.27229	+
22	NIFK	0.23623	0.00053	0.27229	+
23	SRGAP1	0.22851	0.00068	0.32185	-
24	RPS6	0.22807	0.00069	0.32185	+
25	TAP2	0.22281	0.00081	0.32855	+
26	PCYT2	0.22108	0.00085	0.32855	-
27	MKNK2	0.22030	0.00087	0.32855	-
28	ATP7B	0.21822	0.00093	0.32855	+
29	PON2	0.21782	0.00094	0.32855	+
30	GTF2F2	0.21607	0.00100	0.32855	+

COOPER *et al.* made a brief mention of the telomere length for these samples in their study. Using TelSeq they estimated the lengths of the adjacent normal tissue to be 6.3KB for case 6 and 6.2KB for case 7. Additionally, they state that they consider these samples as having "not undergone attrition". However, this claim seems unsubstantiated given that this conclusion has been drawn using only a single measurement of the normal telomere length. Furthermore, when we compare these telomere lengths to those generated by TelSeq

on the TwinsUK data (see Figure 3.8) we see that a telomere length of 6KB is one standard deviation below the mean of that cohort. COOPER *et al.* go on to describe telomeres in the corresponding cancer to be "slightly longer" than the adjacent tissue normal telomere length in both cases. When we applied TelSeq to the corresponding tumour length we could not confirm these claims. In both cases we observed tumour telomere lengths both longer and shorter than the adjacent tissue normal. No mention is made of case 8. Whilst COOPER *et al.* represents an interesting first foray into these data we believe that a more comprehensive analysis might shed more light on the extent of heterogeneity across the cores.

We applied Telomerecat to all samples within the complex men sub-cohort as shown in Figure 4.16. Telomerecat produces estimations of telomere length which seem to complement the story of overall heterogeneity amongst complex men. Cases 6 and 7 seem to display pronounced intra-tumour heterogeneity. Whereas Case 8, although presenting the longest adjacent telomere length, displays exceedingly little telomere heterogeneity across the individual tumour cores. Both case 6 and 7 show a difference of approximately 2KB between the longest and shortest telomere core.

These observations provide insight into the extent of intra-tumour heterogeneity of telomere length. To our knowledge, this is the first analysis to show that telomere length differs across an in-vivo set of tumour cores. Further work is required to assess whether telomere length has any causative actions with the progressions of the individual cores, or whether the difference in telomere length is a downstream effect of different pressures experienced by different parts of the tumour.

4.4 Discussion

This study represents the largest (by sample number) investigation into telomere length within PCa and as such has uncovered several hitherto unreported associations. The size of this study speaks to the advantages of both Parabam and Telomerecat; minimal costs were incurred in estimating telomere length for the 384 samples interrogated as part of this analysis. What's more, thanks to the parallel nature of Parabam the analysis was completed in a timely fashion.

In this section we shall summarise the findings and suggest routes for further investigation.

Identifying associations in PCa Some of the most striking associations are those found within the molecular subtypes. With SPOP known to associate with the DAXX protein, it seems that there is a plausible explanation for telomere shortening with that subtype. Our findings also indicate that there may be an association between TMPRSS2-ERG and blood

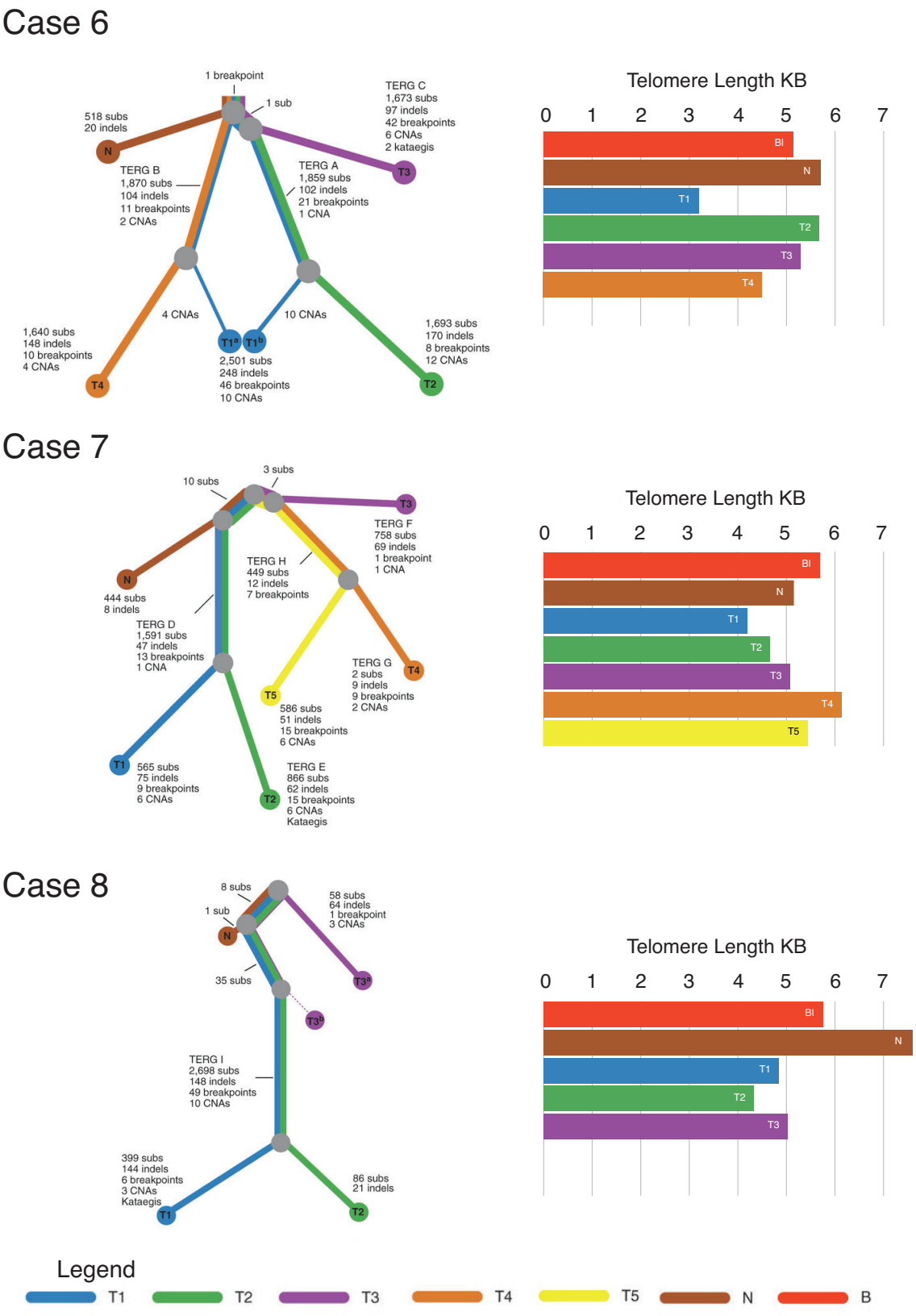


Fig. 4.16 Complex men tumour phylogenies and corresponding telomere lengths as estimated by Telomerecat. The phylogeny diagrams incorporated into these plots are reproduced from COOPER et al. *(2015)*

telomere length, perhaps suggesting a role for longer telomeres in the aetiology of this genotype.

This study represents the first time that SPOP has been linked to telomere length within PCa and represents an exciting avenue for further exploration. Above, we detailed the possibility of a causal effect on telomere length via an altered association to the DAXX protein. A recent study by BLATTNER *et al.* (2017) introduced a murine model which posits a narrative for SPOP mutations in driving tumourigenesis via deregulation of PI3K/mTOR and AR pathways. Interestingly, other studies have shown a link between the PI3K/mTOR pathway and telomerase expression (YAMADA *et al.*, 2012; HEEG *et al.*, 2011). Alongside the previously proposed DAXX association, this represents another viable route of further investigation.

Another interesting result from this study is the observation of a highly significant relationship between telomere length and tumour stage. Previous studies studies have already identified a link between telomerase expression (LIN *et al.*, 1997), so this result certainly has strong precedent. However, to our knowledge, this study represents the first time that telomere length (as opposed to just telomerase expression) has been observed in PCa.

It is interesting to note the dynamics of telomere length across the stages. It is possible that short telomeres in early stage cancers are preferable as a method for accruing crucial SNVs (as we saw above, samples with shorter telomeres also had more SNVs). Perhaps, longer telomeres in later stage cancers are selected for, to allow for greater proliferative potential. It may be that tumour cells that fail to adequately elongate telomere length fail to replicate and contribute to the cell population of later stage tumours.

Of the analyses detailed above, two stand out as having failed to identify significant relationship to telomere length. The first of these is the genome wide and telomere interactome SNV analysis. No somatic SNVs were significantly associated to telomere length (other than the SPOP association uncovered as part of the subtype analysis). As mentioned, this result was not unexpected. PCa is known to display a below average amount of somatic SNVs.

The comparison to micro-array RNA expression data also failed to identify significant relationships. Early attempts at the analysis found numerous associations with the KANK4 and the DMKN protein. However, on inspection of the individual probes it appeared that this effect was driven by a single outlier sample. Figure 4.17 shows relationships for these probes with the outlier included. In the case of the KANK4 probe, when the outlier was included we observed $R^2 = 0.34$, however this dropped to $R^2 = 0.12$ with the outlier removed. In the case of the DMKN probe we observed $R^2 = 0.42$ drop to $R^2 = 0.27$. In both of these cases, once the outlier was removed both relationships were not found to be significant after adjusting for multiple hypothesis testing.

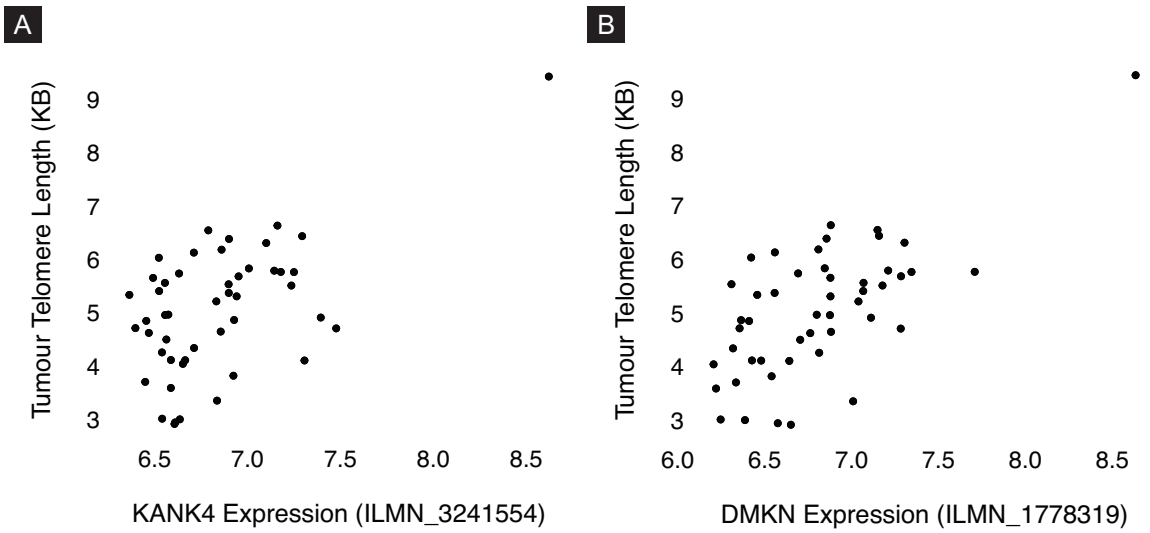


Fig. 4.17 Plots detailing significant RNA associations before removal of an outlier sample. Note the outlier sample in the top right hand corner of both plots ***(A)*** *Tumour telomere length vs KANK4 expression.* ***(B)*** *Tumour telomere length vs DMKN expression*

Need for normalisation We had hoped that the design of Telomerecat would mean that batch normalisation was not needed. However, from the earliest stages of the investigation this appeared not to be the case as we observed significant batch effects.

Our initial optimism was based on the premise that fully telomeric reads (F1) would be subject to the same sequencing biases as boundary reads (F2a). However, this is clearly not the case and as demonstrated above (Section 4.2), normalisation was clearly required in order to produce accurate results. This finding should inform further studies using WGS telomere length estimation. It is our view that a normalisation step will be required in cases where sequencing has been undertaken on multi-platforms or on multiple occasions.

Cellularity In the course of this study, no active attempt has been made to correct for the effects of cellularity. The primary justification for this approach is that it is unclear how "normal" directly adjacent cells are in terms of their telomeres. This is especially the case for solid tumours where normal tissues are fixed in their position within the organ and therefore exposed directly to prolonged effects from their proximity to the tumour.

It is beyond the abilities of any WGS telomere length estimation tool to separate normal and tumour telomere lengths from the reads alone. However, there exist experimental methods, such as qFish that lend themselves better to precise delineation of tumour and normal telomere. This is a limitation of any study using telomere length estimated from WGS

data and has implications for the applicability of the method and their role in identifying underlying cancer telomere biology.

