Telomere dysfunction and the evolution of the cancer genome

A thesis submitted for the degree of Doctor of Philosophy (PhD) by

Alexander Kearsey

September 2024

Division of Cancer and Genetics

School of Medicine

Cardiff University

Supervisors: Prof. Duncan Baird, Dr. Kez Cleal

Abstract

Background
Telomere length has emerged as a prognostic marker in several cancers, including breast and chronic lymphocytic leukaemia (CLL). Telomere dysfunction has also been implicated in the formation of complex genome rearrangement patterns including chromothripsis, and in the progressive evolution of cancer genome architecture. This thesis explores the relationship between telomere length and genomic complexity across multiple cancer cohorts.
Methods
A novel software tool call teltool was developed that outperformed existing methods in predicting telomere length from Whole Genome Sequencing (WGS) data. Methods for assessing Structural Variants (SVs), Copy Number Alterations (CNAs) and genomic complexity in WGS data were developed, and applied to a breast cancer cohort (n=44), before expanding analysis to publicly available datasets of breast cancer (n=1591+80) and CLL (n=98).
Results
Across all cohorts, samples with shorter telomeres consistently exhibited increased genomic complexity. In breast cancer cohorts, the number of CNAs was increased ($p < 0.05$) when stratifying by measured and estimated telomere lengths. Similarly, in CLL, losses and total CNAs were higher ($p < 0.05$) in samples with shorter telomeres. All types of SVs showed increased frequencies ($p < 0.05$) in breast cancer cohorts with shorter telomeres, except for duplications in the ICGC cohort. Patterns of genome complexity were increased in short-telomere groups with increased numbers chained SVs ($p < 0.01$) and complex joining profiles identified. Importantly, we observed a threshold effect where samples could be segregated into "short" and "long" telomere groups, associated with distinct levels of genomic complexity, and this phenomenon was most pronounced in cohorts with accurate telomere length measurements. The optimal threshold for partitioning was remarkably similar to the previously identified 'fusogenic' telomere length threshold that defined prognosis across several tumour types.
Conclusion
The findings of this thesis further establish the relationship between telomere length and cancer genome complexity. The results imply that telomere dysfunction plays a key role in generating large-scale genomic rearrangements and patterns of complexity. This work contributes to our understanding of the role of telomere dysfunction in cancer genomics and may have implications for diagnostics and prognosis.

Acknowledgements

I would first like to thank all the members of the STELA group. They all made me feel welcome and were a tremendous help in furthering my understanding of the field of telomere biology through their answers to my questions and explanations of their research. They were also incredibly patient as I was developing my presentation skills during lab meetings and asked useful questions that shifted my perspective throughout this work.

I am extremely grateful to my supervisors Duncan and Kez, as this project would not have been possible without their guidance, support and advice. Duncan provided me with inspiration and encouragement throughout with his enthusiastic teachings and insights on this corner of biology. Kez was enormously helpful throughout the project; I am thankful and appreciative he was always available to talk to provide feedback and recommendations.

I would also like to extend my thanks to my friends and family. They put up with my ramblings in periods of frustration, and despite not always understanding were sympathetic. Their words of belief helped me continue working through difficult times.

Lastly, thank you to Cardiff University and Cancer Research UK for funding this work.

I would like to dedicate this work to my grandad Ron who died of cancer when I was young. His spirit and legacy have been a positive influence in my life and has been a key source of motivation throughout.

Abbreviations

A-NHEJ	Alternative Non-Homologous end joining
ALT	Alternative lengthening of telomeres
BFB	Breakage-fusion-bridge
BIR	Break induced replication
bp	Base pair
BRCA1 and 2	Breast cancer gene 1 and 2
C-NHEJ	Classical Non-Homologous end joining
CAN/V	Copy number alteration/variation
CLL	Chronic lymphocytic leukaemia
CN	Copy number
CST	CTC1, STN1, and TEN1
D-loop	Displacement loop
DDR	DNA damage response
DNA	Deoxyribonucleic acid
DNB	DNA nanoballs
ds	Double stranded
DSB	Double strand break
DSBR	Double strand break repair
FCR	fludarabine, cyclophosphamide, rituximab
FISH	Fluorescent in situ hybridisation
FN	False negative
FNR	False negative rate
FoSTeS	fork-stalling and template switching
FP	False positive
G1 phase	Growth phase 1
G2 phase	Growth phase 2
G4	Guanine quadruplex
GEL	Genomics England
HR	Homologous recombination
hTERC	Telomerase RNA component
hTERT	Human telomere reverse transcriptase
ICGC	International Cancer Genome Consortium
IGHV	variable region of the immunoglobulin heavy chain
IQR	Inter quartile range
ITS	Interstitial telomeric sequences
kb	Kilobase
KO	Knockout
LIG3 and 4	DNA ligase 3 and 4

LOH	Loss of heterozygosity
M1 phase	Mitotic phase 1 (senescence)
M2 phase	Mitotic phase 2 (crisis)
mb	Megabase
MIR	Multi-Invasion-Induced Rearrangements
MMBIR	Microhomology-Mediated Break Induced Replication
NGS	Next generation sequencing
NHEJ	Non-Homologous end joining
OS	Overall survival
PARP	Poly (ADP-ribose) polymerase
PCF	Piecewise constant fitting
PFR	Progression free survival
POLQ	DNA polymerase theta
Rb	Retinoblastoma
RNA	Ribonucleic acid
S ohase	Synthesis phase
SD	Standard deviation
siRNA	Small interfering RNA
ss	Single stranded
SSB	Single strand break
STELA	Single telomere length analysis
SV	Structural variant
T-loop	Telomeric loop
TL	Telomere length
TMEJ	Theta-mediated end joining
TN	True negative
TP	True positive
Tp53	Tumour protein P53
TPR	True positive rate
TRF1 and 2	Telomeric repeat factor 1 and 2
UCSC	University of California, Santa Cruz
WGS	Whole Genome Sequencing
ZMW	Zero mode waveguide

List of contents

List of figures

List of tables

Chapter 1 Introduction

1.1 Telomeres

1.1.1 Function and biology

Telomeres are structures found at the ends of linear eukaryotic chromosomes. In mammals telomeres are comprised of the hexameric sequence TTAGGG tandemly repeated into variable length arrays, that together with Shelterin protein complex confer primary function of telomere, which is to prevent recognition by DNA damage response (DDR) mechanisms of the natural chromosomal terminus as double-strand DNA breaks (Lange 2005). They have also been shown to contribute to the organisation of chromosomes within the nucleus and participate in gene expression regulation (Chuang et al. 2004; Kim and Shay 2018). The main function of preventing the DDR is achieved through the formation of a T-loop structure, which are a composition of chromatin structures that work in conjuncture with shelterin (Srinivas et al. 2020). Telomeres do not terminate as a blunt end but rather contain a single stranded 200-400nt G-rich overhang, which gives rise to the end replication problem. The end-replication problem arises because synthesis is unable to replicate the last section of the telomere, leading to shortening of the genome every cell cycle (Olovnikov 1973). In stem and cancer cells telomerase with its internal RNA template solves this problem by synthesising single-stranded TTAGGG repeats to the telomere end, enabling complete replication whilst maintaining or elongating telomere length (Collins 2011). Generally, this problem is solved in lagging strand synthesis by Okazaki fragments inability to be initiated at the telomere ends, and in leading strand synthesis via the resection of the 5’ end from Exo1 and Apollo. In both

cases POT1b associated CST (CTC1, STN1, and TEN1 complex) is recruited to fill in the C-rich strand to correct excessive 3' overhangs (Wu et al. 2012). Due to the end replication problem, telomeres also act as an indicator for how many times a cell has divided, with the maximum number of divisions being referred to as the Hayflick limit (Hayflick 1965; Kailashiya et al. 2017). Sufficient telomere shortening can trigger DDR to enter an arrested state of senescence, delaying its advancement in the cell cycle (figure 1.1) (Rossiello et al. 2022). This replicative limit acts as a tumour suppressive mechanism in long lived species by providing an intrinsic capacity for replacement to facilitate normal growth, development, and tissue homeostasis, but not to permit the many divisions seen in oncogenesis (Shay and Wright 2000; Schmutz et al. 2020).

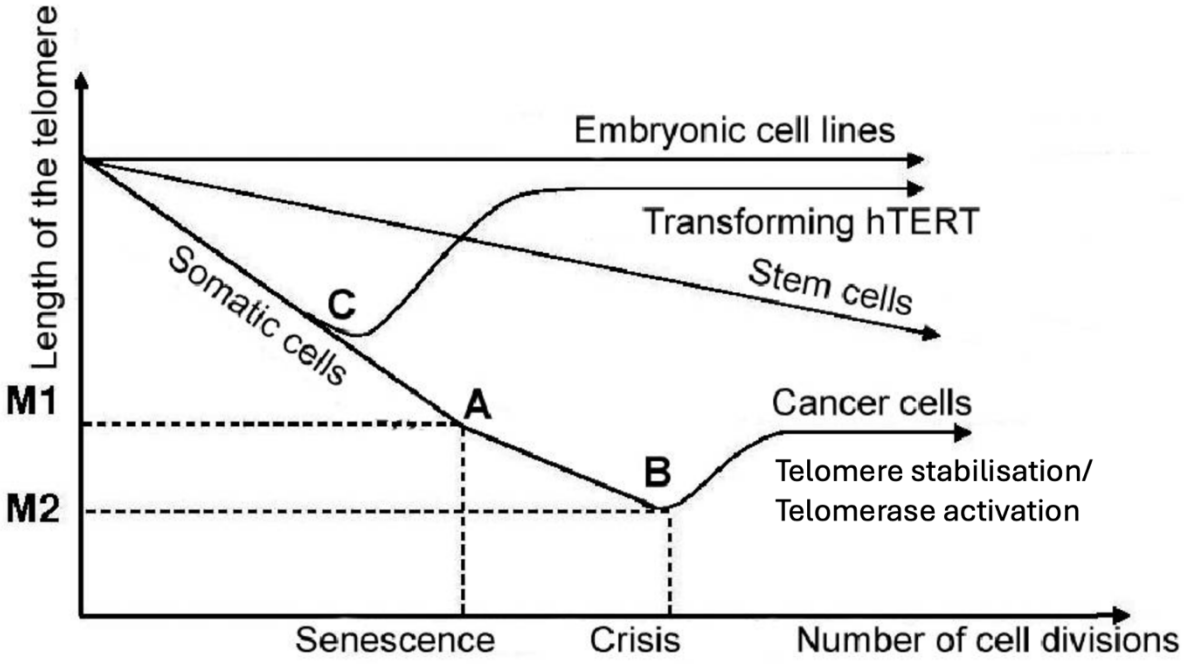


Figure 1.1 Graph to show telomere length (y-axis) against the number of cell divisions (x-axis) for various cell lines. (A) Hayflick limit reached, (B) crisis with the following cell death and transformation of surviving cells into cancer cells, (C) transfection of cells by the hTERT gene. Figure reproduced and altered under Creative Commons Attribution License from: Skvortzov DA, Rubzova MP, Zvereva ME, Kiselev FL, Donzova OA. The regulation of telomerase in oncogenesis. Acta Naturae. 2009 Apr;1(1):51-67. PMID: 22649586; PMCID: PMC3347505 (figure 2).

1.1.1.1 Telomerase and Alternative lengthening of telomeres (ALT pathway)

Telomerase activity is absent in the majority of human somatic cells, yet is present in around 90% of cancers (Shay 2016). Telomerase consists of two core components, a functional telomerase RNA (hTR/hTERC) that provides the telomere repeat template, and the enzymatic reverse transcriptase hTERT. Interestingly hTR is highly expressed in all tissues regardless of telomerase activity, but is expressed at five times higher levels in cancer (Yi et al. 1999). The structure of hTR varies between species, but in humans takes the form of a 451nt non-coding RNA containing 4 conserved domains. These 4 domains conform themselves into secondary structures composed of conserved regions (CR): pseudoknot CR2/3, CR4/5, H/ACA box (CR6/8), and CR7. The 5'-CUAACCCU-3' template (CR1) for the hTERT reverse transcription is located near (45nt away from) the 5' end of the hTR (Chen et al. 2000). Due to the ubiquity of hTR, the expression of hTERT determines telomerase activity and is up-regulated in cancer and stem cell lines compare to normal differentiated cells (Zinn et al. 2007; Leão et al. 2018). The protein component of telomerase hTERT has four domains, telomerase essential N-terminal domain (TEN), RNA-binding domain (RBD), reverse transcriptase (RT) which holds a telomerase RAP motif (TRAP) motif near the middle of the domain, and the C-terminal extension (CTE). The catalytic core is composed of the CR2/3 of hTR, along with the RBD, RT, and CTE hTERT domains which form ring. A complex above the ring made of the TEN and TRAP is what is responsible for binding to TPP1 (at its OB domain) of shelterin (Liu et al. 2022). Telomerase is also known to have several interactions with other proteins which are thought to aid activation and enzymatic activity, either through mediating the assembly process, or regulation of access to the substrate. Some examples of these are interactions between hTERT and P23 which aids assembly, and 14-3-3 which aids nuclear localisation. The RNA element hTR has more interactions with: hGAR1, Dyskerin, hNOP1, hNHP2 are all known to interact with the H/ACA domain on hTR, and TCAB1

which binds to the CR7 domain. These are all thought to play a role in the stability, maturation, and localisation of telomerase, however the full list of interactions and functions are still unclear (Cong et al. 2002). Telomerase is recruited by shelterin to the 3' overhang, where it uses the hTR RNA template to append telomeric repeats to extend the single stranded DNA. CST then recruits Pol α which fills in the C rich strand (figure 1.2) (Chen and Podlevsky 2016).

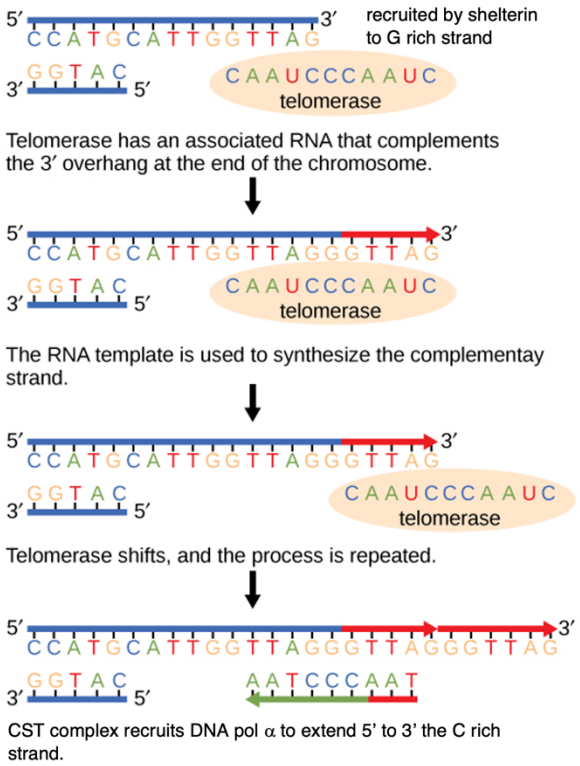


Figure 1.2 Diagram to show how telomerase elongates telomeres. Telomerase is recruited by shelterin to the G-rich strand containing the 3' overhang. A portion of the RNA template anneals to the overhang, and telomerase synthesises new single stranded DNA to extend the overhang. Telomerase is shifted over and this process repeats to continue elongation. CST complex then recruits Pol α to fill in the C rich strand. Figure reproduced and adapted under the Creative Commons license from: Rye, C. Wise, R. Jurukovski, V. DeSaix, J. Choi, J Avissar, Y. Biology (Chapter 14.5). OpenStax 2016 https://openstax.org/books/biology/pages/1-introduction (Figure 14.15)

The alternative lengthening of telomeres (ALT) serves to prevent telomere shortening that occurs during proliferation and is observed in ~10% of all cancers (Rosso et al. 2023). ALT positive cells exhibit extrachromosomal telomeric sequences in both linear and circular (t-circles) forms. The t-circle forms can be double stranded or partially single stranded which

either favour the C or G rich strand (C/G-circles) (Nabetani and Ishikawa 2011). Abundance of C-circles in particular correlate well with ALT activity (Henson et al. 2009). The pathway for elongation of telomeres in the absence of telomerase relies on homologous recombination of overhanging 3' ends invading either a part of the same telomere, a sister chromatid/another chromosome's telomere, or extrachromosomal telomeric sequence (Hou et al. 2022). More details on homologous recombination are explored in a later section of this chapter. TERRA R-loops are present in higher levels for telomerase-negative cancer cells compared to their telomerase-positive counterparts. R-loop formation at telomeres is thought to regulate telomere maintenance and encourage genomic stability by regulating chromatin, priming DNA replication, or promoting homologous recombination amongst telomeres (Hou et al. 2022). Maintenance of telomere length alone is not enough to perform their protective function, which requires DNA to be complexed with shelterin.

1.1.1.2 Shelterin

Shelterin is a six-subunit protein containing: TRF1, TRF2, POT1, RAP1, TIN2 and TPP1 (Lange 2005; Zinder et al. 2022). Telomeric repeat-binding factor 1 and 2 (TRF1 and TRF2) are both homodimeric proteins responsible for binding to the TTAGGG sequences (through the Myb subdomains contained on both proteins) (Ilicheva et al. 2015). Combined they form a core by which the other four proteins are recruited (Diotti and Loayza 2011). TERF1-interacting nuclear factor 2 (TIN2) acts as a stabiliser where its TRF-binding motif (TBM) subdomain binds to the TFR1/2 TRF-homology (TRFH) subdomains (Ye et al. 2004; Storchova et al. 2023). TIN2 also accommodates binding of the tripeptidyl peptidase 1 and protection of telomeres protein 1 (TPP1-POT1) complex (from a TRFH subdomain within TIN2 to a TBM subdomain on TPP1) (Kalathiya et al. 2018). The activity of POT1 is to bind to single stranded DNA at the 3′ end of the telomeres, which has its binding affinity increased

when bound to TPP1 (aiding the formation of a displacement-loop) (figure 1.3) (Wu et al. 2020). The last subunit of shelterin Repressor/Activator Protein 1 (RAP1) is another stabilising protein that inhibits the DDR process (Cai et al. 2017). Altogether shelterin with this conformation prevents the activation of the DDR at the capped chromosome via the formation of a T-loop, it also plays a role in telomerase (reverse transcriptase TERT) activity regulation.

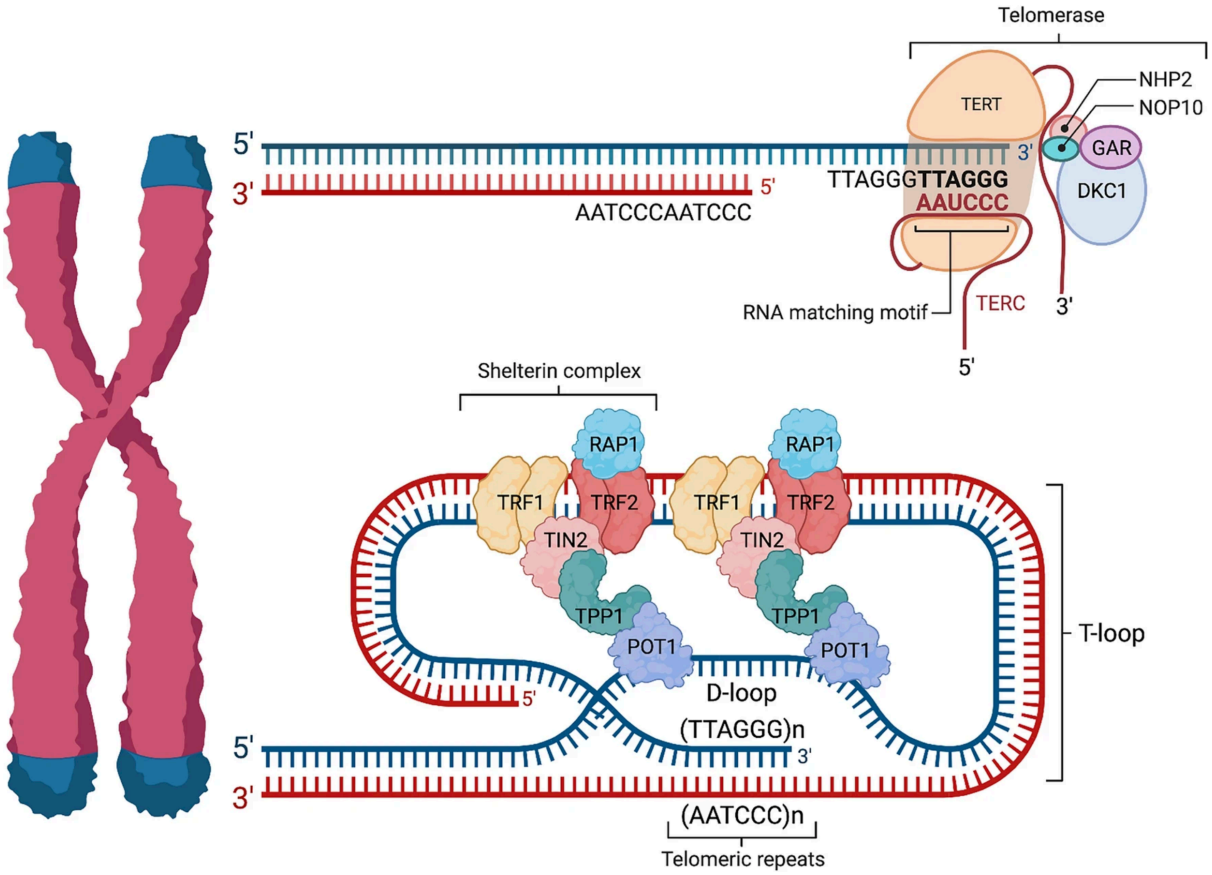


Figure 1.3 . Diagram showing the secondary structure of the T-loop and D-loop with the shelterin complex. TRF1 and TRF2 bind to double stranded telomeric repeats, bridged by TIN2 which also interacts with TPP1. TPP1 further interacts with POT1 which binds the single strand DNA displaced by the 3' overhang. RAP1 binds to TRF2. The top of the figure displays the ribonucleoprotein complex of telomerase containing: the retro-transcriptase hTERT, RNA component (hTERC or hTR), dyskerin (DKC1), NHP2, NOP10, and GAR. Figure reproduced under the Creative Commons Attribution 4.0 International License from: Muoio, D., Laspata, N. & Fouquerel, E. Functions of ADP-ribose transferases in the maintenance of telomere integrity. Cell. Mol. Life Sci. 79, 215 (2022). https://doi.org/10.1007/s00018-022-04235-z (figure 1)

1.1.1.3 T-/D-loops and replicative stress

The majority of the telomeric DNA in conjunction with shelterin forms higher order structures referred to as T-loops. At the 3' ends there is a single stranded portion of G-rich telomeric sequence (~200bp) that invades near the beginning of the T-loop to form a displacement loop (D-loop) (figure 1.2) (Greider 1999). As mentioned previously, the structure and components of the T- and D-loops function to prevent DDR mechanisms from activating, maintaining the stability of the end of the chromosome. However, they also create a replication problem, in tandem with heterochromatin, G-quadruplexes (G4), and telomere repeat containing RNAs (TERRAs) obstacles to replication forks and regular DNA helicases are created. A combination of these factors leads to increased stress during replication as detailed below (Maestroni et al. 2017).

G4 structures can occur at the D-loop and require unwinding before a replication fork can pass without stalling or collapsing. It is possible that POT1 is involved in inhibiting the formation of G4 structures, but it is likely that helicases are also involved in unwinding them (Nandakumar and Cech 2013). Helicases implicated in this function include Werner syndrome RecQ like helicase (WRN), Bloom syndrome RecQ like helicase (BLM), and regulator of telomere elongation helicase 1 (RTEL1). It is theorised that the recruitment of WRN at G4 sites is arbitrated through interactions with replisome cofactors such as replication protein A complex (RPA), proliferating cell nuclear antigen (PCNA), and DNA polymerase delta catalytic subunit (DPOLD1), as well as TRF2 (Shen and Loeb 2001; Machwe et al. 2004). TRF2 has also been shown to recruit and stimulate G4 helicase BLM whilst TRF1 inhibits BLM activity, but is more likely utilised in telomerase negative cells (Lillard-Wetherell et al. 2004). RTEL1 also has the capacity to unwind 5'-3' G4 structures and can be recruited in the replisome by PCNA, and to TRF2 (Hourvitz et al. 2024).

This ability to bind to TRF2 also facilitates RTEL1's other role to be responsible for

unwinding of the T- and D-loop to allow a smooth passage of the replisome. It is suggested that dephosphorylation of TRF2 (at Ser365 by PP6R3) during S-phase opens a RTEL1 interaction site which facilitates opening of the T/D-loop. Phosphorylation of TRF2 outside of the S-phase is likely performed by cyclin A-CDK2 (Sarek et al. 2019). Whilst it is not known whether RTEL1 is the only helicase involved in T-loop disassembly, in the absence of RTEL1, structure-specific endonuclease subunit (SLX) 1 and 4 are used as a last resort to resolve the T-loop (Uringa et al. 2012; Vannier et al. 2012).

Another factor involved in the maintenance of the telomeric structure are telomeric repeat containing RNAs (TERRAs) are G-rich products from transcription of telomeres. They may play an essential role in maintaining telomere length along with shelterin, telomerase, and the CST (related to the RPA) complex (Giraud-Panis et al. 2010). Their expression is cell cycle mediated with its highest transcription rate being early in the S-phase (G1-S transition) and declining towards the transition to the G2 stage (Arnoult et al. 2012; Flynn et al. 2015). TERRAs can anneal to their C-rich DNA counterparts either during or post transcription forming an R-loop structure. R-loops formed at loci of non-G4 and G4 structures can create another source of replicative stress (or even DSBs) with the latter being more severe (Rippe and Luke 2015). Therefore, it is important R-loops must be cleared before replication. RNA endonuclease H (RNase H), Up-frameshift 1 (UPF1), ATP-dependent helicase (ATRX) (which is also required for deposition of histone H3.3 at telomeres and other repeat loci), and flap structure-specific endonuclease 1 (FEN1) are all candidates for resolving R-loop structures (Costantino and Koshland 2015; Flynn et al. 2015; Chib et al. 2016; Maestroni et al. 2017; Ngo et al. 2021).

Despite being a cause of stress, T-loops still prevent telomeres being mistaken for DNA damage sites which is crucial for maintaining genomic stability and a cell's ability to proliferate. Due to semiconservative replication of DNA and exonucleolytic processing,

telomeres shorten with every replicative cycle, which compromises their function (Artandi and DePinho 2010).

1.1.2 Telomere Dysfunction

Attrition of telomeres can have two opposing effects during carcinogenesis. Normally telomere shortening leads to the loss of their end-protection function, which leads to them being mistaken as a double strand break (DSB). This causes a partial DNA damage response (DDR), where cells exit the cell cycle at the p53 checkpoint during G1/S (Vodicka et al. 2021). This further initiates a tumour suppressive function via activating ATM and ATR kinase signalling at unprotected chromosome ends causing cell cycle arrest, senescence, or apoptosis (Hill et al. 2024). However, when the telomeres are not repaired the cell remains in a permanent state of arrest. This is thought to contribute to the ageing process and age-related diseases (Baird et al. 2003). There is a negative correlation between the rate of telomere erosion and organism life-span (Haussmann et al. 2003). Telomere length and telomerase activity have evolved to be as they are in long-live species like humans, compared to short lived species such as birds due to life-history and selective trade-offs in longevity and reproduction (Haussmann and Mauck 2008).

In the absence of the p53 and Rb pathways, cell cycle transitions are unaffected by the normal mechanisms of inhibition through ATM and ATR signalling, and the senescence barrier can be bypassed, thus cells develop into crisis (Jacobs and de Lange 2004; Mijit et al. 2020). In tissue culture, telomere crisis is typified by high rates of cell death, cell-morphological changes and large-scale genomic instability triggered by telomere dysfunction, and fusion (Greenberg 2005). This occurs because ongoing cell division leads to further telomere erosion, even to the point where telomeres are completely denuded of their repeats and are fully processed as a DSB, creating fusion events (Capper et al. 2007). As will be

discussed further in the next section, phenomena such as telomere fusions can occur at multiple inter and intra-chromosomal loci, creating dicentric chromosomes and initiating breakage-fusion-bridge (BFB) cycles and large-scale genomic rearrangements (Letsolo et al. 2010; Jones et al. 2014). During BFB cycles, fused chromosomes break at sites distant from the initial fusion, leading to the gain of DNA for one chromosomes, and loss of another (Murnane 2012).

To restore telomere function and escape crisis cells must establish a telomere maintenance mechanism, either via the reactivation of telomerase (90% of cancers) or through the alternative lengthening of telomeres (ALT) pathway (10% of cancers), both of which allow cells to regain the ability to proliferate (Chang et al. 2003). Genomic rearrangements are one of the mechanisms for the reactivation of telomerase activity through disruption of telomerase silencing, or activation and expression of telomerase mainly through hTERT promoter region mutations or epigenetic means (Sui et al. 2013; Chiba et al. 2015). ALT is a telomerase independent mechanism utilised by neoplasms for maintaining telomere length, and permits continued cell replication (Lawlor et al. 2019). Stabilization of the telomeres by ALT may prolong genome instability. This may be because ALT relies on recombination events involving telomeres which may be prone to rearrangements with other genomic loci. This is shown by telomeric motifs (TTAGGG) being present within chromosome in interstitial telomeric sequences (ITSs) (Aksenova and Mirkin 2019). Prior to regaining stability, the genome may undergo severe rearrangements via the formation of dicentric chromosomes or other processes. During these rearrangement processes, the cell may also acquire other de novo and potentially tumorigenic/oncogenic SNVs through, for example APOBEC mutagenesis (Maciejowski and de Lange 2017). Previous research in adenomatous colorectal polyps and chronic lymphocytic leukaemia has identified a correlation between telomere shortening (despite detectable telomerase activity insufficient to

maintain or elongate), and large-scale genomic rearrangements and likely play a role in early in malignant progression of many cancer types (Lin et al. 2010; Roger et al. 2013).

1.2 Telomeres and genomic complexity

The initial definition of telomere (Greek 'telos' meaning end and 'meros' part) came from Hermann Miller who discovered telomeres in Drosophila, which was also found later from Barber McClintock in maize (Muller 1938; McClintock 1939). McClintock would then observe variegation of chromosomes with broken ends, providing evidence for the function of the telomere that their function is to distinguish between chromosomes and double strand breaks (McClintock 1941). Both Muller and McClintock were awarded Nobel prizes for their discoveries (Varela and Blasco 2010).

1.2.1 Telomere fusions

Telomere fusions arise as a protection mechanism to protect against excessive degradation of exposed telomere ends (Stroik and Hendrickson 2020). One study of telomere fusions showed that approximately half did not contain any telomeric repeats, and the longest repeat observed in those that did was only ~13 repeats long (~78bp). They also found telomere fusion was often complemented by the deletion of one or both telomeres, which extends several kilobases into the telomere-adjacent DNA, with observable microhomology at the fusion points. (Capper et al. 2007). This feature of deletion of telomeres several kilobases was shared in a further study, which also found that a preference (60% of cases) of fusion sites containing short homologous DNA patches with GC biases (Letsolo et al. 2010). Shelterin typically protects from telomere fusions, however, over or under-expression of shelterin components can lead to increased fusion events (Lisaingo et al. 2014; de Lange

2018). Shortened telomeres can affect the mutations and occupancy of shelterin; therefore cells with shorter telomeres have a higher chance of forming telomere fusions which can leading to an increase frequency of genomic instability from fragmentation (chromothripsis) (Counter et al. 1992; Capper et al. 2007; Cleal et al. 2019). Fragmentation can occur by shattering of dicentric chromosomes (formed by telomere fusions) by mechanical force exerted by microtubules during mitosis.

More commonly dicentric chromosomes are not resolved in this manner, instead chromatin bridges are formed between the two daughter cells that persist into the next G1 phase. Resolution of chromatin bridges happens by simple breaks forming fold-back inversions or large terminal deletions, or complex breaks leading to chromothripsis. Chromatin bridges also contain nuclear envelopes connected to both cells (Rodriguez-Muñoz et al. 2022). Resolving them involves rupturing of the envelope causing the blending of nucleic and cytoplasmic contents. Three prime repair exonuclease 1 (TREX1), an abundant component of cytoplasm, leads to the resolution of bridges by degrading the DNA into single strands along the bridge (Jiang and Chan 2024).

This mechanism with TREX1 is not the only mechanism for chromatin bridge resolution. Polymorphisms in the non-coding regions of Ankle1 have been associated with heightened susceptibility to breast and ovarian cancer, affecting both the general population and individuals carrying the BRCA1 mutation. Notably research on the nematode *C. elegans* has revealed that its LEM-3 orthologue localizes to the midbody cleavage plane, facilitating the processing of chromatin bridges (Chan and West 2018). Additionally, LEM-3 cooperates with BRC-1 (the *C. elegans* homologue of BRCA1) to uphold genomic stability (Hong et al. 2018). Further study is required to fully understand the implications and mechanisms in human cancer. Fragmented chromosomes and damaged chromosome-ends stemming from complex breaks, or the resolution of chromatin bridges are susceptible to various DNA

repair mechanisms, potentially resulting in chromothripsis (and kataegis) (Maciejowski et al. 2015).

1.2.2 Breakage-fusion-bridges (BFB) cycles

An alternative outcome of dicentric chromosome formation is breakage-fusion-bridge (BFB) cycles. Breaking of dicentric chromosomes leads to creation of more exposed DNA ends that can produce further fusions within daughter cells. BFB cycles are only terminated when either a telomere from another chromosome is translocated to the affected dicentric chromosome, or a new telomere is seeded at the free end (McClintock 1941). During these BFB cycles, genomic instability can be both generated and transferred between chromosomes (Bailey and Murnane 2006). Amplifications, loss of heterozygosity (LOH), and non-reciprocal translocations are all ways this instability can manifest (Maciejowski and de Lange 2017). Gene dosage variations are a substantial driver in cancer, particularly amplification, which can activate dominantly acting cancer genes (Stratton et al. 2009). Amplification of genes occurs by asymmetrical breaking of fused sister chromatids, and the broken chromosomes being replicated during S phase which persists into G2 phase. Fusion of the four telomere deficient ends can then either join symmetrically and form stabilised chromosomes, or asymmetrically to create large palindromes. These palindromes can either be contained in dicentric or centromere lacking chromosomes (Narayanan et al. 2006). Deletions or duplications can occur when the dicentric structure breaks during chromosome segregation, the daughter cells can then inherit broken chromosomes with alterations in copies of the regions. This broken ends can participate in further BFB cycles leading to palindromic gene amplification (high copy number states) (Tanaka and Yao 2009) .

Conversely loss of heterozygosity can occur if a daughter cell inherits a terminal deletion following a dicentric chromosome break and is common at cancer related loci

(Maciejowski and de Lange 2017). Following a chromosome break, it is possible for DNA from the broken end to join with another chromosome, leading to non-reciprocal translocations facilitated by break-induced replication (Anand et al. 2013; Malkova and Ira 2013). Fusions between chromosome ends lacking telomeres with internal genomic loci can occur, even in an otherwise stable genome (Liddiard et al. 2016; Cleal et al. 2018). This fusion propensity notably increases in cells lacking TP53-mediated mitotic checkpoint control. Moreover, the frequency of fusion events is heavily influenced by the cellular proficiency in classical (C-) and alternative (A-) non-homologous end joining (C-/A-NHEJ), the latter also known as theta-mediated end joining (TMEJ). Intra-chromosomal joining, in contrast to inter-chromosomal joining, exhibits less dependence on LIG4 and instead involves both LIG4 and LIG3 (Liddiard et al. 2016). Further explanations of NHEJ and other end joining mechanism will be given in the later Recombination Pathways section.

1.2.3 Micronuclei pathway

Micronuclei are another potential product formed during telomere crisis, resulting from acentric chromosome fragments which are prone to mis segregation (Hoffelder et al. 2004). Replicative stress applied to the micronuclei can cause DNA damage and substantial chromosome fragmentation which when coupled with non-homologous end joining results in chromothripsis (Hatch and Hetzer 2015; Ye et al. 2019). Spatial segregation from the main nucleus offers a plausible explanation for why this process produces the localised nature of chromothripsis (Dewhurst 2020). Aberrant nuclear envelop assembly can also lead to defective DNA replication and loss of envelop integrity leading to large DNA damage via unknown mechanisms (Liu et al. 2018). It has been shown through live-cell imaging that whilst micronuclei do not form immediately after bridge breakage, the frequency of micronuclei formation after the following mitosis was observed in between 52% and 65% of

cells with chromatin bridges. Cells without chromosome bridge division within the same imaging dish treated identically did not produce micronuclei (Umbreit et al. 2020). Additionally, depletion of TRF2 using siRNA and mitotic checkpoint barriers (Mps1 inhibited with reversine and hesperadine) combined can also cause chromothripsis (Mardin et al. 2015). The complete mechanism and pathway involved in chromothripsis are not fully understood (Cleal et al. 2019). Recently, Engel et al have shown through CRISPR-Cas9 screening that inactivation of the Fanconi anaemia (FA) pathway inhibits chromosome shattering within micronuclei during mitosis without affecting interphase-associated defects. This occurs due to engagement of the FANCI-FANCD2 with under replicated micronuclear chromosomes via the FA core complex during mitosis, leading to increased SLX4-XPF-ERCC1 endonuclease activity that induces large-scale cleavage of intermediaries made during DNA replication. This cleavage subsequently encourages POLD3-dependent DNA synthesis to prepare the resulting shattered fragments for reassembly in the following cell cycle (Engel et al. 2024).

1.2.4 Recombination pathways

Double stranded DNA breaks (DSBs) occur involuntarily as the result of exposure of cells to exogenous agents like radiation or some chemicals, as well as endogenous processes such as DNA replication (and to a lesser extent repair), factors like oxidative stress, and as mentioned previously through telomere erosion leading to the loss of the end-protection function. It is worth noting DSBs also occur deliberately during meiosis, however as this does not relate to cancer or their complex genomic rearrangements will not be discussed further (Murakami and Keeney 2008). During replication, if a replicative fork encounters a single stranded DNA break (SSB), the polymerase can stall leading to the fork collapsing which results in a DSB. Aberrant secondary structures, bulky or oxidative lesions, abasic sites, inter-strand crosslinks,

or transcription machinery can also act as obstacles for the replication fork resulting in stalling (Mirkin and Mirkin 2007). It is possible for a stalled fork to regress, which can cause displacement of the strands, leading to 3' ends of the leading strand to anneal to the 5' end of the lagging strand. This creates a "chicken foot" shaped Holliday junction which when cleaved results in a DBS (Cannan and Pederson 2016).

The DNA damage response (DDR) upon detecting DSBs depending on severity will arrest the cell cycle to allow time for damage to be repaired, or trigger apoptosis or arrest preventing severely damaged cells from proliferating further. Arrest of the cell cycle is also known as senescence or M1 phase, and widespread cell death by apoptosis is also called crisis or telomere crisis (M2 phase) (Cong et al. 2002). Repair of damage can take the form of homologous recombination (preferred if damage encountered in S/G2 phase) or non-homologous end joining (G0/1 phase) (Shrivastav et al. 2008).

1.2.4.1 Homologous recombination (HR)

Homologous recombination (HR) is a method by which cells can exchange genetic material between similar double stranded (ds) or single stranded (ss) DNA sequences. It plays a role in accurate repair of sites of DSBs, although is also responsible for creating new combinations of DNA in meiosis in eukaryotes, and horizontal gene transfer in bacteria generating genetic variation (Alberts et al. 2002; Murakami and Keeney 2008; Blakely 2015). Firstly, the MRN complex binds to both sides of the DSB, where CtBP (C-terminal binding protein) interacting protein (CtIP) is recruited to initiate resection of the 5' ends. Bloom syndrome protein (BLM) is then recruited to open the dsDNA so exonuclease 1 (EXO1) and DNA2-like helicase (DNA2L) can continue the trimming (Mimitou and Symington 2009; Stracker and Petrini 2011). The ssDNA of the remaining 3' end is then bound by replication protein A (RPA) and in a process mediated by several proteins including DNA repair protein RAD51 homolog 1

(RAD51) able to invade loci of similar DNA sequence where a break has not occurred forming a displacement loop. Upregulation of HR during the S and G2 phases of the cell cycle allows utilisation of sister chromatids for the template strand (Fugger and West 2016). Extension of the 3' end converts the displacement loop into a Holliday junction and DNA synthesis continues on the invading strand to restore the strand of the displaced homologous chromosome. From here either the double strand break repair (DSBR), synthesis dependant strand annealing (SDSA), break induced replication (BIR), or single strand annealing (SSA) pathways can resolve the process (Krejci et al. 2012).

In DSBR, the second 3' overhang that was not involved in invasion also forms two Holliday junctions (dHJ) with the homologous chromosome. The two junctions are then either cut on one DNA strand by nicking endonucleases which can result in both a crossover and non-crossover end product, or the dHJ are resolved via dissolution where they migrate towards each other and fuse or collapse (Bizard and Hickson 2014). SDSA in contrast always results in a non-crossover product, as the newly synthesised 3' end anneals to the other 3' overhang of the damaged chromosome and ligated together (Li and Heyer 2008).

Break induced replication (BIR) is distinct from the previous two pathways in its formation of a replication fork, leading to unidirectional synthesis (~6x slower than S-phase replication) from the break point (Malkova and Ira 2013). The exact mechanism is not known for humans, although it is shown minichromosome maintenance protein complex (MCM2-7) acts as the main helicase (with RPA, POLD1, and PCNA undertaking synthesis). BIR plays a role in repairing broken replication forks and contributes to the lengthening of telomeres in the absence of telomerase (Liu and Malkova 2022). It is also a source of chromosomal rearrangements, as BIR invasions of non-homologous chromosomes can lead to non-reciprocal translocations. Initiation of BIR at sites of microhomology (MMBIR) can also lead to copy number variations and insertions (Hastings et al. 2009). It has been

demonstrated that BIR can also engage in template switching, where multiple rounds of strand invasion, synthesis, and dissociation occur (Smith et al. 2007).

SSA repairs DSBs between two repeats, not involving another DNA molecule, instead annealing to itself. RAD52 homolog (RAD52) binds to the flanking repeat sequences of a break enabling the two single stranded complimentary repeats to anneal following resection by CtIP. This process can create overhanging 3' ssDNA tails which require removing in a process mediated by a ERCC1/XPF complex bound and moderated by RAD52 (Bhargava et al. 2016).

The common cancer biomarkers tumour supressor BReast CAncer genes (BRCA) 1 and 2 products have also been shown to play a role in homologous recombination (Stewart et al. 2022). BRCA1 interacts with BARD1 through a RING domain that is present at the N-terminus of both proteins. A coil near the C-terminus of BRCA1 also enables an interaction with PALB2, and a BRCT domain also at the C-terminus interacts with proteins like CtIP. PALB2 has a WD40 domain at the C-terminus that interacts with an N-terminal domain on BRCA2 which also contains BRC repeats that has activity with RAD51 which mediates loading with RPA-coated ssDNA. BRCA2 also contains DNA-binding domains that are not necessary for HR but likely play a role in optimising HR activity (Prakash et al. 2015).

1.2.4.2 Non-homologous end joining (NHEJ)

Non-homologous end joining (NHEJ) is another pathway responsible for repairing double strand breaks in DNA (Stinson and Loparo 2021). Instead of using a template, NHEJ directly ligates broken ends together. Due to lacking the need for a template, NHEJ can occur in both proliferating and non-proliferating cells. Accurate repair via NHEJ occurs if microhomologies of small single stranded (ss) overhangs on the ends are complimentary and guide the ligation of the break. However, there are several other sub-pathways that cover

cases where DSBs are not as accommodating. DSBs can result in: blunt ends, resection-dependant compatible ends, incompatible 5' and 3' ends, and 3'-phosphoglycolated ends, each configuration being dealt with in an iterative manor allowing a flexible pathway to resolution, but each having a preferred most efficient method (Chang et al. 2017). For all sub-pathways, NHEJ is initiated by the binding of the Ku70/80 heterodimer (Ku) to the DSB sites, which encircles a 3-4bp region by connecting to the sugar-phosphate backbone (Walker et al. 2001). Ku acts as a recruitment/loading point for other proteins involved in the end joining process. DNA-dependent protein kinase catalytic subunit (DNA-PKcs) is one such protein that has a high affinity for Ku–DNA ends which binds with a C-terminal portion of Ku80 to form the DNA-PK complex (Yin et al. 2017).

In the simplest case of two blunt ends, no end processing is required, therefore only the ligase complex is recruited. A region near the C-terminus of DNA ligase IV (LIG4) interacts with Ku, where LIG4 also binds to X-ray repair cross-complementing protein 4 (XRCC4). The role of XRCC4 is to stimulate the catalytic activity of LIG4 to ligate the broken DNA ends (Grawunder et al. 1997). XRCC4 also interacts with XRCC4-like factor (XLF) which is thought to steady the positions of the DNA ends pre ligation (Pannunzio et al. 2018). Paralog Of XRCC4 And XLF (PAXX) also interacts with Ku70 in this complex and is thought to act as a stabiliser for Ku (Tang et al. 2022).

For every other sub-pathway, the nuclease complex is also recruited to deal with excess incompatible ssDNA ends. ARTEMIS is one of the nucleases that performs this function as it has intrinsic 5' exonuclease activity on ssDNA, and in complex with DNA-PKcs additionally has endonuclease activity for 3' overhangs (Li et al. 2014). Aprataxin and PNKP-like factor (APLF) is also suggested as another nuclease that can participate in NHEJ (Macrae et al. 2008; Cherry et al. 2015). Resection for NHEJ is much shorter than for HR. For incompatible 5' end and resection-dependant microhomology, ARTEMIS or APLF is

recruited to resect the overhanging flaps that are incompatible to create either blunt ends or overhangs with microhomology which are then ligated by LIG4. A similar process is done for 3'-phosphoglycolated ends, however tyrosyl-DNA phosphodiesterase 1 (TDP1) is recruited first to remove glycolate from the 3' end (Li et al. 2017).

The last complex involved in the NHEJ process is the polymerase complex. This involves incorporating of a PolX family member (mu or lambda) directly to Ku though their N-terminal BRCA1C terminus (BRCT) domains (Ma et al. 2004). Both types exhibit the ability to add nucleotides with and without a template, however, mu seems better in template independent situations (McElhinny et al. 2005).

Microhomology-mediated end joining (MMEJ) also known as alternative (non-homologous) end joining (ALT-EJ/A-NHEJ/T-NHEJ), is similar yet distinct to single strand annealing in HR and NHEJ. There is evidence that suggests MMEJ is both plays a dedicated and back-up role as a DNA repair (Patterson-Fortin and D'Andrea 2020). What is clearer however is that mutagenic repair MMEJ likely contributes to genomic plasticity and driver of carcinogenesis. Differences between MMEJ and other mechanisms is the involvement of LIG3, POLQ, and PARP1 which are not required in HR or NHEJ (Sfeir and Symington 2015). Studies have shown that MMEJ may in some cases be responsible for repair and fusion of uncapped telomeres (Maser et al. 2007; Letsolo et al. 2010). There is evidence that suggests that MMEJ occurs within the G2 stage of the cell cycle and is regulated by the 9–1-1 complex (RAD9A-HUS1-RAD1) alongside RHINO (Brambati et al. 2023). Recombination in all its forms are implicated in generating genomic complexity.

1.3 Genomic complexity analysis methods

1.3.1 Cytogenetic assays

Cytogenics has a rich history stemming back to the 1870s, where Walther Flemming was the first person to use aniline dye to visualise chromosomes during mitosis. His observation of thread-like structures in nuclei was the first observation of what he named chromatin (n.b. the term chromosome was later coined in 1888 by Waldeyer-Hartz using a similar technique) (Paweletz 2001). Grigorii Levitsky later in the early 20th century was responsible for defining karyotype as the appearance of chromosomes, as well as the term ideogram (Rodionov 2009). However, it took until 1956 for Tijo and Levan to formalise the process by: treating cells with a hypotonic solution (to swell cells and spread out the chromosome), using colchicine to arrest cells during metaphase, then squashing the cells to release the chromosomes (Tjio and Levan 1956). Only three years later, Lejeune found that down syndrome was caused by trisomy 21 (Lejeune et al. 1959).

Since then, several techniques for studying the karyotypes of cells have been established. The first involves karyotyping with a stain to create bands within the chromosome, these include: Quinacrine (Q-banding, the first), Giemsa (G-banding, most common, adenine-thymine binding), Reverse banding (R-banding, opposite of G), Constitutive heterochromatin (C-banding, centromere binding), and silver Nuclear Organising Region staining (NOR, rDNA binding). Distinct bands allow chromosomes to be identified, easily allowing detection of aneuploidy. Less obviously, karyotyping with stains can also reveal structural variants such as deletions, duplications, and translocations (Sumner 2001; Clare O'Connor 2008b; Spinner 2013; Spinner et al. 2013; Ozkan and Lacerda 2024).

Another form of cytogenetic analysis is fluorescent in situ hybridisation (FISH). FISH is a more targeted approach for cytogenetic analysis, using a fluorescent probe which is used by denaturing and annealing to target DNA, which can then be observed using a fluorescent microscope (Shakoori 2017). One form of FISH uses multiple colours (multiflour/chromosome painting) where a collection of hybridisation probes relating to a chromosome are labelled with corresponding colours (Lauter et al. 2011) . However, due to relatively low resolution, chromosome painting allows identification of translocations and only large duplications and deletions (Clare O'Connor 2008).

Comparative genomic hybridisation (CGH) is another technique that utilises fluorescence. It is used to detect copy number variations (CNV) by mixing green labelled tumour DNA with red labelled normal DNA in a 1:1 ratio with a normal metaphase reference (from healthy patient) (Nair and Gonzalez-Angulo 2015). The competition to hybridise with the reference creates a profile of green and red along the chromosomal axis, where the ratios can then be analysed with imaging processing software to generate a copy number profile (where green:red ratios < 1 shows losses, > 1 gains). Theoretically a gain or loss of a single chromosome should be 2 and 0.5 respectively, but these values have not been seen experimentally. Sensitivity of CGH can be an issue, especially if the tumour DNA has normal contaminants. The resolution is also somewhat limited, with +50% CNVs requiring a 2mb region, and total loss (-100%) a 1-2mb region to be detected (Weiss et al. 1999). A related methodology is used in microarrays.

Microarrays are microscope slides that contain many fixed DNA sequences (often genes) in set positions. Messenger RNA (mRNA) from a tumour and normal sample are collected and converted into complimentary DNA (cDNA) and labelled with two fluorescent probes. Similar to CGH, the ratios of the two colours can then be analysed to determine whether the genes expression is higher or lower in the tumour sample compared to the

normal (Shaffer et al. 2007).

Patterns of genomic complexity started to emerge when using these techniques in analysing cancer samples. Specifically, when looking at aneuploidies, commonalities were seen across a range of cancer types such as gains in chromosomes 1q, 3q, 8q, 20, (5p, 7, 11q, 12p, 13, 17q, 21 less commonly) and losses in 3p, 8p, 17p, (1p, 4, 5q, 6, 9p, 13, 16q, 18, 19, 22 less commonly) (Cai et al. 2012). A worthy note that breast cancer in particular has characteristics of gains in 1q, 8q, 17q, 20, and losses in 8p, 16q, 17p (Nicholson and Cimini 2013). The effects of aneuploidy in the context of cancer appear to be granting resistance to apoptosis through changes in gene expressions and causing mis-segregation of chromosomes to create new karyotypes which generate selective advantages during chemotherapy (Replogle et al. 2020).

Outside of aneuploidies, a common translocation found across several cancer types such as Ewing sarcoma, leukaemia, prostate, and breast cancer involves Erythroblast Transformation Specific (ETS) coding domains. The ETS family involves proteins responsible for regulating cellular differentiation, cell cycle control, cell migration, proliferation, apoptosis, and angiogenesis, explaining why it might be frequently mutated in cancer cells. A more specific example of this is the t(12;15)(p13;q25) translocation that results in the ETV6-NTRK3 (EN) fusion oncoprotein found in breast cancer (Li et al. 2007).

A modern technique utilising optical genome mapping (OGM) developed by BioNano Genomics has improved on previous methods touting 10,000X higher resolution than karyotyping. Investigations in haematological cancers using OGM have suggested this technology is powerful enough to replace cases where a combinations of karyotyping, FISH, and CNV microarrays were previously required for analysis (Neveling et al. 2021).

1.3.2 High-Throughput Sequencing

Large technological advances lead to the creation of what is now known as next generation sequencing (NGS), which can sequence whole genomes (WGS) in parallel with a high depth in a relatively short time (Goodwin et al. 2016). There are two forms of NGS, long and short read sequencing. Short read sequencing all generally involves a similar process of fragmenting DNA (using sonication or enzymes) pieces and attaching adapters to the fragments to create reads that are typically 100-150bp (although can be between 50-700bp). These sequences are then sequenced in parallel, with quality control measures to discard low quality reads (to generate fasta/fastq files). These sequenced reads are then aligned (mapped) to a reference genome (to create a bam file). This form of sequencing has the advantage of being widely available, low cost (depending on sequencing depth), and accurate. However, they are prone to errors in mapping created by highly repetitive regions.

Examples of short read sequencing technologies include: Illumina (MiniSeq, NextSeq, MiSeq, HiSeq 2500/3000/4000/X), BGI (BGISEQ-50, MGISEQ 200, BGISEQ-500), which are the two most common due to their cost per sequencing run. Other less common types of short read sequencing include Ion torrent, GenapSys, and pyrosequencing, but due to them being outdated and their low adoption rate will not be discussed further (Ronaghi et al. 1996; Rothberg et al. 2011). Illumina is the dominating short-read sequencing platform, it works off a sequencing by synthesis (SBS) principal, with two variants of single and paired end sequencing (Kircher and Kelso 2010; Goodwin et al. 2016). It works by first attaching purified DNA fragments to a flow cell using adaptors (containing a terminal sequence, adapter, index, and primer) which bind to their compliments attached to nanowells on the flow cell. Once attached, bridge amplification PCR is used to make copies of each fragment. Using a cycle of washes, reversible

fluorescent blocking nucleotides are incorporated one at a time, where a camera determines the colour (and therefore the "letter") of the added base, before the block is removed to allow the polymerase to add the next base, and so on in a cycle (figure 1.4) (Clark et al. 2019). As their name implies, single read sequence involves sequencing from only one end, whereas paired end sequencing sequences from both. Paired end sequencing is used for WGS due to its enhanced ability to detect genomic rearrangements, insertions-deletions (indels), gene fusions, and novel transcripts (https://emea.illumina.com/science/technology/next-generation-sequencing/plan-experiments/paired-end-vs-single-read.html#:~:text=Single%2Dread%20sequencing%20involves%20sequencing,%2Dquality%2C%20alignable%20sequence%20data.).

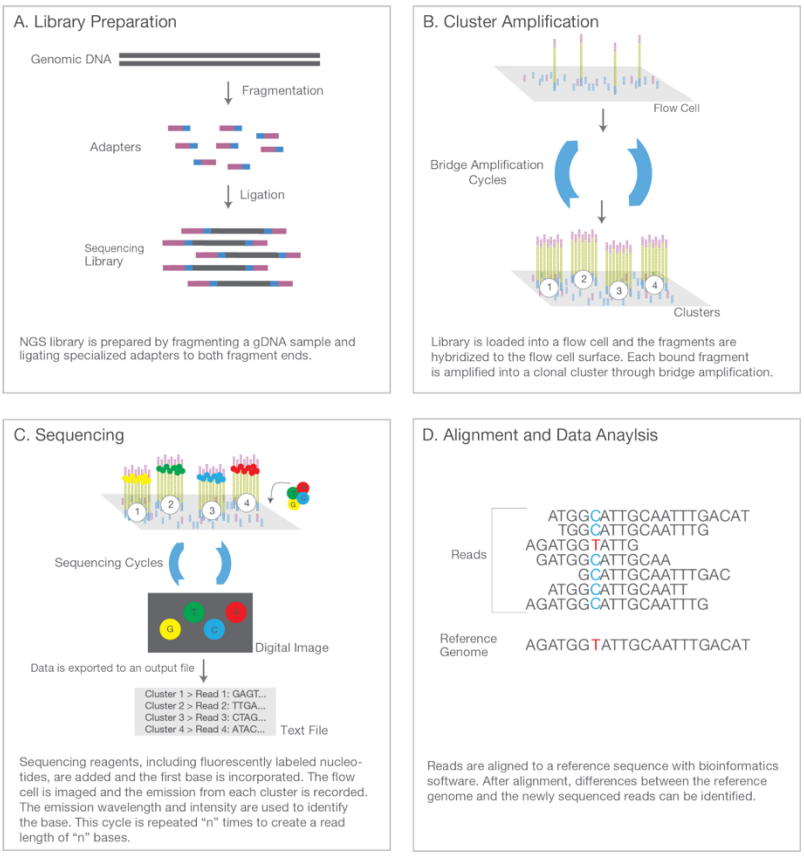


Figure 1.4 Diagram to show the four stages of Illumina sequencing: (A) Library Preparation – adapters ligated to fragmented DNA, (B) Cluster Amplification – amplification used to increase amount of DNA in each cluster, (C) Sequencing – cycles of base incorporations, imaging and base recording, and washing, (D) Alignment and Data Analysis – recorded reads are aligned to reference genome. Figure from "An introduction to Next-Generation Sequencing Technology" by Illumina available at: https://www.illumina.com/content/dam/illumina-marketing/documents/products/illumina_sequencing_introduction.pdf

By contrast, BGI(/MGI) sequencing utilises DNA nanoball technology (Xu et al. 2019). Small fragments (100-350bp) are ligated with an adapter sequence that circularise them. High-fidelity Phi 29 DNA polymerase is then used to create extremely low error rate, and GC unbiased copies through rolling circle replication to create a long-concatenated strand. These long strands are compacted into DNA nanoballs (DNBs) (figure 1.5), which are loaded into the sequencer and are attached to flow cells with evenly spaced positively charged sections that hold the negatively charged DNBs (Drmanac et al. 2010). Like Illumina sequencing, fluorescently labelled dNTPs are incorporated one at a time in a cycle of washes whilst a laser and camera records the type of the incorporated base in between each cycle. Following the first round of sequencing, paired end sequencing is achieved by producing a second strand by introducing a second primer, and a polymerase with stand displacement activity. This process is optimised to make the longest possible strand that stays attached to the original DNB (https://en.mgi-tech.com/products/resources). Both Illumina and BGI sequencing utilise the Phred 33 standard, which uses the signal intensity form the dNTP incorporation to determine the quality of the base called which is added as the last line of a read call in the fastq file format (Ewing et al. 1998).

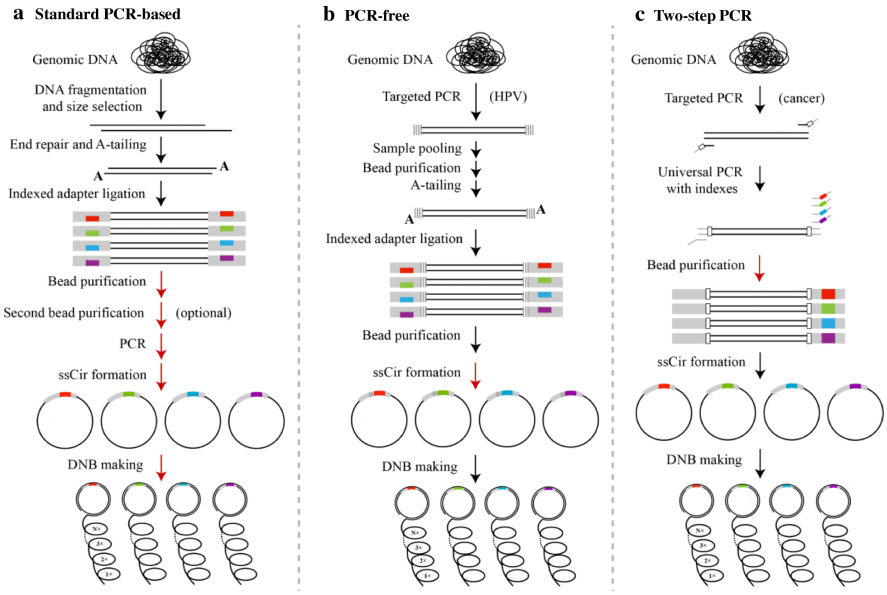


Figure 1.5 Figure displaying three methods of library preparation for BGI sequencing including: (a) Standard PCR-based, (b) PCR-free, (c) Two-step PCR. The end product of each method is labelled DNA nanoballs (DNBs) which are later attached to a flow cell and sequenced. Figure reused under the Creative Commons Attribution 4.0 International License from: Li, Q., Zhao, X., Zhang, W. et al. Reliable multiplex sequencing with rare index mis-assignment on DNB-based NGS platform. BMC Genomics 20, 215 (2019). https://doi.org/10.1186/s12864-019-5569-5 (figure 2).

More recently long read sequencing is becoming more popular due to its capability of detecting previously inaccessible structural variants (as well as drop in cost and increase base calling accuracy). Issues in structural variants will be discussed in more detail in the next section. Long read sequencing also permits more complete assembly of genomes. Examples of long read sequencing technologies include those available from PacBio, Oxford Nanopore, although Illumina also have developed a synthetic long-read technology (Marx 2023).

PacBio HiFi sequencing uses DNA fragments that are up to 100kb in length and utilises single-molecule real-time (SMRT) technology. Fragments are converted into topologically circular DNA (termed SMRTbell) templates by shearing, removing single stranded (ss) overhangs, repairing DNA damage, A' end tailing, and ligation of an adapter (figure 1.6.1) (Travers et al. 2010). Sequencing is performed by binding a DNA polymerase and loading the SMRTbells onto an SMRT cell into millions of zero-mode waveguides (ZMWs), where once again fluorescent dNTPs are incorporated and the wavelengths of light emitted are measured to call bases (figure 1.6.2) (Logsdon et al. 2020).

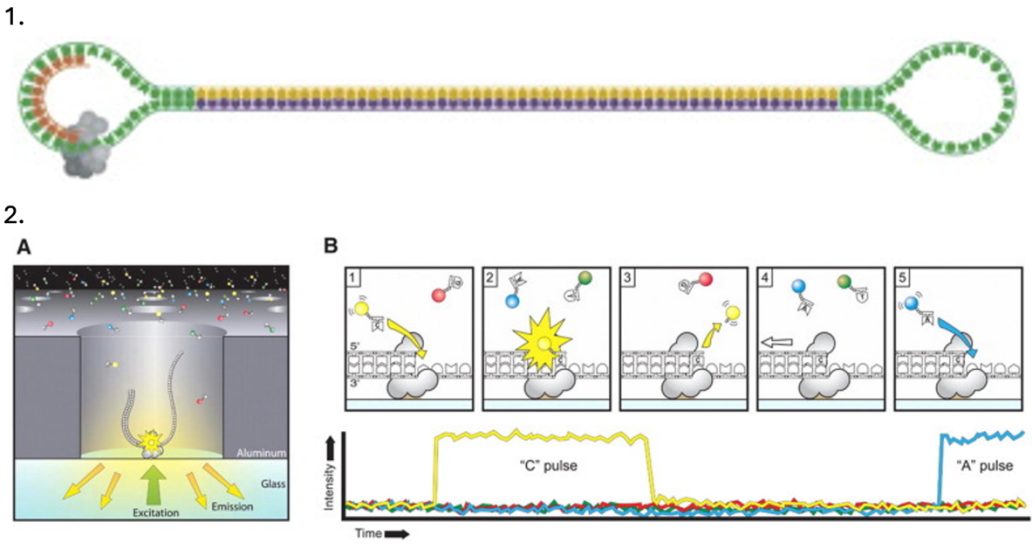


Figure 1.6 (1) SMRTbell structure – Double stranded DNA (yellow and purple) are manipulated into a closed circle by the ligation of hairpin adaptors (green). The polymerase (grey) is anchored to the bottom of a ZMW where it integrates fluorescent bases (orange). (2.A.) Visual representation of a zero-mode waveguides (ZMW), with a SMRTbell anchored by the polymerase at the base. (2.B.) The cycle of fluorescent dye incorporation displayed (1-5) with the output signal shown below. Figures adapted under the Creative Commons Licence from: Rhoads, A., Fai AU, K. PacBio Sequencing and Its Applications. Genomics, Proteomics & Bioinformatics 13(5) p. 278-289 (2015)

Unlike PacBio and other methods, Oxford Nanopore sequencing does not use fluorescence. Oxford Nanopore sequencers are composed of a synthetic membrane that holds a DNA pore. The pore protein along with an electric current feeds DNA through the poor as a single strand. The DNA bases resident in the pore at a given time, cause a distinct disruption to the current across the membrane, allowing real time-analysis of the current to infer the sequence during transit (figure 1.7) (Wang et al. 2021). This technology can sequence reads that are megabases long (https://nanoporetech.com/platform/technology).

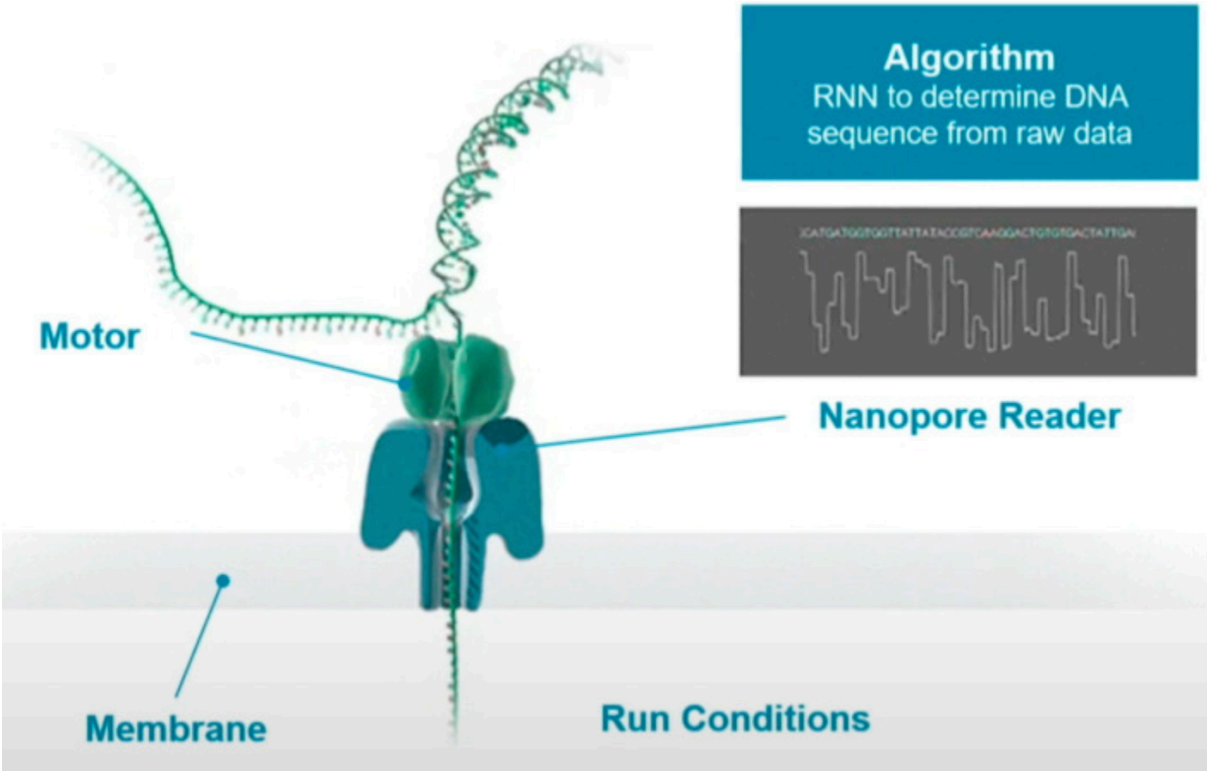


Figure 1.7 Diagram to show the process of nanopore sequencing. A motor protein pushes a single strand from a double stranded DNA molecule through a nanopore reader which measures the current as each nucleotide passes through. Bases can then be called as the structure of each nucleotide is distinct. Figure reused under the Creative Commons Licence from: Lin B, Hui J, Mao H. Nanopore Technology and Its Applications in Gene Sequencing. Biosensors (Basel). 2021 Jun 30;11(7):214. Doi: 10.3390/bios11070214. PMID: 34208844; PMCID: PMC8301755. (figure 2)

1.3.3 Public repositories

Another technological advancement in recent years that has contributed to discoveries in genomic complexity (as well as many other areas of biological/genetic studies) is the more widespread adoption of public repositories. They offer a platform where researchers can access and collaborate in processing larger and more diverse datasets in addition to promoting re-use and re-analysis of datasets to drive further discoveries. Cohorts containing a wide range of analyses such as WGS and microarray data are accessible and can have clinical data from the original trial accompanying them. Perhaps the most famous of these are the ensemble genome databases from the European Bioinformatics Institute, and the NCBI associated databases, which contain a wide range of genetic data including high-quality reference genomes and their annotations for a diverse collection of organisms from yeast to drosophila to mice to humans and many in between (Birney et al. 2004; Yates et al. 2020). Vast amounts of genomic data from patients are also available in public repositories such as The Cancer Genome Atlas (TCGA) which contains more than 2.5 petabytes of data from many different types of analysis such as somatic mutation, RNA and proteins express, and DNA methylation data (*The Cancer Genome Atlas Program (TCGA) - NCI* 2022). However, due to General Data Protection Regulation (GDPR) (and the U.S. Privacy Act) many databases containing sensitive information like WGS such as the International Cancer Genome Consortium, Genomics England, and GenBank employ strict restrictions requiring an application before accessing and publishing of data. To condense the useful information from these large WGS data sets, several methods are employed to analyse them.

1.3.4 Analysis of high throughput sequencing

Structural variants (SV) are defined as genomic rearrangements like insertions, deletions,

duplications, inversions and translocations (over arbitrarily 50bp in size). SV calling is the analysis of reads from WGS data in the form of a binary alignment (bam) to create a set of structural variants for a sample, and is usually presented using as a variant call format (vcf) file (Li et al. 2009; Danecek et al. 2011). There are several programs that are used for SV calling including for short read data: manta, gatk, strelka, delly, lumpy, dysgu, and long read data: nanovar, svim, sniffles, and again dysgu (Rausch et al. 2012; Saunders et al. 2012; Layer et al. 2014; Chen et al. 2016; Cleal and Baird 2022; *Structural variant (SV) discovery* 2024). Short read data presents additional challenges when attempting to call structural variants. Due to their limited read length, it is only possible for SVs to be called from discordant mappings, changes in read-depth, and split reads around breaksites (or through more computationally expensive assembly). Repetitive sequences also provide a challenge when mapping short reads which compounds onto these issues for SV callers. Additionally library preparation issues can also hinder SV calling. An example of this is PCR-free short read data show up to a twofold reduction of coverage of loci exceeding a GC% higher than 45% (Logsdon et al. 2020). This limits SV discovery in regions such as telomeres, centromeres, and acrocentric genomic areas which commonly exhibit large tandem repeats.

Aside from SV calling, WGS data can be used to create a copy number profile with a higher resolution than karyotyping methods. This is achieved through extracting and analysing the read depth across the genome. However, this type of analysis is also hindered in repetitive regions where mapping is difficult such as telomeres and centromeres. There are a diverse range of tools for carrying out such analysis that provide different levels of insight. The staple WGS tool samtools has its own inbuilt function to simply create a copy number table displaying the depth across each position in the reference (Li et al. 2009; Danecek et al. 2021). Other software such as CNVnator apply statistical techniques like correcting GC-biases, mean shifting, and partitioning to take read mapped data (bam file) to

detect copy number variations (Abyzov et al. 2011).

1.3.5 Chromoanagenesis and other complex patterns

The advent of NGS and the resulting methods of analysing WGS data opened the door to new discoveries about the previously unseeable level of complexity of cancer genomes, and patterns in these complexities emerged. Chromoanagenesis is the umbrella term for the phenomena of chromothripsis, chromoplexy, and chromoanasynthesis which describe complex patterns of structural variants/genomic rearrangements (figure 1.8) (Ostapińska et al. 2022; Pellestor et al. 2022).

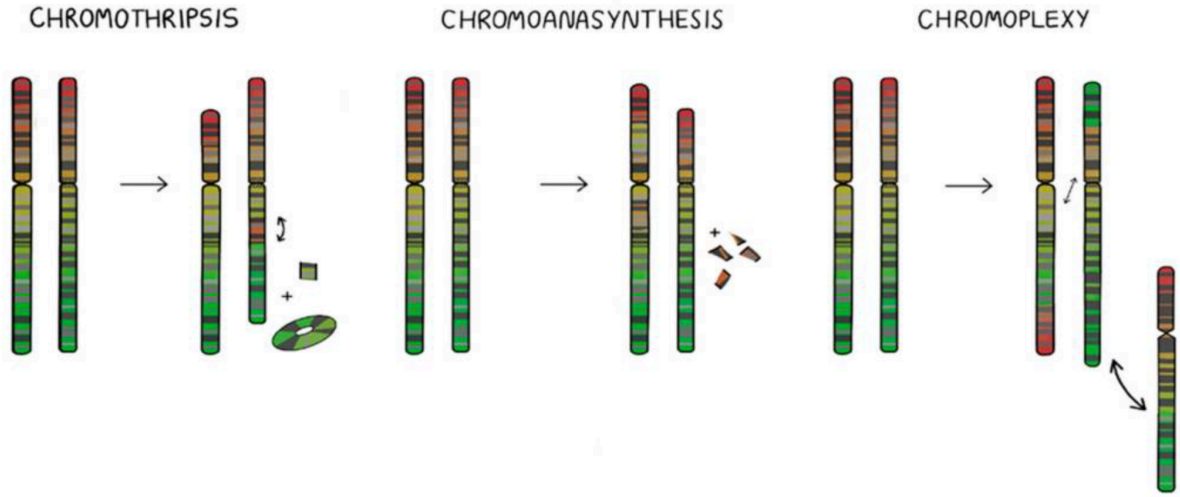


Figure 1.8 Graphic representation of the complex patterns: chromothripsis, chromoanasynthesis, and chromoplexy under the umbrella name Chromoanagenesis. Figure resused under the Creative Commons Licence from: Ostapińska K, Styka B, Lejman M. Insight into the Molecular Basis Underlying Chromothripsis. Int J Mol Sci. 2022 Mar 19;23(6):3318. Doi: 10.3390/ijms23063318. PMID: 35328739; PMCID: PMC8948871. (figure 1)

1.3.5.1 Chromothripsis

Chromothripsis is a chromosomal rearrangement pattern that is thought to result from a single catastrophic chromosomal shattering event, followed by disorderly repair of the resulting fragments within a cell. It often occurs early in tumour development and can lead to segments of chromosomes being affected by multiple mutations and rearrangements that encourages

carcinogenesis (Forment et al. 2012). Chromothripsis is present in a diverse range of tumour types (commonly in cells lacking p53) and has been linked with poorer prognosis in cancer patients (Jones and Jallepalli 2012; Hayashi et al. 2015; Cortés-Ciriano et al. 2020). Chromothripsis is also observed in germline cells, and normal somatic tissues, however generally its contributions to the phenotypes of individuals is unknown (Middelkamp et al. 2017). It is speculated that asymptomatic carriers experience infertility when two or more chromosomes are involved (Nazaryan-Petersen et al. 2020).

Chromothripsis is thought to occur through chromosome shattering and random recombination. However, a complete understanding of mechanisms that lead to chromothriptic rearrangements is lacking. Several potential pathways have been implicated. As mentioned in a previous section, one proposed origin of chromothripsis proposed by Pellman e*t al* involves erroneous segregation of chromosomes that create lagging chromosomes can lead to formation of micronuclei (figure 1.9G) (Crasta et al. 2012). DNA replication stress within micronuclei can result in DNA damage and substantial chromosome fragmentation coupled with non-homologous end joining results in chromothripsis (Hatch and Hetzer 2015; Ye et al. 2019). Aberrant nuclear envelop assembly can also lead to defective DNA replication and loss of envelop integrity leading to large DNA damage via unknown mechanisms (Liu et al. 2018). Pellman *et al* have shown through live-cell imaging that though micronuclei do not form immediately after bridge breakage, frequency of micronuclei formation was seen between 52% and 65% in cells with bridges. This contrasts with cells without bridge division within the same imaging dish treated identically not producing micronuclei (Umbreit et al. 2020).

The original report of chromothripsis also implicated the role that telomere dysfunction and the resulting breakage-fusion-bridge (BFB) cycles play in the generation of chromothripsis (Stephens et al. 2011). Additionally, De Lange *et al* have explored the

impact of telomere crisis on chromothripsis development. They simulated crisis through using dox-inducible dominant negative allele of TRF2, which prevents the protection function of telomeres therefore stimulating telomere fusions. This created chromatin bridges that persisted throughout mitosis, suggesting TREX1-mediated fragmentation of dicentric chromosomes formed in telomere crisis followed by DNA damage repair mechanism is a potential origin of chromothripsis in cancer (figure 1.9Ciii) (Maciejowski et al. 2015). Depletion of TRF2 using siRNA and mitotic checkpoint barriers (Mps1 inhibited with reversine and hesperadine) combined can also cause chromothripsis. The complete mechanism and pathway involved is not fully understood (Mardin et al. 2015; Cleal et al. 2019).

A review of these various mechanisms for chromothripsis generation from Cleal and Baird *et al* 2020 paints a clearer holistic picture of how telomere dysfunction potentially drives genomic instability from simple large deletions and insertions to complex Chromothriptic patterns. It was highlighted that a single dysfunctional telomere is sufficient to trigger a cascade of events that leads to chromothripsis, through micronuclei-dependent and -independent pathways that utilize alternative and non-homologous end-joining or replicative repair pathways (Cleal and Baird 2020). They also emphasise there is growing evidence from multiple sources that complex patterns which are difficult to explain by chromosome shattering alone, likely points to replicative repair playing a key role in generation of complex events that resemble chromothripsis (Umbreit et al. 2020).

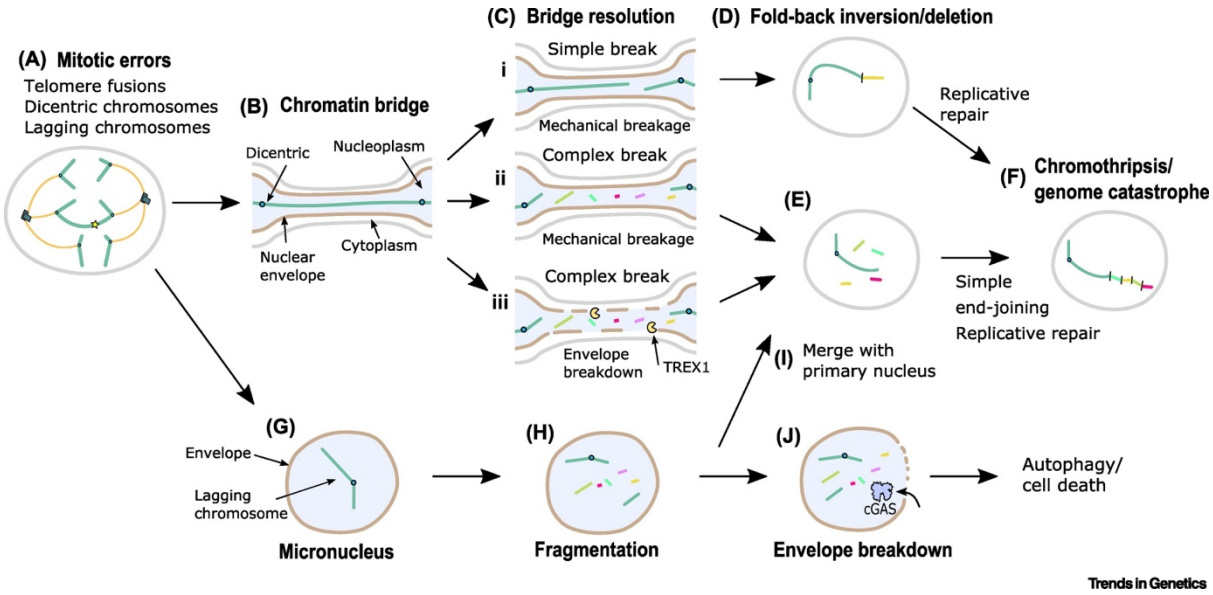


Figure 1.9 Mechanisms proposed which implicate telomere dysfunction and its' resulting telomere fusions in chromothripsis development (F). (A) Mitotic errors from telomere fusion can produce a variety of aberrant outcomes such as dicentric and lagging chromosomes. (B) Dicentric chromosomes for chromatin bridges between daughter cells which can be resolved by (Ci) a simple break, (Cii) a complex break, (Ciii) complex break involving TREX1 breakdown of the nuclear envelope. (D) Simple breaks can lead to fold back inversions or large terminal deletions. I Complex breaks can leave fragmented chromosomes. (G) Dicentric chromosomes or lagging chromosomes can also be enveloped in micronuclei. (H) Micronuclei can cause fragmentation through premature compaction of DNA during mitosis, or defective DNA replication and repair. Fates of broken chromosomes within micronuclei include subsequent rounds of micronuclei entrapment and DNA breakage in granddaughter cells. (I) Another fate involves reintroduction to the main nucleus. (J) Alternatively, its possible for envelope breakdown to cause cGAS localization which leads to autophagy or apoptosis. Figure reused with permission by the authors from: Cleal, K,. Baird, D. Catastrophic Endgames: Emerging Mechanisms of Telomere-Driven Genomic Instability. Trends in Genetics 36(5) p. 347-359 (2020). (figure 3)

1.3.5.2 Chromoplexy

Chromoplexy is similar to chromothripsis, with both displaying breaks and fusions suspected to be caused primarily by non-homologous end joining (NHEJ), although also has been shown to result from alternative end joining (alt-EJ). The distinction is chromoplexy involves less breakpoints (<100) which are not locally clustered on one or two chromosomes, instead involving multiple chromosomes (Shen 2013). Fusion breakpoints of chromoplexy are predominantly precisely joined or contain a ~2bp microhomology which create chimeric chromosomes with minimal gains or losses likely attributable to C-NHEJ or alt-EJ repair of DSBs, but can also feature large deletions between chain fusion junctions (Berger et al. 2011;

Zepeda-Mendoza and Morton 2019).

Originally described in prostate cancer by Baca et al, chromoplexy exhibits a correlation to tumours containing oncogenic ETS fusion and functionally appears to dysregulate cancer genes in early carcinogenesis. Mutations that occur early within carcinogenesis such as deletions of tumour suppressor genes TP53, PTEN, ETV6, NKX3-1, CDKN1B, RB1, and CHEK2 were found at different frequencies in chromoplexy chains (Shen and Abate-Shen 2010; De and Michor 2011; Goodman 2021; Liu et al. 2022b). Due to this observation, it is speculated that rearrangements and deletions across the genome present in chromoplexy simultaneously deactivate tumour suppressor genes from multiple chromosomes. It is also possible that chromoplexy also contributes to carcinogenesis via creation of oncogenic fusions and mutations leading to overexpression of oncogenes, but no recurrent gene fusions were found in the analysis. Evidence also suggests that multiple rounds of chromoplexy occur in tumour subclones implying it also plays a role in late oncogenic progression (Baca et al. 2013). The underlying mechanism that generates chromoplexy is unknown but is suspected to be distinct from that of chromothripsis, as the process is restricted to deletions and chained chromosomal rearrangements as opposed to copy number alterations (Shen 2013; Ostapińska et al. 2022).

1.3.5.3 Chromoanasynthesis

Chromoanasynthesis is defined as a complex rearrangement process of chromosomal duplications and triplications clustered on typically one (up to a few) chromosomes, with associations of various other types of structural rearrangements. The discovery of chromoanasynthesis was made through a combination of CGH, FISH and G-band karyotyping analysis which exhibited fluctuations of regions between duplications, triplications, normal, and deletions in a variety of permutations which exposed complex

clusters of structural variants (Liu et al. 2011). Unlike chromothripsis and chromoplexy, chromoanasynthesis is thought to arise via a replicative origin that involves serial fork stalling and template switching (FoSTeS) or microhomology mediated break-induced replication (MMBIR) mechanisms (Pellestor and Gatinois 2018). Despite its mechanistic origin differing from that of chromothripsis, the rapid formation and outcome of complex rearranged chromosomes is similar. Replicative fork stalling and collapse can be caused by a multitude of endogenous and exogenous events such as tRNA genes, protein-mediated fork barriers, replication "slow zones", inverted repeats, secondary DNA structures, DNA breaks, tightly bound proteins (Branzei and Foiani 2005). Following a stall, the lagging strand disengages and anneals due to microhomology of another replication fork (which could be linearly megabases apart but close in proximity in 3D space) where it is copied. Multiple rounds of disengagement and strand invasion may occur leading to complex clusters of structural variations before replication of the original strand is resumed (Lee et al. 2007; Pellestor 2019). Alongside structural variants, mutations in single nucleotides are also associated with the complex genomic patters.

1.3.5.4 Kataegis

Kataegis is the clustering of localised hypermutations and is often co-localised with somatic rearrangements. Single nucleotide polymorphisms (SNPs) within these regions are almost entirely comprised of CT mutations at TpC dinucleotides or TpCpW (W denoting A or T) trinucleotides, and also, less commonly C>G (and C>A). The mechanism for C>T mutations is caused from the deamination of cytosine to form uracil, which during DNA replication results in the base being read as a thymine (Seplyarskiy et al. 2016). Deamination is proposed to be the results of a family of apolipoprotein B mRNA editing enzyme, catalytic polypeptide-like (APOBEC) activity. APOBEC normally functions as a viral protecting agent

but is also a major factor in carcinogenesis, especially of breast cancers (Nik-Zainal et al. 2012). Telomere crisis and APOBEC activity have been shown to be linked in the process of carcinogenesis. Two members of the APOBEC family APOBEC3A/B are inferred to act upon ssDNA generated by TREX1 activity in dicentric chromosome resolution leading to Kataegis (Maciejowski et al. 2015). Most notably, APOBEC has been shown to be the aetiology (cause) of COSMIC mutational signature 2 (majoritively C>T) and 13 (majoritively C>G) (Jarvis et al. 2018). It has been shown the foci of kataegis often occur around genomic rearrangement (Alexandrov et al. 2013). These patterns have been observed using generalised conventional analysis methods, further patterns have been observed using unique approaches.

1.3.5.5 Pyrgo, Rigma, and Tyfonas

Analysis from Hadi *et al* using a novel genome graph model to investigate junction copy number (JCN) across a large cohort of whole genome sequence data was used to discover new genomic complexity phenomena (Hadi et al. 2020). This approach combines the analysis of CN junctions (determined from read depth) with variant data (determined from breakpoint data), arguing they are intrinsically coupled. By constructing a graph of neighbours with junctions being the nodes and edges based on the adjacency in the reference (REF) or introduced though a rearrangement (ALT) (with loose ends also being added to the graph). From this analysis 3 types of patterns emerged. Pyrgo are increased regions of low-JCN duplications correlating with early replicating and super-enhancer regions, most prevalent in breast and ovarian cancers. Rigma exhibit decreases of low-JCN deletions at late-replicating fragile sites, enriched in gastrointestinal carcinomas. Tyfonas are “typhoons” of high-JCN junctions and fold-back inversions enriched in expressed protein-coding fusions and breakend hypermutation, frequently found in acral melanomas (Hadi et al. 2020). It’s worth noting that there are no known mechanisms that underly these proposed patterns or whether they are

distinct from other complex genomic rearrangements. This is in contrast to for example chromothripsis, where telomeres have been connected to their origins in cancer.

1.4 Telomere length in cancer

1.4.1 Telomere length and cancer prognosis

Initially discovered in chronic lymphocytic leukaemia (CLL), fusogenic telomere length thresholds have been shown to be a highly significant and useful prognostic tool (Lin et al. 2014; Strefford et al. 2015). Telomere fusion assays were conducted using Sothern blotting with XpYp, 17p, and 21q telomere adjacent probes. This was combined with single XpYp telomere length data obtained through the single telomere length analysis (STELA) assay, which utilises a linker or 'telorette' containing seven bases of TTAGGG homology followed by a 20-nucleotide non-complementary tail to the G-rich 3′ overhang of the telomere. The telorette is annealed and ligated to the 5' end of the complementary C-rich strand of the chromosome such that primers for the telorette tail and allele specific subtelomeric regions can be used for PCR amplification and subsequent southern blotting to determine the length (Baird et al. 2003). A fusogenic threshold was then calculated by creating a range of telomere lengths extending the upper limit (3.81kb) to where telomere fusions occur and taking the mean of the range (2.26kb). It was found that telomere length was the dominant variable in multivariant analysis with respect to time to first treatment and progression-free survival (Lin et al. 2014). In later studies these same thresholds were used to predict patient responses to chemotherapy and FCR-based treatment, suggesting that telomere length could be taken into account for risk management of clinical trials surrounding CLL (Strefford et al. 2015; Norris et al. 2019). A similar study was then conducted in breast cancer which showed similar results in the stratification of patients by prognosis using telomere length. This analysis

showed that higher values (2.6kb) within the fusogenic range were also significantly ($p<0.0001$) predictive of prognosis. It was also shown that patients in this study with telomere lengths <3kb were also observed in previously established prognostic subsets like estrogen receptor, progesterone receptor, human epidermal growth factor receptor 2 status, histological grades, and NPI score (Simpson et al. 2015). These studies suggest that telomere shortening and dysfunction occurs prior to clinical progression. This is congruent with previous analysis in colorectal polyps which show a correlation with telomere shortening and aneuploidy, and another study in CLL correlating telomere length and Binet stages (Lin et al. 2010; Roger et al. 2013).

