The Analysis of High-Throughput Biological Datasets Utilising Distributed Computing

Submitted by

Jamie J. Alnasir

Supervisor: Hugh P. Shanahan

Advisor: Gregory Z. Gutin

for the degree of Doctor of Philosophy

of the

Royal Holloway, University of London

2017

Declaration

I, Jamie J. Alnasir, hereby declare that this thesis and the work presented in it is entirely my own. Where I have consulted the work of others, this is always clearly stated.

Signed.....(Jamie J. Alnasir)
Date:

In loving memory of my father, Al Nasir (14th April 1949 - 5th May 2015)
and for my mother Anne and girlfriend Eszter.

Abstract

This thesis explores the analysis of high-throughput biological datasets using distributed computing, particularly sequencing data produced by high-throughput technologies, which is increasing at an unprecedented scale. These large, complex data sets are routinely deposited in public archives such as the SRA (Sequence Read Archive) - as of January 2017 the SRA alone contains over a Petabyte of data.

We conduct a detailed literature review into biochemical protocol steps applied in preparing nucleic acid samples for sequencing. We describe bias that can be introduced at the molecular level of sequencing workflow steps.

Investigating sequencing metadata, we quantified the level of annotation of 29,958 experiments deposited in the SRA by searching for keywords in protocol steps. We found that 7.10%, 5.84% and 7.57% of all records (fragmentation, ligation and enrichment, respectively) had at least one keyword corresponding to one of the three protocol steps. Only 5.58% of all SRA records had annotation for all three steps.

In researching the use of Hadoop in structural Biology, we tested MapReduce for processing semi-structured data in the Protein Data Bank (PDB). Hadoop was tested for executing molecular docking and structural analysis jobs in comparison to a batch-scheduler and was shown to be competitive.

Finally, we develop an analysis system using MapReduce on Spark that quantifies sequence-specific deviations in the distribution of mapped RNA-Seq reads. We apply this to perform analyses of two organisms. First, two transcriptomes of fruit fly *D. melanogaster*, sequenced from the same lab, differing only by mutation *[gl60j]* of the eye antennal disc. Second, three samples from *H. Sapiens*, prepared in a controlled way, using *in-vitro* transcription (IVT) with different RNA-Seq preparatory protocols applied.

The wild type *D. melanogaster* data indicates a variation due to motif GC content that is more significant than that found due to exon GC content. There is a clear variation in the spread of correlations between the two data sets suggesting more variability in these data sets, than one would expect, which we show to be the result of sequencing errors. The *H. sapiens* IVT-plasmids sample, which was the control - and had no ribosomal selection applied in RNA-Seq preparation - showed the least *intra-exon* deviation in the distribution of mapped reads to exons. We demonstrate that the dependence of *intra-exon* correlations on the GC content appears to be due to mRNA selection methods - techniques that are routinely employed in RNA-Seq experiments.

Our system is extremely scalable and suitable for systematic study of large, high-throughput sequencing datasets.

Acknowledgement

I am extremely grateful to Dr. Hugh Shanahan for his supervision over the period of my PhD - he has been enthusiastic and inspiring throughout my PhD. Hugh frequently employed a Socratic style of discussion in our meetings which has helped develop my learning and ideas. I've learned much from his direction, especially with respect to publishing papers. He has also been extremely understanding and kind, especially when my father passed away in the summer of 2015.

I would also like to thank my tennis coach, Craig Thompson for being there for many tennis sessions that have kept me healthy through all the seasons, including those involving sleet and snow! His catch-phrases: "C'mon Jamie!", "Move those feet!" and "Good hustle!" have been useful motivators both on and off the court! I thank Hugh for being accommodating of these sessions.