We see the utility of tools like Telomerecat in acting as a broad first sweep of the data that can highlight areas of interest for further study. Indeed, a trade off of the increased specificity of imaging techniques like qFish is the inherently lo-throughput nature of the methods. By using Telomerecat first on available WGS data researchers can more quickly hone in on specific areas of interest. We will expand upon this discussion for the role of Telomerecat in large scale WGS studies in Chapter 6, the conclusion to this thesis.

Chapter 5

Applying Telomerecat to a cohort of rare disease samples

All the WGS samples analysed as part of this chapter were collected, sequenced and preprocessed by the NIHR BioResource consortium on behalf of the NIHR. All of the additional phenotype (excluding Telomerecat estimates) and genotype data was processed and normalised by the NIHR BioResource consortium. Similarly the GWAS was run by collaborators at the NIHR on an established pipeline.

This chapter is the result of collaboration between Dr James Thaventhiran, Dr Hana Lango Allen and the author. The text in this chapter was written by the author with clinical input provided by Dr James Thaventhiran. All figures and tables in this chapter were produced by the author.

In this chapter we detail the application of Telomerecat to WGS data from 5760 subjects recruited into the National Institute for Health Research (NIHR) rare-disease from the BioResource cohort, a study run by the National Institute for Health Research (NIHR). For quality assurance we measured the ability of Telomerecat to identify known population variables that correlate with telomere length. We confirmed defined age and sex correlations with telomere length and extend this by identifying known single-nucleotide polymorphisms (SNP) that correlate.

After validateing the method accuracy with WGS data from this cohort by observing strong, previously identified trends in the data, we assessed an immunodeficient subcohort for novel genetic variants associated with low telomere length. This analysis uncovered novel variants in the genes DKC1 and TERT that cause monogenic telomerase deficiency, telomere shortening and life threatening illness.

Table 5.1 A set of sub-cohorts within the NIHR BioResource Study for which we generated telomere length. The total number of samples analysed, across all sub-cohorts, was 5760

	Cohort	Affected	Unaffected	Total	Focus
1	BPD	431	117	548	Bleeding, Thrombotic and Platelet Disorders
4	CSVD	103	92	195	Cerebral Small Vessel Disease
5	EDS	17	0	17	Ehlers-Danlos syndrome
6	HCM	194	0	194	Hypertrophic Cardiomyopathy
7	ICP	176	0	176	Intrahepatic Cholestasis of Pregnancy
8	LHON	45	21	66	Leber Hereditary Optic Neuropathy
9	MPMT	369	51	420	Multiple Primary Tumours
10	NPD	139	1	140	Neuropathic Pain Disorder
11	PAH	1043	54	1097	Pulmonary Arterial Hypertension
12	PID	937	358	1295	Primary Immune Disorders
13	PMG	122	0	122	Primary Membranoproliferative Glomerulonephritis
14	SMD	94	98	192	Stem Cell & Myeloid Disorder
15	SPEED	1032	17	1049	Paediatric neurology & Inherited retinal disease
16	SRNS	238	11	249	Steroid Resistant Nephrotic Syndrome

5.1 Background

The NIHR BioResource - Rare Disease cohort was established with the aim of producing a database of high coverage WGS samples for patients suffering from rare disease. The cohort spans a range of sub-cohorts focused on separate rare disease. Table 5.1 details the different sub-cohorts and the diseases on which they focus.

As Table 5.1 demonstrates, some of the subcohorts contain a number of samples from donors recruited into the study as relatives of donors affected by the disease or syndrome of interest. These patients are referred to as unaffected throughout the following text.

Later parts of this chapter focus on samples from the Primary Immune Disorders (PID) subcohort. PID is a heterogeneous group of conditions that leads to an increased susceptibility to infection, malignancies and auto-immunity. The genetic architecture of PID has been progressively defined and now, disease causing genomic variants in over 300 genes have been identified. This cohort was established to uncover novel genetic variants that cause human immunodeficiency.

Within the PID cohort, 75% of patients suffer from the immunodeficiency termed Common Variable Immunodeficiency (CVID). CVID is characterised by deficient antibody production, usually within the first three decades of a person's life (IGLESIAS ALZUETA and MATAMOROS FLORI, 2001). Despite ongoing research from the middle of the previous century, the underlying genetic causes of CVID are unclear (ABOLHASSANI *et al.*, 2013). In addition to CVID, the PID cohort also includes patients suffering from autoinflammatory syndromes (MASTERS *et al.*, 2009) and haemophagocytosis (ISHII *et al.*, 2005), amongst others.

Telomere length for 9,850 samples from the cohort were estimated. Of this original 9,850 samples we had age and gender data for 5,760. Due to the possibly confounding nature of these two variables it was decided to proceed only with samples for which we had age and gender meta-data. Thus the following analyses are conducted on 5,760 WGS samples.

5.2 Preliminary Investigations

Before we could analyse the data for associations to telomere length, we undertook a process to normalise and correct the data for confounding variables. In this section, we detail the process of batch effect reduction and also observe associations between age and gender with telomere length.

5.2.1 Removing batch effects

Investigations detailed previously in this thesis have already demonstrated the extent to which the sequencing platform can alter estimated telomere length (see Sections 3.12 and 4.2). Accordingly, we were unsurprised by plot shown in Figure 5.1A that shows a clear differentiation between samples from the Illumina HiSeq2000 and HiSeqX platforms.

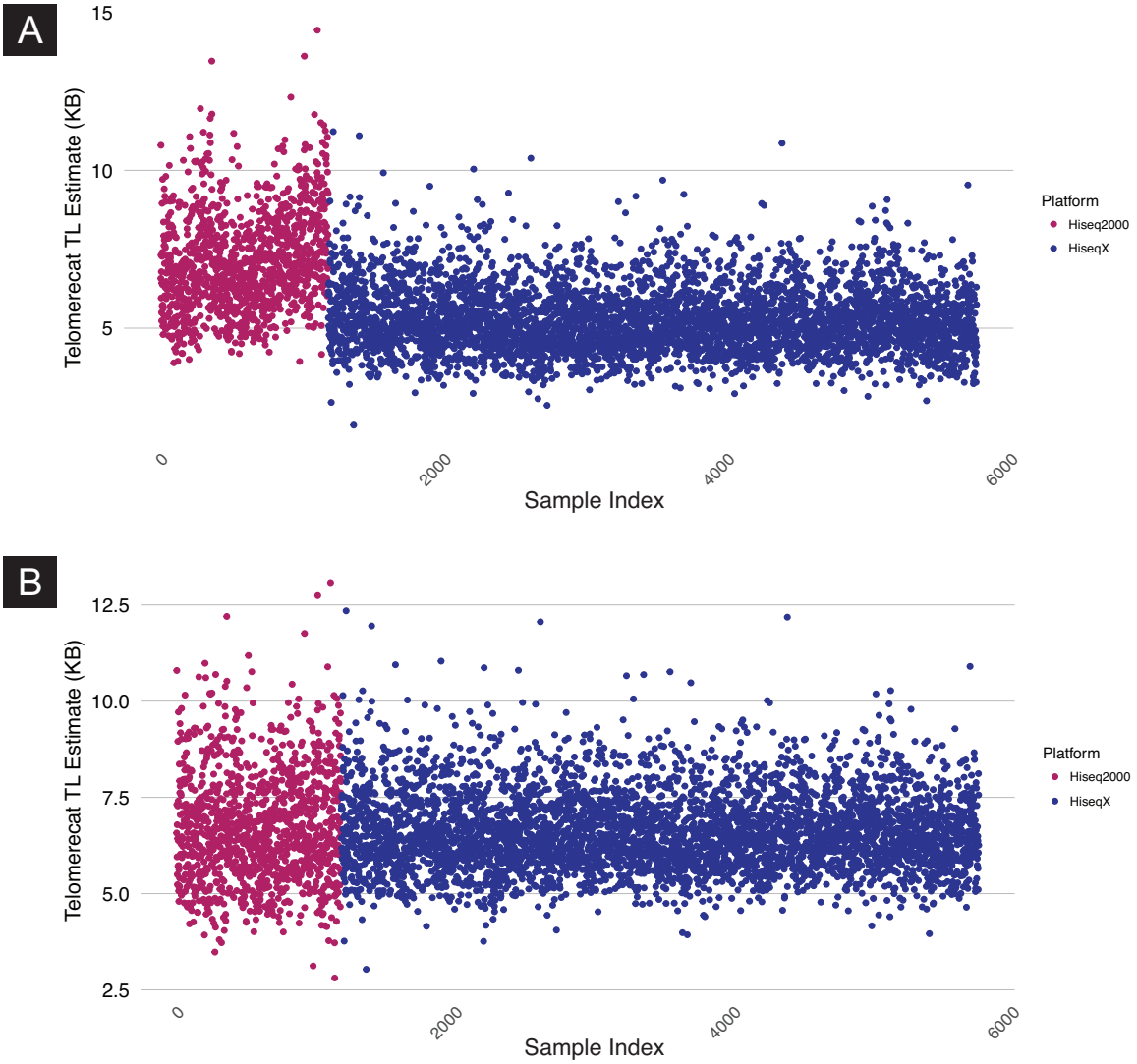


Fig. 5.1 Plots showing all of the samples in the NIHR BioResource WGS cohort ***(A)****: Pre-normalisation* ***(B)****: Post-normalisation*

To handle this batch effect we employed a method similar to the one detailed in Section 4.2. However, the larger cohort size in this study means that we can afford greater granularity in our normalisation process. Rather than use sequencing batch as the dummy variable, we used the sequencing plate ID. Here, plate ID refers to a unique ID assigned to the Illumina TruSeq plate used as part of the sample preparation. All samples with the same plate ID were prepared on the same plate.

Across the entire cohort there were 75 unique plate IDs. The mean number of samples associated with each plate ID was 76, the plate ID with most associated samples had 96 samples whilst the plate ID with fewest samples had 19. Plate ID was provided to the normalisation along with patient age so as not to bias the batch normalisation by age. Due to the way samples were collected for the studies (some subcohorts are comprised primarily of juvenile or geriatric donors), there was a high chance that samples would not be randomly distributed in age.

The coefficient for each plate was subtracted from the relevant samples and we were left with the distribution shown in Figure 5.1B. At this juncture we did not adjust the length estimates for age as we wished to observe how age was associated with telomere length across the cohort.

5.2.2 Associations with potential confounders of telomere length

Once we had minimised batch effects within the data we investigated the association between telomere length and known confounders within the cohort.

Age We observe a strong correlation with age across the cohort. Figure 5.2A shows a scatter plot of age against telomere length demonstrating the extent of the association. A linear model with telomere length as a response variable and age as a predictor reported $R^2 = .31$ (equivalent to a Pearson's correlation of $r = -0.56$) and found that each year of additional age was responsible for a deterioration of 33bp. This negative correlation between age and telomere length is well established within the literature. However, we observe a particularly high correlation in comparison to other studies using different methods (WEISCHER *et al.*, 2014; SONG *et al.*, 2010; BENETOS *et al.*, 2001). This speaks well of the accuracy of the method and the quality of the sequencing data.