Previous analysis has also been carried out to examine the link between telomere length and genomic complexity. Santos et al determined through FISH and other cytogenetic assays that in CLL shorter telomeres (with cases of del11q/17p compared to 13q14 deltions) were associated with abnormal karyotypes (Dos Santos et al. 2015). Titia de Lange and John Maciejowski have also revealed that telomere crisis can lead to phenomena like chromothripsis, kataegis, and tetraploidisation. They argue that dysfunctional telomeres during this stage generates dicentric chromosomes that harbour many genomic rearrangements from processes such as breaking of dicentric chromosomes and BIR being used to repair them, and anaphase bridge resolution creating fragmentation followed by haphazard repair. These findings were from inducing telomere crisis through inactivation of TRF2 in p53 and RB deficient cells and suggest telomerase and ALT as mechanisms for escaping crisis (Maciejowski and de Lange 2017).

1.4.2 Aims and Hypothesis

It is known through WGS analysis of breast cancer that increasing genomic complexity gives rise to poorer prognosis (Curtis et al. 2012). Previous research has clearly shown there

is great potential in using telomere length as part of the prognostic arsenal. Other investigations have also been conducted into the correlation between telomere length and genomic complexity. However, these are much less common, and findings have sparked debate whether the correlation is a cause via mechanism such as BFB cycles, or consequence of increased proliferation of these severe cancers. Even within the parties that argue shorter telomere length is a cause of genomic complexity, questions are being asked about the mechanism by which the complexity arises. This thesis aims to take a combination of bioinformatics approaches to interrogate the complexity of cancers using WGS data paired with measured and predicted telomere length to uncover potential correlations. The aim is to elucidate patterns and correlations in the data and relate these to biological processes. The main hypothesis is that decreasing telomere length will correlate with increasing genomic complexity, most likely not continuously, but rather that there will be TL values where patients can be stratified according to genomic complexity. This building upon previous research mentioned in the previous section (1.4.1) where patient prognosis could be stratified by fusogenic thresholds. Evidence for correlations between telomere length and genomic complexity may hint at the underlying mechanisms behind large rearrangements are and might further aid understanding why this prognostic stratification by telomere length is possible.

Chapter 2 Methods

2.1 Datasets

All datasets used for analysis consisted of (n) WGS tumour-normal pairs. The dataset explored in chapters 3 and 4 is a local breast cancer cohort (n=44) sequenced with BGISeq 500 at ~15x coverage which had accompanying STELA data. Datasets used in chapter 5 were all sequenced using Illumina HiSeq (X/5000) include breast cancer cohorts (n=1591) from Genomics England (GEL) (~30-80x for the germline samples, and ~80-150x for cancer samples) and the International Cancer Genome Consortium (ICGC) (n=80) (~30-80x), as well as a chronic lymphocytic leukemia (CLL) cohort (n=98) (~30-80x germline/~80-150x cancer) with associated STELA data.

2.2 Common requirements

Multiple tools, pipelines, and bash scripts throughout this chapter including the copy number and structural variant analysis will be using similar input files which are required for the software to run correctly. The most frequently referred to input file is a csv called "sample_pairs.csv" which contains three columns: the basename (file name without extension or path) of the cancer and normal samples, and an index column describing how a sample will be ordered for display during analysis. For this analysis the index is the telomere length (TL) in kilobases, unless stated otherwise. This file is used by teltool, the "pipeline" scripts, as well as several bash scripts to pair the cancer files to their normal counterpart to perform various functions, such as normalisation which will be discussed later.

Another common file that will be referred to is named “run_list.csv”. This file has columns that contain the basename for a cancer sample, read length of bam file, coverage of the bam file, and optionally, the reference type (hg38 or hg19). The coverage of samples is used to generate a further variable which will be named “frac” (short-hand for fraction) which is used for subsampling of bam files in various later sections via samtools (github.com/samtools/samtools). For readability, anytime the variable frac has been used without being defined, at the start of looping samples it was set using `frac=$(awk -vcov=${cov} ‘BEGIN { print 40 / cov }’)`. This file can be created quickly with:

```
while IFS=, read -r line; do
   readlen=$(samtools view ${line} | head -n 10000 | gawk ‘{ print length($10)}’ | sort | uniq -x | perl -ane ‘$_ =~ s/^[]+//g;print $_’ | sort -k lnr,lnr | head -1 | cut -f2 -d “”)
   cov=$(samtools idxstats ${line} | awk -vreadlen=${readlen} ‘ { len += $2; nreads += £3 } END  { print nreads * readlen / len { ‘)
   echo “$(basename ${line} .bam),${readlen},${cov}” >> run_list.csv
done < list_of_file_paths.txt
```

Where “list_of_file_paths.txt” could be the output of `ls bam_directory > list_of_file_paths.txt` or a cut of a single column from a pre-existing file containing the paths.

The “run_list.csv” file may also be modified in some analysis to contain both the tumour and normal names, as well as both bam files with full path, and in cases where sample names need to be obscured for general data protection rights (GDPR) the new names for both the tumour and normal pair. Note, for users that sed will have to be used to change the original name to the obscured name in the header of the vcf files in this case.

2.3 Common High-Performance Computing practices

When working within a high-performance computing (HPC) environment, jobs are run using a job manager, which is most often LSF job manager (which uses sbatch) or slurm. Jobs are submitted to the job queue using Bash scripts that also includes a header that provides

information to the job manager about requirements such as number of cores, amount of memory required, how long the job should take, etc. Typically, HPCs are set up in a way that certain programs such as samtools need to be added to path, usually via a command such as: `module load samtools`. This allows system administrators to provide and manage multiple software and library versions to the user through modifying environment variables. For this analysis, the version and environment control within HPCs was managed using containers. Docker was used to build an image (standalone executable container) that contains all the code, dependencies, and runtime environment for teltool and pipeline scripts used in this analysis. This was done as allows a simple and convenient transfer of everything required and ensure reliable execution. Frequently HPCs will use singularity in place of docker as their method for loading and running containers. Singularity can still be used to run docker images by converting them to ".sif" files. These can be downloaded using `singularity pull repo.sif docker://user/repo`. Singularity requires mounting of locations for example ("`--bind path1:target1:rw, path2:target2:ro`") where `rw` and `ro` refer to the permissions to read and write files to `path1` and read only files in `path2` respectively. Assume the correct paths have been mounted following any appearance of "`singularity exec`".

Whilst most tools can accept an input directory, with larger datasets it is faster to process many samples individually in parallel. This can be achieved by procedurally generating scripts for each sample, then submitting the individual scripts to the job manager (eg. `for i in scripts/*.sh; do bsub < ${i}; done`). For readability, the shorthand "while iterating file:" will be used in place of displaying the full bash code:

```
while IFS=, read -r variable1 variable2; do
   # define s variable as sample name (file name without suffix)
   s=$(basename ${path_variable} .suffix)
   # define script variable as path in scripts dir
   script="scripts/${s}.sh"
   # write (>) header to script
   echo ${header} > ${script}
   # append (>>) to script
```

```
    echo “module load requirements” >> ${script}
    echo “code under shorthand” >> ${script}
    # make script executable
    chmod +x ${script}
done < file.csv
```

Note, other cases of the bash variable “s” (standing for sample) will also refer to the basename of a bam file. There were also cases locally where this same effect could be achieved with:

```
while IFS=, read -r cancer normal length
do
    (command) &
    while [ "$(jobs | wc -l | xargs)" -mt 8 ]; do
          sleep 10
    done
done < sample_pairs.csv
```

2.4 Teltool

Teltool is a python package that uses machine learning (ML) to predict telomere length (TL) or categorise samples into short or long TL groups (https://github.com/kearseya/teltool). It is installed using the Python package management tool pip by first installing dependencies with `pip3 install -r requirements.txt` then using `pip3 install .` whilst within the teltool directory.

Teltool uses two stages for creating a prediction: trimming, and testing. Trimming creates intermediary files that comprise of reads containing kmers that are in a predefined list. These files are placed in a directory and have the suffix “_tel.bam” along with their accompanying “.bai” indexes (bam file index). The second step “testing” uses these files to create a dataframe containing information that is passed into a pre-trained ML model to classify TL. For local cohorts (in this analysis the initial breast cancer samples with accompanying STELA data), teltool was run using:

```
teltool trim -i /path/to/bam_direcory -o /path/to/trim_direcory
teltool test -i /path/to/trim_direcory -o output_predictions
```

However, large datasets are often contained on supercomputers where this method might not be suitable. On HPC it may be more convenient to use the docker container version for reasons explained in the previous section. Samples with higher coverage (>40X) may also be down-sampled before trimming.

```
while iterating run_list.csv:
	# make empty files (${s} = sample basename variable)
	touch ${trim_direcotry}/${s}_tel1.fq
	touch ${trim_direcotry}/${s}_tel2.fq
	# subsample bamfile (which will output to stdout) and pipe this to teltool
	samtools view -h -s ${frac} --subsample-seed 123 ${path} | singularity exec
	teltool.sif teltool trim -i - -n ${s} -o ${trim_directory}
```

As a side note, it is recommended that empty fastq files (teltool intermediaries) are created using the Linux utility "`touch`", as this can mitigate certain errors with container write permissions arising from mounting interactions. As the bam file is being piped in, teltool also requires the "-n" flag to provide the sample name. Setting a subsample seed with samtools (using `--subsample-seed`) also allows for reproducibility of the analysis.

2.5 Variant Calling

Variant calling was performed using dysgu (v1.5-1.6.2), a structural variant caller developed by Kez Cleal that uses machine learning to identify and score likely variants (Cleal, K. 2022). The parameters used were min support 3 for the lower coverage (~15-20x) BGISeq breast cancer samples, and min support 5 for the higher coverage (~30-150x) Illumina HiSeq breast cancer samples from the International Cancer Genome Consortium (ICGC), and Illumina HiSeq breast cancer and Chronic Lymphocyte Leukemia (CLL) samples from Genomics England (GEL). These parameters were chosen to capture the greatest number of likely true positive structural variants whilst ignoring the most likely false positives (before later filtering stages), accommodating for the lowest coverage samples in each cohort. The output from running this analysis was a vcf file, and a bed file with binned coverage values

(for copy number analysis) for each sample. Local structural variant calling was performed using the command (with local parallelisation shown in previous section):

```
for i in bams/*.bam; do
   # run the first dysgu stage of fetching
   dysgu fetch /tmp/${s} ${i}
   # create coverage bed files from temporary files from fetching
   dysgu/coverage2bed.py -out-bin-size 10000 -w /tmp/${s} >
${coverage_dir}/${s}_cov.bed
   # run dysgu call stage
   dysgu call --ibam ${i} ${reference} /tmp/${s} -o vcfs/${s}.vcf
   # remove temporary files generated from fetch stage
   rm -rf /tmp/${s}
done
```

ICGC used the combined run function instead of splitting into stages. It also made use of several additional flags such as --low-mem, --exclude bed, and --search bed file to accommodate for the lower RAM size of the instances, speed up the process by not processing known regions to cause issues with SV calling, and only process only the main chromosome contigs (ignore ALTs). Additionally reasonable values were set for mapping quality (15), clip length (30), minimum support (5), and max coverage was set to -1 to denote no limit:

```
dysgu run -v2 --metrics --low-mem --exclude /home/ubuntu/hg19-
blacklist.v2.bed --mq 15 --max-cov -1 -x --clip-length 30 --min-support 5 -
-search /home/ubuntu/chromosome.bed ${ref} tmp_${b} ${i} -o ${b}.vcf 2>
${b}.log ; ${cov} --out-bin-size 1000 -w tmp_${b} > ${b}_cov.bed ; rm -rf
tmp_${b}
```

Within genomics England, samples were very high coverage (100-150X), to save on compute time, teltool and dysgu were run together on subsampled versions of the bam files. The version of dysgu used in this analysis (1.6.2) added an auto minimum support feature so this value did not to be set. The higher coverage also justified increasing the mapping quality (mq) variable to 30 and setting a limit on the maximum coverage value to 5000. This was achieved with:

```
while iterating run_list.csv:
      # subsample bam file to stdout
      samtools view -h -s ${frac} --subsample-seed xxx ${path} |
      # split stdout to teltool, but also pipe to…
      tee >(singularity exec teltool.sif teltool trim -i -n ${s} -o /home) |
```

```
# stdout of subsample bam to dysgu fetch
singularity exec dysgu.sif dysgu fetch --mq 30 --max-cov 5000 --clip-length
30 /tmp/tmp_${s} - 2> ${logdir}/${s}.log &&
# && denoting once command complete continue to coverage bed generation
singularity exec dysgu.sif coverage2bed.py --out-bin-size 10000 -w
/tmp/tmp_${s} > ${coverage_dir}/${s}_cov.bed &&
# finally run dysgu call
singularity exec dysgu.sif dysgu call --ibam /data/${s}.bam -v2 --metrics -
-low-mem -x ${reference} /tmp/tmp_${c} -o /out/${s}.vcf 2>>
${logdir}/${s}.log;
# delete teporary files
rm -rf /tmp/tmp_${s}
```

It was shown in the paper released alongside dysgu that higher coverage samples can lead to lower precision of SV calling, but potentially higher recall (Cleal and Baird 2022). Combining these findings with the fact it was possible to save on compute time further justified the decision to use subsampling in this case.

2.6 Variant Filtering

Several filtering strategies were investigated, including using previously published tools such as giggle and jasmine, as well as custom python scripts, before eventually the final method of using the dysgu in-built function “filter”. Filtering was performed to remove germline variants (those common between the cancer and normal samples), to generate a set of structural variants that were only relevant to the cancer.

The first attempted method was using giggle, a genomic search engine that identifies shared genomic loci between multiple genomic interval files (github.com/ryanlayer/giggle v0.6.3). However, due to unresolved compilation issues involving multiple definitions (perhaps relating to the gcc version v11.2.0), it was not possible to test this method.

Another method that was tried was using jasmine to merge the SVs to find common variants (github.com/mkirsche/Jasmine v1.1.5).

```
#!/bin/bash

# create directories for analysis
mkdir -p list_dir
mkdir -p log_dir
mkdir -p jasmine_filtered
```

```
mkdir -p merged_dir

# for each tumour vcf file
for i in /path/to/project/variants/bgi/tumour/*.vcf;
do
    # create variable of sample basename
    b=$(basename ${i} .vcf)
    echo "Processing ${b}"
    # write file containing paths to normal sample pool
    cat normals.txt > list_dir/${b}.txt
    # add tumour sample to pool
    echo ${i} >> list_dir/${b}.txt
    # run jasmine
    jasmine --mark_specific file_list=list_dir/${b}.txt
out_file=${b}_merge.vcf > log_dir/${b}.log
    # move files to new direcory
    mv output/${b}_markedSpec.vcf jasmine_filtered/
    mv ${b}_merge.vcf merged_dir
done
```

For each tumour sample, the tumour vcf file name is added to a text file along with a pool of all the normal samples. The mark specific flag was used to mark calls that were present in only one sample.

Similarly to jasmine, survivor (v1.0.7 github.com/fritzsedlazeck/SURVIVOR) is another tool that can merge multiple vcf files (Jeffares et al. 2017). SURVIVOR takes several integers as input that relate to the maximum distance between breakpoints (1kb), minimum number of supporting calls (2), take the type into account Boolean (1 = true), take the strands into account Boolean (true), placeholder (0), minimum size of SV (30bp):

```
SURVIVOUR merge ${infile} 1000 2 1 1 0 30 merged.vcf
```

A similar method was also attempted using dysgu merge with output set to csv, and using awk to filter for common variants (The pandas development team 2020). However, there were issues with conversions between the csv and vcf file types involved in files containing multiple samples (see: https://github.com/kearseya/Telomere-Genome-Complexity-Scripts/tree/main/Processing_vcfs/Filtering).

The next attempted method to attempt removing germline variants form the cancer vcfs was a custom python script. The first stage of filtering involves removing duplicate and variants with close loci present in both the normal and cancer vcf files. The second stage

involves iterating over a pool of the normal samples in the dataset and iterating over reads (aln variable in following code) from the loci (±1250bp padding) of the remaining variants, then checking for several conditions. Variants are not considered unique if reads in the pool of the normal samples met the conditions for an SV reported in a tumour sample:

For larger variants (≥ 2500bp) that are not translocations or insertions all the conditions of the read and its mate/next read are on the same chromosome, bam flag 2 (indicating read mapped in proper pair), the position of the mate/next read is within a bound, and there is overlap between the current and next/mate read:

```
all([cram_f.getrname(aln.rnext) == chr2,  # Same chrom (not translocation)
     not aln.flag & 2,  # Is discordant
     pos2 - pad[0] < aln.pnext < pos2 + pad[1],  # Pnext is the same
     reciprocal_overlap((pos1, pos2), (aln.pos, aln.pnext)) > 0, # overlap
     reciprocal_overlap((pos1, pos2), (aln.pos, aln.pnext)) /
float(abs(pos1 - pos2)) > 0.8])
```

For smaller (300-2500bp) variants, if all the conditions of the read and the position of the next/mate read is at least 90% of the reported SV length (gap), the orientation of the read is the same as the join type, and the next read is less than 3000bp away:

```
all([abs(aln.pos - aln.pnext) / gap > 0.90,
     orientation_most_common == pair_orientation(aln),
     abs(aln.pos - aln.pnext) < 3000])
```

For <300bp variants, the cigar string is analysed for if any reads in the area exhibit insertions (I=1), deletions (D=2), or soft clips (S=4) longer than 15 base pairs:

```
cigartuples = [aln.cigartuples for aln in cram_f.fetch(chr1, 0 if pos1 -
1000 < 0 else pos1 - 1000, pos1 + 1000) if aln.cigartuples != None]
for ctt in cigartuples:
    seen = any( (0 < ct[0] <= 2 and ct[1] >= 15) or (ct[0] == 4 and ct[1]
>= 15) for ct in ctt)
```

For translocations, if there is an alignment mate that appears in the reported translocation second chromosome at the loci (allowing for a padded area around the breakpoint):

```
cram_f.getrname(aln.rnext) == chr2 and pos2 - pad[0] < aln.pnext < pos2 +
pad[1]
```

For insertions Striped Smith Waterman alignment (using v0.6 scikit-bio) is used (local pairwise-alignment) to test if the insertion sequence from the vcf is present in reads from the normal samples are present at the same loci, and the optimal alignment score is above a threshold (Rideout et al. 2024).

2.7 Variant Validation by manual curation

Dysgu vcf files contain an extra field in the FORMAT column called “PROB”. This is a probability score the dysgu model assigns each variant that indicates how confident its’ model is that the called variant is real and not a sequencing artifact or general noise. For each dataset, a subset of ~N samples were inspected with the gw genome viewer (github.com/kcleal/gw v0.8.2) at ~X loci for each different variant type called (Cleal et al. 2024). Using the filtering function of bcftools (github.com/samtools/bcftools v.1.17) and piping results to gw, the thresholds of the “PROB” and supporting reads (“SU”) were tweaked until the value of each were found to maximize the number of true variants kept and false positives removed (Danecek et al. 2021).

```
#!/bin/bash

# define file paths
vcfdir="/path/to/vcfs"
bamdir="/path/to/bams"
refpath="/path/to/reference.fa"

# convert command line variable to path
i=$(ls ${vcfdir}/$1*)
b=$(basename $i .vcf)

# filter vcf to variants between upper and lower PROB bound
bcftools filter -i'INFO/SVTYPE == "DEL" && FORMAT/PROB >= 0.1 &&
FORMAT/PROB <= 0.2' ${vcfdir}/${b}.vcf |
# gw opens the bam file at the regions of piped in vcf
gw --link sv --labels yes,no,maybe --out-labels ${i}_del.tsv ${refpath} -b
${bamdir}/${b}.bam -v - --track ${i}
```

2.8 International Cancer Genome Consortium (ICGC) cloud analysis

Data contained within the ICGC is accessed either via Amazon Web Services (AWS) or the Collaboratory (now sunsetted). AWS was initially attempted as a method for analysing the cohort, however the "score client" software (the means of accessing the data) was not working reliably despite the EC2 instances being based in Virginia per instructions. The AWS route of accessing the data was subsequently abandoned in favour of the Collabatory hosted by Google.

This section will focus on the Collaboratory route which is similar to AWS but with extra steps. Once enrolled into the program, the first step was setting up a network and an initial instance. By default, the network was not accessible outside of itself, so the Classless Inter-Domain Routing (CIDR) of the initial instance had to be opened by setting it to "0.0.0.0/0". The last step before accessing the instance was to set up ssh keys, these are encrypted strings of text that afford a higher level of security than regular passwords. Keys are downloaded and are usually stored in "~/.ssh". They are then used to access the remote host by providing the path in the ssh command, or more commonly added to the host alias in "~/.ssh/config". Additionally, the line "IdentitiesOnly yes" is also added to this config file (could also be provided in ssh command). Whilst the score client has an option for mounting the "bucket" (data repository) containing the bam files, throughput when accessing the files is very slow at around 1Mb/s, obviously less than ideal when working with 100+GB files. Similarly to how AWS instances can be specced with (given) extra drives, a 200GB NVMe drive was mounted to the Collabatory instance at the path /mnt/nvme0. A setup script was developed to install all the dependencies and tools required for the analysis (score-client, dysgu, teltool, htslib, etc), configure the score client, set up environment variables, and mount the drive (full script available at: https://github.com/kearseya/Telomere-Genome-Complexity-

Scripts/blob/main/ICGC/Setup/setup.sh). The script takes care of downloading the data, analyses, and sending the output data outside of the cloud compute instance.

```
# create an array or bam file object ids
readarray -t s < /home/ubuntu/nomini_object_ids

echo -e "--------------------\n\tLIST (${#s[@]})\n\n${s[*]}\n"
echo -e "--------------------\n\n\tSTART\n\n--------------------"
# for each object id (s)
for s in ${s[*]}
   do
   # move to the attached nvme storage
   cd /mnt/nvme0n1/
   # download bam file
   ${scli} download --object-id ${s} --output-dir /mnt/nvme0n1
   # assign bam file path and basename to variables
   i="/mnt/nvme0n1/*.bam"
   b=$(basename ${i} .bam)
   echo "Running dysgu ${b}"
   # run dysgu (command shown earlier in variant calling section)
   /home/ubuntu/.local/bin/dysgu run -v2 --metrics --low-mem --exclude
/home/ubuntu/hg19-blacklist.v2.bed --mq 15 --max-cov -1 -x --clip-length 30
--min-support 5 --search /home/ubuntu/chromosome.bed ${ref} tmp_${b} ${i} -
o ${b}.vcf 2> ${b}.log ; ${cov} --out-bin-size 1000 -w tmp_${b} >
${b}_cov.bed ; rm -rf tmp_${b}

   # move output to new directory
   mv ${b}.vcf /mnt/nvme0n1/dysgu_out/
   mv ${b}.log /mnt/nvme0n1/dysgu_out/
   mv ${b}_cov.bed /mnt/nvme0n1/dysgu_out/
   # transfer contents of output directory to new computer
   sshpass -f "/home/ubuntu/pw.txt" scp -r /mnt/nvme0n1/dysgu_out/*
User@Host:/scratch/User/icgc/dysgu
   # if teltool output does not exist
   if grep -Fxq ${b} noteltool
   then
      echo "Trimming ${b}"
      # run teltool trim function
      teltool trim -i ${i} -o teltool_out
      # move outputs
      mv coverages.csv teltool_out/${b}_cov.csv
      mv *_tel.bam teltool_out/${b}_tel.bam
      mv *_tel.bam.bai teltool_out/${b}_tel.bam.bai
      # transfer outputs to new computer
      sshpass -f "/home/ubuntu/pw.txt" scp -r /mnt/nvme0n1/teltool_out/*
User@Host:/scratch/User/icgc/teltool
   else
      echo "Already done teltool ${b}"
   fi

   # delete bam file (and index) to regain storage space
   rm *.bam*
done
```

Once it was confirmed this script could be used on a fresh instance, seven other instances were created. From there, the analyses were started on the whole cohort by setting up aliases for each instance, and then performing the following actions:

```
1. scp requirements/* ec2:/home/ubuntu/
2. ssh ec2
3. chmod +x setup.sh
4. ssh whereToSendData
5. edit nomini_object_ids
6. tmux
7. ./setup.sh
(8. hit y enter ~4x) # yes prompts for installations
(9. ctrl+b - d) # detatch from tmux session
```

The requirements directory was copied via scp into the instances in step 1. This directory contained an application.properties file (contains access token required for score client), a chromosome bed file, a blacklist file (used in the dysgu command), nomini_object_ids which contains the object IDs for the score client to download the bam files, a list of IDs where teltool had not been run during the testing of the script, and pw.txt which contained the password for the destination the output was sent to. Logging into the instance then allows the executable permissions to be set on this script. Additionally logging into location data is being sent to adds the public key to the "~/.ssh/authorized_keys" file which avoids issues later down the pipeline. The list of object ids (related to the bam files) could then be edited so each instance could work on a unique subset of the cohort at the same time. Finally, tmux (github.com/tmux/tmux v.3.2) was used as it is able to keep the processes running whilst not logged into the instances, as it runs and stores its state in a background tmux server.

As there was not an established vcf filtering method before running this first batch of analysis, a separate script was used to obtain a crop of the bam files to the regions dysgu needs for filtering. This ensured that the relevant sections of the bam files could be analysed more efficiently at a later date.

```
# create array of object ids
readarray -t s < /home/ubuntu/normal_ids.txt
# set variable for path of score-client executable
sc=/home/ubuntu/score-client/bin/score-client
# for object id in array (i)
for i in ${s[*]}
do
   echo "attempting download"
   # download bam file, if the download failed, print failure and skip
   ${sc} download --object-id ${i} --output-dir /home/ubuntu || echo "${i}
failed"; continue
   echo "download successful"
```

```
    # create variables for bam file and basename
    bam_file=*.bam
    b=$(basename ${bam_file} .bam)
    # create bam file containing only regions in tumours.vde
    samtools view -hb --region-file tumours.bed ${bam_file} -o
trimmed_normal/${b}.bam
    # index newly created bam file
    samtools index trimmed_normal/${b}.bam
    # remove bam file and index to create space for next file
    rm *.bam*
done
```

The “tumours.bed” file was generated by first merging all the tumour vcf outputs (with `dysgu merge`) to a merged vcf. A script was then used to read each SV in the merged file and apply a padding of 1kb to their loci to generate a region in bed file format with `python3 vcf_to_bed.py tumour_merged.vcf -out tumours.bed` (script found at github.com/kearseya/Telomere-Genome-Complexity-Scripts/blob/main/Processing_vcfs/Filtering/vcf_to_bed.py).

Clinical data and other auxiliary data was downloaded from the ICGC data portal which was retired on the 15th of June 2024. As there was no easy “download all” button, a web scraper was written using the selenium webdriver. By iterating the fids, URLs could be generated to point to the webpage for each sample, then the download button can be located and clicked using the find_element_by_css_selector command. Unfortunately, some of this data was in pdf format, perhaps the most unusable file type for extracting data for individual samples when contained in individual files for each sample. In this case data was attempted to be extracted by cropping the pdf with ghostscript, then extracting the information with optical character recognition (OCR) with tesseract, and aggregating with R.

2.9 Pipeline scripts

Please note that when using the container for these scripts, setting the bash variable “OPENPLAS_NUM_THREADS=1” is requited to prevent an ambiguous python “Memory error” (comment # 1 below) possibly caused from compiling of the container on a CPU with

more threads than the final run destination. The following commands for the containerised version are wrapped in:

```
singularity exec pipelines.sif bash -c “export OPENBLAS_NUM_THREADS=1; # 1
cd /app/pipeline_name;
command;                                                                  # 2
mv out out                                                                # 3
```

As mentioned in the common HPC practices section, the mounting binds have been omitted from the singularity command (comment #1) for clarity. It is also worth noting and not cases is the final move command required to get the data to the correct path (comment #3 above). The pipeline scripts below that take a vcf input contain an input parameter for the probability threshold for including variants in the subsequent analysis. This is a string which takes the form of a python dictionary, with each variant type (DEL, DUP, INV etc) being the key, and value being the desired threshold For the BGI dataset, the default “PASS” filter values for each variant type were used (“INS: 0.45, DEL: 0.45, INV: 0.45, DUP: 0.45, TRA: 0.45”), and for the higher coverage ICGC and GEL datasets the values were increased to “INS: 0.85, DEL: 0.85, INV: 0.45, DUP: 0.25, TRA: 0.35”. The full code for each described analysis can be found at: github.com/kearseya/dysgu-pipelines under their respective directories.

2.10 Copy Number Analysis

The copy number analysis utilised binning of reads within non-overlapping windows, with a size for this analysis was chosen to be 10kb as used in previous analyses by Cleal et al (Cleal et al. 2019). Before the analysis, bed files for the GC percentage and mappability were generated with:

```
# open file buffer to write to
with open(f"{ref}_prep.bed", "w") as f:
   # for each chromosome
   for c in chrom_lengths_dict:
       # for each window size, add window to bed file
        for i in np.arange(0, chrom_lengths_dict[c], winsize)[:-1]:
            f.write(f"{c}\t{i}\t{i+winsize}\n")
       # write last window (last window interval and chromosome length)
```

```
        f.write(f"{c}\t{np.arange(0, chrom_lengths_dict[c], winsize)[-
1]}\t{chrom_lengths_dict[c]}\n")

# count gc percentage of each window from reference and write to file
bedtools nuc -fi ${ref}.fa --bed ${ref}_prep.bed | cut -f1,2,3,5 >
${ref}.${newwin}${winunit}_windows.gc_pct.bed

# convert bigwig to bed file
bigWigToBedGraph ${ref}.k100.Umap.MultiTrackMappability.bw ${ref}_map.bed
# this step reads the bed to generate median mappability for each window
python3 copy_number_pipeline.py preprocess -r ${ref} -w ${winsize} -m
${ref}_map.bed
```

Tools used for this step are bedtools (v2.30.0 github.com/arq5x/bedtools2) and bigWigToBedGraph (v398 available as part of the UCSC Genome Browser's utilities: https://hgdownload.soe.ucsc.edu/downloads.html#utilities_downloads) (Quinlan and Hall 2010). The mappability files were obtained through the UCSC genome track browser (full files can be found at hg19: http://genome.ucsc.edu/cgi-bin/hgTables?db=hg19&hgta_group=hub_67117&hgta_track=hub_67117_Umap_100_quantitative&hgta_table=hub_67117_Umap.100.quantitative&hgta_doSchema=describe+table+schema and hg38: https://genome.ucsc.edu/cgi-bin/hgTables?db=hg38&hgta_group=map&hgta_track=umap&hgta_table=umap100Quantitative&hgta_doSchema=describe+table+schema). These were generated by Hoffman et al by generating all possible kmers of differing lengths (24, 36, 50, and 100 bp) and mapping them to the references using bowtie (Langmead et al. 2009). They then merged data from unique mappings to create a score for how “mappable” each region of the reference genome is (i.e. how unique each region is). This is useful as regions with lower scores are more susceptibile to incorrect mapping from reads from other loci with sequencing errors or unexpected genetic variation (Karimzadeh et al. 2018).

2.10.1 Normalisation

2.10.1.1 GC and mappability normalisation

Raw coverage was normalised to the GC percentage and average mappability score based on the method used by Cleal *et al* (Cleal et al. 2019). This is achieved by creating a matrix of coverage values for all combinations of GC% between 30 and 60, and mappability between 60 and 101, and taking the median of the matrix as the genomic median" (examples in appendix 1).

```
# for coverage, gc%, and mappability for each window
for c, g, m in zip(list(s["coverage"]), list(df["GC"]), list(df["map"])):
    # if coverage is more than 500 ignore
    if c > 500:
        continue
    # if gc% is between 30 and 60 non inclusive
    if 30 < g < 60:
        # if mappability if between 60 and 101 non inclusive
        if 60 < m < 101:
            # add coverage value to array
            norm_array[(g, m)].append(c)
# for s in raw coverage directory, load coverage bed as datafame
# limit dataframe to contain windows that fit above criteria
s = s[(s["GC"].between(30, 60)) & (s["map"].between(70, 101))]
# calculate median coverage of new dataframe
median = np.median(s["coverage"])
```

2.10.1.2 Interpolation

The genomic median is subtracted from all the coverage values within the matrix using NumPy (np v1.26.4) data types, which is then interpolated with the scipy (v1.13.1) function RFBInterpolator (with a smooth value of 5) (Harris et al. 2020; Virtanen et al. 2020).

```
# convert array from above code into dictionary
norm_array = {k: np.median(np.array(v)) for k, v in norm_array.items()}

# initialise arrays
x, y, z = [], [], []
arr = np.zeros((101, 101))
# iterate norm_array (dictionary)
for (i, j), v in norm_array.items():
    if v < 1000:
        # subtract median value from original coverage value
        sub = v - median
        # add values to arrays
        x.append(j), y.append(i), z.append(sub)
```

```
# calculate eplison value
xi = np.stack([x, y])
ximax = np.amax(xi, axis=1)
ximin = np.amin(xi, axis=1)
edges = ximax - ximin
edges = edges[np.nonzero(edges)]
epsilon = np.power(np.prod(edges)/len(x), 1.0/len(edges))

# create interpolator
rbf = RBFInterpolator(np.column_stack([x, y]), np.array(z), smoothing=5,
kernel="multiquadric", epsilon=epsilon)
# evaluate interpolant
z = rbf(np.column_stack([x, y]))
```

RFBInterpolator from scipy is a python implementation of radial basis function interpolation, a method used in approximation theory (approximation of functions with simpler functions) where the output is a weighted sum.

$$f(x) = K(x,y)a + P(x)b \qquad \text{s.t}$$

$$(K(y,y) + \lambda I)a + P(y)b = d$$

$$P(y)^T a = 0$$

Where *d* is an input vector, *y* the locations, *K(x,y)* is a matrix of radial basis functions (centres at *y* for points *x*), *P(x)* describes a matrix of monomials spanning polynomials, and λ a non-negative smoothing parameter (5). The default type of RBF $r^2 \times log(r)$ was used (where $r = ||x - c||$ and *x* is a point in a scalar valued function in N-dimensional space, and *c* the centre of the RBF) (Wahba 1990; Fasshauer 2007).

With the interpolated coverage values, the coverage across the sample can then be normalised to these new values.

```
# convert loose arrays into dictionary
norm_value = {(int(xx), int(yy)): zz for xx, yy, zz in zip(x, y, z)}
normed_counts = []
# for gc%, mappability, and raw coverage in oringal array
for g, m, v in zip(list(s["GC"]), list(s["map"]), list(s["coverage"])):
       # if within bounds for mappability and gc%
      if (m, g) in norm_value:
             # add normal coverage - normalised value to array
             normed_counts.append(v - norm_value[(m, g)])
      # else add 0 to array
      else:
             normed_counts.append(0)
```

2.10.1.3 Denoising

Penultimately, a signal denoising step is applied to the data in the form of a Haar wavelet transformation from the PyWavelet library (pywt v1.6.0) (Haar 1910; Lee et al. 2019). The principal behind wavelet transformation use in denoising is they create a sparse representation for signals (and images), concentrating the signal features into a smaller number of large-magnitude wavelet coefficients (i.e. removing low amplitude noise). This is performed twice, the first acting as a stop gap for denoising (somewhat analogous to "smoothing"). Initially a "level" value of 2 is used, this lowers the value of the frequency divider (as $f_{out} = f_{in} / N$), to generate a noisy set of coefficients. From this set, the mean absolute deviation and "sqtwolog" were calculated and multiplied together to create a "universal threshold" (uthresh). This threshold is then used in soft thresholding of the data to denoise it.

```
# convert previously python array to numpy array
d = np.array(normed_counts)

# apply first round of haar wavelet transform denoising
noisy_coefs = pywt.wavedec(d, 'haar', level=2, mode='per')
# calculate the mean absolute deviate (statsmodels.robust function)
sigma = mad(noisy_coefs[-1])
# calculate universal threahold
uthresh = sigma * np.sqrt(2 * np.log(len(d)))

denoised = noisy_coefs[:]
# apply soft thresholding using previously calculated value
denoised[1:] = (pywt.threshold(i, value=uthresh) for i in denoised[1:])
# apply second round of denoising
sig = pywt.waverec(denoised, 'haar', mode='per')

if len(sig) > len(s["GC"]):
    sig = sig[:-1]

# write results to dataframe
s["normed"] = [sig[i] if i not in bad_indexes else np.NAN for i in
range(len(sig))]
```

The Haar wavelet function is defined as:

$$\varphi(x) = \begin{cases} 1 & \text{if } 0 \leq x < 0.5 \\ -1 & \text{if } 0.5 \leq x < 1 \\ 0 & else \end{cases}$$

Which represents a simple oscillation between positive and negative values which captures changes in a signal. The Haar wavelet transformation can then be described as:

$$W_{j,k} = \int_{-\infty}^{\infty} f(x)\varphi_{j,k}(x)dx$$

Where $\varphi_{j,k}(x)$ is the Haar wavelet function scaled by 2^j and shifted by k:

$$\varphi_{j,k} = 2^{\frac{j}{2}}\varphi(2^j x - k)$$

$W_{j,k}$ represents the wavelet coefficient (weight) at scale j, at position k of the input signal $f(x)$ being transformed (integrating for all values of x).

Soft thresholding is the application of a function to the wavelet coefficients $W_{j,k}$ to shrink them towards zero if they are below a threshold which reduces the effect of noise (hence denoising). The equation for soft thresholding the coefficients:

$$\eta_\lambda(W) = \text{sign}(W)(|W| - \lambda)_+.$$

Where $\eta_\lambda(W)$ denotes the soft threshold value of the wavelet coefficient W, sign(W) a function returning the sign of W coefficients, $|W|$ the modulus of the W, λ a threshold parameter, and +• showing that if the results is negative then set the value to 0.

In this instance, the "universal threshold" was used, calculated as the median absolute deviation multiplied by the "sqrwolog":

$$\lambda = \text{median}(|X_i - \bar{X}|) \times \sqrt{2 \times \log(\text{length}(f(x)))}$$

2.10.1.4 Background removal

Finally, the values produced from all the normalisation processes for the normal samples are subtracted from the cancer samples to remove the background. The end result is a dataframe containing the copy number alternations generated during carcinogenesis.

2.10.2 Piecewise Constant Fitting (PCF)

These background removed dataframes are used as an input for an R script that uses the copynumber library (v1.34.0) (Nilson, G 2018). A couple of the functions from the copynumber library were modified to allow inputting of cytoband files that are not within the package. This R script winsorizes the input dataframe to reduce the affect off the outliers for the next step (ref). Winsorizing is similar to trimming or clipping values, but instead of removing them completely, they are replaced by percentiles $y_j = \psi(y_j)$ where:

$$\psi(y) = \psi(y|\theta) = \begin{cases} -\theta, & y < -\theta \\ \theta, & y > \theta \\ y, & \text{else} \end{cases}$$

The copynumber package defines theta as: $\theta = \tau s$ where τ in this analysis was 2.5 (variable input to function), and s is median absolute deviation. Piecewise constant fitting (PCF) is then used to generate a file containing the chromosome, start, end, and mean ($\bar{I}_m$) for each segment (S_m). Such that: $S \subseteq I$ where $I \subseteq y$, for example for the nth segment $S_n = I_n = \{y_{j \text{ of nth breakpoint}}, \ldots, y_{j \text{ of n+1 breakpoint}}\}$ (so start and end are the j×windowSize (10kb) of n and n+1 breakpoint respectively). Due to the size of the genomes being analysed, the copynumber package defaults to the "fast" implementation of PCF. The fast PCF function applies a heuristic approach for identifying the number of break points, plus a few techniques for reducing the order of the algorithm (quadratic). The likelihood of observing the set of break points S given the set of observed copy number values y using a penalty value of gamma (γ) is:

$$L(S|y,\gamma) = \sum_{I \in S} \sum_{j \in I} (y_j - \bar{y}_I)^2 + \gamma \times \text{length}(S)$$

And its derivative:

$$L'(S|y,\gamma) = -\sum_{I \in S}\left(\sum_{j \in I} y_j\right)^2 \div n_I + \gamma \times \text{length}(S)$$

By using a lower value of gamma, the segmentation is more relaxed. The most likely number of segments is calculated by minimising the least squares criterion of this derivative function. This is achieved through dynamic programming (condensing to simpler subproblems) to reduce the order of operations to $O(q^2)$ where q is the number of potential breakpoints. Due to the optimal segmentations of both sides of a breakpoint being mutually independent, it is possible to iterate values of *k* to find the cost term for the last segment:

$$d_j^k = \frac{1}{j-k-1}\left(\sum_{r=j}^{k} y_r\right)^2$$

Which can then be used to find the total error for the optimal solution of this last segment by iterating the possible start positions (*j*) for this segment:

$$e_k = \min_{j \in \{1,\ldots,k\}}(d_j^k + e_{j-1} + \gamma) \qquad (\text{where } e_0 = 0)$$

By iterating over the possible breakpoints $r_0(=1), \ldots, r_q(=p+1)$ and aggregating with $u_k = \sum_{j=r_{k-1}}^{r_k-1} y_j$ where *k* is kmin (5 in this analysis) to q, it possible to find the optimal segments going backwards ([… 0] indicates initialised as an empty vector):

$$\prod_{k=kmin}^{q} a_k = [a_{k-1}\ 0] + u_k, e_k = [e_{k-1}\ 0] + r_k - r_{k-1}, d_k = a_k \times \frac{a_k}{e_k}, e_k = \left[e_{k-1}\ \min(d_j^k + e_{j-1} + \gamma)\right]$$

When the minimum in the last step is stored, the index t_k is stored so the start indices of the segments can be found. The kmin value determines the minimum number of points that can be used in a segment, so by using a value of 5 and multiplying by the window size, the smallest segments in this analysis are 50kb. Individual chromosomes, as well as whole genomes are then plotted with the winsorized coverage values overlaid with the segments (genome relative coverage and heatmap plots).

2.10.2.1 Complexity score

A ”complexity score” can be constructed with the absolute sum of PCF segments relative copy numbers. Using the sum promotes samples with a larger number of segments to have higher scores than stable samples maintaining a consistent copy number.

2.10.3 Low pass filtering

A low-pass filter is a one that only passes signals below a specified cutoff frequency while diminishing all signals above it. It was also used separately to pcf as a method for analysing the relative copy numbers of regions. This method is less computationally expensive and more sensitive to shorter regions with higher variance, with the trade-off being more affected by noise. The “complexity score” was calculated as the sum of distances between concurrent peaks and troughs that were more than 0.5 in value. This threshold is arbitrary but was set this high as to combat the effect of noise in the end score.

2.11 Circos Plotting

Circos plots were generated using the pycircos library (v0.3.0 github.com/ponnhide/pyCircos). The length values from the vcf header are extracted using regex to create a dictionary of the lengths which can then be used to create the “Garc” objects which are added to a “Gcircle” object.

```
# define Gcircle object with a figure size of 5, 5
circle = pycircos.Gcircle(figsize=(5, 5))
nl = {}
# open vcf file and iterate over each line
with open(fp) as f:
    f.readline()
    for line in f:
        if line.startswith("##contig"):
            # extract the string after ID (chromosome name) and remove
            # chr prefix if one is present
            name    = re.search("ID=([a-zA-Z0-9_]*)",
line).group(1).replace("chr", "")
            # extract the length of the chromosome
```

```
            length  = int(re.search("length=(\d+)", line).group(1))
            # if an underscore is not present in the name indicating
            # not an alt contig, set the chromosome length in dictionary
            if "_" not in name:
                nl[name] = length
# create an array with the desired order to plot chromosomes
order_of_chr = [f"{i}" for i in range (1, 23)] + ["X", "Y"]
# for eaach chromosome
for i in order_of_chr:
      name    = i
      length  = nl[i]
      # add the chromosome to the Gcircle object
      arc     = pycircos.Garc(arc_id=name, size=length, interspace=2,
raxis_range=(low_size, up_size), labelposition=100, label_visible=True,
facecolor="white", labelsize=17)
      circle.add_garc(arc)
      circle.set_garcs()
```

A cytoband file is then parsed (if provided) to add banding information to the outer edges of the plot. This is done with the circle.barplot() function. Optionally the output from the previous copy number analysis can also be added as a ring and is plotted with circle.lineplot(), with highlighting of chromosomes (barplot) that have 3 tiers of "complexity score": yellow 10-30, orange 30-50, red >50, and light orange for those with 9 or more segments. Pysam (v0.22.1 github.com/pysam-developers/pysam) is used to parse the vcf files to provide the arc plotting information.

```
# initialise reading of vcf file
f = pysam.VariantFile(fp)
tra_list = []
ins_dict = collections.defaultdict(dict)
# initialise arrays for plotting insertions
for i in arcdata_dict.keys():
    ins_dict[i]["positions"] = []
    ins_dict[i]["widths"] = []
# for line in vcf
for l in f.fetch():
    # set chromosome name variable with "chr" prefix removed
    name = l.chrom.replace("chr", "")
    # get sv type (for colouring)
    sv_type = l.info["SVTYPE"]
    # get name of sample to access the filtering variables
    sample_name = list(l.samples.keys())[0]
    # if PROB and SU value from dysgu are below threshold skip
    if l.samples[sample_name]["PROB"] < prob_thresholds[sv_type]:
        if l.samples[sample_name]["SU"] < su_thresholds[sv_type]:
            continue
    # get the start and end position of SV
    start = l.pos
    end = l.stop #info["END"]
    # if the variant is below size threshold skip
    if sv_type in {"DEL", "INV"}: #, "INS"}:
        if abs(end-start) <= size_threshold:
```

```
            continue
        # remove chr prefix for chr1 and chr2 of SV
        chr1 = l.contig.replace("chr", "")
        chr2 = l.info["CHR2"].replace("chr", "")
        # if not a translocation and within main chromosomes
        if chr1 == chr2 and name in order_of_chr:
            source = (name, start, start, low_size-aof)
            destination = (name, end, end, low_size-aof)
            if abs(end-start) < 50000: # required for line to show
                destination = (name, end+50000, end+50000, low_size-aof)
            # if not insertion plot the arc of the SV
            if sv_type != "INS":
                circle.chord_plot(source, destination,
facecolor=type_cols[sv_type], edgecolor=type_cols[sv_type], linewidth=0.4)
            else:
                ins_dict[str(name)]["positions"].append(start)
            # if translocation append details to list
            elif chr1 != chr2 and name in order_of_chr and chr2 in
order_of_chr:
                name2 = l.info["CHR2"].replace("chr", "")
                start2 = l.info["CHR2_POS"]
                end2 = start2
            tra_list.append(((name, start, start, low_size-aof), (name2,
start2, end2, low_size-aof)))
        else:
            continue
    # plot translocations
    for t in tra_list:
        circle.chord_plot(t[0], t[1], edgecolor=type_cols[“TRA”]",
linewidth=0.4)
    # plot insertions
    for key in ins_dict:
        if len(ins_dict[key]["positions"]) == 0:
            continue
        circle.scatterplot(key, data=[1]*len(ins_dict[key]["positions"]),
                positions=ins_dict[key]["positions"],
                raxis_range=(low_size-aof, low_size-dof), facecolor="green",
                edgecolor="green",  markershape=".", markersize=20)
```

To overcome rendering issues of some arcs not being displayed when converting the resulting svg to png, it is first rendered at twice the size, before being scaled back down to the original size.

2.12 Chain link finding

This method is based on the ChainFinder algorithm developed by Baca et al with some adaptations (Baca et al. 2013; Cleal et al. 2019). The first stage is to calculate the frequency of breakpoints across the cohort:

```
    # create dictionary entry for each chromosome
    ps = {str(k): 0.0 for k in cl.keys()}
```

```
for key, value in breaks_list.items():
    break_n_per_chrom = defaultdict(int)
    # for each chromosome increment the breakpoint counter
    for chrom, b in value.items():
        break_n_per_chrom[str(chrom)] += len(b)
    # cacluate average "expected" distance of breakpoints across each
      chromosome
    avg = {str(k): v / cl[str(k)] for k, v in break_n_per_chrom.items()}
    for k, v in break_n_per_chrom.items():
        ps[k] += v
ps = {k: (v/len(breaks_list) / cl[k]) for k, v in ps.items()}
```

The frequency of observed breaksites across the cohort is used as an "expected" distribution as generating an expected distribution could unintentionally introduce artifacts from assumptions made. Next a nearest neighbour tree is made using scikit-learn (sklearn v1.5.0) (Pedregosa et al. 2011; Buitinck et al. 2013):

```
# initialise dictionary containing a dictionary
nn = defaultdict(lambda: defaultdict(list))  # {sample: {chromosome:
[neighbour trees]}}
for samp, value in breaks_list.items():
    for chrom, bs in value.items():
        X = np.array(bs)[:, np.newaxis]
        # create tree using sklearn.neighbors.KDETree
        tree = KDTree(X)
        nn[samp][chrom] = tree
```

The p-value is calculated with: `pval = 1-stats.binom.pmf(0, fails, prob_success)` such that 1 or more breaksites are observed within this given distance (using scipy.stats.binom function) (Virtanen et al. 2020). The distance in bp is the number of fails i.e. the number of bp without a break, success is 0, and the probability that no breaks are observed. The list of breakpoints for each sample is then iterated over such that for each breakpoint look at up- and downstream of SV sites, and if the p-value is less than threshold (i.e. closer than "expected" for the cohort distribution), then make an edge on the graph.

```
def find_connected(tree, data, chrom, pos, p_val):
    """Return a set of edges from tree that meet P-value threshold."""
    # get distance and indicies of tree
    dist, ind = tree.query([[pos]], k=len(data))
    # calculate p-values for each breakpoint to array
    pvals = [get_pval(i, ps[chrom]) for i in dist[0]]
    dist, ind = dist.flatten(), ind.flatten()
    c = set([])
    # iterate p-value array and save edges to set
    for i, p in enumerate(pvals[1:]):
        if p < p_val:
```

```
                c.add(("{}:{}".format(chrom, int(pos)), "{}:{}".format(chrom,
        int(data[ind[i + 1]]))))
            return c
```

The output of the above function is used to create a set of connected structural variants i.e. chains of SVs:

```
        def neigh(s, samp, nn, prob):
            """Return a set of edges from tree that meet P-value threshold for both
               breakpoints."""
            # create tree objects and reformat
            tree1 = nn[samp][s[0]]  # Tree for chromosome 1
            tree2 = nn[samp][s[3]]  # Tree for chromosome 2, (intra or inter)
            data1 = np.array(tree1.data).flatten()
            data2 = np.array(tree2.data).flatten()
            # call find connected (shown above)
            c1 = find_connected(tree1, data1, s[0], s[1], prob)
            c2 = find_connected(tree2, data2, s[3], s[5], prob)
            # |= ior results to set
            connected = set([])
            connected |= c1
            connected |= c2
            return connected
```

The connected and all breakpoint edges are added to a networkx (v3.3) Graph (n) with colours (clr) to act as an id to distinguish them (Hagberg et al. 2008). Identified clusters of SVs are then written to a file:

```
        def get_clusters(n, clr, count, samp):
            """Return a set of clustered edges from tree."""
            # extract connected edges using the colour id
            cluster_edges = [i for i, j in zip(n, clr) if j == "lightgrey"]

            # Make a new graph to isolate the clusters
            G = nx.Graph()
            G.add_edges_from(cluster_edges)

            # create new file
            with open("{}/{}/{}_{}_clusters.csv".format(OUTDIR, samp, samp, count),
        "w") as out:
                # write header to file
                out.write("Chrom\tBreak_number\tSize(bp)\tMedian_spacing(bp)\tBreak
        _pos\n")
                # iterate subgraphs
                for k in [G.subgraph(c).copy() for c in
        nx.connected_components(G)]:
                    # extract sorted clusters
                    cluster = sorted([(i.split(":")[0], int(i.split(":")[1])) for i
in k.nodes()], key=lambda x: x[1])

                    # calculate cluster size
                    c_size = cluster[-1][1] - cluster[0][1]
                    # calculate success, failure, probability success
                    success = len(cluster)
                    faliure = c_size - success
                    prob_success = ps[cluster[0][0]]
```

```
            # calculate a negative binomial discrete random variable
            neg_b = stats.nbinom.pmf(k=faliure, n=success, p=prob_success)

            clust_size = cluster[-1][1] - cluster[0][1]
            clust_len = len(cluster)
            # calculate spacing and median spacing
            spacing = np.array([cluster[i+1][1] - cluster[i][1] for i in
range(len(cluster)-1)])
            median_spacing = np.median(spacing)
            pos = ",".join([str(i[1]) for i in cluster])

            chrom = cluster[0][0]

            # write information to file
            out.write("{c}\t{n}\t{s}\t{m}\t{p}\t{nb}\n".format(c=chrom,
n=clust_len, s=clust_size, m=median_spacing, p=pos, nb=neg_b))
```

2.13 Contig assembly

Contigs were assembled from reads collected recursive from breakpoint sites deemed to be close together from the chain linking analysis, which are then aligned to a reference genome. This method is an adaptation of the method used by Cleal et al (Cleal et al. 2019). Loci from the output of the chain linking analysis was used to scan the raw alignment files for discordant reads.

```
# iterate chain linked file output of clusters
for sample, df in data.groupby("Sample")
    # find sample bam path and open using pysam
    sample_bam = bam_paths[sample]
    bam = pysam.AlignmentFile(sample_bam, "rb")
    # create new bam files for single and paired reads
    out_singles =
pysam.AlignmentFile("./{o}/{s}/{s}_singles.bam".format(o=OUT, s=sample),
"wb", template=bam)
    out_pairs =
pysam.AlignmentFile("./{o}/{s}/{s}_paired.bam".format(o=OUT, s=sample),
"wb", template=bam)
    # create array for discordant reads and iterate over the chained df
    discordants = []
    for idx, r in df.iterrows():
        # recursively find reads over first and second breakpoint
        d1 = recursive_find.recurive_find(bam, r.chrom1, r.chr1_start,
r.chr1_end, pad, max_depth=3)
        d2 = recursive_find.recurive_find(bam, r.chrom2, r.chr2_start,
r.chr2_end, pad, max_depth=3)
        # append found reads with duplicates filitered out to array
        discordants += recursive_find.filter_duplicates(d1+d2)
    # filter duplicate reads from discordant array
    unique = recursive_find.filter_duplicates(discordants)
    # split reads into singles and pairs and write to respecitve files
    singles, pairs = recursive_find.sort_disc(unique)
```

```
    with out_singles, out_pairs:
        for read in unique:
            out_singles.write(read)
        for read in pairs:
            out_pairs.write(read)
```

The output bam files were then processed samtools sort, so they could then be converted to fastq using “bedtools bamtofastq” (v2.30.0). These fastq files were then downsampled using the normalise-by-median python script (from github.com/dib-lab/khmer) which removes surplus reads with a median kmer abundance more than coverage 20 for kmers size 21 (Crusoe et al. 2015). Connected reads were then passed to the SPADEs aligner (v3.15.5) (Prjibelski et al. 2020).

```
# create array for spades aligner kmer sizes and set it backwards ([::-1])
kmers = [21, 33, 55, 77, 111][::-1]
# for each kmer size
for k in [",".join(map(str, kmers[:j])) for j in range(len(kmers), 0, -1)]:
    print("Assembling with", k)
    # define paths and basename to variables
    s = "./{o}/{s}/{s}_singles.fastq".format(o=OUT, s=sample)
    pe = "./{o}/{s}/{s}_paired.fastq".format(o=OUT,s=sample)
    base = "./{o}/{s}/{s}".format(o=OUT, s=sample)
    # call script with input variables from above definitions
    out = Popen("bash ./assemble_and_map.sh {s}.keep.fastq {base} {pfix}
{k} {pe} ${ref}".format(s=s, pe=pe, pfix=pfix, k=k, base=base,ref=ref),
shell=True, stderr=PIPE, stdout=PIPE)
    c = out.communicate()

assemble_and_map.sh:
# assemble contigs with spades
spades.py -k $4 -o assembly/$3 -s $1 -careful
# map paired reads to assembled contigs with bwa mem
bwa mem -a $5 assembly/$3/contigs.fasta > $2.sam
# convert mappings to bam, sort and index bam file
samtools view -bF4 $2.sam | samtools sort -o $2.bam -
samtools index $2.bam
```

These contigs were then used as a “reference” to which all the input reads (single and paired-end) were mapped to. Unmapped reads were then removed along with high coverage (>45) contigs. Tantan (v26 gitlab.com/mcfrith/tantan) was then used to filter out contigs that contain ≥80% tandem repeats, resulting in a fasta file of filtered contigs for further analysis (Frith 2011). A similar process is then repeated with the contigs in the vcf INFO (generated by dysgu) field to create a dataframe of reads mapped to contigs. Then dodi

(github.com/kcleal/dodi) was used to select the optimal set of alignments from mapping of the contigs to a reference genome.

```
... ref=command line input (eg. /path/to/hg38.fa)
# define file paths
all_contigs_file = os.path.join(outdir, "all_contigs."+pfix+".fq")
dodi_bam_file = os.path.join(outdir, pfix+".bwa_dodi.bam")
dodi_psl_file = os.path.join(outdir, pfix+".bwa_dodi.psl")
file_path = os.path.dirname(os.path.realpath(__file__))
# run dodi_pipe script
call(f"{file_path}/./dodi_pipe.sh {all_contigs_file} {ref}
{dodi_bam_file}", shell=True)

dodi_pipe.sh:
# align contigs to reference, pipe output to dodi, convert output to bam
bwa mem -c 1000 -A2 -B3 -O5 -E2 -T0 -L0 -D 0.25 -r 1.25 -d 200 -k 11 -a -
t12 ${2} ${1} | dodi --paired False -c 1 -u 21 --ol-cost 2 --max-overlap
50000 --min-aln 50 - | samtools view -bh - | samtools sort -o ${3};
samtools index ${3}
```

The maximum mapping score is used to find the optimal path, iterating to find the highest value of: `node_scores[j] - (micro_h * hom_cost) - (ins * ins_cost) - jump_cost + next_score` (where costs are defined as constants: ins_cost=2, hom_cost=1.5, inter_cost=20, intra_cost=10, and micro_h and ins are calculated from the difference between the current end and next start point) (see full algorithm at github.com/kearseya/dysgu-pipelines/tree/main/RRAssembler). This process is used in creating a dataframe of the best alignments. Directional graphs are then used to find common sites so that they can be removed. The mapping information is then extracted to be used for visualisation.

2.14 Telomere prediction software

TelSeq (v0.0.1/2 github.com/zd1/telseq) appeared to be the most commonly used telomere prediction tool based on citation counts and github stars. They define reads as telomeric if they contain 7 or more TTAGGG repeats, and estimate telomere length using a count of telomeric reads, a size factor, and genome size (Ding et al. 2014). For the local BGI data cohort it was called with the file list option:

```
telseq -f testlist.txt > ${out_path}/telseq_hap1.csv
```

However, in the genomics England research environment it was run individually for each sample, allowing it to be run in parallel with the following command:

```
while iterating sample_pairs.csv: # see common requirements
      telseq
```

Telomerecat (v4.0.1 github.com/cancerit/telomerecat) normalises telomeric content to subtelomeric regions, rather than against the whole genome like telseq to create an estimate. They also use the assumption that the number reads for 3' TTAGGG like Interstitial telomeric repeats (ITRs) is approximately equal to the number of 5' reads to correct for their (ITR) contribution (Farmery et al. 2018). It was run using a loop to run the analysis for each sample:

```
for i in $(basename -s .bam ${data_path}*.bam | uniq); do
   ${telcat_path}/telomerecat bam2length -p 10 --output ${out_path}
${i}.bam --temp_dir /scratch/ProjectDir/tmp
done
```

Computel (v1.3 github.com/lilit-nersisyan/computel) utilises a telomeric index (created with bowtie2-build) designed so that any read containing telomeric sequence can be uniquely mapped to it. The length of this index, and its mapped coverage is then used in conjuncture with number of telomeres and mean base coverage from the reference to create an estimate (Nersisyan and Arakelyan 2015). The same loop was also used for running:

```
for i in $(basename -s .bam ${data_path}*.bam | uniq); do
   bedtools bamtofastq -i ${i}.bam -fq ${i}.fq1 -fq2 ${i}.fq2
   ${computel_path}/telomerecat -1 ${i}.fq1 -2 ${i}.fq2 -o ${out_path}
   rm ${i}.fq*
done
```

Whilst qmotif (v1.2 github.com/AdamaJava/adamajava/tree/master/qmotif) isn't necessarily a telomere prediction tool, it can be used to make a prediction using the following code:

```
for i in $(basename -s .bam ${data_path}*.bam | uniq); do
   java -Xmx20g -jar ${qmotif_path}qmotif.jar\
      -n 8 \
      -bam ${data_path}${i}.bam \
      -bai ${data_path}${i}.bam.bai \
```

```
        --log ${out_path}${i}.qmotif.log \
        -ini /home/User/tools/adamajava/qmotif/qmotif.ini \
        -o ${out_path}${i}.qmotif.xml \
        -o ${out_path}${i}.telomere.bam;
done
```

2.15 Other software

2.15.1 Kmer analysis

Counts for kmers were gathered to try a reference agnostic approach for discovering sequencing depth discrepancies between different sequencing technologies. Kmer counting was done using jellyfish (v2.3.0 github.com/gmarcais/Jellyfish) (Marçais and Kingsford 2011).

```
# for each bam file
for i in /scratch/ProjectDir/aligns/*.bam;
    do b=$(basename ${i} .bam);
    # if analysis not been performed on file
    if [ ! -f /scratch/ProjectDir/jellyfish/done/${b}.bin.done ];
    then
        # run jellyfish on bam file
        jellyfish count -m 21 -s 3G -t 8 --if filter.fa -o
/scratch/ProjectDir/jellyfish/${b}.jf <(samtools fasta ${i});
        # convert jellyfish file format to binary array
        jellyfish dump -c ${b}.jf | python convert.py ${b}.bin;
    fi
    rm ${b}.jf;
done
```

Where convert.py was a simple python script to convert the binary jellyfish file to readable text:

```
import sys
import array
# take command line argument (${b}.bin)
outf = sys.argv[1]
# initialise binary array
a = array.array("B", [])
# for line piped from jellyfish dump
for line in sys.stdin:
    # strip whitespace and take value from 2nd col as an integer
    v = int(line.strip().split(" ")[1])
    # replace value above 255 with 255
    if v > 255:
        v = 255
    # add to array
    a.append(v)
# write array to binary file
```

```
a.tofile(open(outf, "wb"))
```

This data was then analysed by an MSc student Laurie Fabian who was supervised by myself and Kez Cleal.

2.15.2 Low coverage copy-number testing

To simulate low coverage sequencing data for testing the lower limit of the copy number pipeline, bam files were subsampled with samtools and a new coverage profile generated for these subsampled reads.

```
# iterate sample_list.csv with columns as path, readlen, cov
while IFS=, read -r path readlen cov; do
    # define basename and sample name through removing stirng after .
    b=$(basename ${path})
    s=${b%%.*}
    # calculate fraction of coverage to achieve desired coverage
    frac=$(awk -vcov=${cov} 'BEGIN { print 1 / cov }')
    echo "${s}"
    # if file does not exist
    if [ ! -f coverage/${s}_cov.bed ]; then
        # subsample bam, measure depth, and bin depth to windows
        (samtools view -hb --region-file hg38_chroms.bed -s ${frac} --
subsample-seed 123 ${path} | samtools depth - | python3 windows.py ${s}) &
    else
        echo "${s} already done"
    fi
    # run 8 jobs in paralell
    jobs=($(jobs -p))
    while (( ${#jobs[*]} >= 8 ))
    do
        sleep 30
        jobs=($(jobs -p))
    done
done < sample_list.csv
```

Here is one example where the while jobs loop was used to parallelise a process to speed up run time. It is however not exactly 8x faster with this method, due to the I/O limitations of the hard drive they are stored on. The output of the depth calculation script is piped into "windows.py", a script that collects the depth values and averages them over a window pre-specified in a bed file.

```
import pandas as pd
import sys
from rich.progress import Progress
import fileinput
```

```
# read predefined 10kb windows for hg38 as dataframe (df)
df = pd.read_csv("hg38_prep_10k.bed", sep="\t", header=None)
# add column names
df.columns = ["chrom", "start", "end"]
# add new column depth withh value 0 for all rows
df["depth"] = 0

# take sample name from command line input
sample = sys.argv[1]

# create d (depth) dictionary and chroms set
d = {}
chroms = set()

# for the chromosomes in window df, add key chromosome with value array
for i in df.groupby("chrom"):
    d[i[0]] = []
    chroms.add(i[0])
    # for each window in chromosome, add list to previously made array
    for r in i[1].itertuples(index=False, name=None):
        d[i[0]].append(list(r)[1:])

# make total value for progress bar
total_spaces = 0
for c in d:
    total_spaces += int(d[c][-1][1])
print(total_spaces)
total_spaces = total_spaces/100

# make the progress bar
with Progress() as progress:
    task = progress.add_task("Getting coverage: ", total=total_spaces)
    # for line piped into program (from jellyfish dump) add depth to window
    for line in fileinput.input("-", encoding="utf-8"):
        l = line.split()
        if len(l) > 0:
            if l[0] in chroms:
                d[l[0]][int(l[1])//10000][-1] += int(l[2])
                progress.update(task, advance=0.01)

# divide total depth of each position by window size
for c in d:
    for i, _ in enumerate(d[c]):
        d[c][i][-1] = d[c][i][-1]/(int(d[c][i][1])-int(d[c][i][0]))

# convert into dataframe and write to csv
rec = []
for c in d:
    for i in d[c]:
        rec.append([c]+i)

df = pd.DataFrame.from_records(rec)
df.to_csv(f"coverage/{sample}_cov.bed", sep="\t", index=False,
quoting=None)
```

This method was quicker than writing the subsampled bam to an intermediate file and measuring the coverage using any other tool. There is a command within bedtools which

should be able to achieve the same output (coverage bed from piped bam without index), however this did not work.

2.15.3 WGS type conversion

Due to the size of the local bam file dataset, some of the dataset was converted to cram format to save on disk space.

```
for i in $(ls /scratch/ProjectDir/aligns/*.bam); do
    name=$(basename $i .bam);
    samtools view -C -T /scratch/ProjectDir/hg38.fa ${i} >
/scratch/ProjectDir/aligns/${name}.cram;
    rm ${i};
done
```

This process was stopped early to retain some bam files for software development to ensure that processes work on both formats. The split between bam and cram is why separate loops for bam and cram may be seen in code, as well as the s=${b%%.*} method (as seen in the start of the low coverage test loop) for extracting sample names in the previous section.

2.16 Contributions to other projects

Minor bug fixes were contributed to the dysgu repository throughout its development, including in the structural variant filtering, and Linux installation related scripts (github.com/kcleal/dysgu). Functions related to the parsing of vcf information (specifically that of INFO and SAMPLE column fields) were contributed to the gw genome browser, allowing for the printing of the currently viewed SV vcf information to terminal, and filename modification of snapshots. Several contributions were also made regarding the installation process, including Windows OS (and android) proof of concepts (with accompanying Visual Basic install script), alongside Debian packaging build process (github.com/kcleal/gw).

Chapter 3 Prediction of telomere length from WGS data using machine learning

3.1 Abstract

Telomeres are the structures at the ends of chromosomes that protect the terminal DNA from being recognised by the DNA damage response pathways. Due to semiconservative replication, telomeres shorten with every cell cycle, and can become dysfunctional if they become too short leading to complex chromosomal rearrangements such as chromothripsis. To investigate the relationship between telomere length and genomic complexity, a software called teltool has been developed to predict telomere length. Teltool has two methods for predicting telomere length, one based on coverage analysis in defined genomic regions, while the second assessed substring (kmer) abundance. In both methods, reads containing telomeric repeats (telmers) are analysed to create a series of variables that are then passed to a machine learning model. Samples with telomeres above and below 3.81kbp are classified using the model, and we show that teltool outperforms previous software tools for predicting telomere length, with a 0.2 increase in F1 score (~0.7 compared to ~0.5), and two-fold improvement in precision (0.67 compared to 0.33). Development of an accurate telomere length prediction tool was a prerequisite for analysis of population cohorts of the International Cancer Genome Consortium, and Genomics England databases to investigate the relationship between telomere length, prognosis, and genomic complexity

.

3.1.1 Aims

The aim of this project was to develop a method for accurately predicting or classifying telomere length from Whole Genome Sequence (WGS) samples. Telomere length can be accurately measured using the Single TElomere Length Analysis (STELA) protocol; however, this is not commonly performed alongside genome sequencing (Baird et al. 2003). Beyond developing a software package for predicting telomere length, the output from this tool was analysed alongside genomic complexity to investigate their relationship.

3.1.2 Data

The whole genome sequencing datasets used for training the machine learning models consists of 44 tumour-normal pairs from a breast cancer cohort, and 11 samples from experimental cell cultures (ATRX knockouts). All samples have been sequenced at a depth of approximately 15-20 fold coverage, aligned to human reference hg38, and include accompanying clinical and STELA data.

3.2 Introduction

As described at the beginning of the first chapter, telomeres are located at the end of chromosomes and are comprised of TTAGGG repeats which together with shelterin form structures which primarily function to protect DNA from DNA damage response (DDR) mechanisms. Due to semiconservative replications, they shorten with every cell division in differentiated somatic tissues, and excessive shortening can lead to the loss of their protective function. Normally, in the presence of tumour suppressive mechanisms such as p53 and Rb mediated pathways, cells with short telomeres enter a stage of replicative arrest (senescence).

However, bypassing of these pathways can lead to continued shortening where cells then enter telomere crisis. Dysfunctional telomeres in crisis are prone to initiating periods of genomic instability through fusions which through various resolutions (breakage-fusion-bridge cycling, or micronuclei formation) create large-scale genomic rearrangements. This process is thought to occur during early carcinogenesis, and telomere length has also been shown to be predictive of treatment response in chronic lymphocytic leukaemia (CLL), as well as prognostic in both CLL and breast cancer. In this chapter, we utilised a breast cancer cohort of local samples and develop bioinformatics tools and methods, in preparation for analysis of larger cohorts available within public repositories which do not contain assay measured telomere lengths.

3.2.1 Breast cancer

Breast cancer is the most common malignant cancer among women and has a heterogeneous origin (Guo et al. 2023). Frequent molecular attributes of breast cancer include activation of human epidermal growth factor 2, hormone receptors (oestrogen and progesterone), and BRCA mutations (Harbeck et al. 2019). However, this knowledge of cancerous features is still incomplete. Currently patients are diagnosed and stratified for treatment by a limited set of factors such as biomarkers or driver mutations (Schick et al. 2021). There is a great interest in utilizing whole genome sequencing (WGS) data to better inform prognosis and clinical treatment (Rossing et al. 2019). It has come to light that breast tumours exhibit specific mutational signatures which are attributable to the underlying mutational processes (Nik-Zainal et al. 2012). In addition, rearrangement signatures have been identified that are indicative of deficiencies in homologous recombination activity that arise as a consequence of mutations in BRCA1 and/or BRCA2 and other as yet undefined mechanisms (Morganella et al. 2016). Telomere dysfunction has been identified as a key mechanism responsible for

genomic instability in breast cancer (Simpson et al. 2015). The aim of this chapter is to investigate whether it is possible to use machine learning with WGS data to predict telomere length.

3.2.1 Telomere length prediction methods

There are several previously released programs that aim to tackle the approach of telomere length prediction from whole genome sequence data which were briefly discussed in the previous chapter. In this section we will discuss their methodologies in more detail.

3.2.1 Telseq

Telseq is first and perhaps the most regularly used telomere prediction tool, based on the number of citations. Using the TwinUK dataset, they measured the frequency of reads containing varying numbers of TTAGGG (and its frame shifted permutations) repeats. The data showed an overall decline in frequency for each permutation as number of repeats (k) increased, except for TTAGGG which began to increase after 7. This data was plotted against mean length of terminal restriction fragments (mTRFs), which revealed a correlation of ~0.6 for read frequency at k=7. This is the justification they use to classify a read containing k>=7 as telomeric. Using the abundance of telomeric reads, they construct an estimate for telomere length by multiplication with the fraction of all reads with GC% between 48 and 52 (s), and length of the genome divided by number of telomeres (46). The results of their telomere length estimate correlates well with mTRF and age. Technical repeats also showed its reliable replication of results, with standard deviations not leaving expected ranges for experimental error (Ding et al. 2014). As pointed out by the authors of the next tool, the assumption of a fixed number of chromosomes leads telseq to overestimate the length in cases of aneuploidy.

3.2.2 Telomerecat

Telomerecat utilises a few methods to avoid assumptions made by telseq. The first is to use the ratio of read pairs on the telomere/sub-telomere boundary to estimate the number of telomeres. Second, they avoid the inclusion of interstitial telomeric repeats (ITRs) by assuming they contain approximately the same number of reads at the 3' end as the 5' end. With this assumption, they estimate and correct for ITRs, whilst avoiding the challenge presented in mapping these reads. The last is a method for accounting for sequencing errors of telomere and subtelomeric reads, with a model that "automatically adapts to differing error across sequencing preparations". Number of reads are segregated into reads that are completely telomeric (F1), one end of read contains CCCTAA (F2), and one end of read contains TTAGGG (F4). They then estimate the number of reads covering a boundary using (F2a=F2-F4), with the assumption any F4 read will have an F2 companion. Across the cohort, they then define the set of parameters for the distribution θ as F2a/(F2+F4). The insert means (μ) and standard deviations (σ) is also calculated for the cohort, which is then used to calculate the fidelity ($\psi = \mu / \sigma$). This is then used for error correction, using the formula $F2a = \theta^{corr} \bullet (F2 + F4)$. (Farmery et al. 2018).

3.2.3 TelomereHunter

TelomereHunter is takes a far simpler approach, they approximate the telomere content of an input bam file instead of estimating telomere length. They achieve this by extracting reads with (floor(read length x 0.06)) telomeric repeats, then sorting them into bam files for four categories: chromosomal, subtelomeric, junction spanning, and telomeric. Whilst scanning the input bam file, read GC content is also collected, and the number of reads where the GC% is between 48 and 52% is counted. Sorting is performed by classification of regions of the

reference into bands of: chromosome, first/last (subtelomeric), and telomere. Reads are placed in the telomeric category, if both the read and its mate have a mapping quality below a threshold (default 8). The rest of the categories depend on which band they are mapped to. The junction spanning group includes pairs of reads where one mate is has a mapping quality < 8 and the other is mapped to the first or last band of a chromosome. Subtelomeric category holds reads where both mates are aligned to a first or last chromosome band. Lastly, intrachromosomal read pairs include those with one mate mapped to a band that is not the first or last band of a chromosome. They then estimate telomere content per million reads as the number of intratelomeric reads * 1,000,000 / total reads with telomeric GC content (Feuerbach et al. 2019).

3.2.4 Computel

Computel is unique in the fact it uses fastq as input rather than bam (used by all other prediction software). It makes use of a telomere index which is built with the bowtie-build2 program and can be altered by the user to consider the telomeric repeat pattern and read length. The fastq is aligned to this telomere index using bowtie2-align, and the coverage is taken across each point of the mapped region using samtools depth. This telomeric coverage is converted into relative coverage which is done by dividing it by an estimate of the base coverage (n total reads * read length / total genome length). The mean of the relative coverage (MRC) is then used to calculate an average telomere length estimate using: MRC * (read length + repeat pattern length – 1) / (2 * number of chromosomes) (Nersisyan and Arakelyan 2015).

3.2.5 qMotif

This tool was not developed solely for the purpose of telomere length prediction; however, they do provide the parameters required for this task. It uses a two-stage approach to filter reads into four categories: include (telomeric), exclude (centromere/ITR), genomic, and unmapped. The first stage involves a string (faster) match, which if passed leads to determining the region the read is mapped to. If the region of the read is within the include or exclude bounds, reads are passed into stage two which uses regex matching (slower) to determine the number of defined motifs were seen in reads from that region. Data is then output in xml format which includes a summary of input parameters and raw counts of motifs and reads per region, motifs which shows the counts of each motif for a given region Stage 1 and 2 coverage is also recorded along the process, and values from the output file can be used to generate an estimate for the telomere length or content (github.com/AdamaJava/adamajava/tree/master/qmotif) (Holmes et al. 2022).

3.3 Methods and Results

3.3.1 Contextual work

As mentioned above, several software packages for estimating telomere length from genomic data are available. The most prominent of these tools are TelSeq, TelomereCat, and CompuTel (Ding *et al.* 2014; Farmery *et al.* 2018). However, CompuTel was not initially investigated (due to requiring fastq as an input creating issues with storage space). Whilst not directly used for telomere length estimation, qmotif could be adapted with its parameters to estimate the telomere lengths of WGS data (Holmes *et al.* 2022). Details on how these tools were run are shown in Chapter 2 Telomere prediction software section. Unfortunately, as we

show below, these tools are not accurate in their predictions when benchmarked against the STELA method of telomere length measurement. Performance can be measured using mean absolute error (MAE), which is calculated with:

$$\frac{\Sigma|actual - prediction|}{n_{samples}}$$

The mean average error for TelomereCat was 2140bp, TelSeq 2310bp, and qmotif 1917bp. Absolute errors for each tool were also plotted (figure 3.1). To put these numbers into context, the telomere lengths in the breast cancer samples in this dataset range between 2kb and 7kb. On average, a sample with a mean telomere length could be predicted to be on the outer edges of the real distribution.

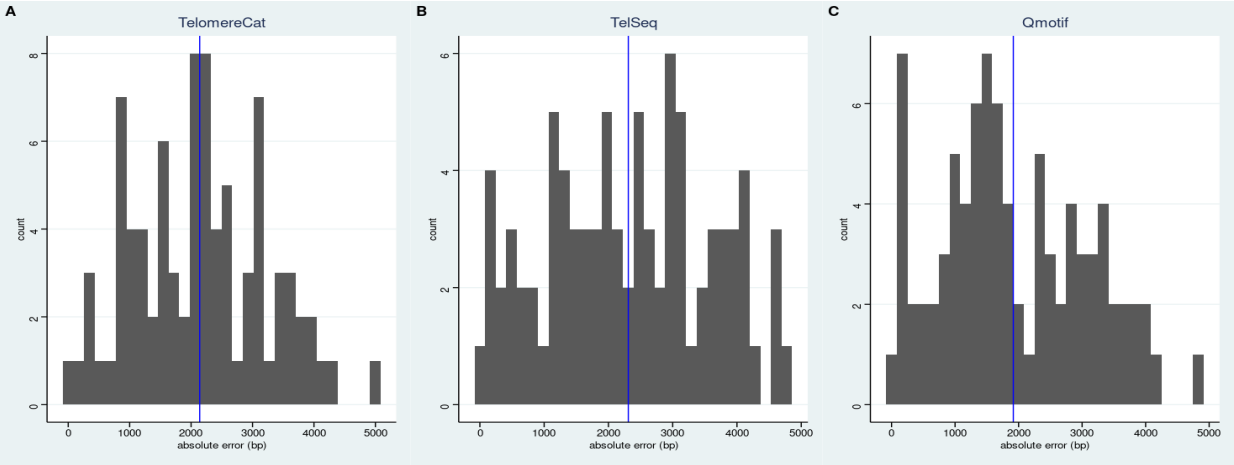


Figure 3.1 *Histograms of absolute errors for TelomereCat (A), TelSeq (B), and qmotif (C). Mean absolute errors shown as blue vertical lines (A) 2140, (B) 2310, (C) 1917.*

Previous analysis performed by Michalis Mylonas, an MSc student who worked at the Baird laboratory, showed that supervised machine learning can be used to improve the predictions of telomere length from TelSeq and TelomereCat. Initially the approach from Michalis' work was used as the basis for creating a pipeline that would analyse telomere length. A python module was written that could combine the results output by TelSeq, TelomereCat, and qmotif to create the input for a random forest regression model from the scikit-learn

library (Pedregosa et al. 2011). This considerably reduced the mean absolute error from around 2000bp to between 883 and 682bp depending on the percentage contamination removal.

For percentage contamination removal, Isolation Forest modelling was used by iterating through contamination values (0 to 20%) to selectively remove samples from training to see how it improved regression results. Isolation forest is an unsupervised learning algorithm for anomaly detection, it utilises separation of sub-samples and fitting then comparing tree lengths to identify anomalies (Liu et al. 2008). Utilising an Isolation Forest was problematic however, as the best performing model used a contamination value of 19%. This removed all samples with telomere lengths, as determined by STELA above 6.2kbp and below 3.2kbp significantly narrowing the input range. As a result, the model would tend to predict all samples between 4 and 6kbp (figure 3.2). Since the main interest of the project is investigating short telomeres, this was deemed unsuitable for further development.

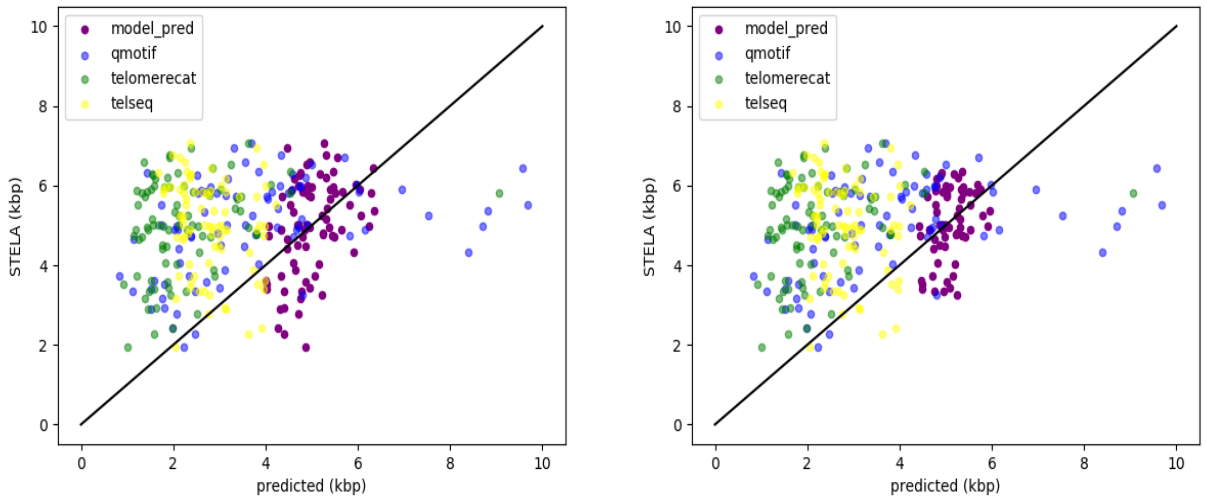


Figure 3.2: Scatter plots of prediction against STELA (in kbp) for the model using contamination

This method also presented some other issues, including a large number of dependencies (software required to be installed on the system) which hinders the ease of use. Setting up of TelomereCat was especially difficult, requiring the use of several package managers pip and conda, downloading an executable binary, and testing/building of several docker containers were attempted and were unsuccessful. Analysis for large number of samples was also computationally expensive with long run times for two of the tools used (40 minutes/sample telseq, 6+ hours telomerecat). This would also potentially cause an issue when analysing cohorts containing larger numbers of samples which was planned later in the project (Chapter 5). Qmotif was significantly faster as it is a region-based method, taking around 2 seconds per file. Also depending on tools that others had written meant it would be difficult to improve the performance much further. It was decided that a new approach that could analyse the raw data directly should be investigated.

3.3.2 Region-based method

The issues faced above lead to developing a new software called teltool in attempt to more accurately predict telomere length. Teltool extracted reads that had been aligned to telomeric regions, which has the benefit of being a fast method for finding and extracting reads containing telomeric sequence. However, it can unintentionally miss reads of interest which could mapped to other loci due to sequencing variance and mapping complications. Features are extracted from these reads corresponding to properties that were considered to potentially be useful for predicting telomere length. For example, each region would have the average GC%, mapping quality, telmer (telomeric substring e.g. blue CCCTTAGGGCCC) length, and number of separated telmer sets (“fragments"). The resulting data along with the STELA measurements could be passed to a supervised machine learning algorithm.

Extracting of the reads from raw files is handled by pysam, a python wrapper for the

htslib C-API, the library that powers samtools (github.com/pysam-developers/pysam) (Li et al. 2009). Telomere regions were initially gathered from the UCSC genome browser table browser, with a filter for 'CCCTAAn' kmer (substring) repeats using the repeat masker annotation table. Later these regions were checked with a brute force search to find all the telmer (telomeric kmer) containing regions in the hg38 reference genome. This search was performed using a python script to perform a sliding window search of chromosome contigs. Each region had all the included reads analysed and processed together for that region (figure 3.3). Regions shown in Figure 3.3 are not located at the telomere ends, closer instead to the centromeres, which are interstitial telomeric repeats (ITRs). They were included as its possible for telomeric reads to be mis-mapped to these repetitive regions.

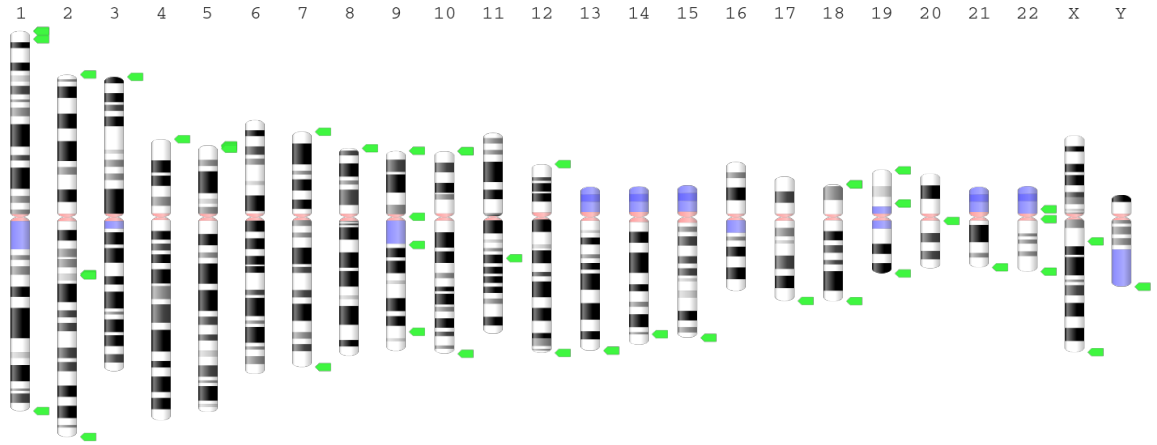


Figure 3.3. Genomic locations on hg38 containing CCCTAAn substrings as found by brute force search visualised by UCSC genome browser

3.3.3 Coverage

Coverage was considered a useful feature for telomere prediction as a higher coverage in telomeric regions would theoretically be predictive of longer telomeres. Collecting the average coverage for each sample was initially included (but removed later) as a variable for the model. The first method for estimating whole genome coverage was to measure the coverage of 1000 1kbp random regions from each sample and take the median. This

subsampling method was chosen as it was appreciably faster than parsing the whole bam file. Based on the TelSeq methodology, random regions were changed to the same number of positions on the genome instead that had a 48-52% GC% as it improves performance as PCR can exhibit GC biases inflating coverage in areas of higher GC (affirmed by appendix 2) (Dohm et al. 2008; Ding et al. 2014). To save time these regions were calculated prior measuring from the reference genome, but new sets of regions could also be calculated and measured at run time. This approach was subsequently considered inaccurate after measuring the true coverage value with `samtools depth`, one sample estimated at 15x with the sampling method was measured to be 18x using samtools (Li et al. 2009). This was likely because there is so much variance of coverage over a whole sample that either not enough samples of coverage were taken or the region sizes for sampling from were too short.

The whole sample coverage average was also attempted to be measured using both a cython and C++ implimentation of Indexcov, a tool written in the go language which gives a close approximation of the coverage of a sample using the index of a bam file. This tool reads the number of bytes within the index over 16,384bp tiles to determine a “proxy” coverage value (Pedersen et al. 2017). Although the method was much faster than the previous methods, there were some issues with cython objects reading and storing the bam index files . The coverage problem was solved with an equally fast method which uses the index to get the number of mapped reads in the sample and multiply by the read length, then divide by the total reference length. The number of mapped reads was read from the index with the pysam `get_index_stats` function. With a fixed 0.98x multiplier to account for not all reads being exactly 100bp, this yielded coverage predictions within 0.1 (±0.7%) of the samtools depth values, although the effect on different sequencing runs and read lengths is unknown and might require adjusting.