I am indebted to Royal Holloway and the Computer Science department, not only for the financial support during my PhD, but also for providing an inspiring and positive environment that has fostered my creativity in the context of my research. I thank all the staff and students in the department. I thank Dr Zhiyuan Luo for many conversations on a wide variety of topics. I would also like to thank the CS IT support department, in particular Bob Vickers and Adrian Thomas, who have been very helpful in supporting my IT requirements for the research I've conducted. I also thank my advisor Prof. Gregory Gutin, as well as Prof. Kostas Stathis, for their enthusiasm and encouragement. I thank my lab members and fellow PhD students (too many to mention without the risk of omission) for the fond memories we've created. We have also shared inter-disciplinary knowledge as well as relaxing and interesting lunches.

I am grateful to my parents for being supportive of my niche interests as a child, their patience with my habit of dismantling household electronics goods for endless experiments, and for financing many of my earlier projects that have helped me learn.

I am thankful to Thomas and Jane Holloway for founding such a wonderful campus and institution, which has evolved wonderfully since the time of its conception. I wish to thank the British Council who, on numerous occasions, funded my travel expenses to Morocco to present at conferences. Finally I thank Saif Al-Kuwari from the ISG department (Information Security Group) for his Latex template which I have modified and expanded to typeset this thesis.

Contents

List of Figures

List of Tables

List of Abbreviations

API	Application Programming Interface
AWS	Amazon Web Services
BAM	Binary Alignment Map
BASH	Bourne Again Shell
BGI	Beijing Genomics Institute
BLAST	Basic Local Alignment Search Tool
BRCA1/2	Breast Cancer gene
cDNA	Complementary DNA
CE	Combinatorial Extension
ChIP	Chromatin Immunoprecipitation
CIGAR	Concise Idiosyncratic Gapped Alignment Report
CRLF	Carriage-Return-Line-Feed
CSE	Context Specific Error
DAG	Directed Acyclic Graph
DALI	Distant Alignment
dAMP	deoxy Adenine Mono-Phosphate
dbGaP	NIH Database of Genotypes and Phenotypes
dCMP	deoxy Cytosine Mono-Phosphate
DDBJ	DNA Data Bank of Japan
dGMP	deoxy Guanosine Mono-Phosphate
DNA	Deoxyribonucleic acid
dsDNA	Double-stranded DNA
DRMAA	Distributed Resource Management Application API
dTMP	deoxy Thymine Mono-Phosphate
ENA	European Nucleotide Archive
ENCODE	Encyclopaedia of DNA Elements
EOL	End Of Line delimiter
FCFS	First Come First Serve

FIFO	First In First Out
GATK	Genome Analysis Toolkit
GC	Guanosine and Cytosine
GEO	Gene Expression Omnibus
HiFi	Hi-Fidelity
HPC	High Performance Computing
HTS	High-Throughput Sequencing
IaaS	Infrastructure as a Service
ICGC	International Cancer Genome Consortium
ICAANN	Internet Corporation for Assigned Names and Numbers
IETF	The Internet Engineering Task Force
IHGSC	International Human Genome Sequencing Consortium
INSDC	International Nucleotide Sequence Database Collaboration
IVT	In-vitro Transcription
LIM	Load Information Manager
LSF	Load Sharing Facility
MAGE-ML	Microarray Gene Expression Markup Language
MAQC	Microarray Quality Consortium
MBD	Master Batch Daemon
MDA	Multiple-strand Displacement Amplification
MIAME	Minimum Information for a Microarray Experiment
MINSEQE	Minimum Information for a Sequencing Experiment
MPI	Message Passing Interface
mRNA	Messenger RNA
MUSTANG	Multiple Sequence Alignment Algorithm
NCBI	National Center for Biotechnology Information
ncRNA	Non-coding RNA
NFS	Network File System
NGS	Next-Generation Sequencing
NHGRI	National Human Genome Research Institute, USA
NIH	National Institute of Health, USA
NIOS	Network I/O Service
NMR	Nuclear Magnetic Resonance Spectroscopy
NOWs	Network of Workstations
OOP	Object-orientated Programming
ORF	Open Reading Frame
PaaS	Platform as a Service