The high correlation may also be the result of non-linearity of the relationship across age. Figure 5.2A also shows the line of best fit provided by a local regression. We noticed that the slope of the fit shows a steeper decline (indicating increased association between age and telomere length) in earlier ages. It is interesting to speculate that this increased

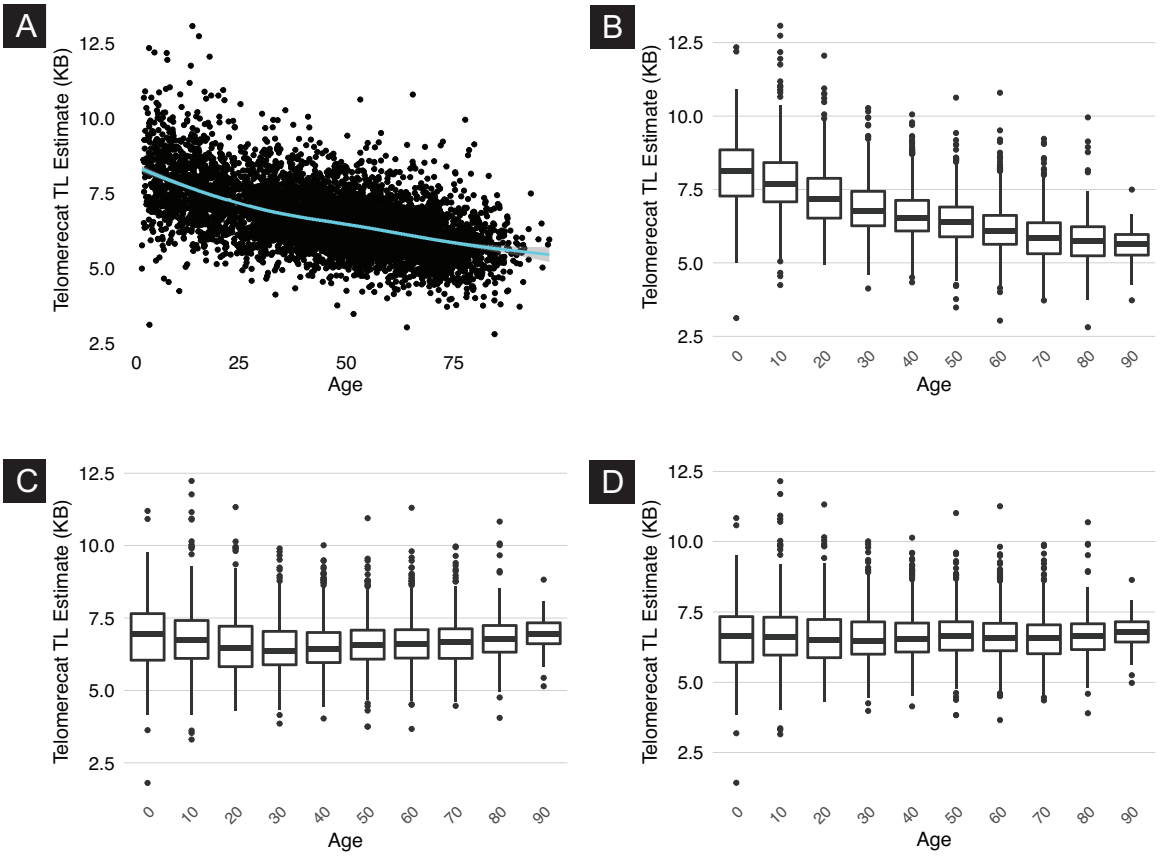


Fig. 5.2 The relationship between age and telomere length in the NIHR BioResource cohort ***(A):*** *Age versus telomere length, line of best fit provided by loess (local regression).* ***(B):*** *The data subdivided by age groupings.* ***(C):*** *Length by age grouping when we used a linear model to adjust for age. Note that there is still a bias towards younger samples having longer telomeres* ***(D):*** *Length by age grouping after using a cubic model to adjust for age. There is no observable bias across the age groupings*

association in samples from younger donors is the result of fewer mitigating environmental effects experienced by younger donors.

Gender We observed that telomere length was shorter in men across all but one age category (see Figure 5.3A). We see in Figure 5.3B that when we controlled for age telomere length was significantly differentiated according to gender. Our analysis indicates that being male accounts for a -217bp deficit in telomere length. A difference roughly equivalent to six and a half years of life.

A recent comprehensive meta analysis, detailed in GARDNER *et al.* (2014), found that most studies identify females as having longer telomere than males and that this was the case across age groups. However, GARDNER *et al.* found that the difference in telomere

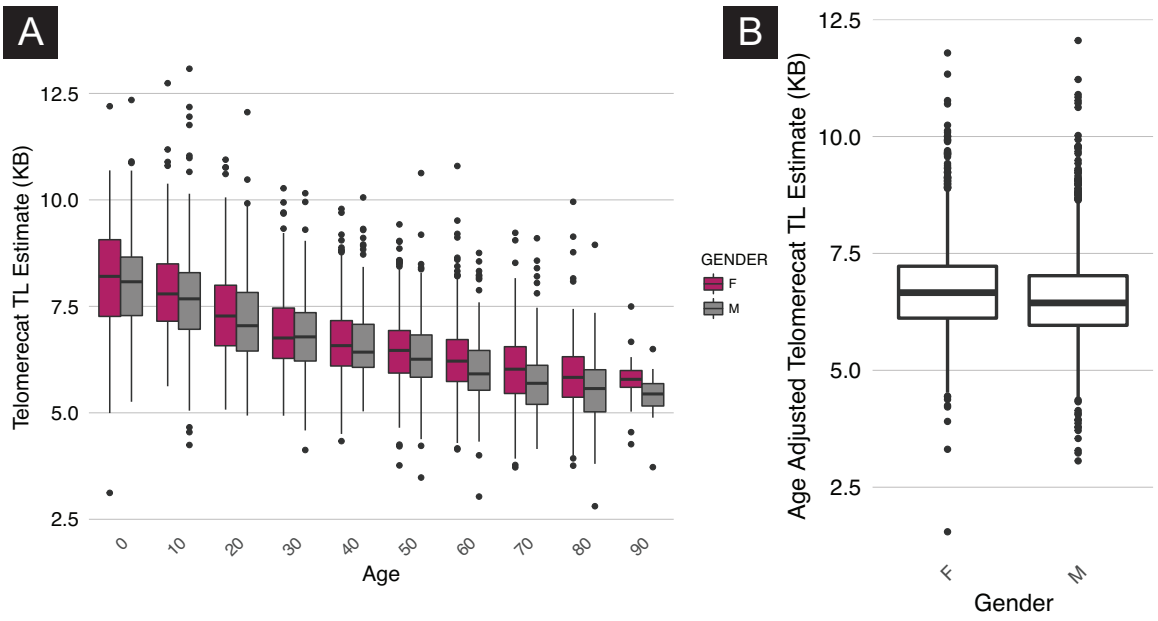


Fig. 5.3 Plots highlighting the difference in telomere length between genders ***(A):*** *Boxplots of telomere length by age group separated by donor gender. We observe shorter male telomere length in almost all of the age groups* ***(B):*** *A boxplot showing age adjusted telomere length by gender. Mean male telomere length is significantly shorter ($p < 4.8 \cdot 10^{-15}$; Two sided T Test)*

length between males and females was more often identified by mTRF than by either qPCR and flowFISH. GARDNER *et al.* conclude that this difference is not attributable to random sampling error but rather is likely due to inherent bias in the methodologies. The results presented here indicate that, at least within this cohort, Telomerecat reliably identifies a relationship between telomere length and gender. The strength of our observation seems to lend weight to the hypothesis that there is a genuine and measurable difference between male and female telomere lengths.

This study represents the largest application of a WGS telomere length estimation method where age and gender meta-data was available. Accordingly it sheds considerable light on the ability of these methods to detect these known relationships within the data. Conversely, accepting that Telomerecat is a reliable method for estimating telomere length, these results provide insight into the ongoing debate as to whether the difference in telomeres between genders is a biological or technical effect.

5.2.3 Adjusting for age and gender

The identification of the genetic basis of inherited disease is dependent on the correlation of a defined phenotype with an inherited genotype. For our purposes this phenotype is the telomere length as estimated by Telomerecat.

Once we had observed the extent to which Telomerecat identified an association between gender and age in this cohort, we wished to normalise the telomere lengths so that we could more easily compare samples from disparate age and gender groups. To do this we fit the following linear model to the data using age as a continuous variable and gender as a dummy variable. We refer to this as "the cubic model":

$$\text{length} = \beta_0 + \beta_1 age + \beta_2 age^2 + \beta_3 age^3 + \beta_4 \text{gender} + \varepsilon \tag{5.1}$$

Previous attempts to fit a more straightforward linear model to the data using single variables for age and gender showed that the residual versus age was not linear. The effects of this poor fit can be seen in Figure 5.2C where there is a clear bias when we plot the adjusted length to the relevant age groups. When we added a squared and cubed age term to the model we found that the residuals were randomly distributed and accordingly the adjusted lengths showed less bias (see Figure 5.2D).

To finish, we subtracted the relevant residuals, as produced by the cubic model, from the mean telomere length of the cohort. We used these as our adjusted telomere lengths in further analyses. The resultant adjusted telomere lengths can be seen plotted against age and gender in the Figure 5.4.

5.3 Results

After preparing the data to control for sequencing batch effects, age and gender we analysed the samples in the hope of shedding light on telomere length in rare disease.

5.3.1 GWAS

We started our investigation by conducting a genome wide association study (GWAS) with the data. The primary purpose of this analysis was to serve as further orthogonal validation of the Telomerecat method. Previous studies, using larger cohort sizes, have already identified numerous significantly associated regions with telomere length (CODD *et al.*, 2013; POOLEY *et al.*, 2013).

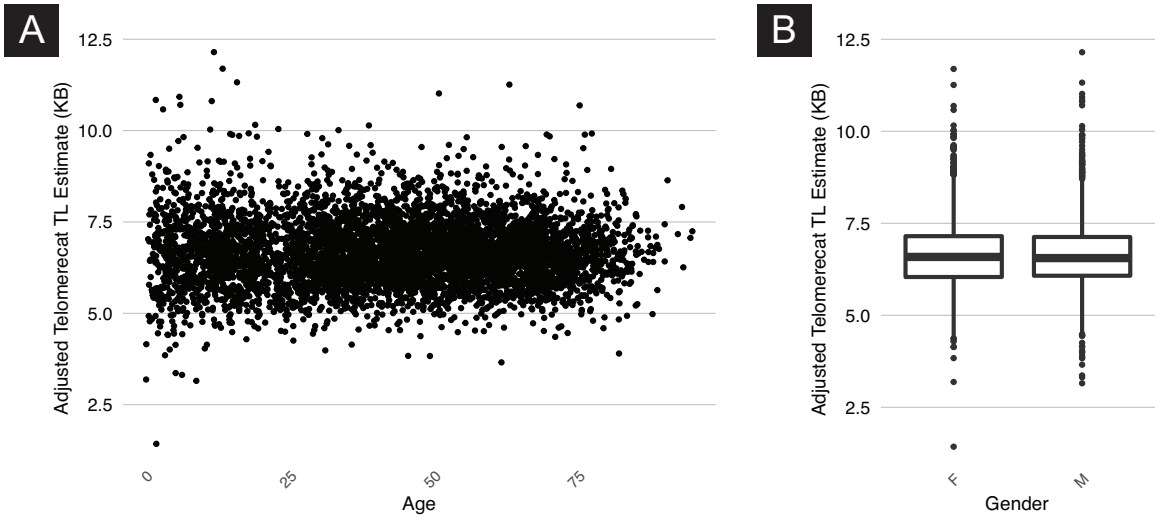


Fig. 5.4 Plots of telomere measurement after adjustment for age and gender

The purpose of a GWAS is to identify polygenic risk alleles that correlate with a defined phenotype. Success in a GWAS is dependent on accuracy of the phenotype measured and the size of the cohorts studied. The CODD *et al.* GWAS was carried out with a discovery cohort of 37,684 individuals with the telomere phenotype defined by qPCR. Similarly, the study conducted by POOLEY *et al.* used qPCR and was conducted on 26,089 donors, a mixture of healthy controls and cancer cases.

Our telomere length GWAS, with approximately one seventh the cohort size, is substantially underpowered to detect the genetic associations identified in these prior studies. However, identification of the same effect in terms of shortening or increasing telomere length by the previously identified loci would provide additional validation that our measure of phenotype was accurate

Table 5.2 compares our results to those of CODD *et al.* and Figure 5.5 shows the resultant Manhattan plot.

Despite the comparatively diminutive sample size we managed to confirm the strongest association highlighted by CODD *et al.* and POOLEY *et al.* in the form of a locus in the region of the TERC gene. Furthermore, we observed that for the other significant hits obtained by CODD *et al.* our estimated β coefficients show the same direction of effect in each case. Whilst we could not observe genome wide significance amongst all of the variants in Table 5.2, it is reassuring that where CODD *et al.* observe a significant association we at least agree with the direction of the effect.