Coverage values for individual regions were measured with an implementation of the

mosdepth algorithm. Mosdepth utilises an elegant algorithm that processes cigar strings from aligned reads to cumulatively add where new alignment blocks begin and subtract where alignment blocks ends (Pedersen and Quinlan 2018). Using the cigar strings also allows accounting for INDELs, as Pedersen and Quinlan in Figure 3.4.

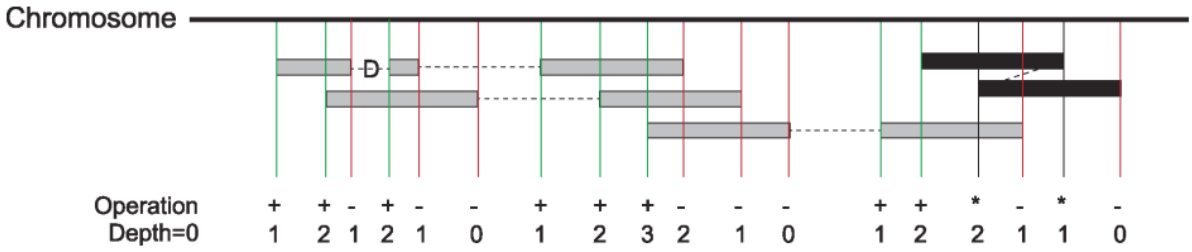


Figure 3.4 Figure from the mosdepth paper explaining the method. The value at the start of each read is incremented and each stop is decremented. As the CIGAR string is used, deletions (D in figure) can be taken into account as an "stop". The left most read also has the mate stored in hash table so if the mate is encountered, the identifier is removed from the hash table, and the read is not counted twice (*) to prevent double counting. Once all reads in the specified region have been encountered, the per-base coverage can be calculated as the cumulative sum of the positions measured. Reused under the Creative Commons licence form: Pedersen BS, Quinlan AR. Mosdepth: quick coverage calculation for genomes and exomes. Bioinformatics. 2018 Mar 1;34(5):867-868. doi: 10.1093/bioinformatics/btx699. PMID: 29096012; PMCID: PMC6030888.

3.3.4 Feature selection

Feature selection is the selection of a subset of the most relevant variables (features) for construction of a model. It is an important part of machine learning as it reduces noise and complexity (dimensions) of a model, generally improving its performance (Guyon and Elisseeff 2003). The feature selection functions explored during teltool development include, performing removal of low variance features, univariate, sequential, and feature selection from model, recursive feature elimination, and removal of correlated features. The best combination was found to be feature selection from model then recursive feature selection.

Features with low variance will most likely not have any predictive power as there will be little distinction between samples. Univariate feature selection involves individually examining the strength of each feature's relationship with the different outcomes. Sequential feature selection adds or removes features in a greedy fashion (i.e. creates the local optimal

with the hope of finding the global optima) and uses cross-validation to determine when the best set of features has been selected. “From model” feature selection is when the model is fit to all the provided data, and only features with feature importance value above a threshold are kept. Feature importance is a weight the model gives a variable to describe its predictive power. Recursive feature elimination is a similar process to feature selection from model, except that the least important feature is eliminated, the model is refit, and the process is repeated until a specified number of features is selected (Pedregosa et al. 2011). Removing correlated features allows simplification of the model by condensing a pair of features into a single variable.

3.3.4 Feature modifications

Unmapped reads composed of >50% telmer were also included at one stage of development. However, it was found that including these significantly increased run time (+10 minutes/sample) but not the prediction power of the model. Small regions (<100bp) were also found to not increase prediction power so were removed. All regions between 100-700bp were also grouped into their own category named “short" instead of giving them their own region. It is suspected this improved performance due to variance introducing noise that could potentially confuse the model from multiple features providing limited information being condensed whilst retaining the information and denoised into one variable.

3.3.5 Normalisation

Data to be analysed may not have similar sequencing depth (eg. ICGC 30-80x coverage compared to the breast cancer cohort 15-20x), therefore normalisation is required to allow the model to work accurately on new data. Average coverage was not included as a feature as the model would not have seen values much beyond 20 (due to training on the breast cancer

cohort only). Number of reads, read and telmer lengths all needed to be normalised for coverage and read length values. Normalising by coverage involves dividing values of the coverage of regions of interest by the sample's average coverage. Whilst not a significant variable and eventually not being used, it is worth mentioning that read length was also normalised to length 100bp.

3.3.6 Regression performance

The end performance for regression analysis using the region-based method was an improvement over the previous method using the pre-existing software. It was found that random forest regression was the best model type with recursive and "from model" feature selection. Other regressors such as: lasso, ridge, linear regression, ransac, elastic net, multi-layered perception from scikit-learn and lightGBM's light gradient boosting machine were also tried (Pedregosa et al. 2011; Zhang et al. 2017). Using a random forest model, the mean average error was 771, with the largest over prediction being +2761bp away, and furthest under prediction being -1777bp (figure 3.5a). These values are close to the average errors seen by the previous tools (~2000bp). The R squared was 0.332, and the root-mean square error was 959. This method was also able to overcome the issue with the previous method (combining the results from pre-existing software and passing values to a machine learning model), and had a wider predictive range with predictions between 3.2kbp to 5.9kbp (compared to 4kbp to 6kbp). Plotting of the predicted value divided by the real value shows that in most cases the predictions are within 10% either side of the real values (figure 3.5b). As the predictions from this regression model are going to be used to stratify samples by an already known value, the question was asked if a classifier would be able to be more accurate

at this task.

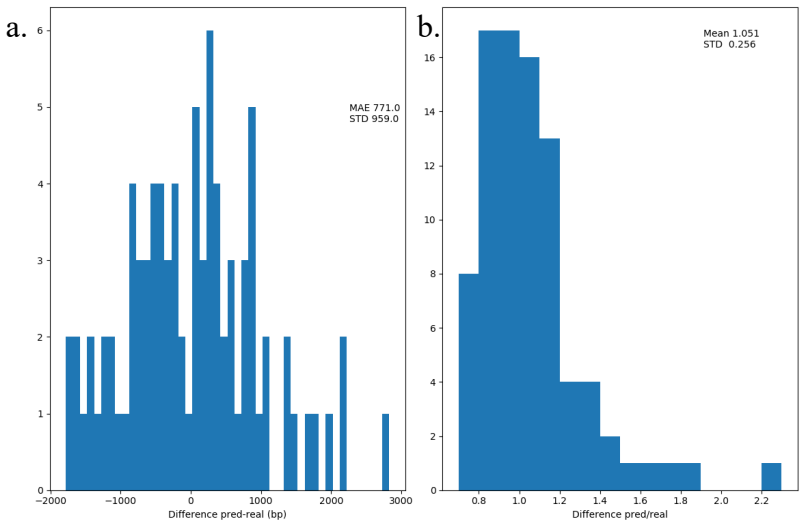


Figure 3.5: Difference plot for regression analysis (A) and prediction divided by real (B)

3.3.7 Changing from regression to classification

As the telomere length predictions were likely going to be categorised in further analysis, it was suitable to change focus from predicting the absolute telomere length with a regression model to a classification system. Using a classifier based method would remove any form of ranking label, but could potentially be more accurate than segregation through a regression predicted value. The fusion threshold of 3.81kbp (upper TL telomere fusions were found to occur, see 1.4.1) discovered previously by Duncan Baird et al was used to categorise samples into long and short (Capper *et al.* 2007). These labels were then used in the supervised learning of classification models.

The Light Gradient Boosting Machine Classification from the python module LightGBM proved to have the best performance. Other classifiers from the sklearn module were also experimented with such as: K nearest neighbours, C-support vector, gaussian process, decision tree, random forest, multi-layered perception, ada boost, and quadratic discrimination analysis (Pedregosa *et al* 2011). The metrics for the lightGBM classifer were

an F1 score of 0.6 (accuracy 86%), precision 0.75, recall 0.5, and specificity of 0.96 for

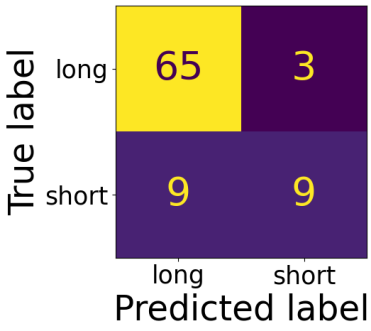


Figure 3.6 Confusion matrix for region-based method classification

predictions using leave one out cross validation (table 3.1, figure 3.6).

The telomere length predictions from the previous software were also used to classify samples into long and short using a threshold of above and below 3.81kbp (figure 3.7).

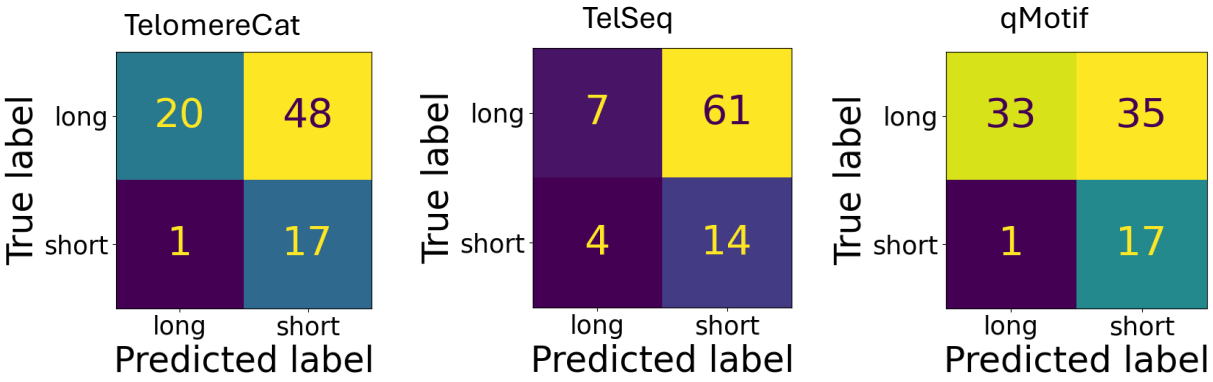


Figure 3.7: Confusion matrices for telomerecat (left), telseq (centre) and qmotif (right) using length prediction

The classification metrics of these software are also shown in Table 1. Overall, there is a trend with these software to have underestimated the telomere lengths across the cohort, as seen by the majority of the samples with true long labels being predicted as short (figure 3.7), and the high recall (>0.78) with very low precision (<0.33) (table 3.1).

	F1 score	**Specificity**	**Precision**	**Recall**
telomerecat	0.410	0.294	0.261	0.944
telseq	0.301	0.103	0.187	0.778
qmotif	0.486	0.485	0.327	0.944

	F1 score	Specificity	Precision	Recall
teltool (R)	0.600	0.956	0.750	0.500

Table 3.1 F1 score, specificity, precision, and recall for each software (R) indicating the region based teltool method

Metrics were calculated from the confusion matrices to evaluate the performance of the classification.

$$F1 = 2 \times \frac{tp \times tn}{tp + 0.5(fp + fn)} \quad , \quad Specificity = \frac{tn}{tn + fp} \quad , \quad Precision = \frac{tp}{tp + fp} \quad , \quad Recall = \frac{tp}{tp + fn}$$

In the equations above, t = true, f = false, p = positive (short), n = negative (long). F1 score is a harmonic mean of precision and recall, and is being used as the groups have unequal weighting (68 long, 18 short). Specificity is also known as the true negative rate. Precision is also known as the positive predictive value, the fraction of relevant results. Recall is also known as sensitivity or the true positive rate.

Whilst the recall for all the methods appears high, the F1 scores, specificity, and precision values are all under 50%. This shows that the previous software tools tend to overly predict samples as short (also shown in figure 3.2), whereas the region-based teltool method is much more balanced. Teltool using the region-based method also runs much faster than two of the other methods, with a run time of about 1 second per file (~1.5X faster than qmotif).

3.3.8 Reference issues

Unfortunately, throughout the majority of the region-based method development, data to be analysed with the tool was not available, so some assumptions had to be made. One of these assumptions was data would be aligned to the hg38 reference genome, and if not, the UCSC LiftOver tool would be sufficient to convert coordinates between hg38 and other references. Upon gaining access to the ICGC database, it was discovered the data was aligned to hg19. Using LiftOver to convert the coordinates used on hg38 to hg19, the compatibility of the

model with the hg19 regions was tested (example of data drift shown in appendix 3). After discovering that performance was heavily impacted by the change (as all samples were predicted to be short which seemed unlikely), it became apparent a new approach had to be considered.

3.3.9 Kmer-based method

Whilst the region-based method remains a function available in the tool, a separate kmer-based method was developed to tackle the issues encountered when dealing with input files that were aligned to different reference genome builds. This method involves extracting kmers of length k from all the telomeric regions from hg38 reference genome, as during the time of development the telomere-to-telomere T2T reference was not available (Nurk et al. 2022). Each read in an input bam file is then checked for the occurrence of this target list of kmers. This is performed using a sliding window approach, and if a matching kmer is found, the candidate read is saved to a fastq file. The resulting fastq files could then be aligned to a standardised reference genome, thus harmonising data analysis for files aligned to different references. In particular, this also controls for any variance caused by different alignment tools or other intermediate tools. A more detailed explanation is given in the overview of the next section.

It was decided that this method should be written in cython, a compiled language that allows interfacing between C/C++ and python. As python is a dynamic interpreted language, the performance benefit from reducing the overhead using a static compiled language adds up significantly over millions of reads per file. A pure-python version of the kmer method was written to compare the speed. Performing a similar process without manually hashing kmers (instead checking if strings are in a set), the pure python version performed 3.2x slower than the cython implementation. Not only is this a general

convenience, but import as time is money when processing massive online dataset using cloud infrastructure. For a sample with an average coverage of 17, the average run time is around 38 minutes and 30 seconds. This is comparable to the run time of TelSeq, but significantly faster than TelomereCat.

3.3.10 Detailed workflow

3.3.10.1 Pre-processing and Overview

A more detailed description of the process described on the previous page is provided here. Pre-analysis, a set of sub-strings (k=32) is collected from a reference fasta file at given coordinates. The coordinates used in the default are the regions from the region-based method, but where appropriate extended ±32bp to accommodate for sequences flanking these regions. Each sub-string is hashed using the rolling hash function (discussed below) and saved to a default python set. This set is then saved to a file using the dump function from the joblib library. Upon analysis, before files are scanned, this set is loaded from the file, and all hashes are added to an unordered set from the robinhood library, a C++ hashmap/set library written by Martin Ankerl, and was considered the fastest C++ hashmap available at the time (github.com/martinus/robin-hood-hashing).

All files within the directory supplied to the input flag are then listed using the listdir function from the python os library. Relevant file formats (bam, cram, and fastq) are then sent to respective scanning function. The scanning function will iterate over all the reads in the file. In the case of bam and cram files, when a suitable read pair (primary paired alignment) is found, the `pysam_bam_get_seq(AlignedSegment._delegate)` function is used to retrieve a pointer to the binary representation (nibble array) of the alignment sequence. The nibble (4 bits) array represents each base in a 4-bit format. This binary

representation can also be read as an integer, the key for each base can be seen below in the table below (table 3.2).

Base	Binary	Integer
A	0001	1
C	0010	2
G	0100	4
T	1000	8
N	1111	15

Table 3.2: Binary and integer representations for each base in the nibble array format used in bam

This binary representation is then passed to the rolling hash function (discussed below in Scan read and rolling hash functions section), which hashes each kmer then checks if it is in the unordered set of permitted kmers (previously filled out), and returns a binary integer (bint) 1 (true) if hash is in the set and 0 (false) if it is not. If the function returns true, a Python f-string is used to generate the fastq format of the read, and write it to the appropriate file. The process for fastq files is similar, except instead of a retrieving a pointer to the nibble array, the string is converted into a nibble array through a separate function. As previously mentioned, the “scan bam” function starts this process by extracting reads from a bam file.

3.3.10.2 Scan bam function

This function is responsible for iterating over all reads within a bam file to extract telomeric reads by scanning each one for telmers. The scan file function takes in the variables of the input file path, the output directory path, kmer hash set to check, reference path, and an optional Boolean (flag) keep fastq variable. The variable file is defined as a pysam AlignmentFile structure and reads in the input file with the read binary (code 1 “rb" line 2)

option using the AlignmentFile function (the read option is changed to “rc” in the read cram version of this function). Per the suggestion in the pysam documentation, AlignedSegments a and b are defined as this can improve efficiency. Total reads, matched reads are defined as integers with value of 0, and n_reads is defined as an integer with a value from the `get_n_reads` function (reads index to get total reads in file). The basename of the input file is also stored as a variable using the `os.path.basename` function, and is used in generating the names of the output fastq files. A telomere reference (contigs of regions from hg38 and pure TTAGGGn/CCCTAAn repeats) is also read in (as an Aligner object code 1 line 12) using the mappy module, a convenient python interface for minimap2 (pypi.org/project/mappy/) (Li 2018; Li 2021). For the bam function a progressbar from the progressbar2 module is also displayed, however, in the cram version this is absent (as number of reads can’t be extracted from index file).

For each read in the file (in AlignedSegment format), the flag is checked with the & operator to remove reads with the following bitwise flags set: “not primary alignment", PCR duplicate, or a supplementary alignment, and that the read is paired. The AND operation is used to check the flags as it uses less operations than getting the values from memory and comparing individually the AlignedSegment `.is_type` variables. When these conditions are met, the query name of the read is checked to see if it has been seen before. If the read has not been seen before, the AlignedSegment structure is added as a value in a dictionary with the key as the query name. When the read has been seen before, the previously seen read is assigned to the b variable, and the pair is passed to the `scan_read_pair` function, and if this returns true, a list of alignments to the telomeric reference is created. This alignment list is created with the mappy reference_object.map function for fast and accurate alignments. An alignment filtering stage is employed to remove reads that contain flanking kmers from the collection stage that do not contain telomeric sequence. If the reads do not align to the

telomere reference (i.e. alignment list is empty), then the loop will continue to the next read. If at least one of the reads do align, the for loop will allow them to be appended to their respective files (dictionary also maintains the read order). The fastq format is constructed with an format (f-)string. The break at the end of the for loop prevents the reads being written to the files multiple times, and the next read is processed (code 3.1 line 29).

```
 1  cdef scan_bam_no_bar(in_file, out_dir, l, robin_set[uint64_t]& kmers, reference,
keep_fq):
 2      cdef AlignmentFile file = AlignmentFile(in_file, "rb")
 3      cdef AlignedSegment a
 4      cdef AlignedSegment b
 5      cdef int total_reads = 0
 6      cdef int match_reads = 0
 7      read_pairs = dict()
 8      basename = os.path.basename(in_file)
 9      fastq_prefix = os.path.splitext(basename)[0]
10      first_file = os.path.join(out_dir, fastq_prefix+"_tel1.fq")
11      second_file = os.path.join(out_dir, fastq_prefix+"_tel2.fq")
12      reference_check = mappy.Aligner(os.path.join(os.path.dirname(__file__),
"reference", "hg38_cutout_edit.fa"),   preset="sr")
13      for a in file:
14          total_reads += 1
15          if not a.flag & 3328 and a.flag & 1:
16              if a.qname not in read_pairs:
17                  read_pairs[a.qname] = a
18              else:
19                  b = read_pairs[a.qname]
20                  del read_pairs[a.qname]
21                  if scan_read_pair(a, b, l, kmers):
22                      align = reference_check.map(a.query_sequence, b.query_sequence)
23                      for hit in align:
24                          match_reads += 2
25                          with open(first_file, "a") as paired1:
26
paired1.write(f"@{a.qname}\n{a.query_sequence}\n+\n{''.join([chr(i+33) for i in
a.query_qualities])}\n")
27                          with open(second_file, "a") as paired2:
28
paired2.write(f"@{b.qname}\n{b.query_sequence}\n+\n{''.join([chr(i+33) for i in
b.query_qualities])}\n")
29                          break
30      if reference != None:
31          run_alignment(out_dir, fastq_prefix, reference, keep_fq)
32      return (total_reads, match_reads)
```

Code 3.1 Scan bam file function

Once the fastq files have been generated, if a reference is provided, they will be aligned using minimap2 and converted from sam to bam format using samtools. Minimap2 is used because it is consistent with the mapping step used to filter the reads in the scan bam function. BWA also has a python binding (bwapy), but it is no longer maintained. The "keep_fastq" variable can be used in this function also to prevent the fastq files from being

deleted (code 3.1 and 3.2, line 1), this can be useful for debugging purposes.

```
1  def run_alignment(file_directory, file_prefix, reference, keep_fq=False):
2      ## align command to run
3      align_command = f"minimap2 -ax sr {reference} {file_prefix}_tel1.fq
{file_prefix}_tel2.fq -o {file_prefix}_tel.sam"
4      ## run command (outputs aligned SAM file)
5      align = subprocess.run(align_command.split(), cwd=file_directory)
6      ## remove fastq files if not wanted (trim -k)
7      if keep_fq == False:
8        os.remove(os.path.join(file_directory, file_prefix+"_tel1.fq"))
9        os.remove(os.path.join(file_directory, file_prefix+"_tel2.fq"))
10     ## convert to bam file
11     con_command = f"samtools view -hbS {file_prefix}_tel.sam -o {file_prefix}_tel.bam"
12     con = subprocess.run(con_command.split(), cwd=file_directory)
13     # pysam.view("-hbS", os.path.join(file_directory, file_prefix+".sam"), "-o",
os.path.join(file_directory, file_prefix+".bam"))
14     ## remove sam file
15     os.remove(os.path.join(file_directory, file_prefix+"_tel.sam"))
```

Code 3.2: Alignment command for converting fastq to bam files

Sorting and indexing of bam files is handled outside of this process in the trim class of teltool.py as the region-based method also requires these after trimming files by region.

3.3.10.3 Scan read and rolling hash functions

Read pairs are passed to the scan read pairs function after both have been seen in the file. Along with both reads, the integer variable denoting the length of kmer to search, and the hash set kmers are passed into the function. For read variable a, an unsigned char pointer is defined and given its value from the pysam function `pysam_bam_get_seq`. The bam1_t structure for the read is also retrieved, as nested within it contains the bam1_core_t structure which holds all the information about the reads (position, name, sequence, etc). The pointer to the binary sequence, length of the query sequence from the bam1_core_t struct, and the kmer set are then passed to the `rolling_nibble_hash_ptr` function. If this returns a 1 (true) value meaning the read contains a kmer in the hash set, then the scan read function returns a value of true, resulting in the read being saved for further analysis. If the hash function does not return true, the process is repeated for the second read. If the second read also does not

return true, the function returns a 0 (false) value (code 3.3), and the read-pair is dropped from further analysis.

```
1  cdef bint scan_read_pair(AlignedSegment r1, AlignedSegment r2, int l,
robin_set[uint64_t]& kmers):
 2      ## Rolling hash method
 3      cdef uint8_t* uint_ptr_rseq1 = pysam_bam_get_seq(r1._delegate)
 4      cdef bam1_t * src1 = r1._delegate
 5
 6      if rolling_nibble_hash_ptr(uint_ptr_rseq1, l, src1.core.l_qseq, kmers):
 7          return 1
 8
 9      cdef uint8_t* uint_ptr_rseq2 = pysam_bam_get_seq(r2._delegate)
10      cdef bam1_t * src2 = r2._delegate
11
12      if rolling_nibble_hash_ptr(uint_ptr_rseq2, l, src2.core.l_qseq, kmers):
13          return 1
```

Code 3.3: Scan read pair function

The rolling hash function creates a hash for all kmers in their binary form in a sliding window fashion. An empty hash value is initialised as an empty 64-bit wide unsiged integer block of memory (figure 3.8A). The hash is filled from right to left adding one base at a time until it is filled (figure 3.8B). Bit-shifting by two steps is used to move each value right-wards. Once the hash is filled the hash value is simply the number of the 64bit integer memory space. If this hash value is contained within the desired hash set then return 1 (True) (figure 3.8C). If the first hash is not in the set, the hash function iteratively checks each hash along the string (figure 3.8D).

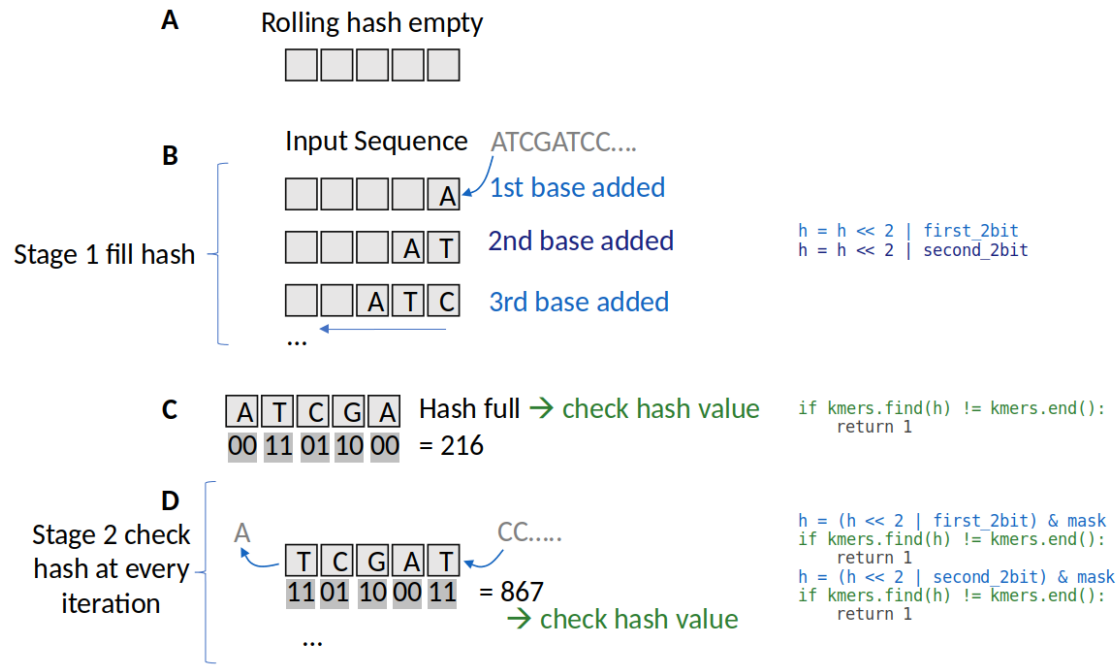


Figure 3.8. Rolling hash function workflow

of the sequence is encoded in 4 bits which can also be interpreted as an integer (see table 3.3). A nib array is defined before the function which is used for encoding the 4-bit nibble base representations into 2-bit (row 3 in table 3.3). The index of each bases nibble (4-bit) integer value contains a new integer value, all which can be encoded in 2 bits. One limitation of this method is that N values are encoded as A values, as 2 bits can only encode 4 bases, however this is a rare occurrence due to the low error rate of paired-end sequencing data (table 3.3).

base		A	C		G				T						N
base nibble		0001	0010		0100				1000						1111
base 4-bit int		1	2		4				8						15
nib (2-bit int)		0	1		2				3						0
nib (2-bit bin)		00	01		10				11						00

Table 3.3: Table to demonstrate the relationship between the integer values of original base nibbles

The rolling hash function first defines some variables: **i** which is used for iterating over sequence, **h** which is the hash being written to, and a **mask** which is used in the second loop to prevent the hash from overflowing from size k. Also, **v** the sequence bytes,

first_2bit, and **second_2bit** which are, first nibble and second nibble of **v** in 2-bit form respectively. The global interpreter lock (GIL) is released (code 3.4 line 6), preventing python pausing operation occasionally and potentially in future allowing parallel execution of code over multiple threads. A **mask** is created to limit the hash size. It is generated with a bitwise NOT operation comparing **h** to 0 setting all the bits to value 1. The string of 1s is then right shifted so they only occupy the last 2x k (length of kmer) bits. All bit-wise operations used here are described in appendix 4.

A while loop is used when **i** is less than half of the kmer length to recursively perform the following to fill the hash (code 4 line 11). The nibble array is assigned to the unsigned char **v** by de-referencing the pointer provided in the function input to the first byte (8 bits) of the nibble array. The **second_2bit** variable is assigned by getting the value in the nib array at an index equal to the integer returned by performing the bitwise AND operation of **v** with 15. The number 15 has the binary representation 00001111, so the AND operation returns the byte with the first nibble set to 0s. The **first_2bit** variable is retrieved in the same way, but variable **v** is right shifted by a nibble before performing the AND operation with 15.

The first 2 bits of the hash are then written by left shifting the bits of **h** by 2 and using the OR operator. The second 2 bits is written the same way, the bits in **h** are left shifted by 2 and it is OR-ed with the **second_2bit** variable. As both the first and second 2-bit variables can only have the values 0, 1, 2, and 3, the first 6 bits of these variables will always be 0. Left shifting **h** by 2 each time, the new 2-bit base encodings can be written to create a hash one by one. By incrementing the pointer and the **i** value inside the while loop, the first kmer of the sequence can be hashed as each byte of the kmer until length of the kmer is reached. After the first hash is created, it is checked to see if it is in the hash set and returns true if it is.

If the first hash is not in the set, a second while loop is used to hash and the remaining kmers in a sliding window fashion (when **i** is less than length of sequence) (figure 3.7D). These hashes are generated in a similar fashion, except **h** is bitwise AND operated with the **mask** to prevent the hash (**h**) from extending beyond the set kmer length. After each hash has been generated, it is checked against the kmer set. Any match will result in the function returning 1 (binary integer value for true), if none of the hashed kmers are present the function returns 0 (false) (code 3.4).

```
cdef uint8_t[16] nib = [0, 0, 1, 0, 2, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0]

## Takes pointer to binary sequence and performs rolling hash until in set or end of sequence
cdef bint rolling_nibble_hash_ptr(uint8_t* t, int kmer_length, int arr_len, robin_set[uint64_t]& kmers):
    cdef int i = 0
    cdef uint64_t h = 0
    cdef uint64_t mask
    cdef uint8_t v, first_2bit, second_2bit
    with nogil:
        mask = ~(h & 0) >> (64 - (kmer_length * 2))
        while i < kmer_length / 2:
            v = dereference(t)
            second_2bit = nib[v & 15]
            first_2bit = nib[(v >> 4) & 15]
            h = h << 2 | first_2bit
            h = h << 2 | second_2bit
            i += 1
            t += 1
        if kmers.find(h) != kmers.end():
            return 1
        while i < arr_len:
            v = dereference(t)
            second_2bit = nib[v & 15]
            first_2bit = nib[(v >> 4) & 15]
            h = (h << 2 | first_2bit) & mask
            if kmers.find(h) != kmers.end():
                return 1
            h = (h << 2 | second_2bit) & mask
            if kmers.find(h) != kmers.end():
                return 1
            i += 1
            t += 1
    return 0
```

Code 3.4: Rolling hash function

The speed from the rolling hash function was compared against that of the xxhash function from the xxhash64 C library by Stephan Brumme (github.com/stbrumme/xxhash),

an implementation of Yann Collet's algorithm which is widely regarded as one of the fastest high-quality hashing functions for short strings (benchmarks available at: github.com/Cyan4973/xxHash/wiki/Performance-comparison#benchmarks-concentrating-on-small-data-). This was done by sampling 1000 random times for reads that did not contain a kmer in the hash set and taking the mean time for the functions to hash all kmers length 32 for the forward and reverse sequence. The rolling hash function developed here was 1.5x faster with a mean time of 1.622×10^{-6} seconds compared to xxhash mean time of 2.453×10^{-6} seconds. The hashing function is one of the main bottle necks for computation, apart from the IO. xxhash is often the fastest known non-cryptography hash function, and usually is only limited by RAM speed. This method is so fast due to its small number of simple operations, once a mask is made, only 14 total operations (1 dereference, 2 array indexes, 3 AND operations, 2 OR operations, 2 bit shifts, 4 assign to variables) are required to generate the hash for a window. Additionally, outside of the N base case there are no collisions for this hashing method. On top of this optimised hashing function, other methods were attempted for further performance improvements.

3.3.11 Attempted optimisations

The function that iterates through the reads of a file uses several checks before writing the read to a fastq file. To test the speed of the various implementations, a trimmed version (first 10Mbp containing 43,782,139 reads) of one of the bam files was used. Simply iterating through all reads in this file takes 1 minute 17 seconds using pysam. Including "`if not a.flag & 3328 and a.flag & 1:`" which filters reads by primary alignment, not PCR or optical duplicate, not supplementary alignment, and to only paired reads increased this time by 3 seconds (+4%). The next filter uses a dictionary to check the query name to test read pairs increases this runtime by a further 25 seconds (+31%). Finally, the "`scan_read_pairs`"

function which checks if the read contains any kmers within the set adds an additional 35 seconds (+33%). To attempt optimising this process, both a bloom filter, and minimization approach was attempted before the `scan_read_pairs` function.

3.3.11.1 Bloom Filter

Bloom filters are probabilistic data structures used to assess if a value is part of a set. It allows for false positives but not false negatives (Bloom 1970). In this way, an if statement that returns true to value being in a bloom filter can be passed to a check if that value is in a hash set. If the check is more efficient for a bloom filter, a speed benefit would be seen. This was not the case however, as using a bloom filter was 5.6 times slower than just checking the hash set alone, with a 13 minute run time for the test file. It's worth noting, the python version from the module pybloomfilter was used, and not interacting with the cython bloom filter objects directly which is expected to be faster. However, the efficiency benefits from doing so were not expected to be faster than just checking the hash set, so this option was not considered further.

3.3.11.2 Minimization

A sliding window minimum function was written in cython and used before the scan read pair function (checks hashed kmers in hash set). A window size of 12, and a kmer size of 6 was used, then if a count of more than 15 minimizers was returned, the read pair would be passed to scan read pairs. The minimization approach added an extra 1 minute 10 seconds (1.5x slower) compute time on top of the base run speed.

3.3.11.3 Modified reference

A cut-out of the hg38 reference at the telomeric regions was generated. This was done so that

the entire reference could be included within the package, and a set of locations to analyse could be pre-determined without needing the user to specify them on the command line. This cut-out was also modified by consolidating multiple near-identical TTAGGG and CCCTAA repeat regions into one region. Compacting these multiple regions into one contig improved the performance of the model, possibly through reducing the noise created by the original alignment. A cut-out from the telomere to telomere reference was also tested, but unfortunately performed much worse with an F1 score of 0.105. Decrease in F1 score was likely due to the T2T reference containing generic non-telomeric repeats such as long pure C repeats. The cut-out version of the reference also allowed for another optimisation step using mappy to filter reads being written to the fastq file (code 1 line 22).

3.3.11.4 Mappy alignment

Writing all the reads that contained one of the kmers length 32 would result in a large number of reads in the resulting fastq files that were not aligned (e.g. 300,586 for DB143). This significantly inflated the size of the intermediate files from between 17MB to 83MB each. File sizes would increase for samples with higher coverage. A further step was added, where after a read had been identified to contain a relevant kmer, the read would be checked using the mappy modules map command to see if it could be aligned to the modified reference. The speed of mapping was comparable to that of writing the read to a file so speed was not impacted. This step significantly reduced the size of the intermediary fastq files, with DB143 being reduced from 37MB down to 2MB.

3.3.12 Model performance

The kmer-based method for classification slightly outperformed the region-based method. The difference between the kmer and region method is that the kmer region predicted 5 more

true positives with 3 more false negatives (figure 3.9). The F1 score and recall is slightly higher (0.13 and 0.27 respectively), with slightly lower values for specificity and precision (~0.05 each) (table 3.4). This performance increase likely comes from the consolidation of multiple reads into a single feature making it easier and less noisy than when they were spread across multiple in the region-based method.

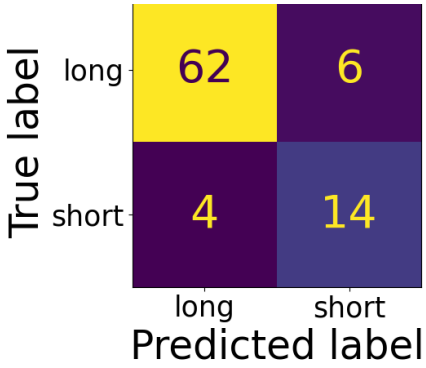


Figure 3.9: Confusion matrix for kmer-based classification method

	F1 score	Specificity	Precision	Recall
teltool (R)	0.600	0.956	0.750	0.500
teltool (K)	0.736	0.911	0.700	0.778

Table 3.4: F1 score, specificity, precision, and recall for teltool in region (R) and kmer (K) modes

3.3.13 Sequencing technologies

One unknown aspect of remapping reads to a custom reference method was whether it would be sequencing technology agnostic. Initial analysis from the International Cancer Genome Consortium (ICGC) which was sequenced using Illumina HiSeq (details on how this data was collected provided in chapters 1 and 5), and colorectal polyps sequenced with a different BGI protocol (DNBSEQ PE150 compared to BGISEQ500 from the training set) showed unexpected differences between the technologies (figure 3.10). These differences are notable as from the model's perspective, the predictions will not be accurate when given values it has

previously not encountered.

For GC percentage, the general trend was Illumina sequencing exhibits a higher value than BGI for the majority of regions (figure 3.10a and appendix 2). Mapping quality shows more overlap with BGI data than GC percentage, however the distributions are skewed to either the higher or lower end on average (figure 3.10b). Both datasets were aligned with bwa mem (illumina v0.6.2, BGI v0.7.17-r1188). The soft clip quality correlation displays much tighter (narrower range) in its distribution for the Illumina sequencing over BGI (figure 3.10c). The "fragments" variable has a similar issue to mapping quality where some overlap is present, but the distribution is shifted either higher or lower depending on the region (figure 3.10d). Similar behaviour is also present in the "kc percentage" (percentage of telmer sequence in reads) variable except with some narrower distributions also found in some regions (figure 3.10e). Coverage also showed little overlap between the sequencing technologies (figure 3.10f). Comparative analysis has previously been performed between BGI and Illumina sequencing platforms, however they focused on their ability to detect single nucleotide polymorphisms (SNPs), gene quantification, and bacterial genome assembly, and do not report on the differences in coverage of regions with similar GC% content (Zhu et al. 2018; Senabouth et al. 2020; Hu et al. 2024).

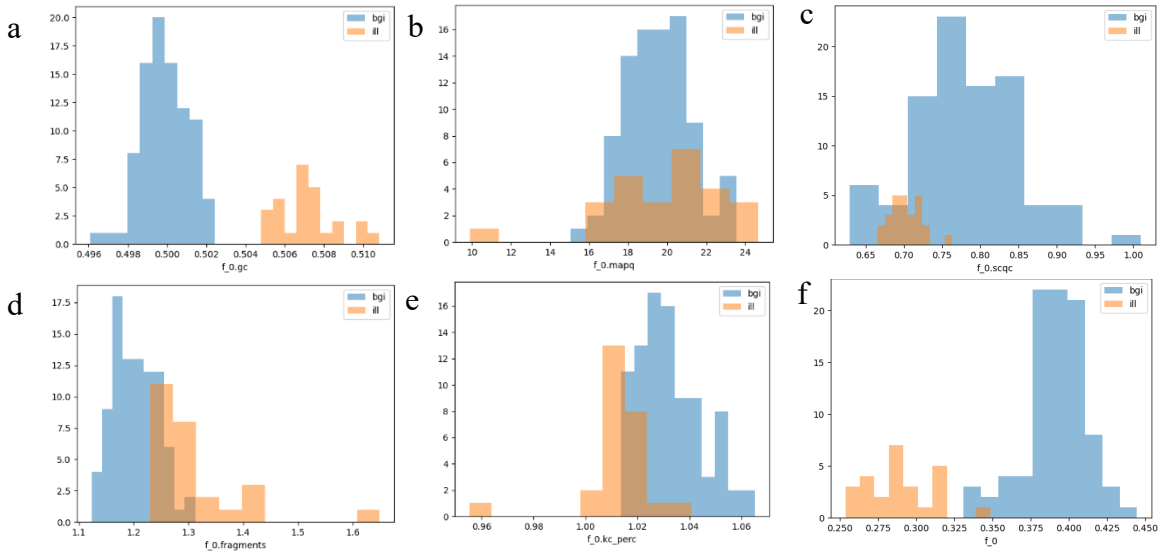


Figure 3.10: Metrics measured by teltool for both BGISEQ500 (blue) and Illumina HiSeq (orange) samples for the forward region. GC% (a), mapping quality (b), soft clip quality correlation (c), fragments (d), kc percentage (e), coverage (f).

3.3.13.1 Coverage

Alongside the sequencing technology used for samples, the coverage at which they were sequenced was the other main difference between the training and test sets. The initial testing also provided insight that the basic normalisation that was being used was insufficient (as shown in Figure 3.10f with little overlap between the technologies). Various methods for transforming the values were attempted (figure 3.11).

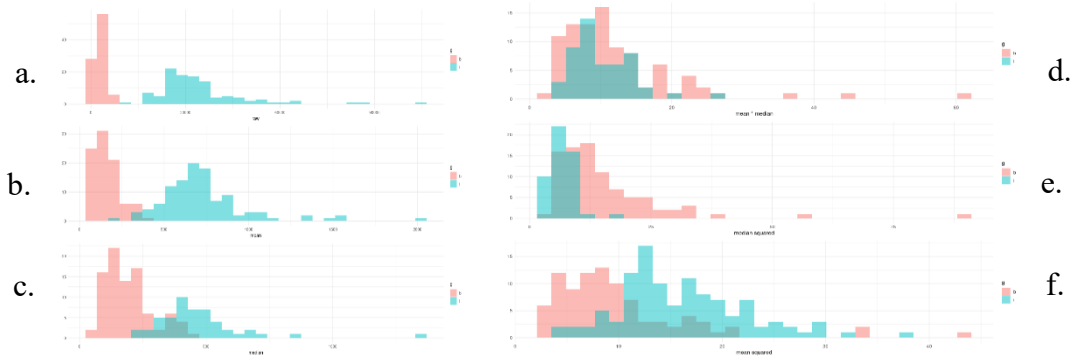


Figure 3.11: Raw coverage (a), coverage normalized by mean (b), median (c), mean times median (d), mean squared (e), median squared (f).

3.3.14 Updated approach

Without a robust and reliable coverage normalisation method, a new approach for the coverage variable was implemented. Instead of using a transformation of the coverage, the coverage was summed, and a percentage for each region was generated. Due to the dominant number of reads within the "forward" and "reverse" regions, the remaining regions were pooled together to create the "other" (or "rest") region. A model was trained using only these three values. This model was then tested on the training set, and the three values were plotted on a graph with the outcome labelled in colour to understand if this was a viable method (figure 3.12).

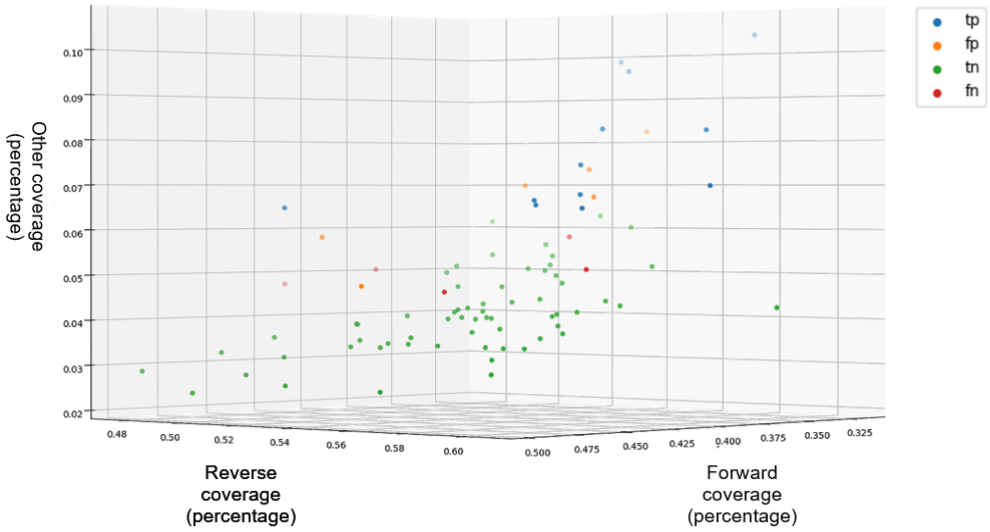


Figure 3.12: 3D graph showing the percentage coverage for the forward, reverse, and "other" regions, with true positive (tp) (blue), false positive (fp) (orange), true negative (tn) (green), and false negative (fn) (red) labels.

The graph shows a surprising trend, generally the shorter telomeres showed a higher percentage coverage reads in the "other" regions. Despite using less variables compared to the previous iteration of the model, the performance is similar with only a 5%, ~3%, and ~7% reduction in F1 score, precision, and recall respectively (table 3.5).

	F1 score	**Specificity**	**Precision**	**Recall**
Previous	0.736	0.911	0.700	0.778
Updated	0.686	0.912	0.667	0.706

Table 3.5: Table showing the model performance metrics of the percentage coverage (updated) model, compared to the previous iteration.

Despite the model performing slightly worse, due to the simplicity (containing only three variables) it is arguably better for the application of testing new data as it might generalise better to new data. However, there was still an issue with the values from samples outside the training set not conforming to the limits of the training data (figure 3.13). This graph shows there is a difference between the strand biases of the sequencing technologies.

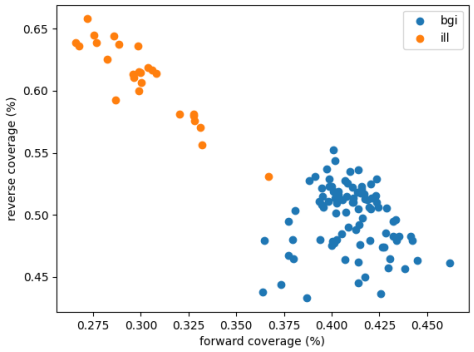


Figure 3.13: Values for both BGISEQ500 (blue) and Illumina HiSeq (orange) samples for the percentage coverage of the forward region, plotted against the percentage coverage of reverse region.

3.3.14.1 Strand bias correction

A simple strand bias correction was introduced to try account for the difference between sequencing technology values in coverage. For reads mapped to the forward and reverse "pure telomeric repeat" regions, a count was kept for the number of reads with and without the "`is_reverse`" flag. The coverage value for the forward region was then divided by $\frac{n_f}{n_f+n_r}$ and the reverse region was divided by $\frac{n_r}{n_f+n_r}$ where n is the number of reads in the forward region with subscript f and r denoting false and true for the value of the "`is_reverse`" flag. This improved the distribution of the data (from the model's perspective) with some overlap in the values for the training and test set (figure 3.14).

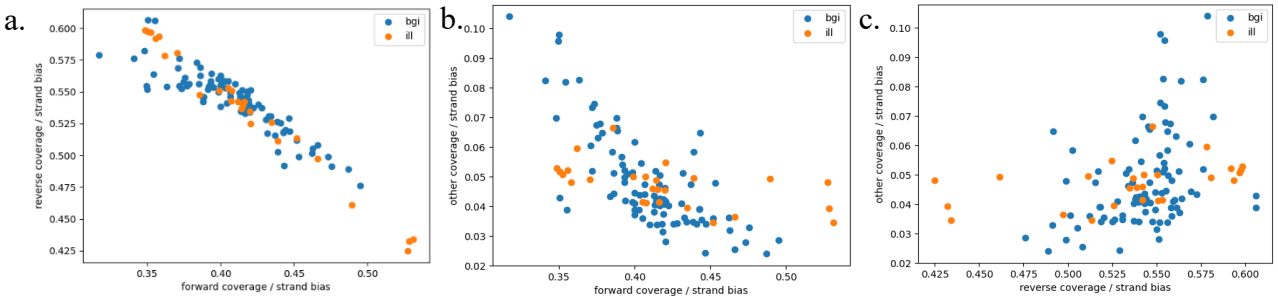


Figure 3.14: Percentage coverage for both the BGISEQ500 (blue) and Illumina HiSeq (orange) samples normalised by the strand bias metric for the forward against reverse (a), forward against other (b), and reverse against other (c).

Other methods for strand bias correction were also attempted such as using the strand bias scores from various publications. For the following equations: *a* denotes the number of reads on the forward strand flagged forward, *b* forward strand flagged reverse, *c* reverse strand flagged forward, *d* reverse strand flagged reverse. A method used in a mitochondria heteroplasmy study used the formula (Guo *et al.* 2012):

$$\left|\frac{b}{a+b}-\frac{d}{c+d}\right| / \left(\frac{b+d}{a+b+c+d}\right) \quad (1)$$

The Genome Analysis Toolkit (GATK) method of using (McKenna *et al.* 2010):

$$Max\left[\frac{\frac{b}{a+b}*\frac{c}{c+d}}{\frac{a+c}{a+b+c+d}}, \frac{\frac{d}{c+d}*\frac{a}{a+b}}{\frac{a+c}{a+b+c+d}}\right] \quad (2)$$

Using the 1-p-value of the fisher score of the 2 by 2 table (${}_{c}{}^{a}\,{}_{d}{}^{b}$) was also tried (Guo *et al.* 2012). As these methods are originally supposed to provide a score for the strand bias instead of a correction, various modifications were made in an attempt to create more suitable correction values. The most successful (resulting in most overlap between sequencing groups) of these variations for the mitochondria method was multiplying the coverage for the reverse region by the equation (1) above. The forward region was multiplied by the equation by substituting the numerators *b* and *d* for *a* and *c* values respectively (figure 3.15a). For the GATK metric, the forward region was divided by the

maximum value in the array, and the reverse the minimum (figure 3.15b). Both methods increase the overlap between the datasets for these variables, with the first causing a stricter correlative relationship between the two and more overlap, and the second closing the previously present gap.

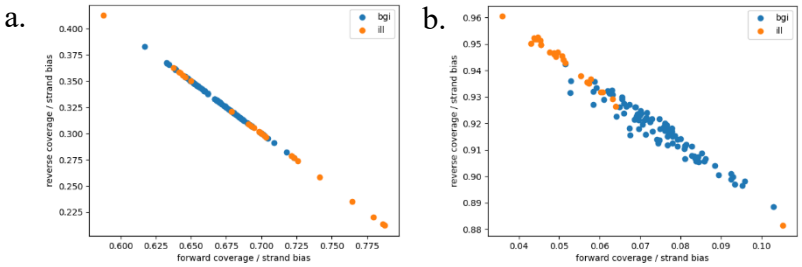


Figure 3.15: Percentage coverage of forward plotted against reverse region for both BGISEQ500 (blue) and Illumina HiSeq (orange) when normalised by modifications of strand bias scores from Guo et al 2012 mitochondria heteroplasmy study (a) and the Genomic Analysis Toolkit (b).

These were however less successful than the simple method presented above (figure 3.14). The fisher score was also not a viable method as the p-values were almost always 1, resulting in a division by 0 error. There was another method that was subsequently tested in an attempt to solve this problem.

3.3.14.2 Optimal Transport

Optimal transport uses geodesics to find the most efficient (shortest) path for transforming data with one distribution to another (Peyré and Cuturi 2019). One of its many uses is that of domain adaptation, a process of aligning a source domain to a target domain. This is extremely useful in the case of machine learning, as models can be trained on one specific dataset, and if the input represents the same class of information, a model should be able to make a reasonable prediction (reference).

The python optimal transport (POT) library was used to test if the strand bias corrected Illumina data could be transformed to look more like the BGI training set (Flamary *et al.* 2021). This yielded results that were considered more helpful (from the models perspective, as training and test data values are overlapping) than the previous methods using just the strand bias correction/normalisation alone (figure 3.16).

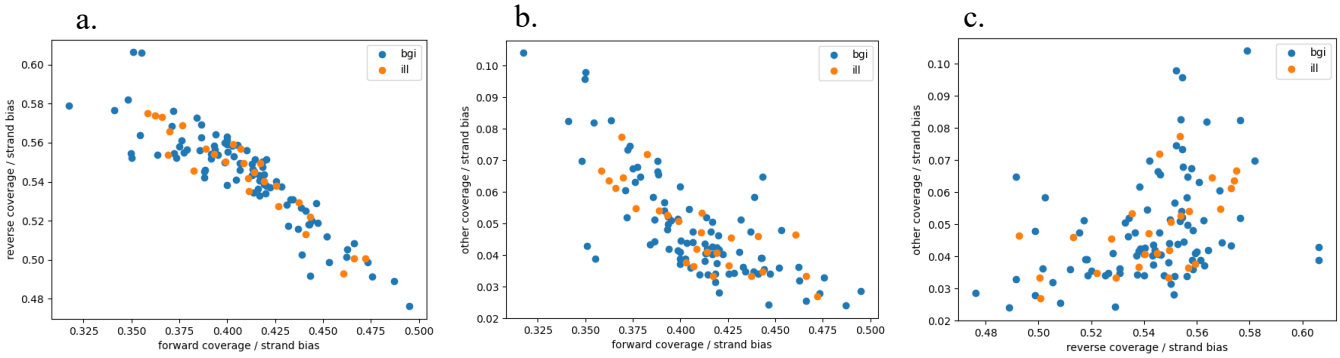


Figure 3.16: Percentage coverage for both the BGISEQ500 (blue) and Illumina HiSeq (orange) normalised by strand bias, with transformation using optimal transport used on the Illumina samples for the forward against reverse (a), forward against other (b), and reverse against other (c).

Whilst optimal transport could be used on the variables (such as GC% etc), relying on the transformations of many more variables would add some complexity and doubt to the validity of results for untested data. Therefore it was important to validate this method using a dataset sequenced with Illumina technology with accompanying STELA determined telomere length.

3.3.15 Validation from Genomics England Dataset

A second dataset of Chronic Lymphocytic Leukemia (CLL) cancer samples with matched telomere lengths from STELA was analysed with teltool inside the Genomics England research environment. These patients were selected for sequencing due to their short telomere lengths and were sequenced with mainly Illumina HiSeq X with a couple with HiSeq 4000. The raw values from the length, coverage, and telmer length metrics show a considerable

difference between the BGI and Illumina datasets due to Illumina being sequenced at a higher genomic coverage (~30-80) compared to BGI (~15-20). Therefore, these variables were normalised to their respective sum across all groups (regions). These raw and normalized variables were then plotted against telomere length for each dataset to determine whether they would be appropriate for use in a predictive model. The ICGC dataset was also plotted in these graphs with all telomere lengths set to 4kb to display the general distribution and provide a visualisation of overlap between each dataset (figure 3.17).

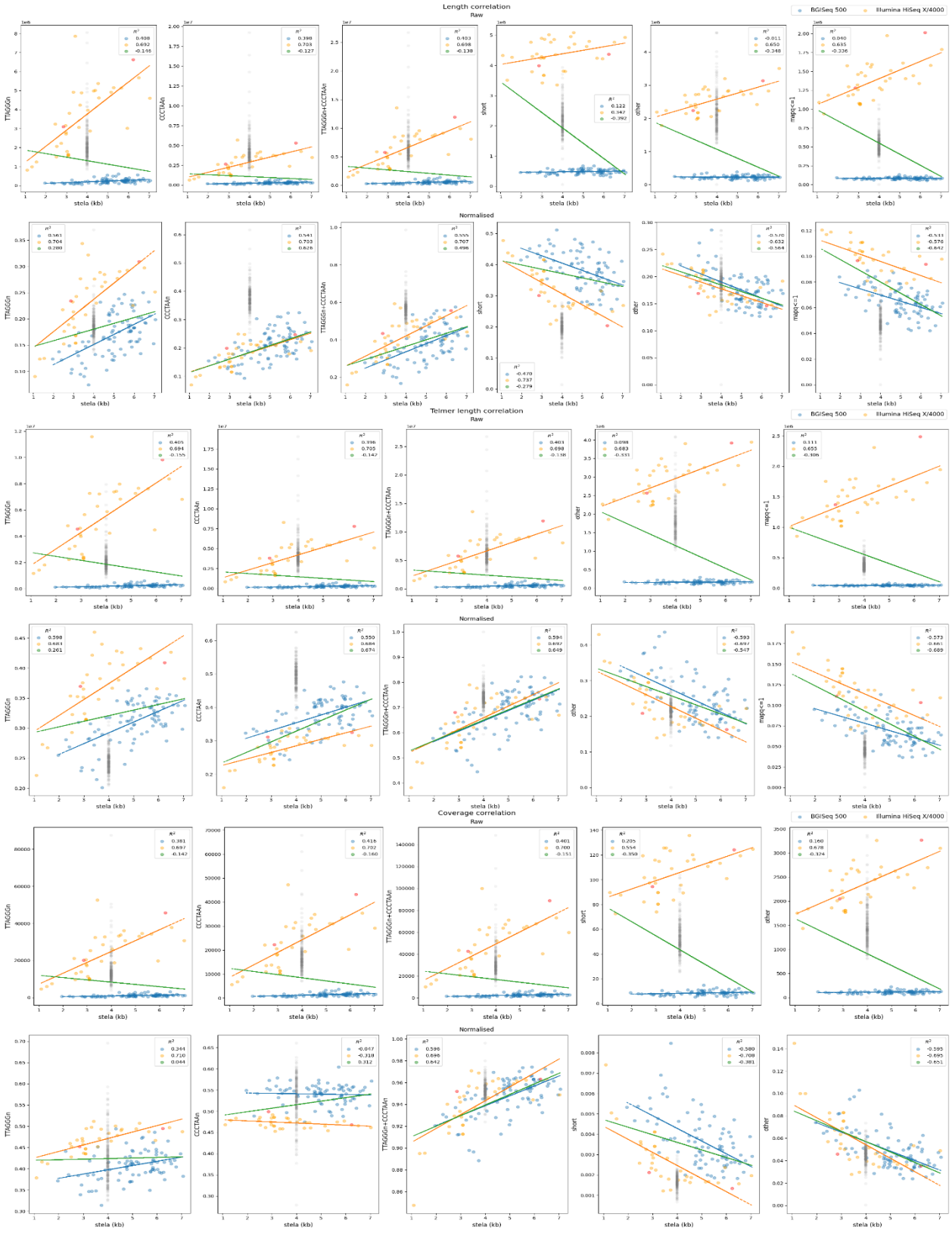


Figure 3.17. Raw and normalised values for the length (a), telmer length (b), and coverage (c) variables across each teltool region (TTAGGGn, CCCTAAn, TTAGGGn+CCCTAAn, short, other, mapq<=1). Each graphs shows the distribution for the BGISeq breast cancer dataset (blue), CLL dataset (HiSeq X orange, HiSeq 4000 red), and ICGC breast cancer dataset (Illumina HiSeq) with all stela values set to 4kb (grey), with lines of best fit and R^2 for datasets with stela values, and BGISeq breast cancer combined with CLL data (green).

The criteria for the most suitable machine learning variables are overlapping distributions between all datasets, uniformity between best fit lines between each and combined sequencing technologies, and similar R^2 values for each line. For this reason, normalised: length of "other", telmer length and coverage of "TTAGGGn+CCCTAAn" regions (figure 3.17) were selected for training of a new linear regression telomere prediction model (figure 3.18).

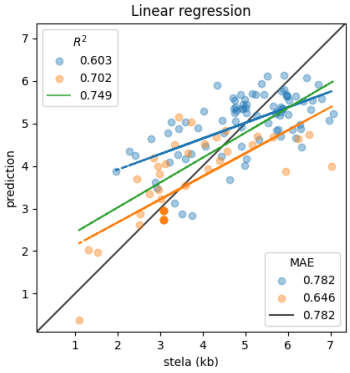


Figure 3.18. Teltool linear regression model using the features common between both datasets with telomere lengths, BGISeq breast cancer (blue), HiSeq CLL (orange).

This model outperformed all other previous telomere prediction tools (figure 3.19). Some data for the Illumina tests are missing due to issues with running the software in the environment the data was stored. TelomereHunter is also missing for BGISeq data as there were errors which despite attempts to patch could not be resolved.

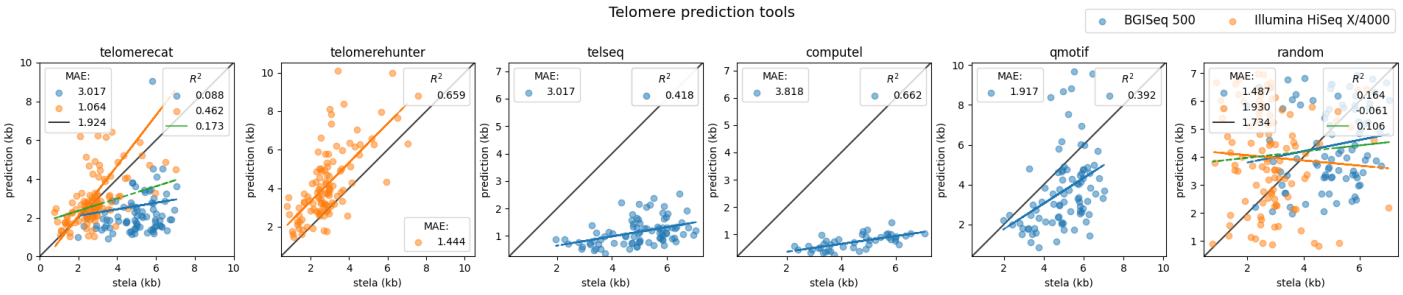


Figure 3.19. Predictions for telomerecat, telomerehunter, telseq, computel, qmotif, and random assignment for BGISeq 500 (blue) and Illimina HiSeq X (orange), with mean absolute errors (MAE) displayed in the legend for each tool.

A surprising result of comparing the results from previous tools shows how accuracy differs between the different sequencing technologies. For example, telomerecat tends to overestimate telomere lengths for Illumina HiSeq samples, but underestimate when analysing BGISeq samples. Whilst the R^2 of randomly assigning telomere lengths to samples uniformly between the datasets respective minimum and maximum values is poor, the mean absolute error is lower than most of the predictive tools. This indicates that most telomere prediction tools are best used for ranking a cohort by telomere length, rather than prediction of an absolute telomere length. This highlights the importance of taking into consideration a combination of metrics to evaluate the estimated telomere length values.

3.4 Conclusion and future work

In this chapter we explored and benchmarked various methods for predicting telomere length from whole genome sequence (WGS) data. Two machine learning based methods developed here outperform previous software for absolute telomere length prediction in the case of regression, and segregation of samples by telomere length thresholding by classification, whilst in most instances being faster to run. In the case of regression, the mean absolute error (MAE) for teltool was between 782bp and 646bp depending on the sequencing technology. This is an improvement on the next best telomere length predictors telomerecat which had a MAE of 1064bp and telomerehunter with a MAE of 1444bp both exclusively using Illumina input. This improvement in MAE was accompanied by an increase in R^2 with values of 0.6 (BGI) and 0.7 (Illumina) higher than any other predictor. Most notably the performance increases seen for classification were shown in both the F1 score (~0.67 from ~0.49 next best qmotif) and precision increase by two-fold (~0.67 from ~0.33 next best qmotif).

During the development of this software, we observed differences in the characteristics of data from BGI and Illumina sequencing data. This extends to the

performance of previously released software as shown in Figure 3.19 where telomerecat performs noticeably differently depending on the type of sequencing used. This likely extends to the other software, which unfortunately was unable to be tested. Whilst previous analyses comparing the two technologies found no significant differences in use cases of SNP detection, gene quantification, and bacterial genome assembly; the results here indicate that there are underlying characteristics that should be considered when developing and employing software that uses certain metrics in their algorithms (Zhu et al. 2018; Senabouth et al. 2020; Hu et al. 2024). It is suspected that variance in for example the strand bias shown to occur between these two sequencing platforms is a result of the type of polymerases used.

There is a possibility with further development the methods presented here, results could be improved and increase the scope of what its capable of. There is also potential merit in incorporating a new method for telomere length prediction utilising adjacency graphs which was also considered (see appendix). This new method was initially conceived as it has the potential to classify by ALT and telomerase positives samples. However, this would require a larger training dataset than was available at the time. Adaptations for use with long read sequencing data would also potentially lead to better performance in this area. As it stands though, the software teltool developed in this chapter has also been packaged into a python module that can be installed in one line (`pip install setup.py`). This method will be used in chapter 5 to investigate breast cancer samples from the International Cancer Genome Consortium, to predict the order of shortest to longest telomeres in the cohort so the correlation between telomere length and genomic complexity can be explored.

Chapter 4 Telomere length and genome complexity

4.1 Abstract

Previous research has shown links between telomere length and genomic complexity. Genomes with shorter telomeres have been suggested to be more prone to phenomena such as chromothripsis and other processes that generate genomic complexity. In this chapter, methods are developed for analysing and characterising genomic complexity using WGS data. The correlation between telomere length and genomic complexity through various analyses of structural variants, relative copy number changes, and assembled contigs are presented.

4.1.1 Data

The whole genome sequencing datasets used in this chapter consists of 44 tumour-normal pairs from a breast cancer cohort. All samples were sequenced at a depth of around 15-fold coverage with BGISeq 500 sequencing, read length 100, aligned to human reference hg38, and include accompanying STELA data.

4.2 Introduction

As discussed in chapter 1, telomeres are structures located at the chromosome ends comprised of hexameric TTAGGGn repeats and shelterin complex which together function to

protect the end of the chromosome from being mistaken as a double strand break (Lange 2005). With every cell division, telomeres shorten due to semiconservative replication and the end replication problem. Eventually shortening will lead to the DNA damage response arresting the cell cycle in a stage known as senescence (Rossiello et al. 2022). In the absence of tumour suppressive pathways such as p53 and Rb, cells can bypass senescence and enter crisis (Jacobs and de Lange 2004). Cells in telomere crisis are typified by large-scale genomic instability, caused by telomere fusions (Greenberg 2005).

4.2.1 Telomere fusions

Telomere fusions occur as a backup protection mechanism to protect telomere ends that have lost their function against excessive degradation (Stroik and Hendrickson 2020). Normally shelterin is responsible for protecting the telomeric ends and prevents telomere fusions. However, over or under-expression of shelterin can lead to increased fusion events (Lisaingo et al. 2014; de Lange 2018). Shortened telomeres can influence the mutations and occupancy of shelterin (Mir et al. 2020). Consequently, cells with shorter telomeres have a higher chance of forming telomere fusions leading to the formation of dicentric chromosomes, and subsequently exhibit an increase in genomic instability from fragmentation (chromothripsis) (Cleal et al. 2019; Dewhurst 2020).

Fragmentation occurs in some cases due to dicentric chromosomes being shattered by mechanical force exerted during mitosis (Guérin et al. 2019). More frequently though, dicentric chromosomes are instead resolved via chromatin bridges that form between daughter cells that persist into the next G1 phase. Chromatin bridges can be resolved in simple breaks which form fold-back inversions or large terminal deletions, or complex breaks leading to chromothripsis (Maciejowski et al. 2015). It is possible as these bridges contain nuclear envelopes connected to both cells, their resolution can lead to the envelope

rupturing causing the mixing of nucleic and cytoplasmic contents (Hatch 2018). Three prime repair exonuclease 1 (TREX1) which is abundant within the cytoplasm could then resolve bridges by degrading the DNA into single strands along the bridge (Jiang and Chan 2024).

TREX1 is not the only mechanism involved in chromatin bridge resolution. Polymorphisms in non-coding regions of Ankle1 have been associated with increased susceptibility to breast and ovarian cancer, affecting both the general population and individuals carrying the BRCA1 mutation. Research on the nematode *C. elegans* has shown that its LEM-3 orthologue localizes to the midbody cleavage plane, facilitating the processing of chromatin bridges (Chan and West 2018). Additionally, LEM-3 cooperates with the *C. elegans* homologue of BRCA1 to uphold genomic stability (Hong et al. 2018). Further study is required to fully understand the implications and mechanisms in human cancer. Fragmented chromosomes and damaged chromosome-ends generated from complex breaks, or the resolution of chromatin bridges are prone to various DNA repair mechanisms, potentially resulting in complex rearrangements seen in chromothripsis (and kataegis) (Maciejowski et al. 2015). It is also possible this process can result on other complex genomic patters such as chromoplexy (via NHEJ) which is similar to chromothripsis but involves more chromosomes, and chromoanasynthesis where chromosomes are duplicated or triplicated when MMBIR and FoSTeS are involved (Pellestor 2019).

4.2.2 Breakage-fusion-bridges (BFB) cycles

Another consequence of dicentric chromosome formation is breakage-fusion-bridge cycles. Breakage of the dicentric chromosomes creates more exposed ends which can produce further fusions within the daughter cells (McClintock 1938; Gisselsson et al. 2000). This cycle is frequently broken when a telomere from another chromosome is translocated to the affected dicentric chromosome. During these cycles, genomic instability can be generated

and transferred between chromosomes (Bailey and Murnane 2006). This instability can lead to: gene amplification, loss of heterozygosity (LOH), and non-reciprocal translocations (Maciejowski and de Lange 2017). Gene dosage variations are a important driver in cancer, especially amplifications which can activate dominantly acting cancer genes (Stratton et al. 2009).

Gene amplification occurs when fused sister chromatids break asymmetrically, and the broken chromosomes are then replicated during S phase and persist into G2 phase. The four telomere deficient ends can then either fuse symmetrically and form repaired chromosomes, or asymmetrically to create large palindromes (Kinsella and Bafna 2012). These palindromes can be encompassed in both dicentric and centromere lacking chromosomes. Deletions or duplications can occur from dicentric chromosomes through breakage during chromosome segregation, leading to daughter cells inheriting broken chromosomes with copy number alterations. Broken DNA ends can also participate in further BFB cycles leading to palindromic gene amplification (Tanaka and Yao 2009).

Conversely loss of heterozygosity can occur when a daughter cell inherits a terminal deletion post dicentric chromosome breakage and is common at cancer related loci (Maciejowski and de Lange 2017). Following a chromosome break, DNA from the broken end may invade another chromosome, leading to non-reciprocal translocations facilitated by break-induced replication, primarily through homologous recombination (Anand et al. 2013; Malkova and Ira 2013). Chromosomes without telomere caps can fuse with various internal genomic loci, even in an otherwise stable genome (Cleal et al. 2018). Cells lacking p53-mediated mitotic checkpoint control are particularly more prone to these telomere fusions. Additionally, the frequency of fusion events is also heavily influenced by the cellular proficiency in classical (C-) and alternative (A-) non-homologous end joining (C-/A-NHEJ), the latter also known as theta-mediated end joining (TMEJ). Intra-chromosomal joining, in

contrast to inter-chromosomal joining, exhibits less dependence on LIG4 and instead involves both LIG4 and LIG3 (Liddiard et al. 2016).

4.2.3 Micronuclei pathway

Another origin of chromothripsis is via the formation of micronuclei which can arise from the erroneous segregation of dicentric chromosomes or from other segregation errors via so called lagging chromosomes (Zhang et al. 2015). Micronuclei are thought to be defective in various ways, and entrapped DNA is thought to be liable to undergo replication stress followed by fragmentation and re-joining by non-homologous end joining, resulting in chromothripsis (Hatch and Hetzer 2015; Ye et al. 2019). Aberrant nuclear envelope assembly can also lead to defective DNA replication and loss of micronuclei envelope integrity leading to large DNA damage via unknown mechanisms (Liu et al. 2018). Live-cell imaging has shown that though micronuclei do not form immediately after bridge breakage, rather the frequency of micronuclei formation after the next mitosis was observed in between 52% and 65% in cells with bridges. Cells without chromosome bridge division within the same imaging dish treated identically did not producing micronuclei (Umbreit et al. 2020). Additionally, depletion of TRF2 using siRNA and mitotic checkpoint barriers (Mps1 inhibited with reversine and hesperadine) combined can also cause chromothripsis. The complete mechanism and pathway involved is not fully understood (Mardin et al. 2015; Cleal et al. 2019).

Previous research has implicated telomere shortening in the process of generating genomic complexity (Rampazzo et al. 2010). Cells that are undergoing telomere crisis are subject to large genomic rearrangements (Dewhurst 2020). In this chapter, results from a local cohort of breast cancer samples was used in the development of methods for interrogating the relationship between genomic complexity and telomere length. Using

WGS data, copy number profiles and different facets of structural variants are explored. We expected to see a trend from previous analyses which examine this same cohort, where there is a hard threshold value that can be used to divide the cohort into distinct complexity profiles (Dos Santos et al. 2015).

4.3 Methods

4.3.1 Structural variant calling and filtering

The first stage in investigating the genomic complexity of the dataset was to create a set of unique somatic structural variants (SVs) using dysgu. Dysgu is a suite of tools including an SV caller, merger, and SV filter. Whilst SV calling can be performed in one command (`dysgu run input bam > output.vcf`) this analysis used the "fetch" and "call" method to generate coverage bed files using the fetch output (Cleal and Baird 2022).

```
for i in bams/*.bam; do
   dysgu fetch /tmp/${s} ${i}
   dysgu/coverage2bed.py -out-bin-size 10000 -w /tmp/${s} >
${coverage_dir}/${s}_cov.bed
   dysgu call --ibam ${i} ${reference} /tmp/${s} -o vcfs/${s}.vcf
   rm -rf /tmp/${s}
done
```

The output vcf files were then filtered against the germline using the dysgu filtering method:

```
while IFS=, read -r t n l; do
      dysgu filter -o /home/out/${t} --normal-vcf
/home/vcfs/merged_normal.vcf /home/vcf/${t}.vcf /home/bam/${n}.bam
done < sample_pairs.csv
```

Further filtering was achieved by investigating the PROB variable in the genotype field. This refers to the machine learning confidence value given to the called SV by the dysgu model. Manual investigation of the SVs was done using bcftools and gw, a terminal based genome browser (Danecek et al. 2021; Cleal et al. 2024).

```
bcftools filter -i'INFO/SVTYPE == "DEL" && FORMAT/PROB >= 0.4 &&
FORMAT/PROB <= 1.0' ${vcfdir}/${b}.vcf | gw --link sv --labels yes,no,maybe
--out-labels ${i}_del.tsv ${refpath} -b ${bamdir}/${b}.bam -v - --track
${i}
```

After establishing a suitable filtering methodology and filtering threshold, the precision and recall were then calculated by separating variants above and below the threshold. A random selection of 50 variants from above and below the threshold were used to estimate performance metrics including true/false positive/negative rates.

4.3.2 Copy Number Analysis

The first pipeline to be run was the copy number analysis as a prerequisite for downstream analysis pipelines.

```
python3 copy_number_pipeline.py -i raw_cov -r hg38 -w 10000 –subsets 1 -
threshold 3.81 --bg sample_pairs.csv tumour length normal
```

The -i specifies the input directory, -r is the genome type, -w specifies a window size of 10000 base pairs, --threshold is the telomere length to split the groups by, and –-bg is the telomere length file with the tumour, telomere length, and normal file names. This script takes the raw coverage bed files for both the tumour and normal and normalises the values based on the GC percentage and map-ability of the region, based on the methods used by Cleal et al (Cleal et al. 2019). A matrix for coverage for each combination of GC% between 30 and 60, and mappability values between 60 and 101 is calculated. The median of all these values is then used as the “genomic median” which is then subtracted from the matrix coverage values. This is then interpolated with RBFInterpolator from scipy before using these normalised values to subtract from the raw coverage values (Virtanen et al. 2020). A Haar wavelet transformation is then applied (using the PyWavelets package) to normalised coverage values (Lee et al. 2019). Finally, the normal coverage can be subtracted from the

tumour coverage to create a relative copy number profile displaying the gains and losses of the tumour sample. The tumour values which have now been normalised by their normal counterparts are winsorized (similar to clipped) before performing piecewise constant segmentation (pcf) (Nilsen et al. 2012).

4.3.2.1 Complexity score

Copy number segments identified using the the copy number calling pipeline were used to calculate a "complexity score", which is the sum of the absolute relative copy number of all segments. This method of scoring gives higher values to samples with more segments, whilst minimising the effect of a copy number gain of a chromosome arm, for example. Another more experimental form of analysis is also performed by the script, that uses a low pass filter over relative copy number values. In this case, the score is calculated by gathering all peaks and troughs of the processed signal and measuring the absolute sum of all deviations, where the peak and trough are separated by a relative step in the signal equivalent to >0.5 A simulation of low coverage samples was also achieved by using samtools view -s $fraction to create subsampled bam files that were then used in the copy number pipeline (Li et al. 2009).

4.3.3 Circos Plotting

Following copy number analysis, copy number profiles were used in conjunction with SV calls to produce circos plots.

```
python3 circos.py -l sample_pairs.csv -c stela -r hg38 --cnp
cnp/segmented.copynumber.csv --var-size 10000 --ncols 8
filtered_vcf_directory
```

Pycircos is the python library used to generate the circos plots within this script (github.com/ponnhide/pyCircos). The length for each chromosome is extracted from the vcf header using regex to create a dictionary which was then used to create the "Garc" and

"Gcircle" objects. Optionally at this stage, a cytoband file (-r hg38 alias) can be used to add banding in the outer chromosome arcs and reading of a copy number profile from the previous pipeline to add copy number data around the inner edge of each chromosome (highlighting chromosomes with high scores or number of segments). Pysam is then used to iterate over the variants within the vcf to extract the SV type, loci, length data and excluding variants that do not meet the filtering criteria, which included filtering by probability, size SV length, and number of supporting reads (github.com/pysam-developers/pysam) (Li et al. 2009). Variants that are not classed as insertions are plotted as chord plots, and insertions as a scatter around the edge. The plots are then saved in SVG format and stitched together using svgutils transform commands (github.com/btel/svg_utils). Plots are arranged in order of telomere length, using the (-c) stela columns in (-l) sample_pairs, with (--ncols) 8 columns in the stitched plot.

4.3.4 Chain Linking

The chain linking method is an adaptation on the approach used by ChainFinder algorithm developed by Baca et al, it find SVs which are closer than expected using a binomial distribution and therefore likely related (Baca et al. 2013; Cleal et al. 2019). This step was performed using different variables for each dataset. Generally, the command below was used to gain a rough estimate for the end probability value using the elbow/knee plot, then the search was narrowed from there.

```
python3 chain_link_finder.py -b 20 -l 0.05 -u 0.5 -n 10 -p 2 -t
chain_link/sample_pairs.csv -s 3.81 -e tumor_db -a tumor_stela -c
hg38_cytoBand.txt --size-thresh 10000 vcfs
```

A description of the flags is as follows: -b indicates the min number of breaks, -l lower bound limit for probability, -u upper bound limit for probability, -n number of steps between upper and lower bound, -p is set to 2 to activate the bound search method, -t is sample pairs list, -s

threshold to split groups, -e tumour name column, -a tumour length column, -c cytoband file, a size threshold for minimum SV length, and finally the input vcf directory.

For each chromosome, the frequency of break-sites is calculated across all samples, then for each sample and each chromosome, a K-nearest neighbour tree is made. Then all breakpoints are iterated over checking for links where a binomial probability mass function gives a p-value less than a threshold. If the probability falls below a threshold then an edge is added to a graph. Connected components in the graph are then processed as chains of SVs, only keeping those chains that have a minimum number of breakpoints.

4.3.5 Contig Assembly

Output from the chain linking method is used to scan for discordant reads that are assembled into contigs to analyse the rearrangement patterns in more detail. The contig assembly script simply takes an input bam/cram file glob pattern "*.*am", an input vcf pattern "vcfs/*.vcf", and the output file from the chain link pipeline "all_svs.unique.chains.csv".

```
python3 RRAssembler.py -b '/mnt/breast/*.*am' -v vcfs/*.vcf -c
chain_link_output/all_svs.unique.chains.csv
```

This pipeline runs through several steps: fetching, mapping, analysing, removing, collection then plotting. In summary, fetching involves iteratively collecting regions from input csv or vcf files with some padding (650bp) and collecting discordant reads in those regions (Cleal et al. 2019). The discordant reads are saved to a fasta files for both single and paired reads. These are passed to the spades assembler, before output contigs are aligned using bwa mem (Li 2013; Prjibelski et al. 2020). Realignment of reads to these assembled contigs is performed to filter unmapped reads which are then sorted and indexed.. Coverage arrays (calculated from cigar strings) and the percentage of repetition (calculated with tantan) are then analysed in these new mappings, and contigs with a max coverage more than 45 and

repetition more than 80 percent are filtered out (Frith 2011). The filtered reads are saved to a fasta file with a new name including the length and mean coverage data. Contigs from the dysgu vcf files are then gathered and mapped the same way with a temporary reference as before.

The StripedSmithWaterman function from skbio is then used to map the dysgu contigs to the assembled contigs (Rideout et al. 2024). Dodi (github.com/kcleal/dodi) is then used to analyse the contigs as it chooses an optimal set of alignments from a list. A directed graph is created for each sample where the central nodes are the contigs and peripheral nodes point to break points. Multiple graphs can be merged to find common breakpoints across samples and contigs. These are removed to create a set of unique contigs. The mapping information is collected, and written to a bed file where the primary alignment contains the sequence. This bed file is then plotted as a broken horizontal bar graph with the chromosome indicated by the colour of the bar, and additional information with the proximity to telomere ends is also added.

4.4 Results

Previous findings have reported an inverse association between telomere length and genome complexity (Letsolo et al. 2010; Jones et al. 2014; Dos Santos et al. 2015; Cleal et al. 2019). In this study, the relationship between telomere length and large-scale structural changes in the genome was investigated in detail using a local cohort of 44 breast cancer tumour-normal pairs.

4.4.1 Copy Number

Copy number variants were first identified using the copy number pipeline and visualised in

Figure 4.1, displaying the normalised and denoised relative copy number changes, with the segmentation overlain. Segmentation plots such as this give a relatively high-resolution display of the complexity of a sample by visualizing the gains and losses across the genome. Figure 4.1 shows an example of a relatively stable and simple copy number profile (top), with the copy number usually not deviating far from the 0 value. This shows within the sample there are few gains and losses across the gnome. In contrast, the bottom sample shows numerous fluctuations in the copy number profile showing large amounts of gains and losses sporadically throughout the sample, with notably large fragmentations in 6q, 8, 11, 12, 17 and 20p. Chromosome 8 and 17 in particular, are common sites of CNVs in breast cancer found by Choschzick et al and Hyun et al which will be discussed in more detail later.

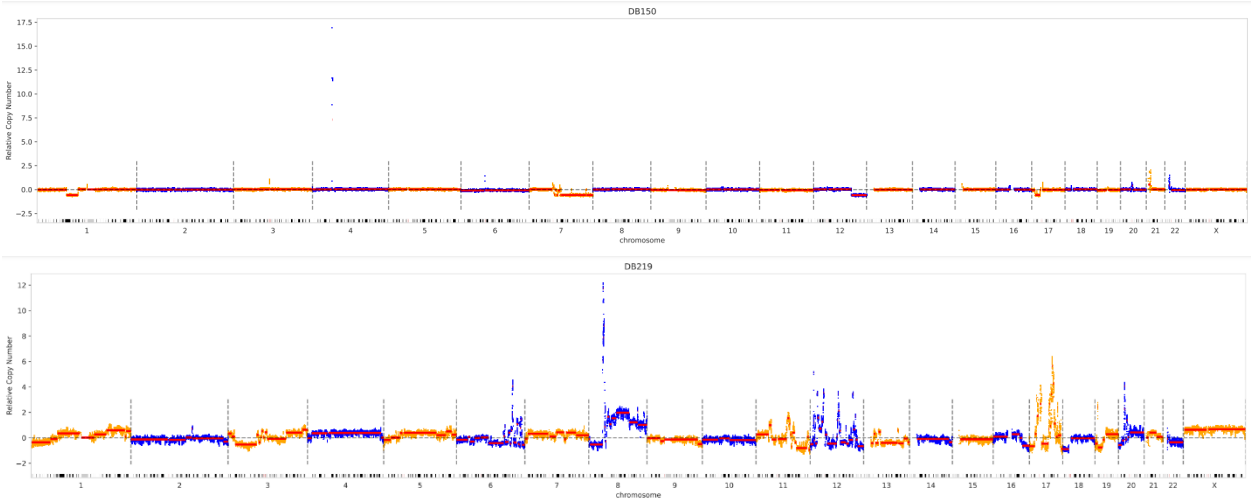


Figure 4.1 Relative copy number after denoising and winsorization for 10kb windows (alternating yellow and blue) with pcf segments overlayed (red). Here are two examples with the top being a relatively stable copy number plot exhibiting little variance (Sample DB150 3.9kb telomere length) and bottom a more complex genome showing large regions with significant variance in relative copy number (sample DB219 2.3kb telomere length).

However, this form of visualization is inefficient when trying to display the complexity throughout the whole cohort. The heatmap shown in Figure 4.2 is an alternative display of the data that can convey equivalent information in a dense format. Relative copy number

values of the segments were clipped at -1.5 and +1.5, so whilst some of the extreme values (such as the short segment at a RCN 11 in chromosome 8) is lost, displaying data using a heatmap allows the genomic complexity of the entire cohort to be visualized in a semi-quantitative manner. Samples are ordered by telomere length with the longest at top, and shortest at the bottom. Visually a distinction can be made between roughly the top two thirds, and the bottom third. Generally, the top of this divide appears “flat” with modest or no gains and losses (indicated by the fainter colours), whilst below the divide, sporadic gains and losses can be seen across many chromosomes. These gains and losses appear to show no bias for any chromosomes outside of common features found in both groups.

Despite the division, there are also some common features seen across both groups. These include gains of 1q, 8q, 16p, 20q, 21q and losses of 8p, 16q. Chromosome 7 shows a general trend to have gains in both arms, with a few exceptions showing losses in the long arm. Chromosome 17 also shows some level of complexity with varying states of copy number across the cohort. Less frequently gains can also be seen across some samples in 5p accompanied by a mix of weak gains and losses in 5q.

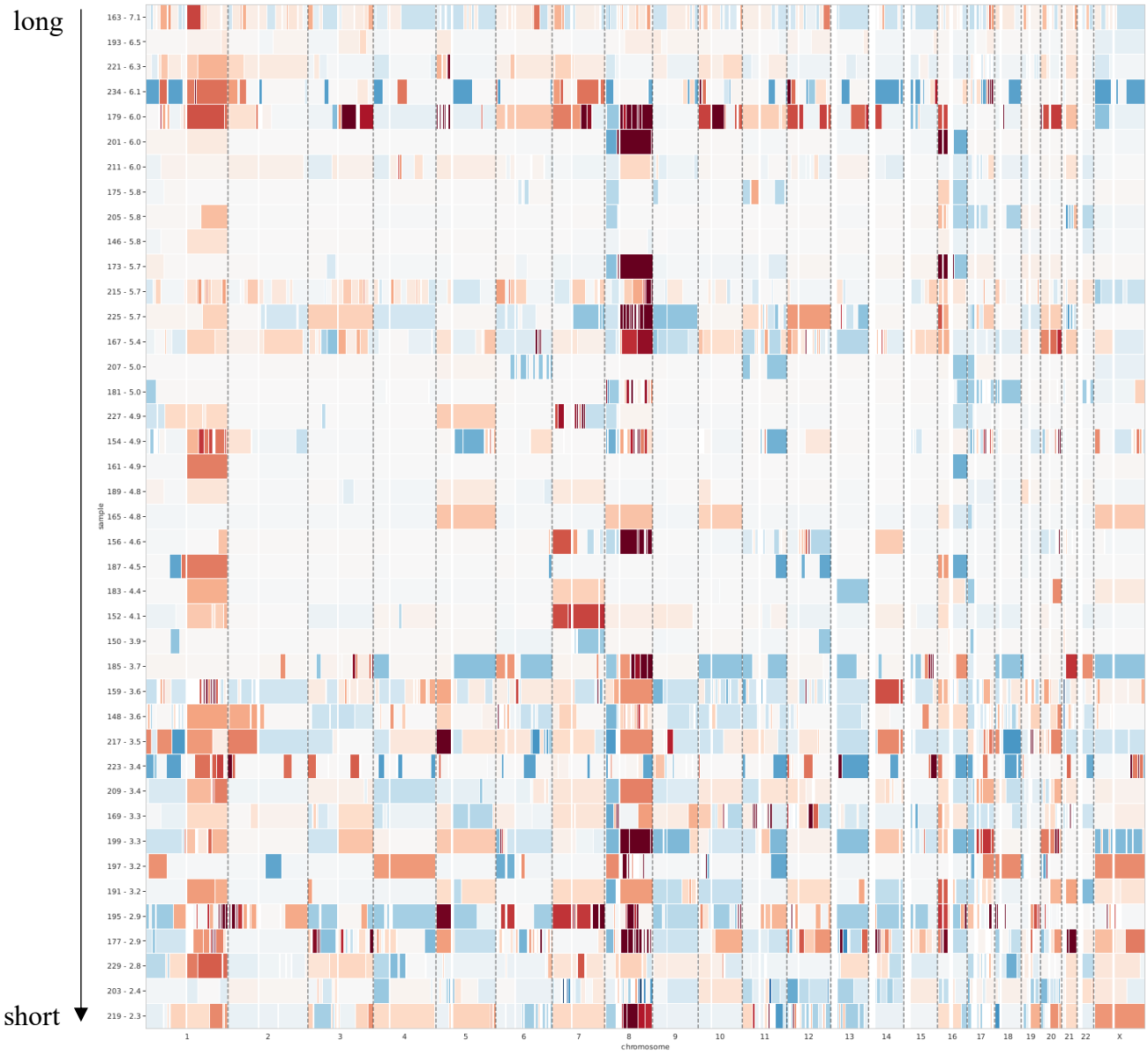


Figure 4.2 Heatmap showing relative copy number segments from pcf analysis (input of 10kb coverage windows) clipped between -1.5 and 1.5 with the Y-axis displaying ordered from longest telomeres (top) to shortest (bottom).