PCR	Polymerase Chain Reaction
PDB	Protein Data Bank
POSIX	Portable Operating System Interface
RDD	Resilient Distributed Dataset
R&D	Research and Development
RES	Remote Execution Service
RNA	Ribonucleic Acid
SaaS	Software as a Service
SBS	Sequencing by Synthesis
SBD	Slave Batch Daemon
SEQC	Sequencing Quality Consortium
SNP	Single Nucleotide Polymorphism
SOAP	Simple Object Access Protocol
SRA	Sequence Read Archive
ssDNA	Single-stranded DNA
SSAP	Sequential Structure Alignment Program
STRUCTAL	Structural Analysis Algorithm
T4-PNK	T4-Polynucleotide Kinase
UMI	Unique Molecular Identifier/Index
VAST	Vector Alignment Search Tool
VM	Virtual Machine
WAH	Windows Azure Hypervisor
WDL	Workflow Definition Language
XML	Extensible Markup Language
YARN	Yet Another Resource Negotiator

Chapter 1

Introduction

"Hiding within those mounds of data is knowledge that could change the life of a patient, or change the world" – Atul Butte

1.1 Context of thesis

Life could not exist without nucleic acids or their protein products. Nucleic acids such as DNA and RNA are large self-replicating biomolecules known as polymers, and comprise sequences of small monomer units, specifically nucleotides. Nucleotides are designated A,T,G,C,U, which represent the chemical units Adenine, Thymine, Guanine, Cytosine and Uracil respectively, and such designation facilitates their digitisation - the process is termed sequencing. Sequences of nucleotides forming nucleic acids are essential for life (Gilbert, 1986; Pace and Marsh, 1985), forming the genome and transcriptome of living organisms and dictating their characteristics, structure and function (Stryer, 1998). They are defined as follows:

- Genome: The complete set of genetic material (DNA, or RNA in viruses) and consists of genes, exons (coding regions) and introns (non-coding regions).
- Transcriptome: The sum total of all the messenger RNA (mRNA) molecules expressed from the genes of that organism. It is highly dynamic, complex and in a constant state of flux and accommodates the cells constantly changing requirement - a result of *intra-* and *extra*-cellular stimuli as well as disease pathology.

Knowledge of the genome and transcriptome of an organism is essential for understanding the biochemical products of nucleic acids - proteins and enzymes - as well as regulatory cellular mechanisms and disease processes. As a result there have been numerous large scale projects to this end (Bernal et al., 2001; Regalado, A., 2015). A

fully quantified transcriptome involves sequencing an extremely large number of total bases. Consider *H. sapiens* the human organism, it is estimated to comprise of approximately 3.72x10^{13} cells, a genome of approximately 3.0810^{9} bases (International Human Genome Sequencing Consortium and others, 2004) with an average gene size of about 28 kilobase pairs (Venter et al., 2001) and approximately 300,000 RNA molecules per cell (Velculescu et al., 1999). Thus, a full representation of the transcriptome comprises of approximately 3.1x10^{23} (28,000 x 300,000 x 3.7x10^{13}) RNA bases. Sequencing of genomes and transcriptomes, therefore, generates large amounts of very complicated raw experimental sequencing data.

Various state of the art techniques in the fields of genomics, transcriptomics and computational biology, as well as large public repositories of experimental data, have come to the fore over the years to drive forward the study of nucleic acids and protein macromolecular structures (Heather and Chain, 2016). For example, in the recent past, the predominant technology for quantifying the transcriptome have been Microarrays, which allow researchers to interrogate and quantify specific sequences using complementary probe sequences applied to the surface of the array (by printing on Agilent arrays, or by photolithography for Affymetrix GeneChips), and have generated a significant amount of sequencing data to research (Sealfon and Chu, 2011). More recently, the introduction of massively parallel next-generation sequencing technologies such as DNA-Seq and RNA-Seq have transformed the fields of genomics and transcriptomics (Metzker, 2010; Mardis, 2011). In particular RNA-Seq (RNA Sequencing) is a next-generation, high-throughput sequencing technology that enables researchers in the biomedical and basic science fields to study various aspects of the transcriptome from alternative splicing isoforms, post-transcriptional modifications to mutations and gene expression. Research published concerning these next-generation sequencing technologies, in particular RNA-Seq, has increased on a yearly basis (Figure 1.1).