Table 5.2 A table showing results captured by our GWAS in comparison to the result from CODD et al. *(2013). We have confirmed that the direction of effect is the same in each case. This may not be clear on first appraisal because of the way that variants are reported by each GWAS.* CODD et al. *reports the effect allele as whichever corresponds to shorter telomere length. Our pipeline reports variants based on a major and minor variant paradigm. B is always the minor allele and so for the first three entries we report the result in terms of this allele and so give a positive β*

Variant Information				CODD *et al.* results				Our results			
ID	Chrom	Position	Gene	Effect	Alternative	β	p	Major	Minor	β	p
rs10936599	3	170974795	TERC	T	C	-0.097	$2.54 \cdot 10^{-31}$	T	C	0.147	$1.68 \cdot 10^{-10}$
rs2736100	5	1339516	TERT	A	C	-0.078	$4.38 \cdot 10^{-19}$	A	C	0.052	$8.37 \cdot 10^{-3}$
rs7675998	4	164227270	NAF1	A	G	-0.074	$4.35 \cdot 10^{-16}$	A	G	0.071	$3.04 \cdot 10^{-3}$
rs9420907	10	105666455	OBFC1	A	C	-0.069	$6.90 \cdot 10^{-11}$	C	A	-0.056	$3.82 \cdot 10^{-2}$
rs8105767	19	22007281	ZNF208	A	G	-0.048	$1.11 \cdot 10^{-9}$	G	A	-0.037	$8.68 \cdot 10^{-2}$
rs755017	20	61892066	RTEL1	A	G	-0.062	$6.71 \cdot 10^{-9}$	G	A	-0.025	$3.87 \cdot 10^{-1}$
rs11125529	2	54329370	ACYP2	C	A	-0.056	$4.48 \cdot 10^{-8}$	A	C	-0.081	$4.99 \cdot 10^{-3}$
rs2967374	16	80767362	MPHOSPH6	G	A	-0.045	$2.70 \cdot 10^{-7}$	A	G	-0.088	$2.26 \cdot 10^{-4}$

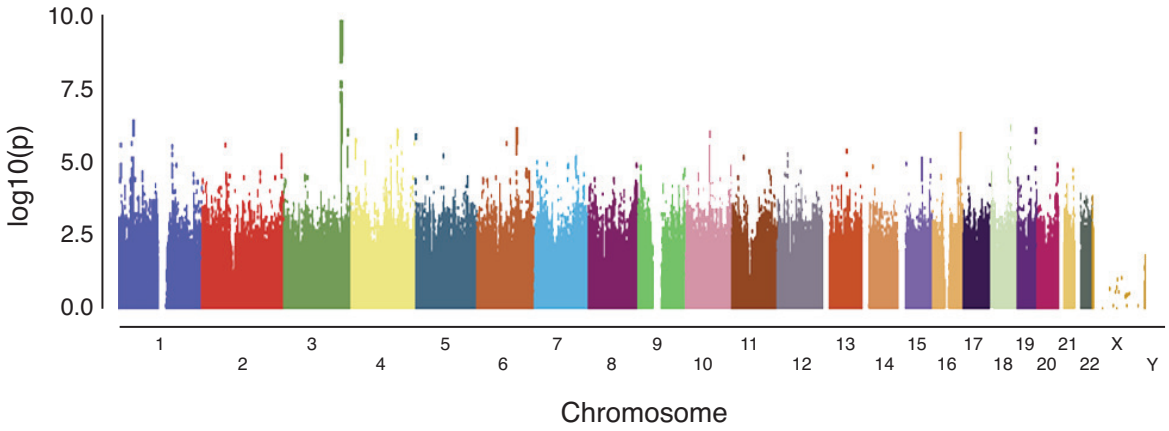


Fig. 5.5 A Manhattan plot showing the results of the GWAS on unrelated samples within the NIHR BioResource cohort. The only result reaching genome wide significance ($p < 10^{-8}$) is the variant in TERC on chromosome 3

5.3.2 Telomere length across the sub-cohorts

After confirming known genotype associations within the wider cohort we wished to examine whether any of the sub-cohorts demonstrated differential telomere lengths according to affected status.

Figure 5.6 shows boxplots for each subcohort separated by affected status. To test whether any of these relationships were statistically significant we applied a Student T test to each of the subcohorts, with affected status separating each cohort. The only cohort with significantly differentiated telomere lengths was the SMD subchort ($p < 0.01$, adjusted for multiple hypothesis testing with FDR). The SMD cohort contains patients with primary and secondary bone-marrow failure. This is associated with primary and secondary telomere defects and the results provide additional supportive evidence that Telomerecat can identify shorter telomeres in patient populations associated with reduced telomere length.

5.3.3 Rare variants in telomere genes within the PID sub-cohort

Deleterious mutations in the genes associated with telomere maintenance have been identified as causing rare monogenic disease. This led us to question whether this diagnosis could account for disease in any of the undiagnosed patients recruited into the NIHR-BioResource.

The initial clinical description of monogenic telomerase dysfunction, dyskeratosis congenita, is described as a condition characterised by the triad of nail dystrophy, lacy reticular discolouration of the neck and oral leukoplakia. More in-depth phenotyping of these patients identified that all patients were at risk of bone-marrow failure. Further study has also iden-

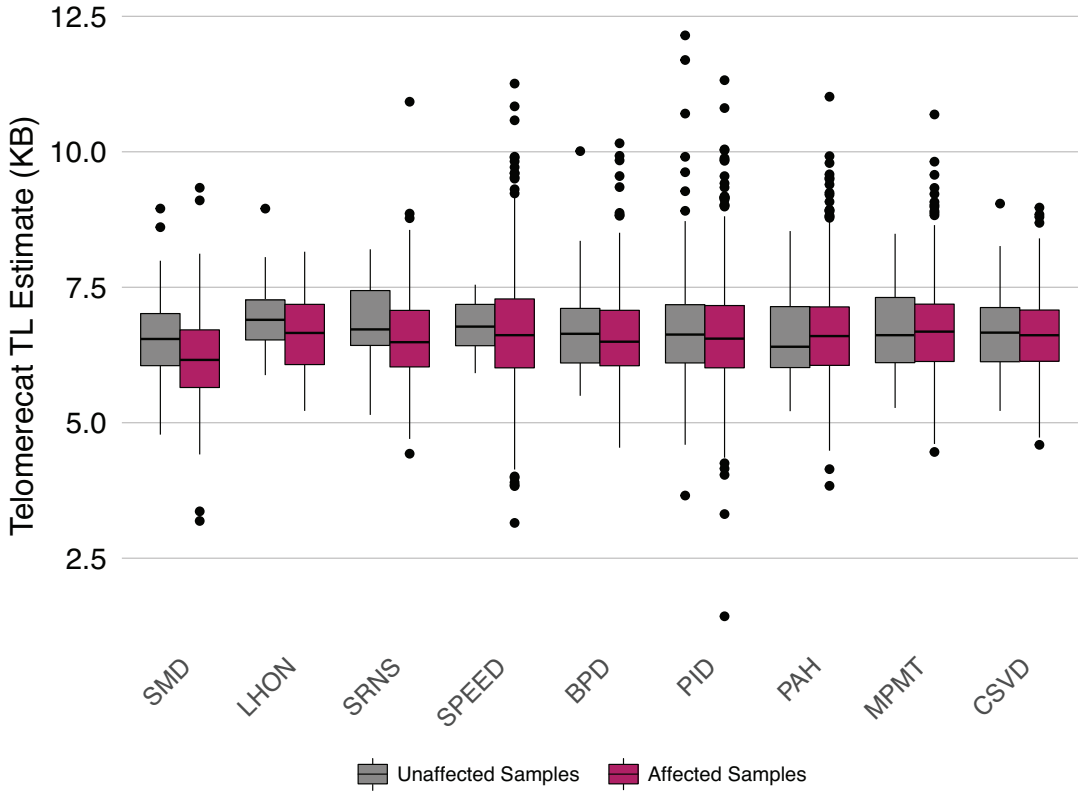


Fig. 5.6 Telomere length by cohort affiliation, separated by affected status. Only sub-cohorts for which unaffected telomere length estimates were available are included in this plot

tified that a severe variant of dyskeratosis congenita, the Hoyeraal-Hreidarsson syndrome, features immune deficiency (GLOUSKER *et al.*, 2015).

The PID is of particular interest as many of the cases in the subchort display an immunodeficient phenotype with no known genetic cause. Furthermore, it is known that primary and secondary telomerase dysfunction can lead to short telomeres in bone-marrow failure patients

We wished to see whether any of the samples that display reduced telomere length had evidence of known genetic diseases caused by telomerase deficiency. We prioritised the cases within the PID cohort that had exceedingly short telomeres. In order to achieve this, we first identified samples in the PID cohort with an adjusted telomere length that was below the 10th centile of estimates for the entire NIHR BioResource cohort. We identified 105 such samples in the PID cohort.

Then, for each of these short telomere samples we extracted rare coding variants in a number of genes previously identified as causing dyskeratosis congenita (TUMMALA *et al.*,

Table 5.3 Genes linked to dyskeratosis congenita as identified in TUMMALA et al. *(2015). Reference locations according to the GRCh37 version of the human genome*

	Name	Chromsome	Start	End
1	TERC	3	169482397	169482847
2	TERT	5	1253287	1295162
3	NHP2	5	177576465	177580961
4	TINF2	14	24708851	24711880
5	NOP10	15	34633917	34635362
6	PARN	16	14529557	14724128
7	ACD	16	67691414	67694717
8	WRAP53	17	7589389	7606820
9	CTC1	17	8128139	8151413
10	RTEL1	20	62289163	62328544
11	DKC1	X	153991016	154005963

2015). The genes sampled and their genomic locations are given in Table 5.3. A genotype was considered "rare" if it was estimated to occur in less than one in a thousand genomes. Since synonymous mutations have not been reported to cause telomerase dysfunction, these variants were excluded.

Of the original 105 short telomere samples we identified 18 with a rare coding variant in one of the aforementioned target genes. These 18 variants are shown in Table 5.4. We see that, of the identified variants, most are heterozygous.

Only one of the identified variants is homozygous, a missense mutation in the first exon of the TERT gene (entry one in Table 5.4). The sample with this mutation displays critically short telomeres (in the bottom 0.001 percentile) and displays a typical Hoyeraal-Hreidarsson phenotype. Unfortunately, this donor died as a direct consequence of their condition and associated immune system failure. The severity of this phenotype is highlighted in Figure 5.7 where we see that this sample (labelled F003462) is the shortest of all the measured telomere lengths. These findings demonstrate the ability of Telomerecat to identify the extremely short telomere lengths of patients with monogenic telomerase deficiency.

Before the outset of this study, F003462 was already known to exhibit a Hoyeraal-Hreidarsson phenotype and a previous study identified the sample as having extremely short telomeres. While it is reassuring that this sample was identified by this analysis, it cannot be considered a novel finding.

Two of the entries in Table 5.4 were hemizygous (the variant existed upon the X chromosome of which both male donors only have one copy). Interestingly, these samples were

Table 5.4 The variant identified by our search for rare variants across donors with short telomere in the NIHR BioResource cohort. All of the variants are missense mutations apart from the heterozygous TINF2 mutation which is a splice variant

	Sample	Gene	Chromsome	Position	Zygosity
1	F003462	TERT	5	1293620	Homozygous
2	F005482	DKC1	X	154004469	Hemizygous
3	F005484	DKC1	X	154004469	Hemizygous
4	F005990	TERT	5	1293799	Heterozygous
5	F005989	TERT	5	1293799	Heterozygous
6	F008867	NHP2	5	177576825	Heterozygous
7	F006467	TINF2	14	24709624	Heterozygous
8	F010434	PARN	16	14540824	Heterozygous
9	F011418	ACD	16	67692018	Heterozygous
10	F009265	ACD	16	67693143	Heterozygous
11	F000954	CTC1	17	8133942	Heterozygous
12	F012660	CTC1	17	8135253	Heterozygous
13	F012560	RTEL1	20	62293790	Heterozygous
14	F004704	RTEL1	20	62293798	Heterozygous
15	F009265	RTEL1	20	62309633	Heterozygous
16	F007172	RTEL1	20	62321444	Heterozygous
17	F008048	RTEL1	20	62321559	Heterozygous
18	F009718	RTEL1	20	62326159	Heterozygous

derived from two brothers, both of whom have experienced classic symptoms of compromised immune systems at a relatively advanced age.

5.3.4 Examining the link between DKC1 variants and short telomeres

These brothers had suffered from the viral eye infection CMV retinitis, and the lung infection *Pneumocystis jirovecii*. These infections are ultra rare in the normal population, the incidence of both dramatically increased with the AIDS epidemic. Both infections are classified as AIDS defining illness, as these infections denote when a patient with HIV infection becomes sufficiently CD4+ T cell immunodeficient that they fulfil the diagnostic criteria for the Aquired Immunodeficiency Syndrome. Both brothers had been tested for HIV infection and had confirmed negative test results. This result suggested that the brothers had a severe primary CD4+ T cell immunodeficiency. The X-linked nature of these variants as well as the fraternal relationship of the two donors made the hemizygous missense mutation in DKC1 an interesting topic of further study.

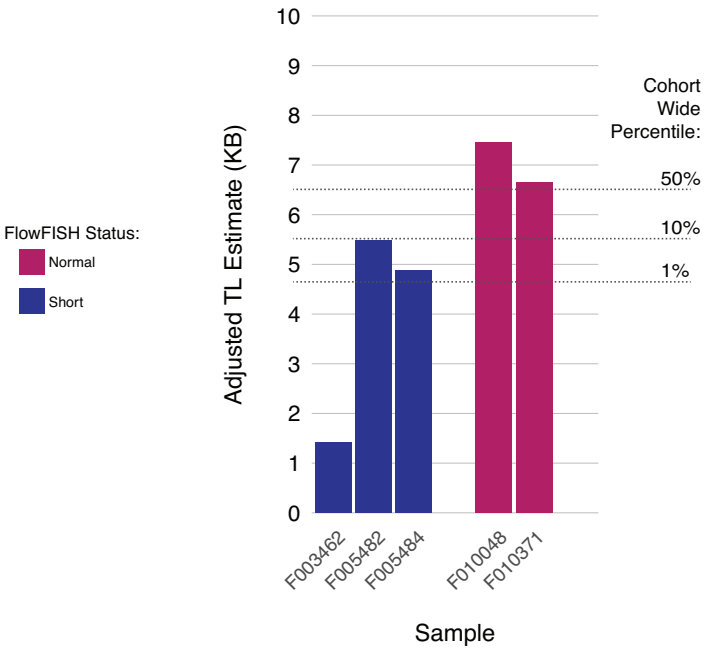


Fig. 5.7 A bar chart of samples subjected to flowFISH analysis. The samples coloured blue were identified as displaying telomere length below the first percentile by flowFISH. Samples coloured red were shown by flowFISH to have normal telomere lengths, (between the 25th and 75th percentile. The dotted lines show the length corresponding to the annotated percentile. Percentile shown is of adjusted length for the entire NIHR BioResource Cohort

Since both brothers appear to have a fully penetrant genetic variant that is rare and shared, we assessed the data so as to exclude other candidate variants. By filtering for rare genomic variants that were shared in the brothers we identified 23 variants (Table 5.5). Of these only the DKC1 variants and the MAP1B variant are private to the two brothers. We assessed all of these genes for expression in the nucleated populations that make up blood by using the BLUEPRINT expression atlas (CHEN *et al.*, 2016) DKC1 is expressed in all blood populations whilst MAP1B is restricted to mesenchymal stem cells and endothelial tissue.