This division in the cohort is confirmed by recursive partitioning, the results of which are shown in the bar graph displaying the number of gains and losses for each sample (n segments that are greater or less than ±0.05) (figure 4.3). The green line shows the optimal split where the p-value is lowest when performing a Mann Whitney U test. Results for this split is shown on the right with significant test p-values for gains ($3x10^{-4}$), losses (10^{-4}), and combined total of both ($2.4x10^{-4}$). The spearman's rank test also indicates there is an inverse

correlation between telomere length and copy number gains and losses which implies samples with longer telomere lengths have fewer gains and losses. The correlation coefficients for gains -0.35, losses -0.37, and total -0.35 all had significant p-values for the test at 0.026, 0.016, and 0.027 respectively. This is in line with expectations and previous findings from Brown et al in leukocytes (Brown et al. 2020).

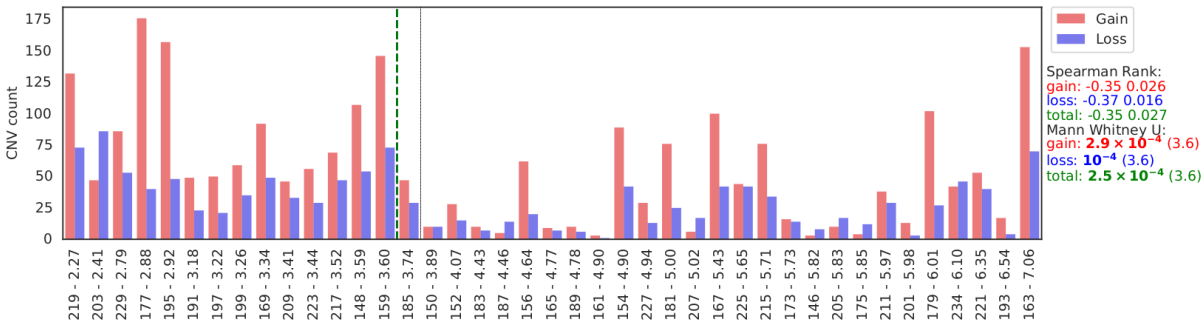


Figure 4.3 Gains and losses sorted by sample telomere length (float after name). The general trend of the data shows higher numbers of gains and losses the lower the telomere length with spearman's rank correlation of: 0.027 for combined gains and losses. The Mann Whitney U (greater than) values are showing the lowest p-value from recursively partitioning from left to right of the gains, losses and totals with telomere length lowest value appears in brackets.

Whilst the heatmap is useful for giving an overall picture of the RCN gains and losses for the cohort, information on gains or losses of shorter chromosomal regions can be harder to read. To combat this, the "squiggle plot" show in Figure 4.4A scales each chromosome to the same size and uses the log of the length for each segment in base-pairs. In contrast to the copy number heat plot (figure 4.2), the squiggle plot avoids issues in displaying amplitude as when the clipping limit (-2, +3) via height is reached, colour values are used to represent the RCN (figure 4.4A). However, the trade-off with this visualisation method, is that when scaling of segments on a log scale, it is not possible to infer loci (x-coordinates for each sample do not linearly refer to the same chromosome position). What is clearer in this plot however, is the presence of oscillations that are indicative of chromothripsis (sample 199 chromosome X, figure 4.4A). The common complexity of gains in chromosome 8 is once again highlighted, but here it is more evident, rather than one large consistent gain of the q arm. Using the squiggle plot, chromosomes frequently displayed a fragmented pattern

involving multiple large gains.

More evidence for the division in copy number profiles between samples above and below TL 3.81kb is shown in the complexity score box plots and recursive partitioning graph (figure 4.4B/C). Generally, the bottom third of samples display a higher median (red line) and mean (green diamond) complexity score. The optimal partitioning also occurs in a similar position to that shown in the number of gains and losses method. Smaller p-values for the Mann Whitney U testing of the complexity score ($1x10^{-40}$ chromosome orange, $9x10^{-4}$ genomic blue) suggests that the gains and losses are larger in amplitude between the two partitions. As the complexity score is a sum of all the absolute values, the two factors influencing it are the number of gains and losses and their amplitude. A similar spearman's rank correlation coefficient of -0.38 (with a p-value of 0.013) is also seen to the number of gains and losses. The low pass version of this analysis is shown in appendix

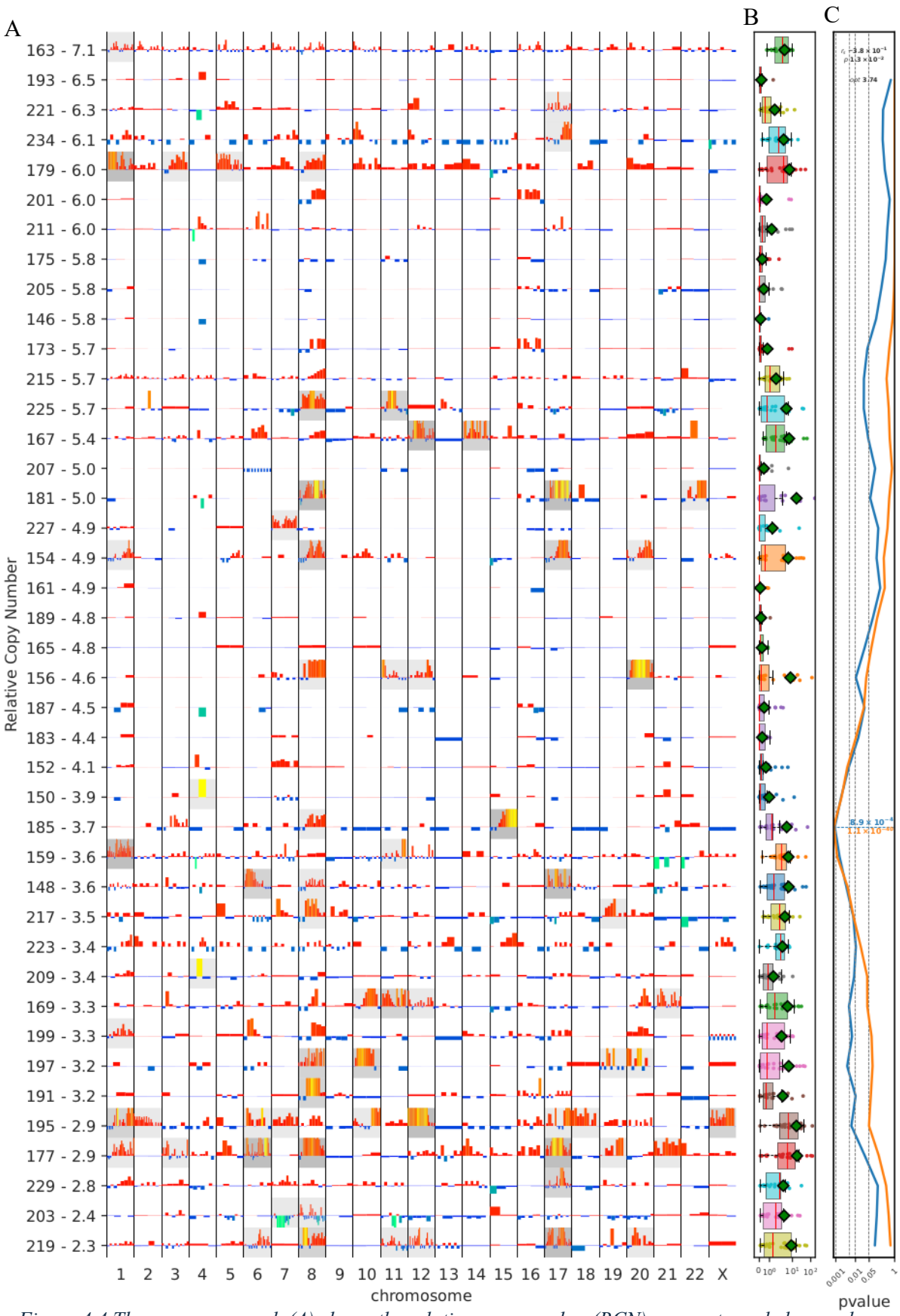


Figure 4.4 The coverage graph (A) shows the relative copy number (RCN) segments scaled so each chromosome is uniformly sized, with the length of each segment on the x-axis being logged. Segnent amplitude is displayed in height between -2 and +3, where colour is used to display higher gains (yellow highest). The complexity score boxplots (B) show the distribution of the complexity scores across all chromosomes for each sample. The recursive partition line graph (C) shows the smallest p-value for the Mann Whitney U test for x > y is $3.6x10^{-4}$ for the sum (genome) and $6x10^{-38}$ individual chromosome array pools.

To visualise the difference between the complexity scores of the cohort split by the optimal partition, the complexity scores (on a logarithmic scale) for both the PCF and low pass methods was visualised (figure 4.5). Both methods displayed a significantly higher complexity score in the shorter telomere group across individual chromosomes (Figure 5A/C) and the genome as a whole (figure 4.5B/D).

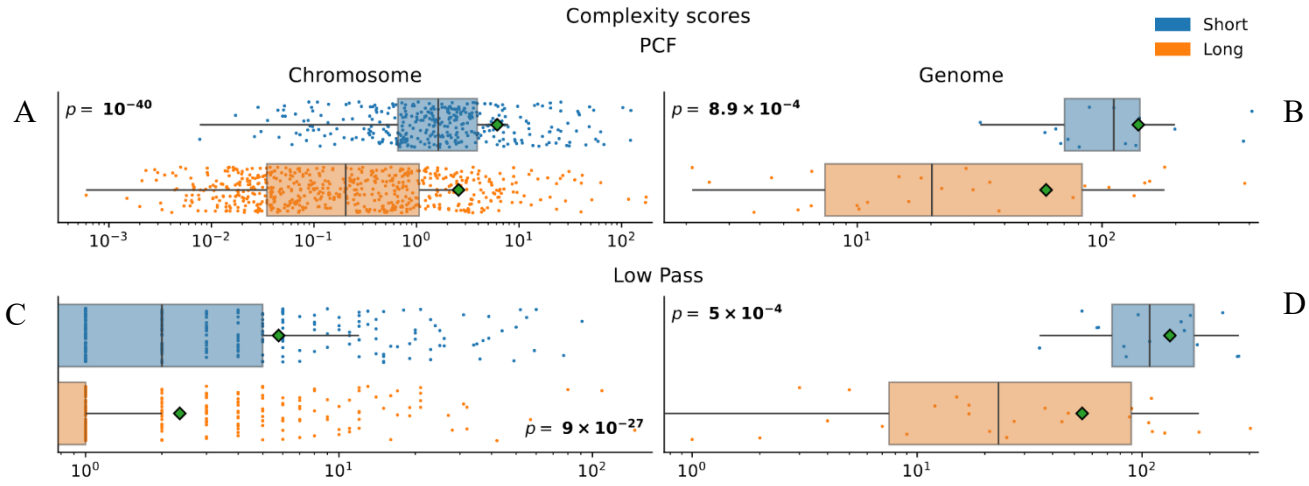


Figure 4.5 . Box plot displaying the complexity scores for both chromosomes and genomes (sum of chromosomes) for both the PCF and low pass methods for calculating scores for samples above (long) and below (short) the 3.81kb telomere length threshold. The PCF complexity score is calculated as the sum of RCN for all segments across each chromosome. The low pass complexity score is calculated as the sum of all differences between peaks and troughs of the RCN signal after being passed through a low pass filter.

To highlight the difference in visualisation methods, the same sample (DB195 with TL 2.9kb) is displayed using the standard copy number plot (figure 4.6A) along with the transformed copy number plot (figure 4.6B). The standard plot is useful for displaying the loci of RCN changes; however, it is harder to distinguish the finer detail especially in the smaller chromosomes. For example, the complex RCN changes in chromosome 20, the oscillations are somewhat visible but are much more apparent in the transformed plot (figure 4.6B). Copy number variations are only able to give insight into part of difference in genomic complexity between shorter and longer telomere samples, so structural variants

were analysed next to uncover more of the story.

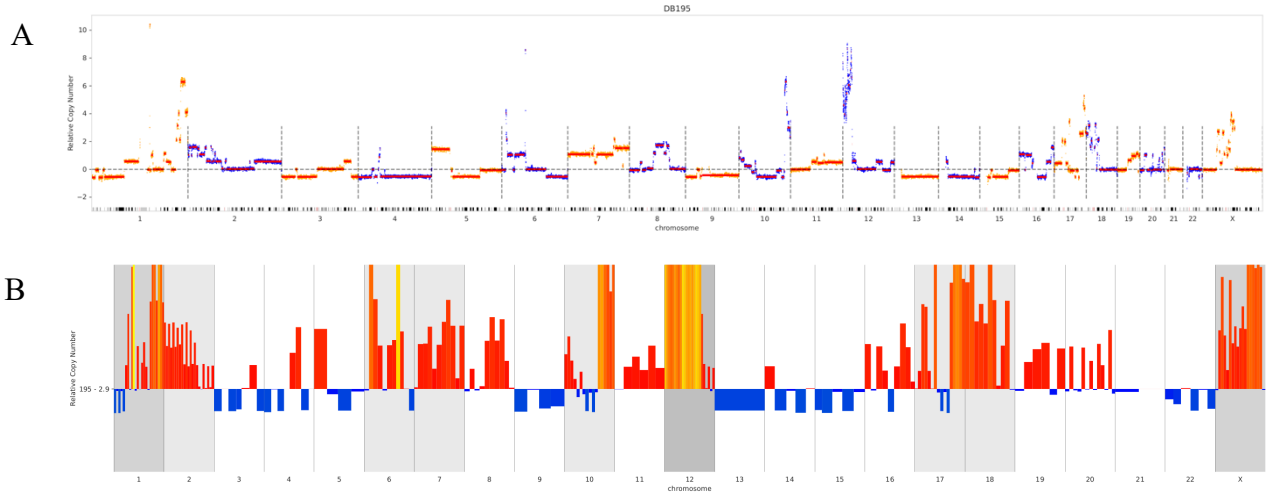


Figure 4.6 A comparison of the visualisation methods of plotting the processed RCN values. The top (A) shows RCN (y-axis) of 10kb windows (yellow and blue) across the genome (loci x-axis) with PCF segments overlayed (red). The bottom (B) shows the same PCF segments with their lengths scaled by logging, and the y-axis is clipped to +3 and -2. RCN is displayed beyond the clipping using colour (with yellow indicating higher RCN values).

4.4.2 Structural variants

Following from the copy number analysis, which found that telomere length was able to segregate samples with a high complexity score ($p<0.001$), structural variants were investigated to see if a similar trend could be found. A random selection of structural variants identified by dysgu were first manually curated using the gw genome browser to determine suitable PROB thresholds for filtering, followed by narrowing of the PROB threshold to minimise false positives and maximise sensitivity. The end result of this interrogation are shown in Table 4.1 which shows the approximate values for true positive rate and 1-FNR (similar to recall).

SV Type (PROB threshold)	DEL (0.15)	INV (0.15)	DUP (0.15)	TRA (0.45)	INS (0.2)
TPR	0.86	1.00	0.98	0.9	0.94
1-FNR	0.90	0.79	0.82	0.88	0.51

Table 4.1 Table displaying the True Positive Rate (TPR=Ntp/Ntotal positive) and 1 minus False Negative Rate (1 – Nfn/Ntotal negative) for each structural variant type when filtering using the dysgu PROB value (shown in brackets). Manual classification was performed on a random subset of 50 variants above and 50 below the PROB threshold.

Probability values used for threshold filtering appeared to be low compared to the default threshold of 0.45 using for germline calling. However, this is partially explainable as the dysgu pipeline was trained on single germline samples, whereas this pipeline uses a tumour-normal subtraction, so a lower threshold appeared to be suitable for this cohort.

Following a similar trend as the copy number complexity scores, the data displayed in Figure 4.7 shows that the overall number of structural variants appeared to be higher in the shorter telomere category of samples ($p<0.05$). Most significant is the difference in the number of deletions and duplications ($p<0.005$), whilst duplications and translocations also show an increase ($p<0.01$). Whilst the filter confidence value remains consistent across variant types, only translocations and insertions are not subjected to the size filter of 10kb as translocations do not have an length by definition, and insertion length is not possible to determine using short paired-end reads in an accurate manner. This value of 10kb was chosen to eliminate smaller SVs, and instead aim to focus on large-scale genomic rearrangements that are thought to be prevalent in complex genome rearrangements. Insertions, whilst following a similar trend are not included in Figure 4.7 due to not being amenable to filtering by size (present in Appendix).

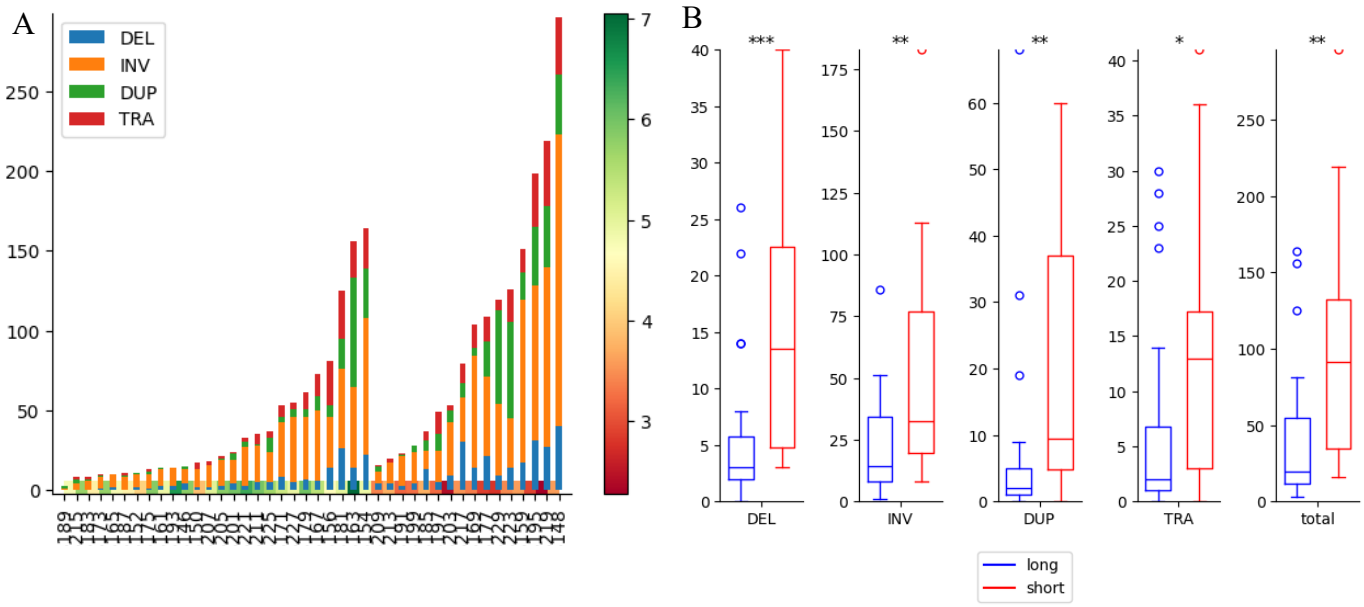


*Figure 4.7 . Stacked bar plot displaying the number of each type of structural variant, with the gradient bar showing the telomere length of sample in kb (A). Box plot showing the distributions of each SV type in the short (<3.81kb TL) and long (>3.81kb TL) groups with * displaying the 0.05, 0.01 and 0.005 confidence level of Mann Whitney U short greater than long (B)*

The circos plots in Figure 4.8 show the distribution of structural variants across the genomes in order of telomere length (displayed top right of each plot). A red line drawn across the plot indicates that sample above are below the 3.81kb threshold, and below plots are samples above the 3.81kb threshold. Samples above the line generally exhibit more SVs distributed across chromosomes. There are some samples with complex patterns below the line such as the last sample with a telomere length of 7kb, however they tend to be less frequent. An expected yet notable characteristic of all the samples is that SVs are aligned with the chromosomes that exhibit large copy number changes. Loci that appear more than others include: 1q23.3, 1q44, 3q26.33-28, 6q22.1-27, 8p12-11.21, 8q13.1-24.23, 10q23.31-26.3 (with translocations in 10q26.13-26.2), 12p13.11-24.23, 17q21.31-25.3 (close to BRCA1), 20q13.13-13.33 (lots of translocations 13.31).

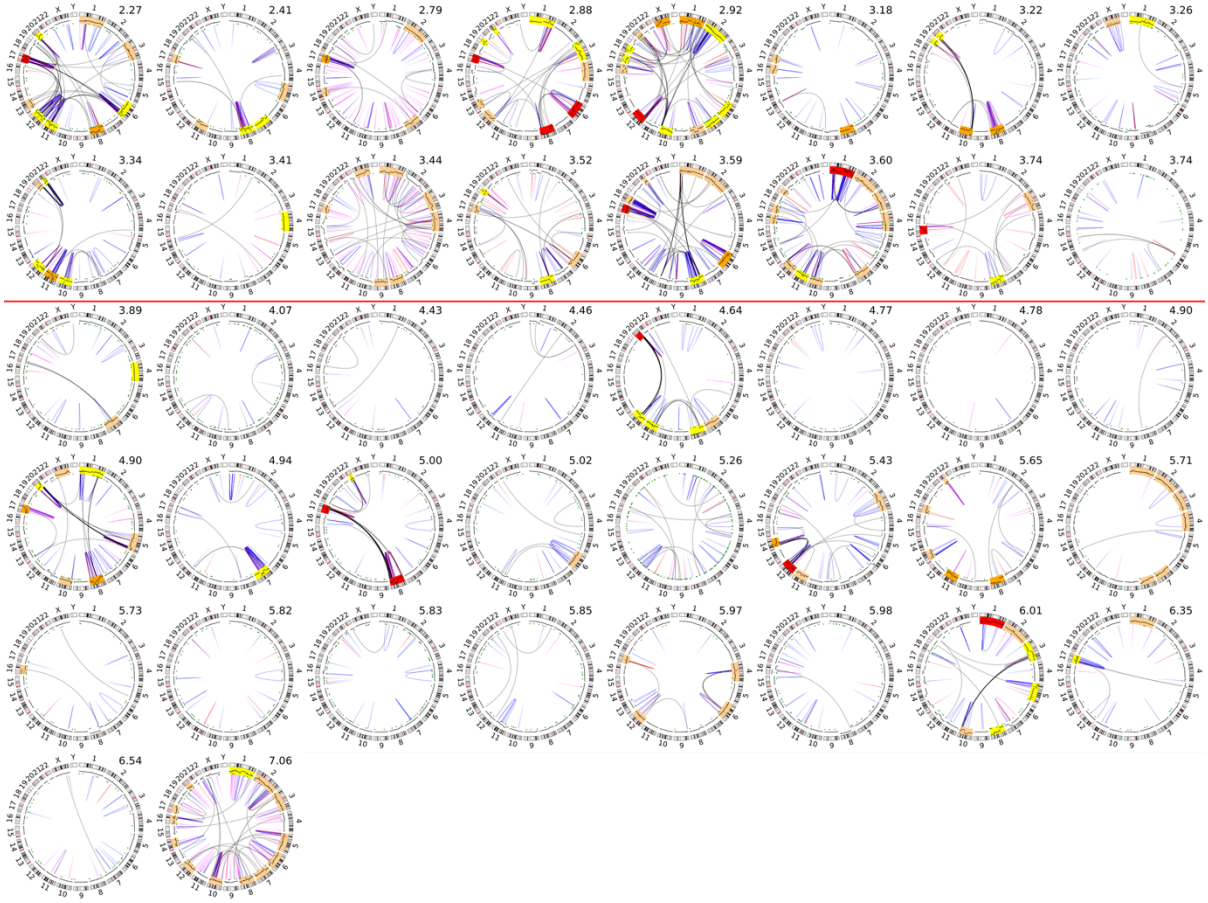


Figure 4.8 Circos plots showing structural variants by - black: translocation, red: deletions, green dots: insertions, blue: inversions, magenta: duplications. Copy number segments are plotted around the circumference with highlighting showing chromosomes that reach different complexity score thresholds 10-30 yellow, 30-50 orange, and > 50 red. Chromosomes with more than 8 segments are also highlighted in light orange. Red line across plot indicating split between 3.81kb threshold.

Structural variant counts in isolation give us a clue as to the levels of genomic complexity, but to investigate how these genomic rearrangements have occurred, we need to investigate the proximity and contents of these variants. This was done through chain linking and contig assembly analysis.

4.4.3 Chain link

The aim of the chain linking analysis is to gather insight into how structural variants are related and arranged across the genome. Through statistical testing of the proximity of

breakpoints, clusters of SVs that potentially occur during a complex genome rearrangement event can be isolated and characterised. Elbow plots displayed in Figure 4.9 were used to determine which p-value to use as a cut-off for the binomial distribution, testing for whether SVs are closer than expected by chance alone. The left plot shows a clear dip between 0.1 and 0.15, with a slower decrease until 0.45 before the p-value starts increasing (figure 4.9A). The right plot displaying the results for p-values between 0.01 and 0.05 shows the first significant difference ($p<0.05$) between the partition occurs at 0.04 (figure 4.9B). Ultimately the value of 0.4 was decided to be the value for this dataset for calling SVs as being closer than expected (and therefore possibly linked) as it exhibited the clearest difference between the shorter and longer categories of telomere length. A possible explanation for the high values of p-value thresholds (0.3-0.45) where the optimal split lies, is that the majority of the SVs within the shorter telomere length group were in fact chained SVs. As a result, a higher p-value is needed to account for these SVs being used to calculate the expected breakpoint distribution being tested against is needed to include these variants.

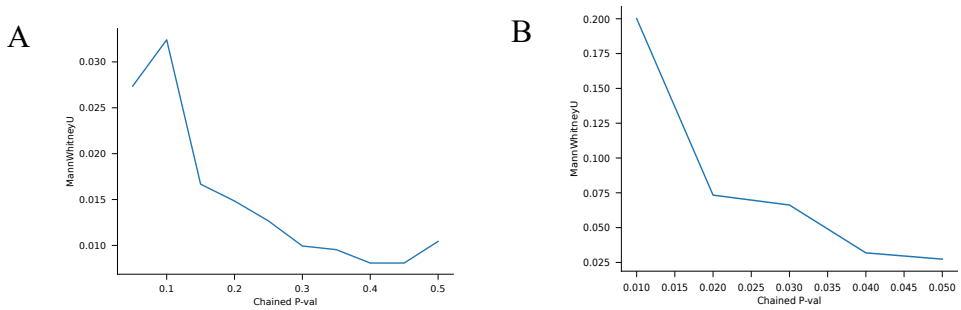


Figure 4.9 Elbow plots displaying the p-value for split in groups by the binomial p-value used to test whether a cluster of SVs is closer than expected with two different ranges (A 0.05-0.5 left), (B 0.01-0.05 right). Together they display an increase in significance in difference of number of linked SVs above and below the partition of TL 3.81kb between 0.01 and 0.45 except for a small increase between 0.05 and 0.1

Clusters of chain linked structural variants appear at higher frequencies in the shorter telomere category of samples. Only 4 out of 16 (25%) samples above the line (figure 4.10A) have no obvious clusters, whereas 18 out of 26 (~61%) below are without clusters (figure

10B, p-value 0.01 Fisher exact test). Clusters of chain-linked SVs in the shorter category of telomere length also appear to be larger, consisting of more SVs than the longer category. As well as being larger, clusters on samples with shorter telomere lengths appear to involve SVs on more chromosomes (figure 4.10A). This pattern of clusters of SVs on multiple (>3) chromosomes is consistent with the characteristics of chromoplexy (Baca et al. 2013).

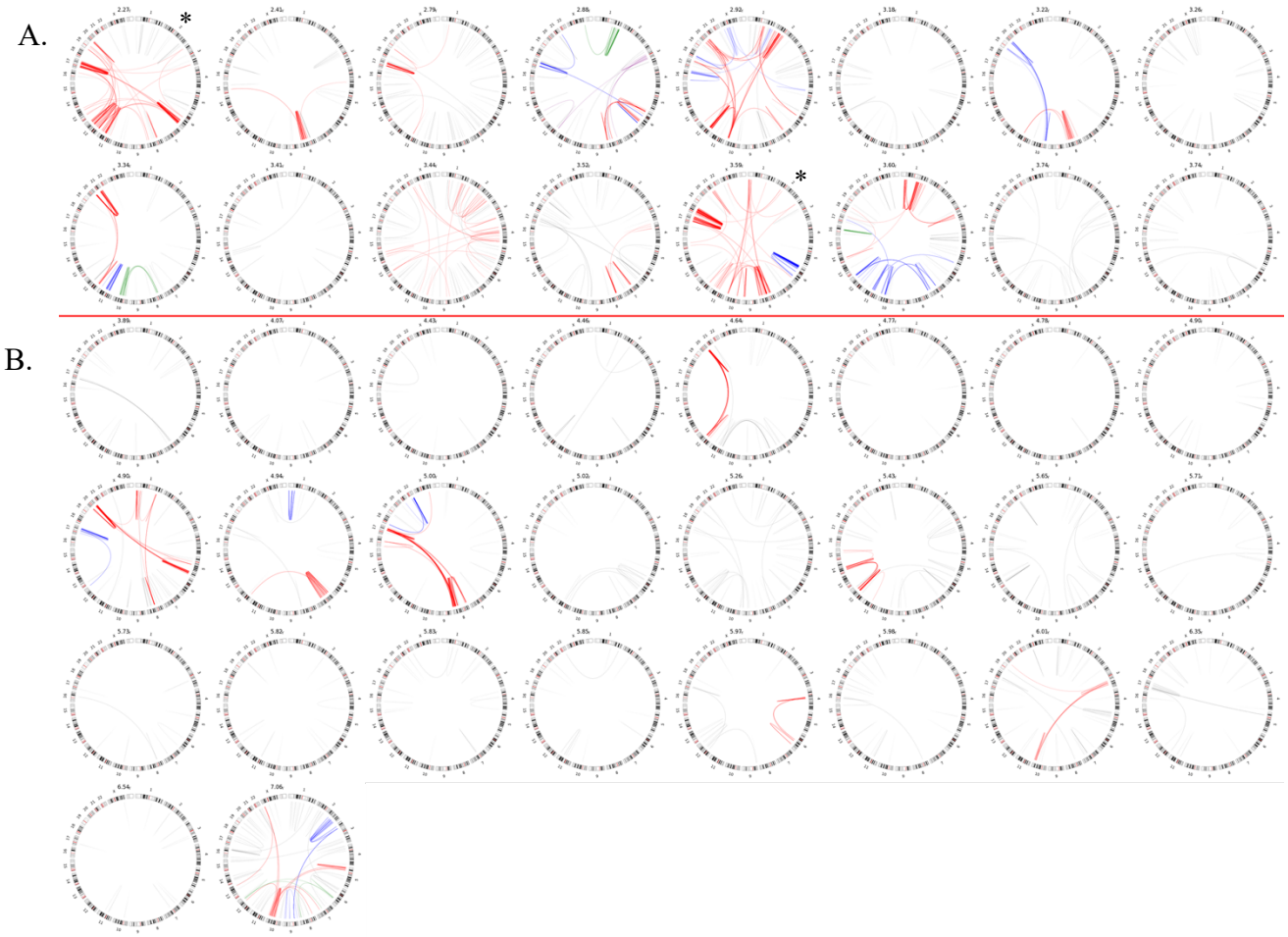


Figure 4.10 . Circos plots displaying the clusters of chain-linked variants. Each colour (red, blue, green) depicts a different chain, while non-clustered variants are shown in grey. Examples given in Figure 4.11 are marked by an asterisk.

Performing the linking with a much smaller p-value such as 0.04 shows some of the larger (red) clusters as seen in the first (DB219) and 5th from the left on the second row (DB148) may be comprised of smaller subclusters (red and blue) (figure 4.11). Long read sequencing may aid in the process of resolving the complexity shown in these plots. With longer reads it

would be possible to assemble individual genomes (contigs), or more accurately call SVs, more definitely defining the links between the SVs.

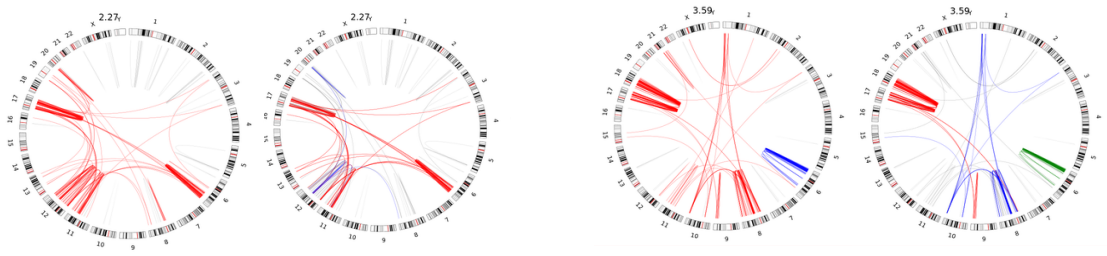


Figure 4.11 Circos plots from two examples: DB219 left two circus plots and DB148 right, displaying clustering of structural variants using two different p-value thresholds (0.4 left, 0.04 right)

Plotting the number of chained and unchained SVs as a stacked bar plot in order of telomere length reinforces the trend shown in the circos plots that the SVs tend to be closer in shorter telomere samples (figure 4.12). The overall shape of the stacked bar graph displaying the number of chained and unchained SVs is also similar to Figure 4.3 which shows the number of gains and losses as defined in the copy number analysis, with a tendency for shorter telomeres samples to have higher numbers of chained and non-chained SVs.

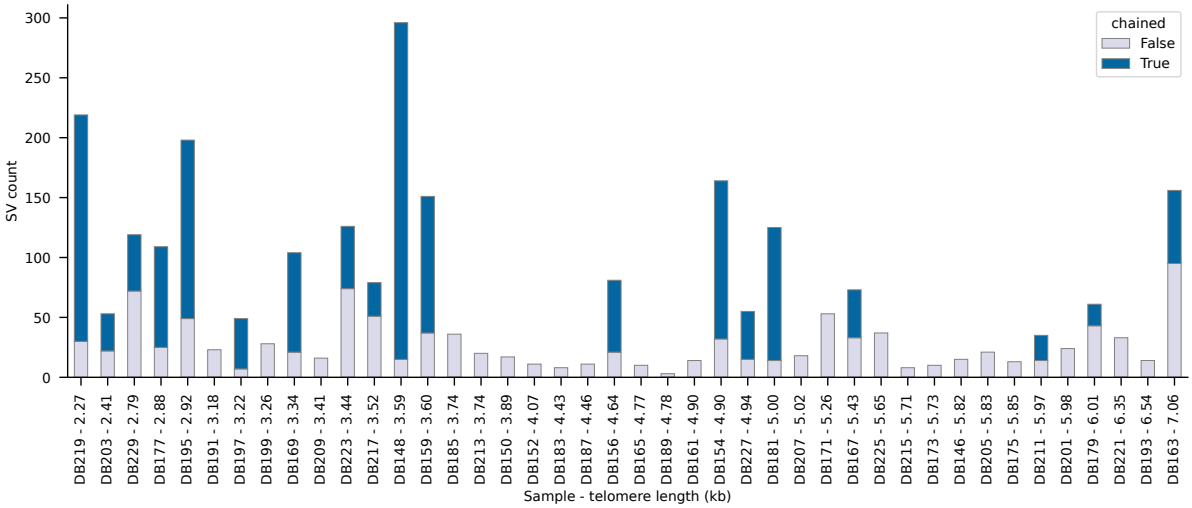


Figure 4.12 Stacked bar plot showing the number of chained and non-chained SVs ordered by sample telomere length.

Interestingly, this difference was statistically significant for both chained and non-chained SVs (figure 4.13). The p-values from Mann Whitney U (is greater than) testing were 0.008 for number of chained, and 0.023 for non-chained.

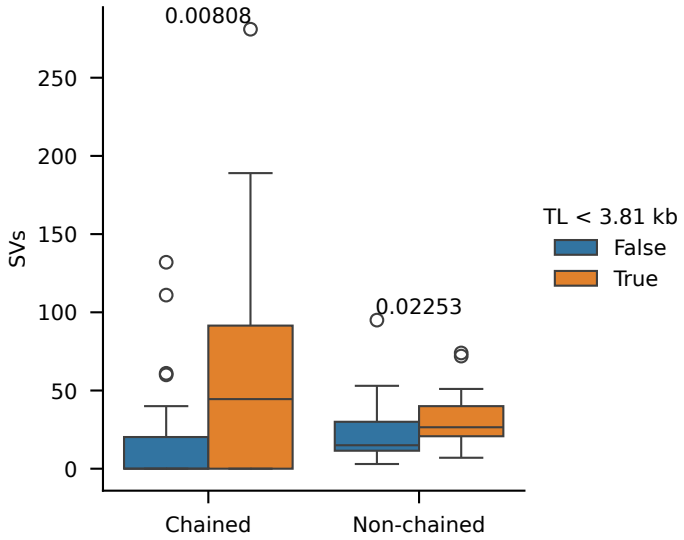


Figure 4.13 Box plot showing the distribution in number of chained and non-chained SVs for samples above (blue) and below (orange) a telomere length threshold of 3.81kb. The Mann Whitney U p-value for testing if shorter is greater than longer above the bars (p=0.008 for chained, p=0.02 for non-chained).

To verify the validity of the clustering, the assortativity and modularity of the clusters were compared to that of a randomly generated graph (figure 4.14). As explained in the methods chapter, the algorithm constructs a graph, using breakpoints as nodes, with linking nodes if they are closer in distance than expected by chance (Baca et al. 2013; Cleal et al. 2019). Assortativity in this case measures the tendency for nodes to be connected to neighbouring nodes with a similar degree (number of out-edges), where high-degree nodes tend to be joined with other high-degree nodes, and vice versa. Modularity is another way of measuring the strength of division of a graph into modules (groups of chained SVs). The assortativity and modularity of graphs generated from the breast cancer cohort were compared to those measured from equivalent but randomly joined graphs, permitting an analysis of the randomness of joining within the identified chains of SVs. For both metrics, when comparing the distribution of the coefficients compared to those from the randomly

generated graphs a significant difference is shown with assortativity at $2.7x10^{-8}$ and modularity $1.4x10^{-7}$. The figure displays this information as a histogram overlayed with a KDE fit for the distributions in the values. This data is in line with previous findings, that reported SV chains identified in samples that had undergone a telomere crisis, showed important deviations from randomness (Cleal et al., 2019). These findings argue against a completely random end joining process which is often presumed to occur in chromothripsis-like events, and instead suggest a multi-step process with multiple rearrangement events occurring over time, or being spatially confined on the genome in some way. However, more experiments using model systems will be needed to confirm this hypothesis.

To investigate chain linked (clustered) SVs further, the output from this analysis was used as input for contig assembly analysis.

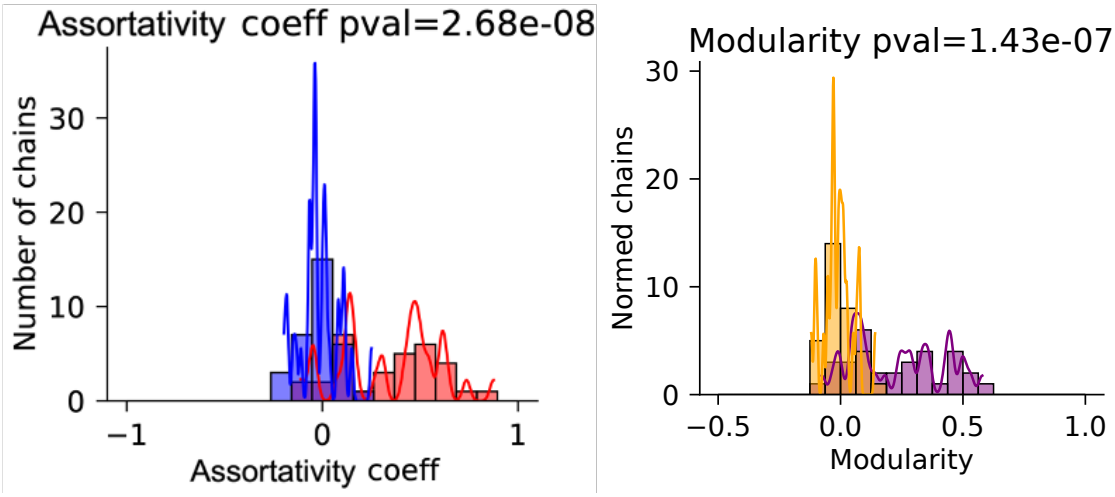


Figure 4.14 Assortativity coefficient values for random (blue) and SV clusters (red) left. Modularity of random (yellow) and SV clusters (purple).

4.4.4 Contig assembly

Assembly of chain linked SVs into contigs was performed to find phenomena that would explain potential mechanistic origins. As the goal of this analysis was less about comparing subgroups of the cohort split by telomere length, the efforts to interrogate this data was

focused on a per sample basis. An example of the visual output is shown in Figure 4.15 for the sample DB229 with a telomere length of 2.79kb. This figure shows the mappings of each contig with the chromosome denoted by the larger coloured rectangle, and unmapped insertion sequences as a line that joins them. The bar displayed across the top of the rectangle shows the proximity to the of the region to the telomere end through a gradient, with black indicating the mapped loci being ≥1mb away, reds ~600kb, orange ~400kb, yellow ~200kb (key shown to right of plot). Although difficult to distinguish, the colour of each rectangle is slightly transparent to show cases where mappings overlap which suggest microhomologies.

Figure 4.15 shows multiple complex contigs that include regions from several chromosomes. Of these, multiple contigs are observed with microhomologies (e.g. third from the top middle two regions), suggesting NHEJ or MMEJ as potential mechanism for repair that led to the rearrangement. These regions of microhomology are mostly ~4-8bp except for a larger overlap between breakpoints seen in the second to top and bottom contigs (~10-20bp). It is also shown it is not uncommon for regions relatively close to telomeres to be involved in these rearrangement events, with several mappings being within 1mb of the telomere.

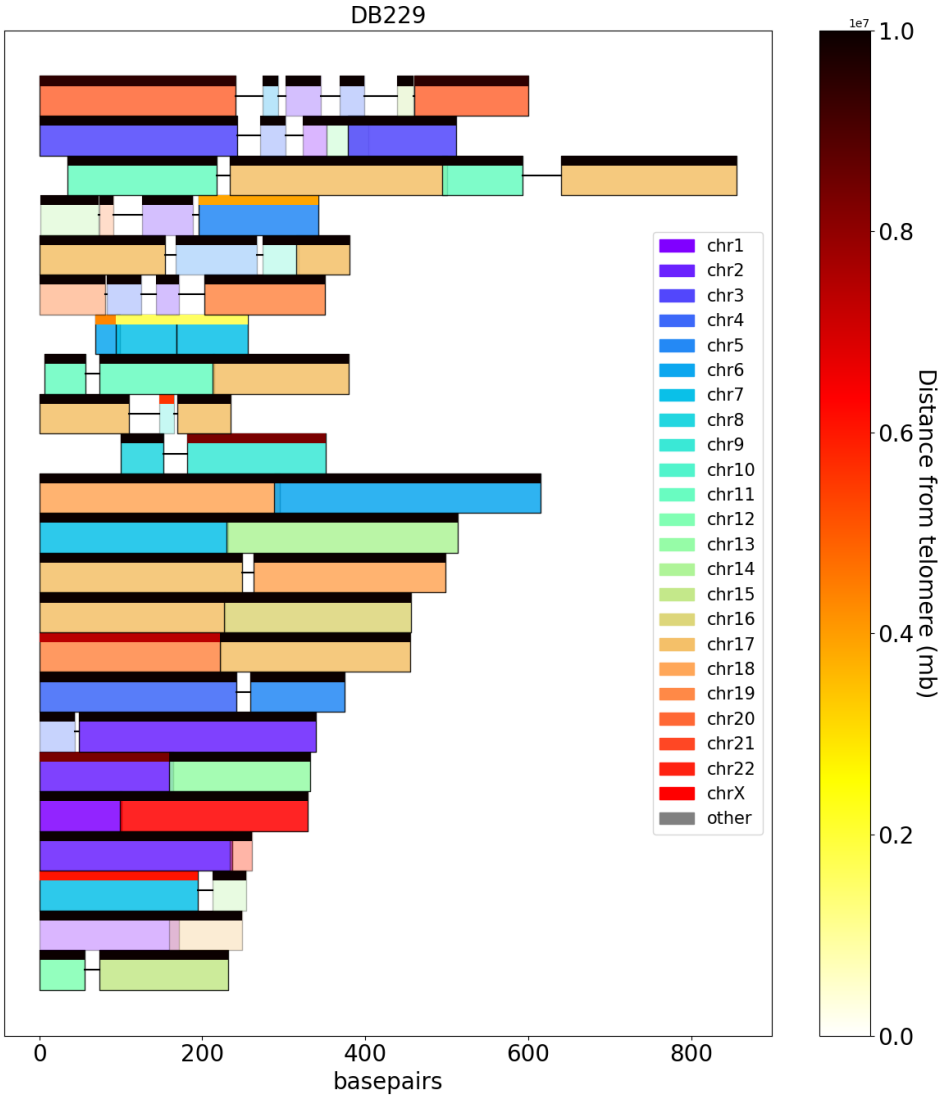


Figure 4.15 Visual representation of assembled chain linked SV contigs for sample DB219. The x-axis indicates the bp length of each contig, with chromosome of each region of the contig is mapped to by the large, coloured rectangle, and unmapped insertions as a line connecting them. The colour of the bar at the top of each rectangle indicates the proximity of the region in the chromosome to the telomere end with the key shown as the white, yellow (~200kb), orange (~400kb), red (~600kb), black (≥1mb) gradient bar to the right of the plot. The histogram located at the bottom right of the plot shows the total count (y-axis) of contigs containing for each number of distinct regions that are mapped in the contig (x-axis).

4.4.5 Low coverage copy number testing

Aside from investigating the correlation between telomere length and genomic complexity, we wanted to test whether the copy number pipeline was viable as a means of investigating complexity in extremely low coverage sequencing runs. The method for simulating this is

outlined in the methods chapter under the low coverage copy number testing section, but in summation utilises samtools to subsample bam files to a coverage of 1x for each sample. It was expected this input would generate a far lower resolution version of the full coverage analysis, compared to the ~15x coverage of the cohort alignment files. What was observed however, was a considerable similarity between the relative copy number analysis of the full ~15x coverage and subsampled 1x coverage datasets as shown in the Figure 4.16 heatmap. As expected, there are differences, such as the finer detail of chromosome 1q in sample DB159, 2p in DB195 being lost in the subsampled version.

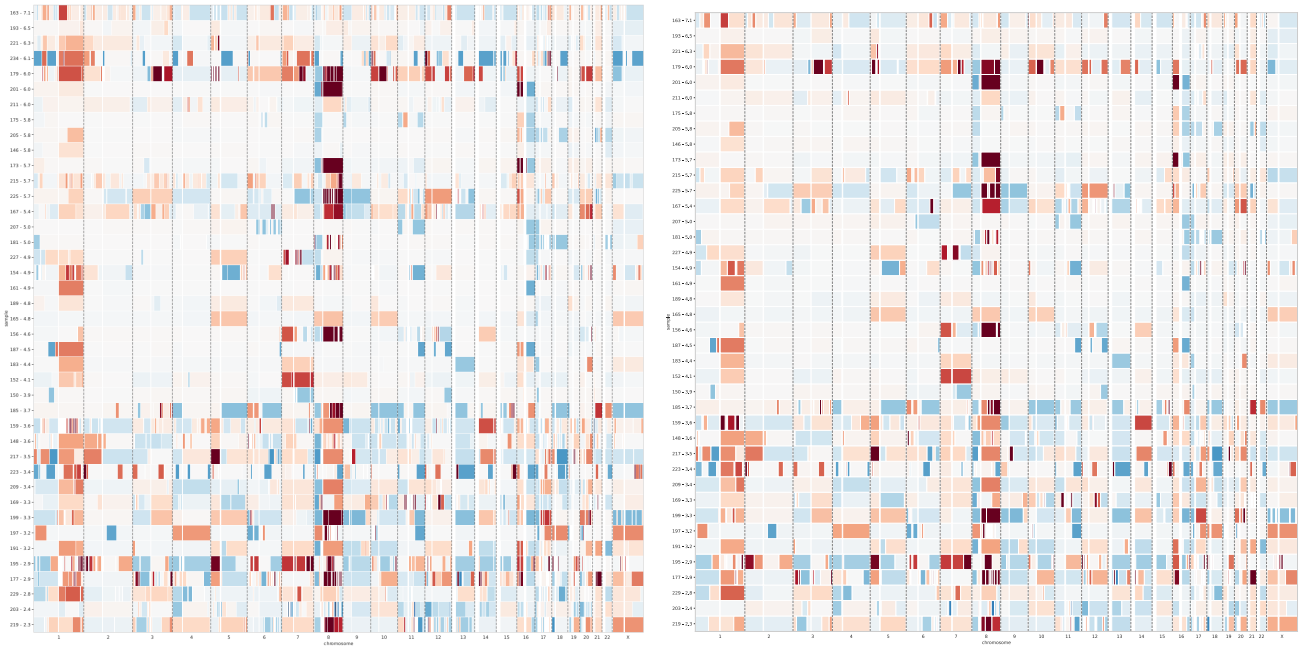


Figure 4.16 Heatmaps showing relative copy number of the full ~15x coverage cohort (left) and subsampled ~1x cohort (right) segments from pcf analysis (input of 10kb coverage windows) clipped between -1.5 and 1.5 with the Y-axis displaying ordered from longest telomeres (top) to shortest (bottom).

Similar differences in the complexity scores between the longer and shorter cohorts can also be seen with the subsampled RCN analysis. P-values for Mann Whitney U (short > long) tests are comparable with the low coverage PCF method being $8.5x10^{-45}$ (~15x $10x10^{-40}$) for chromosomes, and genomes $2.9x10^{-4}$ ($8.9x10^{-4}$), and low pass chromosomes 1.4x10-26 ($9x10^{-27}$), and genomes $2.9x10^{-4}$ ($5x10^{-4}$).

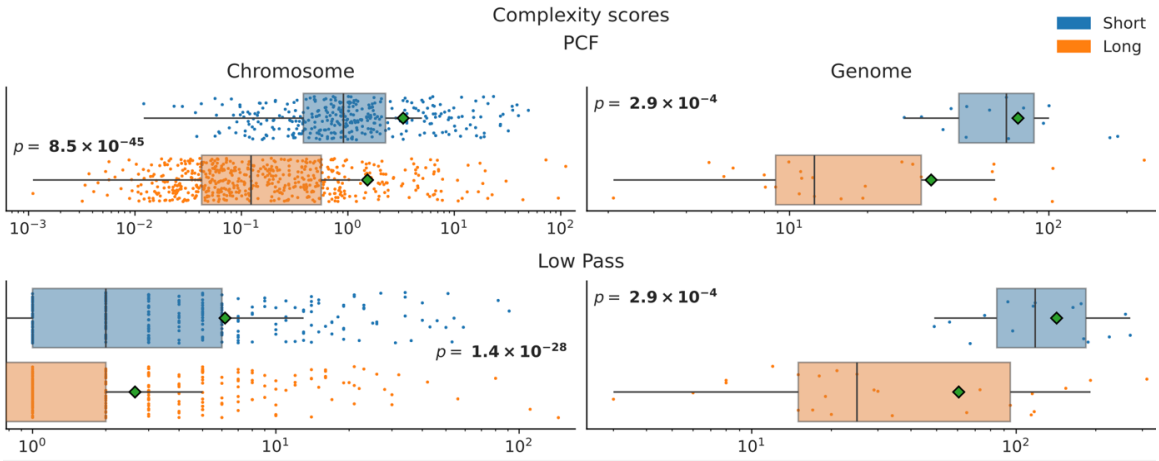


Figure 4.17 Box plot displaying the complexity scores for both chromosomes and genomes (sum of chromosomes) for both the PCF and low pass methods for calculating scores for samples above (long) and below (short) the 3.81kb telomere length threshold in the subsampled to ~1x coverage dataset. The PCF complexity score is calculated as the sum of RCN for all segments across each chromosome. The low pass complexity score is calculated as the sum of all differences between peaks and troughs of the RCN signal after being passed through a low pass filter that is greater than 0.5.

These results show that it is possible to use much lower coverage sequencing in analysis of genomic complexity, albeit restricted to copy number profiling. However, more importantly the results pertaining to telomere length and genomic complexity, it has been shown there is a correlation between the two.

4.5 Discussion

4.5.1 Copy Number Variation

CNVs can be responsible for the aberrant expression of RNA, which are potential driver in the development and progression of many types of cancer when involving onco and tumour-suppressor genes (Shao et al. 2019). Common CNVs presented here in Figure 4.2 heatmap have also been identified by Courjal and Theillet as frequent sites of CNV, and correlate well with the findings of Beroukhim et al looking at CNVs across multiple cancer types (Courjal and Theillet 1997; Beroukhim et al. 2010). For example, Figure 1b from

Beroukhim *et al.* 2010 shows 5p having a significant z-score for gains, whilst 5q is 0 for gains and losses correlating with the low amplitude mix of both seen in the cohort. The clinical significance of 5p gains relates to the alteration of telomerase reverse transcriptase (TERT) gene expression, which Gay-Bellile et al linked to a poorer prognosis with a hazard ratio of 3 (Gay-Bellile et al. 2017).

Orsetti et al note that 50-60% of breast tumours exhibit abnormalities of CNV in chromosome 1 in the form of mainly 1p losses and 1q gains (Orsetti et al. 2006). Both features seem to be present in this cohort alongside their finding that high magnitude amplification is infrequence as shown in Figure 4.2. They also found that 20 out of 28 candidate oncogenes present on the 1q arm were overexpressed. Ideally our data would be accompanied by RNAseq analysis so this could be tested for in this cohort.

Chromosome 8 also shows a similar pattern where losses in the short arm and gains of the long arm have previously been recorded. Choschzick et al showed a strong correlation between 8q21 amplification and the expression of multiple oncogenes such as HER2, CCND1, MDM2, and MYC (Choschzick et al. 2010). Yaremko et al report that roughly half of breast cancers show a LOH in 8p and suggest that it occurs early in carcinogenesis (Yarernko et al. 1995; Yaremko et al. 1996).

Inversely, chromosome 16 shows gains of the p and losses of the q arm. A study by Rakha *et al.,* 2006 highlights LOH of chromosome arm 16q as one of the most regular events in breast cancer being present in 30-75% of breast cancers depending on the type (Rakha et al. 2006).

It is possible that the large magnitude of CN gains seen in chromosome 17 in Figure 4.4 could be the result of polysomy (>2 copies of the chromosome), which is present in ~32% of breast cancers (Hyun et al. 2008). It is unclear as to how polysomy 17 affects cancer development, with one study showing that it is not related to HER-2 overexpression

or has any effect on clinicopathological outcome (Vanden Bempt et al. 2008).

Chromosome 20q gains are present within a wide range of solid cancer types, and its been implicated in early cancer development. Cells with spontaneous 20q amplifications have been shown to exhibit deregulation in several specific cancer-related pathways including the MAPK, p53, and Polycomb group factors, as well as activations in several cancer initiation hallmarks such as Myc and ETS family transcription factors (Tabach et al. 2011; Gabay et al. 2014).

Outside of these previous observations, the low coverage simulation shows a remarkably similar output compared to the 15-20x coverage analysis. This is promising as large-scale low coverage sequencing (such as Illumina multiplexing) could be used as a cheap method for performing this kind of investigation. Theoretically, it is possible that Oxford Nanopore sequencing technologies such as the MinION or Flongle flow cells could provide sufficient coverage for such analysis as well. It is untested whether this low coverage long read approach would also work for analysis of structural variants.

4.5.2 Structural variants

Structural variants (SVs) are a common feature in cancers that cause alterations of gene copy numbers, gene regulations, protein sequences, epigenetic signalling, and 3D genomic structure. The increased frequency of large (>10kb) deletions, inversions, duplications, and translocations shows that by raw count the shorter telomere group is more complex in the structure of their genomes. This increase was seen in Figure 4.7 with the boxplot displaying the distributions, and Figure 4.8 where it was seen with the location contexts (and often harmonising with the copy number analysis displayed in the outer rings). Gene Set Enrichment Analysis (GSEA) of chromosome bands that exhibited most frequent SVs and follow up RNAseq would be interesting to carry out. Cursory glances were made at these

regions using the UCSC genome browser, which found that the band 17q21.31-25.3 where increased SVs overlaps with the loci of BRCA1.

Whilst dysgu is an effective structural variant caller, the underlying short read sequencing technology is not ideal for identifying SVs. Measures were taken during the filtering stages to improve the end set of SV calls. These provided to be effective as shown by the lowest true positive rate being 86%, and (outside of insertions which were excluded) the "inverse" (one minus) false negative rate of 79% showing that even in the worst cases, the majority of SVs are likely to be real. To effectively include insertions in this type of analysis, ideally long read sequencing data such as that from oxford nanopore should be investigated. Long read data would also aid in the following chain linking analysis.

4.5.3 Chain linking

Chain linking analysis was performed to identify clusters of SVs that more likely than not came from the same event due to their proximity. The original authors of the ChainFinder algorithm (which was implemented and adapted here), used this approach to define these interdependent SVs chromoplexy (Baca et al. 2013). We expected to find larger and higher numbers of chain linked clusters within the shorter telomere side of the cohort as we suspected the chromoplexy phenomena (similarly to chromothripsis) is related to telomere shortening and dysfunction.

Unfortunately, there is no concrete way to determine a suitable or optimum p-value threshold for classifying whether SVs should be classed as linked. However, the fact there is a large window between 0.04 and 0.45 where significance is shown and generally increases between the two groups of telomere length suggests there is a difference. Further evidence for the validity of this observation is shown in the assortativity coefficients and modularity. Assortativity can be thought to be the correlation between linked nodes, and modularity as

the strength of (confidence nodes belong within) clusters. The increased number and frequency of structural variants in close proximity suggests that SVs in shorter telomere patients are likely to be related to a common event(s).

In other words, there appears to be a correlation between telomere length and the phenomena of chromoplexy. The original paper in which chromoplexy was observed shows a commonality between the occurrences of chromoplexy and ERG gene fusions (located on chromosome 21). Their dataset was working with prostate cancer, whereas this analysis is based on breast cancer. Whilst some SV chains are found to interact with chromosome 21, it was not found at the same frequency as Baca et al., which may be due to the difference in cancer types. A study conducted in multiple myeloma found a correlation between instances of chromoplexy and the deletion of 8p (Ashby et al. 2022). In this analysis however, 8p deletions were common regardless of telomere length, but chromoplexy was exhibited more frequently in shorter telomere samples. This suggests that the underpinning mechanism of chromoplexy is more dependent on telomere fusions and phenomena such as BFB cycling similar to that of chromothripsis.

4.6 Conclusion

Results from each analysis in this chapter demonstrate a clear relationship between telomere length and genomic complexity. This correlates with results found from Santos et al. 2015 which used florescent in situ hybridisation (FISH) and microarray based cytogenetic analysis (Dos Santos et al. 2015). Whilst the relationship between telomere length and genomic complexity is not linear, partitioning of these 44 breast cancer patients based on telomere length provides evidence for the hypothesis that cancer cells with telomeres below a “fusogenic” threshold exhibit significantly more complexity in their karyotypes. Visible in all aspects of this analysis, there is a clear divide between samples with “longer” (>3.81kb)

and “shorter” (<3.81kb) telomeres with regards to the levels of genomic complexity. This divide is shown in patterns of relative copy number variation, increased amount and proximity of large structural variants. The significant differences in complexity when partitioning by telomere length suggests that the mechanism behind the increased complexity is one by which chromothripsis and chromoplexy are caused by telomere fusion events of critically shortened and dysfunctional telomeres. Shorter and therefore dysfunctional telomeres are more prone to fusions, which are then processed through mechanisms such as BFB cycling (or micronuclei) to generate the complex patterns that have been observed. Similar analysis will be conducted on more samples in the next chapter; however, this analysis will be conducted with telomere length predictions instead of absolute telomere length as determined with STELA analysis as in this chapter (Baird et al. 2003). Ideally a dataset for further investigation would be composed of more samples sequenced with a long-read method with accompanying absolute telomere length from STELA. Long read data would provide a suitable platform to further investigate insertions (missing from this analysis) with additional benefits in contig assembly being easier and more accurate.

Chapter 5 Using publicly available genomic data to analyse the relationship between telomere length and genome complexity

5.1 Abstract

Previous research and data from the previous chapter have shown links between telomere length and genomic complexity (Dos Santos et al. 2015). In this chapter we expect to see similar results from the previous chapter where cohorts are divided in their level of genomic complexity clearly by a telomere length threshold. This chapter aims to analyse cancer cohorts from public repositories to explore their complexity in relation to their predicted telomere length. This exploration utilises the methods outlined in the previous and methods chapter exploring copy number variation and structural variant analysis. In the largest cohort (Genomics England breast cancer) a telomere length threshold established through analysis of the relative copy number profiles significantly segregates the cohort in all analysis types. Similar but less clear differences are also found in the ICGC breast cancer and Genomics England chronic lymphocytic leukaemia cohorts. In all cases, the relationship is one by which there is classification into above and below a threshold rather than a direct linear correlation. These results indicate that telomere length is correlated with genomic complexity, supporting the theory that telomeres reach a critical length and generate fusions which leads to the observed increased genomic instability through the pathways outlined in the introduction (e.g. breakage-fusion-bridge cycling and micronuclei formation)

5.2 Introduction

5.2.1 Breast cancer

Breast cancer is the most frequent cancer in women (second most in combination of both sexes) and is heterogeneous with diverse genetic origins (Akar and Oktay 2005; Łukasiewicz et al. 2021). The previous chapter and other research have implicated telomere length in being a key driver in genomic instability, a big part of tumorigenesis. Currently patients are diagnosed and stratified for treatment by a limited set of factors such as biomarkers or driver mutations (Neves Rebello Alves et al. 2023). There has been a growing interest and shift to utilising whole genome sequencing (WGS) data to better inform prognosis and clinical treatment (Rossing et al. 2019). Breast tumours exhibit specific mutational signatures which are attributable to the underlying mutational processes. In addition, rearrangement signatures have been identified that are indicative of deficiencies in homologous recombination activity that arise because of mutations in BRCA1 and/or BRCA2 and other yet undefined mechanisms (Nik-Zainal and Morganella 2017). Telomere dysfunction has been identified as a key mechanism responsible for genomic instability in breast cancer (Maciejowski and de Lange 2017). Additionally, this chapter will also investigate a cohort of chronic lymphocytic leukaemia patients.

5.2.2 Chronic lymphocytic leukaemia (CLL)

CLL is the most common type of leukaemia found in adults, characterised by a build-up of dysfunctional lymphocytes (Mukkamalla et al. 2024). CLL was one of the original cancer types where fusogenic telomere-length thresholds were identified and has been well established that telomere length is a prognostic factor when in this disease (Lin et al. 2014).

Previous research has also hinted at the correlation between telomere length and genomic complexity being negative, with patients exhibiting shorter telomeres having more complex genomes. Alongside poor prognosis significant correlation of CLL samples with short telomeres and deletions of 17p and 11q have also been found (Britt-Compton et al. 2012). Cell cycle checkpoint genes TP53 and ATM which trigger upon telomere shortening and dysfunction are located in 17p and 11q respectively. Suggesting that this may be one mechanism by which CLL is able to undergo continued telomere shortening (compared to those without deletions) whilst avoiding apoptosis (Jebaraj and Stilgenbauer 2021).

5.2.3 Public repositories

Public repositories take many different forms but share the same goal of making biological data available for researchers around the world.

5.2.3.1 Genomics England

Genomics England provides a contained environment to access and process their wide range of data, which includes over one hundred thousand sequenced genomes from various cancers and rare disease patients. The system is broken into two main parts. The outer layer is a desktop “research environment” (RE), a remote desktop originally accessed via a webpage, now through amazon “WorkSpaces”. Due to security reasons, this environment does not have access to the wider internet, with only internal IPs being accessible. From the RE the second layer can be reached, their HPC “Double Helix” (its predecessor “Helix”). Data is accessible from both the RE and Double Helix, but only limited analysis can be run from the RE due to the resource restraints.

5.2.3.2 International Cancer Genome Consortium (ICGC)

The ICGC takes a different approach to Genomics England by not providing an environment to process data, instead only a means for accessing it. At the core of ICGC is the "score client" which is the method for downloading or mounting data from an AWS "bucket". This bucket is only accessible through EC2 instance from specified regions (West-Virginia), or through their now sunsetted platform for creating cloud compute instances "collaboratory". Sample IDs are extracted from a separate data portal, which facilitates the filtering of a cohort down to the cancer type and file types desired to be processed. These encoded names are what is given to the score client to download the chosen files.

Previous work has identified telomere length thresholds defined by assays detecting the presence of telomere fusions, which provide powerful prognostic information. In the previous chapter we showed that similar telomere length thresholds defined threshold for genomic complexity. In this chapter we aim to validate these findings by undertaking a similar analysis in large publicly available genomic data repositories.

5.3 Methods

5.3.1 Data

The majority of data in this analysis is Illumina whole genome sequenced breast cancer data with ~1550 samples from Genomics England (GEL) and 98 samples from the International Cancer Genome Consortium (ICGC) repository. There are also ~70 chronic lymphocytic leukaemia samples also from Genomics England with real telomere length data from STELA. All data was sequenced with paired-end Illumina (majority HiSeq X), with GEL having a

read depths around 30 to 80X for the germline samples, and 80 to 150X for cancer samples (all read length 150). ICGC was sequenced at a read depth of 30 to 80X for all samples.

5.3.2 Data access

Both Genomics England (GEL) and the International Cancer Genome Consortium (ICGC) require the submission of an application that requires approval before being able to use their respective methods of accessing data.

Genomics England (GEL) provides a "research environment" (RE, accessed via amazon workspaces) with access to a High Performance Compute (HPC) cluster. This HPC uses the LSF job scheduler, which uses bsub for the submission of jobs. Whilst there is a restrictive "airlock" system for importing scripts into the RE, due to dependency and path issues it is more convenient to utilise containerisation. Each tool was added to a docker container which can then be downloaded as a sif file by singularity (using the old "helix" cluster as the newer "double helix" cluster doesn't have the Docker hub IP whitelisted). Whilst this is easier for getting runnable code into the RE, it does however mean even a single line edit in a script does require recompilation and uploading of a container each time a bug is encountered. This is still probably faster however than getting past the initial airlock and dependency importation pains, as vim can take upwards of 45 minutes to close when not writing files. Labkey was then used to download a csv of file paths and germline pairs, which after heavy manipulation was converted to a sample pairs csv.

The International Cancer Genome Consortium (ICGC) requires setting up your own compute cluster using amazon EC2 instances or as used in this analysis their sunsetted collaboratory instance service. A successful project application grants you access tokens for the score-client, their proprietary method of being able to interact with the data. Whilst possible to mount the dataset as a bucket to an instance, the throughput of reading these files

is very slow at <1mbps, so downloading is required for analysis to be done in a timely fashion. Using the ICGC data portal a cohort of suitable files was generated for download across 8 machines for parallel processing.

5.3.3 Structural variant calling

Genomics England breast cancer and CLL cohorts were analysed with the same method. This involved piping a subsampled bam (using variable frac to set to a coverage of 40) file into dysgu.

```
samtools view -h -s ${frac} --subsample-seed xxx ${path} | tee
>(singularity exec teltool.sif teltool trim -i -n ${s} -o /home) |
singularity exec dysgu.sif dysgu fetch --mq 30 --max-cov 5000 --clip-length
30 /tmp/tmp_${s} - 2> ${logdir}/${s}.log && singularity exec dysgu.sif
coverage2bed.py --out-bin-size 10000 -w /tmp/tmp_${s} >
${coverage_dir}/${s}_cov.bed && singularity exec dysgu.sif dysgu call --
ibam /data/${s}.bam -v2 --metrics --low-mem -x ${reference} /tmp/tmp_${c} -
o /out/${s}.vcf 2>> ${logdir}/${s}.log; rm -rf /real/tmp/dir/tmp_${s}
```

Both subsampling and using tee to also pass the output into teltool was to save processing time by saving a second pass and in some cases reducing the number of reads both programs were reading by more than half. Dysgu fetch was given the parameters of mapping quality 30, a max coverage of 5000, and clip length of 30 as these are suitable values for processing a 40x coverage sample. After dysgu fetch, the coverage2bed script was also used to create a coverage profile for each sample using a window size of 10kb for later copy number analysis. Finally dysgu call is then used passing the necessary inputs as well as the v2, metrics, and low mem flags, that last of which was to minimise the memory usage to decrease the required reserved memory for each job, increasing the number of jobs that could be run at the same time. The v2 flag was used in case it was required to merge vcfs downstream, and metrics would make it more convenient for running a newer model if one was made during the analysis.

The ICGC cohort was analysed using the dysgu run method instead of fetch and call, then also calling the coverage2bed script (${cov}) to retrieve the coverage profile. Due to there being a range in the coverage each bam file in the cohort (~30-100x), dysgu was given the parameters for analysing the lower end of the coverages. This involved using a mapping quality of 15, unlimited max coverage, clip length of 30, and a min support of 5. This version of dysgu did not yet have the support fraction option, which would have been more suitable for this kind of analysis with varying amounts of coverage.

```
dysgu run -v2 --metrics --low-mem --exclude /home/ubuntu/hg19-
blacklist.v2.bed --mq 15 --max-cov -1 -x --clip-length 30 --min-support 5 -
-search /home/ubuntu/chromosome.bed ${ref} tmp_${b} ${i} -o ${b}.vcf 2>
${b}.log ; ${cov} --out-bin-size 1000 -w tmp_${b} > ${b}_cov.bed ; rm -rf
tmp_${b}
```

For filtering the vcf files, as GEL has all data available on the HPC (Helix/Double Helix, it is possible to simply loop over a csv file containing the names of the tumour normal pairs and pass these into dysgu filter. For practicality, it was decided to only use the paired normal instead of a pool of normals as both container mounting of a random set of samples, and processing time were both hurdles not worth jumping. However, for the ICGC data as a machine can only hold one sample at a time a different method had to be used. The first step was to create a bed file containing all the loci of all SVs across all the tumour vcf files. This bed file was then used to create “cropped” normal bam files, from which dysgu filter could search for the relevant reads

```
While iterating normal list, download normal bam:
      samtools view -hb --region-file tumours.bed ${bam_file} -o
      trimmed_normal/${b}.bam
```

5.3.4 Telomere prediction

The Genomics England CLL samples had accompanying telomere length data generated using STELA for the ARTIC and ADMIRE clinical trials (Norris et al. 2019). GEL breast cancer data had telomere length data predicted using telseq using the command below.

```
while iterating sample_pairs.csv:
    bsub < ${s}.sh
${s}.sh = (singularity exec telseq.sif telseq /path/*.bam -o /path/${s}.tsv)
```

The outputs of the individual telseq files were then combined using python and pandas to create a dataframe where each sample could be concatenated and written to a csv file. Additionally, all telomere length estimates from telseq were incremented by a size of 1kb as the telseq authors note a constant underprediction by approximately this amount (compared to mTRF) (Ding et al. 2014). For the ICGC breast cancer data, teltool was used to predict the telomere lengths. This involved creating the precursor trimmed teltool file on the remote machine, then processing these locally once transferred.

```
On remote machine:
      teltool trim -i ${i} -o teltool_out
On local machine:
      teltool test teltool_out
```

5.4 Results

For each form of analysis, the results will be displayed for each cohort separately. All analyses used in the previous chapter besides contig assembly are repeated here, where we expected to find similar results in the breast cancer cohorts. Due to the differences between the cancer types, CLL was not anticipated to follow the previously observed complexity profiles, rather that it would follow the same general trend in differences between the levels of genomic complexity for samples above and below a telomere length threshold.

5.4.1 Copy Number

5.4.1.1 Genomics England breast cancer

A plot showing the relative copy number of the GEL breast cancer cohort (n=1591) as a heatmap with gains as red and losses as blue, where the amplitude is indicated by colour intensity (clipped at values ±1.5) ordered based on telomere length prediction is displayed in Figure 5.1. It shows there was a trend where the ~1/3 of predicted shortest telomere length patients (indicated by the arrow) exhibit a darker (thus more complex) relative copy number profile. Despite the distinction however, there are still expected commonalities between all the samples. Notably gains of 1q, 8q, 16p, 20q, 21q and losses of 8p, 13q, 16q. Most of these were also observed in the previous chapter, and as mentioned in the discussion have been previously noted as features common in breast cancer.

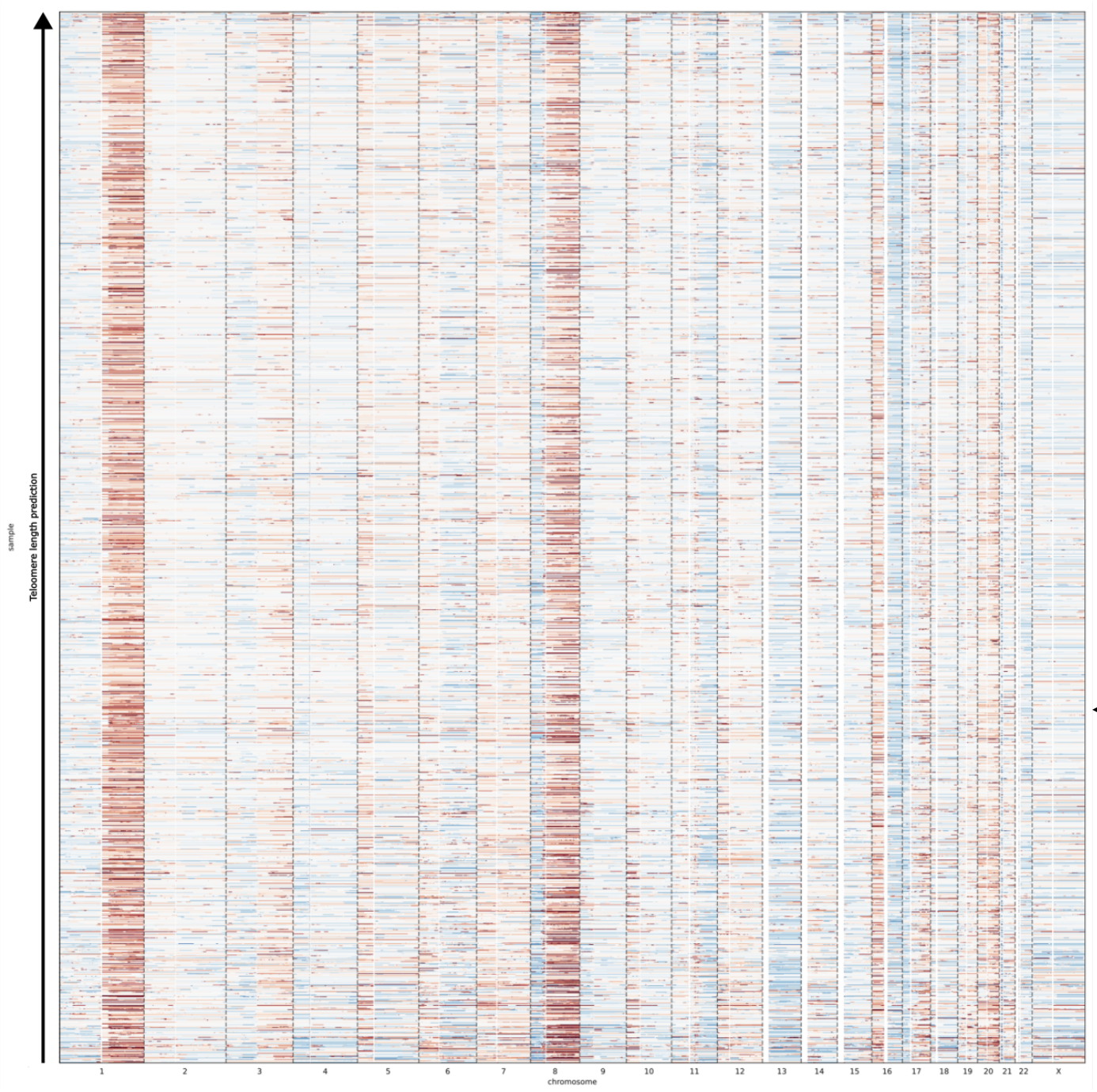


Figure 5.1 Heatmap showing relative copy number segments from pcf analysis (input of 10kb coverage windows) clipped between -1.5 (blue loss) and 1.5 (red gain) with the Y-axis displaying ordered from longest telomeres (top) to shortest (bottom). Data displayed is breast cancer (n=1591) from Genomics England. The arrow on the right is to display the telomere length threshold referenced in text where the samples appear to be more complex below this point.

This divide between the complexity profiles seen in samples above and below the telomere length threshold was also be confirmed by recursive partitioning of the complexity score (absolute sum of RCN segment values) across the whole cohort as shown in the “squiggle

plot" (figure 5.2A). As mentioned previously, this plot scales each chromosome to the same size and uses the log of the length for each segment, and uses a higher clipping thresholds (-2/+3) alongside colour past these limits to display more information about individual (and especially smaller) gains and losses than the heatmap. However, with such a large cohort this information is difficult to distinguish. Plotted next to this is a box plot with the distribution of complexity scores for each chromosome for a sample (figure 5.2B). Once again, due to the large sample size, it is difficult to read this information due to the overlapping of green diamonds indicating the mean, but has been included for the sake of consistency with other plots. Recursive partitioning was performed by taking the telomere length ordered array of complexity scores, and performing a Mann Whitney U test for shorter greater than longer and plotting the resulting p-values (figure 5.2C). Whilst there was an optimal split shown at 3.62kb, the recursive partition plot shows a significant peak that extends upwards towards the visible partition (as indicated by the arrow) that can be drawn from the heat plot (figure 5.2C). The correlation between the complexity score and telomere length (telseq predicted) was also reflected in the spearman's rank correlation coefficient albeit small at -0.16 being significant with a p-value of $5x10^{-10}$.

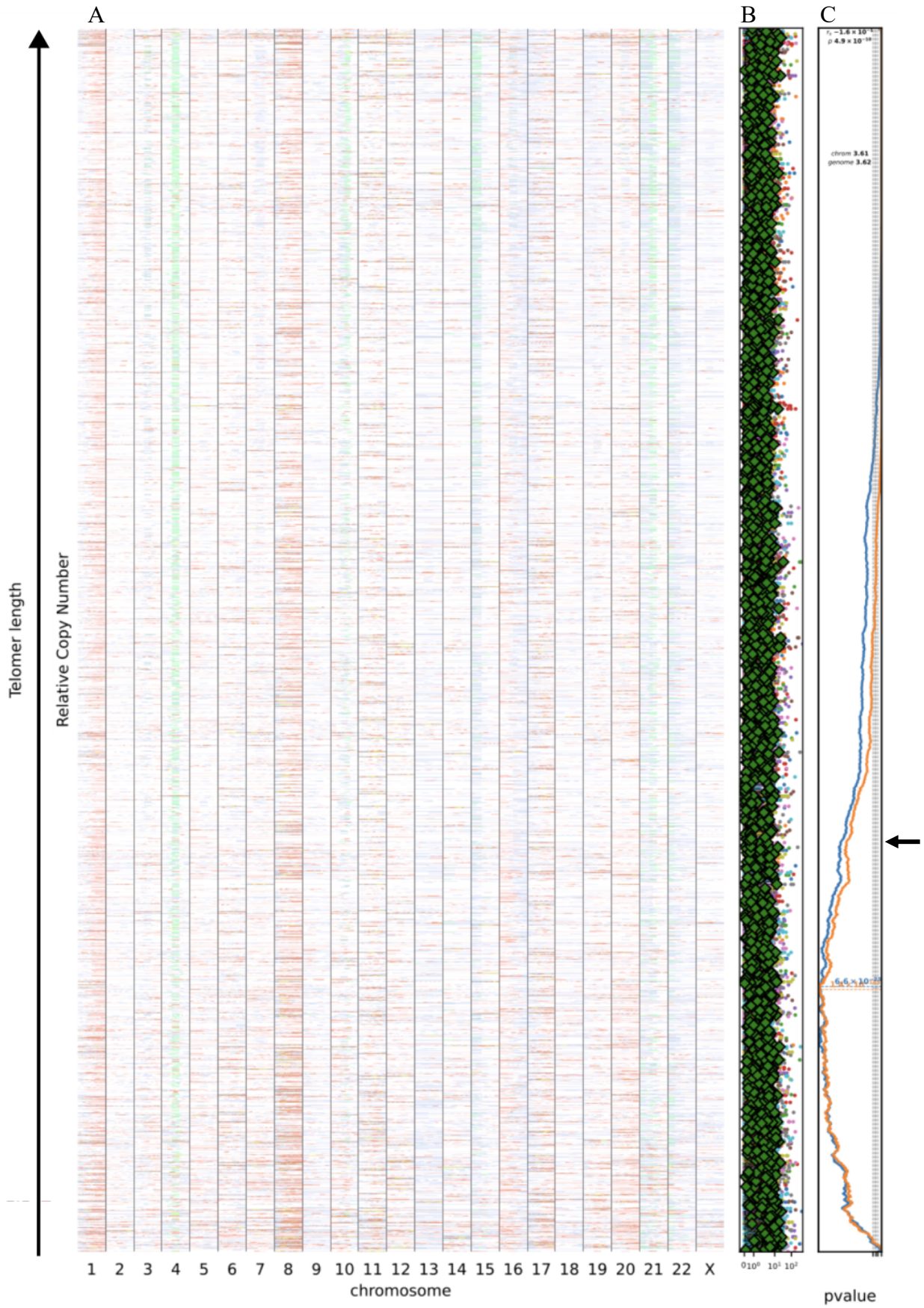


Figure 5.2. The coverage graph (A) shows the relative copy number (RCN) segments scaled so each chromosome is uniformly sized, with the length of each segment on the x-axis being logged. The complexity score boxplots (B) show the distribution of the complexity scores across all chromosomes for each sample, with means indicated by the green diamond. The recursive partition line graph (C) shows the smallest p-value for the Mann Whitney U test for x > y is $6.6x10^{-25}$ for the sum (genome blue at 3.62kb) and $1.4x10^{-145}$ individual chromosome (orange) at 3.61kb. The spearman's rank corelation coefficient is -0.16 with a p-value of $5x10^{-10}$. The arrow on the right is to display the telomere length threshold referenced in text where the samples appear to be more complex below this point. Data displayed is GEL breast cancer cohort (n=1591).

The difference in complexity scores (absolute sum for y-values of all segments) between shorter and longer partitions was once again be highlighted in Figure 5.3 where the threshold for separation between long and short telomeres was 3.81kb the telomere length threshold previously established by detection of fusions to significantly predict prognosis in several tumour types (Lin et al. 2014). This threshold was close to the optimal splits (3.61kb chromosome, 3.62kb genome) as defined by recursive partitioning (figure 5.2C) but still show substantially significant differences between "long" and "short" categorised (above and below telomere length threshold respectively) sub cohorts. Compared to the optimal threshold, it shows p-values for Mann Whitney U tests short greater than long of $8.7x10^{-100}$ (optimal $1.4x10^{-145}$) for chromosomes and $6x10^{-18}$ (optimal $6.6x10^{-25}$) for genome complexity scores.

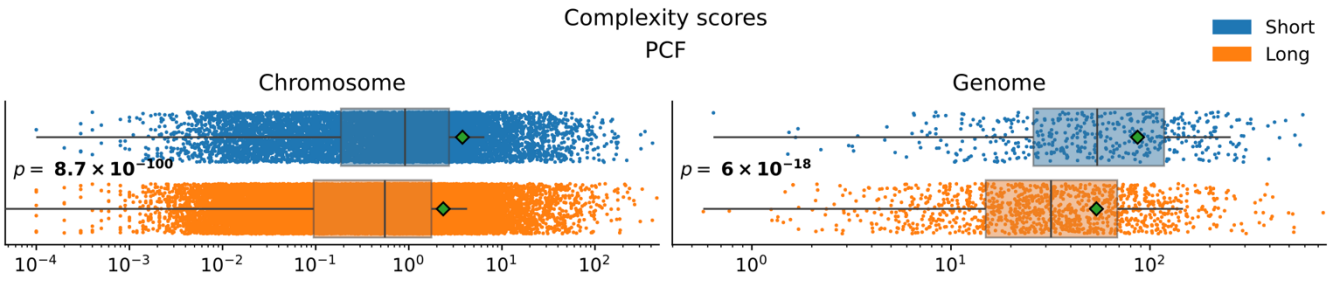


Figure 5.3. Box plot displaying the PCF based complexity scores for both chromosomes and genomes (absolute sum of chromosomes) for samples above (long) and below (short) the 3.81kb telomere length threshold. The PCF complexity score is calculated as the sum of RCN for all segments across each chromosomes. Data displayed is GEL breast cancer cohort (n=1591).

Further evidence for there being a significant difference in the RCN profile of shorter telomere length breast cancer compared to longer telomere length was shown in the recursive partitioning in the counts of gains and losses (figure 5.4). Similar optimal partitions are found when examining the gains and losses, with gains sharing a threshold of genomic complexity scores of 3.62kb (p-value $2.2x10^{-23}$) and losses sharing the chromosome complexity score of 3.61kb (p-value $6.4x10^{-10}$).

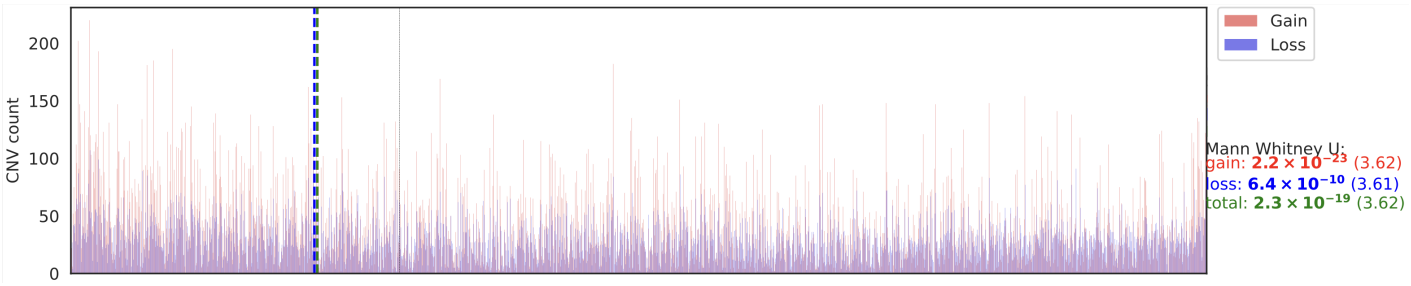


Figure 5.4. Gains and losses (n segments more or less than ±0.05) sorted by sample telomere length (ascending left to right). The Mann Whitney U (greater than) values are showing the lowest p-value from recursively partitioning from left to right of the gains, losses and totals with telomere length lowest value appears in brackets. Dotted lines indicate the telomere length of the optimal splits for gains (red which is overlapped by) total (green) and losses (blue). Data displayed is GEL breast cancer cohort (n=1591).

Whilst plotting the whole cohort is useful for providing an overview, the large scale of the data (n=1591 samples) means that much of the detail is hard to see in the heatmaps. For this reason, subsamples were taken where half the samples would be below a TL threshold and half above. Examples for one subsampled cohort is shown in the main text, with the rest being available in the appendix.

The heat map shown in Figure 5.5 shows a distinction between the RCN profiles of the top and bottom halves (top above, bottom below 3.81kb indicated by arrow). In general, this distinction could be summarised as the top half containing longer and lighter (less severe) CN segments compared to the bottom half. Conversely, samples below the TL threshold exhibit more frequent and severe sporadic gains and losses throughout the genome. As previously shown the commonalities between the two halves of gains of 1q, 8q, 16p, 20q, 21q and losses of 8p, 13q, 16q as shown above in Figure 5.1 are also present.