Much of the raw sequence read data from these technologies are being deposited in public repositories such as the Sequence Read Archive (SRA) (Leinonen et al., 2011), Gene Expression Omnibus (GEO) (Edgar, 2002) and ArrayExpress (Brazma, 2003). In just a year (from 2010 to July 2011) the amount of data deposited in the SRA, just one of the big sequence data repositories, increased by an order of magnitude from 10 tera bases to surpass 100 tera bases. As of January 2017 the SRA contains over 9 peta bases (9.377x10^{15}) of sequencing data, that is in excess of a terabyte (Sequence Read Archive, 2017), and ArrayExpress contains 44.5 TB of archived data (ArrayExpress, EMBL-EBI, 2017) - this without a doubt constitutes Big Data, which is characterised as data possessing large volume, velocity or variety (Laney, 2001; Borgman, 2015). This level of data production has resulted from the technological advancements in massively-

parallel high-throughput techniques; developments which have now made sequencing cheaper than storage and computation Stein (2010) - this is depicted in Figure 1.2.

This ever increasing yield of sequencing data is set to surpass other fields such as astronomy and particle physics (Stephens et al., 2015). Therefore, in order to overcome the challenges this increasing yield of data poses, and to make sense of it, collaborative approaches that make use of emerging improved computational infrastructure and new methods for interpreting the data are required (Ward et al., 2013).

In January 2008 the 1000 Genomes Project was launched with the aim of characterising the geographic and functional spectrum of human variation. It achieves this by including the genomes of participants from various populations and therefore facilitates the understanding of the genetic contribution to disease (1000 Genomes Project Consortium et al., 2012). By 2012 over one thousand genomes from anonymous participants were sequenced and the project was announced as completed in a Nature publication in 2015 (1000 Genomes Project Consortium and others, 2015) in which the project authors describe the sequencing of 1,092 individual genomes from 14 populations. Their study resulted in a haplotype map of 38 million SNPs (Single Nucleotide Polymorphisms), 1.4 million indels (insertion and deletion events) and more than 14,000 larger deletions. NHGRI (National Human Genome Research Institute) director Eric D. Green states that, "The newly published findings provide deeper insights about the presence and pattern of variants in different people's genomes, which is critical information for studying the genomic basis of human disease.". According to the National Institute of Health in the US, as of 2012, the project is the world's largest, most detailed catalogue of human genetic variation, and phase I of the project produced 180 terabytes of raw sequencing data. The resultant data set from the 1000 Genome Project, $\sim$ 7.3 TB at the time of release, has been made publicly available via the computing cloud through Amazon Web Services (AWS) (Kahn, 2011).

Moreover, in keeping with this endeavour, a number of government and research institutions worldwide have announced yet larger sequencing projects. For example, in 2014 the UK government (contracting the Illumina biotechnology company discussed shortly) announced the 100,000 genomes project (founding Genomics England for this task), which aims to study the genomes of patients with cancer or rare diseases (Siva, 2015; James Gallagher, BBC, 2014), and the BGI (Beijing Genome Institute) in China announced a 1 million genomes project (Cyranoski, 2016). In 2015 the US president Barack Obama announced the NIH (National Institute of Health) would also be launching a 1 million genomes project (Clarke, T. and Begley S., Reuters, 2015a,b), and in 2016 the pharmaceutical company AstraZeneca announced it was launching a project to sequence 2 million genomes within the coming decade with the aim of discovering

disease associated sequences and response to diseases (Ledford, 2016). These projects, as well as worldwide collaboration, such as by the International Cancer Genome Consortium (ICGC), which coordinates cancer genomics research across different nations (NIH, 2008), and the 1000 Genomes Project (1000 Genomes Project Consortium et al., 2012), will no doubt contribute extremely large amounts of sequencing data to that which already exists, a proportion of which may not yet have been fully exploited.