Before proceeding we confirmed that Telomerecat had correctly identified shortened telomeres in the samples with heterozygous and hemizygous variants in DKC1 and TERT. As we see in Figure 5.7 separate orthogonal analysis by flowFISH indicates that these samples do indeed exhibit shortened telomeres. FlowFISH identifies both brothers as having telomere lengths in the bottom one percentile. Telomerecat estimates that both the samples fall between the 10th and 1st percentile of all age adjusted telomere lengths. When we consider the unadjusted telomere lengths we see that F005484 is below the first percentile and F005482 is at around the 7th percentile. FlowFISH and Telomerecat both see that the brothers possess short telomeres, however, flowFISH provides shorter estimates than Telomerecat.

DCK1 is a highly conserved gene that serves as a regulator of telomerase through interaction with ribonucleic portion of the protein (VULLIAMY *et al.*, 2001; COLLINS, 2000). DCK1 was proposed as a causative gene for dyskeratosis congenita before DCK1 was known to associate with telomere and even before dyskeratosis congenita was linked to shortened telomeres (HEISS *et al.*, 1998).

The missense variant we have identified in donors F005482 and F005484 is located in the penultimate exon of the DCK1 gene. The G-to-A variant causes an Argenine to Glutamine transformation. This variant is private to this pedigree, by assessment of all available publicly accessible databases of human genetic variation and has not been observed in any of the large scale sequencing cohorts commonly used to ascertain rarity. We have also confirmed that this variant is private to this pedigree within the rest of the NIHR BioResource cohort.

The brothers present with an a-typical dyskeratosis congenita phenotype, most noticeable because of their advanced age, as detailed at the beginning of this section. The brothers' dyskerin[1] levels are now being measured. We expect these results will confirm a-typical dyskerin activity across the affected brothers and prove conclusively the penetrance of this variant.

We think it is likely that this variant in DKC1 is driving the immunodeficient phenotype as CD4+ T-cells are failing to forgo replicate as a result of severely shortened telomeres. It

[1] the protein encoded by DKC1

Table 5.5 A table showing variants shared by the DCK1 variant brothers

Inheritance	Gene	Position	Type	Observed Frequency
XL	DKC1	X:154004469	missense	.
AD	MAP1B	5:71491412	missense	.
XL	ZNF75D	X:134421668	nonsense	0.00106
CH	ABCA13	7:48232580	missense	0.00452
CH	ABCA13	7:48284168	intron substitution	0.00752
CH	EXOC3L1	16:67221737	missense	0.001
CH	EXOC3L1	16:67218391	missense	0.00644
AD	IPP	1:46206577	missense	0.00019
AD	PDE8B	5:76627270	missense	0.00003
AD	DMBX1	1:46978039	missense	0.00028
AD	MYCBP	1:39338760	synonymous	0.00049
AD	CLIP3	19:36510207	missense	0.00022
AD	PI4KA	22:21066785	missense	0.00005
AD	LIMK1	7:73535337	missense	0.00027
AD	FOXJ3	1:42657223	missense	0.00045
AD	SUOX	12:56396504	missense	0.00076
AD	CALCOCO1	12:54110127	missense	0.0007
AD	DNAH11	7:21658833	missense	0.00013
AD	GPI	19:34890208	missense	0.00094
AD	FGL2	7:76828923	missense	0.00008
AD	MAP3K10	19:40720883	missense	0.00321
AD	LRRC8D	1:90400863	missense	0.00002
AD	HECW1	7:43483897	missense	0.00052

stands to reason that the highly proliferative nature of the T-cells would mean that the effects of short telomere would strike there preferentially.

5.4 Discussion

This analysis is a prime example of using Telomerecat as a diagnostic first sweep. The inexpensive nature of using pre-existing WGS data to extract phenotype information combines well with the efficient and fast running nature of Telomerecat and Parabam.

We have seen how Telomerecat identifies strong associations with age and gender as well as confirming previous GWAS results. These serve as good indications of Telomerecat's ability to estimate telomere. The key story in this chapter, however, is the discovery of a hitherto unknown mutation in DKC1 and its effect on the two brothers.

We have seen how an extremely rare mutation in the highly conserved DKC1 is associated with shortened telomere length in both brothers. Other shared variants seem not to be localised in known telomere encoding genes or are not private to the pedigree. Given DKC1s known highly conserved status and relationship to telomere length it seems extremely likely this is a causal variant. The last piece of the puzzle, an *in vivo* assay of dyskerin activity across cell types, will hopefully prove this.

Other members of the DKC1 variant brother's pedigree were sequenced as part of the PID study. Interestingly, the telomere lengths of the children of both brothers were substantially below the cohort-wide average. This is despite the fact that both the mothers of these children displayed normal telomere lengths. This is particularly interesting in the case of the sole male progeny of F005484 whom we can guarantee as not having inherited the DKC1 variant: he must have inherited his X chromosome from his unaffected mother. These observations have implications for an "epigenetic" mode of telomere length inheritance. In support of this hypothesis, previous studies have shown that the father's telomere length has greater bearing than the mother's in influencing inherited telomere length (NORDFJALL *et al.*, 2005).

This chapter is a collaborative effort, leveraging the expertise of clinicians directly involved in the NIHR BioResource study. As well as highlighting the utility of Telomerecat, it highlights the strength afforded when collaborators of different disciplines combine to a common goal. This chapter represents exciting new territory for Telomerecat in driving new understandings within the clinic.

Chapter 6

Conclusions

Computational biology is an inherently interdisciplinary field. This variety is reflected by the diversity of topics covered in this thesis. The thesis is broad in scope and attempts to encompass explorations in software design, statistics and biology. While the tools described in this thesis are perhaps the most significant contribution, we hope that we have demonstrated the ability of these tools to uncover new knowledge in the field of biology.

Some future work in improving and applying Telomerecat

As part of this thesis we have detailed two applications of the Telomerecat tool to separate data sets. In both cases Telomerecat identified novel results. We see that the best use case for Telomerecat, and other tools like it, is in conducting a broad first sweep of the data in order to identify possible sources that warrant further study. A good example of this approach is demonstrated in Chapter 5 where we identified a novel association with a previously uncharacterised high penetrate mutation in DKC1, which was later verified with the FlowFISH method. As a result of this finding, the two brothers now know more about their condition. This is as direct a clinical result as a computational method could hope to achieve. Furthermore, we are hopeful that the associations highlighted in Chapter 4, particularly associations between telomere length and molecular subtypes, can go on to inform further work in PCa.

Telomere length estimation from WGS provides an extremely cheap and efficient way of estimating telomere length when WGS data has already been generated. These tools will be particularly advantageous on the next generation of large cohort studies like the 100,000 Genomes project run by Genomics England. A study of this size will provide unprecedented power to detect genotypic and phenotypic associations with telomere length the likes of which are as yet unseen. The fact that telomere length can be discerned from these samples

without the cost of additional experiments adds real value to the state of telomere biology in general.

Telomerecat will be particularly well suited to these applications because of the speed with which it can generate telomere length estimates. We hope that the method will also be held in high regard for its adaptability in being able to natively account for ITSs and its agnostic approach to ploidy. Furthermore, Telomerecat will have additional usage as currently the only tool that works on non-human WGS samples.

In Chapter 3 we demonstrated Telomerecat's ability to correctly identify telomere length across a range of validation experiments. Furthermore, in Chapter 5 we saw that Telomerecat agreed with FlowFish and showed agreement with previous GWAS results. Additionally, each time it has been applied in this thesis it has observed a moderate correlation with age, as expected. These factors considered together provide very strong indication that Telomerecat is capable of estimating telomere length from WGS samples with a useful degree of accuracy.

That being said there are numerous ways in which Telomerecat might be improved. Foremost amongst these is the way that Telomerecat deals with genuine telomere heterogeneity. We saw in Chapter 3 that Telomerecat performs poorly if we apply too strict a threshold for inclusion as a true telomere read (even when accounting for sequencing error). We see the most likely reason for this is that there is genuine heterogeneity in the sequence of telomere.

An improved approach to modelling telomere heterogeneity? One way to deal with this heterogeneity, that future versions of Telomerecat may employ, is to model the phenomenon more closely. Figure 3.18 in Chapter 3 shows a histogram of all reads binned by the amount of mismatching loci in each read. We propose that this is a mixture of three different distributions. The left of the plot shows reads with few mismatches, these are likely to be genuine telomere reads and look as though they may be defined by an exponential distribution. In the middle, the distribution becomes flat, perhaps this is the contribution of subtelomere reads and could be described by a uniform distribution. To the right most extreme of the figure are reads with high mismatches, these are unlikely to be telomere reads, perhaps modelled by the normal distribution. We then might be able to fit a combined mixture model to these distributions such that we could predict the amount of reads contributed by the distribution representing full telomere.

This method has the advantage of obtaining an estimate for the reads in the sample without enforcing a hard coded threshold. This approach could even shed light on how far telomere heterogeneity differs between samples. It would be interesting to observe how the distribution fit to the complete reads differed between samples.

Considering different types of telomere sequence Currently, Telomerecat only works on samples where the telomeres consist of the TTAGGG canonic repeat. The TTAGGG hexamer is found in all vertebrates, however other organisms use a different repeat. By allowing the user to specify their own hexamer of choice we could widen the applicability of the program. In these cases a careful examination would need to be made of how other parts of the program were affected. For example, certain strains of yeast are known to display extremely heterogeneous telomeres and it is not clear how this extreme heterogeneity would affect the ITS filtering system. We would need to be precise in how we defined complete telomere reads so as not to create an influx of false positives.

A related problem is the consideration of variable telomere repeats. The TelomereHunter program already incorporates a consideration of G-type, C-type and J-type variant telomere repeats. These variant repeats are often found in telomeres having undergone ALT but are hypothesised to be interspersed within healthy telomere as well (VARLEY *et al.*, 2002). While Telomerecat does not directly account for variant telomere repeats some reads containing variant repeats will naturally be included as fully telomeric. Telomerecat allows for a maximum of 10% disagreement from the telomere sequence after accounting for sequence error.

We have briefly tested a prototype version of Telomerecat that accepts occurrences of the variant telomere repeat and found that the results did not differ significantly from the existing method. Although, admittedly these tests were not conducted on samples that had undergone ALT. Furthermore, we realised that in hard coding for human variant repeats we could be further limiting the applicability of Telomerecat to human only samples.

Estimating telomere length in minION and 10X sequencing Further to improvements to the existing algorithm, an area of future work could adapting Telomerecat to work on new types of sequencing technologies. A number of new sequencing technology present interesting opportunities for deriving even more information regarding telomere length. Foremost amongst these new technologies are minION and 10X.

minION, developed and distributed by Oxford Nanopore Technologies (LU *et al.*, 2016), is an "ultra long-read" technology that can produce fragment lengths of hundred of KB. As telomere in humans are routinely not longer than 15KB this brings the telomere firmly into view as entirely sequenceable. It stands to reason that it would be possible to sequence an entire telomere along with enough subtelomere sequence for the read to be aligned to a specific chromosome. Estimating the length of telomeres on individual chromosomes has so far been the preserve of imaging based techniques like qFISH and so capturing this information with a sequencing approach would be desirable.

In a similar vein the GemCode platform distributed by 10X could be used to provide information about individual telomeres using WGS data (EISENSTEIN, 2015). GemCode is a library preparation that adds a barcode to all of the reads from the same molecule. This library preparation technique can be used in conjunction with the Illumina sequencing platform so can be accommodate by existing technologies.

It might be possible to directly apply Telomerecat to these data. We could simply apply the algorithm to sets of reads associated to the barcodes found in the sample. This relies on there being high enough sequencing depth so that there were observable reads over each boundary in the sample. Allowing for this constraint, we could plausibly provide a distribution of telomere lengths throughout the sample.

A potential application for Parabam

In Chapter 2 we detailed our method for processing BAM files in parallel. Parabam is the first tool to allow users to apply a function to each read in the BAM file using higher order language features like conditional statements and loops. In this section we discuss potential applications for Parabam.