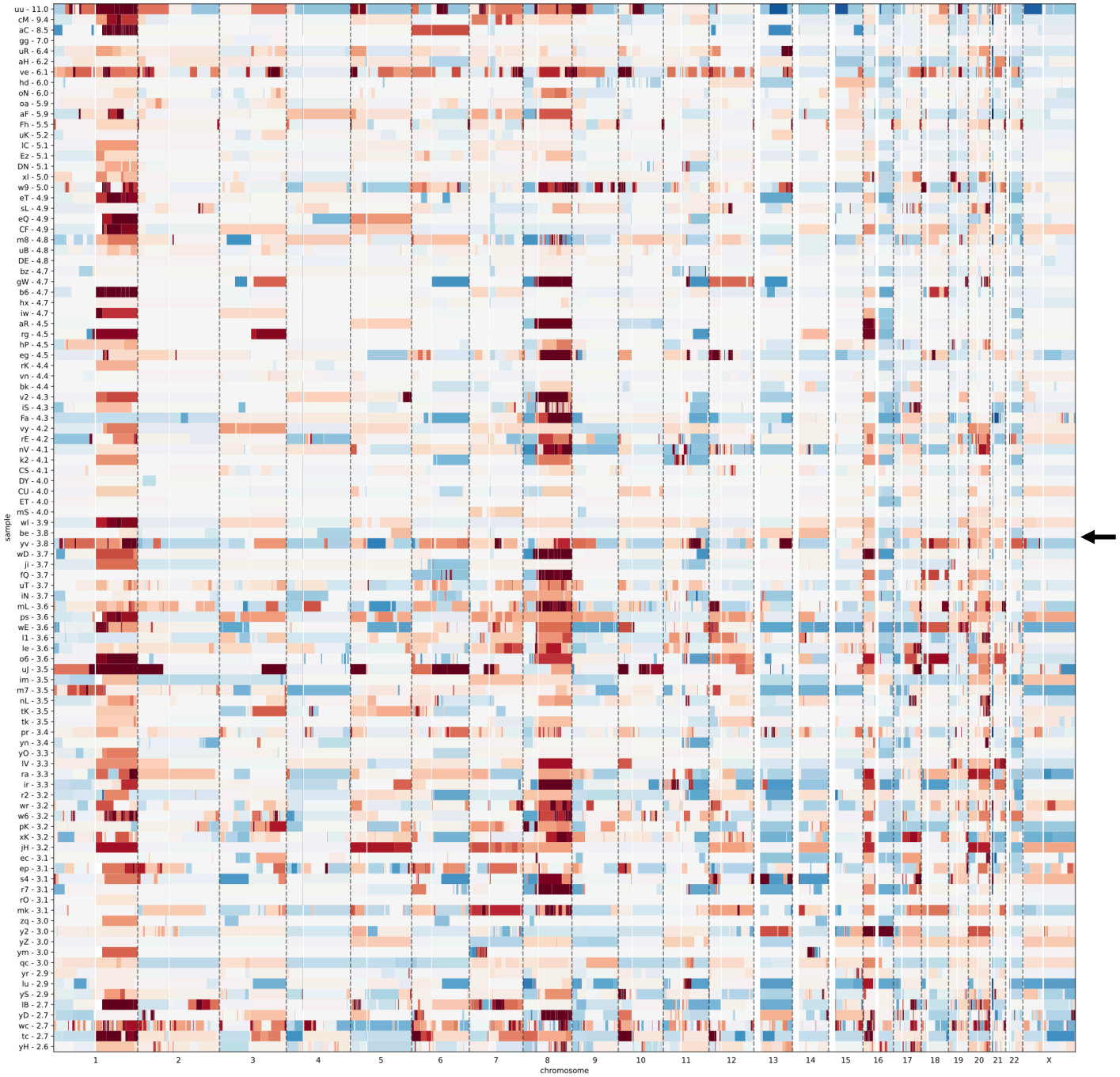


Figure 5.5. Heatmap showing relative copy number segments from pcf analysis (input of 10kb coverage windows) clipped between -1.5 (blue) and 1.5 (red) with the Y-axis displaying ordered from longest predicted (with telseq) telomeres (top) to shortest (bottom). Data displayed is a random subset of 100 samples from the Genomics England breast cancer cohort, 50 selected from 3.81kb or below, and 50 from above 3.81kb (halfway point indicated by arrow).

This visual distinction was consistent with the recursive partitioning plot shown in Figure 5.6C. It shows that variance of RCN changes was higher in smaller length telomeres. This is highlighted in the surrounding plots showing significant differences in the complexity score distribution recursively partitioning from shortest to longest. Whilst the optimal split for chromosomal complexity score was drawn at 3.67kb ($p=1.9x10^{-30}$), there was a secondary dip that occurs at 3.8kb before the Mann Whitney U p-value (for scores below the line being greater than the scores above) starts increasing. The dip also occurs for genome complexity scores at the same place, however due to a small cluster of samples with short telomeres being complex at the shortest end (2.6-2.9kb) the optimal split for this metric was placed at 2.85kb ($p=6.6x10^{-5}$).

The results of the copy number analysis within this breast cancer cohort closely resemble that shown in the previous chapter. The 3.81kb threshold used in the previous analysis in the chapter 4 cohort (with STELA measured telomere length) was also able to significantly stratify this cohort (telseq predicted telomere length) into more and less complex samples with respect to their relative copy number profiles. It is worth noting, that whilst this was not the optimal value, it is interesting that 3.81kb was close to the optimal values (3.62kb) of segregation from recursively partitioning analysis. This was unexpected as telomere length predictions were more for providing a method to rank samples based on relative telomere repeat content.

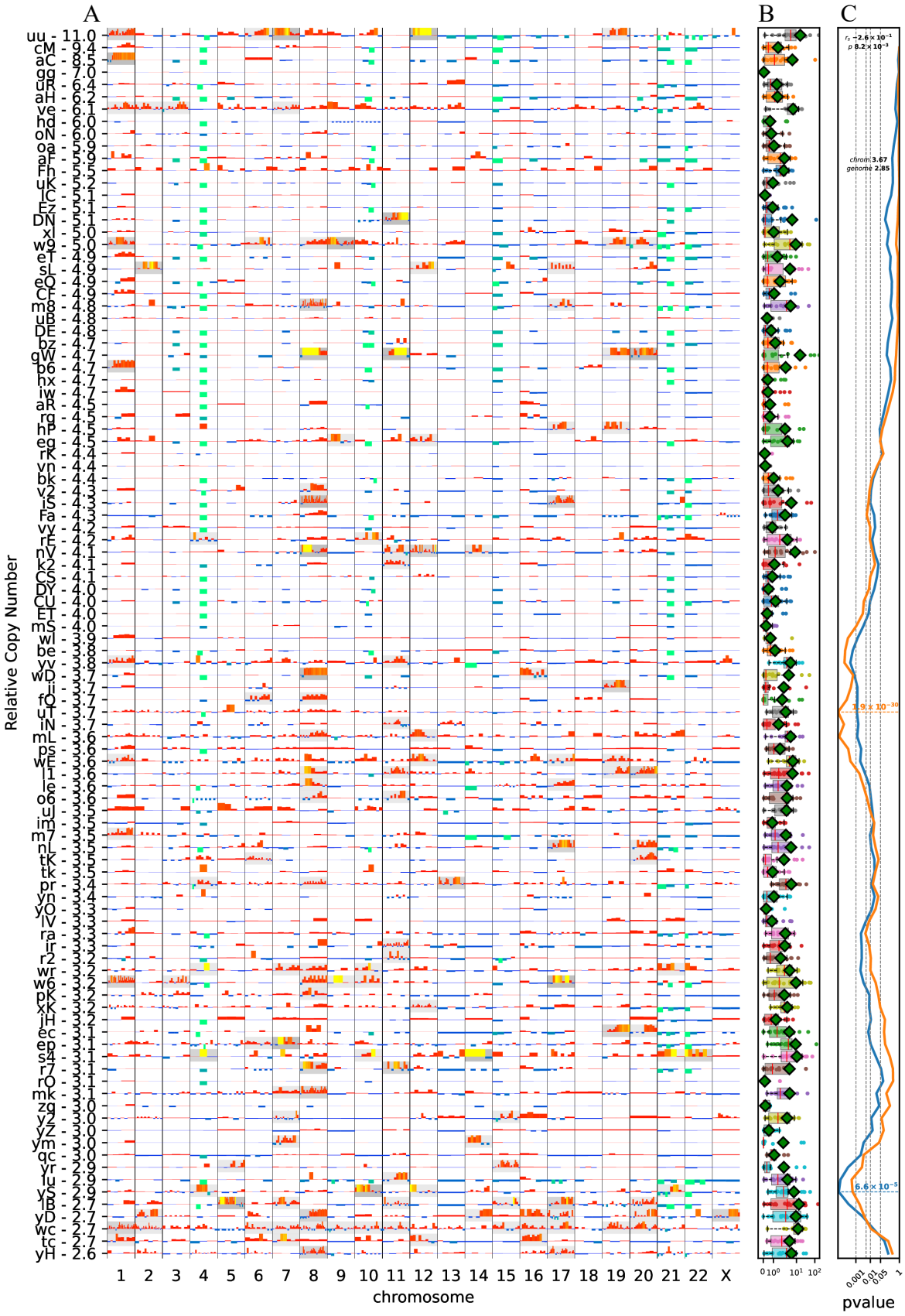


Figure 5.6. The coverage graph (A) shows the relative copy number (RCN) segments scaled so each chromosome is uniformly sized, with the length of each segment on the x-axis being logged. Red indicates gains, which is indicated by the height of the bar until +3, where values between 3 and 10 are then indicated by colour (red to yellow). Losses are indicated by the blue bars and are coloured throughout the clip threshold of -2 to aid visibility. The y-axis displays the sample name followed by telomere length in kb. The complexity score boxplots (B) show the distribution of the complexity scores across all chromosomes for each sample (with means shown as green diamonds). The recursive partition line graph (C) shows the smallest p-value for the Mann Whitney U test for x > y is $6.6x10^{-5}$ for the sum (genome at 2.85kb) and $1.9x10^{-30}$ individual chromosome at 3.67kb. The spearman's rank corelation coefficient is -0.26 with a p-value of $8.2x10^{-3}$. Data displayed is a random subset of 100 samples from the Genomics England breast cancer cohort, 50 selected from telomere length 3.81kb or below, and 50 from above 3.81kb

5.4.1.2 International Cancer Genome Consortium (ICGC) breast cancer

Further validation of the observed differences in complexity of RCN profiles between samples above and below a telomere was shown in the ICGC breast cancer cohort. Telomere lengths for this cohort were estimated using teltool (method outlined in chapter 3). As this method of prediction was different to the GEL breast cancer cohort (telseq), we did not expect that thresholds would be equivalent between the cohorts including the STELA measured cohort from chapter 4. Similar to other relative copy number heat plots shown in this analysis, the ICGC breast cancer cohorts (n=80) exhibited a comparable visual partition with more complex profiles in the bottom third from sample 56c 4.6kb (indicated by arrow) (figure 5.7). Unlike other cohorts however, there was a band (indicated by purple line) containing some more complex RCN profiles (772-ea4) within the top two thirds.

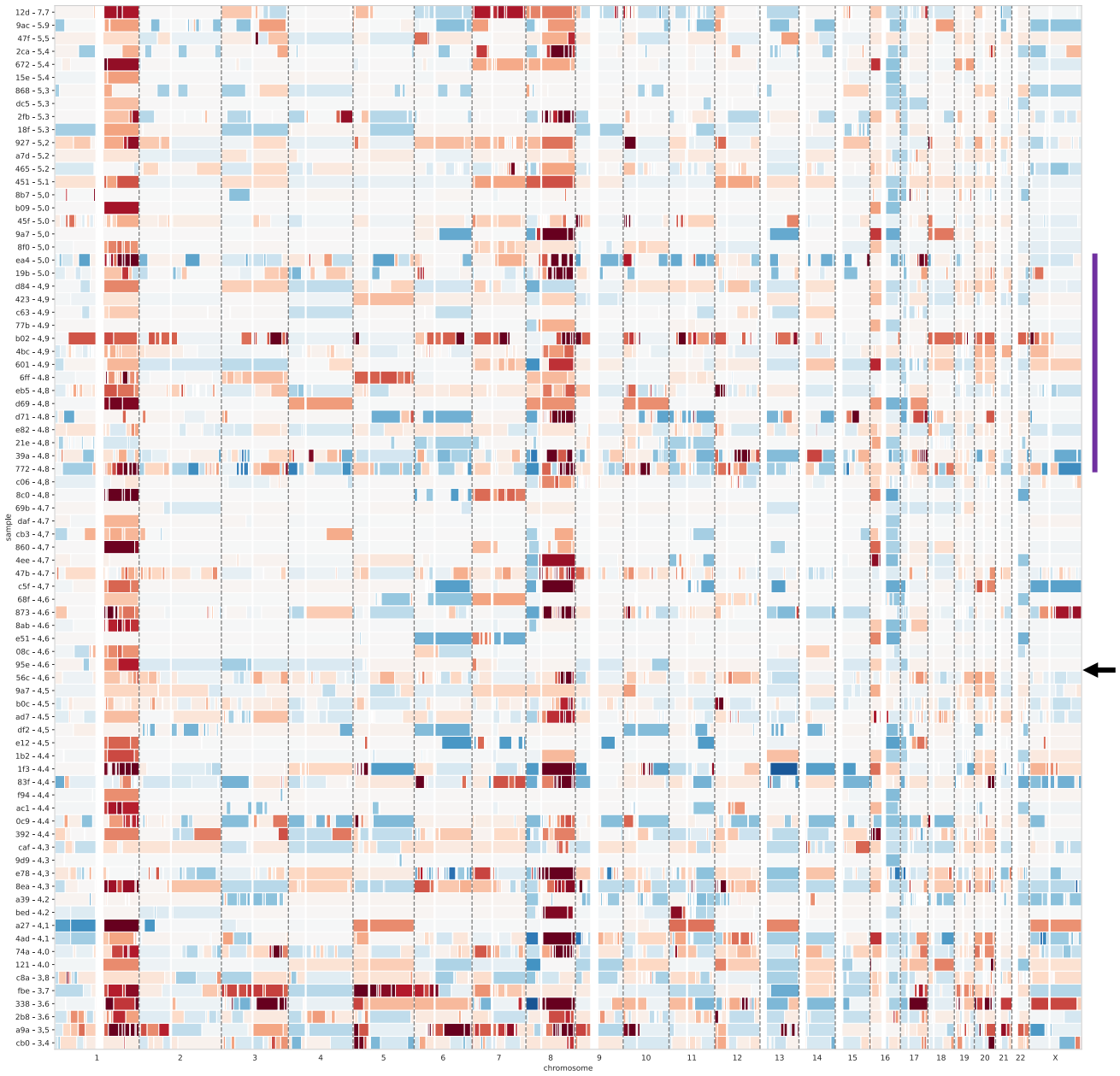


Figure 5.7. Heatmap showing relative copy number segments from pcf analysis (input of 10kb coverage windows) clipped between -1.5 (blue loss) and +1.5 (red gain) with the Y-axis displaying ordered from longest predicted (with teltool) telomeres (top) to shortest (bottom). Data displayed is the ICGC breast cancer (n=80) cohort. The arrow on the right is to display the telomere length threshold referenced in text where the samples appear to be more complex below this point. The purple line indicates a band of samples in the upper portion which appear

This visual distinction (at sample 56c 4.6kb) was confirmed by recursive partitioning of complexity scores. Firstly, the visual partition occurs close to the optimal threshold from genomic complexity score recursive partitioning at 4.38kb ($p=1.5x10^{-3}$) (figure 5.8). However, it was also far from the optimal split from recursive partitioning at 4.06kb ($p=2.5x10^{-23}$) from chromosome scores. The use of "close to" and "far from" in this case is referring to the number of samples between these two predicted values, as telomere length predictions are less important than the ranking of the samples. Despite this, there was a sharp increase in p-value after 56c (indicated by arrow) from significant values of <0.001 for chromosome and <0.01 for genome, to >0.05 for both 3 samples later. This sharp increase reinforced that this may be a more suitable telomere length to partition the cohort by genomic complexity. The secondary complex band above the threshold also caused a minor dip in significance with it being most obvious at sample b02.

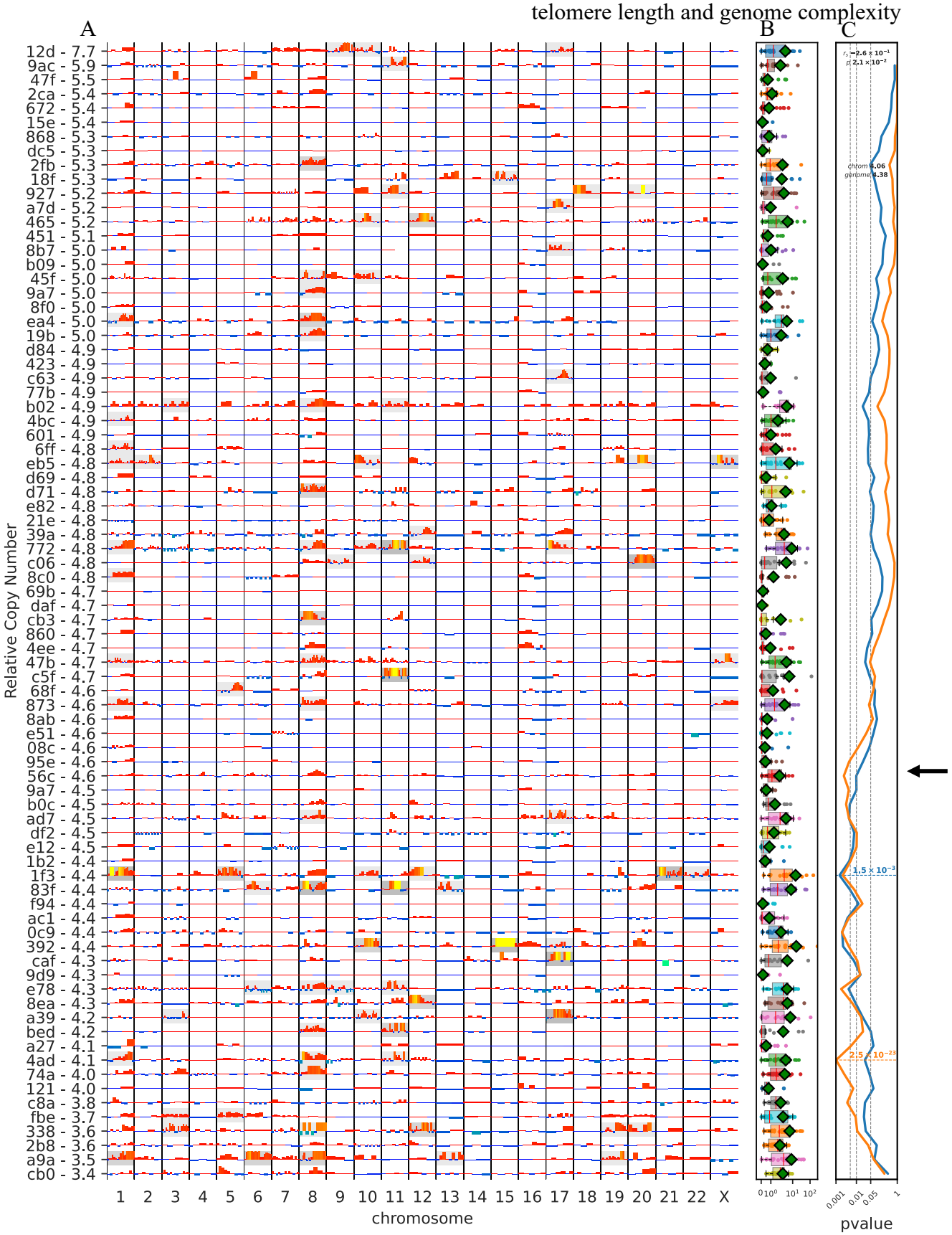


Figure 5.8. The coverage graph (A) shows the relative copy number (RCN) segments scaled so each chromosome is uniformly sized, with the length of each segment on the x-axis being logged. The complexity score boxplots (B) show the distribution of the complexity scores across all chromosomes for each sample. The recursive partition line graph (C) shows the smallest p-value for the Mann Whitney U test for x > y is $2.5x10^{-23}$ for the sum (genome blue at 4.38kb) and $1.5x10^{-3}$ individual chromosome (orange) at 4.06kb. The spearman's rank corelation coefficient is -0.26 with a p-value of $2.1x10^{-2}$. Data displayed is the ICGC breast cancer (n=80) cohort. The arrow on the right is to display the telomere length threshold referenced in text where the samples appear to be more complex below this point.

Plotting the complexity scores as a boxplot for this threshold of 4.55kb (predicted) highlighted the significance between the difference in complexity between longer and shorter telomere length patients (figure 5.9). All metrics displayed values ≤0.01 with the Mann Whitney U p-values for short greater than long being PCF based evaluations of $1.2x10^{-18}$ for chromosome, 10^{-2} for genome, and low pass scores showed p-values $8.3x10^{-20}$ for chromosome and $2.8x10^{-3}$ for genome.

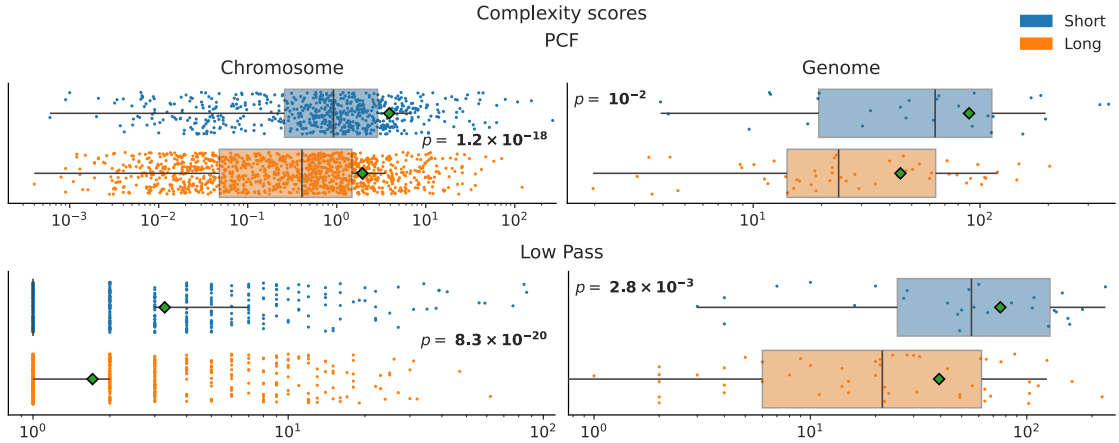


Figure 5.9. Box plot displaying the complexity scores for both chromosomes and genomes (sum of chromosomes) for both the PCF and low pass methods for calculating scores for samples above (long) and below (short) the 4.55kb telomere length threshold. The PCF complexity score is calculated as the sum of RCN for all segments across each chromosomes. The low pass complexity score is calculated as the sum of all differences between peaks and troughs of the RCN signal after being passed through a low pass filter that is greater than 0.5. Data displayed is the ICGC breast cancer (n=80) cohort.

The correlation between telomere length and genomic complexity was also reflected when looking at the raw gains and losses count. For both gains a losses there was a negative spearman's rank correlation coefficient (-0.23 and -0.29 resp.) with significant p-values (0.04 and 0.009 resp.) (figure 5.10). Once again, the "true optimal" threshold for segregation for this cohort at 4.55kb was reinforced by this being the most significant telomere length to split the cohort for number of losses with a p-value of 0.001 that the n losses below the threshold was greater than above.

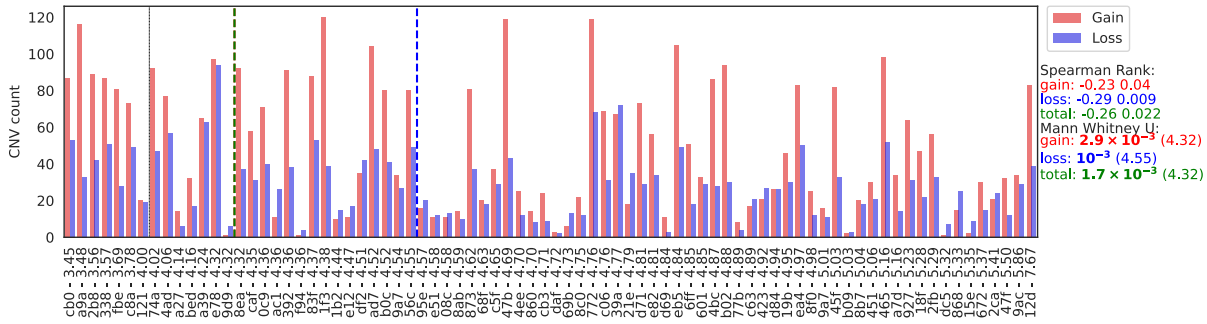


Figure 5.10. Gains and losses (n segments more or less than ±0.05) sorted by sample telomere length (ascending left to right). The Mann Whitney U (greater than) values are showing the lowest p-value from recursively partitioning from left to right of the gains, losses and totals with telomere length lowest value appears in brackets. Data displayed is the ICGC breast cancer (n=80) cohort.

The results of the ICGC copy number analysis confer with those from both the cohort examined in chapter 4 and the GEL breast cancer cohorts. Performing recursive partitioning ordered by a prediction of telomere length was able to form statistically significant differences in the RCN related metrics of complexity. As expected, unlike the GEL breast cancer cohort, it was not possible to use the previously established threshold values due to the telomere prediction method. The next cohort however, despite being a different type of cancer does have accurate measurements of telomere lengths.

5.4.1.3 Genomics England CLL

The GEL CLL cohort (n=98) consists of samples from the ARTIC and ADMIRE clinical trials which were sequenced and had telomere lengths measured by STELA. These patients were deemed to have a poor prognosis which makes them more likely to have shorter telomeres, as the research looking at telomere fusions showed. The heatmap of the relative copy number profile for chronic lymphocytic leukaemia shows far less gains and losses compared to those seen in the breast cancer cohorts (figure 5.11). It was expected that CLL would be less complex than breast cancer as literature shows CLL exhibits far less chromothripsis than breast cancer, although perhaps not to this extent (Cortés-Ciriano et al.

2020). Visually there appears to be a difference between samples 205 (3.2kb) and below (indicated by arrow) which is within the 3.81kb range found in the telomere fusion analysis from Lin et al 2014. Notable characteristics displayed by this CLL cohort are the common gain seen in chromosome 12, and loss seen in 11q. Small bands of losses in 13q14 were also observable in the majority of samples regardless of telomere length. These are consistent with findings by others in previous research (Döhner et al. 1997; Abruzzo et al. 2018; Khalid et al. 2021).

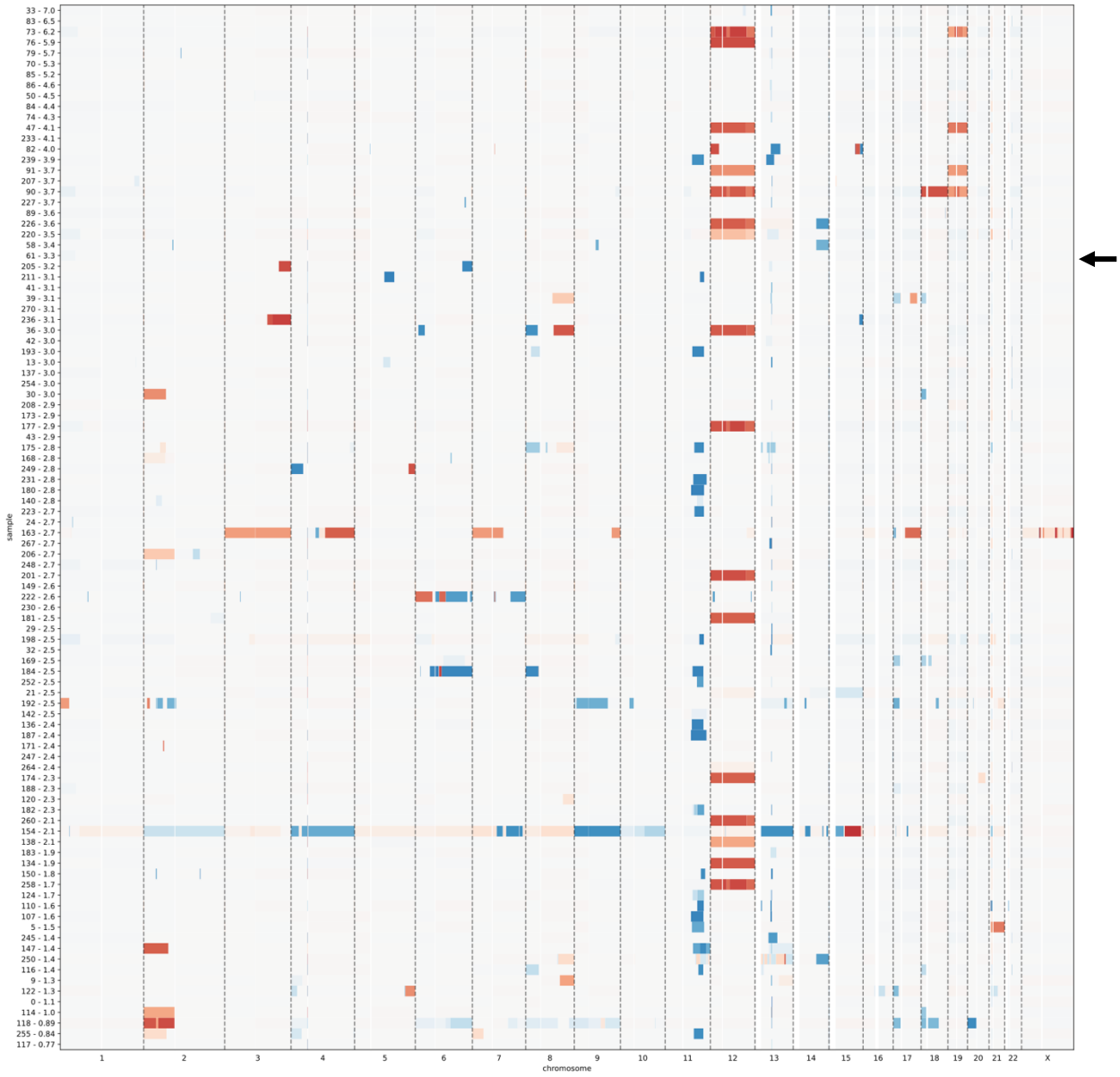


Figure 5.11. Heatmap showing relative copy number segments from pcf analysis (input of 10kb coverage windows) clipped between -1.5 (blue loss) and +1.5 (red gain) with the Y-axis displaying ordered from long telomeres (top) to shortest (bottom). Data displayed is the Genomics England chronic lymphocytic leukaemia cohort (n=98).

There was no evidence from the recursive partitioning of complexity scores to support the visual threshold seen in Figure 5.11 at 3.2kb, but it is possible this was due to the both the sparsity of gains and losses throughout the samples and the limited number of samples above this threshold (24). Despite this infrequency in CNVs, there was a significant ($p=2.7x10^{-2}$ for genome and $p=2.9x10^{-4}$ for chromosome) partition found at 2.89kb (figure 5.12). The only other point where both genome and chromosome complexity scores showed a significant stratification was at 3.7kb. Both significant thresholds are within the 3.81kb TL range where fusions were detected by Lin et al 2014.

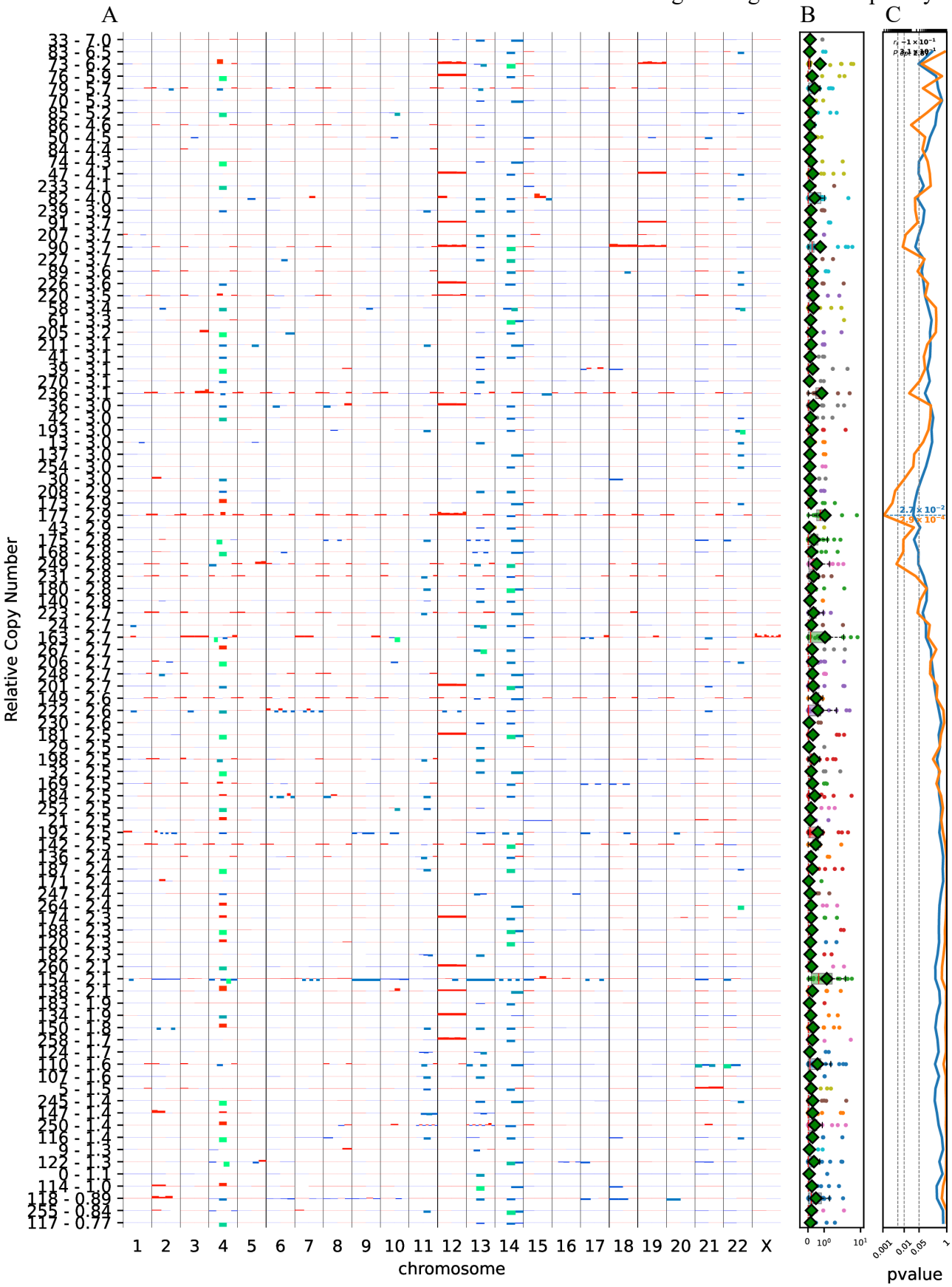


Figure 5.12. The coverage graph (A) shows the relative copy number (RCN) segments scaled so each chromosome is uniformly sized, with the length of each segment on the x-axis being logged. The complexity score boxplots (B) show the distribution of the complexity scores across all chromosomes for each sample. The recursive partition line graph (C) shows the smallest p-value for the Mann Whitney U test for x > y is 2.7x10^{-2} for the sum (genome) and 2.9x10^{-4} individual chromosome both at 2.89kb. The spearman's rank corelation coefficient is -0.1 with a p-value of 3.1x10^{-1}. Data displayed is the Genomics England chronic lymphocytic leukaemia cohort (n=98).

Comparing the complexity score distribution of the short compared to long category (threshold 2.89kb), the shorter telomere samples had a significantly higher complexity score for both chromosome ($2.9x10^{-4}$) and genome ($2.7x10^{-2}$) (figure 5.13). Notably whilst this characteristic was shared for the breast cancer cohort, overall the complexity scores for the CLL cohort were less than those seen in breast cancer (e.g. chromosome max <10 CLL and >200 breast cancer).

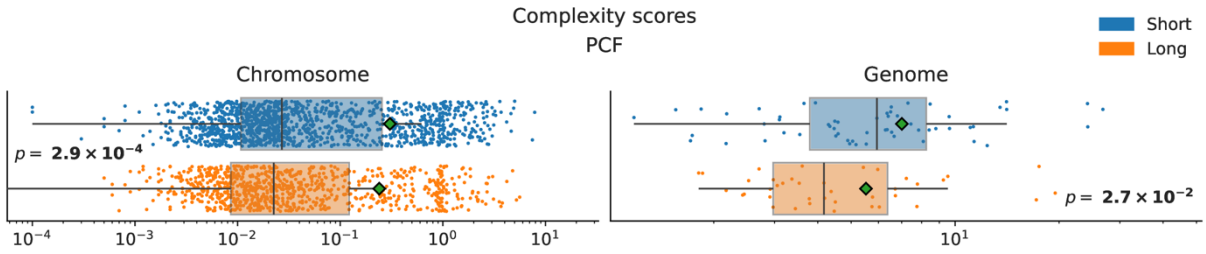


Figure 5.13. Box plot displaying the PCF based complexity scores for both chromosomes and genomes (sum of chromosomes) for samples above (long) and below (short) the 2.89kb telomere length threshold. The PCF complexity score is calculated as the absolute sum of RCN for all segments across each chromosomes. Data displayed is the GEL CLL cohort (n=98).

Plotting the raw counts instead of the complexity scores, revealed a predominance of losses in the shorter category (TL ≤ 2.87kb) of samples (figure 5.14). Whilst not significant (p=0.2) the number of losses showed a negative correlation (-0.12) with telomeres. The number of gains on the other hand show no correlation (0.02 p=0.88), with no partition where the cohort can be separated with shorter telomere samples having a significant increase in gains. However, when combined with losses, there was a partition where the total CNVs was significant at 2.87kb with a p-value of 0.043. This corresponds with the same sample (177) that the partitioning by complexity score was optimal for both chromosome and genome.

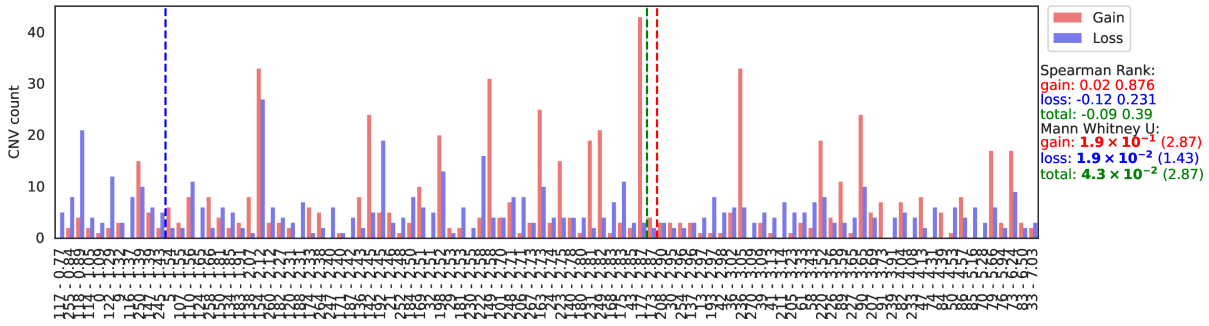


Figure 5.14. Gains and losses (n segments more or less than ±0.05) sorted by sample telomere length (ascending left to right). The Mann Whitney U (greater than) values are showing the lowest p-value from recursively partitioning from left to right of the gains, losses and totals with telomere length lowest value appears in brackets. Data displayed is the GEL CLL cohort (n=98).

Despite an overall less complex genome compared to breast cancer, CLL still exhibited significant ($p < 0.05$) differences in complexity score and counts of losses and total CNVs when partitioning by telomere length (2.89kb).

5.4.2 Structural variants

Copy number analysis provided evidence and values for telomere length thresholds where each cohort can significantly be divided by metrics of complexity. The next step of the analysis was to test if these thresholds extended to structural variants and whether any type was affected more than others. As outlined in the methods chapter and above, structural variants were called and filtered using dysgu. Filtering was performed against the match normal for both GEL cohorts, and on pools of trimmed normal bam files for the ICGC cohort.

5.4.2.1 GEL breast cancer

The counts for each structural variant type and total (excluding insertions due to limitations from short read data and resulting inability to filter by size) for SVs >10kb in the GEL breast cancer cohort (n=1591) separated into short (TL ≤ 3.81kb) and long (TL > 3.81kb) subgroups were plotted as a box plot (figure 5.15). When comparing the counts of each type

of structural variants (apart from insertions due to limitations from short read data and resulting inability to filter by size), the shorter (TL ≤ 3.81kb) subset exhibited significantly more (Mann Whitney U p-value < 0.001) SVs than the longer (TL > 3.81kb) subset. Whilst there were outliers within the >3.81kb subset whereby a few samples have a higher number of deletions, duplications, and translocations than the maximum ≤3.81kb sample, overall, it was clear the shorter subset generally display larger numbers of each variant type. This was clear in within all four categories of SV analysed and for the total SVs.

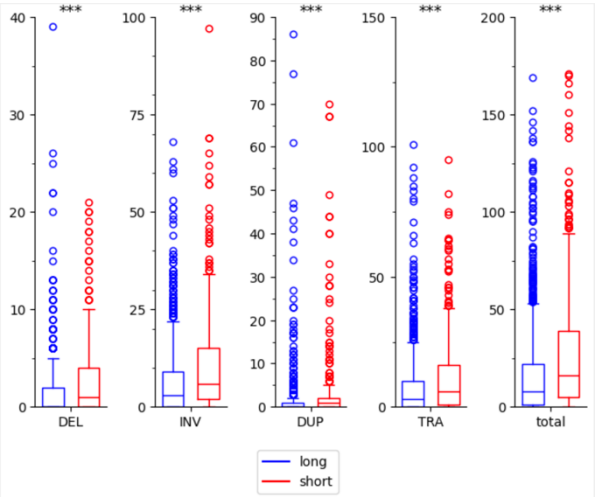


*Figure 5.15. Box plot showing the count of deletions, inversions, duplications, translocations, and total of all listed types, separated by long (>3.81kb blue) and short (≤3.81kb red) telomeres. Box displays the interquartile range, with the 25th percentile (Q1), median, and 75th percentile (Q3) in ascending order. Whiskers show the values for Q1-1.5xIQR and Q3+1.5xIQR, with outliers shown as circles. Significance of Mann Whitney U short > long displayed by * above, with 1 < 0.05, 2 < 0.01, 3 < 0.0001. Data displayed is for the Genomics England breast cancer cohort. Structural variants have been filtered to be >10kb in size. Data displayed is GEL breast cancer cohort (n=1591).*

Whilst box plots are useful for condensing the information regarding counts, they are only capable of displaying this one aspect. Circos plots can convey additional information regarding the loci of these SVs for each sample. Using the method outlined in chapter 2, circos plots were generated for each sample and are then arranged in order of telomere length. Due to the large number of samples in the GEL breast cancer cohort, it was not possible (nor practical) to plot all circos plots for all samples in the same figure. For this reason,

subsampling of the long and short categories were taken. Figure 5.16 shows one of these subsamples, with shorter samples (TL ≤ 3.81kb) above the red line and longer (TL > 3.81kb) below. Complex sets of SVs can be seen on both sides with a slight increase in their frequency seen above the line

One subjective way of approximating the appearances of samples within this plot is dense, complex/clustered, and sparse. The number of "sparse" samples appears much higher in the longer half (~41) of the plot compared to the shorter half (~31). Generally, the "clustered" samples in the shorter half appear to contain more SVs than their counter parts in the longer half. Relatively "dense" samples are also far more frequent in the shorter telomere (~16) length subset compared to the long subset (~6).

An uncommon characteristic displayed on both sides were samples presented with abundant and ubiquitous inversions. It is possible these are as the result of the subsampling of the bam files before being passed to the structural variant caller, or perhaps the one size fits all SV filtering parameters.

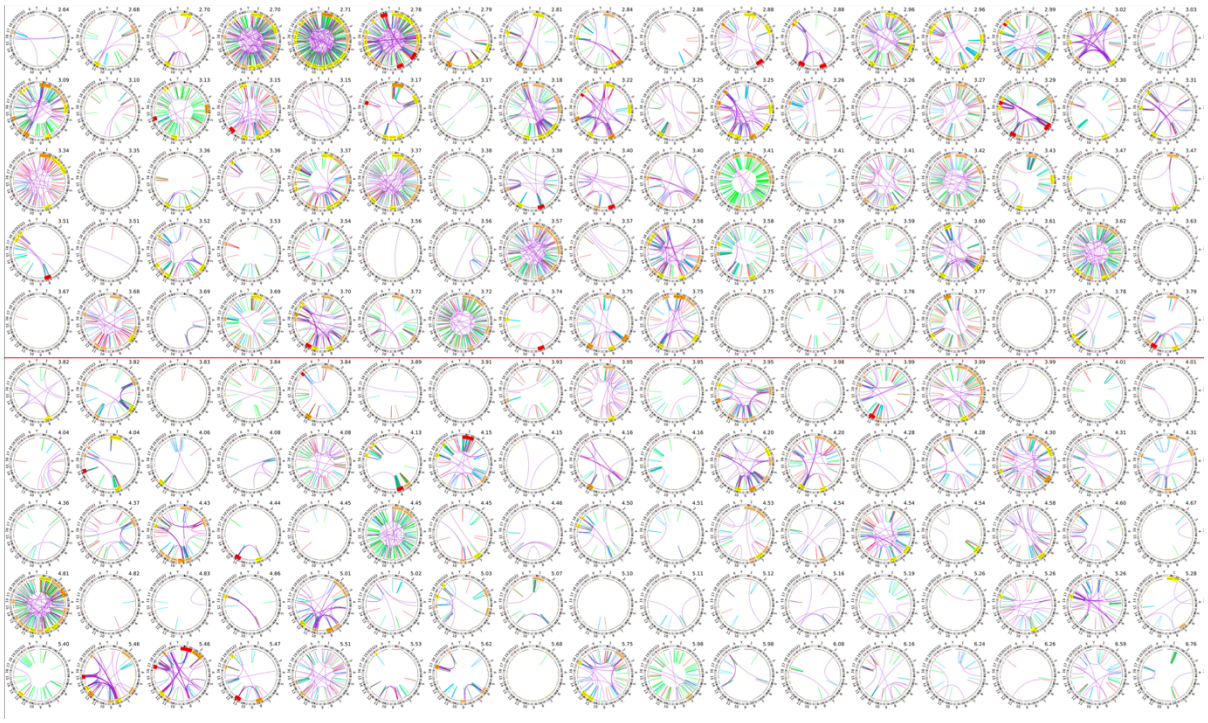


Figure 5.16. Circos plots showing structural variants by - purple: translocation, red: deletions, yellow dots: insertions, green: inversions, blue: duplications. Copy number segments are plotted around the circumference with highlighting showing chromosomes that reach different complexity score thresholds 10-30 yellow, 30-50 orange, and > 50 red. Chromosomes with more than 8 segments are also highlighted in light orange. SVs are filtered to be >10kb for deletions, inversions, and duplications. Data displayed is a random subset (n=170) from the Genomics England breast cancer cohort.

5.4.2.2 ICGC breast cancer

The smaller ICGC breast cancer cohort (n = 77) displayed a different profile to the GEL cohort. From figure 5.17, there was a trend where the shorter subgroup of samples displayed higher numbers of deletions, inversions, and translocations with some (4-5) anomalous samples in the longer category having more SVs. Despite the anomalies, for the deletion, inversion, and translocation SV types (and total) there was still a significant ($p < 0.05$) increase in SV numbers.

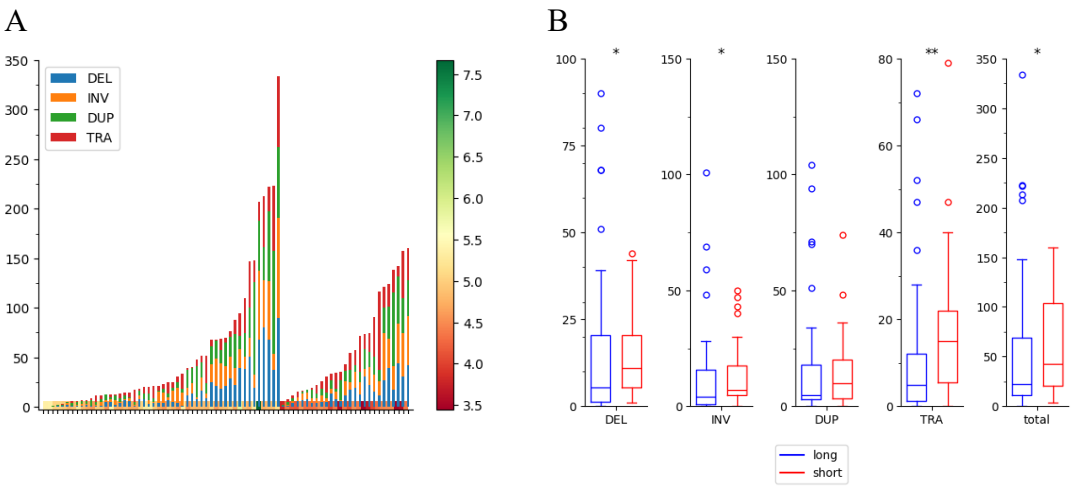


*Figure 5.17. Stacked bar plot displaying the number of each type of structural variant, with the gradient bar showing the telomere length of sample in kb (A). Box plot showing the distributions of each SV type in the short (≤3.81kb TL) and long (>3.81kb TL) groups with * displaying the 0.05, 0.01 and 0.001 confidence level of Mann Whitney U short greater than long (B). Data displayed is the ICGC breast cancer cohort (n=77).*

As a consequence of the reduced significant differences in SVs, the distinction between the longer and shorter samples was less obvious from the circos plots in Figure 5.18. Whilst less obvious looking at the plot, it is notable the "sparser" samples are more prevalent in the longer telomere length samples of the circos plot (as also shown in the left stacked bar plot of Figure 5.17).

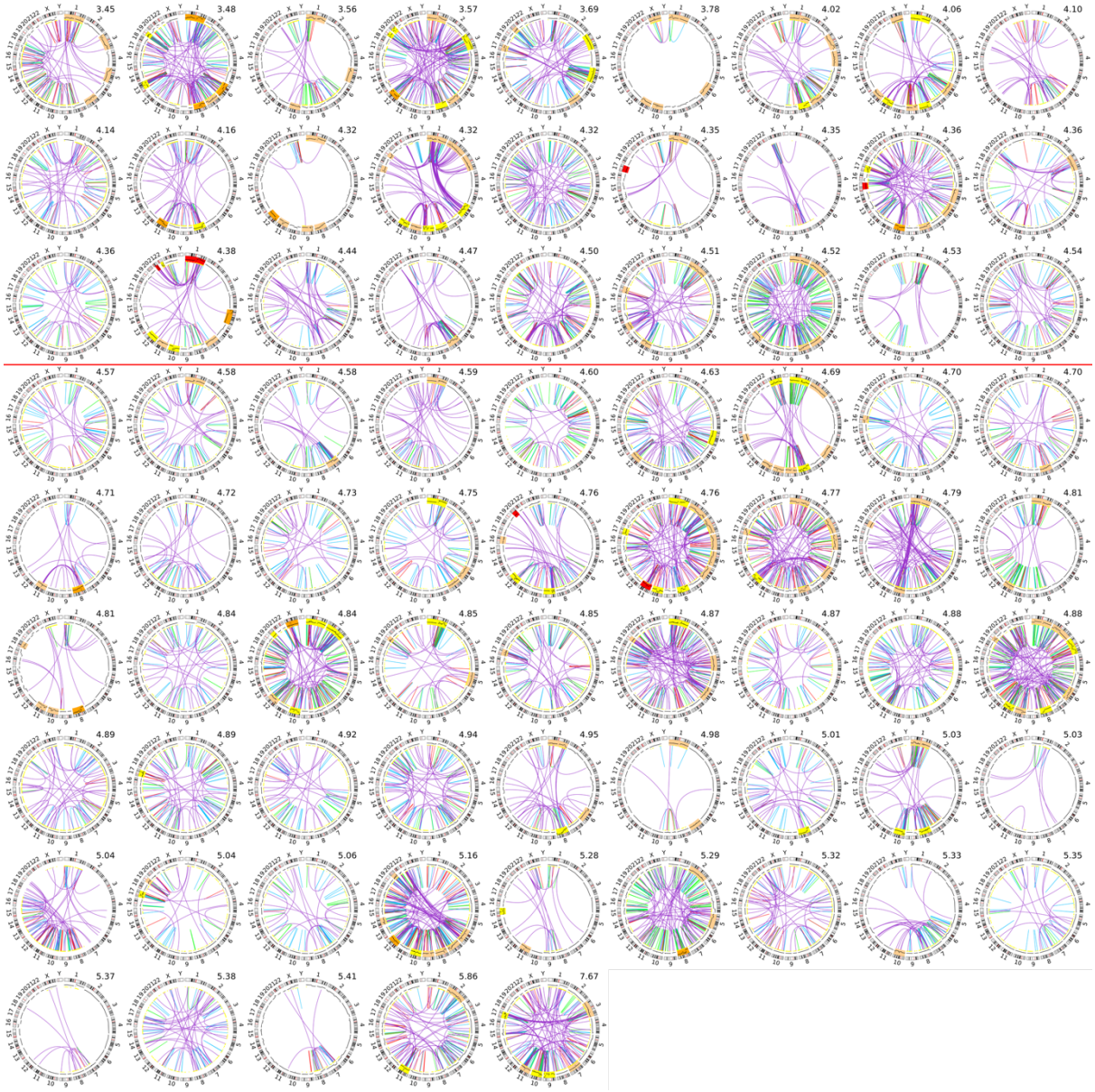


Figure 5.18. Circos plots showing structural variants by - purple: translocation, red: deletions, yellow dots: insertions, green: inversions, blue: duplications. Copy number segments are plotted around the circumference with highlighting showing chromosomes that reach different complexity score thresholds 10-30 yellow, 30-50 orange, and > 50 red. Chromosomes with more than 8 segments are also highlighted in light orange. SVs are filtered to be >10kb for deletions, inversions, and duplications. Data displayed is the ICGC breast cancer cohort (n=77).

5.4.2.3 GEL CLL

Splitting the Genomics England chronic lymphocytic leukaemia cohort (n=98) by the 2.87kb threshold previously defined by recursive partitioning of RCN complexity scores, a significant increase was only seen in the total number of SVs ($p < 0.05$) (figure 5.19B). A couple notable disparities between the GEL CLL cohort and ICGC breast cancer cohort is there are fewer of each (and total) SV type overall, and the anomalous samples with longer telomeres having more SV is missing in the CLL cohort.

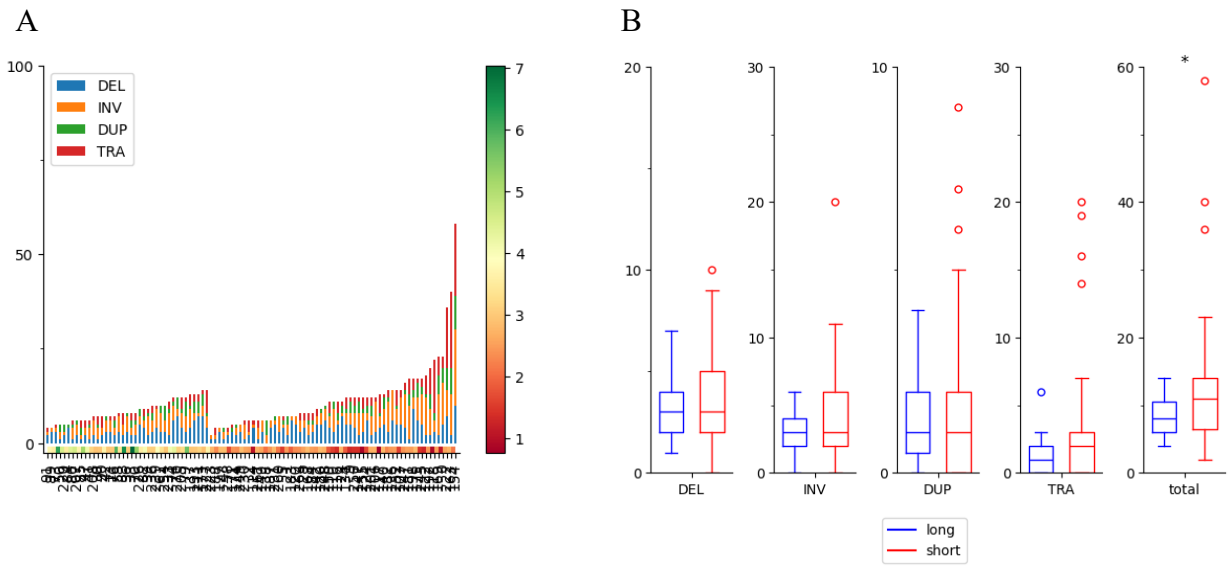


*Figure 5.19. Stacked bar plot displaying the number of each type of structural variant, with the gradient bar showing the telomere length of sample in kb (left). Box plot showing the distributions of each SV type in the short (≤3.81kb TL) and long (>3.81kb TL) groups with * displaying the 0.05, 0.01 and 0.001 confidence level of Mann Whitney U short greater than long (right). Data displayed is the Genomics England chronic lymphocytic leukaemia cohort (n=98).*

However, using a threshold of 2.26kb (identified previously as prognostic in CLL from telomere fusion analysis by Lin et al 2014), a significant increase ($p < 0.05$) in deletions and translocations can be observed (as well as total SVs) (figure 5.20B).

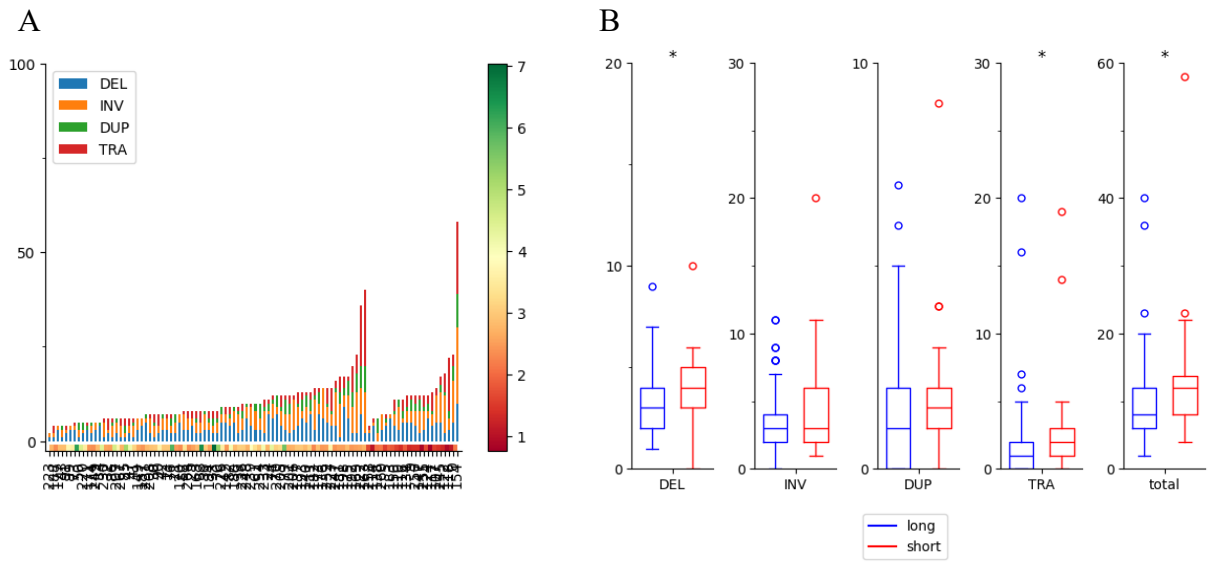


*Figure 5.20. Stacked bar plot displaying the number of each type of structural variant, with the gradient bar showing the telomere length of sample in kb (A). Box plot showing the distributions of each SV type in the short (≤2.26kb TL) and long (>2.26kb TL) groups with * displaying the 0.05, 0.01 and 0.001 confidence level of Mann Whitney U short greater than long (B). Data displayed is the Genomics England chronic lymphocytic leukaemia cohort (n=98).*

This decrease in each type and overall number of structural variants is also seen in the circos plots shown in Figure 5.21. Previously in the breast cancer samples there were very "dense" plots showing ubiquitous SVs, which are not seen in the CLL cohort.

For each cohort, the telomere length thresholds defined by the copy number analysis were able to significantly ($p < 0.05$) stratify patients by at least total structural variant count. Whilst the largest GEL breast cancer cohort showed increased frequency of all SV types ($p < 0.001$), this was not reflected by the ICGC breast cancer and GEL CLL cohorts, neither of which displaying a significant increase in duplications. Using the pre-established telomere threshold (2.26kb) in the GEL cohort also did not show an increase in inversions. Further interrogation of the SVs for each cohort was carried out in the chain linking analysis.

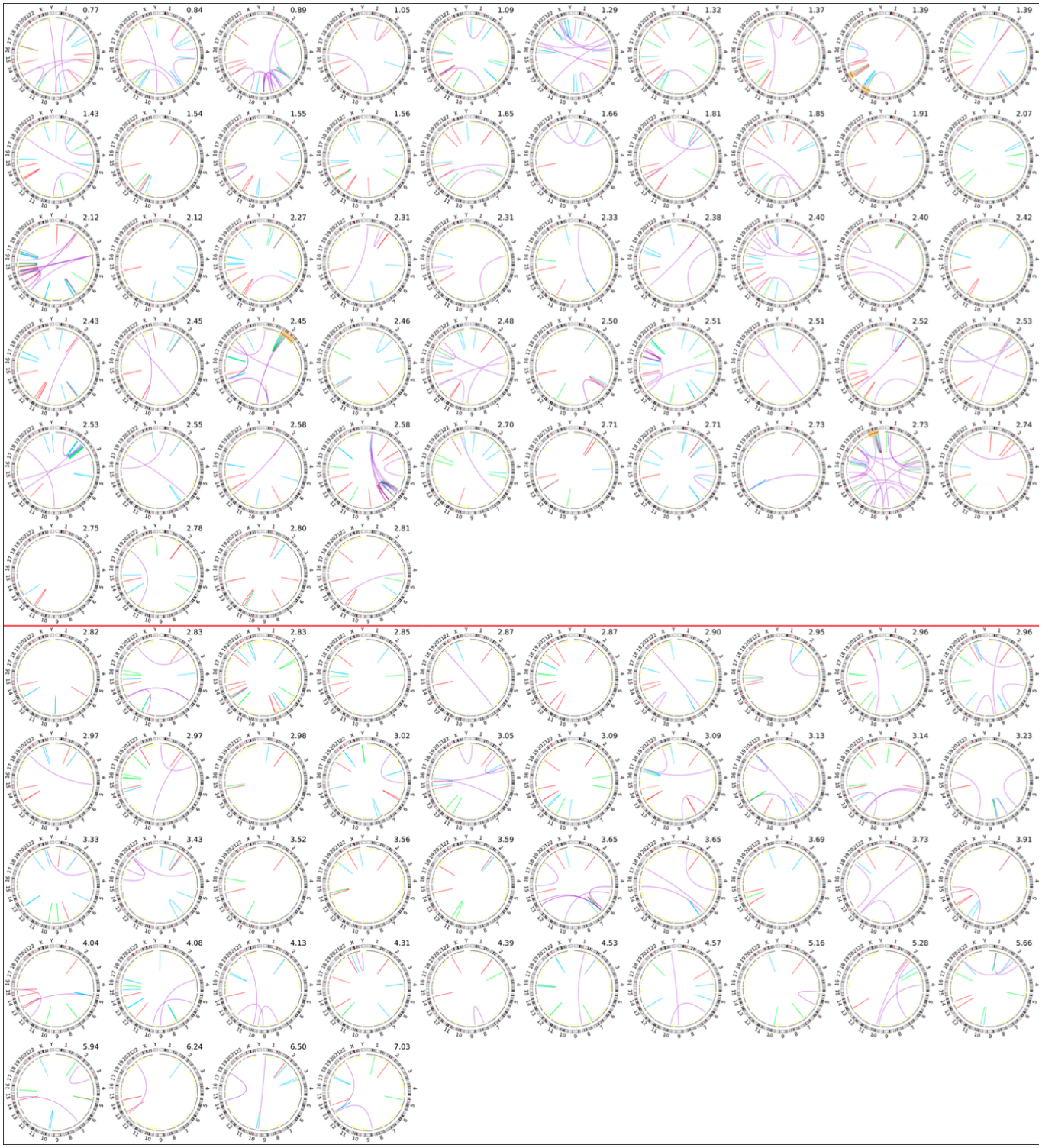


Figure 5.21. Circos plots showing structural variants by - purple: translocation, red: deletions, yellow dots: insertions, green: inversions, blue: duplications. Copy number segments are plotted around the circumference with highlighting showing chromosomes that reach different complexity score thresholds 10-30 yellow, 30-50 orange, and > 50 red. Chromosomes with more than 8 segments are also highlighted in light orange. SVs are filtered to be >10kb for deletions, inversions, and duplications. Data displayed is the Genomics England chronic lymphocytic leukaemia cohort (n=98).

5.4.3 Chain link

The aim of the chain linking analysis was to gather insight into how these structural variants are related and identify cases of chromoplexy. Through statistical testing the probability of the proximity of breakpoints, clusters of SVs likely the result of the same events can be identified. The method is based on the ChainFinder algorithm developed by Baca et al and is described in chapter 2.

5.4.3.1 GEL breast cancer

As this analysis requires a p-value threshold for the testing of proximity of SVs, it is important to establish a suitable threshold to be used. In this case, this threshold was determined through precursive analysis across a range of values to create an elbow plot. This is a heuristic method where diminishing returns are rejected in favour a smaller cost (in this instance p-value). The elbow plot in Figure 5.22 shows that for any p-value used to classify whether SVs are linked, there is a significant ($p < 0.05$) increase in linked variants for samples predicted to have telomeres ≤3.81kb. Ultimately the decision was made to use a value of 0.2 due to the elbow formed at this value.

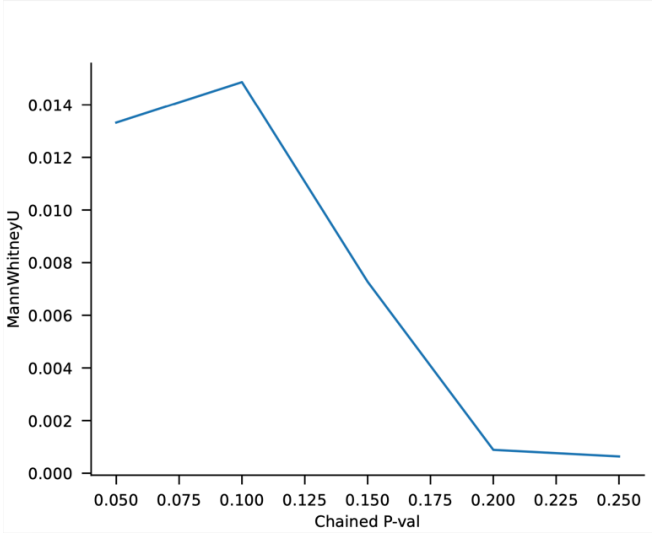


Figure 5.22. Elbow plot displaying the Mann Whitney U p-value (y-axis) for chained SV count (shorter TL predicted with telseq ≤3.81kb greater than longer > 3.81kb) by the binomial p-value (x-axis) used to test whether a cluster of SVs is closer than expected (i.e. classify cluster) ranging between 0.05-0.25. Data relating to the GEL breast cancer cohort (n=1591).

A subset of circos plots in Figure 5.23 shows (n=170) samples above and below the 3.81kb threshold shows far more chained SVs (coloured) lines for samples below the threshold. In addition to overall increased numbers, it is worth noting that the overall complexity of event appears to be increased in the tumours with shorter telomeres with circos plots showing clusters extending into a larger number of chromosomes. Chains found in the longer telomere length subset have a reduced complexity with fewer chromosomes involved.

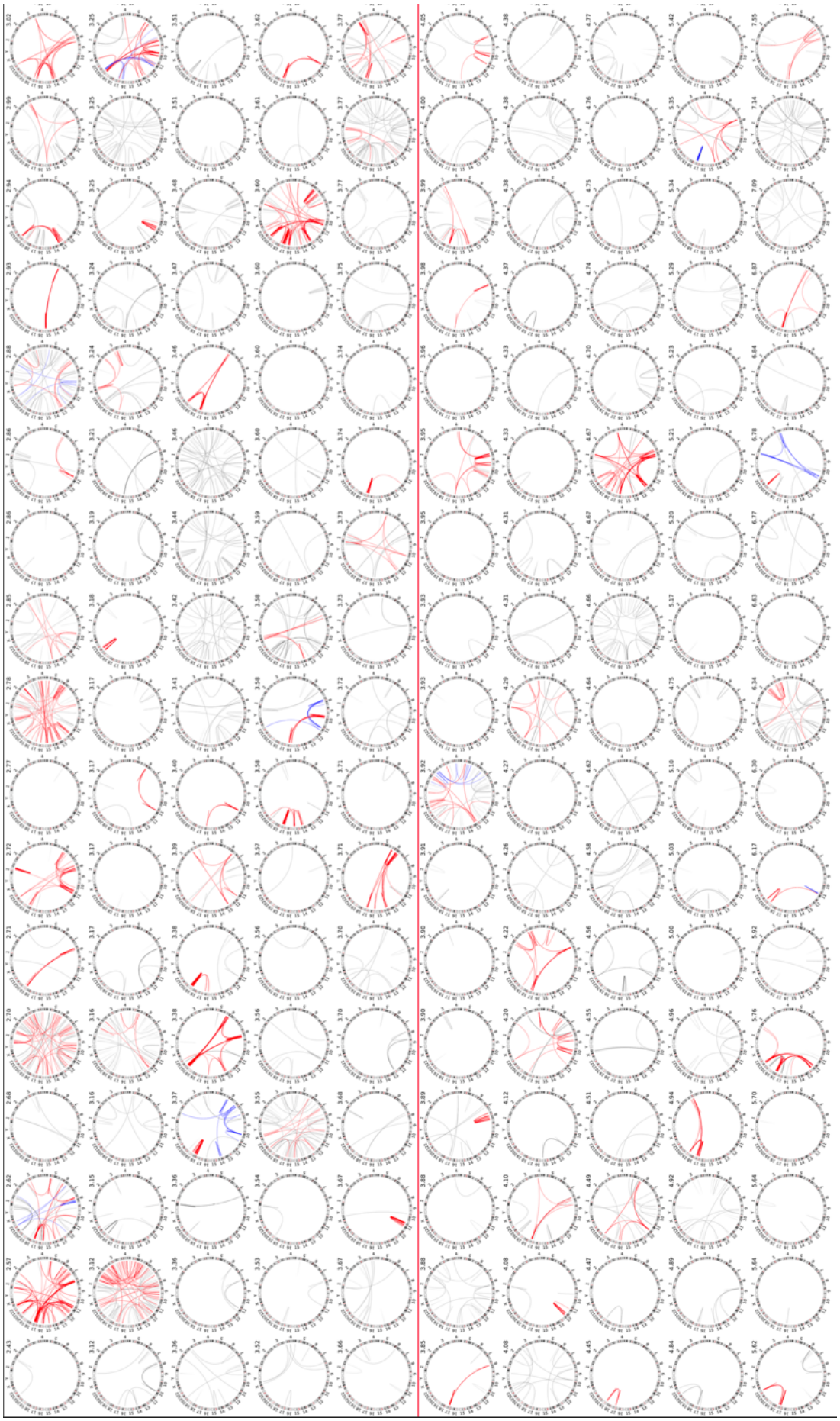


Figure 5.23. Circos plots displaying the clusters of variants in colour (red, blue, green), with non-clustered variants in grey as determined by the chain linking pipeline using a binomial p-value threshold of 0.2. Number shown top right of each circos plot is the telomere length as predicted by telseq, and samples are separated by the red line in the middle by a telseq prediction threshold of 3.81kb. Data displayed is a subsample(n=170) of the GEL breast cancer cohort (n=1591).

The overall increase in chained SVs is reflected in Figure 5.24 showing a box plot for the distribution in count across the samples above and below the 3.81kb threshold. The Mann Whitney U p-value for SV count of chained SVs for shorter samples being larger than longer is 8.9×10^{-4}. The value of 0.0 is displayed above the non-chained count boxplot as the value is less than 10^{-5} and a rounding error having occurred.

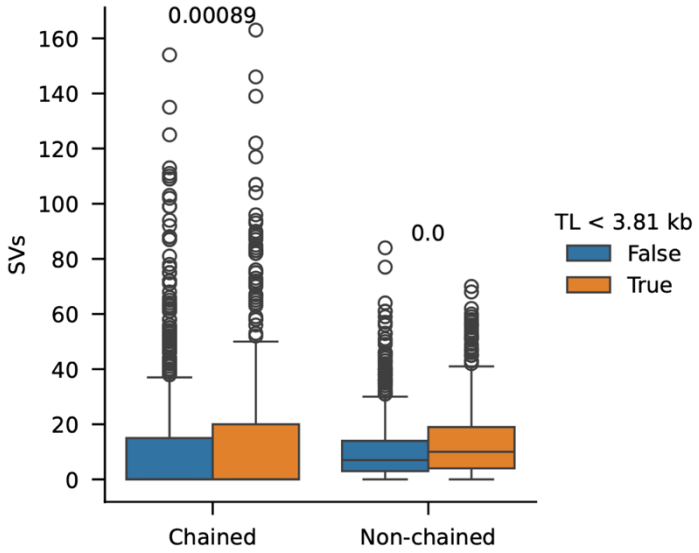


Figure 5.24. Box plot showing the number of chained and non-chained SVs for samples above and below a predicted (with telseq) telomere length threshold of 3.81kb, with the Mann Whitney U p-value for testing if shorter is greater than longer above the boxes. SV clusters determined by the chain linking pipeline using a binomial p-value threshold of 0.2. Data displayed is from the GEL breast cancer cohort (n=1591). The value of 0.0 is displayed above the non-chained count boxplot as the value is less than 10^{-5} and a rounding error having occurred.

The validity of these clusters is confirmed by the assortativity, and modularity graphs shown in Figure 5.25. As explained in the methods chapter, the algorithm constructs a graph, using breakpoints as nodes, and edges between them are created if the p-value is below a threshold for how close they are expected to be (Baca et al. 2013; Cleal et al. 2019). Assortativity can be thought of as the correlation between nodes (breakpoints) in a cluster, and modularity as the uniqueness of a cluster (group of chained SVs). Comparing the difference between the observed data and a randomly generated dataset, the probability these correlations and unique clusters are formed through chance are calculated.. Both metrics when compared to a

randomly generated set show highly significant coefficient p-values at $1.1x10^{-36}$ and $1.2x10^{-42}$ with respect to assortativity and modularity. The figure displays this information as a histogram overlayed with a KDE fit for the distributions in the values.

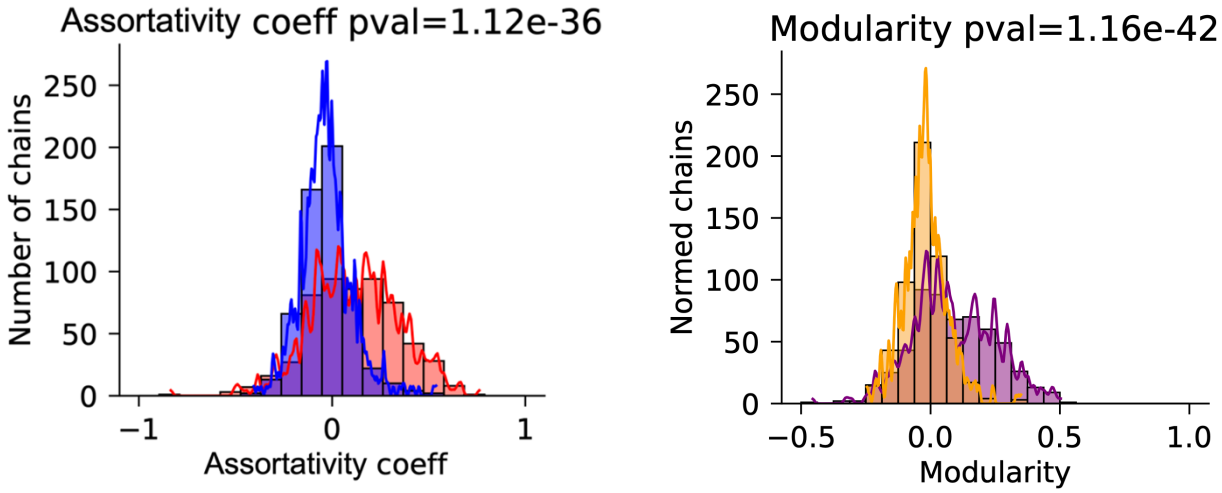


Figure 5.25. Assortativity coefficient values for random (blue) and SV clusters (red) left. Modularity of random (yellow) and SV clusters (purple). SV clusters determined using a binomial p-value threshold of 0.2. Both assortativity and modularity shown as histogram overlayed with a KDE fit for the distributions in the values. Data relates to the GEL breast cancer cohort (n=1591).

5.4.3.2 ICGC breast cancer

Elbow plots in Figure 5.26 show there is an uneven pattern when increasing the binomial p-value threshold required for cluster classification. Due to the unexpected shape of the plot from the range tested in GEL, additionally values between 0.01 and 0.05 were also evaluated for a potential elbow. Aside from a general decrease in p-value for the n clustered SVs for short greater than long, there are small increases at 0.03, 0.04, and 0.1, with a much larger increase at 0.25 (from 0.2). The value of 0.2 for binomial testing was determined as optimal as it had the lowest Mann Whitney U p-value.

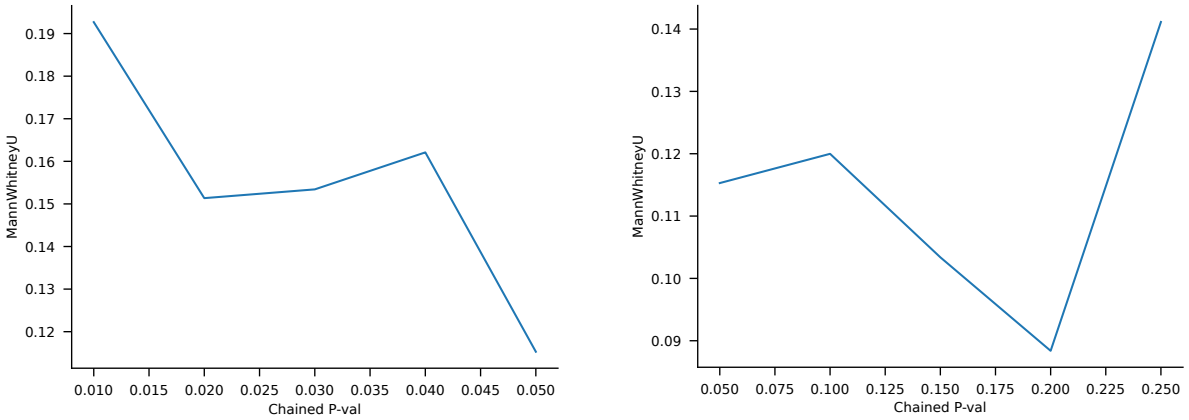


Figure 5.26. Elbow plot displaying the Mann Whitney U p-value (y-axis) for chained SV count (shorter TL predicted with teltool ≤4.45kb greater than longer > 4.45kb) by the binomial p-value (x-axis) used to test whether a cluster of SVs is closer than expected (i.e. classify cluster) ranging between 0.01-0.05 (left) and 0.05-0.25 (right). Data relating to the ICGC breast cancer cohort (n=75).

Whilst the number of chained SVs not significant (p = 0.088) between the shorter (TL ≤ 4.45kb) and longer (TL > 4.45kb) subsets, the circos plots in Figure 5.27 reveal that the frequency of samples containing chained SVs interacting with 2 or more chromosomes was higher in the shorter subset. For the shorter subgroup 16 out of 27 (59%) of samples contain chains interacting with 2 or more chromosomes compared to the 20 out of 48 (42%) for the longer classified samples. This implies that chromoplexy is more common in patients with cancers that have shorter telomeres.

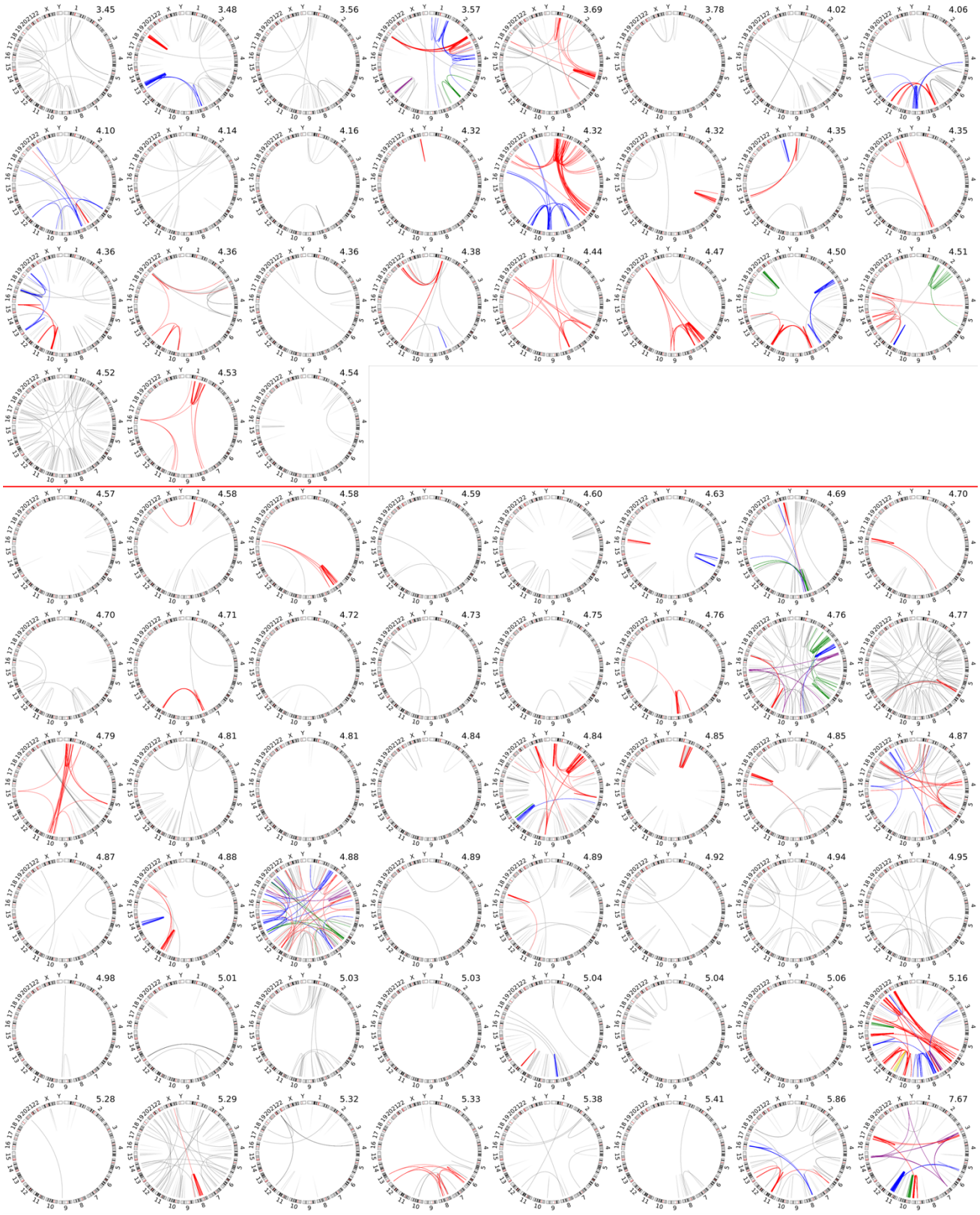


Figure 5.27. Circos plots displaying the clusters of variants in colour (red, blue, green, purple), with non-clustered variants in grey as determined by the chain linking pipeline using a binomial p-value threshold of 0.2. Number shown top right of each circos plot is the telomere length as predicted by teltool, and samples are separated by the red line in the middle by a teltool prediction threshold of 4.45kb. Data displayed relating to ICGC breast cancer cohort (n=75).

An increase in chained SVs was also apparent from the box plot (figure 5.28), the distribution of the number of chained SVs between samples above and below the predicted 4.55kb telomere length threshold is higher whilst not significant (p = 0.088). However, if the two largest outliers in samples above 4.55kb are discounted this value is then significant at 0.036 (with removing just the top outlier dropping 0.088 down to 0.058).

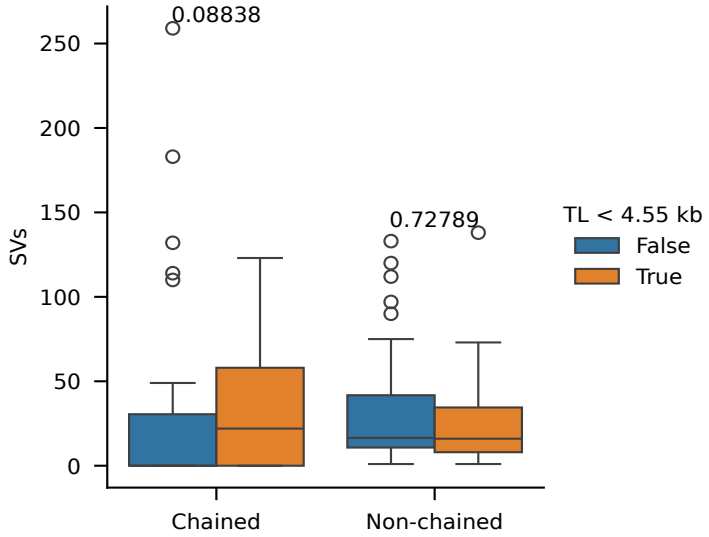


Figure 5.28. Box plot showing the number of chained and non-chained SVs for samples above and below a predicted (with teltool) telomere length threshold of 4.55kb, with the Mann Whitney U p-value for testing if shorter is greater than longer above the bars. SV clusters determined using a binomial p-value threshold of 0.2. Data displayed relating to ICGC breast cancer cohort (n=75).

Figure 5.29 confirms these groupings of SVs into chained are valid with assortativity coefficient p-value of 2.7×10^{-4}, and modularity p-value 8.3×10^{-5} when compared to a randomly generated set.

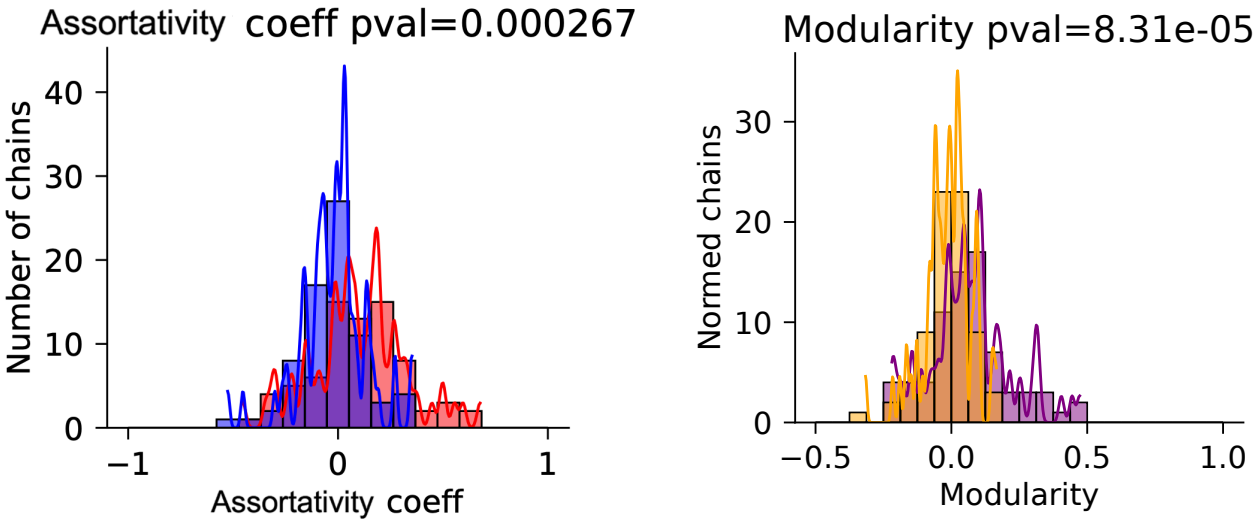


Figure 5.29. Assortativity coefficient values for random (blue) and SV clusters (red) left. Modularity of random (yellow) and SV clusters (purple). SV clusters determined using a binomial p-value threshold of 0.2. Both assortativity and modularity shown as histogram overlayed with a KDE fit for the distributions in the values. Data relates to the ICGC breast cancer cohort (n=75).

5.4.3.3 GEL CLL

The "elbow plot" shown in Figure 5.30 does not demonstrate a classic elbow pattern, this likely due to the smaller number of structural variants. The graph is striated with bands showing the same Mann Whitney U p-value for 0.05 and 0.10 of ~0.100, then decreasing to ~0.065 for binomial p-values between 0.15 to 0.25. This is not unexpected as the smaller SV count to begin with leads to the same smaller groups of SVs being classified as chained between varying the binomial classification value. Due to it being the lowest value in the band, 0.15 was used for the binomial p-value threshold.