These developments have been made possible by a revolutionary technique, referred to as sequencing by synthesis (SBS), that has been a mainstay method in Next-generation sequencing since the late 1990s. The technology was conceived in Cambridge university by Shankar Balasubramanian and David Klenerman, who founded the sequencing company Solexa in 1998 that implemented this method. The underlying molecular technique used by Solexa was originally developed by Canard and Sarfati (1994) at the Pasteur Institute in Paris. In their research paper (Bentley et al., 2008) Balasubramanian and Klenerman explain how SBS facilitates the accurate sequence elucidation of several billions of bases of accurate nucleotide sequence per experiment, at low cost. Essentially, individual molecules of DNA are attached to a flat surface and amplified *in situ*. These anchored DNA molecules are used as templates for synthesis of complementary strands using fluorescent reversible terminator deoxyribonucleotides that fluoresce at different wavelengths for each nucleotide incorporation, thereby yielding sequence information that is captured an imaging sensor.

Over a decade, Solexa continued to develop its technologies, for instance by acquiring molecular clustering technology essential to massively parallel sequencing from Manteia - Solexa's first sequencing machine was the Genome Analyzer, launched in 2006 (Bennett, 2004). Solexa was acquired by Illumina in 2007, and through the application of this technology in its sequencing machines, Illumina is currently one of the worlds leading biotechnology companies in the field of Next-generation sequencing. The vast majority of sequencing experiments deposited in the SRA have been sequenced using equipment that Illumina have developed and manufactured - as of November 2017 a search of the SRA for sequencing experiments performed on the Illumina platform[1] reveals over 3.1 million sequencing experiment records. It is, therefore, of no surprise that after Genomics England was founded for the 100,000 genomes project, the UK government decided to secure sequencing services for the project from Illumina (Marx, 2015).

Computational Biology is also served by large archives of other types of experimental data. The Protein Data Bank (PDB), for instance, consists of models of the

[1]SRA search term used was "illumina[platform]", where [platform] refers to the search field in the database.

macromolecular structures of proteins, nucleic acids and complex assemblies derived from x-ray crystallographic, Nuclear Magnetic Resonance spectroscopy (NMR) and electron microscopy techniques (Abola et al., 1997; Berman et al., 2000). As of March 2017 there are 128,330 structures deposited there and the snapshot of the entire FTP archive is $\sim$ 757 GB[2]. High throughput analyses of these structures are a feature of Computational Biology (e.g. identifying structurally and/or functionally related proteins, binding ligands). These sources of experimental data, and their associated techniques, provide detailed insight into gene and protein function essential in various areas of biomedical research from improving crop yield, understanding genetic mechanisms of cellular pathology, drug development to pharmacogenomics and personalised medicine.

As we have seen the applications of nucleic acid sequencing are far reaching, for instance, recently NASA have decided to install a biomolecule sequencer (an Oxford Nanopore device) on the International Space Station (McIntyre et al., 2015) demonstrating the diversity of applications for sequencing technologies. According to an experiment mission statement (Smith and Burton, 2015) they propose that a molecular biology contingent on the space station allows for R&D in the following areas: Detection of nucleic acid based life, in-flight identification of microbes, testing for integration into robotics of rover vehicles, and medical intervention, all of which is necessary for travel beyond our moon (Bywaters et al., 2016; John et al., 2016). The Oxford Nanopore device, generates long, albeit noisy reads (Laver et al., 2015), at relatively low cost. As Nanopore devices require low sample amounts (Feng et al., 2015b), they may presently be employed for detection of nucleic acid sequences, as opposed to the *de novo* assembly of large genome organisms, which requires greater precision (Rhee and Burns, 2006), although the assembly of a shorter bacterial genome has been achieved (Loman et al., 2015). Nanopore devices, therefore, are another technology that is set to contribute to the further increase in sequencing data production given the low operating cost and wide variety of tasks the technology could be applied to. With this in mind, we now address the problem to be approached in this thesis.