Through the development of Telomerecat and in preparing this thesis Parabam has proven useful as a tool of general application. For example, it was used multiple times to extract pertinent data for figures, for instance Figure 3.15 where we needed to extract mapping data for each read in a TELBAM file and all reads mapped to chromosome 14. Another interesting supplementary use of Parabam was in creating the random subset of the file used to benchmark the software in Section 2.5 (the user-rule to facilitate this is shown in Appendix A).

PolyQ tracts in the AR gene Parabam lends itself particularly well to analyses that require the entire BAM file to be processed. Firstly, because of speed with which Parabam can conduct complex analyses. Furthermore, because the user-rule based paradigm is well suited to analysing unmapped reads that are otherwise difficult to summon from the BAM file.

One possible application, with special relevance to PCa, is the analysis of a polyglutamine expansion (polyQ) in the AR gene. PolyQs are repetitive stretches of the genome found within coding regions of the genome[1]. PolyQ expansion has been identified as the cause of Huntington's disease (WALKER, 2007). In Huntington's disease an unreliable mechanism of DNA replication of the PolyQ sequence causes expansion in the gene over progressive generations.

[1] so called for their constituent sequence "CAG" that codes for the glutamine amino acid, whose symbol is "Q"

In contrast to Huntington's disease where long polyQ tracts are associated with the disease phenotype, a recent meta-analysis suggested that shorter polyQ tracts in the AR gene were a risk factor for PCa (QIN *et al.*, 2017). Clearly this is not the final word on the matter though, as previous studies have suggested an association between longer polyQ tracts and reduced time to progression in prostate cancer (POWELL, 2011).

Parabam could be used to investigate PolQ tracts in AR. A similar approach to the one utilised for the study of telomere reads could be applied to PolyQ tracts. We could create a subset of reads and their pairs which have been observed to contain the PolyQ repeat sequence and then conduct further analysis into the maximum observed length of repeats. Our experience in dealing with repetitive sequencing reads as part of the Telomerecat method may have particular pertinence to this problem as when assessing the length of a tract an error in the sequence can shorten the observed tract length, skewing the analysis.

Most of the studies in this field have observed the risk of AR polyQ tract length in the germline. An application of this tool to the Prostate ICGC cohort could be used to see how the tract length differed between normal and tumour samples from the same donor.

Software design and practices in bioinformatics

Chapter 2 is software orientated and focuses on the design and implementation of the Parabam algorithm. We hope that in Chapter 2 we demonstrated the importance of good design in bioinformatics software. So often in the field of computational biology, concern with the actual design of software comes as a distant second to the primary concern of methodology.

It is certainly important that the methodology of these tools is sound; the tools must provide accurate and useful information. However, a lack of focus on best practices in software design leads to a multitude of frustrations. It is our view that software engineers, distinct even from computer scientists, must stake a claim for importance in the field by impressing the need for coding best practices.

The computational advances over the past few years have caused a huge proliferation in data. This proliferation has only emphasised the need for standardisation of working practices. Tools are produced at a great rate, but how many are maintained or cited, primarily because they do not work beyond a handful of test cases? Who ever heard of a computational biology research groups with a dedicated code tester? How many computational biology labs conduct code reviews or enforce the use of version repository system?

Perhaps the problem is that often computational biology research groups are structured so as to mirror that of their traditional biologists (so called "wet labbers"). We wonder whether a better mode of practice would be for computational biology groups to mirror agile

software development companies. How much more reproducible would our science be if we transferred some of the practices of professional software engineering firms to academia?

Managing software A problem familiar to many computational biologists is the difficulty in installing software in order to run it in the first place. This is a direct consequence of prioritising immediate results over software design. As long as the software compiles on the authors computer in a manner sufficient to generate results, then the software is considered complete. To improve our field, time must be taken to ensure that software is compatible on a range of user machines.

One way of combating this problem is in the use of programs like Docker (MERKEL, 2014) that aim to modularise the installation of software by providing "containers". Containers are lightweight environments that provide a layer of abstraction between the program and the host operating system, without the overheads of a virtual machine. Conveniently, the software designer may then ensure only that the container is run-able and not worry about ensuring their code works on a broad range of machines.

Both Telomerecat and Parabam are available as Docker containers. Furthermore, we have also ensured that both tools are easily installed via the Python Package Index such that only a single user command is required to install the tools. We have ensured that both tools work on a variety of Linux, Fedora, and MacOS platforms.

Over-parametrisation in software Another problem that plagues bioinformatics software is over parametrisation. Some notable bioinformatics software is designed such that the onus is put upon the user to provide many individual parameters to a piece of software in order to make it run. This is poor design.

Ideally, users should be able to run a piece of software with a single parameter; the input data. While it is good to provide the user with options to customise software, in our experience it is often the case that frequently an analysis will not require fine parametrisation and in these cases a default parameter should be provided. No one is better placed to provide default parameters than the designer of the software.

To ensure that our programs do not suffer from these problems, Parabam and Telomerecat have followed a command line interface design similar to that of samtools (the programmatic interface for BAM and SAM files). Samtools is well designed software that follows this principle of simplified parametrisation. Perhaps the success of samtools is best captured by it's near ubiquity on the computers of computational biologists. The samtools-esque paradigm for command line interface specifies:

```
program command /path/to/inputfile.type
```

For instance when running Telomerecat the following command is sufficient to generate a length estimation for a given sample:

```
telomerecat bam2length /path/to/inputfile.bam
```

Final thoughts

At the core of this thesis is a desire to add additional value to the combined sum of science, through reinterpretation of existing data. This comes with its own set of challenges. A key development hurdle to Parabam and Telomerecat was ensuring that the tools worked on a varied set of inputs. Considerable effort has gone into preparing the software for unexpected occurrences in features of the data or even the way it was formatted. There is still work to be done in this area.

This said, we feel there is tremendous value in creating tools that provide new insights from pre-existing data. In particular, we hope that Parabam finds a place as a software framework, relied upon to manipulate these high density data sets in hitherto unfathomed ways. Additionally, we are pleased to see Telomerecat has been used in at least three cancer studies already (aside from Chapter 4, see Section 3.4.4) to uncover new associations. We attest that there is no greater satisfaction for a computational biologist than to see their code being put to good use.

There's plenty of work left to do, especially in shedding new light on the life altering diseases highlighted in this thesis. We hope that the tools presented here assist as we approach these ends.

References

ABESHOUSE, A., J. AHN, R. AKBANI, A. ALLY, S. AMIN, *et al.*, 2015 The Molecular Taxonomy of Primary Prostate Cancer. Cell **163**: 1011–1025.

ABOLHASSANI, H., B. T. SAGVAND, T. SHOKUHFAR, B. MIRMINACHI, N. REZAEI, *et al.*, 2013 A review on guidelines for management and treatment of common variable immunodeficiency. Expert Rev Clin Immunol **9**: 561–574.

ADAMS, J., 1853 A case of scirrhus of the prostate gland with a corresponding affection of the lymphatic glands in the lumbar region and in the pelvis. The Lancet **61**.

ALLSHIRE, R. C., M. DEMPSTER, and N. D. HASTIE, 1989 Human telomeres contain at least three types of G-rich repeat distributed non-randomly. Nucleic Acids Res. **17**: 4611–4627.

AUBERT, G., M. HILLS, and P. M. LANSDORP, 2012 Telomere length measurement-caveats and a critical assessment of the available technologies and tools. Mutat. Res. **730**: 59–67.

AVIV, A., S. C. HUNT, J. LIN, X. CAO, M. KIMURA, *et al.*, 2011 Impartial comparative analysis of measurement of leukocyte telomere length/DNA content by Southern blots and qPCR. Nucleic Acids Res. **39**: e134.

AZZALIN, C. M., P. REICHENBACH, L. KHORIAULI, E. GIULOTTO, and J. LINGNER, 2007 Telomeric repeat containing RNA and RNA surveillance factors at mammalian chromosome ends. Science **318**: 798–801.

BAERLOCHER, G. M., and P. M. LANSDORP, 2003 Telomere length measurements in leukocyte subsets by automated multicolor flow-fish. Cytometry Part A **55A**: 1–6.

BARBIERI, C. E., S. C. BACA, M. S. LAWRENCE, F. DEMICHELIS, M. BLATTNER, *et al.*, 2012 Exome sequencing identifies recurrent SPOP, FOXA1 and MED12 mutations in prostate cancer. Nat. Genet. **44**: 685–689.

BARTHEL, F. P., W. WEI, M. TANG, E. MARTINEZ-LEDESMA, X. HU, *et al.*, 2017 Systematic analysis of telomere length and somatic alterations in 31 cancer types. Nat. Genet. **49**: 349–357.

BENETOS, A., K. OKUDA, M. LAJEMI, M. KIMURA, F. THOMAS, *et al.*, 2001 Telomere length as an indicator of biological aging: the gender effect and relation with pulse pressure and pulse wave velocity. Hypertension **37**: 381–385.

BLATTNER, M., D. LIU, B. D. ROBINSON, D. HUANG, A. POLIAKOV, *et al.*, 2017 SPOP Mutation Drives Prostate Tumorigenesis In Vivo through Coordinate Regulation of PI3K/mTOR and AR Signaling. Cancer Cell **31**: 436–451.

BODNAR, A. G., M. OUELLETTE, M. FROLKIS, S. E. HOLT, C. P. CHIU, *et al.*, 1998 Extension of life-span by introduction of telomerase into normal human cells. Science **279**: 349–352.

CAI, J., X. MIAO, Y. LI, C. SMITH, K. TSANG, *et al.*, 2014 Whole-genome sequencing identifies genetic variances in culture-expanded human mesenchymal stem cells. Stem Cell Reports **3**: 227–233.

CALLICOTT, R. J., and J. E. WOMACK, 2006 Real-time PCR assay for measurement of mouse telomeres. Comp. Med. **56**: 17–22.

CASTLE, J. C., M. BIERY, H. BOUZEK, T. XIE, R. CHEN, *et al.*, 2010 DNA copy number, including telomeres and mitochondria, assayed using next-generation sequencing. BMC Genomics **11**: 244.

CAWTHON, R. M., 2002 Telomere measurement by quantitative PCR. Nucleic Acids Res. **30**: e47.

CENTER, M. M., A. JEMAL, J. LORTET-TIEULENT, E. WARD, J. FERLAY, *et al.*, 2012 International variation in prostate cancer incidence and mortality rates. Eur. Urol. **61**: 1079–1092.

CHEN, L., B. GE, F. P. CASALE, L. VASQUEZ, T. KWAN, *et al.*, 2016 Genetic Drivers of Epigenetic and Transcriptional Variation in Human Immune Cells. Cell **167**: 1398–1414.

CHENG, L., R. MONTIRONI, D. G. BOSTWICK, A. LOPEZ-BELTRAN, and D. M. BERNEY, 2012 Staging of prostate cancer. Histopathology **60**: 87–117.

CHINEGWUNDOH, F., M. ENVER, A. LEE, V. NARGUND, T. OLIVER, *et al.*, 2006 Risk and presenting features of prostate cancer amongst African-Caribbean, South Asian and European men in North-east London. BJU Int. **98**: 1216–1220.

CODD, V., C. P. NELSON, E. ALBRECHT, M. MANGINO, J. DEELEN, *et al.*, 2013 Identification of seven loci affecting mean telomere length and their association with disease. Nat. Genet. **45**: 422–427.

COLLINS, K., 2000 Mammalian telomeres and telomerase. Curr. Opin. Cell Biol. **12**: 378–383.

COOPER, C. S., R. EELES, D. C. WEDGE, P. VAN LOO, G. GUNDEM, *et al.*, 2015 Analysis of the genetic phylogeny of multifocal prostate cancer identifies multiple independent clonal expansions in neoplastic and morphologically normal prostate tissue. Nat. Genet. **47**: 367–372.

DANIALI, L., A. BENETOS, E. SUSSER, J. D. KARK, C. LABAT, *et al.*, 2013 Telomeres shorten at equivalent rates in somatic tissues of adults. Nat Commun **4**: 1597.

DE KOK, J. B., G. W. VERHAEGH, R. W. ROELOFS, D. HESSELS, L. A. KIEMENEY, *et al.*, 2002 DD3(PCA3), a very sensitive and specific marker to detect prostate tumors. Cancer Res. **62**: 2695–2698.

DING, Z., M. MANGINO, A. AVIV, T. SPECTOR, R. DURBIN, *et al.*, 2014 Estimating telomere length from whole genome sequence data. Nucleic Acids Res. **42**: e75.

DING, Z., C. J. WU, M. JASKELIOFF, E. IVANOVA, M. KOST-ALIMOVA, *et al.*, 2012 Telomerase reactivation following telomere dysfunction yields murine prostate tumors with bone metastases. Cell **148**: 896–907.

EISENSTEIN, M., 2015 Startups use short-read data to expand long-read sequencing market. Nat. Biotechnol. **33**: 433–435.

ELBERS, C. C., M. E. GARCIA, M. KIMURA, S. R. CUMMINGS, M. A. NALLS, *et al.*, 2014 Comparison between southern blots and qPCR analysis of leukocyte telomere length in the health ABC study. J. Gerontol. A Biol. Sci. Med. Sci. **69**: 527–531.