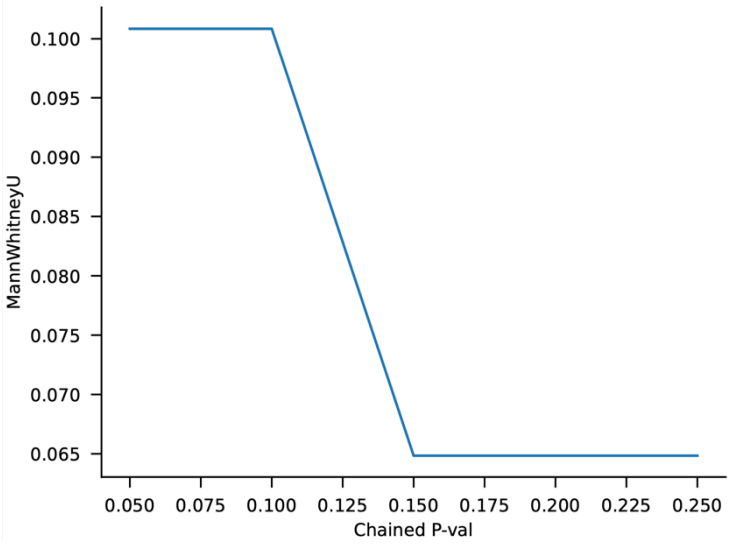


Figure 5.30. Elbow plot displaying the Mann Whitney U p-value (y-axis) for chained SV count (shorter TL measured with STELA ≤2.87kb greater than longer > 2.87kb) by the binomial p-value (x-axis) used to test whether a cluster of SVs is closer than expected (i.e. classify cluster) ranging between 0.05-0.25. Data relating to the GEL CLL cohort (n=98).

The circos plots displayed in Figure 5.31 show that only 5 samples in the cohort have chained SVs for the p-value threshold of 0.15, all five of which have telomere lengths below the 2.87kb threshold (as determined through complexity score partitioning analysis). Out of these five, only one sample (2.53kb) has a chain across 2 chromosomes. Two samples have chains connected to three chromosomes, with one (2.12kb) containing a separate chain existing solely on chromosome 15. Out of the remaining two samples with chained SVs; the shorter (0.87kb) has one cluster spanning chromosomes 2, 6, 7, 8, 9, and 11, with the longer (2.73kb) having two clusters each spanning more than 3 chromosomes.

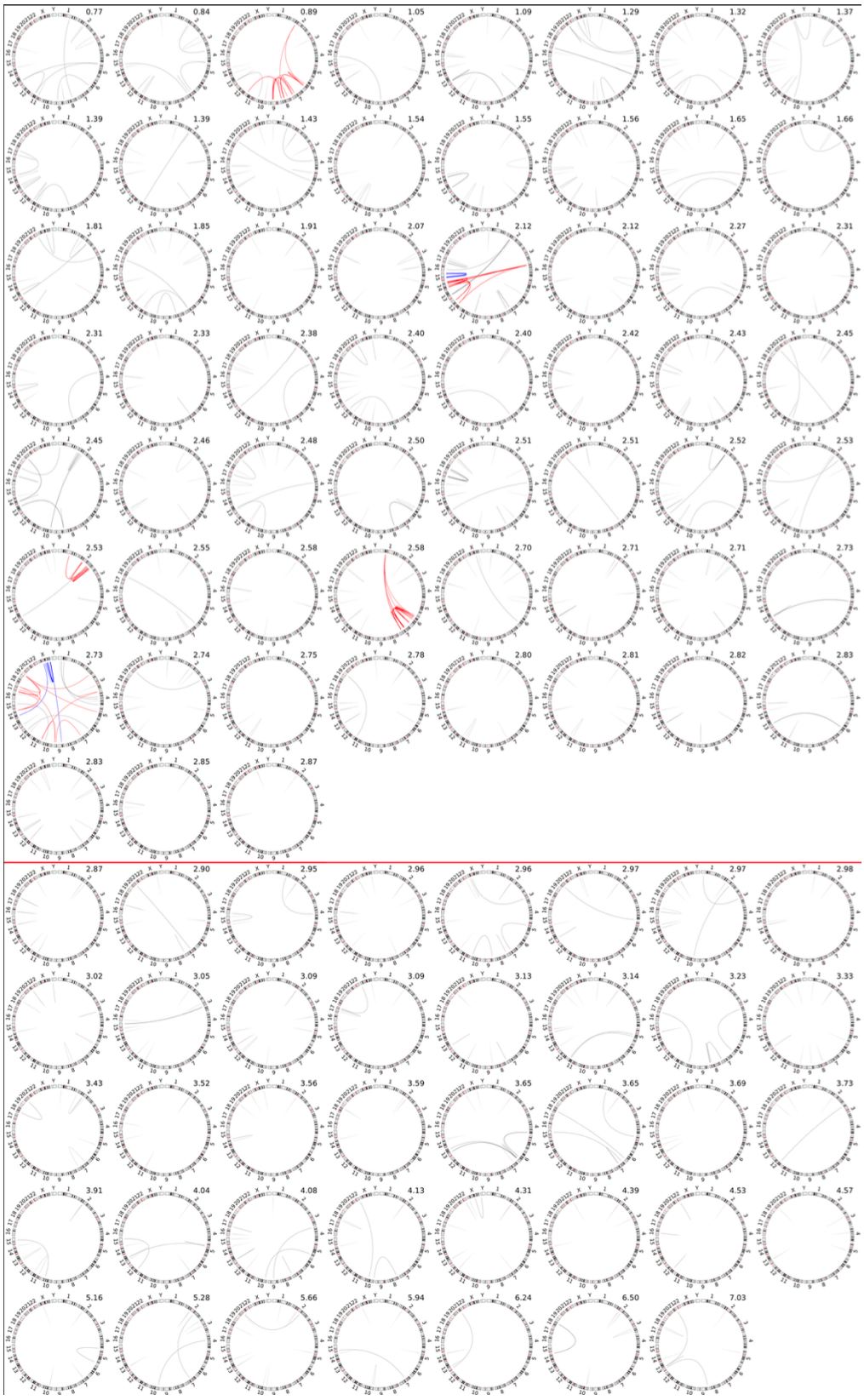


Figure 5.31. Circos plots displaying the clusters of variants in colour (red, blue), with non-clustered variants in grey as determined by the chain linking pipeline using a binomial p-value threshold of 0.15. Number shown top right of each circos plot is the telomere length as measured by STELA, and samples are separated by the red line in the middle by a TL threshold of 2.87kb. Data displayed relating to GEL CLL cohort (n=98).

Plotting the number of chained and non-chained SVs against telomere length as a stacked bar plot, the 5 samples containing chained SVs can be seen clearly (figure 5.32).

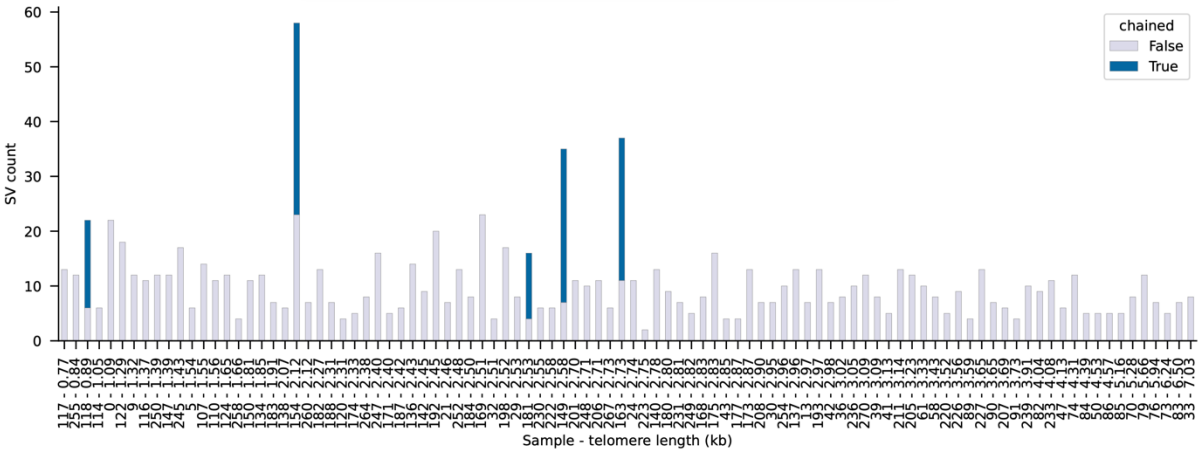


Figure 5.32. Stacked bar plot showing the number of chained and non-chained SVs (binomial p-value threshold 0.15) ordered by sample telomere length. Data displayed relating to GEL CLL cohort (n=98).

The small number of samples containing SVs are once again highlighted in Figure 5.33. Despite the small number of samples containing chained SVs, the Mann Whitney U statistic is close to significant with a p-value of 0.065. Generally, the number of non-chained SVs is also higher whilst again the Mann Whitney U p-value is not significant at 0.22.

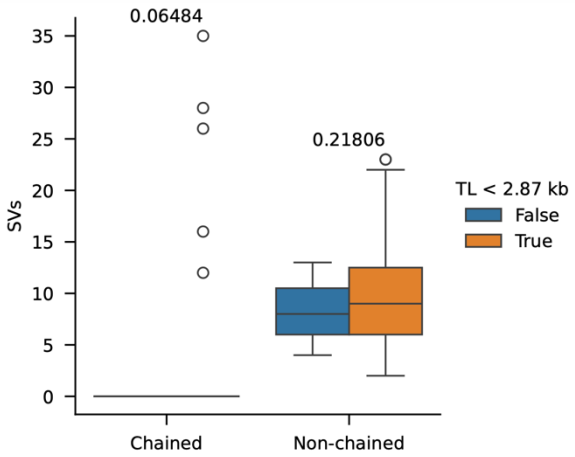


Figure 5.33. Box plot showing the number of chained and non-chained SVs for samples above and below a measured (with STELA) telomere length threshold of 2.87kb, with the Mann Whitney U p-value for testing if shorter is greater than longer above the boxes. SV clusters determined using a binomial p-value threshold of 0.15. Data displayed is from the GEL CLL cohort (n=98).

As expected with the small number of samples containing chained SVs the assortativity and modularity coefficients are not significant at 0.16 each (figure 5.34).

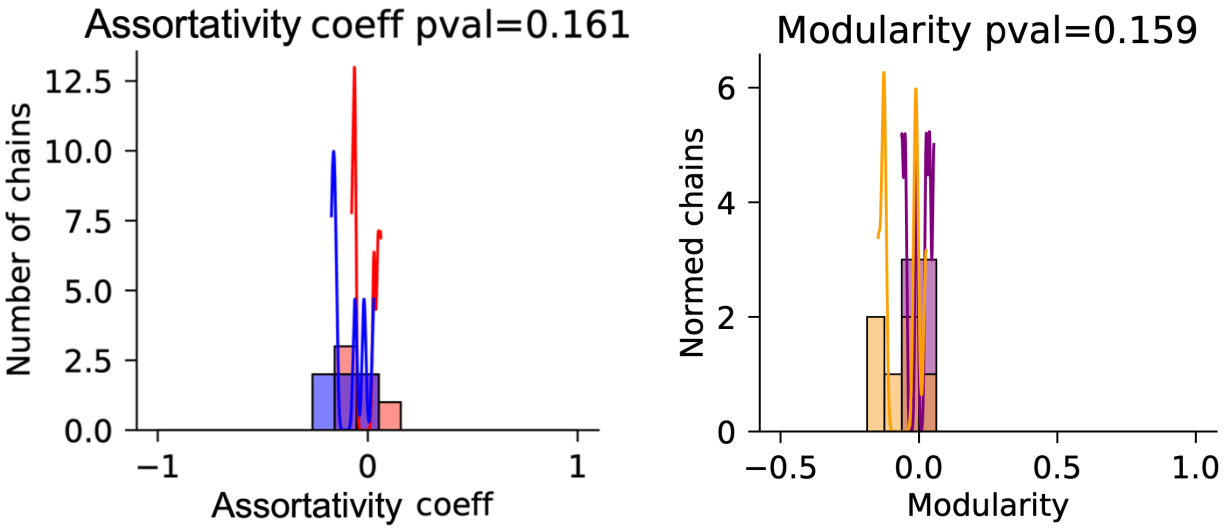


Figure 5.34. Assortativity coefficient values for random (blue) and SV clusters (red) left. Modularity of random (yellow) and SV clusters (purple). SV clusters determined using a binomial p-value threshold of 0.15. Both assortativity and modularity shown as histogram overlayed with a KDE fit for the distributions in the values. Data relates to the GEL breast cancer cohort (n=98).

The observations from this analysis showed that clustered chains of SVs, which are indicative of chromoplexy, are more common in cancers with shorter telomere lengths for breast cancer.

A significant increase in chained SVs ($p < 0.001$) is observed in the GEL breast cancer cohort, whereas in the case of the ICGC breast cancer cohort, significance ($p < 0.05$) is only achieved by removal of two anomalous results. In both cases, the validity of these clusters was confirmed by testing against the assortativity and modularity of a randomly generated set of clusters. Whilst not statistically significant, it is accurate to state that chained SVs (even if not fully validated by testing against a randomly generated dataset) only occurred in CLL for samples below a telomere length threshold of 2.87kb.

5.5 Discussion

It is clear from the relative copy number analysis, there appears to be a difference in the copy number profiles of shorter telomere length samples when compared with their longer counter parts. This division is most apparent in the largest data set (Genomics England breast cancer), but even within the smaller ICGC breast cancer cohort, a similar pattern of the shortest third of samples exhibit more complex genomes regarding the copy number profile. Using teltool for the telomere length predictions of the ICGC breast cancer cohort was not necessarily about getting accurate absolute telomere length predictions, but more so served as a means of ordering the samples by predicted telomere length. For this reason, the 4.55kb threshold determined should not be taken absolutely. For CLL it appears that whilst less complex and with a different proportion of samples (roughly two thirds instead of one), this cohort can be separated by recursive partitioning along telomere length to reveal a significant difference between a short and long class of samples. Another dissimilarity between CLL and breast cancer, is whilst breast cancer sees increases in both gains and losses when decreasing telomere length, it is majority losses which are affected in CLL.

Structural variant analysis for the most part corroborates the findings of the relative copy number analysis. The GEL breast cancer cohort, once again most likely due to its size, displays the most obvious increases in SVs (all types and total at $p < 0.001$) when using a pre-established telomere length threshold (3.81kb). Although the threshold used for the ICGC cohort (4.55kb) was at a different, it should not take away from the fact significant increases can be seen in all SV types except duplications ($p < 0.01$ for translocations and 0.05 for others) when ranking samples by a prediction of telomere length. On the other hand, there are potential issues that must be acknowledged when looking at this dataset. Both the accuracy of the telomere length, and "one size fits all" approach to structural variant calling may be contributing factors as to why this cohort looks different in both the copy number and structural variant analysis. The likely artifact of ubiquitous inversions which arose potentially either through the subsampling and/or uniform settings for SV calling across a varied cohort which were only apparent in the GEL breast cancer cohort and were relatively uncommon. It would have been preferable to find a method to remove these, however because of time and the Genomics England (GEL) research environment (RE) restriction this was not carried out. Due to their frequency, it is unlikely they skewed results too heavily as they appeared randomly across all telomere lengths.

These issues should not mean however that these results should be totally disregarded. Without absolute telomere length data, which is missing from the vast majority of datasets, this is the only method for performing such analysis. However, a case where real telomere length data is present in the form of the GEL CLL data, also shows significant differences in the number of SVs in short compared to long telomere length samples. In this cohort, total SV number increase ($p < 0.05$) was seen in a threshold generated from copy number analysis (2.87kb), and significant ($p < 0.05$) increases in deletions, translocations, and total SV count when using a literature defined telomere length threshold (2.26kb). This

threshold (2.26kb) determined through telomere fusion analysis and had the highest hazard ratio (HR) score for overall survival out of values below the prognostic value of 3.81kb. The value of 2.87kb determined through the copy number analysis was relatively close (~610bp) to this value.

An important point to consider is the CLL cohort was part of a clinical trial. This is relevant to this analysis as selected patients had a poorer prognosis, which has been shown to be correlated with telomere length, so it was more likely the cohort would and do have shorter telomere lengths. Whilst the trials did not fully exclude patients with deletions of 17p (as shown in figure 5.11 and mentioned in the original publications), it was taken into consideration as part of the trial (Howard et al. 2017; Munir et al. 2017). This means the majority of patients would have intact Tp53, which could lead to more stable genomes (via mechanisms outlined in chapter 1).

Chain link analysis revealed that significant differences ($p < 0.001$) were observable in the GEL breast cancer dataset. Upon removal of two outlies for the ICGC breast cancer cohort, a significant difference ($p < 0.05$) can also be seen, however in small cohort it is questionable whether it is appropriate to remove them. For both cohorts, the classification into chains is suitable as revealed by the assortativity and modularity values. Where this is not the case though is the CLL data. Despite this, the data does suggest there may be some correlation with a p-value close to 0.05 at 0.065, however further analysis is required to validate this suspicion. Unfortunately, due to the difficulties of working within the GEL RE, it was not possible to experiment further with testing different variables such as the number of breakpoints to find potentially improved outcomes of the chain linking analysis. Results from the analysis indicated that perhaps with a refined process, which is able to iterate over multiple variables, it may be possible to use a recursive partitioning approach to independently stratify cohorts from the copy number approach. Alluded to by the CLL

results, this would possibly be a more effective method for stratifying (TL threshold discovery) in less complex cancers where phenomena such as chromothripsis and chromoplexy are less common.

Chromothriptic oscillations (between 2 or 3 CN states) are more commonly observed within the shorter sections of the copy number "squiggle" plots. Whilst not exclusively occurring below the recursive partitioning threshold outlined by the complexity score for breast cancer, this oscillating pattern is not seen above 2.71kb for CLL. It is worth noting even though it was first reported in CLL, it has one of the lowest frequencies of chromothripsis; breast cancer on the other hand has one of the highest across all cancer types (Cortés-Ciriano et al. 2020). One explanation for the chromothripsis observed in the RCN analysis is breakage fusion bridge resolution outlined by Maciejowski *et al*, however without further data it is unclear which mechanism is responsible in each instance.

The chain linking analysis which is based on the work by Baca *et al* also displays the type of clustered SVs across multiple chromosomes that are indicative of chromoplexy. The most likely cause for the observed differences between the shorter and longer predicted telomere length samples is of replicative origin following escape of telomere crisis. Two models for chromoplexy are presented by Baca *et al*: the "sequential-dependant" and "simultaneous" model, distinguished by multiple and a single cell cycle being involved respectively. Without multiple rounds of sequencing over the cancer cells progression it is not possible to declare which type can be attributed to each sample.

Instances of consistent gains across singular (possible chromoanasynthesis) or pairs of chromosomes are observable within both cancer types. The gains of chromosome 12 observed frequently in the CLL cohort is around the expected ~20% for trisomy 12 (Howard et al. 2017; Munir et al. 2017; Abruzzo et al. 2018). Along with gains in chromosome 12, the deletions in 11q and bands of deletions in 13q (and less commonly deletions in 17p, and 6q)

are consistent with previous research from Döhner et al (Döhner et al. 2000; Hallek et al. 2018). These common lesions are distinct for different cancer types and are selected for. An example of this is the loss of 11q (especially 11q23) in CLL is a frequent event during pathogenesis due to its association with the ATM gene loci, heavily involved in the DNA damage response (Stankovic and Skowronska 2014). Equally, 13q(14) deletions which were also observed are linked to the loss of microRNAs involved in apoptosis related pathways (Khalid et al. 2021). Unlike with CLL, it is possible that the gains seen in chromosomes 1, 8, 17, and 20 are the result of chromoanasynthesis, which is congruent with results shown with MPseq of breast cancer found by Vasmatzis *et al*, as along with gains of copy number, large clusters of SVs can also be commonly found within these chromosomes (Vasmatzis et al. 2018).

Taken together the findings from all the different analyses designed to investigate various aspects of genomic complexity, a conclusion that telomere length is associated with the complexity of cancer genomes can be drawn. Potential biological mechanisms of how telomere dysfunction may drive genomic instability and the clinical implications of this relating to cancer will be discussed in the next chapter.

Chapter 6 Conclusion

6.1 Telomere length and genomic complexity

6.1.1 Results

The data presented in this thesis demonstrates a correlation between telomere length and genomic complexity. Overall, the data shows a trend where patients can be separated into a shorter and longer group classified by a specific telomere length threshold for each cohort that marks a transition in genomic complexity. Chapter 4 which used telomere length data generated with a high-resolution molecular assay, showed the clearest distinction between the genomic complexity of samples with telomeres above and below a telomere length threshold. Across each type of analysis: copy number, counting and chaining of structural variants, there was a clear visual distinction between "short" and "long" samples. This distinction was further reinforced by the bioinformatic analysis that provided significant p-values in each respective analysis. In chapter 5, this phenomena of segregation of complex profiles by telomere length was also established, however the clarity in some cases was less well defined. It was considered that this difference in clarity may have arisen because less accurate bioinformatic based approaches to determine telomere repeat content and infer telomere length were employed. The largest cohort (GEL breast cancer n=1591) showed highly significant ($p<0.001$) results across every analysis aimed at investigating different aspects of genomic complexity. The ICGC breast cancer cohort (n=~77) also exhibited significant p-values for differences in the copy number profile ($p<0.01$), and for counts of all structural variant types outside of duplications ($p<0.05$). The genomes analysed in the CLL cohort

(n=98) overall appeared less complex than breast cancer, which is congruent with previous research. Although first being uncovered in CLL, it has since been shown that CLL has one of the lowest frequencies of chromothripsis of any cancer, whilst breast cancer shows one of the highest rates (Stephens et al. 2011; Cortés-Ciriano et al. 2020). Despite not all tests showing significant values, there were distinct thresholds that yielded significant ($p<0.05$) results for copy number, deletions, translocations, and total structural variant count.

The telomere length thresholds defined on the basis of genomic complexity are consistent with independently established telomere length thresholds that define both prognosis and prediction of response to treatments of patients with cancer (Lin et al. 2010; Simpson et al. 2015; Strefford et al. 2015; Hyatt et al. 2017; Williams et al. 2017; Norris et al. 2019; Pepper et al. 2022). These observations point to a fundamental characteristic of telomere length and biology that is clearly important for the progression to malignancy. The existence of a telomere length threshold revealed by these data, point to telomere length and dysfunction as a biological origin for the increased genomic complexity seen in these samples. It is thought that dysfunction of short telomeres arises because of their inability to bind sufficient shelterin through the depletion of binding sites for TRF1 and TRF2, and the T-loop structure is unable to form (de Lange 2009; Zhu et al. 2019; Rossiello et al. 2022). A model proposed by Cesare and Karlseder supposes telomeres have three states: closed with functional intact T-loop structure, intermediate where excessive shortening has prevented protective structure formation, and uncapped with all TRF2 binding sites have been eroded (Cesare and Karlseder 2012). If there were a gradient of increasing genomic complexity with decreasing telomere length, it could be argued that this correlation is either the result of complex cancers increased proliferation causes shortening of telomeres, or that shorter telomeres cause the processes that lead to increased complexity. The “hard line” present at the threshold is less explainable with the former theory, whereas mechanisms for the latter have

been explored previously by Baird et al. Furthermore, the presence of a threshold implies there are at least 2 states for telomeres: “long” and stable, or “short” and dysfunctional which is in line with the model proposed by Cesare and Karlseder with regards to the outcome of the intermediate and uncapped states being similar. Once the short and dysfunctional telomeres state is reached, the phenomenon of increased genomic complexity is easier to explain mechanistically through events such as telomere fusions.

Several pathways have the potential to drive genomic complexity in the context of short dysfunctional telomeres. When shorter telomeres are processed in vivo as double strand breaks, telomere fusions occur, which can then be subjected to a variety of recombination pathways are used to stabilise the cell to prevent cell death, or to cycles of anaphase bridging breakage and fusion. Telomere to telomere fusions can create dicentric chromosomes, which within breakage-fusion-bridge cycles can break simply to form fold-back inversions, or complexly from mechanical forces or nuclease attacks to create multiple fragments. These fragments can be repaired through replicative repair such as Microhomology-Mediated Break Induced Replication (MMBIR), Multi-Invasion-Induced Rearrangements pathway (MIR), or Non-Homologous/Theta mediated End Joining (NHEJ/TMEJ) (Cleal et al. 2019; Liddiard et al. 2022). Complexity can also be induced through micronuclei formation fragmentation through faulty DNA repair mechanisms, and then reincorporated into the main nucleus with the alterations (Zhang et al. 2015; Cleal and Baird 2020; Lin et al. 2023).

The data presented in chapters 4 and 5 relating to chain linking analysis of breast cancer cohorts revealed loci of SV chains deviated from randomness. The presence of such clusters suggests spatial confinement of the genome or multi-step processes as mechanisms behind part of the observed complexity. The mechanisms are not known for how chromoplexy is generated, however it can be speculated based on these two characteristics. Lagging chromosomes or chromosome fragments that are incorporated into micronuclei

offers a plausible method of how these loci are spatially confined. It is also possible that replication-based strand invasions of spatially nearby loci within the main nucleus could occur. Equally, oscillations characteristic of chromothripsis were observed in the copy number analysis in these chapters. Chromothripsis is presumed to occur via mechanisms involving random end joining resulting from a single event and has been associated with NHEJ and TMEJ. Relative copy number analysis also alluded to the presence of chromoanasynthesis with whole chromosomes and chromosome arms exhibiting large gains indicative of duplication or triplication. This process has been linked to fork-stalling and template switching (FoSTeS) or MMBIR. In aggregate these results imply a plethora of mechanisms are responsible in driving a range of complexity, with origins relating to dysfunctional telomeres. Further analysis perhaps using long read sequencing of telomere fusions could help uncover more specifics and remove the shroud of speculation.

6.1.2 Clinical Implications and Applications

It has been documented there is a correlation between increased genomic complexity and poorer prognosis (Goergen and Al-Sawaf 2024). In this study, the complex karyotypes of CLL were subdivided into 3 groups depending on the number of aberrations (low ≤ 2, intermediate 3-4, and high ≥ 5) based on previous analysis that also identified this correlation (Leeksma et al. 2020). Goergen and Al-Sawaf in their meta-analysis compared treatment outcomes for each category of complexity. They found chemo- and chemoimuno- therapies had worse outcomes in complex karyotyped patients than low complexity patients. High complex karyotyped patients showed better outcomes in ibrutinib-containing treatment regimens compared to those treated with chemo and chemoimmuno therapies. Approaches utilising copy number alteration detection of specified regions have also been proven to be prognostic. Meta-analysis from Chun et al has shown there are several genomic regions with

accompanying candidate genes that range from established markers (13q - miR15a/16–1 and 17p13 – Tp53), suspected (11q13 - ATM), and recurrent (20q11 and 1p36) which are and could be prognostic when loss of heterozygosity occurs (Chun et al. 2018).

It has previously been shown that the telomere length threshold, defined functionally by the presence or absence of telomere fusions, is able to stratify patients in prognostic sub-groups (Lin et al. 2014). Alongside prognosis, telomere length also has potential power in the prediction of patient treatment responses. A study by Norris et al on the ARTIC/ADMIRE CLL cohort (also used in Chapter 5) have shown TL as a predictor of both progression free survival (PFS) and overall survival (OS) in patients treated with chemoimmunotherapy combination of fludarabine, cyclophosphamide, rituximab (FCR) based therapies (Norris et al. 2019). Subsequent analysis from Pepper et al on the same cohort showed TL in combination with IGHV mutation status and CD49d expression could be used to assess risks in assigning treatment. For cases where wild-type Tp53 was present but IGHV was mutated, shorter telomeres and cases of longer telomeres but with CD49d mutations it would be suggested to use chemo-free treatment options, whereas long telomeres with no CD49d mutations would be suitable for FCR based treatment (Pepper et al. 2022). It is possible that shorter telomere cancers are so unstable this confers the resistance shown against to traditional chemo (and radio) therapies that aim to damage DNA to the point apoptosis is induced.

Telomere fusions are likely mediated through A-NHEJ which is a LIG3 dependant pathway as opposed to C-NHEJ requiring LIG4. This was demonstrated by Jones et al who showed through gene knockout (KO) experiments that LIG3 is required for escape of telomere crisis, whereas LIG4 is not (Jones et al. 2014). Further experiments from Ngo et al have also demonstrated PARP inhibitors olaparib and rucaparib were successful in preventing cells from escaping crisis as well (Ngo et al. 2018). This suggests cancer cells with shorter

telomeres might therefore respond better to PARP (involved in signalling LIG3 to sites of DNA damage) and POLQ (polymerase involved in A-NHEJ) inhibitors which act to inhibit DNA repair mechanisms. Inhibition of A-NHEJ should prevent stabilisation and subsequent escape of telomere crisis, therefore making it more likely the DDR triggers apoptosis. Combining the previous observations in CLL with the findings of this investigation, it is reasonable to extrapolate the idea that telomere length plays a role in the oncogenesis and severity of other cancer types.

6.2 Challenges

6.2.1 Sequencing technology differences

During the creation of a machine learning (ML) telomere length prediction tool, unforeseen issues were encountered relating to differences in data quality between sequencing technologies. For example, with the method outlined in the teltool section, one could reasonably assume that reads containing TTAGGG repeats when extracted and aligned to a reference sequence containing purely TTAGGGn should yield a GC% value of 50% regardless of sequencing technology. However, consistently BGISeq would yield values of ~52% and Illumina ~48% (figure 3.10A). Although this appears a relatively small deviation, ML models can be very sensitive to deviations from training data, known as data drift (Rahmani et al. 2022). In an attempt to address this issue methods such as optimal transport were investigated to transform data prior to analysis, aiming to improve the alignment between the distributions of test data and training data (figure 3.16) (Peyré and Cuturi 2019). The difference in even a simple variable such as GC% is likely due to the polymerases used in each sequencing technology, or possibly library preparation issues diminishing/enriching sequences sensitive to GC content, or even the chemistry determining the annealing to

flowcells/ZMWs. The observation of this difference led to the generation of a reference agnostic kmer count of BGI and Illumina samples (see Chapter 2.14.1). This was analysed with the help of an MSc student Laurie Fabian, who found that BGI data displayed a flatter coverage profile across the genome compared to Illumina. These differences are not often discussed, accounted, or tested for when considering bioinformatic tools and workflows. As Figure 3.19 shows, the same bioinformatic tool can have noticeably different performance in accuracy of prediction, depending on which platform is used for sequencing. This thesis highlights that this should especially be considered when developing or using a machine learning approach for genome data analysis.

6.2.2 Public repositories

I expected there to be challenges when working with public repositories, but I was not prepared for the reality. Restriction appears to be a trend for GEL (and other repositories), as the whitelist of IPs appears to be extremely small. Looking at this is from a security perspective this is ideal for minimising malicious traffic, but from a researcher standpoint this can create limitations. From their policy it appears as if they want everything in and outbound of their system to go through the "airlock" system which requires admin review before passage is approved with a strict rubric for what is permitted (even screenshotting the environment is a breach of policy which can lead to pursuit of "legal or criminal action against both the individual and the institution"). However, the inbound airlock can be bypassed through importing files in containers which can be pulled from Docker hub. This asks the question what the purpose is for restricting inbound files. As well as limited information which could be bypassed outbound in the form of taking a picture of the screen which whilst breaking policy, this method would have no way of being detectable in software and is based solely on trust. Another issue faced as a researcher is the organisation of data

within the Helix HPC. All genomes are stored under one directory ordered by date, so to access paths relating to a cohort, researchers go through the LabKey system to find these. The data pertaining to the CLL cohort had three fields of information which could be indicators of whether a file was sequenced germline or cancer (type column with “gl” or other like “t1”, parent directory starting with CANC or other, sample name CancerX_NormalY or X). These conflicted in a way that there is no actual correlation between any of these variables, and there was a duplication where two unrelated columns had been swapped with each other. Even asking a researcher who had previously worked on the cohort confirmed an incorrect assumption about which was the correct one, and analysis of samples was under this false concept of germline and cancer samples was conducted before a correction was made. Storage of this information in a highly esoteric format such as this feels like an unnecessary barrier which could lead to incorrect analyses being conducted. There is probably an explanation for why it is stored this way, but why does the result of this storage solution appear so confusing to people not on the backend?

The International Cancer Genome Consortium (ICGC) uses a vastly different approach to how researchers access data, storing data within an AWS bucket that can be accessed via their proprietary “score-client”. The score-client would regularly provide cryptic error messages and frequently refuse to download data on AWS. This prompted as switch to using the Collaboratory for analysis. Setting up instances requires configuration of a network system before being able to access created instances. The advantage over GEL in this case was having full control of most parts of the system. After the time spent to write a script that would have a fresh operating system install to the point where it could perform desired analysis, speed could be scaled by creating more instances to compute samples in parallel. It was tested whether mounting the data instead of downloading it would be viable, which would save both on time and cost of not having to give each instance large amounts of

storage. Unfortunately, the speed with mounting approach was not viable as transfer speeds amounted to kb/s for large, upwards of 100gb files. At the time of analysis (issue was later fixed), I could not find the IDs the score client used to download files in the csv downloaded from data portal, so I wrote a web scraper to pull them directly from the webpages hosting the information about the samples along with all the associated files accessible from the. With this approach, it would technically be possible to transfer any file from the repository locally, given enough storage space.

It's clear that some of the issues faced were due to systems whose design had good intentions. The airlock policy for example, adheres to data protection laws. However, in one case, whilst attempting to extract a csv file through the airlock system, I had informed the data admin that data outside of the research environment had been provided by a previous researcher who had worked on the dataset. Despite this it was cited that the csv in the requested format was in breach of criteria 6.8 of the airlock policy, so I was made to "anonymise" the names before being allowed to extract the data from GEL. It was made clear to them in an online meeting this "anonymisation" process was technically reversible as this is the only way the extracted information could be usable. Under a different name this data was now acceptable to export. I think this example highlights there are flawed areas in the system. To conform to guidelines, the same "anonymisation" process was performed outside of the research environment, but this felt unnecessary especially as the original was available.

If bioinformatics is meant to be a collaborative effort and reproducibility is taught as one of the core tenants, why then are there so many restrictions when using public repositories? An argument could be made that sufficient anonymisation at the first link in the chain (e.g. assigning random strings to identify patients where a record of this link is kept only on paper which is destroyed at the end of a trail) would allow genomic and clinical data to be publicly shared through open source means such as torrent networks. Storage has

decreased in price in recent years, where a 12TB hard drive can now be purchased for around £120, providing a relatively affordable route to store data for processing. Whilst it is true that genomic information is highly identifiable, without a reference how likely would it be sensitive information is linked back to an individual? What would be gained through doing this? Is this information valuable enough to go through the effort? If someone has the knowledge on how to do this, would it not already be occurring within these environments if it were profitable to exploit? Are there ethical concerns if the benefits and risks are explained to patients who voluntarily agree to the open sharing of their data? This model of data sharing is similar but more open than the European Genome-Phenome Archive (EGA), who has shown its possible for data permission classes to be unrestricted in their use (https://ega-archive.org/access/data-access-committee/data-use-ontology/).

6.3 Future research

If the project were to be continued there would be several changes that could increase the quality of the analysis. In an ideal world, there would be a large dataset containing WGS sequenced with the same model sequencer along with matched STELA telomere length measurements. This would allow for validation of telomere length thresholds alleviating variance from prediction-based methods and ensure consistency across methods interrogating the data.

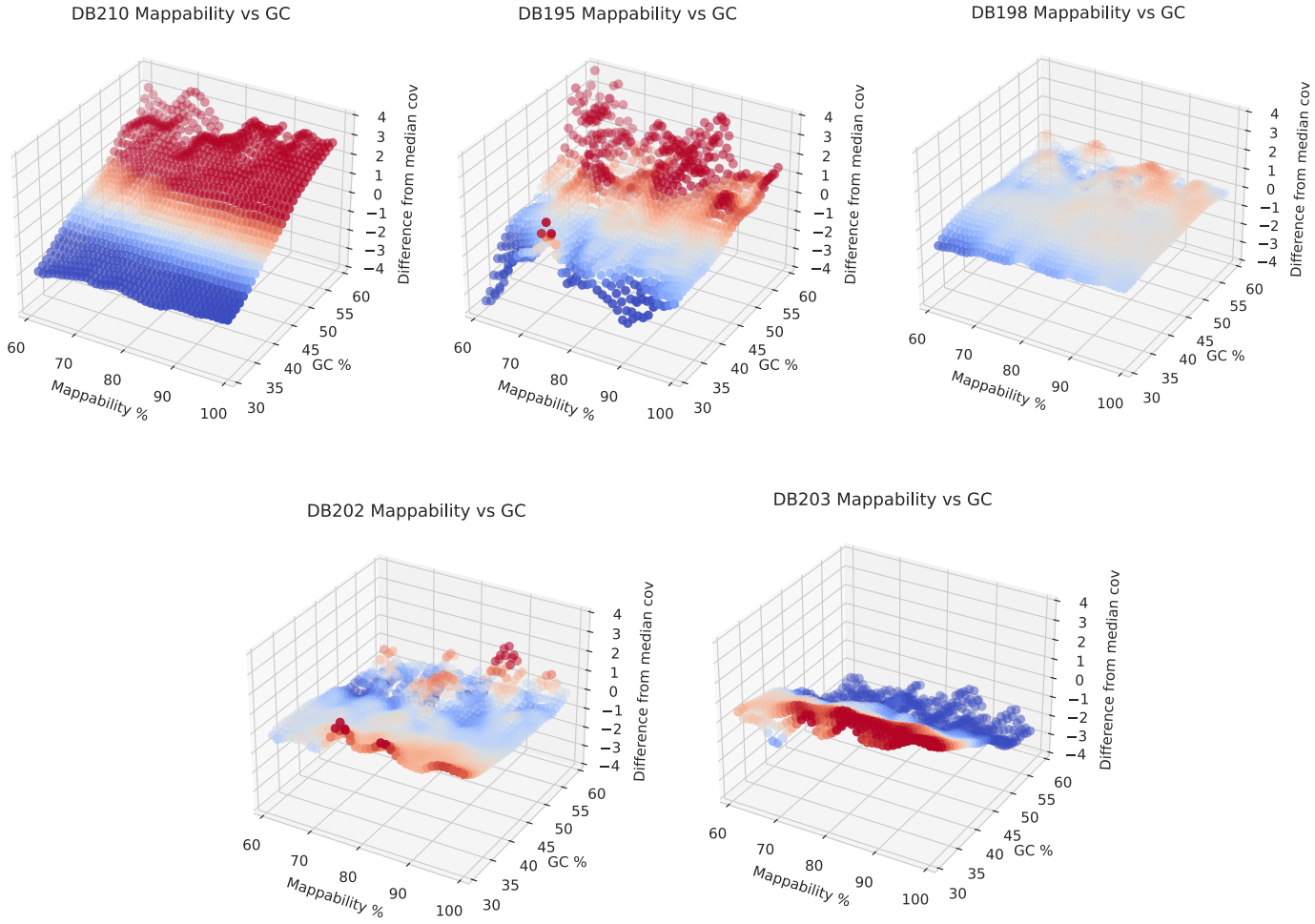
Sequencing multiple cancer types could also help elucidate the genomic complexity profiles of different cancers, along with comparing if thresholds are consistent across cancer types and whether there is a universal threshold. As previous research from Lin and Simpson et al has shown, the fusogenic range of 3.81kb initially defined in CLL is optimal in breast cancer, with stratification by the subset mean for CLL of 2.26kb is optimal (Lin et al. 2014; Simpson et al. 2015). A speculative explanation of why these are different could be due to the

higher purity of CLL compared to solid cancers such as breast cancer which contain varying amounts of tumour, stromal, lymphocytic cells. Additionally, telomere length has also been shown to be prognostic in Multiple Myeloma and Myelodysplastic syndromes (MDS) (Hyatt et al. 2017; Williams et al. 2017). The underlying biology is likely conserved between cancer types, so it is plausible the telomere length threshold where genomic complexity is increased could be a biological constant.

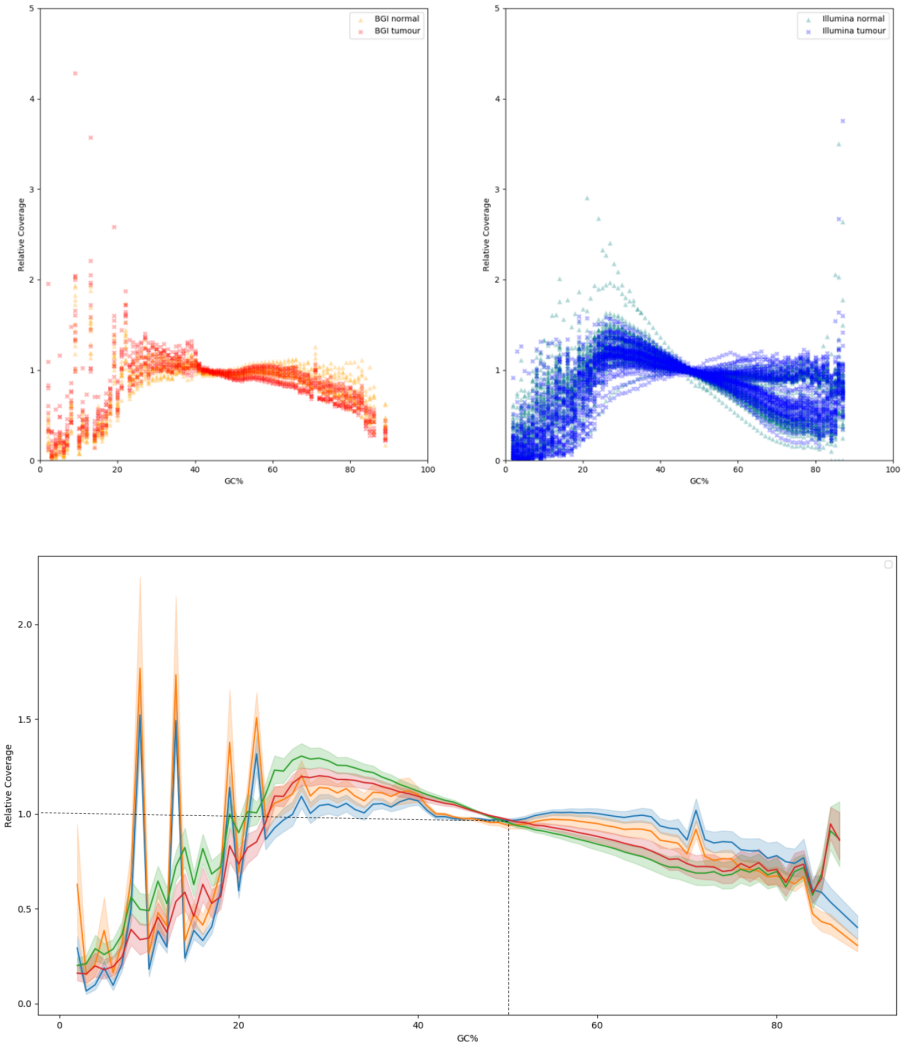
It would also be interesting to test for mutations in foci related to DNA repair and damage response pathways and explore any correlations. Mutational signatures could also be tested for any associations to see more specifically what is contributing to the observed complexity. These targets could help explain or hint towards biological assays which could be performed to understand underlying mechanisms for what has been observed. Previously mentioned moving to a long-read sequencing approach could improve contig assembly and help elucidate details of telomere fusions in greater detail. Additionally, clinical data to accompany this dataset would be a large bonus. Complementary data from optical genome mapping could also verify findings from SV calling.

A dataset with these improvements could more certainly define mutational signatures of cancers that have been through telomere crisis. Overall, these suggestions would further aid in establishing the main findings of this thesis. These findings can be summarised telomere length thresholds generated from copy number analysis and from the literature are able to significantly partition cohorts where below the threshold increased levels of genomic complexity in the form of: copy number alterations, raw counts of structural variants across multiple types, and proximate clusters of structural variants.

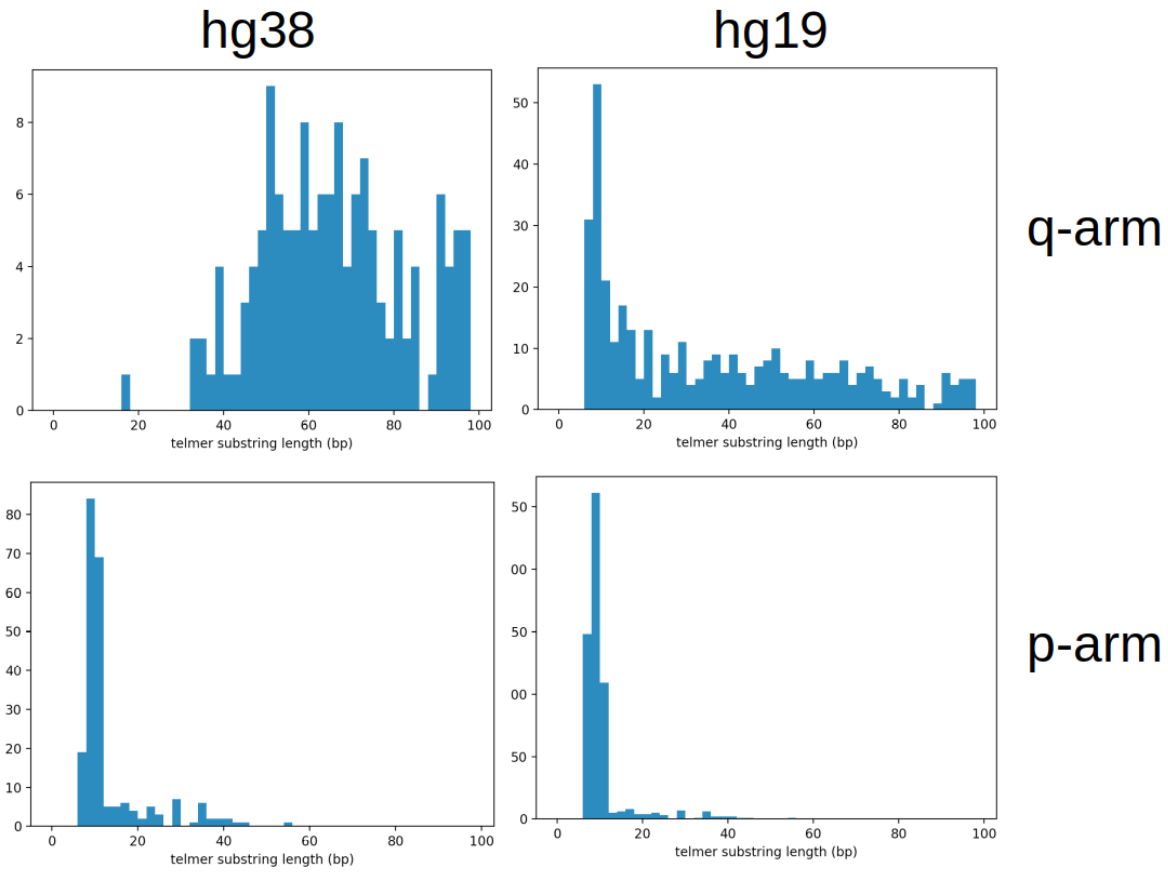
Appendices



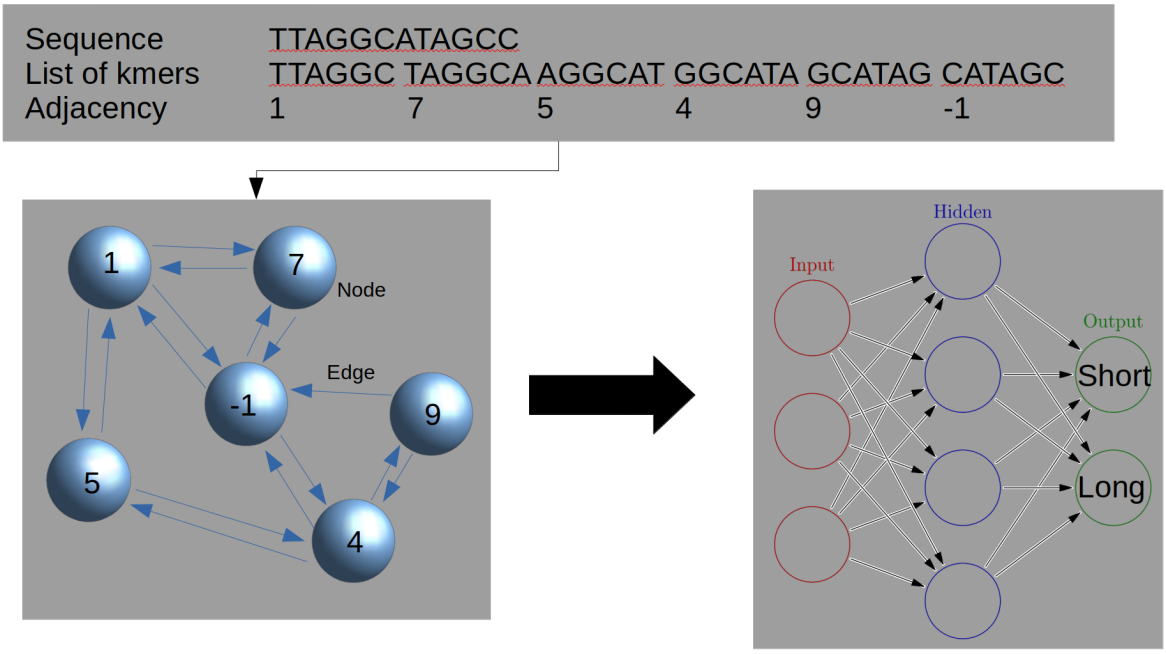
Appendix 1 Mappability vs GC matrices plotted in 3D space for 5 samples.



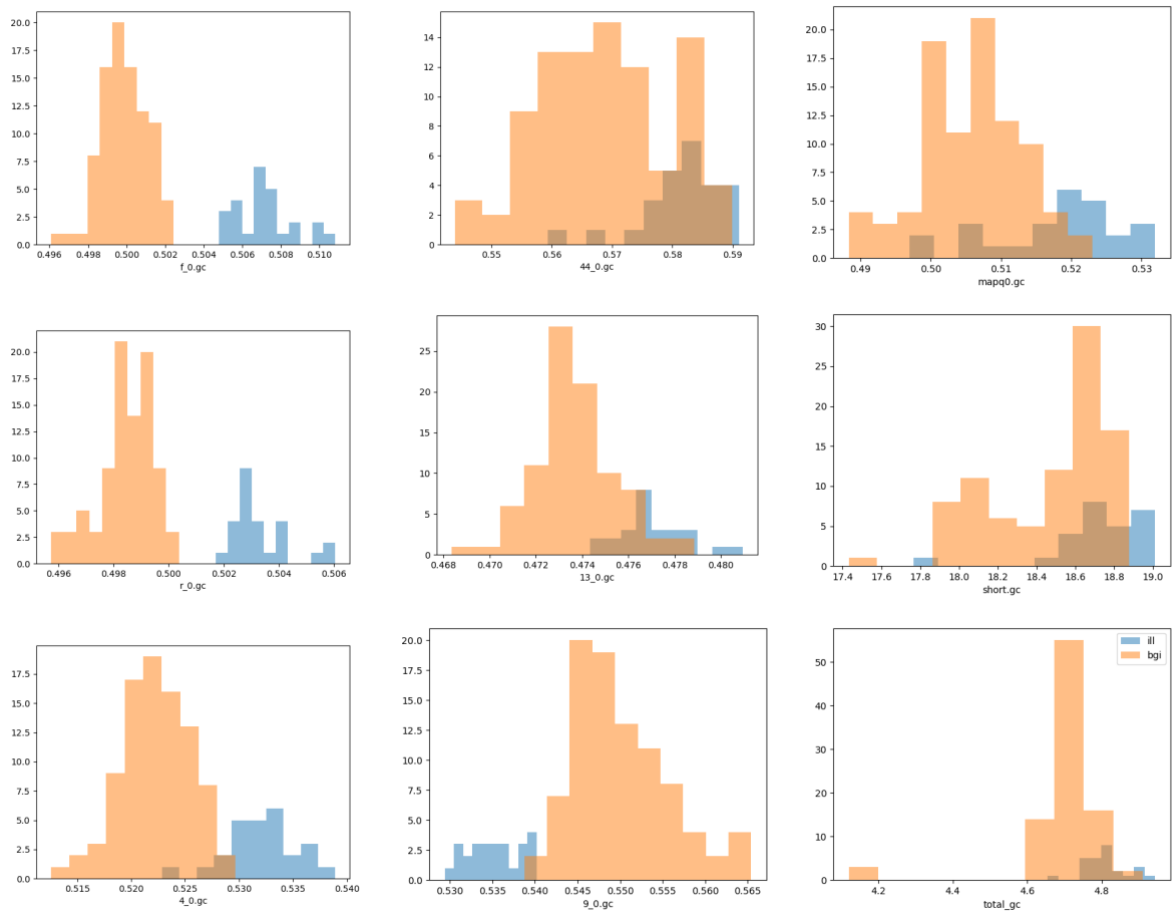
Appendix 2 Relative coverage plotted against GC%, split by BGI and Illumina sequencing as well normal and tumour samples.



Appendix 3 Telmer length variable using LiftOver conversion between hg38 and hg19 regions.



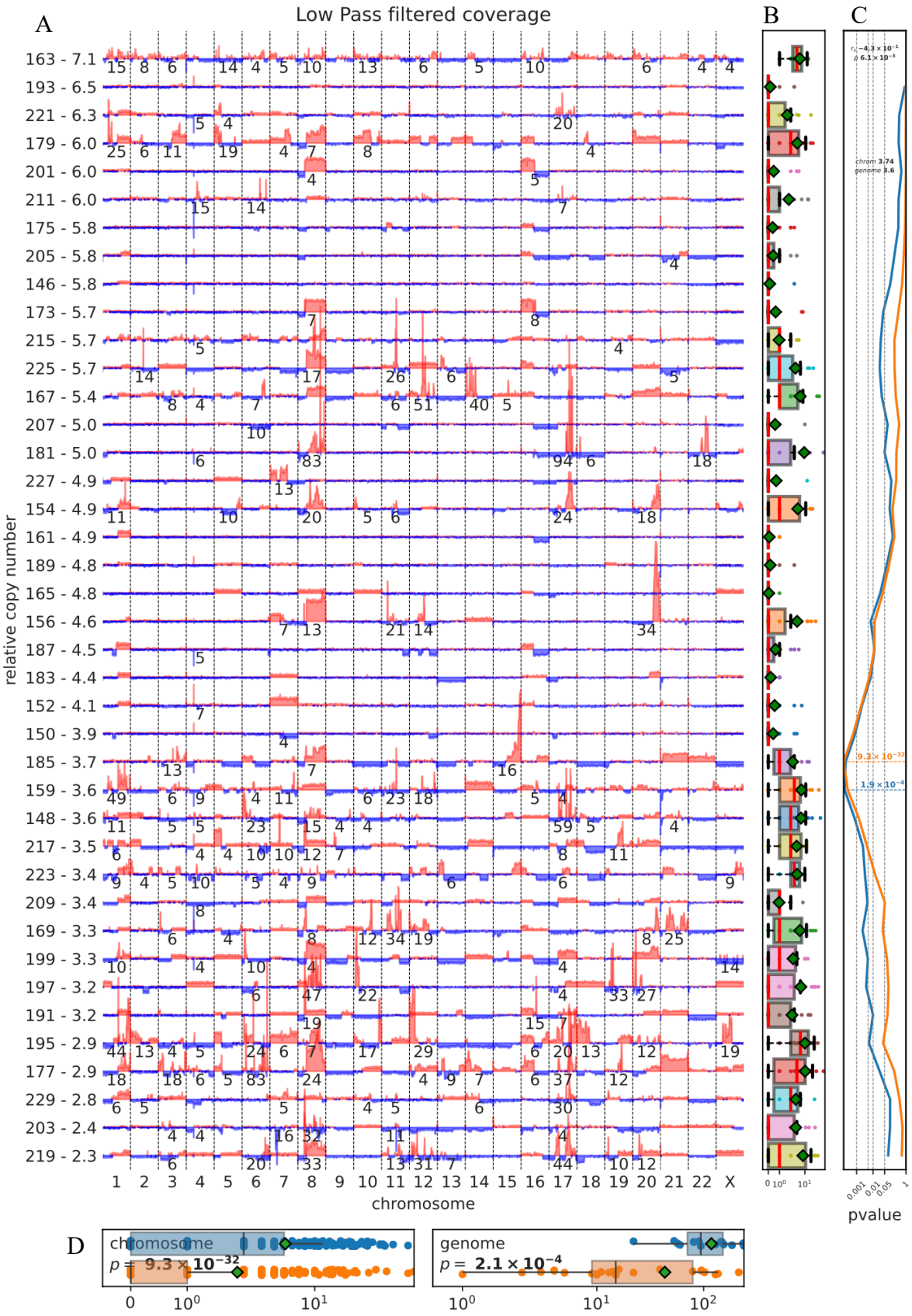
Appendix 4 Model for adjacency graph approach theorised for telomere length prediction.



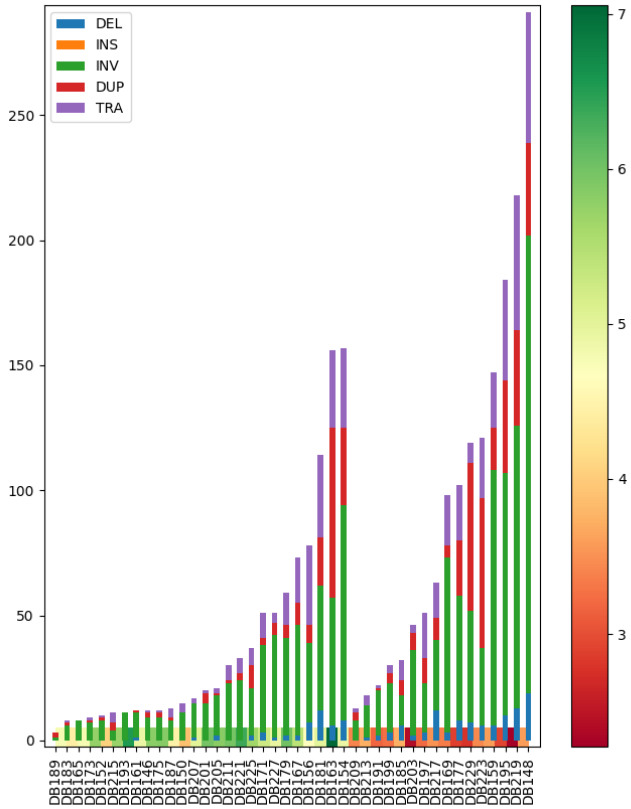
Appendix 5 GC% histograms for each region split by sequencing technology (blue: Illumina, orange: BGISeq)

A	B	A|B (OR)	A&B (AND)	~A (NOT)
0	0	0	0	1
0	1	1	0	1
1	0	1	0	0
1	1	1	1	0

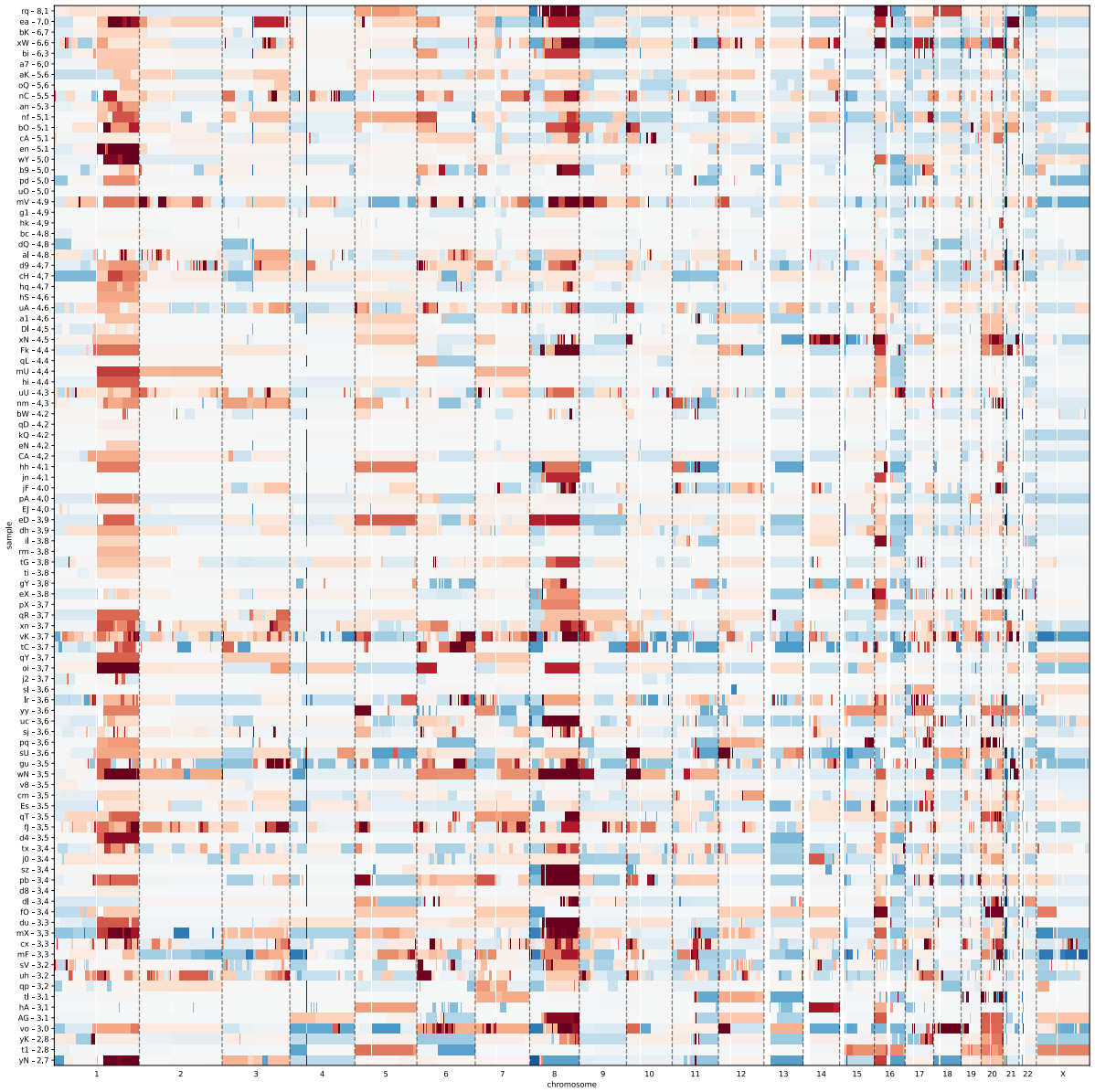
Appendix 6 Table to show the outcome of OR (|), AND (&), and NOT (~) bitwise operations



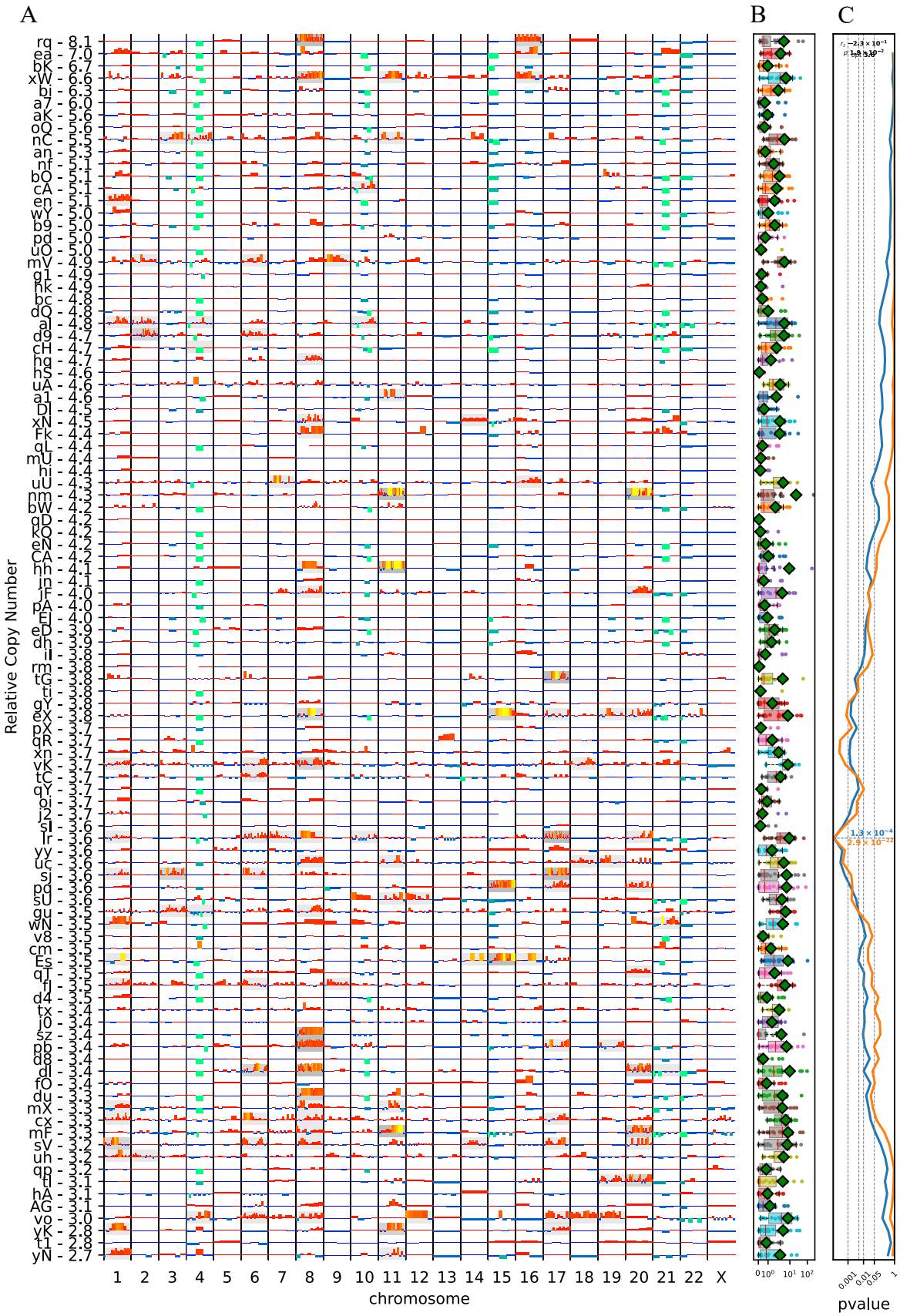
Appendix 7 Relative copy number after being processed by low pass filtering (A). Box plot to show the chromosome complexity score (shown under chromosomes > 4, sum of all instances peak and trough > 0.5 apart) (B). Recursive analysis of below complexity scores greater than above (orange chromosome, blue genome) (C), Box plot to condense all complexity scores for above and below recursive threshold (D). Data displayed is a local breast cancer cohort (n=40).



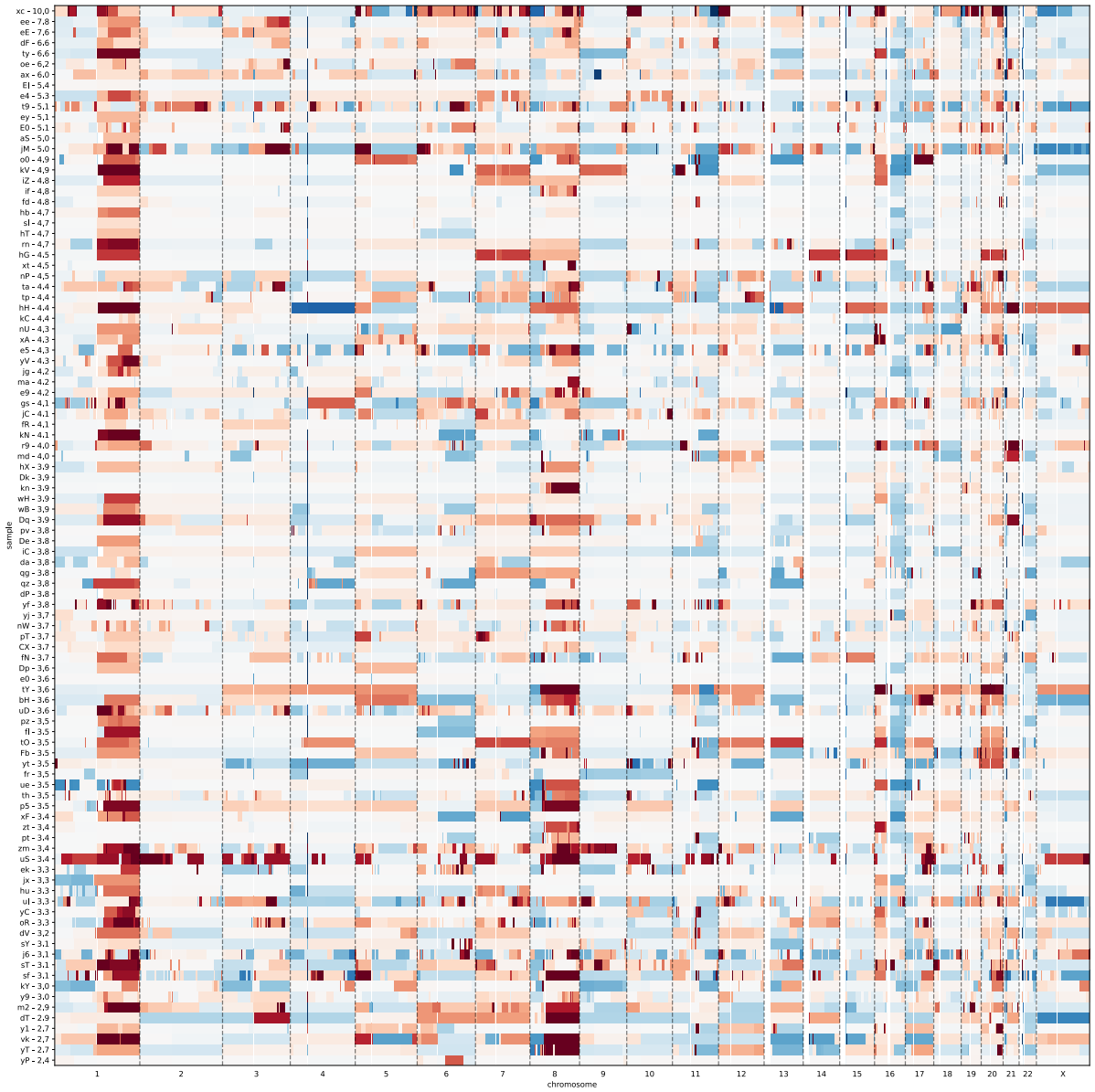
Appendix 8 Stacked bar plot to show number of SV types in samples with TL indicated by colour from right gradient bar.



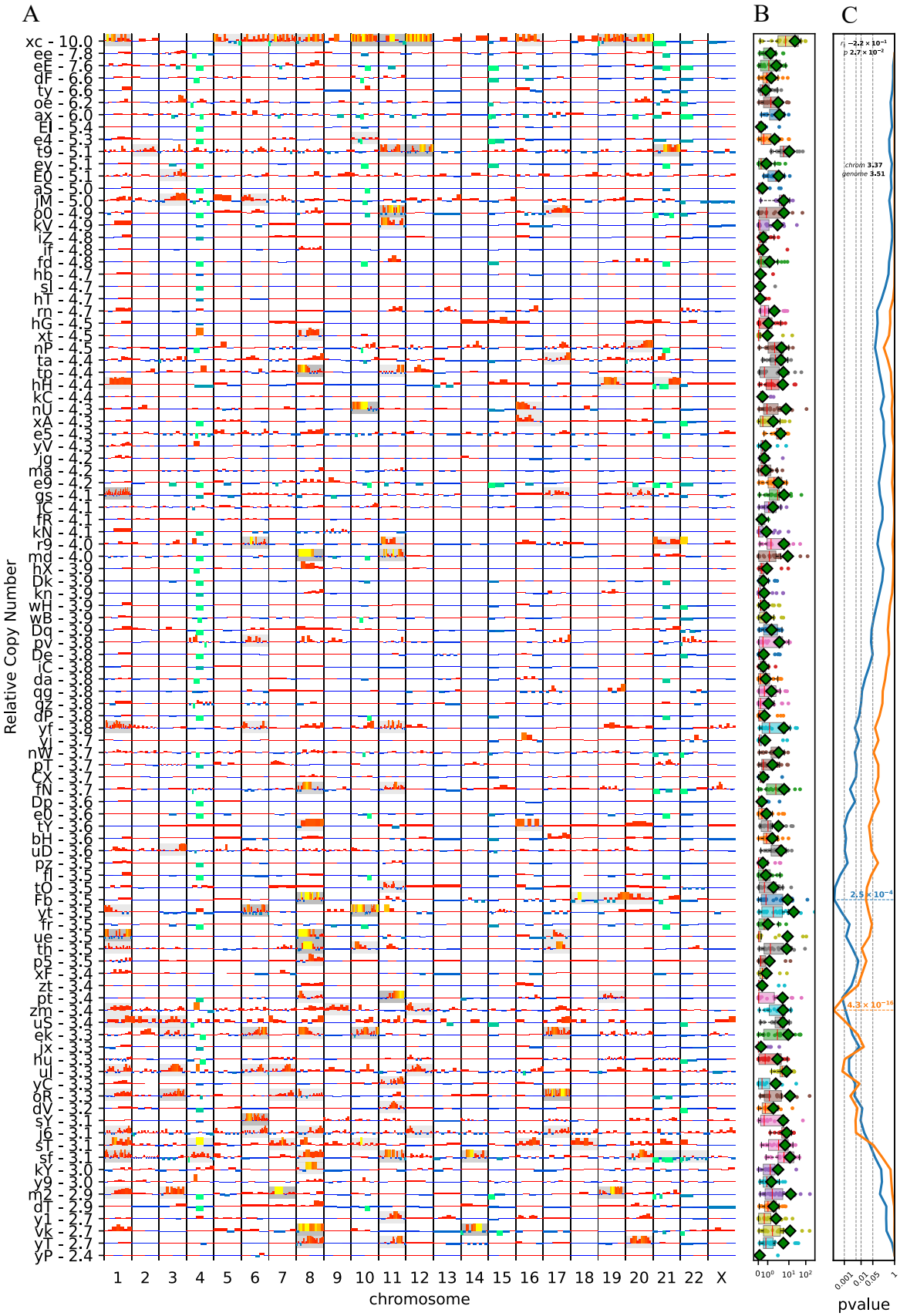
Appendix 9 Heatmap showing relative copy number segments from pcf analysis (input of 10kb coverage windows) clipped between -1.5 (blue loss) and 1.5 (red gain) with the Y-axis displaying ordered from longest telomeres (top) to shortest (bottom). Data displayed is a random subsample (n=100) of the GEL breast cancer cohort



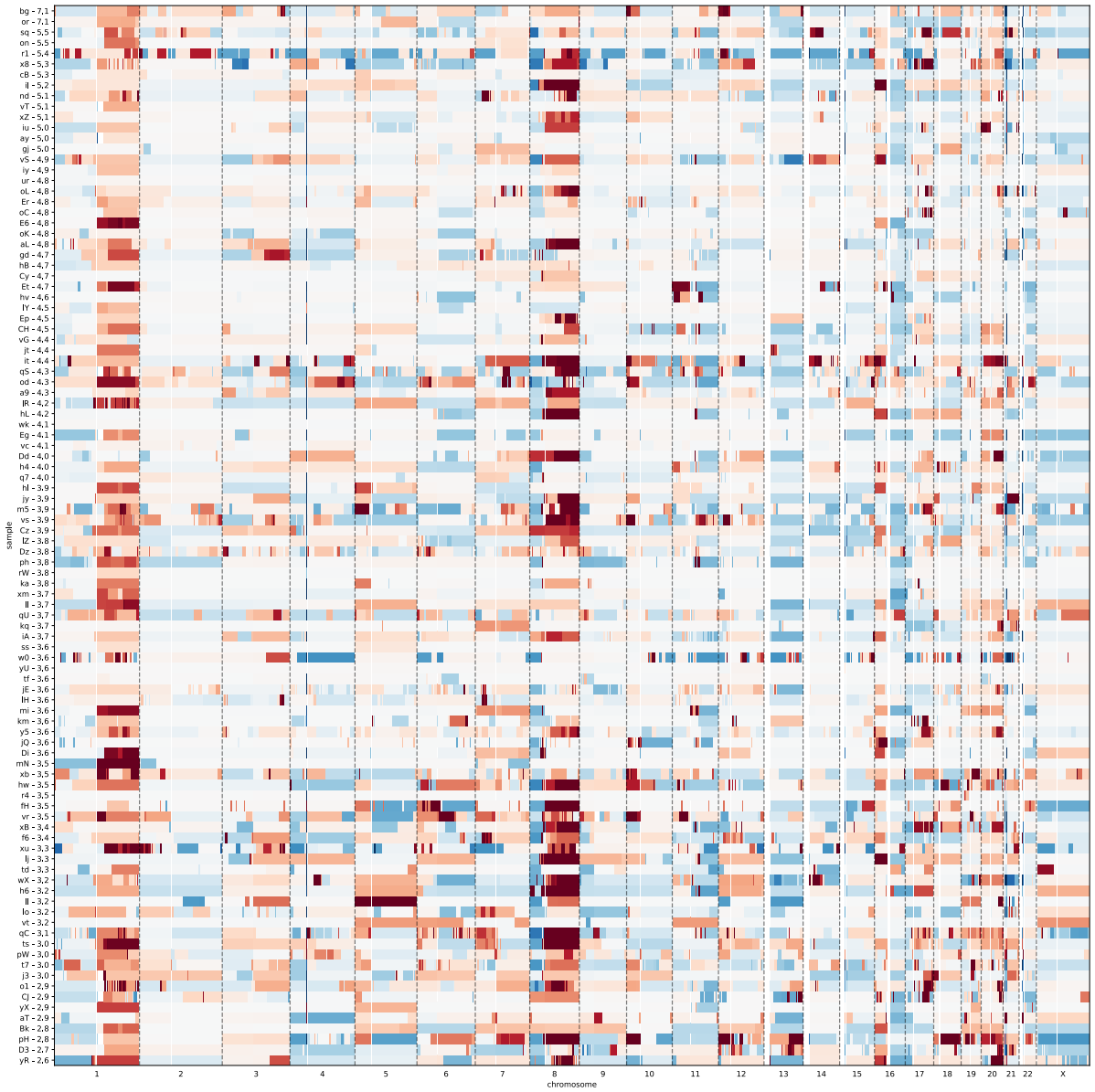
Appendix 10 The coverage graph (A) shows the relative copy number (RCN) segments scaled so each chromosome is uniformly sized, with the length of each segment on the x-axis being logged. The complexity score boxplots (B) show the distribution of the complexity scores across all chromosomes for each sample, with means indicated by the green diamond. The recursive partition line graph (C) shows the t p-value for the Mann Whitney U test for complexity score at and below each position > above for the sum (genome blue) and individual chromosome (orange). Data displayed is a random subsample (n=100) of the GEL breast cancer cohort.



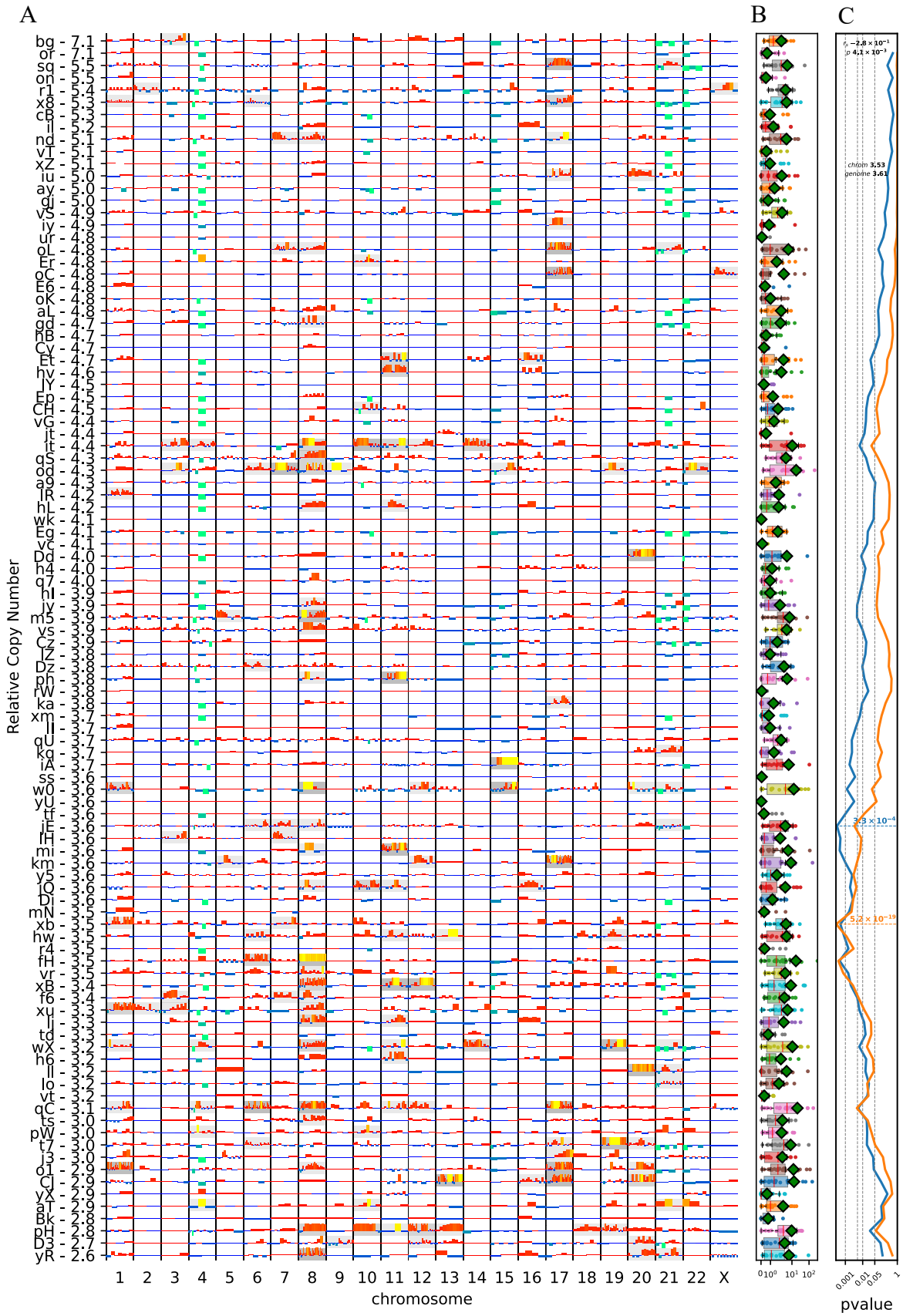
Appendix 11 Heatmap showing relative copy number segments from pcf analysis (input of 10kb coverage windows) clipped between -1.5 (blue loss) and 1.5 (red gain) with the Y-axis displaying ordered from longest telomeres (top) to shortest (bottom). Data displayed is a random subsample (n=100) of the GEL breast cancer cohort.



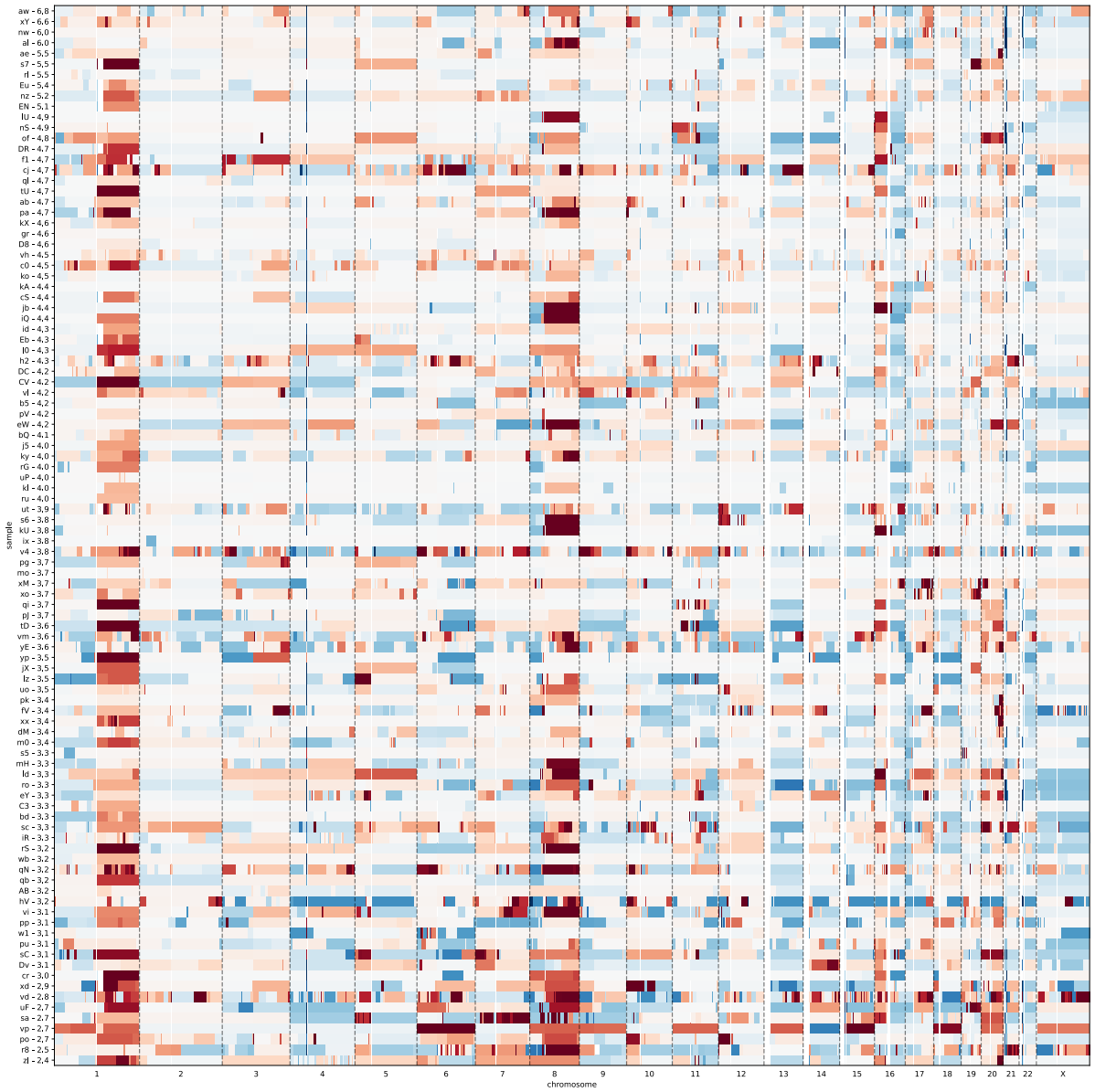
Appendix 12 The coverage graph (A) shows the relative copy number (RCN) segments scaled so each chromosome is uniformly sized, with the length of each segment on the x-axis being logged. The complexity score boxplots (B) show the distribution of the complexity scores across all chromosomes for each sample, with means indicated by the green diamond. The recursive partition line graph (C) shows the t p-value for the Mann Whitney U test for complexity score at and below each position > above for the sum (genome blue) and individual chromosome (orange). Data displayed is a random subsample (n=100) of the GEL breast cancer cohort.



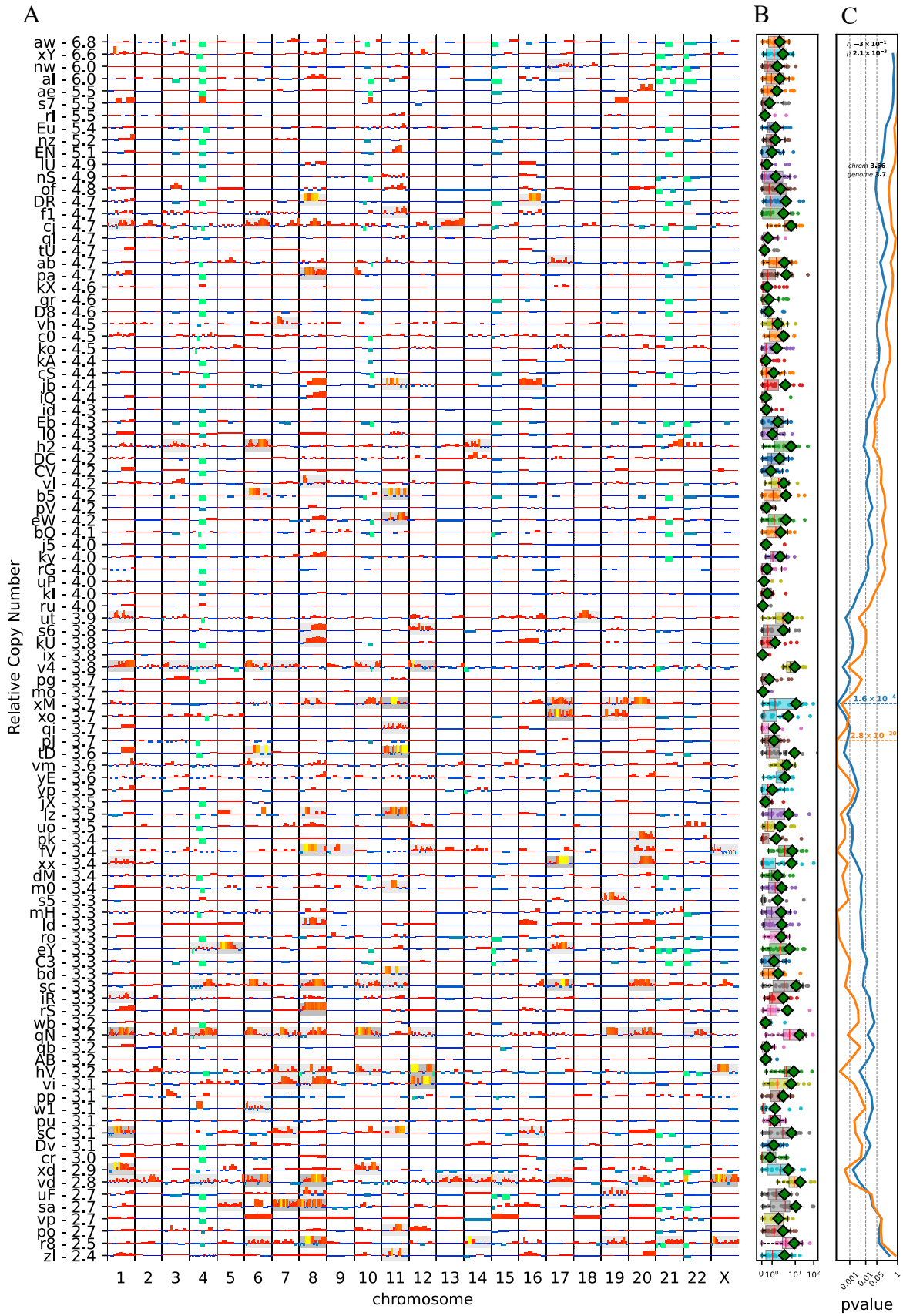
Appendix 13 Heatmap showing relative copy number segments from pcf analysis (input of 10kb coverage windows) clipped between -1.5 (blue loss) and 1.5 (red gain) with the Y-axis displaying ordered from longest telomeres (top) to shortest (bottom). Data displayed is a random subsample (n=100) of the GEL breast cancer cohort.