1.2 Problem statement

The developments in technologies for interrogating, quantifying and elucidating sequence information from nucleic acids (DNA and RNA) over the past two decades, together with the falling cost of sequencing, have resulted in large amounts of biological data deposited in public repositories - the SRA (Sequence Read Archive) alone contains in excess of a Petabyte of data (as of January 2017). In particular, the

[2]PDB entries are duplicated for different file-types.

emergence of Next-generation sequencing technologies such as RNA-Seq, which employ massively parallel sequencing techniques, are high-throughput, and allow for study of various types of RNA at higher fidelity, have and will continue to generate very large amounts of complex sequencing data. This problem is currently further compounded by the initiation of multiple, extremely large sequencing projects worldwide. In the UK alone, Genomics England is currently sequencing 100,000 genomes (75,000 patients and 25,000 genomes of tumours), and as of November 2017, have sequenced 39,540 so far (Genomics England, 2014) posing significant computational challenges (Marx, 2015) in advancing Precision Medicine. Internationally, the ICGC is coordinating research with the aim of, according to their mission statement, "obtaining a comprehensive description of genomic, transcriptomic and epigenomic changes in 50 different tumour types and/or subtypes which are of clinical and societal importance across the globe.". Currently there are 90 ICGC cancer genomics projects that have been committed to worldwide (ICGC, 2017). Additionally, as discussed, internationally there are also governmental backed large sequencing endeavours, such as those aiming to sequencing the genomes of 1 and 2 million individuals. Furthermore, there are data from large-scale sequencing projects that have already been completed, such as the 1000 Genomes Project (1000 Genomes Project Consortium and others, 2015), that are available to contribute to the increasingly data-driven field of genomic medicine (Dudley et al., 2010).

Large datasets necessitate processing by means of distributed computing. Traditionally this has been achieved using in-house resources, and in the scientific research community batch-scheduled cluster computing is typically the choice of technology employed. However, such large datasets are impractical to download and process locally, and running such services was a substantial overhead for intermittent use. As a result providers of distributed cloud computing services have started to enable direct access to these data sources from their own infrastructure which can be leased. Whilst the main service providers offer Infrastructure as a Service (IaaS) which allows for the deployment of traditional batch-scheduled software for processing of large datasets on the cloud, newer distributed computing technologies, centred around the MapReduce programming paradigm, have emerged. They have been developed and are used by companies in the commercial industry who deal with extremely large data on a daily basis, such as Google and Facebook. MapReduce based distributed computing platforms such as Hadoop and Spark have been designed to be extremely scalable and fault-tolerant, and are now delivered by cloud-service providers. Although MapReduce is an extremely powerful technique for processing high-throughput data, implementing applications in MapReduce is not trivial. Often it can also be difficult and time-consuming to re-develop existing applications if their architecture is not structured so

as to lend itself to being re-implemented in MapReduce which is based on applying operations to key-value pair data structures in a distributed fashion.

Another potentially serious issue with the vast amounts of sequencing data being produced is that metadata accompanying such experiments often lack requisite information. This is especially concerning because RNA-Seq and other Next-generation sequencing techniques are prone to biases that may be introduced in a number of the steps of a typical sequencing workflow. Workflows employed for these methods involve a number of complicated steps carried out in the *wet-lab* (laboratory work that involves the direct handling and biochemical processing of the nucleic acid samples), as well as downstream computational methods *dry-lab* (computational laboratory work performed *in-silico* on the data generated). A sequencing experiment's metadata, therefore, should contain information on the protocol steps and techniques used to produce the data as this is vital to understanding the types of bias that may have been introduced in sequencing, and which often require mitigation. Crucially, many basic medical science studies and precision medicine rely on a common task in transcriptomics: expression estimation - the identification of those transcripts whose expression abundance is altered by experimental conditions that differ between sets of samples. This process typically employs complex computational methods in the quantification of expression levels for observed transcripts. It is, therefore, necessary that bias in transcriptomics data be quantifiable to allow for systematic study; such methods are particularly important, given the amounts and complexity of the data being generated by these techniques.