ETZIONI, R., R. CHA, E. J. FEUER, and O. DAVIDOV, 1998 Asymptomatic incidence and duration of prostate cancer. Am. J. Epidemiol. **148**: 775–785.

FARMERY, J. H. R., M. L. SMITH, and A. G. LYNCH, 2018 Telomerecat: A ploidy-agnostic method for estimating telomere length from whole genome sequencing data. Sci Rep **8**: 1300.

FELDMAN, B. J., and D. FELDMAN, 2001 The development of androgen-independent prostate cancer. Nat. Rev. Cancer **1**: 34–45.

FEUERBACH, L., L. SIEVERLING, K. DEEG, P. GINSBACH, B. HUTTER, *et al.*, 2016 Telomerehunter: telomere content estimation and characterization from whole genome sequencing data (preprint). bioRxiv .

FORDYCE, C. A., C. M. HEAPHY, N. E. JOSTE, A. Y. SMITH, W. C. HUNT, *et al.*, 2005 Association between cancer-free survival and telomere DNA content in prostate tumors. J. Urol. **173**: 610–614.

GAO, J., B. A. AKSOY, U. DOGRUSOZ, G. DRESDNER, B. GROSS, *et al.*, 2013 Integrative analysis of complex cancer genomics and clinical profiles using the cBioPortal. Sci Signal **6**: pl1.

GARDNER, M., D. BANN, L. WILEY, R. COOPER, R. HARDY, *et al.*, 2014 Gender and telomere length: systematic review and meta-analysis. Exp. Gerontol. **51**: 15–27.

GLOUSKER, G., F. TOUZOT, P. REVY, Y. TZFATI, and S. A. SAVAGE, 2015 Unraveling the pathogenesis of Hoyeraal-Hreidarsson syndrome, a complex telomere biology disorder. Br. J. Haematol. **170**: 457–471.

GREENLEE, R. T., T. MURRAY, S. BOLDEN, and P. A. WINGO, 2000 Cancer statistics, 2000. CA Cancer J Clin **50**: 7–33.

GREIDER, C. W., and E. H. BLACKBURN, 1985 Identification of a specific telomere terminal transferase activity in Tetrahymena extracts. Cell **43**: 405–413.

GUTIERREZ-RODRIGUES, F., B. A. SANTANA-LEMOS, P. S. SCHEUCHER, R. M. ALVES-PAIVA, and R. T. CALADO, 2014 Direct comparison of flow-FISH and qPCR as diagnostic tests for telomere length measurement in humans. PLoS ONE **9**: e113747.

HANAHAN, D., and R. A. WEINBERG, 2000 The hallmarks of cancer. Cell **100**: 57–70.

HAYFLICK, L., and P. S. MOORHEAD, 1961 The serial cultivation of human diploid cell strains. Exp. Cell Res. **25**: 585–621.

HEAPHY, C. M., R. F. DE WILDE, Y. JIAO, A. P. KLEIN, B. H. EDIL, *et al.*, 2011a Altered telomeres in tumors with ATRX and DAXX mutations. Science **333**: 425.

HEAPHY, C. M., A. P. SUBHAWONG, S. M. HONG, M. G. GOGGINS, E. A. MONTGOMERY, *et al.*, 2011b Prevalence of the alternative lengthening of telomeres telomere maintenance mechanism in human cancer subtypes. Am. J. Pathol. **179**: 1608–1615.

HEAPHY, C. M., G. S. YOON, S. B. PESKOE, C. E. JOSHU, T. K. LEE, *et al.*, 2013 Prostate cancer cell telomere length variability and stromal cell telomere length as prognostic markers for metastasis and death. Cancer Discov **3**: 1130–1141.

HEEG, S., N. HIRT, A. QUEISSER, H. SCHMIEG, M. THALER, *et al.*, 2011 EGFR overexpression induces activation of telomerase via PI3K/AKT-mediated phosphorylation and transcriptional regulation through Hif1-alpha in a cellular model of oral-esophageal carcinogenesis. Cancer Sci. **102**: 351–360.

HEEMERS, H. V., and D. J. TINDALL, 2007 Androgen receptor (AR) coregulators: a diversity of functions converging on and regulating the AR transcriptional complex. Endocr. Rev. **28**: 778–808.

HEISS, N. S., S. W. KNIGHT, T. J. VULLIAMY, S. M. KLAUCK, S. WIEMANN, *et al.*, 1998 X-linked dyskeratosis congenita is caused by mutations in a highly conserved gene with putative nucleolar functions. Nat. Genet. **19**: 32–38.

HEMANN, M. T., and C. W. GREIDER, 2000 Wild-derived inbred mouse strains have short telomeres. Nucleic Acids Res. **28**: 4474–4478.

HUANG, W., L. LI, J. R. MYERS, and G. T. MARTH, 2012 ART: a next-generation sequencing read simulator. Bioinformatics **28**: 593–594.

HUDSON, T. J., W. ANDERSON, A. ARTEZ, A. D. BARKER, C. BELL, *et al.*, 2010 International network of cancer genome projects. Nature **464**: 993–998.

HUMPHREY, P. A., 2004 Gleason grading and prognostic factors in carcinoma of the prostate. Mod. Pathol. **17**: 292–306.

HURWITZ, L. M., C. M. HEAPHY, C. E. JOSHU, W. B. ISAACS, Y. KONISHI, *et al.*, 2014 Telomere length as a risk factor for hereditary prostate cancer. Prostate **74**: 359–364.

ICGC, 2017 ICGC Prostate Cancer Project Summary. `http://icgc.org/icgc/cgp/70/508/71331`. [Online; accessed 1-August-2017].

IGLESIAS ALZUETA, J., and N. MATAMOROS FLORI, 2001 [Common variable immunodeficiency. Review]. Allergol Immunopathol (Madr) **29**: 113–118.

ILLUMINA INC., 2010 Technology Spotlight: Illumina Sequencing Technology. https://www.illumina.com/documents/products/techspotlights/techspotlight_sequencing.pdf. [Online; accessed 23-August-2017].

ILLUMINA INC., 2017 An Introduction to Next-Generation Sequencing Technology. https://www.illumina.com/content/dam/illumina-marketing/documents/products/illumina_sequencing_introduction.pdf. [Online Video; accessed 23-August-2017].

ISHII, E., S. OHGA, S. IMASHUKU, N. KIMURA, I. UEDA, *et al.*, 2005 Review of hemophagocytic lymphohistiocytosis (HLH) in children with focus on Japanese experiences. Crit. Rev. Oncol. Hematol. **53**: 209–223.

JEMAL, A., F. BRAY, M. M. CENTER, J. FERLAY, E. WARD, *et al.*, 2011 Global cancer statistics. CA Cancer J Clin **61**: 69–90.

JONES, D., K. M. RAINE, H. DAVIES, P. S. TARPEY, A. P. BUTLER, *et al.*, 2016 cgpCaVEManWrapper: Simple Execution of CaVEMan in Order to Detect Somatic Single Nucleotide Variants in NGS Data. Curr Protoc Bioinformatics **56**: 1–15.

KEANE, T. M., L. GOODSTADT, P. DANECEK, M. A. WHITE, K. WONG, *et al.*, 2011 Mouse genomic variation and its effect on phenotypes and gene regulation. Nature **477**: 289–294.

KHEIRANDISH, P., and F. CHINEGWUNDOH, 2011 Ethnic differences in prostate cancer. Br. J. Cancer **105**: 481–485.

KIPLING, D., and H. J. COOKE, 1990 Hypervariable ultra-long telomeres in mice. Nature **347**: 400–402.

KLOTZ, L., and M. EMBERTON, 2014 Management of low risk prostate cancer-active surveillance and focal therapy. Nat Rev Clin Oncol **11**: 324–334.

LA, M., K. KIM, J. PARK, J. WON, J. H. LEE, *et al.*, 2004 Daxx-mediated transcriptional repression of MMP1 gene is reversed by SPOP. Biochem. Biophys. Res. Commun. **320**: 760–765.

LANSDORP, P. M., N. P. VERWOERD, F. M. VAN DE RIJKE, V. DRAGOWSKA, M. T. LITTLE, *et al.*, 1996 Heterogeneity in telomere length of human chromosomes. Hum. Mol. Genet. **5**: 685–691.

LATIL, A., D. VIDAUD, A. VALERI, G. FOURNIER, M. VIDAUD, *et al.*, 2000 htert expression correlates with MYC over-expression in human prostate cancer. Int. J. Cancer **89**: 172–176.

LEE, M., C. E. NAPIER, S. F. YANG, J. W. ARTHUR, R. R. REDDEL, *et al.*, 2017 Comparative analysis of whole genome sequencing-based telomere length measurement techniques. Methods **114**: 4–15.

LI, H., B. HANDSAKER, A. WYSOKER, T. FENNELL, J. RUAN, *et al.*, 2009 The Sequence Alignment/Map format and SAMtools. Bioinformatics **25**: 2078–2079.

LI, J., J. A. DJENABA, A. SOMAN, S. H. RIM, and V. A. MASTER, 2012 Recent trends in prostate cancer incidence by age, cancer stage, and grade, the United States, 2001-2007. Prostate Cancer **2012**: 691380.

LIN, K. W., and J. YAN, 2008 Endings in the middle: current knowledge of interstitial telomeric sequences. Mutat. Res. **658**: 95–110.

LIN, Y., H. UEMURA, K. FUJINAMI, M. HOSAKA, M. HARADA, *et al.*, 1997 Telomerase activity in primary prostate cancer. J. Urol. **157**: 1161–1165.

LIU, D., M. S. O'CONNOR, J. QIN, and Z. SONGYANG, 2004 Telosome, a mammalian telomere-associated complex formed by multiple telomeric proteins. J. Biol. Chem. **279**: 51338–51342.

LU, H., F. GIORDANO, and Z. NING, 2016 Oxford Nanopore MinION Sequencing and Genome Assembly. Genomics Proteomics Bioinformatics **14**: 265–279.

MACINTYRE, G., T. GORANOVA, D. DE SILVA, D. ENNIS, A. M. PISKORZ, *et al.*, 2017 Copy-number signatures and mutational processes in ovarian carcinoma. bioRxiv .

MARION, R. M., K. STRATI, H. LI, A. TEJERA, S. SCHOEFTNER, *et al.*, 2009 Telomeres acquire embryonic stem cell characteristics in induced pluripotent stem cells. Cell Stem Cell **4**: 141–154.

MARTIN, R. C., 2008 *Clean Code: A Handbook of Agile Software Craftsmanship*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1 edition.

MARTINCORENA, I., and P. J. CAMPBELL, 2015 Somatic mutation in cancer and normal cells. Science **349**: 1483–1489.

MARX, V., 2015 The DNA of a nation. Nature **524**: 503–505.

MASTERS, S. L., A. SIMON, I. AKSENTIJEVICH, and D. L. KASTNER, 2009 Horror autoinflammaticus: the molecular pathophysiology of autoinflammatory disease (*). Annu. Rev. Immunol. **27**: 621–668.

MCCLINTOCK, B., 1941 The Stability of Broken Ends of Chromosomes in Zea Mays. Genetics **26**: 234–282.

MERKEL, D., 2014 Docker: Lightweight linux containers for consistent development and deployment. Linux J. **2014**.

MINGUELL, J. J., A. ERICES, and P. CONGET, 2001 Mesenchymal stem cells. Exp. Biol. Med. (Maywood) **226**: 507–520.

MIRABELLO, L., W. Y. HUANG, J. Y. WONG, N. CHATTERJEE, D. READING, *et al.*, 2009 The association between leukocyte telomere length and cigarette smoking, dietary and physical variables, and risk of prostate cancer. Aging Cell **8**: 405–413.

MOAYYERI, A., C. J. HAMMOND, D. J. HART, and T. D. SPECTOR, 2013 The UK Adult Twin Registry (TwinsUK Resource). Twin Res Hum Genet **16**: 144–149.

MULLER, H., 1938 The Remaking of Chromosomes. The Collecting Net **13**: 181.

NERSISYAN, L., and A. ARAKELYAN, 2015 Computel: computation of mean telomere length from whole-genome next-generation sequencing data. PLoS ONE **10**: e0125201.

NORDFJALL, K., A. LAREFALK, P. LINDGREN, D. HOLMBERG, and G. ROOS, 2005 Telomere length and heredity: Indications of paternal inheritance. Proc. Natl. Acad. Sci. U.S.A. **102**: 16374–16378.

OLOVNIKOV, A. M., 1973 A theory of marginotomy. The incomplete copying of template margin in enzymic synthesis of polynucleotides and biological significance of the phenomenon. J. Theor. Biol. **41**: 181–190.

ORNISH, D., J. LIN, J. M. CHAN, E. EPEL, C. KEMP, *et al.*, 2013 Effect of comprehensive lifestyle changes on telomerase activity and telomere length in men with biopsy-proven low-risk prostate cancer: 5-year follow-up of a descriptive pilot study. Lancet Oncol. **14**: 1112–1120.