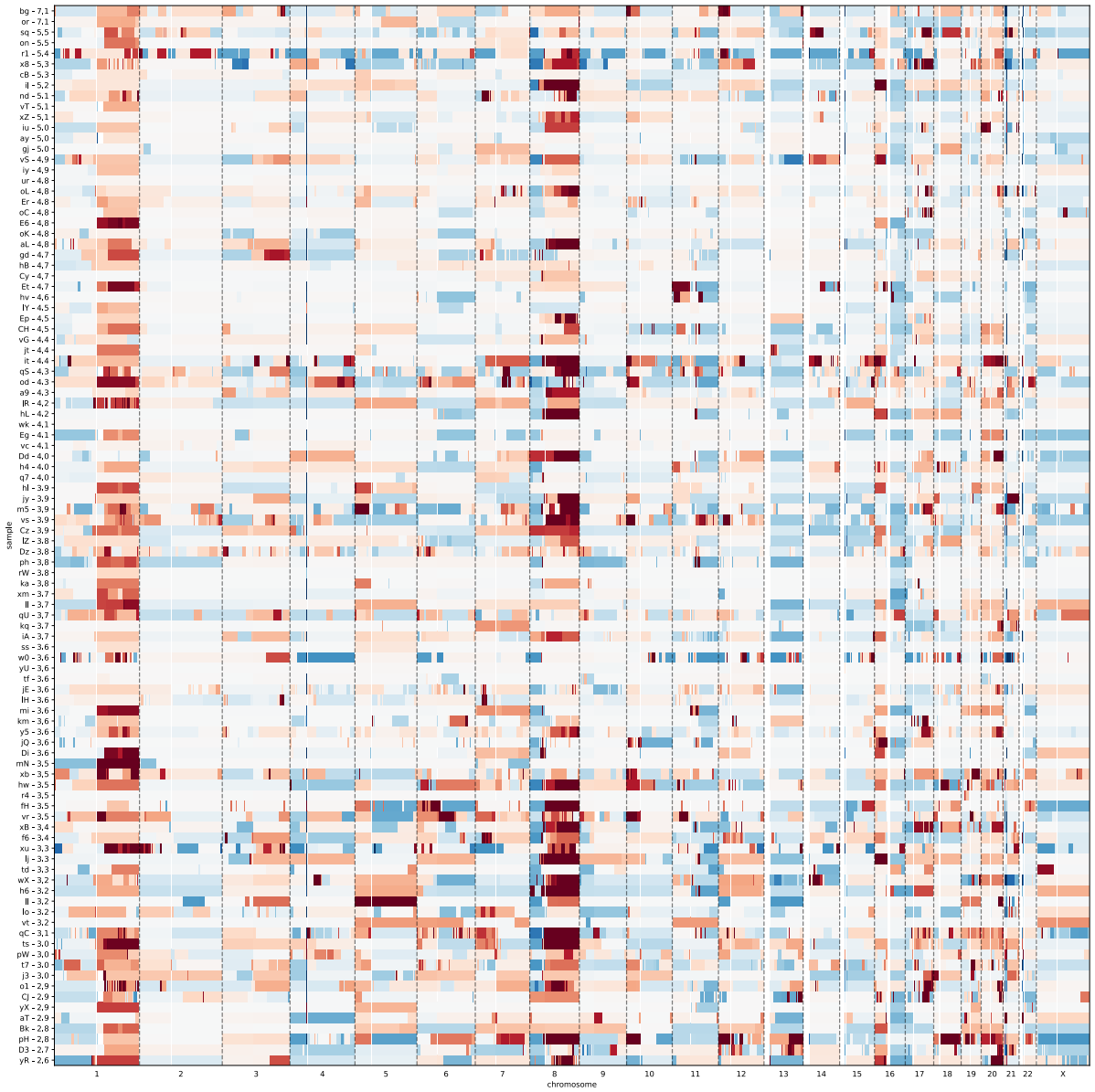
Appendix 14 The coverage graph (A) shows the relative copy number (RCN) segments scaled so each chromosome is uniformly sized, with the length of each segment on the x-axis being logged. The complexity score boxplots (B) show the distribution of the complexity scores across all chromosomes for each sample, with means indicated by the green diamond. The recursive partition line graph (C) shows the t p-value for the Mann Whitney U test for complexity score at and below each position > above for the sum (genome blue) and individual chromosome (orange). Data displayed is a random subsample (n=100) of the GEL breast cancer cohort.



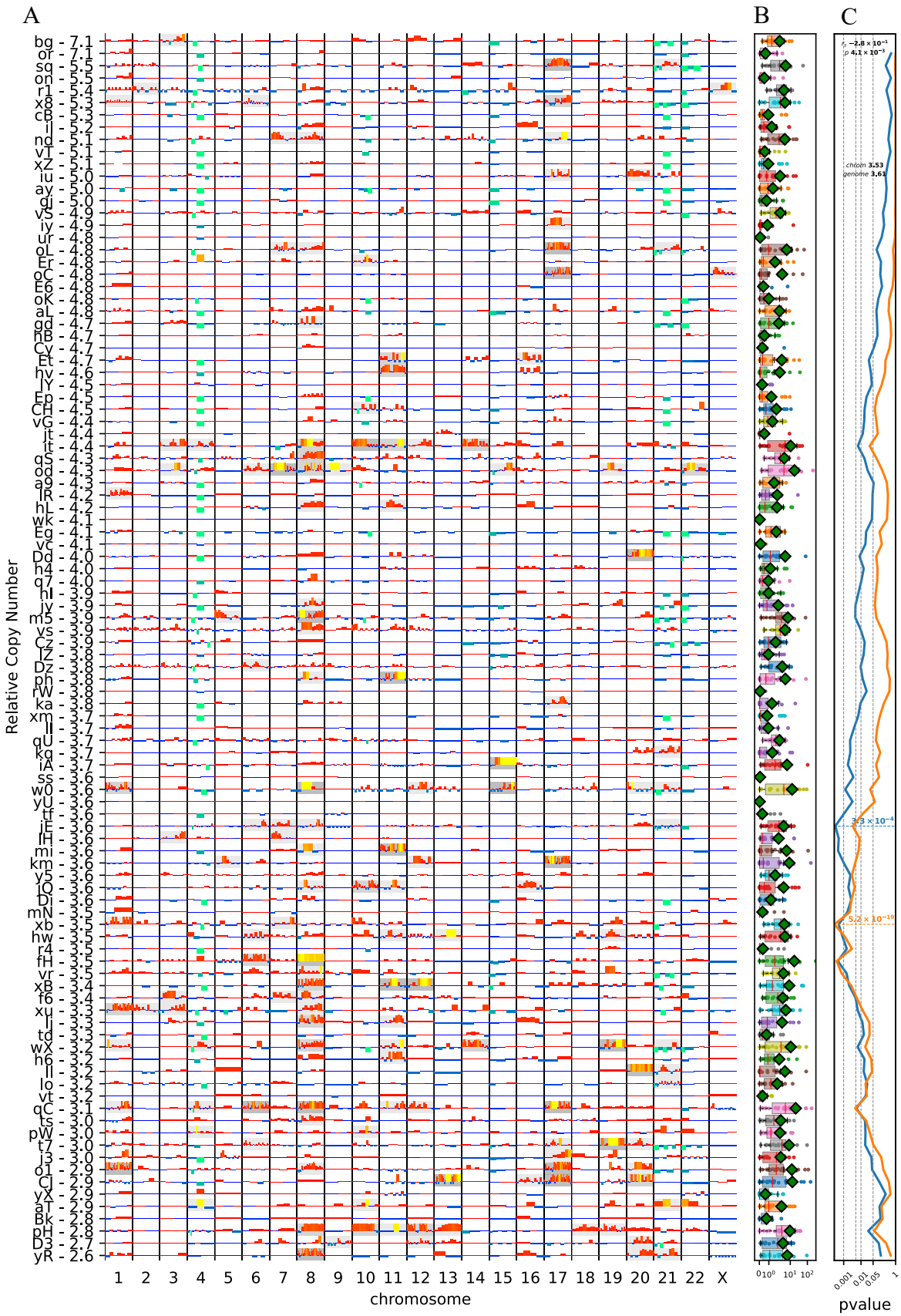
Appendix 15 Heatmap showing relative copy number segments from pcf analysis (input of 10kb coverage windows) clipped between -1.5 (blue loss) and 1.5 (red gain) with the Y-axis displaying ordered from longest telomeres (top) to shortest (bottom). Data displayed is a random subsample (n=100) of the GEL breast cancer cohort.



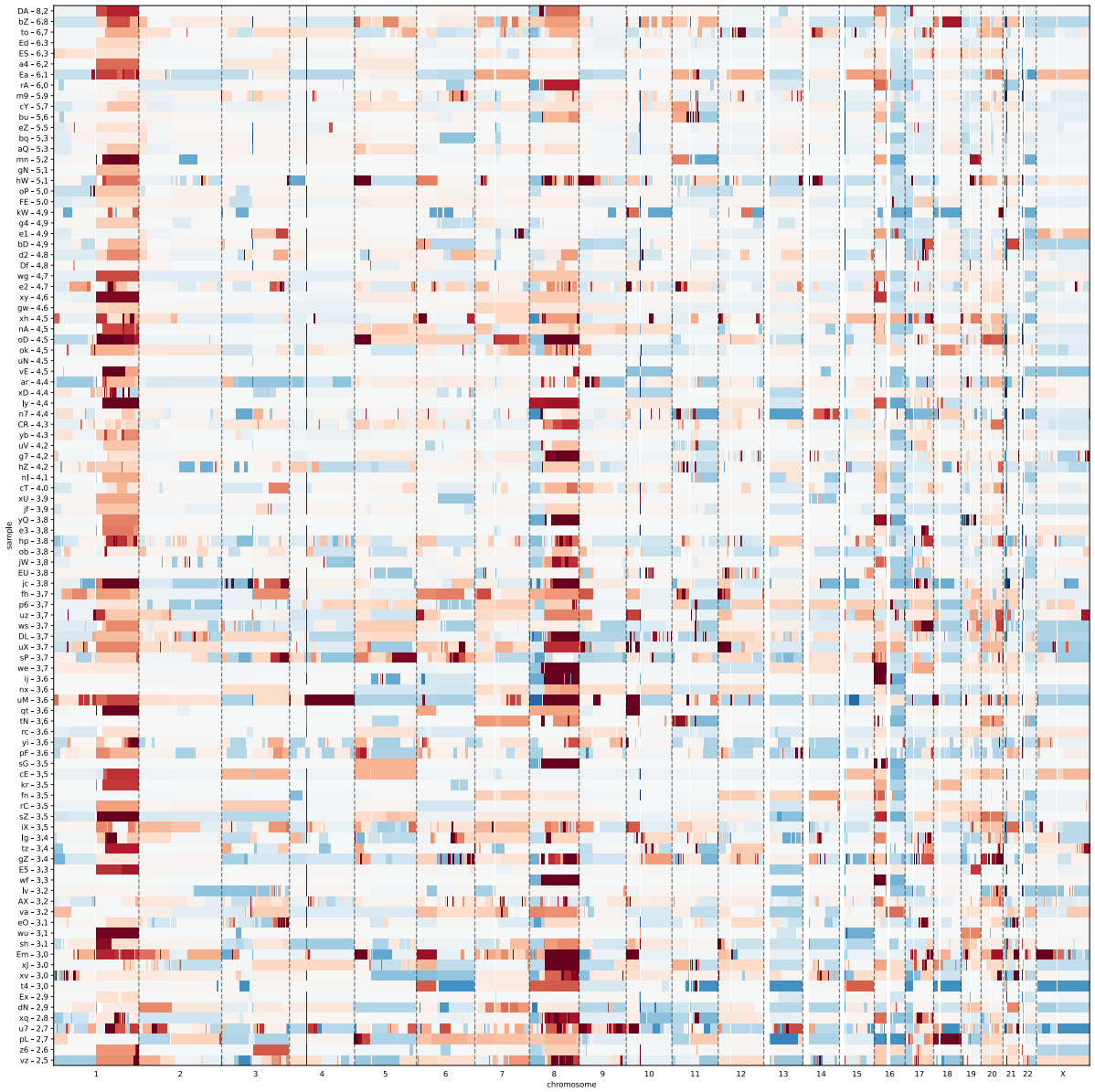
Appendix 16 The coverage graph (A) shows the relative copy number (RCN) segments scaled so each chromosome is uniformly sized, with the length of each segment on the x-axis being logged. The complexity score boxplots (B) show the distribution of the complexity scores across all chromosomes for each sample, with means indicated by the green diamond. The recursive partition line graph (C) shows the t p-value for the Mann Whitney U test for complexity score at and below each position > above for the sum (genome blue) and individual chromosome (orange). Data displayed is a random subsample (n=100) of the GEL breast cancer cohort.



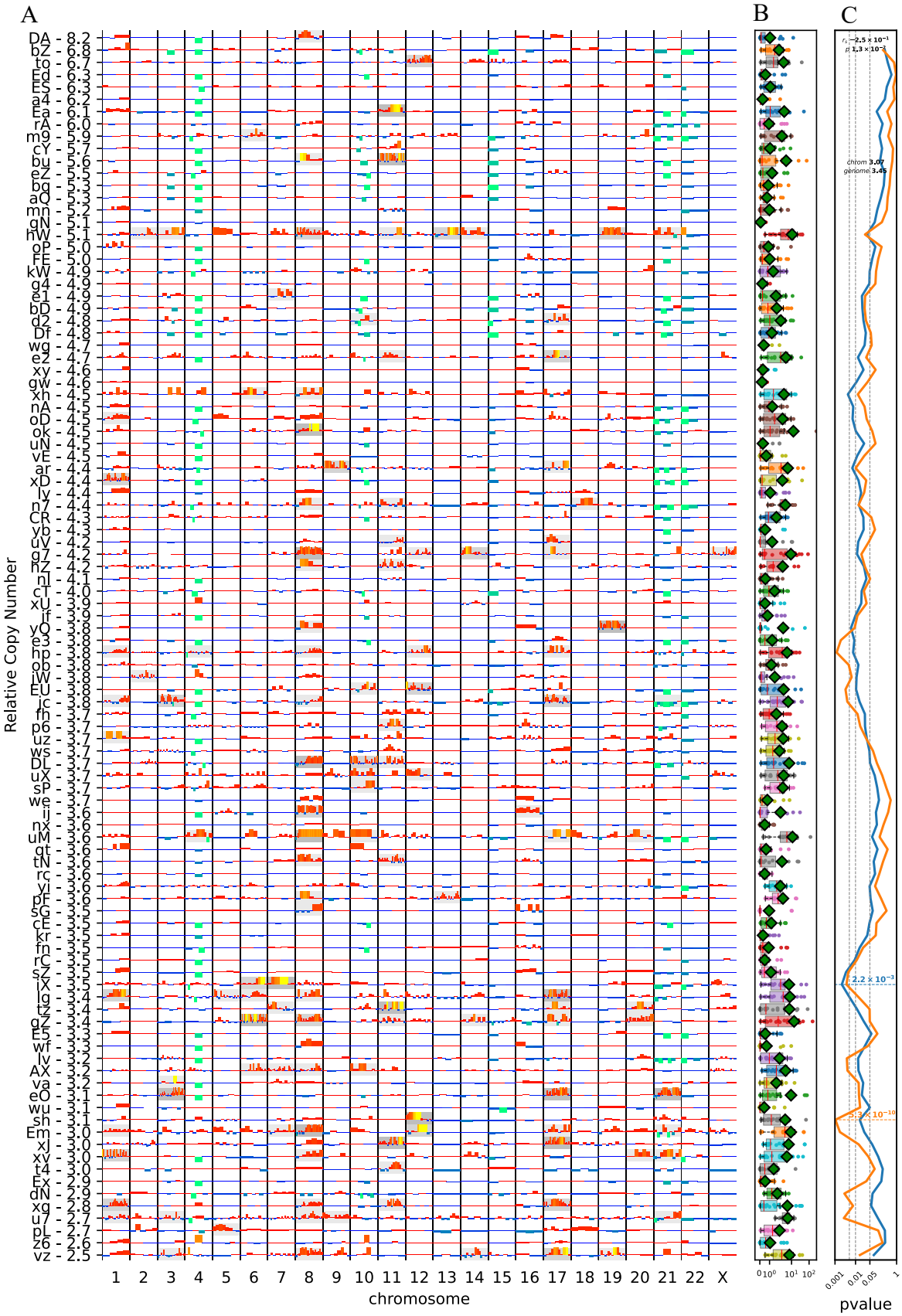
Appendix 17 Heatmap showing relative copy number segments from pcf analysis (input of 10kb coverage windows) clipped between -1.5 (blue loss) and 1.5 (red gain) with the Y-axis displaying ordered from longest telomeres (top) to shortest (bottom). Data displayed is a random subsample (n=100) of the GEL breast cancer cohort.



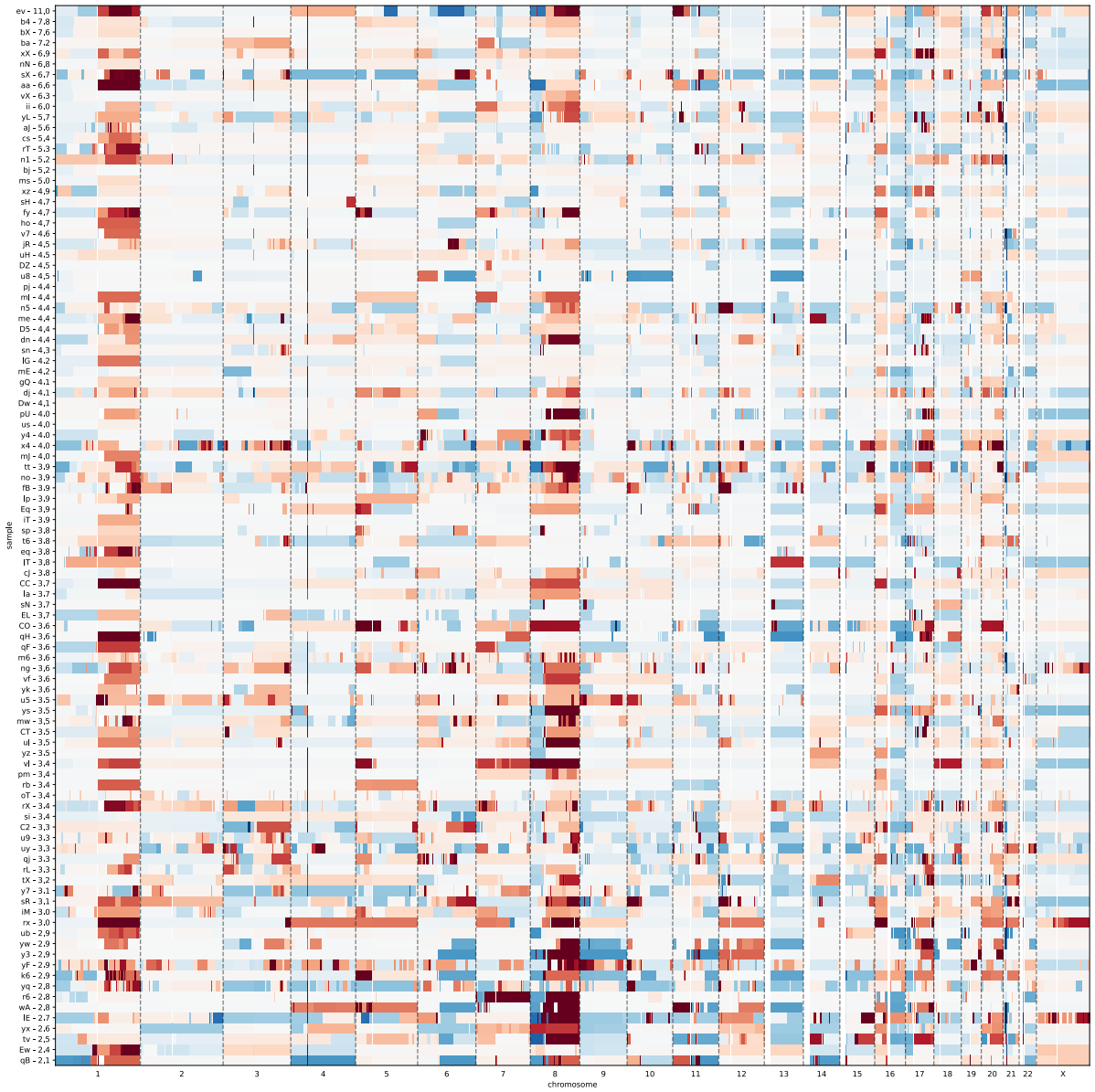
Appendix 18 The coverage graph (A) shows the relative copy number (RCN) segments scaled so each chromosome is uniformly sized, with the length of each segment on the x-axis being logged. The complexity score boxplots (B) show the distribution of the complexity scores across all chromosomes for each sample, with means indicated by the green diamond. The recursive partition line graph (C) shows the t p-value for the Mann Whitney U test for complexity score at and below each position > above for the sum (genome blue) and individual chromosome (orange). Data displayed is a random subsample (n=100) of the GEL breast cancer cohort.



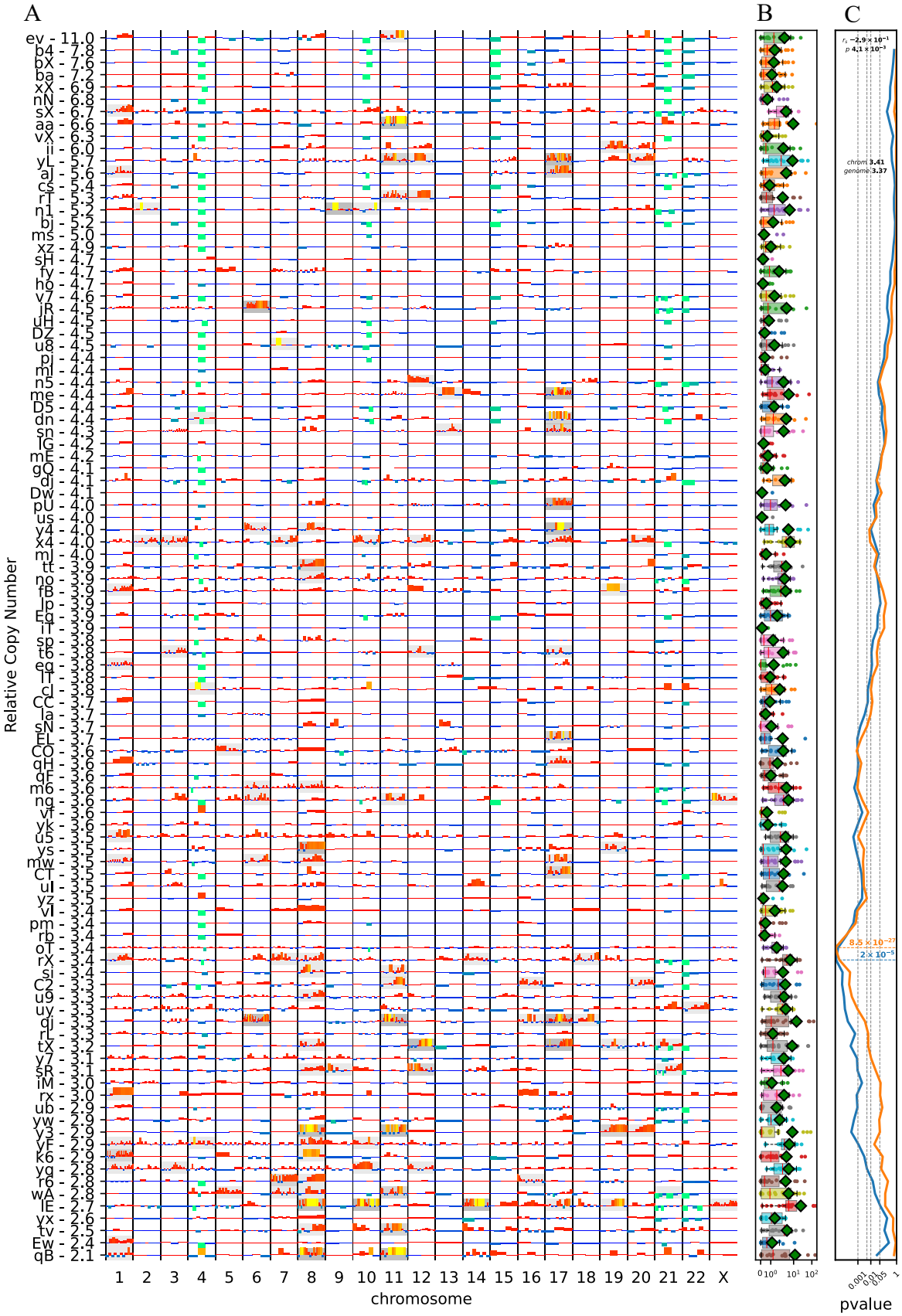
Appendix 19 Heatmap showing relative copy number segments from pcf analysis (input of 10kb coverage windows) clipped between -1.5 (blue loss) and 1.5 (red gain) with the Y-axis displaying ordered from longest telomeres (top) to shortest (bottom). Data displayed is a random subsample (n=100) of the GEL breast cancer cohort.



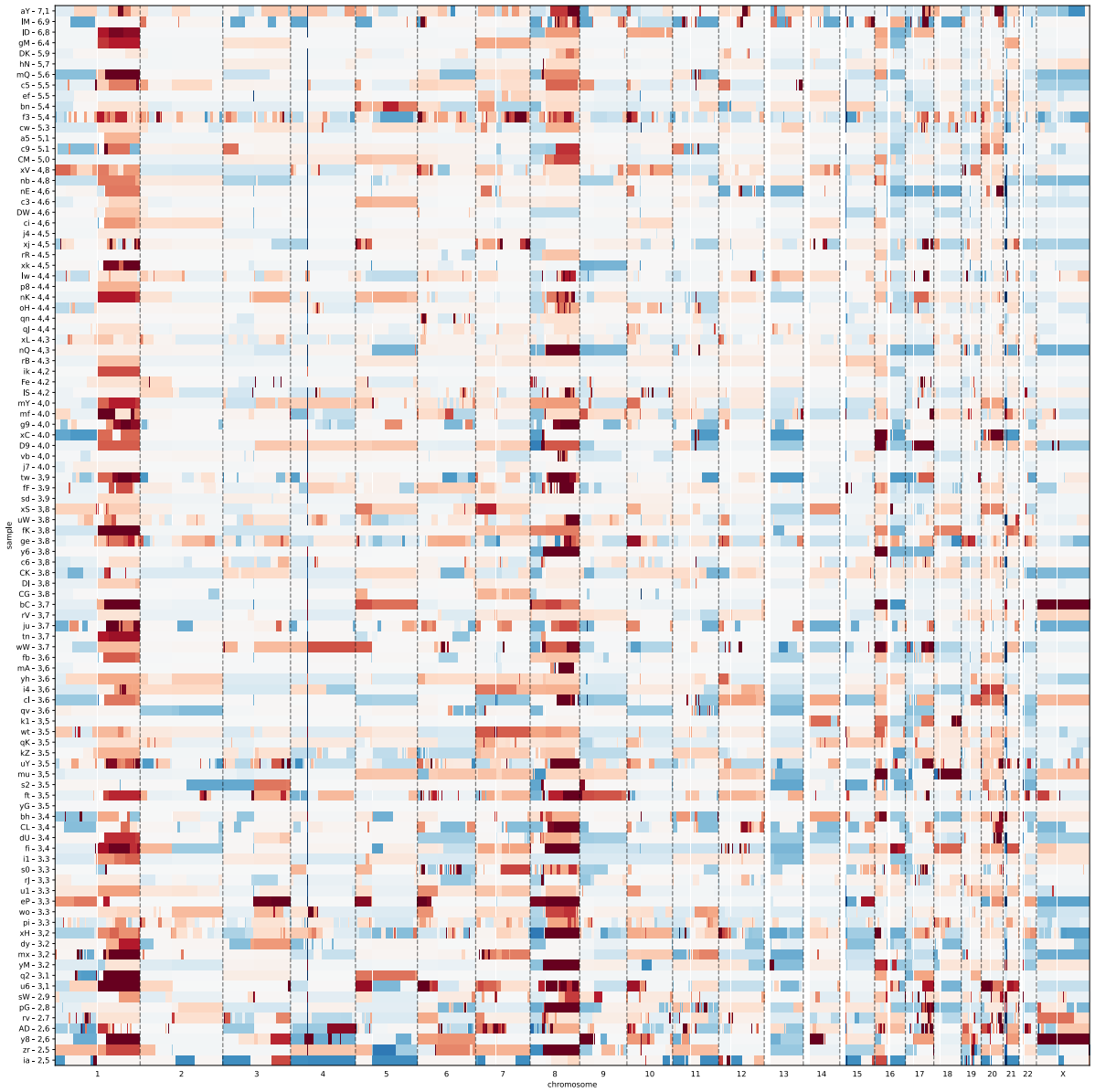
Appendix 20 The coverage graph (A) shows the relative copy number (RCN) segments scaled so each chromosome is uniformly sized, with the length of each segment on the x-axis being logged. The complexity score boxplots (B) show the distribution of the complexity scores across all chromosomes for each sample, with means indicated by the green diamond. The recursive partition line graph (C) shows the t p-value for the Mann Whitney U test for complexity score at and below each position > above for the sum (genome blue) and individual chromosome (orange). Data displayed is a random subsample (n=100) of the GEL breast cancer cohort.



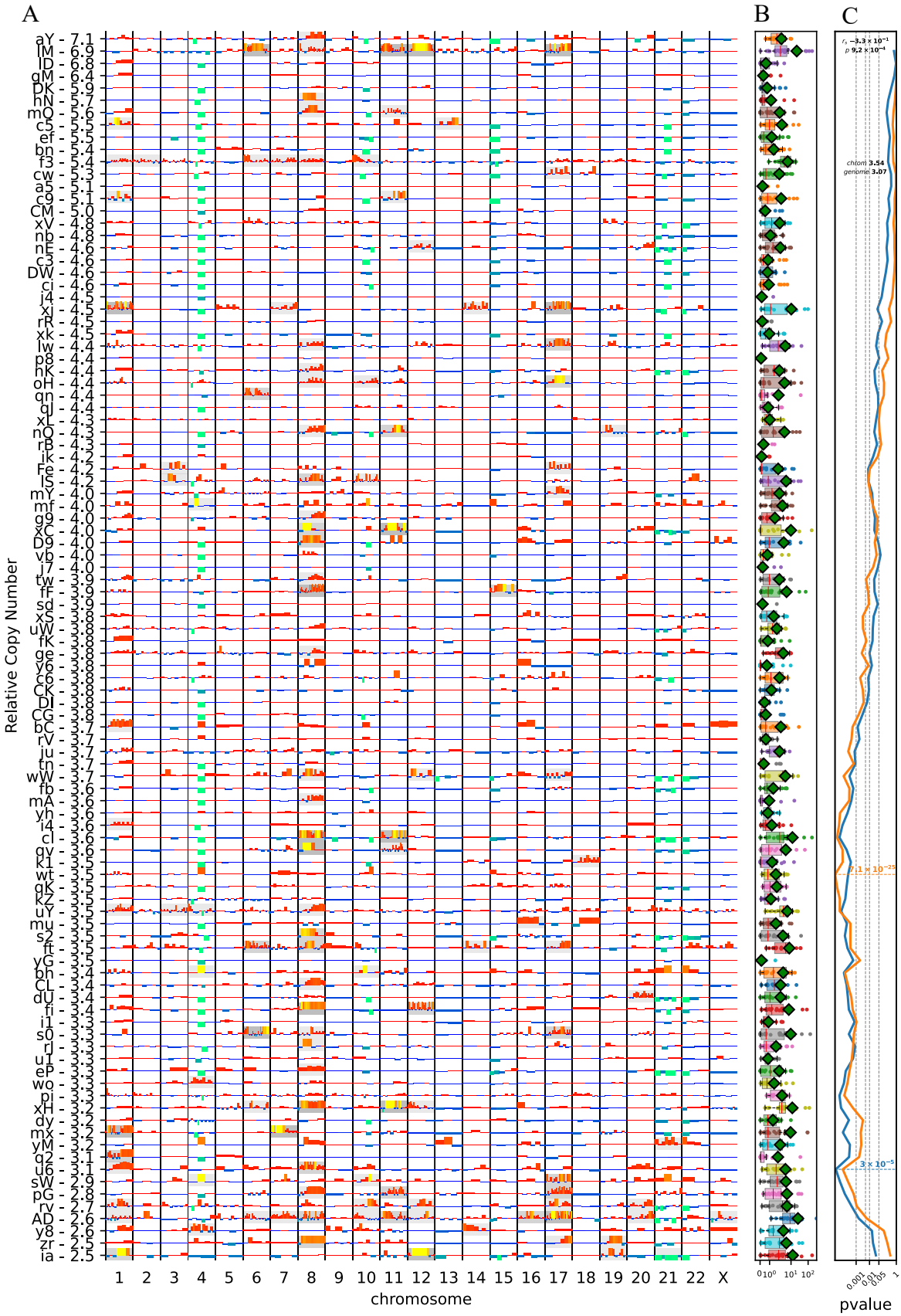
Appendix 21 Heatmap showing relative copy number segments from pcf analysis (input of 10kb coverage windows) clipped between -1.5 (blue loss) and 1.5 (red gain) with the Y-axis displaying ordered from longest telomeres (top) to shortest (bottom). Data displayed is a random subsample (n=100) of the GEL breast cancer cohort.



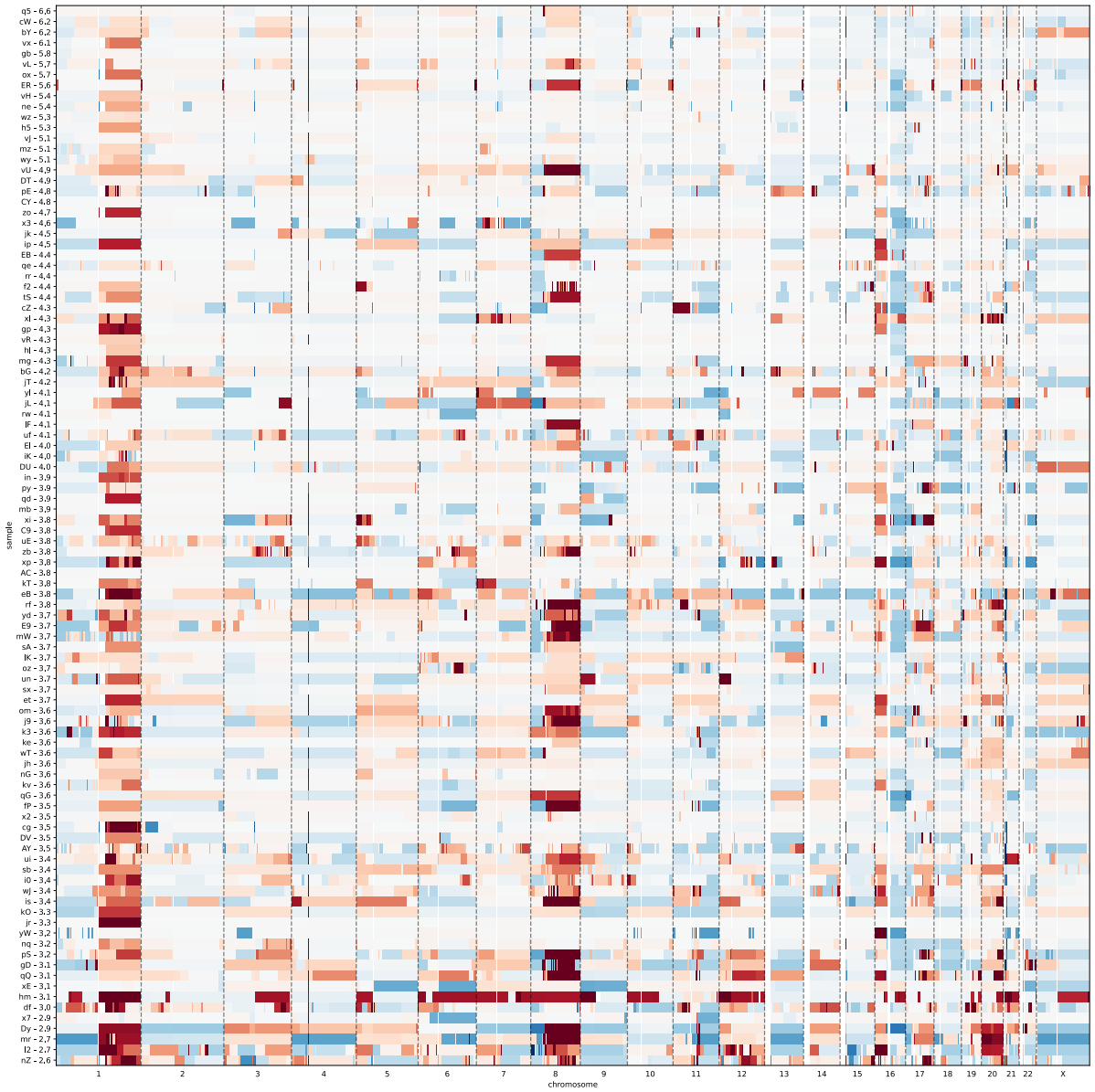
Appendix 22 The coverage graph (A) shows the relative copy number (RCN) segments scaled so each chromosome is uniformly sized, with the length of each segment on the x-axis being logged. The complexity score boxplots (B) show the distribution of the complexity scores across all chromosomes for each sample, with means indicated by the green diamond. The recursive partition line graph (C) shows the t p-value for the Mann Whitney U test for complexity score at and below each position > above for the sum (genome blue) and individual chromosome (orange). Data displayed is a random subsample (n=100) of the GEL breast cancer cohort.



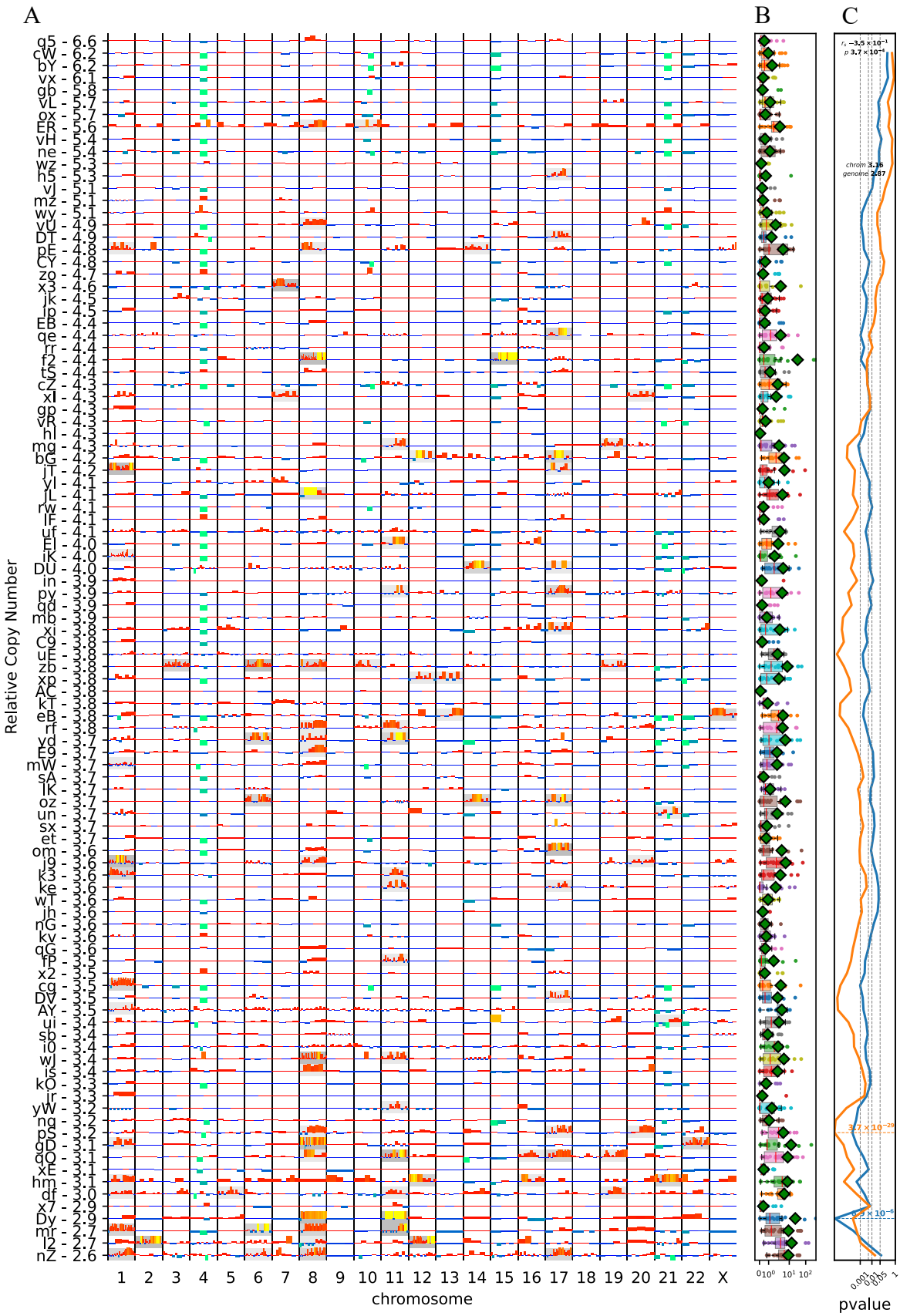
Appendix 23 Heatmap showing relative copy number segments from pcf analysis (input of 10kb coverage windows) clipped between -1.5 (blue loss) and 1.5 (red gain) with the Y-axis displaying ordered from longest telomeres (top) to shortest (bottom). Data displayed is a random subsample (n=100) of the GEL breast cancer cohort



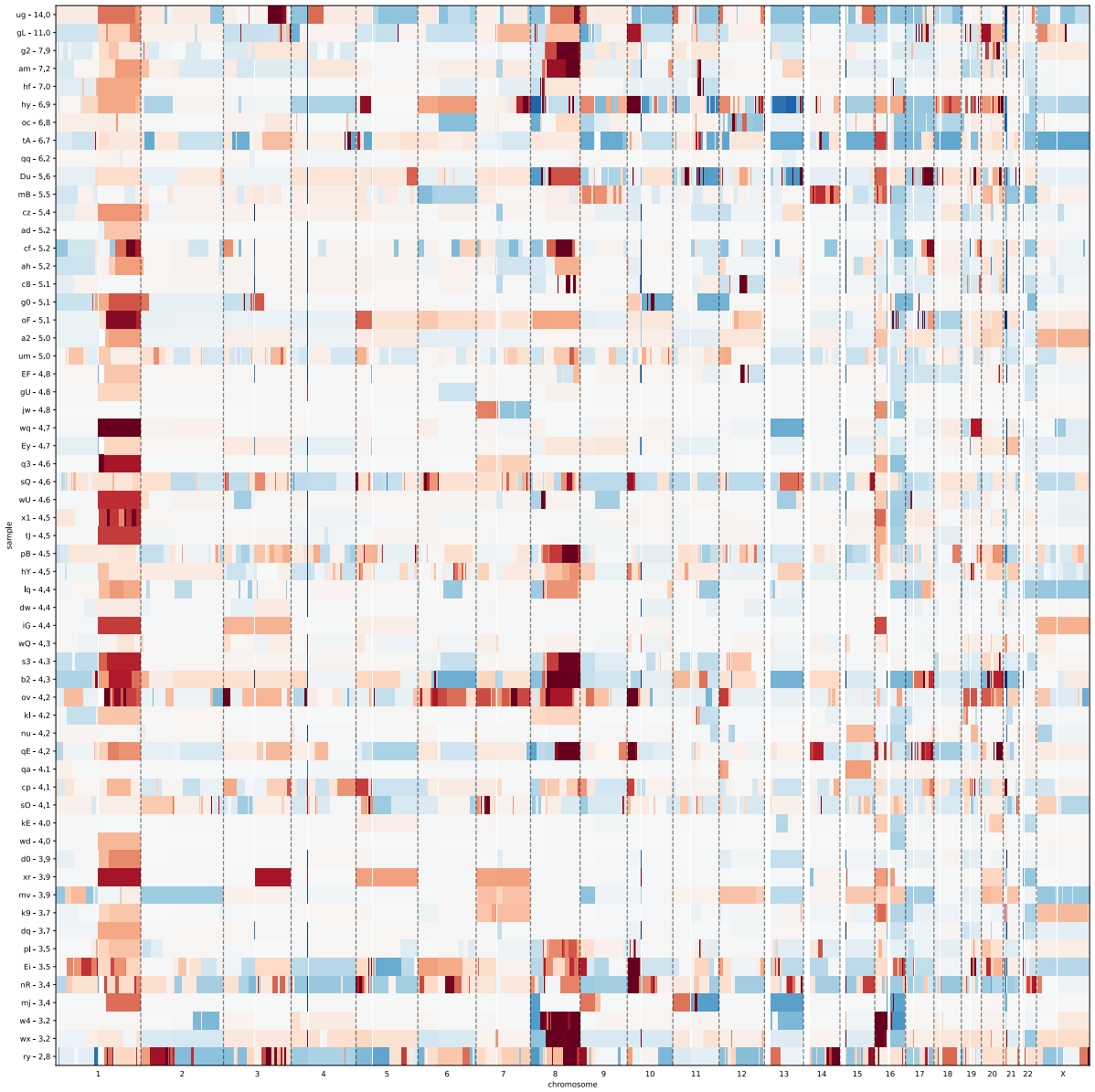
Appendix 24 The coverage graph (A) shows the relative copy number (RCN) segments scaled so each chromosome is uniformly sized, with the length of each segment on the x-axis being logged. The complexity score boxplots (B) show the distribution of the complexity scores across all chromosomes for each sample, with means indicated by the green diamond. The recursive partition line graph (C) shows the t p-value for the Mann Whitney U test for complexity score at and below each position > above for the sum (genome blue) and individual chromosome (orange). Data displayed is a random subsample (n=100) of the GEL breast cancer cohort.



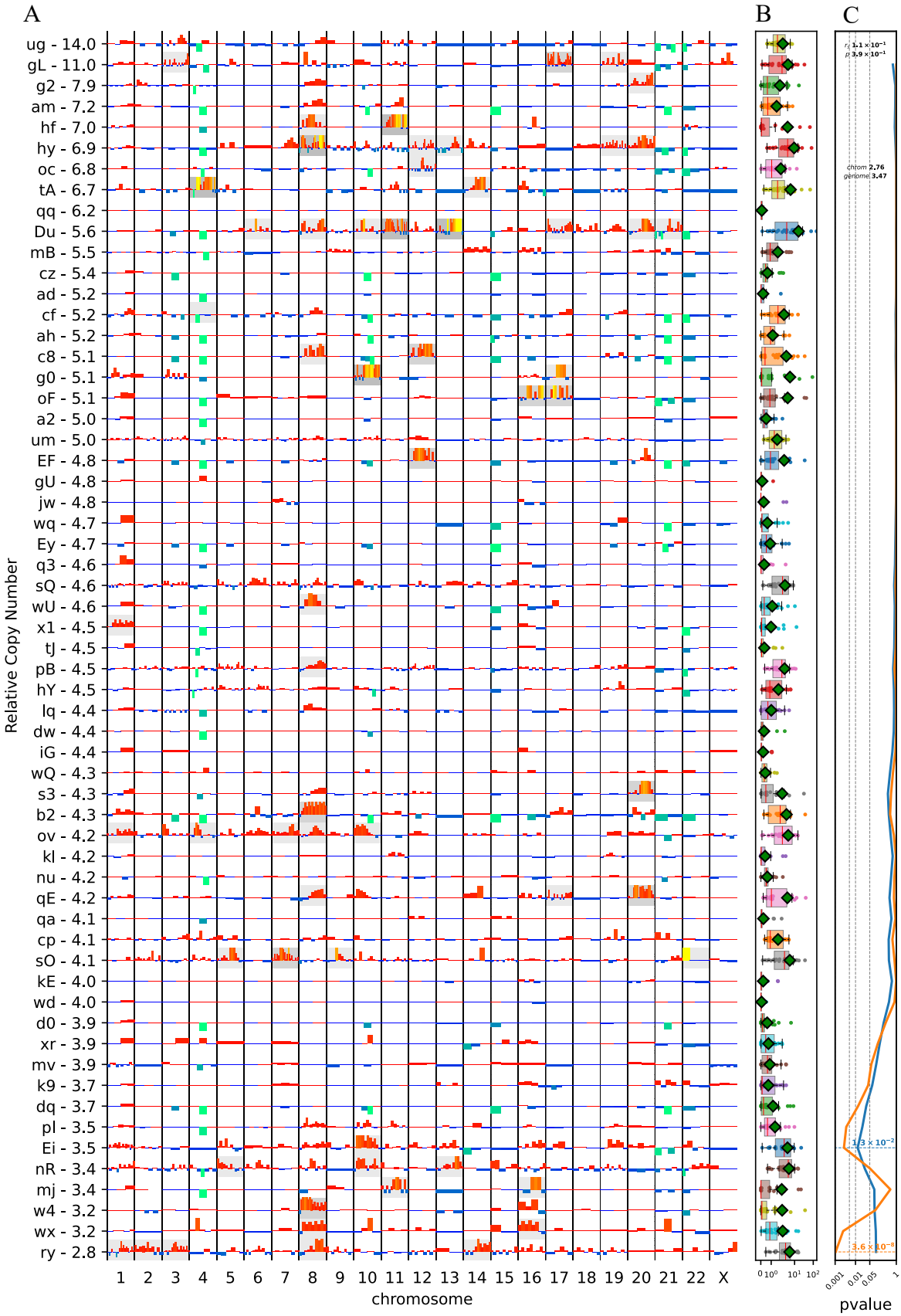
Appendix 25 Heatmap showing relative copy number segments from pcf analysis (input of 10kb coverage windows) clipped between -1.5 (blue loss) and 1.5 (red gain) with the Y-axis displaying ordered from longest telomeres (top) to shortest (bottom). Data displayed is a random subsample (n=100) of the GEL breast cancer cohort.



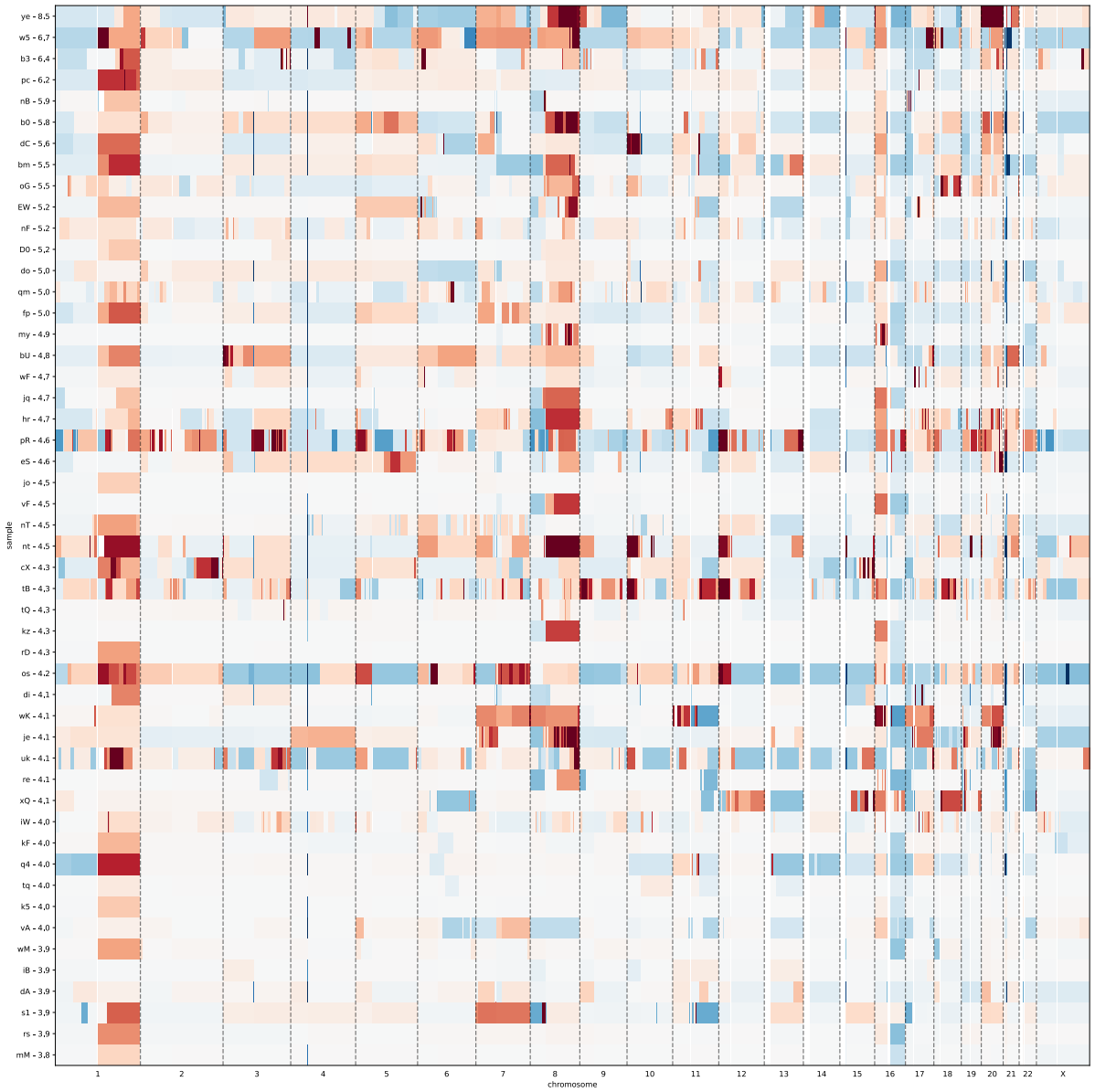
Appendix 26 The coverage graph (A) shows the relative copy number (RCN) segments scaled so each chromosome is uniformly sized, with the length of each segment on the x-axis being logged. The complexity score boxplots (B) show the distribution of the complexity scores across all chromosomes for each sample, with means indicated by the green diamond. The recursive partition line graph (C) shows the t p-value for the Mann Whitney U test for complexity score at and below each position > above for the sum (genome blue) and individual chromosome (orange). Data displayed is a random subsample (n=100) of the GEL breast cancer cohort.



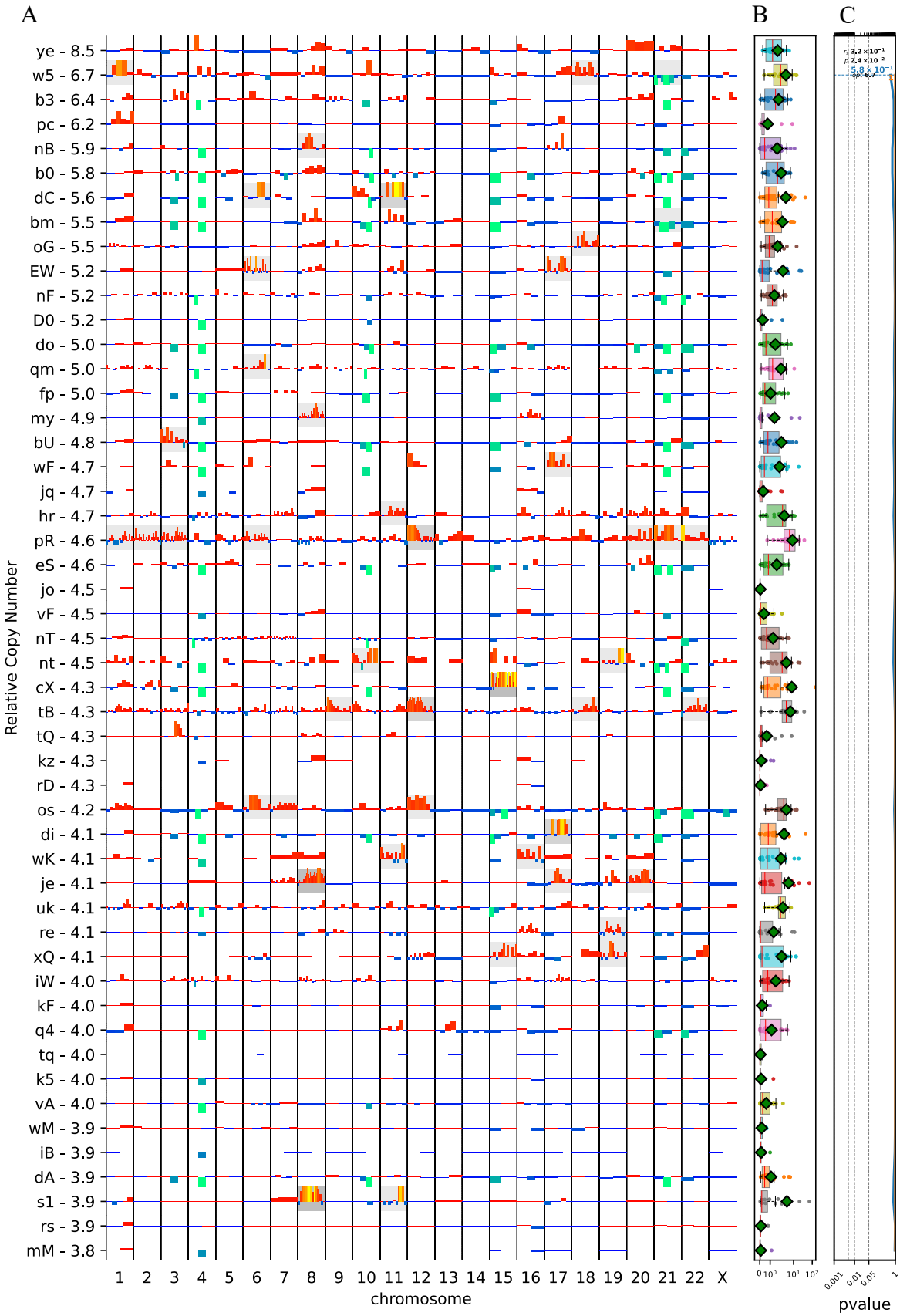
Appendix 27 Heatmap showing relative copy number segments from pcf analysis (input of 10kb coverage windows) clipped between -1.5 (blue loss) and 1.5 (red gain) with the Y-axis displaying ordered from longest telomeres (top) to shortest (bottom). Data displayed is a random subsample (n=50) of the GEL breast cancer cohort.



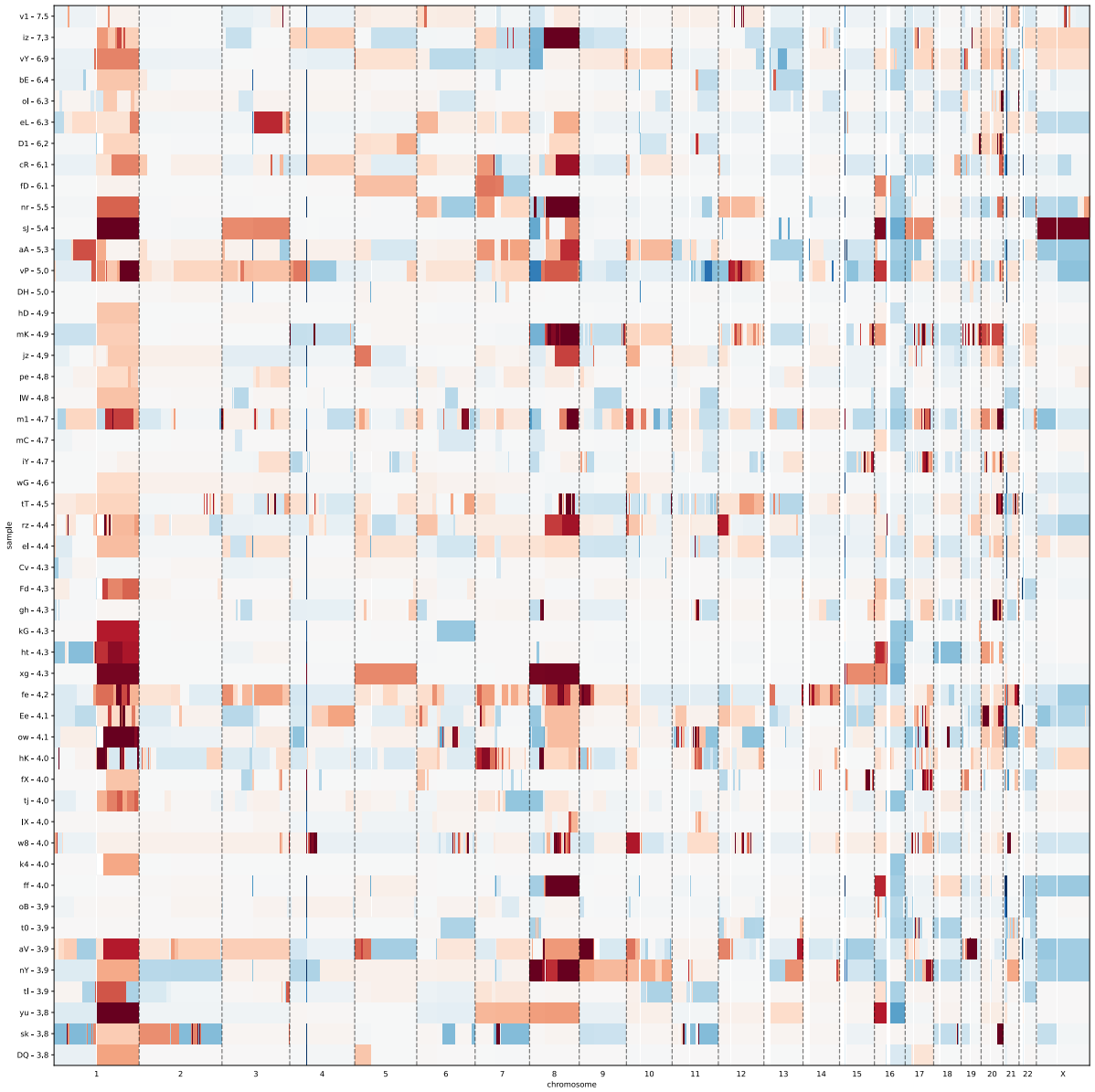
Appendix 28 The coverage graph (A) shows the relative copy number (RCN) segments scaled so each chromosome is uniformly sized, with the length of each segment on the x-axis being logged. The complexity score boxplots (B) show the distribution of the complexity scores across all chromosomes for each sample, with means indicated by the green diamond. The recursive partition line graph (C) shows the t p-value for the Mann Whitney U test for complexity score at and below each position > above for the sum (genome blue) and individual chromosome (orange). Data displayed is a random subsample (n=50) of the GEL breast cancer cohort.



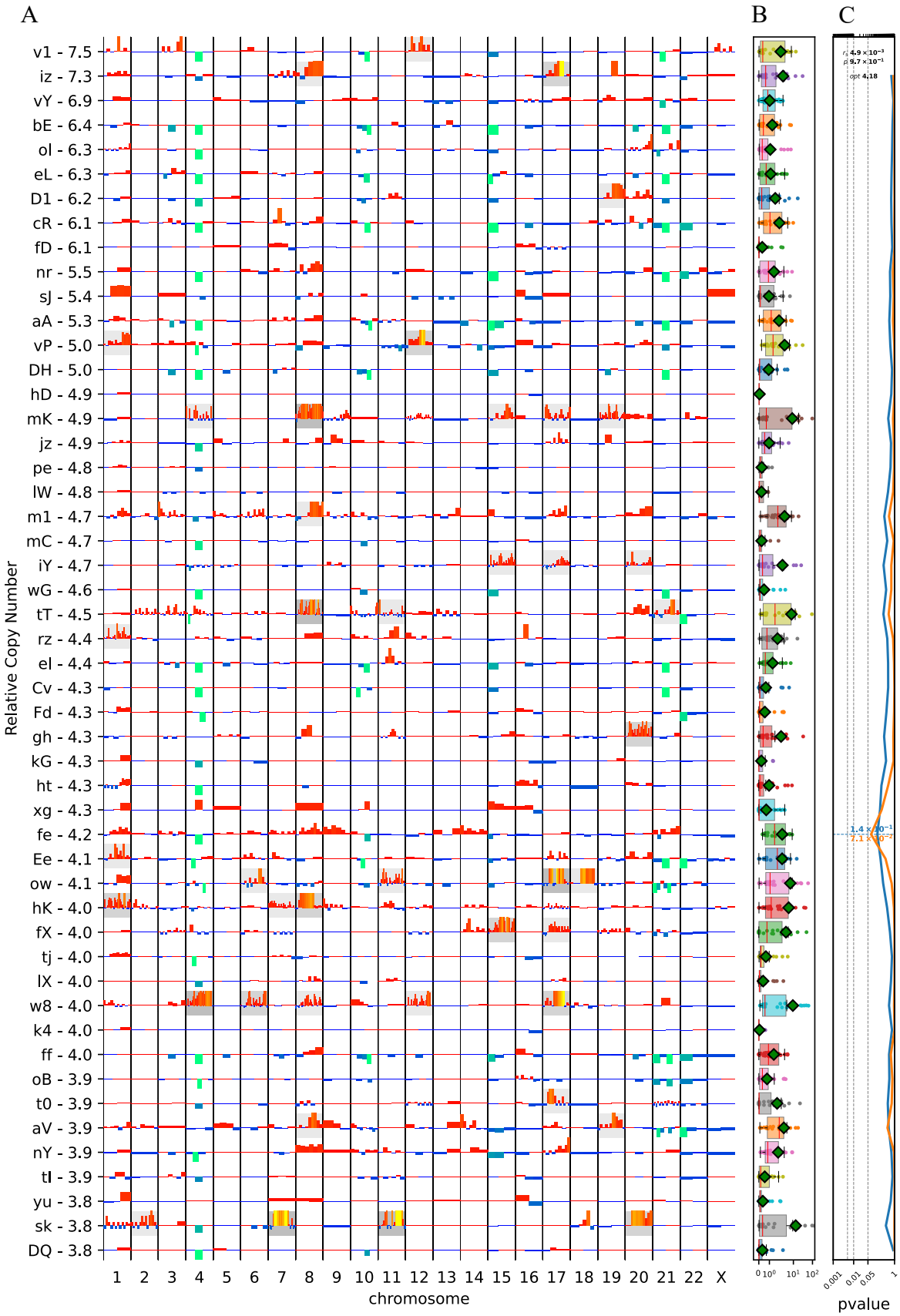
Appendix 29 Heatmap showing relative copy number segments from pcf analysis (input of 10kb coverage windows) clipped between -1.5 (blue loss) and 1.5 (red gain) with the Y-axis displaying ordered from longest telomeres (top) to shortest (bottom). Data displayed is a random subsample (n=50) of the GEL breast cancer cohort.



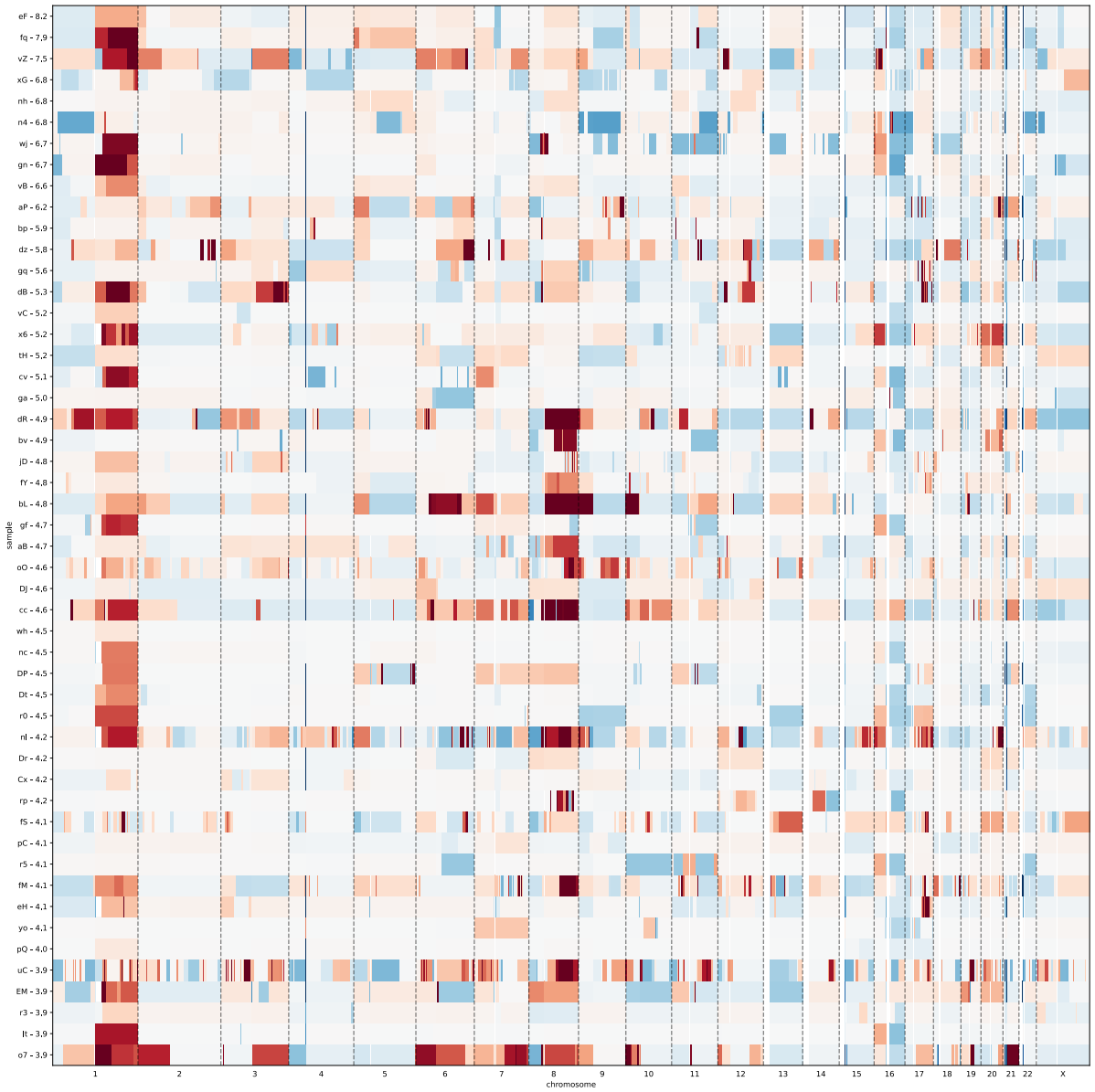
Appendix 30 The coverage graph (A) shows the relative copy number (RCN) segments scaled so each chromosome is uniformly sized, with the length of each segment on the x-axis being logged. The complexity score boxplots (B) show the distribution of the complexity scores across all chromosomes for each sample, with means indicated by the green diamond. The recursive partition line graph (C) shows the t p-value for the Mann Whitney U test for complexity score at and below each position > above for the sum (genome blue) and individual chromosome (orange). Data displayed is a random subsample (n=50) of the GEL breast cancer cohort.



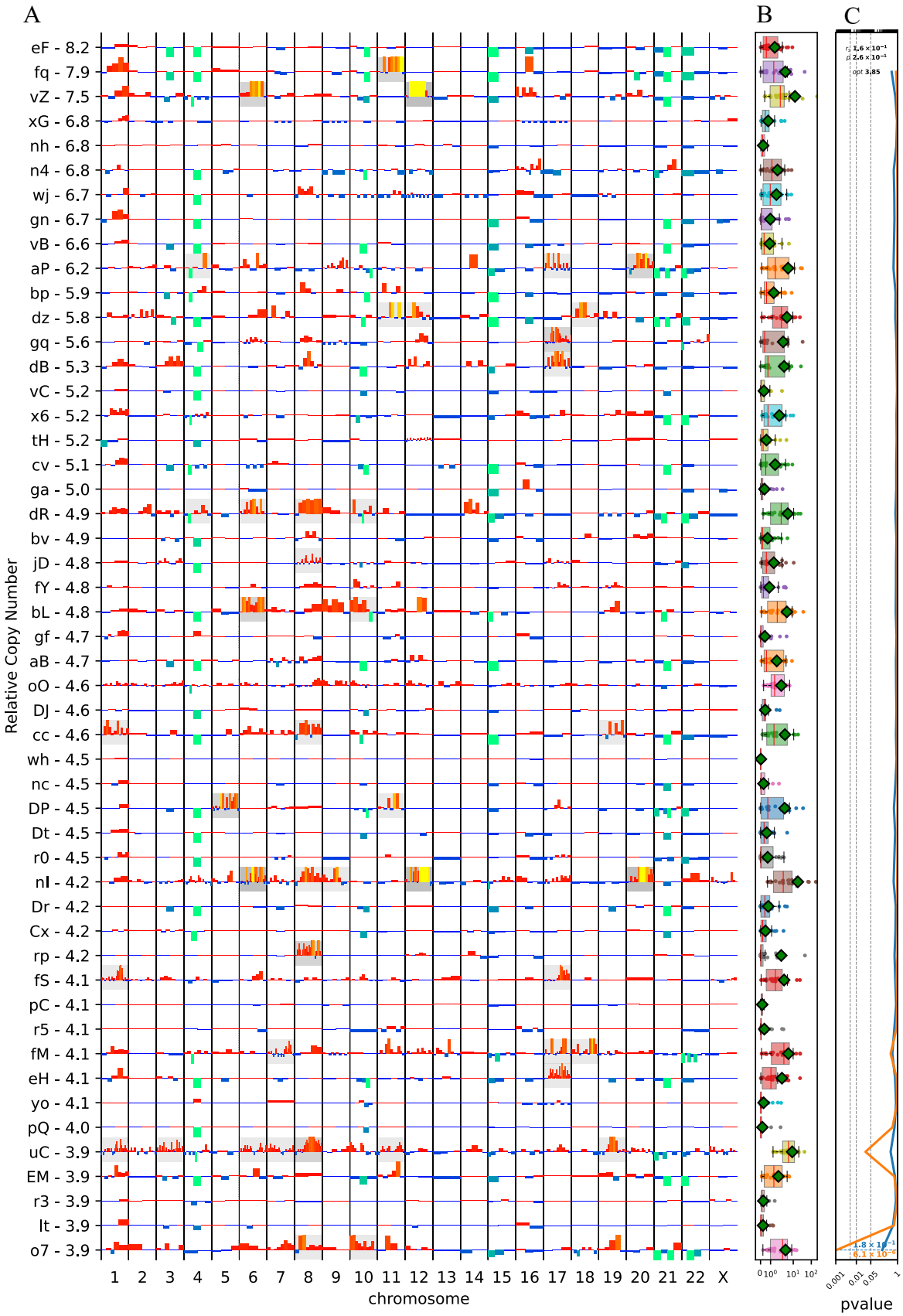
Appendix 31 Heatmap showing relative copy number segments from pcf analysis (input of 10kb coverage windows) clipped between -1.5 (blue loss) and 1.5 (red gain) with the Y-axis displaying ordered from longest telomeres (top) to shortest (bottom). Data displayed is a random subsample (n=50) of the GEL breast cancer cohort.



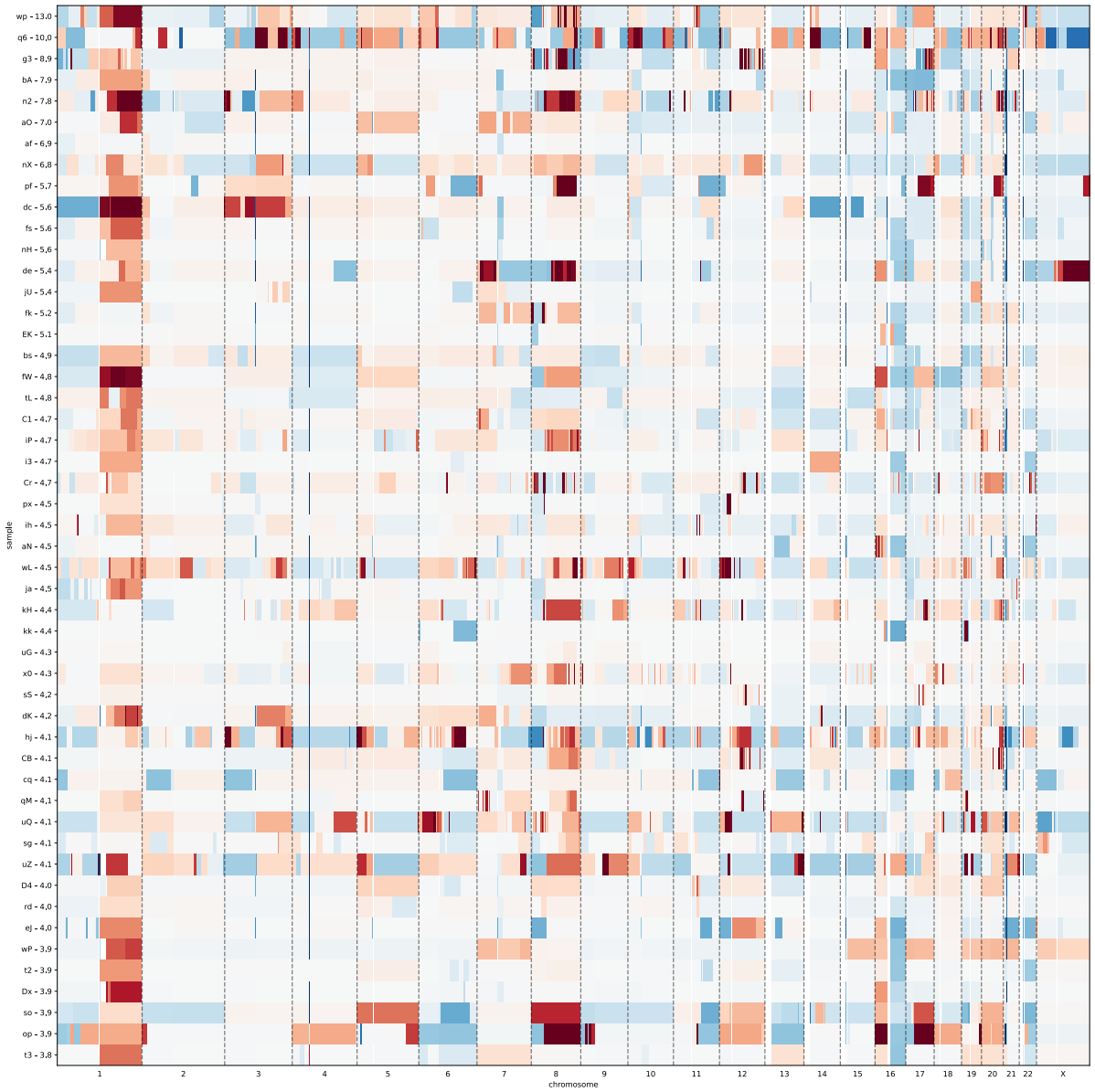
Appendix 32 The coverage graph (A) shows the relative copy number (RCN) segments scaled so each chromosome is uniformly sized, with the length of each segment on the x-axis being logged. The complexity score boxplots (B) show the distribution of the complexity scores across all chromosomes for each sample, with means indicated by the green diamond. The recursive partition line graph (C) shows the t p-value for the Mann Whitney U test for complexity score at and below each position > above for the sum (genome blue) and individual chromosome (orange). Data displayed is a random subsample (n=50) of the GEL breast cancer cohort.



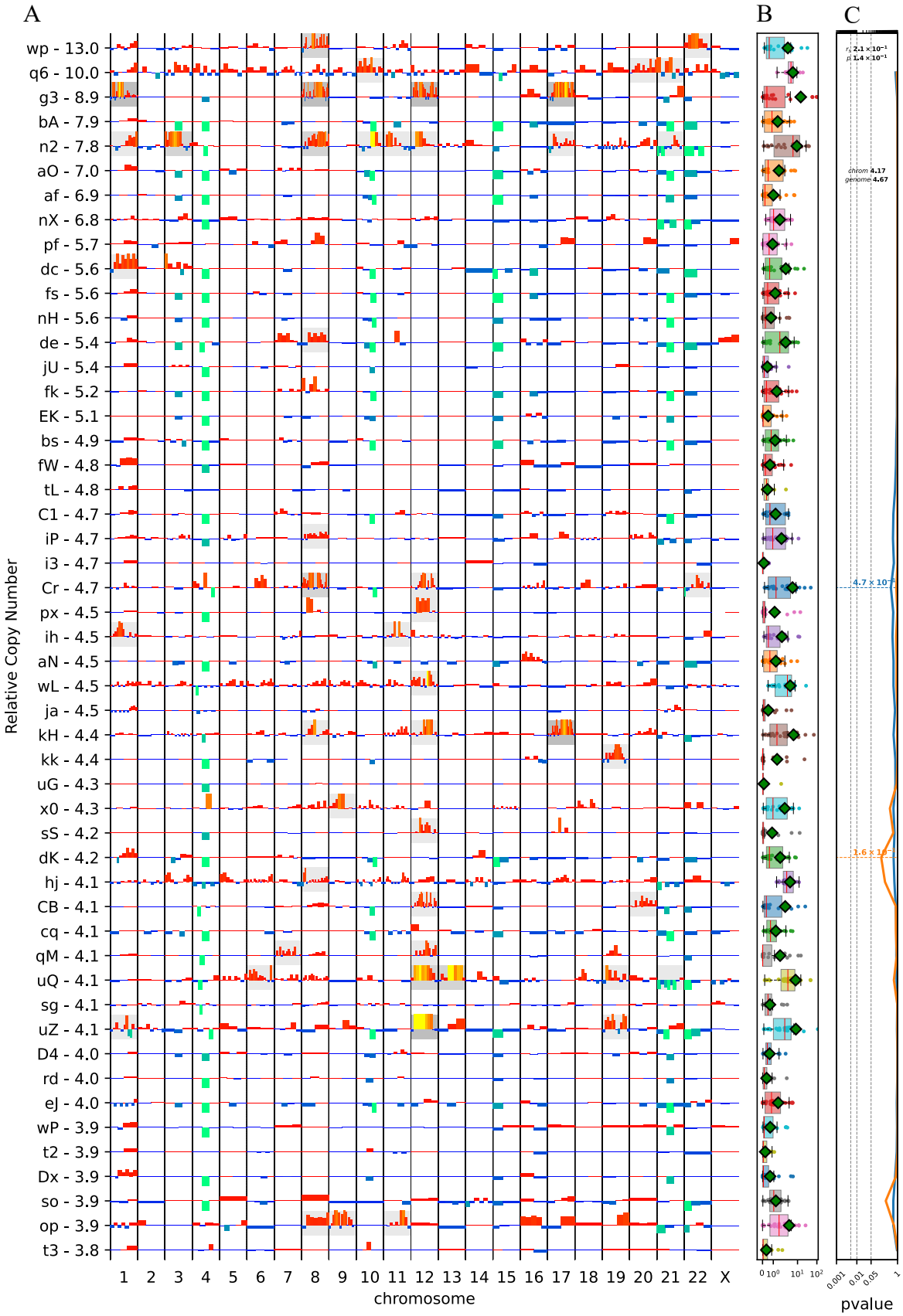
Appendix 33 Heatmap showing relative copy number segments from pcf analysis (input of 10kb coverage windows) clipped between -1.5 (blue loss) and 1.5 (red gain) with the Y-axis displaying ordered from longest telomeres (top) to shortest (bottom). Data displayed is a random subsample (n=50) of the GEL breast cancer cohort.



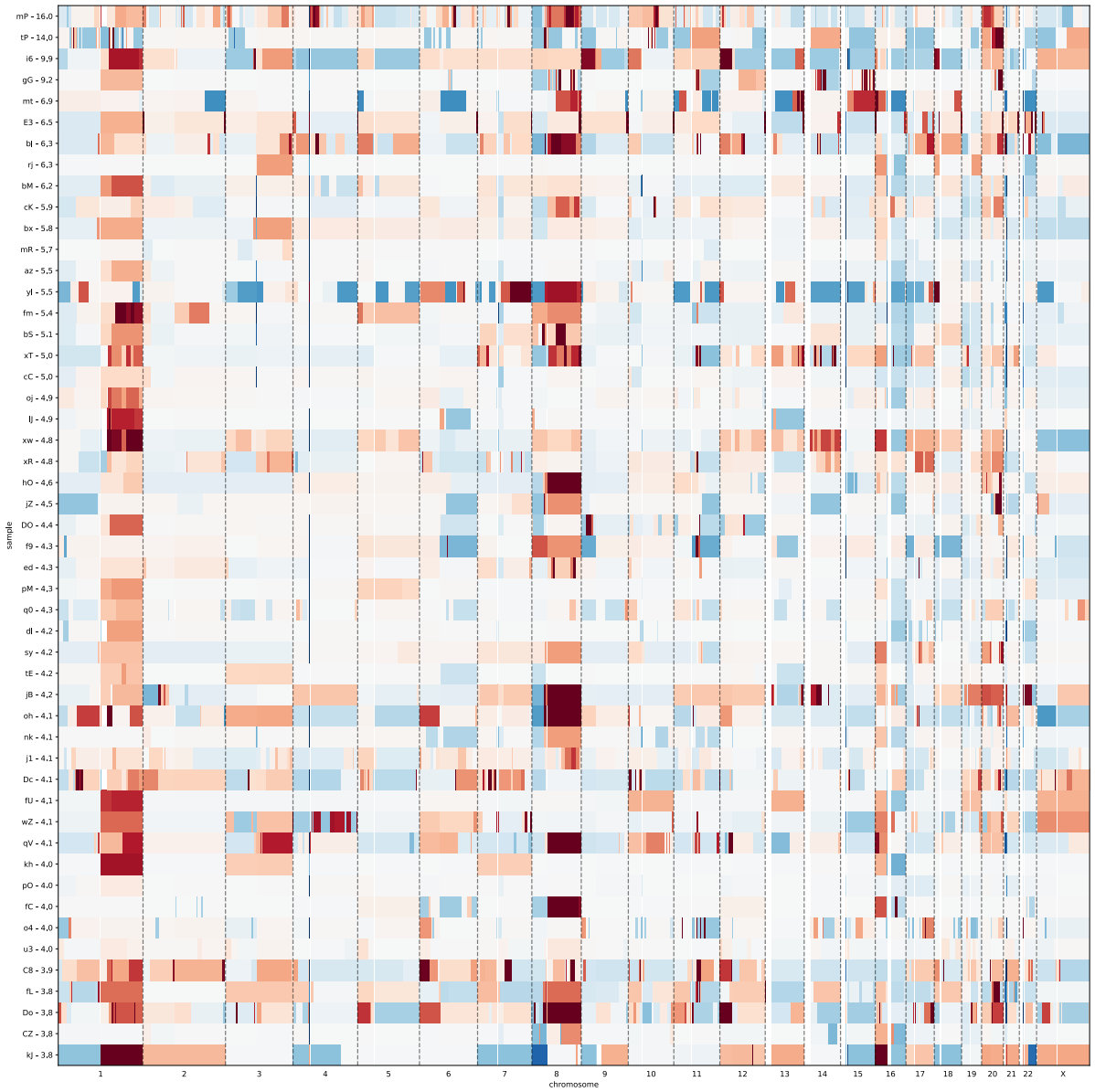
Appendix 34 The coverage graph (A) shows the relative copy number (RCN) segments scaled so each chromosome is uniformly sized, with the length of each segment on the x-axis being logged. The complexity score boxplots (B) show the distribution of the complexity scores across all chromosomes for each sample, with means indicated by the green diamond. The recursive partition line graph (C) shows the t p-value for the Mann Whitney U test for complexity score at and below each position > above for the sum (genome blue) and individual chromosome (orange). Data displayed is a random subsample (n=50) of the GEL breast cancer cohort.