O'SULLIVAN, R. J., and J. KARLSEDER, 2010 Telomeres: protecting chromosomes against genome instability. Nat. Rev. Mol. Cell Biol. **11**: 171–181.

PALLER, C. J., and E. S. ANTONARAKIS, 2013 Management of biochemically recurrent prostate cancer after local therapy: evolving standards of care and new directions. Clin Adv Hematol Oncol **11**: 14–23.

PARKER, C., 2004 Active surveillance: towards a new paradigm in the management of early prostate cancer. Lancet Oncol. **5**: 101–106.

PARKER, M., X. CHEN, A. BAHRAMI, J. DALTON, M. RUSCH, *et al.*, 2012 Assessing telomeric DNA content in pediatric cancers using whole-genome sequencing data. Genome Biol. **13**: R113.

PARKINSON, G. N., M. P. LEE, and S. NEIDLE, 2002 Crystal structure of parallel quadruplexes from human telomeric DNA. Nature **417**: 876–880.

PERNER, S., J. M. MOSQUERA, F. DEMICHELIS, M. D. HOFER, P. L. PARIS, *et al.*, 2007 TMPRSS2-ERG fusion prostate cancer: an early molecular event associated with invasion. Am. J. Surg. Pathol. **31**: 882–888.

PFITZENMAIER, J., W. J. ELLIS, E. W. ARFMAN, S. HAWLEY, P. O. MCLAUGHLIN, *et al.*, 2006 Telomerase activity in disseminated prostate cancer cells. BJU Int. **97**: 1309–1313.

POOLEY, K. A., S. E. BOJESEN, M. WEISCHER, S. F. NIELSEN, D. THOMPSON, *et al.*, 2013 A genome-wide association scan (GWAS) for mean telomere length within the COGS project: identified loci show little association with hormone-related cancer risk. Hum. Mol. Genet. **22**: 5056–5064.

POWELL, I. J., 2011 Germline CAG repeat length of the androgen receptor and time to progression in patients with prostate cancer treated with androgen deprivation therapy. BJU Int. **108**: 1092.

QIN, Z., X. LI, P. HAN, Y. ZHENG, H. LIU, *et al.*, 2017 Association between polymorphic CAG repeat lengths in the androgen receptor gene and susceptibility to prostate cancer: A systematic review and meta-analysis. Medicine (Baltimore) **96**: e7258.

REBHAN, M., V. CHALIFA-CASPI, J. PRILUSKY, and D. LANCET, 1998 GeneCards: a novel functional genomics compendium with automated data mining and query reformulation support. Bioinformatics **14**: 656–664.

RIETHMAN, H., A. AMBROSINI, C. CASTANEDA, J. FINKLESTEIN, X. L. HU, *et al.*, 2004 Mapping and initial analysis of human subtelomeric sequence assemblies. Genome Res. **14**: 18–28.

RIETHMAN, H., A. AMBROSINI, and S. PAUL, 2005 Human subtelomere structure and variation. Chromosome Res. **13**: 505–515.

ROBLES-ESPINOZA, C. D., M. HARLAND, A. J. RAMSAY, L. G. AOUDE, V. QUESADA, *et al.*, 2014 POT1 loss-of-function variants predispose to familial melanoma. Nat. Genet. **46**: 478–481.

SAMSONRAJ, R. M., M. RAGHUNATH, J. H. HUI, L. LING, V. NURCOMBE, *et al.*, 2013 Telomere length analysis of human mesenchymal stem cells by quantitative PCR. Gene **519**: 348–355.

SCHRODER, F. H., P. HERMANEK, L. DENIS, W. R. FAIR, M. K. GOSPODAROWICZ, *et al.*, 1992 The TNM classification of prostate cancer. Prostate Suppl **4**: 129–138.

SHAMPAY, J., J. W. SZOSTAK, and E. H. BLACKBURN, 1984 DNA sequences of telomeres maintained in yeast. Nature **310**: 154–157.

SHAY, J. W., and W. E. WRIGHT, 2011 Role of telomeres and telomerase in cancer. Semin. Cancer Biol. **21**: 349–353.

SIEVERLING, L., C. HONG, S. D. KOSER, P. GINSBACH, K. KLEINHEINZ, *et al.*, 2017 Genomic footprints of activated telomere maintenance mechanisms in cancer. bioRxiv .

SIMS, D., I. SUDBERY, N. E. ILOTT, A. HEGER, and C. P. PONTING, 2014 Sequencing depth and coverage: key considerations in genomic analyses. Nat. Rev. Genet. **15**: 121–132.

SOMMERFELD, H. J., A. K. MEEKER, M. A. PIATYSZEK, G. S. BOVA, J. W. SHAY, *et al.*, 1996 Telomerase activity: a prevalent marker of malignant human prostate tissue. Cancer Res. **56**: 218–222.

SONG, Z., G. VON FIGURA, Y. LIU, J. M. KRAUS, C. TORRICE, *et al.*, 2010 Lifestyle impacts on the aging-associated expression of biomarkers of DNA damage and telomere dysfunction in human blood. Aging Cell **9**: 607–615.

STEPHENSON, A. J., M. W. KATTAN, J. A. EASTHAM, Z. A. DOTAN, F. J. BIANCO, *et al.*, 2006 Defining biochemical recurrence of prostate cancer after radical prostatectomy: a proposal for a standardized definition. J. Clin. Oncol. **24**: 3973–3978.

STEURER, S., P. S. MAYER, M. ADAM, A. KROHN, C. KOOP, *et al.*, 2014 TMPRSS2-ERG fusions are strongly linked to young patient age in low-grade prostate cancer. Eur. Urol. **66**: 978–981.

STOCKING, J. J., M. V. FIANDALO, E. A. POP, J. H. WILTON, G. AZABDAFTARI, *et al.*, 2016 Characterization of Prostate Cancer in a Functional Eunuch. J Natl Compr Canc Netw **14**: 1054–1060.

TARASOV, A., A. J. VILELLA, E. CUPPEN, I. J. NIJMAN, and P. PRINS, 2015 Sambamba: fast processing of NGS alignment formats. Bioinformatics **31**: 2032–2034.

TISCHLER, G., and S. LEONARD, 2014 biobambam: tools for read pair collation based algorithms on BAM files. Source Code Biol Med **9**.

TOMLINS, S. A., A. BJARTELL, A. M. CHINNAIYAN, G. JENSTER, R. K. NAM, *et al.*, 2009 ETS gene fusions in prostate cancer: from discovery to daily clinical practice. Eur. Urol. **56**: 275–286.

TOMLINS, S. A., D. R. RHODES, S. PERNER, S. M. DHANASEKARAN, R. MEHRA, *et al.*, 2005 Recurrent fusion of TMPRSS2 and ETS transcription factor genes in prostate cancer. Science **310**: 644–648.

TUMMALA, H., A. WALNE, L. COLLOPY, S. CARDOSO, J. DE LA FUENTE, *et al.*, 2015 Poly(A)-specific ribonuclease deficiency impacts telomere biology and causes dyskeratosis congenita. J. Clin. Invest. **125**: 2151–2160.

VALDES, A. M., T. ANDREW, J. P. GARDNER, M. KIMURA, E. OELSNER, *et al.*, 2005 Obesity, cigarette smoking, and telomere length in women. Lancet **366**: 662–664.

VALLS, C., C. PINOL, J. M. RENE, J. BUENESTADO, and J. VINAS, 2011 Telomere length is a prognostic factor for overall survival in colorectal cancer. Colorectal Dis **13**: 1265–1272.

VARLEY, H., H. A. PICKETT, J. L. FOXON, R. R. REDDEL, and N. J. ROYLE, 2002 Molecular characterization of inter-telomere and intra-telomere mutations in human ALT cells. Nat. Genet. **30**: 301–305.

VULLIAMY, T. J., S. W. KNIGHT, P. J. MASON, and I. DOKAL, 2001 Very short telomeres in the peripheral blood of patients with X-linked and autosomal dyskeratosis congenita. Blood Cells Mol. Dis. **27**: 353–357.

WALKER, F. O., 2007 Huntington's disease. Lancet **369**: 218–228.

WARDE-FARLEY, D., S. L. DONALDSON, O. COMES, K. ZUBERI, R. BADRAWI, *et al.*, 2010 The GeneMANIA prediction server: biological network integration for gene prioritization and predicting gene function. Nucleic Acids Res. **38**: W214–220.

WARREN, A. Y., H. C. WHITAKER, B. HAYNES, T. SANGAN, L. A. MCDUFFUS, *et al.*, 2013 Method for sampling tissue for research which preserves pathological data in radical prostatectomy. Prostate **73**: 194–202.

WEISCHER, M., S. E. BOJESEN, and B. G. NORDESTGAARD, 2014 Telomere shortening unrelated to smoking, body weight, physical activity, and alcohol intake: 4,576 general population individuals with repeat measurements 10 years apart. PLoS Genet. **10**: e1004191.

WENTZENSEN, I. M., L. MIRABELLO, R. M. PFEIFFER, and S. A. SAVAGE, 2011 The association of telomere length and cancer: a meta-analysis. Cancer Epidemiol. Biomarkers Prev. **20**: 1238–1250.

YAMADA, O., K. OZAKI, M. AKIYAMA, and K. KAWAUCHI, 2012 JAK-STAT and JAK-PI3K-mTORC1 pathways regulate telomerase transcriptionally and posttranslationally in ATL cells. Mol. Cancer Ther. **11**: 1112–1121.

YANG, X., R. KHOSRAVI-FAR, H. Y. CHANG, and D. BALTIMORE, 1997 Daxx, a novel Fas-binding protein that activates JNK and apoptosis. Cell **89**: 1067–1076.

YATES, L. R., S. KNAPPSKOG, D. WEDGE, J. H. R. FARMERY, S. GONZALEZ, *et al.*, 2017 Genomic Evolution of Breast Cancer Metastasis and Relapse. Cancer Cell **32**: 169–184.

YU, J., J. YU, R. S. MANI, Q. CAO, C. J. BRENNER, *et al.*, 2010 An integrated network of androgen receptor, polycomb, and TMPRSS2-ERG gene fusions in prostate cancer progression. Cancer Cell **17**: 443–454.

ZHU, L., K. S. HATHCOCK, P. HANDE, P. M. LANSDORP, M. F. SELDIN, *et al.*, 1998 Telomere length regulation in mice is linked to a novel chromosome locus. Proc. Natl. Acad. Sci. U.S.A. **95**: 8648–8653.

ZIMMERMANN, S., M. VOSS, S. KAISER, U. KAPP, C. F. WALLER, *et al.*, 2003 Lack of telomerase activity in human mesenchymal stem cells. Leukemia **17**: 1146–1149.

ZLOTTA, A. R., S. EGAWA, D. PUSHKAR, A. GOVOROV, T. KIMURA, *et al.*, 2013 Prevalence of prostate cancer on autopsy: cross-sectional study on unscreened Caucasian and Asian men. J. Natl. Cancer Inst. **105**: 1050–1058.

Appendix A

Online code repositories and documentation

The complete code for Telomerecat is available from the following online code repository:

`http://www.github.com/jhrf/telomerecat`

Documentation for Telomerecat can be found at:

`http://telomerecat.readthedocs.io/en/latest/`

Likewise, the code for Parabam can be found in the following repository:

`http://www.github.com/jhrf/parabam`

Documentation for Parabam can be found at:

`http://parabam.readthedocs.io/en/latest/`

The *AlignedSegment* interface as provided by pysam

When reads are passed to user-rules in Parabam, the read is of type *AlignedSegment*. The *AlignedSegment* type is provided by the Python interface to samtools called pysam. It provides the a great many variables, all of which are accessible by the user rule. For a full specification of the pysam *AlignedSegment* class please refer to:
`http://pysam.readthedocs.io/en/latest/api.html#pysam.AlignedSegment`

An example instruction file for the subset operation mode

Below is the instruction file written to subsample the file used in benchmarking as part of Chapter 2. This function meant that 25% of reads were retained at random. We think that the brevity of this function demonstrates the simplicity of code required for complex operations.

```
import random

def rule(reads, constants, parent):
    if random.random() < 0.25:
        return {"subsample": reads}
    return {}

def get_subset_types():
    return ["subsample"]
```

An example instruction file for the stat operation mode

The following is an example of a user-rule and blueprint definition that counts the number of reads that have exactly 50/50 GC content. The output of this user rule would be a single CSV file with a row for each of the input BAM files. Each row would have two columns, the name of sample and the amount of reads with exactly 50/50 GC ratio.

```

def get_gc_ratio(read):
    gc_count = 0
    total = 0
    for base in read.seq:
        if base == "G" or base == "C":
            gc_count += 1
        total += 1

    return gc_count / total

def rule(read, constants, parent):
    results = {}
    if get_gc_ratio(read) == .5:
        results["gc_count"] = 1
    return results

def get_blueprints(blueprints):

    blueprints["gc_count"] = {"data": 0, "store_method": "cumu"}
```