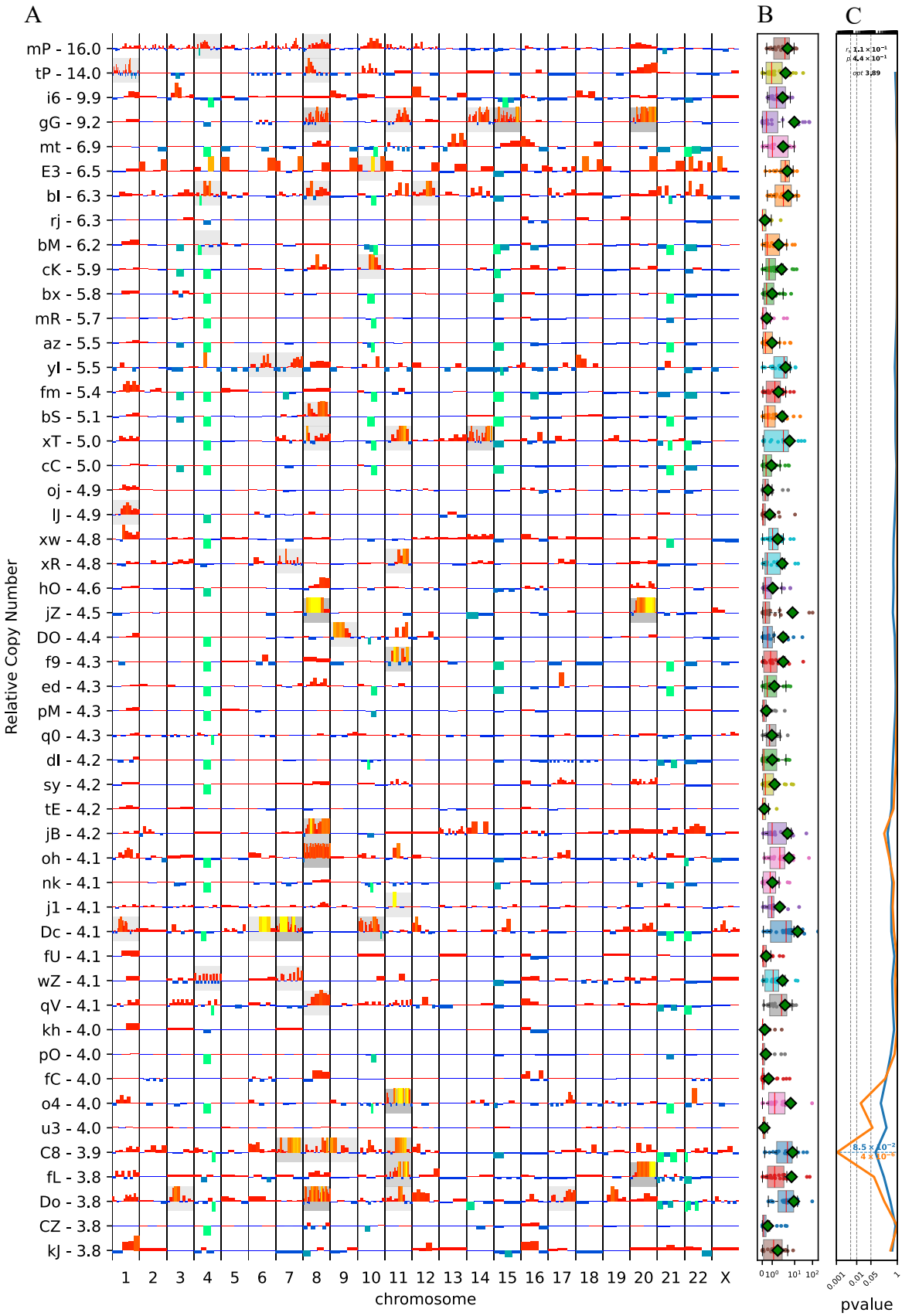
Appendix 35 Heatmap showing relative copy number segments from pcf analysis (input of 10kb coverage windows) clipped between -1.5 (blue loss) and 1.5 (red gain) with the Y-axis displaying ordered from longest telomeres (top) to shortest (bottom). Data displayed is a random subsample (n=50) of the GEL breast cancer cohort.



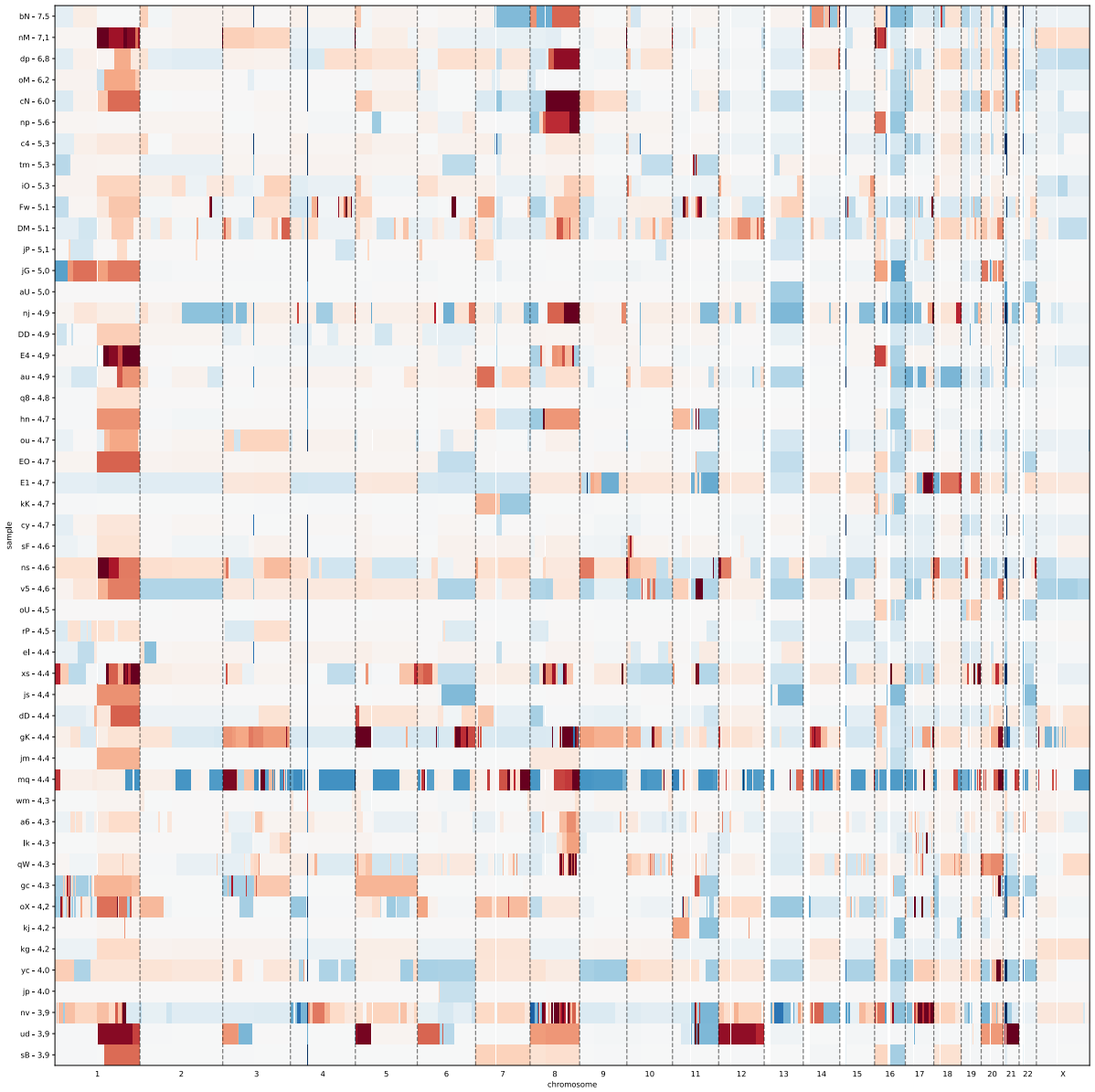
Appendix 36 The coverage graph (A) shows the relative copy number (RCN) segments scaled so each chromosome is uniformly sized, with the length of each segment on the x-axis being logged. The complexity score boxplots (B) show the distribution of the complexity scores across all chromosomes for each sample, with means indicated by the green diamond. The recursive partition line graph (C) shows the t p-value for the Mann Whitney U test for complexity score at and below each position > above for the sum (genome blue) and individual chromosome (orange). Data displayed is a random subsample (n=50) of the GEL breast cancer cohort.



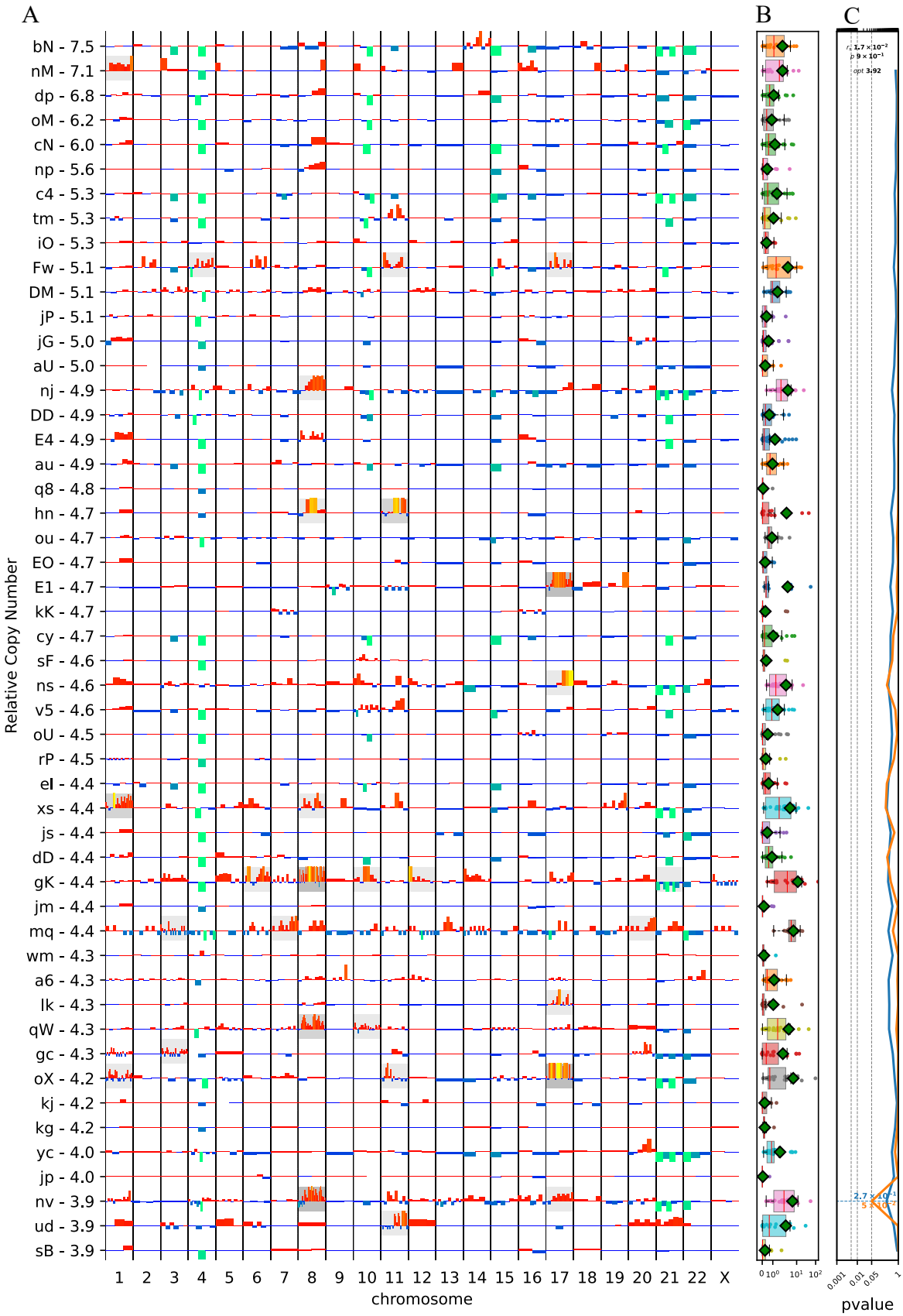
Appendix 37 Heatmap showing relative copy number segments from pcf analysis (input of 10kb coverage windows) clipped between -1.5 (blue loss) and 1.5 (red gain) with the Y-axis displaying ordered from longest telomeres (top) to shortest (bottom). Data displayed is a random subsample (n=50) of the GEL breast cancer cohort.



Appendix 38 The coverage graph (A) shows the relative copy number (RCN) segments scaled so each chromosome is uniformly sized, with the length of each segment on the x-axis being logged. The complexity score boxplots (B) show the distribution of the complexity scores across all chromosomes for each sample, with means indicated by the green diamond. The recursive partition line graph (C) shows the t p-value for the Mann Whitney U test for complexity score at and below each position > above for the sum (genome blue) and individual chromosome (orange). Data displayed is a random subsample (n=50) of the GEL breast cancer cohort.



Appendix 39 Heatmap showing relative copy number segments from pcf analysis (input of 10kb coverage windows) clipped between -1.5 (blue loss) and 1.5 (red gain) with the Y-axis displaying ordered from longest telomeres (top) to shortest (bottom). Data displayed is a random subsample (n=50) of the GEL breast cancer cohort.



Appendix 40 The coverage graph (A) shows the relative copy number (RCN) segments scaled so each chromosome is uniformly sized, with the length of each segment on the x-axis being logged. The complexity score boxplots (B) show the distribution of the complexity scores across all chromosomes for each sample, with means indicated by the green diamond. The recursive partition line graph (C) shows the t p-value for the Mann Whitney U test for complexity score at and below each position > above for the sum (genome blue) and individual chromosome (orange). Data displayed is a random subsample (n=50) of the GEL breast cancer cohort.

References

Abruzzo, L.V. et al. 2018. Trisomy 12 chronic lymphocytic leukemia expresses a unique set of activated and targetable pathways. *Haematologica* 103(12), pp. 2069–2078. doi: 10.3324/haematol.2018.190132.

Abyzov, A., Urban, A.E., Snyder, M. and Gerstein, M. 2011. CNVnator: an approach to discover, genotype, and characterize typical and atypical CNVs from family and population genome sequencing. *Genome Research* 21(6), pp. 974–984. doi: 10.1101/gr.114876.110.

Akar, M. and Oktay, K. 2005. Restoration of ovarian endocrine function by ovarian transplantation. *Trends in Endocrinology & Metabolism* 16(8), pp. 374–380. doi: 10.1016/j.tem.2005.06.011.

Aksenova, A.Y. and Mirkin, S.M. 2019. At the Beginning of the End and in the Middle of the Beginning: Structure and Maintenance of Telomeric DNA Repeats and Interstitial Telomeric Sequences. *Genes* 10(2). Available at: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6410037/ [Accessed: 9 September 2024].

Alberts, B., Johnson, A., Lewis, J., Raff, M., Roberts, K. and Walter, P. 2002. General Recombination. In: *Molecular Biology of the Cell. 4th edition*. Garland Science. Available at: https://www.ncbi.nlm.nih.gov/books/NBK26898/ [Accessed: 12 August 2024].

Alexandrov, L.B. et al. 2013. Signatures of mutational processes in human cancer. *Nature* 500(7463), pp. 415–421. doi: 10.1038/nature12477.

Anand, R., Lovett, S. and Haber, J. 2013. Break-induced DNA replication. *Cold Spring Harbor perspectives in biology* 5(12). Available at: https://pubmed.ncbi.nlm.nih.gov/23881940/ [Accessed: 9 September 2024].

Arnoult, N., Van Beneden, A. and Decottignies, A. 2012. Telomere length regulates TERRA levels through increased trimethylation of telomeric H3K9 and HP1α. *Nature Structural & Molecular Biology* 19(9), pp. 948–956. doi: 10.1038/nsmb.2364.

Ashby, C. et al. 2022. Structural variants shape the genomic landscape and clinical outcome of multiple myeloma. *Blood Cancer Journal* 12(5), p. 85. doi: 10.1038/s41408-022-00673-x.

Baca, S.C. et al. 2013. Punctuated Evolution of Prostate Cancer Genomes. *Cell* 153(3), pp. 666–677. doi: 10.1016/j.cell.2013.03.021.

Bailey, S.M. and Murnane, J.P. 2006. Telomeres, chromosome instability and cancer. *Nucleic Acids Research* 34(8), p. 2408. doi: 10.1093/nar/gkl303.

Baird, D.M., Rowson, J., Wynford-Thomas, D. and Kipling, D. 2003. Extensive allelic variation and ultrashort telomeres in senescent human cells. *Nature Genetics* 33(2), pp. 203–207. doi: 10.1038/ng1084.

Berger, M.F. et al. 2011. The genomic complexity of primary human prostate cancer. *Nature* 470(7333), pp. 214–220. doi: 10.1038/nature09744.

Beroukhim, R. et al. 2010. The landscape of somatic copy-number alteration across human cancers. *Nature* 463(7283), pp. 899–905. doi: 10.1038/nature08822.

Bhargava, R., Onyango, D.O. and Stark, J.M. 2016. Regulation of Single Strand Annealing and its role in genome maintenance. *Trends in genetics : TIG* 32(9), pp. 566–575. doi: 10.1016/j.tig.2016.06.007.

Birney, E. et al. 2004. An Overview of Ensembl. *Genome Research* 14(5), pp. 925–928. doi: 10.1101/gr.1860604.

Bizard, A.H. and Hickson, I.D. 2014. The Dissolution of Double Holliday Junctions. *Cold Spring Harbor Perspectives in Biology* 6(7), p. a016477. doi: 10.1101/cshperspect.a016477.

Blakely, G.W. 2015. Chapter 15 - Mechanisms of Horizontal Gene Transfer and DNA Recombination. In: Tang, Y.-W., Sussman, M., Liu, D., Poxton, I., and Schwartzman, J. eds. *Molecular Medical Microbiology (Second Edition)*. Boston: Academic Press, pp. 291–302. Available at: https://www.sciencedirect.com/science/article/pii/B9780123971692000159 [Accessed: 12 August 2024].

Bloom, B.H. 1970. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM* 13(7), pp. 422–426. doi: 10.1145/362686.362692.

Brambati, A., Sacco, O., Porcella, S., Heyza, J., Kareh, M., Schmidt, J.C. and Sfeir, A. 2023. RHINO directs MMEJ to repair DNA breaks in mitosis. *Science (New York, N.Y.)* 381(6658), pp. 653–660. doi: 10.1126/science.adh3694.

Branzei, D. and Foiani, M. 2005. The DNA damage response during DNA replication. *Current Opinion in Cell Biology* 17(6), pp. 568–575. doi: 10.1016/j.ceb.2005.09.003.

Britt-Compton, B. et al. 2012. Extreme telomere erosion in ATM-mutated and 11q-deleted CLL patients is independent of disease stage. *Leukemia* 26(4), pp. 826–830. doi: 10.1038/leu.2011.281.

Brown, D.W., Lin, S.-H., Loh, P.-R., Chanock, S.J., Savage, S.A. and Machiela, M.J. 2020. Genetically predicted telomere length is associated with clonal somatic copy number alterations in peripheral leukocytes. *PLOS Genetics* 16(10), p. e1009078. doi: 10.1371/journal.pgen.1009078.

Buitinck, L. et al. 2013. API design for machine learning software: experiences from the scikit-learn project. In: *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*. pp. 108–122.

Cai, H., Kumar, N. and Baudis, M. 2012. arrayMap: A Reference Resource for Genomic Copy Number Imbalances in Human Malignancies. *PLoS ONE* 7(5). Available at: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3356349/ [Accessed: 27 September 2024].

Cai, Y., Kandula, V., Kosuru, R., Ye, X., Irwin, M.G. and Xia, Z. 2017. Decoding telomere protein Rap1: Its telomeric and nontelomeric functions and potential implications in diabetic cardiomyopathy. *Cell Cycle* 16(19), pp. 1765–1773. doi: 10.1080/15384101.2017.1371886.

Cannan, W.J. and Pederson, D.S. 2016. Mechanisms and Consequences of Double-strand DNA Break Formation in Chromatin. *Journal of cellular physiology* 231(1), pp. 3–14. doi: 10.1002/jcp.25048.

Capper, R. et al. 2007. The nature of telomere fusion and a definition of the critical telomere length in human cells. *Genes & Development* 21(19), pp. 2495–2508. doi: 10.1101/gad.439107.

Cesare, A.J. and Karlseder, J. 2012. A three-state model of telomere control over human proliferative boundaries. *Current opinion in cell biology* 24(6), p. 731. doi: 10.1016/j.ceb.2012.08.007.

Chan, Y.W. and West, S.C. 2018. A new class of ultrafine anaphase bridges generated by homologous recombination. *Cell Cycle* 17(17), pp. 2101–2109. doi: 10.1080/15384101.2018.1515555.

Chang, H.H.Y., Pannunzio, N.R., Adachi, N. and Lieber, M.R. 2017. Non-homologous DNA end joining and alternative pathways to double-strand break repair. *Nature Reviews Molecular Cell Biology* 18(8), pp. 495–506. doi: 10.1038/nrm.2017.48.

Chang, S., Khoo, C.M., Naylor, M.L., Maser, R.S. and DePinho, R.A. 2003. Telomere-based crisis: functional differences between telomerase activation and ALT in tumor progression. *Genes & Development* 17(1), pp. 88–100. doi: 10.1101/gad.1029903.

Chen, J.J.-L. and Podlevsky, J.D. 2016. Telomeres and Telomerase. In: Bradshaw, R. A. and Stahl, P. D. eds. *Encyclopedia of Cell Biology*. Waltham: Academic Press, pp. 418–425. Available at: https://www.sciencedirect.com/science/article/pii/B9780123944474100422 [Accessed: 9 September 2024].

Chen, J.-L., Blasco, M.A. and Greider, C.W. 2000. Secondary Structure of Vertebrate Telomerase RNA. *Cell* 100(5), pp. 503–514. doi: 10.1016/S0092-8674(00)80687-X.

Chen, X. et al. 2016. Manta: rapid detection of structural variants and indels for germline and cancer sequencing applications. *Bioinformatics (Oxford, England)* 32(8), pp. 1220–1222. doi: 10.1093/bioinformatics/btv710.

Cherry, A.L., Nott, T.J., Kelly, G., Rulten, S.L., Caldecott, K.W. and Smerdon, S.J. 2015. Versatility in phospho-dependent molecular recognition of the XRCC1 and XRCC4 DNA-damage scaffolds by aprataxin-family FHA domains. *DNA Repair* 35, pp. 116–125. doi: 10.1016/j.dnarep.2015.10.002.

Chib, S., Byrd, A.K. and Raney, K.D. 2016. Yeast Helicase Pif1 Unwinds RNA:DNA Hybrids with Higher Processivity than DNA:DNA Duplexes. *The Journal of Biological Chemistry* 291(11), pp. 5889–5901. doi: 10.1074/jbc.M115.688648.

Chiba, K., Johnson, J.Z., Vogan, J.M., Wagner, T., Boyle, J.M. and Hockemeyer, D. 2015. Cancer-associated TERT promoter mutations abrogate telomerase silencing. *eLife* 4, p. e07918. doi: 10.7554/eLife.07918.

Choschzick, M. et al. 2010. Amplification of 8q21 in breast cancer is independent of MYC and associated with poor patient outcome. *Modern Pathology* 23(4), pp. 603–610. doi: 10.1038/modpathol.2010.5.

Chuang, T.C.Y. et al. 2004. The three-dimensional organization of telomeres in the nucleus of mammalian cells. *BMC Biology* 2, p. 12. doi: 10.1186/1741-7007-2-12.

Chun, K. et al. 2018. Assessing copy number aberrations and copy-neutral loss-of-heterozygosity across the genome as best practice: An evidence-based review from the Cancer Genomics Consortium (CGC) working group for chronic lymphocytic leukemia. *Cancer Genetics* 228–229, pp. 236–250. doi: 10.1016/j.cancergen.2018.07.004.

Clare O'Connor. 2008a. Fluorescence In Situ Hybridization (FISH). *Nature Education* 1(1), p. 171.

Clare O'Connor. 2008b. Karyotyping for Chromosomal Abnormalities. *Nature Education* 1(1), p. 27.

Clark, D.P., Pazdernik, N.J. and McGehee, M.R. 2019. *Molecular biology*. Third edition. London: Academic Press is an imprint of Elsevier. Available at: https://www.sciencedirect.com/science/book/9780128132883 [Accessed: 5 August 2024].

Cleal, K. and Baird, D.M. 2020. Catastrophic Endgames: Emerging Mechanisms of Telomere-Driven Genomic Instability. *Trends in Genetics* 36(5), pp. 347–359. doi: 10.1016/j.tig.2020.02.001.

Cleal, K. and Baird, D.M. 2022. Dysgu: efficient structural variant calling using short or long reads. *Nucleic Acids Research* 50(9), p. e53. doi: 10.1093/nar/gkac039.

Cleal, K., Jones, R.E., Grimstead, J.W., Hendrickson, E.A. and Baird, D.M. 2019. Chromothripsis during telomere crisis is independent of NHEJ, and consistent with a replicative origin. *Genome Research* 29(5), pp. 737–749. doi: 10.1101/gr.240705.118.

Cleal, K., Kearsey, A. and Baird, D.M. 2024. GW: ultra-fast chromosome-scale visualisation of genomics data. p. 2024.07.26.605272. Available at: https://www.biorxiv.org/content/10.1101/2024.07.26.605272v5 [Accessed: 11 September 2024].

Cleal, K., Norris, K. and Baird, D. 2018. Telomere Length Dynamics and the Evolution of Cancer Genome Architecture. *International Journal of Molecular Sciences* 19(2), p. 482. doi: 10.3390/ijms19020482.

Collins, K. 2011. Single-stranded DNA repeat synthesis by telomerase. *Current opinion in chemical biology* 15(5), pp. 643–648. doi: 10.1016/j.cbpa.2011.07.011.

Cong, Y.-S., Wright, W.E. and Shay, J.W. 2002. Human Telomerase and Its Regulation. *Microbiology and Molecular Biology Reviews* 66(3), p. 407. doi: 10.1128/MMBR.66.3.407-425.2002.

Cortés-Ciriano, I. et al. 2020. Comprehensive analysis of chromothripsis in 2,658 human cancers using whole-genome sequencing. *Nature Genetics* 52(3), pp. 331–341. doi: 10.1038/s41588-019-0576-7.

Costantino, L. and Koshland, D. 2015. The Yin and Yang of R-loop biology. *Current Opinion in Cell Biology* 34, pp. 39–45. doi: 10.1016/j.ceb.2015.04.008.

Counter, C.M., Avilion, A.A., LeFeuvre, C.E., Stewart, N.G., Greider, C.W., Harley, C.B. and Bacchetti, S. 1992. Telomere shortening associated with chromosome instability is arrested in immortal cells which express telomerase activity. *The EMBO Journal* 11(5), pp. 1921–1929.

Courjal, F. and Theillet, C. 1997. Comparative genomic hybridization analysis of breast tumors with predetermined profiles of DNA amplification. *Cancer Research* 57(19), pp. 4368–4377.

Crasta, K. et al. 2012. DNA breaks and chromosome pulverization from errors in mitosis. *Nature* 482(7383), pp. 53–58. doi: 10.1038/nature10802.

Crusoe, M.R. et al. 2015. The khmer software package: enabling efficient nucleotide sequence analysis. Available at: http://dx.doi.org/10.12688/f1000research.6924.1.

Curtis, C. et al. 2012. The genomic and transcriptomic architecture of 2,000 breast tumours reveals novel subgroups. *Nature* 486(7403), pp. 346–352. doi: 10.1038/nature10983.

Danecek, P. et al. 2011. The variant call format and VCFtools. *Bioinformatics* 27(15), pp. 2156–2158. doi: 10.1093/bioinformatics/btr330.

Danecek, P. et al. 2021. Twelve years of SAMtools and BCFtools. *GigaScience* 10(2), p. giab008. doi: 10.1093/gigascience/giab008.

De, S. and Michor, F. 2011. DNA replication timing and long-range DNA interactions predict mutational landscapes of cancer genomes. *Nature biotechnology* 29(12), pp. 1103–1108. doi: 10.1038/nbt.2030.

Dewhurst, S.M. 2020. Chromothripsis and telomere crisis: engines of genome instability. *Current opinion in genetics & development* 60, pp. 41–47. doi: 10.1016/j.gde.2020.02.009.

Ding, Z., Mangino, M., Aviv, A., Consortium, U., Spector, T. and Durbin, R. 2014. Estimating telomere length from whole genome sequence data. *Nucleic Acids Research* 42(9), p. e75. doi: 10.1093/nar/gku181.

Diotti, R. and Loayza, D. 2011. Shelterin complex and associated factors at human telomeres. *Nucleus* 2(2), pp. 119–135. doi: 10.4161/nucl.2.2.15135.

Dohm, J.C., Lottaz, C., Borodina, T. and Himmelbauer, H. 2008. Substantial biases in ultra-short read data sets from high-throughput DNA sequencing. *Nucleic Acids Research* 36(16), p. e105. doi: 10.1093/nar/gkn425.

Döhner, H. et al. 1997. 11q deletions identify a new subset of B-cell chronic lymphocytic leukemia characterized by extensive nodal involvement and inferior prognosis. *Blood* 89(7), pp. 2516–2522.

Döhner, H. et al. 2000. Genomic Aberrations and Survival in Chronic Lymphocytic Leukemia. *New England Journal of Medicine* 343(26), pp. 1910–1916. doi: 10.1056/NEJM200012283432602.

Dos Santos, P., Panero, J., Palau Nagore, V., Stanganelli, C., Bezares, R.F. and Slavutsky, I. 2015. Telomere shortening associated with increased genomic complexity in chronic

lymphocytic leukemia. *Tumor Biology* 36(11), pp. 8317–8324. doi: 10.1007/s13277-015-3556-2.

Drmanac, R. et al. 2010. Human Genome Sequencing Using Unchained Base Reads on Self-Assembling DNA Nanoarrays. *Science* 327(5961), pp. 78–81. doi: 10.1126/science.1181498.

Engel, J.L. et al. 2024. The Fanconi anemia pathway induces chromothripsis and ecDNA-driven cancer drug resistance. *Cell*. Available at: https://www.sciencedirect.com/science/article/pii/S0092867424008924 [Accessed: 27 September 2024].

Ewing, B., Hillier, L., Wendl, M.C. and Green, P. 1998. Base-Calling of Automated Sequencer Traces UsingPhred. I. Accuracy Assessment. *Genome Research* 8(3), pp. 175–185. doi: 10.1101/gr.8.3.175.

Farmery, J.H.R., Smith, M.L. and Lynch, A.G. 2018. Telomerecat: A ploidy-agnostic method for estimating telomere length from whole genome sequencing data. *Scientific Reports* 8(1), p. 1300. doi: 10.1038/s41598-017-14403-y.

Fasshauer, G.E. 2007. *Meshfree Approximation Methods with MATLAB*. World Scientific. Available at: https://books.google.co.uk/books?id=gtqBdMEqryEC.

Feuerbach, L. et al. 2019. TelomereHunter – in silico estimation of telomere content and composition from cancer genomes. *BMC Bioinformatics* 20(1), p. 272. doi: 10.1186/s12859-019-2851-0.

Flynn, R.L. et al. 2015. Alternative lengthening of telomeres renders cancer cells hypersensitive to ATR inhibitors. *Science* 347(6219), pp. 273–277. doi: 10.1126/science.1257216.

Forment, J.V., Kaidi, A. and Jackson, S.P. 2012. Chromothripsis and cancer: causes and consequences of chromosome shattering. *Nature Reviews. Cancer* 12(10), pp. 663–670. doi: 10.1038/nrc3352.

Frith, M.C. 2011. A new repeat-masking method enables specific detection of homologous sequences. *Nucleic Acids Research* 39(4), p. e23. doi: 10.1093/nar/gkq1212.

Fugger, K. and West, S.C. 2016. Keeping homologous recombination in check. *Cell Research* 26(4), pp. 397–398. doi: 10.1038/cr.2016.25.

Gabay, M., Li, Y. and Felsher, D.W. 2014. MYC Activation Is a Hallmark of Cancer Initiation and Maintenance. *Cold Spring Harbor Perspectives in Medicine* 4(6), p. a014241. doi: 10.1101/cshperspect.a014241.

Gay-Bellile, M. et al. 2017. TERT promoter status and gene copy number gains: effect on TERT expression and association with prognosis in breast cancer. *Oncotarget* 8(44), pp. 77540–77551. doi: 10.18632/oncotarget.20560.

Giraud-Panis, M.-J., Teixeira, M.T., Géli, V. and Gilson, E. 2010. CST Meets Shelterin to Keep Telomeres in Check. *Molecular Cell* 39(5), pp. 665–676. doi: 10.1016/j.molcel.2010.08.024.

Gisselsson, D. et al. 2000. Chromosomal breakage-fusion-bridge events cause genetic intratumor heterogeneity. *Proceedings of the National Academy of Sciences* 97(10), pp. 5357–5362. doi: 10.1073/pnas.090013497.

Goergen, E. and Al-Sawaf, O. 2024. The prognostic significance of genomic complexity in patients with CLL. *Leukemia & Lymphoma*. Available at: https://www.tandfonline.com/doi/full/10.1080/10428194.2024.2333448 [Accessed: 6 September 2024].

Goodman, S.R. ed. 2021. Cell Cycle and Cancer. In: *Goodman's Medical Cell Biology (Fourth Edition)*. Academic Press, pp. 295–313. Available at: https://www.sciencedirect.com/science/article/pii/B9780128179277000107 [Accessed: 5 August 2024].

Goodwin, S., McPherson, J.D. and McCombie, W.R. 2016. Coming of age: ten years of next-generation sequencing technologies. *Nature Reviews Genetics* 17(6), pp. 333–351. doi: 10.1038/nrg.2016.49.

Grawunder, U., Wilm, M., Wu, X., Kulesza, P., Wilson, T.E., Mann, M. and Lieber, M.R. 1997. Activity of DNA ligase IV stimulated by complex formation with XRCC4 protein in mammalian cells. *Nature* 388(6641), pp. 492–495. doi: 10.1038/41358.

Greenberg, R.A. 2005. Telomeres, crisis and cancer. *Current Molecular Medicine* 5(2), pp. 213–218. doi: 10.2174/1566524053586590.

Greider, C.W. 1999. Telomeres Do D-Loop–T-Loop. *Cell* 97(4), pp. 419–422. doi: 10.1016/S0092-8674(00)80750-3.

Guérin, T.M. et al. 2019. Condensin-Mediated Chromosome Folding and Internal Telomeres Drive Dicentric Severing by Cytokinesis. *Molecular Cell* 75(1), pp. 131-144.e3. doi: 10.1016/j.molcel.2019.05.021.

Guo, L. et al. 2023. Breast cancer heterogeneity and its implication in personalized precision therapy. *Experimental Hematology & Oncology* 12(1), p. 3. doi: 10.1186/s40164-022-00363-1.

Guyon, I. and Elisseeff, A. 2003. An introduction to variable and feature selection. Kaelbling, L. P. ed. *Journal of Machine Learning Research* 3(7–8), pp. 1157–1182. doi: 10.1162/153244303322753616.

Haar, A. 1910. Zur Theorie der orthogonalen Funktionensysteme. *Mathematische Annalen* 69(3), pp. 331–371. doi: 10.1007/BF01456326.

Hadi, K. et al. 2020. Distinct Classes of Complex Structural Variation Uncovered across Thousands of Cancer Genome Graphs. *Cell* 183(1), pp. 197-210.e32. doi: 10.1016/j.cell.2020.08.006.

Hagberg, A.A., Schult, D.A. and Swart, P.J. 2008. Exploring Network Structure, Dynamics, and Function using NetworkX. In: Varoquaux, G., Vaught, T., and Millman, J. eds. *Proceedings of the 7th Python in Science Conference*. Pasadena, CA USA, pp. 11–15.

Hallek, M. et al. 2018. iwCLL guidelines for diagnosis, indications for treatment, response assessment, and supportive management of CLL. *Blood* 131(25), pp. 2745–2760. doi: 10.1182/blood-2017-09-806398.

Harbeck, N. et al. 2019. Breast cancer. *Nature Reviews Disease Primers* 5(1), pp. 1–31. doi: 10.1038/s41572-019-0111-2.

Harris, C.R. et al. 2020. Array programming with NumPy. *Nature* 585(7825), pp. 357–362. doi: 10.1038/s41586-020-2649-2.

Hastings, P.J., Ira, G. and Lupski, J.R. 2009. A Microhomology-Mediated Break-Induced Replication Model for the Origin of Human Copy Number Variation. *PLoS Genetics* 5(1), p. e1000327. doi: 10.1371/journal.pgen.1000327.

Hatch, E.M. 2018. Nuclear envelope rupture: little holes, big openings. *Current opinion in cell biology* 52, pp. 66–72. doi: 10.1016/j.ceb.2018.02.001.

Hatch, E.M. and Hetzer, M.W. 2015. Linking Micronuclei to Chromosome Fragmentation. *Cell* 161(7), pp. 1502–1504. doi: 10.1016/j.cell.2015.06.005.

Haussmann, M.F. and Mauck, R.A. 2008. Telomeres and Longevity: Testing an Evolutionary Hypothesis. *Molecular Biology and Evolution* 25(1), pp. 220–228. doi: 10.1093/molbev/msm244.

Haussmann, M.F., Winkler, D.W., O'Reilly, K.M., Huntington, C.E., Nisbet, I.C.T. and Vleck, C.M. 2003. Telomeres shorten more slowly in long-lived birds and mammals than in short-lived ones. *Proceedings of the Royal Society B: Biological Sciences* 270(1522), pp. 1387–1392. doi: 10.1098/rspb.2003.2385.

Hayashi, M.T., Cesare, A.J., Rivera, T. and Karlseder, J. 2015. Cell death during crisis is mediated by mitotic telomere deprotection. *Nature* 522(7557), pp. 492–496. doi: 10.1038/nature14513.

Hayflick, L. 1965. The limited in vitro lifetime of human diploid cell strains. *Experimental Cell Research* 37(3), pp. 614–636. doi: 10.1016/0014-4827(65)90211-9.

Henson, J.D., Cao, Y., Huschtscha, L.I., Chang, A.C., Au, A.Y.M., Pickett, H.A. and Reddel, R.R. 2009. DNA C-circles are specific and quantifiable markers of alternative-lengthening-of-telomeres activity. *Nature Biotechnology* 27(12), pp. 1181–1185. doi: 10.1038/nbt.1587.

Hill, R.J., Bona, N., Smink, J., Webb, H.K., Crisp, A., Garaycoechea, J.I. and Crossan, G.P. 2024. p53 regulates diverse tissue-specific outcomes to endogenous DNA damage in mice. *Nature Communications* 15(1), p. 2518. doi: 10.1038/s41467-024-46844-1.

Hoffelder, D.R., Luo, L., Burke, N.A., Watkins, S.C., Gollin, S.M. and Saunders, W.S. 2004. Resolution of anaphase bridges in cancer cells. *Chromosoma* 112(8), pp. 389–397. doi: 10.1007/s00412-004-0284-6.

Holmes, O. et al. 2022. qmotif: determination of telomere content from whole-genome sequence data. *Bioinformatics Advances* 2(1), p. vbac005. doi: 10.1093/bioadv/vbac005.

Hong, Y. et al. 2018. LEM-3 is a midbody-tethered DNA nuclease that resolves chromatin bridges during late mitosis. *Nature Communications* 9(728), pp. 1–11. doi: 10.1038/s41467-018-03135-w.

Hou, K. et al. 2022. Alternative Lengthening of Telomeres and Mediated Telomere Synthesis. *Cancers* 14(9), p. 2194. doi: 10.3390/cancers14092194.

Hourvitz, N., Awad, A. and Tzfati, Y. 2024. The many faces of the helicase RTEL1 at telomeres and beyond. *Trends in Cell Biology* 34(2), pp. 109–121. doi: 10.1016/j.tcb.2023.07.002.

Howard, D.R. et al. 2017. Results of the randomized phase IIB ARCTIC trial of low-dose rituximab in previously untreated CLL. *Leukemia* 31(11), pp. 2416–2425. doi: 10.1038/leu.2017.96.

Hu, T. et al. 2024. Comparison of the DNBSEQ platform and Illumina HiSeq 2000 for bacterial genome assembly. *Scientific Reports* 14(1), p. 1292. doi: 10.1038/s41598-024-51725-0.

Hyatt, S. et al. 2017. Telomere length is a critical determinant for survival in multiple myeloma. *British Journal of Haematology* 178(1), pp. 94–98. doi: 10.1111/bjh.14643.

Hyun, C.L., Lee, H.E., Kim, K.S., Kim, S.-W., Kim, J.H., Choe, G. and Park, S.Y. 2008. The effect of chromosome 17 polysomy on HER-2/neu status in breast cancer. *Journal of Clinical Pathology* 61(3), pp. 317–321. doi: 10.1136/jcp.2007.050336.

Ilicheva, N.V., Podgornaya, O.I. and Voronin, A.P. 2015. Chapter Three - Telomere Repeat-Binding Factor 2 Is Responsible for the Telomere Attachment to the Nuclear Membrane. In: Donev, R. ed. *Advances in Protein Chemistry and Structural Biology*. Academic Press, pp. 67–96. Available at: https://www.sciencedirect.com/science/article/pii/S1876162315000401 [Accessed: 5 August 2024].

Jacobs, J.J.L. and de Lange, T. 2004. Significant Role for p16INK4a in p53-Independent Telomere-Directed Senescence. *Current Biology* 14(24), pp. 2302–2308. doi: 10.1016/j.cub.2004.12.025.

Jarvis, M.C., Ebrahimi, D., Temiz, N.A. and Harris, R.S. 2018. Mutation Signatures Including APOBEC in Cancer Cell Lines. *JNCI cancer spectrum* 2(1), p. pky002. doi: 10.1093/jncics/pky002.

Jebaraj, B.M.C. and Stilgenbauer, S. 2021. Telomere Dysfunction in Chronic Lymphocytic Leukemia. *Frontiers in Oncology* 10, p. 612665. doi: 10.3389/fonc.2020.612665.

Jeffares, D.C. et al. 2017. Transient structural variations have strong effects on quantitative traits and reproductive isolation in fission yeast. *Nature Communications* 8, p. 14061. doi: 10.1038/ncomms14061.

Jiang, H. and Chan, Y.W. 2024. Chromatin bridges: stochastic breakage or regulated resolution? *Trends in Genetics* 40(1), pp. 69–82. doi: 10.1016/j.tig.2023.10.004.

Jones, M.J.K. and Jallepalli, P.V. 2012. Chromothripsis: chromosomes in crisis. *Developmental Cell* 23(5), pp. 908–917. doi: 10.1016/j.devcel.2012.10.010.

Jones, R.E. et al. 2014. Escape from telomere-driven crisis is DNA ligase III dependent. *Cell Reports* 8(4), pp. 1063–1076. doi: 10.1016/j.celrep.2014.07.007.

Kailashiya, C., Sharma, H.B. and Kailashiya, J. 2017. Telomerase based anticancer immunotherapy and vaccines approaches. *Vaccine* 35(43), pp. 5768–5775. doi: 10.1016/j.vaccine.2017.09.011.

Kalathiya, U., Padariya, M. and Baginski, M. 2018. The structurally similar TRFH domain of TRF1 and TRF2 dimers shows distinct behaviour towards TIN2. *Archives of Biochemistry and Biophysics* 642, pp. 52–62. doi: 10.1016/j.abb.2018.02.005.

Karimzadeh, M., Ernst, C., Kundaje, A. and Hoffman, M.M. 2018. Umap and Bismap: quantifying genome and methylome mappability. *Nucleic Acids Research* 46(20), p. e120. doi: 10.1093/nar/gky677.

Khalid, K. et al. 2021. 13q14 Deletion and Its Effect on Prognosis of Chronic Lymphocytic Leukemia. *Cureus* 13(8), p. e16839. doi: 10.7759/cureus.16839.

Kim, W. and Shay, J.W. 2018. Long-range Telomere Regulation of Gene Expression: Telomere Looping and Telomere Position Effect Over Long Distances (TPE-OLD). *Differentiation; research in biological diversity* 99, pp. 1–9. doi: 10.1016/j.diff.2017.11.005.

Kinsella, M. and Bafna, V. 2012. Combinatorics of the Breakage-Fusion-Bridge Mechanism. *Journal of Computational Biology* 19(6), pp. 662–678. doi: 10.1089/cmb.2012.0020.

Kircher, M. and Kelso, J. 2010. High-throughput DNA sequencing--concepts and limitations. *BioEssays: News and Reviews in Molecular, Cellular and Developmental Biology* 32(6), pp. 524–536. doi: 10.1002/bies.200900181.

Krejci, L., Altmannova, V., Spirek, M. and Zhao, X. 2012. Homologous recombination and its regulation. *Nucleic Acids Research* 40(13), pp. 5795–5818. doi: 10.1093/nar/gks270.

Lange, T. de. 2005. Shelterin: the protein complex that shapes and safeguards human telomeres. *Genes & Development* 19(18), pp. 2100–2110. doi: 10.1101/gad.1346005.

de Lange, T. 2009. How Telomeres Solve the End-Protection Problem. *Science (New York, N.Y.)* 326(5955), p. 948. doi: 10.1126/science.1170633.

de Lange, T. 2018. Shelterin-Mediated Telomere Protection. *Annual Review of Genetics* 52, pp. 223–247. doi: 10.1146/annurev-genet-032918-021921.

Langmead, B., Trapnell, C., Pop, M. and Salzberg, S.L. 2009. Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. *Genome Biology* 10(3), p. R25. doi: 10.1186/gb-2009-10-3-r25.

Lauter, G., Söll, I. and Hauptmann, G. 2011. Multicolor fluorescent in situ hybridization to define abutting and overlapping gene expression in the embryonic zebrafish brain. *Neural Development* 6, p. 10. doi: 10.1186/1749-8104-6-10.

Lawlor, R.T. et al. 2019. Alternative lengthening of telomeres (ALT) influences survival in soft tissue sarcomas: a systematic review with meta-analysis. *BMC Cancer* 19(1), p. 232. doi: 10.1186/s12885-019-5424-8.

Layer, R.M., Chiang, C., Quinlan, A.R. and Hall, I.M. 2014. LUMPY: a probabilistic framework for structural variant discovery. *Genome Biology* 15(6), p. R84. doi: 10.1186/gb-2014-15-6-r84.

Leão, R., Apolónio, J.D., Lee, D., Figueiredo, A., Tabori, U. and Castelo-Branco, P. 2018. Mechanisms of human telomerase reverse transcriptase (hTERT) regulation: clinical impacts in cancer. *Journal of Biomedical Science* 25, p. 22. doi: 10.1186/s12929-018-0422-8.

Lee, G.R., Gommers, R., Waselewski, F., Wohlfahrt, K. and O'Leary, A. 2019. PyWavelets: A Python package for wavelet analysis. *Journal of Open Source Software* 4(36), p. 1237. doi: 10.21105/joss.01237.

Lee, J.A., Carvalho, C.M.B. and Lupski, J.R. 2007. A DNA Replication Mechanism for Generating Nonrecurrent Rearrangements Associated with Genomic Disorders. *Cell* 131(7), pp. 1235–1247. doi: 10.1016/j.cell.2007.11.037.

Leeksma, A.C. et al. 2020. Genomic arrays identify high-risk chronic lymphocytic leukemia with genomic complexity: a multicenter study. *Haematologica* 106(1), pp. 87–97. doi: 10.3324/haematol.2019.239947.

Lejeune, J., Gautier, M. and Turpin, R. 1959. [Study of somatic chromosomes from 9 mongoloid children]. *Comptes Rendus Hebdomadaires Des Seances De l'Academie Des Sciences* 248(11), pp. 1721–1722.

Letsolo, B.T., Rowson, J. and Baird, D.M. 2010. Fusion of short telomeres in human cells is characterized by extensive deletion and microhomology, and can result in complex rearrangements. *Nucleic Acids Research* 38(6), pp. 1841–1852. doi: 10.1093/nar/gkp1183.

Li, H. et al. 2009. The Sequence Alignment/Map format and SAMtools. *Bioinformatics* 25(16), pp. 2078–2079. doi: 10.1093/bioinformatics/btp352.

Li, H. 2013. Aligning sequence reads, clone sequences and assembly contigs with BWA-MEM. Available at: http://arxiv.org/abs/1303.3997 [Accessed: 27 September 2024].

Li, H. 2018. Minimap2: pairwise alignment for nucleotide sequences. *Bioinformatics* 34(18), pp. 3094–3100. doi: 10.1093/bioinformatics/bty191.

Li, H. 2021. New strategies to improve minimap2 alignment accuracy. *Bioinformatics* 37(23), pp. 4572–4574. doi: 10.1093/bioinformatics/btab705.

Li, J., Summerlin, M., Nitiss, K.C., Nitiss, J.L. and Hanakahi, L.A. 2017. TDP1 is required for efficient non-homologous end joining in human cells. *DNA Repair* 60, pp. 40–49. doi: 10.1016/j.dnarep.2017.10.003.

Li, S. et al. 2014. Evidence That the DNA Endonuclease ARTEMIS also Has Intrinsic 5′-Exonuclease Activity. *The Journal of Biological Chemistry* 289(11), pp. 7825–7834. doi: 10.1074/jbc.M113.544874.

Li, X. and Heyer, W.-D. 2008. Homologous recombination in DNA repair and DNA damage tolerance. *Cell Research* 18(1), pp. 99–113. doi: 10.1038/cr.2008.1.

Li, Z. et al. 2007. ETV6-NTRK3 Fusion Oncogene Initiates Breast Cancer from Committed Mammary Progenitors via Activation of AP1 Complex. *Cancer Cell* 12(6), pp. 542–558. doi: 10.1016/j.ccr.2007.11.012.

Liddiard, K., Aston-Evans, A.N., Cleal, K., Hendrickson, E.A. and Baird, D.M. 2022. POLQ suppresses genome instability and alterations in DNA repeat tract lengths. *NAR Cancer* 4(3), p. zcac020. doi: 10.1093/narcan/zcac020.

Liddiard, K., Ruis, B., Takasugi, T., Harvey, A., Ashelford, K.E., Hendrickson, E.A. and Baird, D.M. 2016. Sister chromatid telomere fusions, but not NHEJ-mediated inter-chromosomal telomere fusions, occur independently of DNA ligases 3 and 4. *Genome Research* 26(5), pp. 588–600. doi: 10.1101/gr.200840.115.

Lillard-Wetherell, K. et al. 2004. Association and regulation of the BLM helicase by the telomere proteins TRF1 and TRF2. *Human Molecular Genetics* 13(17), pp. 1919–1932. doi: 10.1093/hmg/ddh193.

Lin, T.T. et al. 2010. Telomere dysfunction and fusion during the progression of chronic lymphocytic leukemia: evidence for a telomere crisis. *Blood* 116(11), pp. 1899–1907. doi: 10.1182/blood-2010-02-272104.

Lin, T.T. et al. 2014. Telomere dysfunction accurately predicts clinical outcome in chronic lymphocytic leukaemia, even in patients with early stage disease. *British Journal of Haematology* 167(2), pp. 214–223. doi: 10.1111/bjh.13023.

Lin, Y.-F. et al. 2023. Mitotic clustering of pulverized chromosomes from micronuclei. *Nature* 618(7967), pp. 1041–1048. doi: 10.1038/s41586-023-05974-0.

Lisaingo, K., Uringa, E.-J. and Lansdorp, P.M. 2014. Resolution of telomere associations by TRF1 cleavage in mouse embryonic stem cells. *Molecular Biology of the Cell* 25(13), pp. 1958–1968. doi: 10.1091/mbc.E13-10-0564.

Liu, B., He, Y., Wang, Y., Song, H., Zhou, Z.H. and Feigon, J. 2022a. Structure of active human telomerase with telomere shelterin protein TPP1. *Nature* 604(7906), pp. 578–583. doi: 10.1038/s41586-022-04582-8.

Liu, F.T., Ting, K.M. and Zhou, Z.-H. 2008. Isolation Forest. In: *2008 Eighth IEEE International Conference on Data Mining*. pp. 413–422. Available at: https://ieeexplore.ieee.org/document/4781136 [Accessed: 9 September 2024].

Liu, L. and Malkova, A. 2022. Break-induced replication: unraveling each step. *Trends in Genetics* 38(7), pp. 752–765. doi: 10.1016/j.tig.2022.03.011.

Liu, P. et al. 2011. Chromosome catastrophes involve replication mechanisms generating complex genomic rearrangements. *Cell* 146(6), pp. 889–903. doi: 10.1016/j.cell.2011.07.042.

Liu, S., Kwon, M., Mannino, M., Yang, N., Renda, F., Khodjakov, A. and Pellman, D. 2018. Nuclear envelope assembly defects link mitotic errors to chromothripsis. *Nature* 561(7724), pp. 551–555. doi: 10.1038/s41586-018-0534-z.

Liu, Y. et al. 2022b. Pan-cancer analysis on the role of PIK3R1 and PIK3R2 in human tumors. *Scientific Reports* 12(1), p. 5924. doi: 10.1038/s41598-022-09889-0.

Logsdon, G.A., Vollger, M.R. and Eichler, E.E. 2020. Long-read human genome sequencing and its applications. *Nature Reviews Genetics* 21(10), pp. 597–614. doi: 10.1038/s41576-020-0236-x.

Łukasiewicz, S., Czeczelewski, M., Forma, A., Baj, J., Sitarz, R. and Stanisławek, A. 2021. Breast Cancer—Epidemiology, Risk Factors, Classification, Prognostic Markers, and Current Treatment Strategies—An Updated Review. *Cancers* 13(17), p. 4287. doi: 10.3390/cancers13174287.

Ma, Y. et al. 2004. A Biochemically Defined System for Mammalian Nonhomologous DNA End Joining. *Molecular Cell* 16(5), pp. 701–713. doi: 10.1016/j.molcel.2004.11.017.

Machwe, A., Xiao, L. and Orren, D.K. 2004. TRF2 recruits the Werner syndrome (WRN) exonuclease for processing of telomeric DNA. *Oncogene* 23(1), pp. 149–156. doi: 10.1038/sj.onc.1206906.

Maciejowski, J. and de Lange, T. 2017. Telomeres in cancer: tumour suppression and genome instability. *Nature reviews. Molecular cell biology* 18(3), pp. 175–186. doi: 10.1038/nrm.2016.171.

Maciejowski, J., Li, Y., Bosco, N., Campbell, P.J. and de Lange, T. 2015. Chromothripsis and Kataegis Induced by Telomere Crisis. *Cell* 163(7), pp. 1641–1654. doi: 10.1016/j.cell.2015.11.054.

Macrae, C.J., McCulloch, R.D., Ylanko, J., Durocher, D. and Koch, C.A. 2008. APLF (C2orf13) facilitates nonhomologous end-joining and undergoes ATM-dependent hyperphosphorylation following ionizing radiation. *DNA Repair* 7(2), pp. 292–302. doi: 10.1016/j.dnarep.2007.10.008.

Maestroni, L., Matmati, S. and Coulon, S. 2017. Solving the Telomere Replication Problem. *Genes* 8(2), p. 55. doi: 10.3390/genes8020055.

Malkova, A. and Ira, G. 2013. Break-induced replication: functions and molecular mechanism. *Current opinion in genetics & development* 23(3), pp. 271–279. doi: 10.1016/j.gde.2013.05.007.

Marçais, G. and Kingsford, C. 2011. A fast, lock-free approach for efficient parallel counting of occurrences of k-mers. *Bioinformatics* 27(6), pp. 764–770. doi: 10.1093/bioinformatics/btr011.

Mardin, B.R. et al. 2015. A cell-based model system links chromothripsis with hyperploidy. *Molecular Systems Biology* 11(9), p. 828. doi: 10.15252/msb.20156505.

Marx, V. 2023. Method of the year: long-read sequencing. *Nature Methods* 20(1), pp. 6–11. doi: 10.1038/s41592-022-01730-w.

Maser, R.S. et al. 2007. DNA-Dependent Protein Kinase Catalytic Subunit Is Not Required for Dysfunctional Telomere Fusion and Checkpoint Response in the Telomerase-Deficient Mouse. *Molecular and Cellular Biology* 27(6), pp. 2253–2265. doi: 10.1128/MCB.01354-06.

McClintock, B. 1938. The Production of Homozygous Deficient Tissues with Mutant Characteristics by Means of the Aberrant Mitotic Behavior of Ring-Shaped Chromosomes. *Genetics* 23(4), pp. 315–376.

McClintock, B. 1939. The Behavior in Successive Nuclear Divisions of a Chromosome Broken at Meiosis. *Proceedings of the National Academy of Sciences of the United States of America* 25(8), p. 405. doi: 10.1073/pnas.25.8.405.

McClintock, B. 1941. The Stability of Broken Ends of Chromosomes in Zea Mays. *Genetics* 26(2), pp. 234–282.

McElhinny, S.A.N. et al. 2005. A Gradient of Template Dependence Defines Distinct Biological Roles for Family X Polymerases in Nonhomologous End Joining. *Molecular Cell* 19(3), pp. 357–366. doi: 10.1016/j.molcel.2005.06.012.

Middelkamp, S. et al. 2017. Molecular dissection of germline chromothripsis in a developmental context using patient-derived iPS cells. *Genome Medicine* 9(1), p. 9. doi: 10.1186/s13073-017-0399-z.

Mijit, M., Caracciolo, V., Melillo, A., Amicarelli, F. and Giordano, A. 2020. Role of p53 in the Regulation of Cellular Senescence. *Biomolecules* 10(3), p. 420. doi: 10.3390/biom10030420.

Mimitou, E.P. and Symington, L.S. 2009. DNA end resection: Many nucleases make light work. *DNA Repair* 8(9), pp. 983–995. doi: 10.1016/j.dnarep.2009.04.017.

Mir, S.M. et al. 2020. Shelterin Complex at Telomeres: Implications in Ageing. *Clinical Interventions in Aging* 15, pp. 827–839. doi: 10.2147/CIA.S256425.

Mirkin, E.V. and Mirkin, S.M. 2007. Replication Fork Stalling at Natural Impediments. *Microbiology and Molecular Biology Reviews : MMBR* 71(1), pp. 13–35. doi: 10.1128/MMBR.00030-06.

Morganella, S. et al. 2016. The topography of mutational processes in breast cancer genomes. *Nature Communications* 7(1), p. 11383. doi: 10.1038/ncomms11383.

Mukkamalla, S.K.R., Taneja, A., Malipeddi, D. and Master, S.R. 2024. Chronic Lymphocytic Leukemia. In: *StatPearls*. Treasure Island (FL): StatPearls Publishing. Available at: http://www.ncbi.nlm.nih.gov/books/NBK470433/ [Accessed: 21 August 2024].

Muller, H.J. 1938. The Remaking of Chromosomes. *Woods Hole*, pp. 181–198.

Munir, T. et al. 2017. Results of the randomized phase IIB ADMIRE trial of FCR with or without mitoxantrone in previously untreated CLL. *Leukemia* 31(10), pp. 2085–2093. doi: 10.1038/leu.2017.65.

Murakami, H. and Keeney, S. 2008. Regulating the formation of DNA double-strand breaks in meiosis. *Genes & Development* 22(3), pp. 286–292. doi: 10.1101/gad.1642308.

Murnane, J.P. 2012. Telomere dysfunction and chromosome instability. *Mutation Research/Fundamental and Molecular Mechanisms of Mutagenesis* 730(1), pp. 28–36. doi: 10.1016/j.mrfmmm.2011.04.008.

Nabetani, A. and Ishikawa, F. 2011. Alternative lengthening of telomeres pathway: Recombination-mediated telomere maintenance mechanism in human cells. *The Journal of Biochemistry* 149(1), pp. 5–14. doi: 10.1093/jb/mvq119.

Nair, L. and Gonzalez-Angulo, A.M. 2015. Chapter 13 - Personalized Therapies for Cancer Treatment. In: Singh, M. and Salnikova, M. eds. *Novel Approaches and Strategies for Biologics, Vaccines and Cancer Therapies*. San Diego: Academic Press, pp. 317–346. Available at: https://www.sciencedirect.com/science/article/pii/B9780124166035000134 [Accessed: 27 September 2024].

Nandakumar, J. and Cech, T.R. 2013. Finding the end: recruitment of telomerase to the telomere. *Nature reviews. Molecular cell biology* 14(2), pp. 69–82. doi: 10.1038/nrm3505.

Narayanan, V., Mieczkowski, P.A., Kim, H.-M., Petes, T.D. and Lobachev, K.S. 2006. The Pattern of Gene Amplification Is Determined by the Chromosomal Location of Hairpin-Capped Breaks. *Cell* 125(7), pp. 1283–1296. doi: 10.1016/j.cell.2006.04.042.

Nazaryan-Petersen, L., Bjerregaard, V.A., Nielsen, F.C., Tommerup, N. and Tümer, Z. 2020. Chromothripsis and DNA Repair Disorders. *Journal of Clinical Medicine* 9(3), p. 613. doi: 10.3390/jcm9030613.

Nersisyan, L. and Arakelyan, A. 2015. Computel: Computation of Mean Telomere Length from Whole-Genome Next-Generation Sequencing Data. *PLOS ONE* 10(4), p. e0125201. doi: 10.1371/journal.pone.0125201.

Neveling, K. et al. 2021. Next-generation cytogenetics: Comprehensive assessment of 52 hematological malignancy genomes by optical genome mapping. *American Journal of Human Genetics* 108(8), pp. 1423–1435. doi: 10.1016/j.ajhg.2021.06.001.

Neves Rebello Alves, L. et al. 2023. Biomarkers in Breast Cancer: An Old Story with a New End. *Genes* 14(7), p. 1364. doi: 10.3390/genes14071364.

Ngo, G., Hyatt, S., Grimstead, J., Jones, R., Hendrickson, E., Pepper, C. and Baird, D. 2018. PARP inhibition prevents escape from a telomere-driven crisis and inhibits cell immortalisation. *Oncotarget* 9(101), pp. 37549–37563. doi: 10.18632/oncotarget.26499.

Ngo, G.H.P., Grimstead, J.W. and Baird, D.M. 2021. UPF1 promotes the formation of R loops to stimulate DNA double-strand break repair. *Nature Communications* 12(1), p. 3849. doi: 10.1038/s41467-021-24201-w.

Nicholson, J.M. and Cimini, D. 2013. Cancer Karyotypes: Survival of the Fittest. *Frontiers in Oncology* 3, p. 148. doi: 10.3389/fonc.2013.00148.

Nik-Zainal, S. et al. 2012. Mutational processes molding the genomes of 21 breast cancers. *Cell* 149(5), pp. 979–993. doi: 10.1016/j.cell.2012.04.024.

Nik-Zainal, S. and Morganella, S. 2017. Mutational Signatures in Breast Cancer: The Problem at the DNA Level. *Clinical cancer research : an official journal of the American Association for Cancer Research* 23(11), pp. 2617–2629. doi: 10.1158/1078-0432.CCR-16-2810.

Nilsen, G. et al. 2012. Copynumber: Efficient algorithms for single- and multi-track copy number segmentation. *BMC Genomics* 13(1), p. 591. doi: 10.1186/1471-2164-13-591.

Norris, K., Hillmen, P., Rawstron, A., Hills, R., Baird, D.M., Fegan, C.D. and Pepper, C. 2019. Telomere length predicts for outcome to FCR chemotherapy in CLL. *Leukemia* 33(8), pp. 1953–1963. doi: 10.1038/s41375-019-0389-9.

Nurk, S. et al. 2022. The complete sequence of a human genome. *Science* 376(6588), pp. 44–53. doi: 10.1126/science.abj6987.

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. *Journal of Theoretical Biology* 41(1), pp. 181–190. doi: 10.1016/0022-5193(73)90198-7.

Orsetti, B. et al. 2006. Genetic profiling of chromosome 1 in breast cancer: mapping of regions of gains and losses and identification of candidate genes on 1q. *British Journal of Cancer* 95(10), pp. 1439–1447. doi: 10.1038/sj.bjc.6603433.

Ostapińska, K., Styka, B. and Lejman, M. 2022. Insight into the Molecular Basis Underlying Chromothripsis. *International Journal of Molecular Sciences* 23(6), p. 3318. doi: 10.3390/ijms23063318.

Ozkan, E. and Lacerda, M.P. 2024. Genetics, Cytogenetic Testing And Conventional Karyotype. In: *StatPearls*. Treasure Island (FL): StatPearls Publishing. Available at: http://www.ncbi.nlm.nih.gov/books/NBK563293/ [Accessed: 25 July 2024].

Pannunzio, N.R., Watanabe, G. and Lieber, M.R. 2018. Nonhomologous DNA end-joining for repair of DNA double-strand breaks. *The Journal of Biological Chemistry* 293(27), pp. 10512–10523. doi: 10.1074/jbc.TM117.000374.

Patterson-Fortin, J. and D'Andrea, A.D. 2020. Exploiting the Microhomology-Mediated End-Joining Pathway in Cancer Therapy. *Cancer research* 80(21), p. 4593. doi: 10.1158/0008-5472.CAN-20-1672.

Paweletz, N. 2001. Walther Flemming: pioneer of mitosis research. *Nature Reviews Molecular Cell Biology* 2(1), pp. 72–75. doi: 10.1038/35048077.

Pedersen, B.S., Collins, R.L., Talkowski, M.E. and Quinlan, A.R. 2017. Indexcov: fast coverage quality control for whole-genome sequencing. *GigaScience* 6(11), p. gix090. doi: 10.1093/gigascience/gix090.

Pedersen, B.S. and Quinlan, A.R. 2018. Mosdepth: quick coverage calculation for genomes and exomes. *Bioinformatics (Oxford, England)* 34(5), pp. 867–868. doi: 10.1093/bioinformatics/btx699.

Pedregosa, F. et al. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12(85), pp. 2825–2830.

Pellestor, F. 2019. Chromoanagenesis: cataclysms behind complex chromosomal rearrangements. *Molecular Cytogenetics* 12(1), p. 6. doi: 10.1186/s13039-019-0415-7.

Pellestor, F., Gaillard, J., Schneider, A., Puechberty, J. and Gatinois, V. 2022. Chromoanagenesis, the mechanisms of a genomic chaos. *Seminars in Cell & Developmental Biology* 123, pp. 90–99. doi: 10.1016/j.semcdb.2021.01.004.

Pellestor, F. and Gatinois, V. 2018. Chromoanasynthesis: another way for the formation of complex chromosomal abnormalities in human reproduction. *Human Reproduction* 33(8), pp. 1381–1387. doi: 10.1093/humrep/dey231.

Pepper, A.G.S. et al. 2022. Combined analysis of IGHV mutations, telomere length and CD49d identifies long-term progression-free survivors in TP53 wild-type CLL treated with FCR-based therapies. *Leukemia* 36(1), pp. 271–274. doi: 10.1038/s41375-021-01322-1.

Peyré, G. and Cuturi, M. 2019. Computational Optimal Transport: With Applications to Data Science. *Foundations and Trends® in Machine Learning* 11(5–6), pp. 355–607. doi: 10.1561/2200000073.

Prakash, R., Zhang, Y., Feng, W. and Jasin, M. 2015. Homologous Recombination and Human Health: The Roles of BRCA1, BRCA2, and Associated Proteins. *Cold Spring Harbor Perspectives in Biology* 7(4), p. a016600. doi: 10.1101/cshperspect.a016600.

Prjibelski, A., Antipov, D., Meleshko, D., Lapidus, A. and Korobeynikov, A. 2020. Using SPAdes De Novo Assembler. *Current Protocols in Bioinformatics* 70(1), p. e102. doi: 10.1002/cpbi.102.

Quinlan, A.R. and Hall, I.M. 2010. BEDTools: a flexible suite of utilities for comparing genomic features. *Bioinformatics* 26(6), pp. 841–842. doi: 10.1093/bioinformatics/btq033.

Rahmani, K., Thapa, R., Tsou, P., Chetty, S.C., Barnes, G., Lam, C. and Tso, C.F. 2022. Assessing the effects of data drift on the performance of machine learning models used in clinical sepsis prediction. *medRxiv*, p. 2022.06.06.22276062. doi: 10.1101/2022.06.06.22276062.

Rakha, E.A., Green, A.R., Powe, D.G., Roylance, R. and Ellis, I.O. 2006. Chromosome 16 tumor-suppressor genes in breast cancer. *Genes, Chromosomes & Cancer* 45(6), pp. 527–535. doi: 10.1002/gcc.20318.

Rampazzo, E. et al. 2010. Relationship between telomere shortening, genetic instability, and site of tumour origin in colorectal cancers. *British Journal of Cancer* 102(8), pp. 1300–1305. doi: 10.1038/sj.bjc.6605644.

Rausch, T., Zichner, T., Schlattl, A., Stütz, A.M., Benes, V. and Korbel, J.O. 2012. DELLY: structural variant discovery by integrated paired-end and split-read analysis. *Bioinformatics* 28(18), pp. i333–i339. doi: 10.1093/bioinformatics/bts378.

Replogle, J.M. et al. 2020. Aneuploidy increases resistance to chemotherapeutics by antagonizing cell division. *Proceedings of the National Academy of Sciences of the United States of America* 117(48), pp. 30566–30576. doi: 10.1073/pnas.2009506117.

Rideout, J.R. et al. 2024. scikit-bio/scikit-bio: scikit-bio 0.6.2. Available at: https://doi.org/10.5281/zenodo.12682832.

Rippe, K. and Luke, B. 2015. TERRA and the state of the telomere. *Nature Structural & Molecular Biology* 22(11), pp. 853–858. doi: 10.1038/nsmb.3078.

Rodionov, A.V. 2009. Grigorii Andreevich Levitsky (1878–1942). *Russian Journal of Genetics* 45(11), pp. 1261–1266. doi: 10.1134/S1022795409110015.

Rodriguez-Muñoz, M., Anglada, T. and Genescà, A. 2022. A matter of wrapper: Defects in the nuclear envelope of lagging and bridging chromatin threatens genome integrity. *Seminars in Cell & Developmental Biology* 123, pp. 124–130. doi: 10.1016/j.semcdb.2021.03.004.

Roger, L., Jones, R.E., Heppel, N.H., Williams, G.T., Sampson, J.R. and Baird, D.M. 2013. Extensive Telomere Erosion in the Initiation of Colorectal Adenomas and Its Association With Chromosomal Instability. *JNCI: Journal of the National Cancer Institute* 105(16), pp. 1202–1211. doi: 10.1093/jnci/djt191.

Ronaghi, M., Karamohamed, S., Pettersson, B., Uhlén, M. and Nyrén, P. 1996. Real-time DNA sequencing using detection of pyrophosphate release. *Analytical Biochemistry* 242(1), pp. 84–89. doi: 10.1006/abio.1996.0432.

Rossiello, F., Jurk, D., Passos, J.F. and d'Adda di Fagagna, F. 2022. Telomere dysfunction in ageing and age-related diseases. *Nature Cell Biology* 24(2), pp. 135–147. doi: 10.1038/s41556-022-00842-x.

Rossing, M., Sørensen, C.S., Ejlertsen, B. and Nielsen, F.C. 2019. Whole genome sequencing of breast cancer. *Apmis* 127(5), pp. 303–315. doi: 10.1111/apm.12920.

Rosso, I. et al. 2023. Alternative lengthening of telomeres (ALT) cells viability is dependent on C-rich telomeric RNAs. *Nature Communications* 14(1), p. 7086. doi: 10.1038/s41467-023-42831-0.

Rothberg, J.M. et al. 2011. An integrated semiconductor device enabling non-optical genome sequencing. *Nature* 475(7356), pp. 348–352. doi: 10.1038/nature10242.

Sarek, G. et al. 2019. CDK phosphorylation of TRF2 controls t-loop dynamics during the cell cycle. *Nature* 575(7783), pp. 523–527. doi: 10.1038/s41586-019-1744-8.

Saunders, C.T., Wong, W.S.W., Swamy, S., Becq, J., Murray, L.J. and Cheetham, R.K. 2012. Strelka: accurate somatic small-variant calling from sequenced tumor–normal sample pairs. *Bioinformatics* 28(14), pp. 1811–1817. doi: 10.1093/bioinformatics/bts271.

Schick, J., Ritchie, R.P. and Restini, C. 2021. Breast Cancer Therapeutics and Biomarkers: Past, Present, and Future Approaches. *Breast Cancer : Basic and Clinical Research* 15, p. 1178223421995854. doi: 10.1177/1178223421995854.

Schmutz, I. et al. 2020. TINF2 is a haploinsufficient tumor suppressor that limits telomere length. White, R. M., Wellinger, R. J., and Lingner, J. eds. *eLife* 9, p. e61235. doi: 10.7554/eLife.61235.

Senabouth, A. et al. 2020. Comparative performance of the BGI and Illumina sequencing technology for single-cell RNA-sequencing. *NAR Genomics and Bioinformatics* 2(2), p. lqaa034. doi: 10.1093/nargab/lqaa034.

Seplyarskiy, V.B., Soldatov, R.A., Popadin, K.Y., Antonarakis, S.E., Bazykin, G.A. and Nikolaev, S.I. 2016. APOBEC-induced mutations in human cancers are strongly enriched on the lagging DNA strand during replication. *Genome Research* 26(2), pp. 174–182. doi: 10.1101/gr.197046.115.

Sfeir, A. and Symington, L.S. 2015. Microhomology-mediated end joining: a back-up survival mechanism or dedicated pathway? *Trends in biochemical sciences* 40(11), p. 701. doi: 10.1016/j.tibs.2015.08.006.

Shaffer, L.G. et al. 2007. Microarray analysis for constitutional cytogenetic abnormalities. *Genetics in Medicine* 9(9), pp. 654–662. doi: 10.1097/GIM.0b013e31814ce3d9.

Shakoori, A.R. 2017. Fluorescence In Situ Hybridization (FISH) and Its Applications. *Chromosome Structure and Aberrations*, pp. 343–367. doi: 10.1007/978-81-322-3673-3_16.

Shao, X. et al. 2019. Copy number variation is highly correlated with differential gene expression: a pan-cancer study. *BMC medical genetics* 20(1), p. 175. doi: 10.1186/s12881-019-0909-5.

Shay, J.W. 2016. Role of Telomeres and Telomerase in Aging and Cancer. *Cancer Discovery* 6(6), pp. 584–593. doi: 10.1158/2159-8290.CD-16-0062.

Shay, J.W. and Wright, W.E. 2000. Hayflick, his limit, and cellular ageing. *Nature Reviews Molecular Cell Biology* 1(1), pp. 72–76. doi: 10.1038/35036093.

Shen, J.-C. and Loeb, L.A. 2001. Unwinding the molecular basis of the Werner syndrome. *Mechanisms of Ageing and Development* 122(9), pp. 921–944. doi: 10.1016/S0047-6374(01)00248-2.

Shen, M.M. 2013. Chromoplexy: a new category of complex rearrangements in the cancer genome. *Cancer cell* 23(5), pp. 567–569. doi: 10.1016/j.ccr.2013.04.025.

Shen, M.M. and Abate-Shen, C. 2010. Molecular genetics of prostate cancer: new prospects for old challenges. *Genes & Development* 24(18), pp. 1967–2000. doi: 10.1101/gad.1965810.

Shrivastav, M., De Haro, L.P. and Nickoloff, J.A. 2008. Regulation of DNA double-strand break repair pathway choice. *Cell Research* 18(1), pp. 134–147. doi: 10.1038/cr.2007.111.

Simpson, K., Jones, R.E., Grimstead, J.W., Hills, R., Pepper, C. and Baird, D.M. 2015. Telomere fusion threshold identifies a poor prognostic subset of breast cancer patients. *Molecular Oncology* 9(6), pp. 1186–1193. doi: 10.1016/j.molonc.2015.02.003.

Smith, C.E., Llorente, B. and Symington, L.S. 2007. Template switching during break-induced replication. *Nature* 447(7140), pp. 102–105. doi: 10.1038/nature05723.

Spinner, N.B. 2013. Chromosome Banding. In: Maloy, S. and Hughes, K. eds. *Brenner's Encyclopedia of Genetics (Second Edition)*. San Diego: Academic Press, pp. 546–548. Available at: https://www.sciencedirect.com/science/article/pii/B9780123749840002382 [Accessed: 5 August 2024].

Spinner, N.B., Ferguson-Smith, M.A. and Ledbetter, D.H. 2013. Chapter 22 - Cytogenetic Analysis. In: Rimoin, D., Pyeritz, R., and Korf, B. eds. *Emery and Rimoin's Principles and*

Practice of Medical Genetics (Sixth Edition). Oxford: Academic Press, pp. 1–18. Available at: https://www.sciencedirect.com/science/article/pii/B978012383834600029X [Accessed: 5 August 2024].

Srinivas, N., Rachakonda, S. and Kumar, R. 2020. Telomeres and Telomere Length: A General Overview. *Cancers* 12(3), p. 558. doi: 10.3390/cancers12030558.

Stankovic, T. and Skowronska, A. 2014. The role of ATM mutations and 11q deletions in disease progression in chronic lymphocytic leukemia. *Leukemia & Lymphoma* 55(6), pp. 1227–1239. doi: 10.3109/10428194.2013.829919.

Stephens, P.J. et al. 2011. Massive Genomic Rearrangement Acquired in a Single Catastrophic Event during Cancer Development. *Cell* 144(1), pp. 27–40. doi: 10.1016/j.cell.2010.11.055.

Stewart, M.D. et al. 2022. Homologous Recombination Deficiency: Concepts, Definitions, and Assays. *The Oncologist* 27(3), pp. 167–174. doi: 10.1093/oncolo/oyab053.

Stinson, B.M. and Loparo, J.J. 2021. Repair of DNA Double-Strand Breaks by the Non-homologous End Joining Pathway. *Annual review of biochemistry* 90, pp. 137–164. doi: 10.1146/annurev-biochem-080320-110356.

Storchova, R., Palek, M., Palkova, N., Veverka, P., Brom, T., Hofr, C. and Macurek, L. 2023. Phosphorylation of TRF2 promotes its interaction with TIN2 and regulates DNA damage response at telomeres. *Nucleic Acids Research* 51(3), pp. 1154–1172. doi: 10.1093/nar/gkac1269.

Stracker, T.H. and Petrini, J.H.J. 2011. The MRE11 complex: starting from the ends. *Nature Reviews Molecular Cell Biology* 12(2), pp. 90–103. doi: 10.1038/nrm3047.

Stratton, M.R., Campbell, P.J. and Futreal, P.A. 2009. The cancer genome. *Nature* 458(7239), pp. 719–724. doi: 10.1038/nature07943.

Strefford, J.C. et al. 2015. Telomere length predicts progression and overall survival in chronic lymphocytic leukemia: data from the UK LRF CLL4 trial. *Leukemia* 29(12), pp. 2411–2414. doi: 10.1038/leu.2015.217.

Stroik, S. and Hendrickson, E.A. 2020. Telomere fusions and translocations: a bridge too far? *Current Opinion in Genetics & Development* 60, pp. 85–91. doi: 10.1016/j.gde.2020.02.010.

Structural variant (SV) discovery. 2024. Available at: https://gatk.broadinstitute.org/hc/en-us/articles/9022487952155-Structural-variant-SV-discovery [Accessed: 5 August 2024].

Sui, X., KONG, N., WANG, Z. and PAN, H. 2013. Epigenetic regulation of the human telomerase reverse transciptase gene: A potential therapeutic target for the treatment of leukemia (Review). *Oncology Letters* 6(2), pp. 317–322. doi: 10.3892/ol.2013.1367.

Sumner, A.T. 2001. Chromosome Banding. In: Brenner, S. and Miller, J. H. eds. *Encyclopedia of Genetics*. New York: Academic Press, pp. 348–350. Available at: https://www.sciencedirect.com/science/article/pii/B0122270800002044 [Accessed: 5 August 2024].

Tabach, Y. et al. 2011. Amplification of the 20q Chromosomal Arm Occurs Early in Tumorigenic Transformation and May Initiate Cancer. *PLoS ONE* 6(1), p. e14632. doi: 10.1371/journal.pone.0014632.

Tanaka, H. and Yao, M.-C. 2009. Palindromic gene amplification--an evolutionarily conserved role for DNA inverted repeats in the genome. *Nature Reviews. Cancer* 9(3), pp. 216–224. doi: 10.1038/nrc2591.

Tang, J., Li, Z., Wu, Q., Irfan, M., Li, W. and Liu, X. 2022. Role of Paralogue of XRCC4 and XLF in DNA Damage Repair and Cancer Development. *Frontiers in Immunology* 13, p. 852453. doi: 10.3389/fimmu.2022.852453.

The Cancer Genome Atlas Program (TCGA) - NCI. 2022. Available at: https://www.cancer.gov/ccg/research/genome-sequencing/tcga [Accessed: 5 August 2024].

The pandas development team. 2020. pandas-dev/pandas: Pandas. Available at: https://doi.org/10.5281/zenodo.3509134.

Tjio, J.H. and Levan, A. 1956. The Chromosome Number of Man. *Hereditas* 42(1–2), pp. 1–6. doi: 10.1111/j.1601-5223.1956.tb03010.x.

Travers, K.J., Chin, C.-S., Rank, D.R., Eid, J.S. and Turner, S.W. 2010. A flexible and efficient template format for circular consensus sequencing and SNP detection. *Nucleic Acids Research* 38(15), p. e159. doi: 10.1093/nar/gkq543.

Umbreit, N.T. et al. 2020. Mechanisms generating cancer genome complexity from a single cell division error. *Science (New York, N.Y.)* 368(6488), p. eaba0712. doi: 10.1126/science.aba0712.

Uringa, E.-J. et al. 2012. RTEL1 contributes to DNA replication and repair and telomere maintenance. *Molecular Biology of the Cell* 23(14), pp. 2782–2792. doi: 10.1091/mbc.e12-03-0179.

Vanden Bempt, I. et al. 2008. Polysomy 17 in breast cancer: clinicopathologic significance and impact on HER-2 testing. *Journal of Clinical Oncology: Official Journal of the American Society of Clinical Oncology* 26(30), pp. 4869–4874. doi: 10.1200/JCO.2007.13.4296.

Vannier, J.-B., Pavicic-Kaltenbrunner, V., Petalcorin, M.I.R., Ding, H. and Boulton, S.J. 2012. RTEL1 Dismantles T Loops and Counteracts Telomeric G4-DNA to Maintain Telomere Integrity. *Cell* 149(4), pp. 795–806. doi: 10.1016/j.cell.2012.03.030.

Varela, E. and Blasco, M.A. 2010. 2009 Nobel Prize in Physiology or Medicine: telomeres and telomerase. *Oncogene* 29(11), pp. 1561–1565. doi: 10.1038/onc.2010.15.

Vasmatzis, G. et al. 2018. Chromoanasynthesis is a common mechanism that leads to ERBB2 amplifications in a cohort of early stage HER2+ breast cancer samples. *BMC Cancer* 18. Available at: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6045826/ [Accessed: 24 September 2024].

Virtanen, P. et al. 2020. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods* 17, pp. 261–272. doi: 10.1038/s41592-019-0686-2.

Vodicka, P., Andera, L., Opattova, A. and Vodickova, L. 2021. The Interactions of DNA Repair, Telomere Homeostasis, and p53 Mutational Status in Solid Cancers: Risk, Prognosis, and Prediction. *Cancers* 13(3), p. 479. doi: 10.3390/cancers13030479.

Wahba, G. 1990. *Spline Models for Observational Data*. Society for Industrial and Applied Mathematics. Available at: https://books.google.co.uk/books?id=ScRQJEETs0EC.

Walker, J.R., Corpina, R.A. and Goldberg, J. 2001. Structure of the Ku heterodimer bound to DNA and its implications for double-strand break repair. *Nature* 412(6847), pp. 607–614. doi: 10.1038/35088000.

Wang, Y., Zhao, Y., Bollas, A., Wang, Y. and Au, K.F. 2021. Nanopore sequencing technology, bioinformatics and applications. *Nature Biotechnology* 39(11), pp. 1348–1365. doi: 10.1038/s41587-021-01108-x.

Weiss, M.M., Hermsen, M.A., Meijer, G.A., van Grieken, N.C., Baak, J.P., Kuipers, E.J. and van Diest, P.J. 1999. Comparative genomic hybridisation. *Molecular Pathology* 52(5), pp. 243–251.

Williams, J. et al. 2017. Telomere length is an independent prognostic marker in MDS but not in de novo AML. *British Journal of Haematology* 178(2), pp. 240–249. doi: 10.1111/bjh.14666.

Wu, P., Takai, H. and de Lange, T. 2012. Telomeric 3′ overhangs derive from resection by Exo1 and Apollo and fill-in by POT1b-associated CST. *Cell* 150(1), pp. 39–52. doi: 10.1016/j.cell.2012.05.026.

Wu, Y., Poulos, R.C. and Reddel, R.R. 2020. Role of POT1 in Human Cancer. *Cancers* 12(10), p. 2739. doi: 10.3390/cancers12102739.

Xu, Y. et al. 2019. A new massively parallel nanoball sequencing platform for whole exome research. *BMC Bioinformatics* 20(1), p. 153. doi: 10.1186/s12859-019-2751-3.

Yaremko, M.L., Kutza, C., Lyzak, J., Mick, R., Recant, W.M. and Westbrook, C.A. 1996. Loss of heterozygosity from the short arm of chromosome 8 is associated with invasive behavior in breast cancer. *Genes, Chromosomes & Cancer* 16(3), pp. 189–195. doi: 10.1002/(SICI)1098-2264(199607)16:3<189::AID-GCC6>3.0.CO;2-V.

Yarernko, M.L., Recant, W.M. and Westbrook, C.A. 1995. Loss of heterozygosity from the short arm of chromosome 8 is an early event in breast cancers. *Genes, Chromosomes and Cancer* 13(3), pp. 186–191. doi: 10.1002/gcc.2870130308.

Yates, A.D. et al. 2020. Ensembl 2020. *Nucleic Acids Research* 48(D1), pp. D682–D688. doi: 10.1093/nar/gkz966.

Ye, C.J. et al. 2019. Micronuclei and Genome Chaos: Changing the System Inheritance. *Genes* 10(5), p. 366. doi: 10.3390/genes10050366.

Ye, J.Z.-S. et al. 2004. TIN2 Binds TRF1 and TRF2 Simultaneously and Stabilizes the TRF2 Complex on Telomeres *. *Journal of Biological Chemistry* 279(45), pp. 47264–47271. doi: 10.1074/jbc.M409047200.

Yi, X., Tesmer, V.M., Savre-Train, I., Shay, J.W. and Wright, W.E. 1999. Both Transcriptional and Posttranscriptional Mechanisms Regulate Human Telomerase Template RNA Levels. *Molecular and Cellular Biology* 19(6), pp. 3989–3997. doi: 10.1128/MCB.19.6.3989.

Yin, X., Liu, M., Tian, Y., Wang, J. and Xu, Y. 2017. Cryo-EM structure of human DNA-PK holoenzyme. *Cell Research* 27(11), pp. 1341–1350. doi: 10.1038/cr.2017.110.

Zepeda-Mendoza, C.J. and Morton, C.C. 2019. The Iceberg under Water: Unexplored Complexity of Chromoanagenesis in Congenital Disorders. *American Journal of Human Genetics* 104(4), pp. 565–577. doi: 10.1016/j.ajhg.2019.02.024.

Zhang, C.-Z. et al. 2015. Chromothripsis from DNA damage in micronuclei. *Nature* 522(7555), pp. 179–184. doi: 10.1038/nature14493.

Zhang, H., Si, S. and Hsieh, C.-J. 2017. GPU-acceleration for Large-scale Tree Boosting. Available at: http://arxiv.org/abs/1706.08359 [Accessed: 15 September 2024].

Zhu, F.-Y. et al. 2018. Comparative performance of the BGISEQ-500 and Illumina HiSeq4000 sequencing platforms for transcriptome analysis in plants. *Plant Methods* 14(1), p. 69. doi: 10.1186/s13007-018-0337-0.

Zhu, Y., Liu, X., Ding, X., Wang, F. and Geng, X. 2019. Telomere and its role in the aging pathways: telomere shortening, cell senescence and mitochondria dysfunction. *Biogerontology* 20(1), pp. 1–16. doi: 10.1007/s10522-018-9769-1.

Zinder, J.C. et al. 2022. Shelterin is a dimeric complex with extensive structural heterogeneity. *Proceedings of the National Academy of Sciences* 119(31), p. e2201662119. doi: 10.1073/pnas.2201662119.

Zinn, R.L., Pruitt, K., Eguchi, S., Baylin, S.B. and Herman, J.G. 2007. hTERT Is Expressed in Cancer Cell Lines Despite Promoter DNA Methylation by Preservation of Unmethylated DNA and Active Chromatin around the Transcription Start Site. *Cancer Research* 67(1), pp. 194–201. doi: 10.1158/0008-5472.CAN-06-3396.