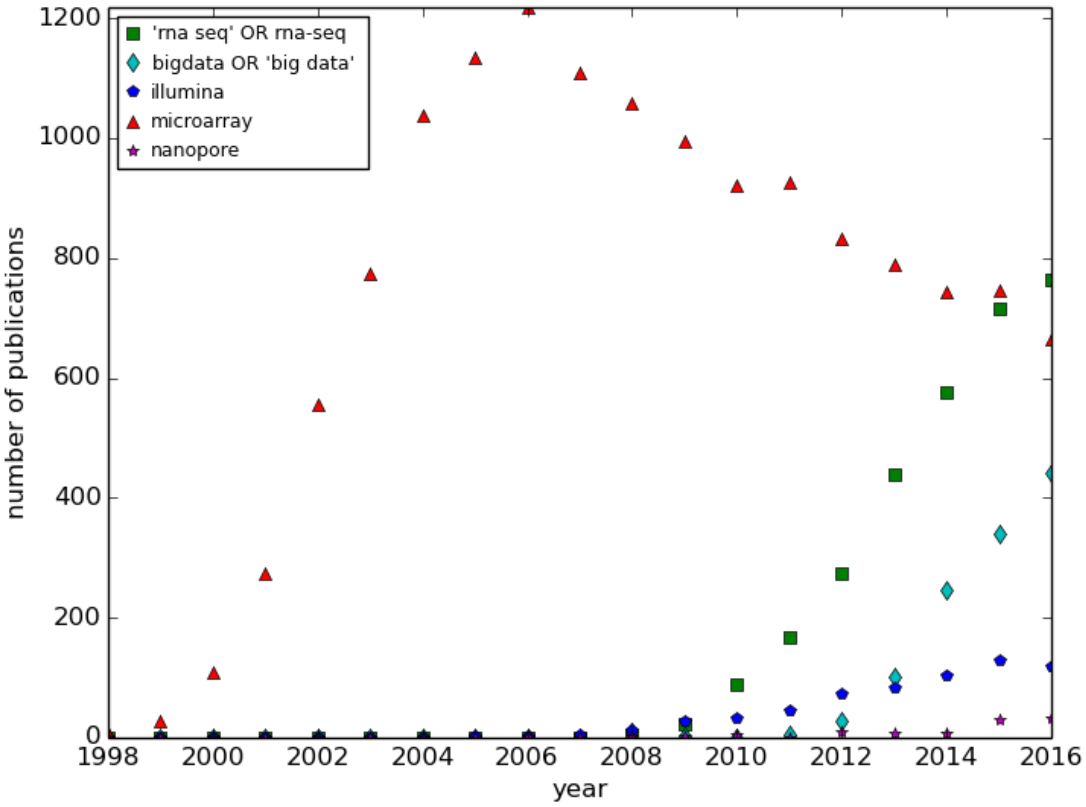


Figure 1.1: The number of publications deposited in PubMed, each year, for a selection of relevant research areas is depicted (the search terms used are shown in the legend). The number of deposited publications concerning Microarray techniques (Red) peaked at 2006 but has subsequently fallen. There is a noteworthy correlation in the trends of increasing RNA-Seq (Green) and bigdata publications (Cyan), with both research areas showing a significant increase in publications on a yearly basis from around 2011 onwards. Publication of research papers focussed on Nanopore technology (Purple) have increased slightly on yearly basis from 2014 onwards at around the same time as the Oxford Nanopore MinION device was trialled (Mikheyev and Tin, 2014).

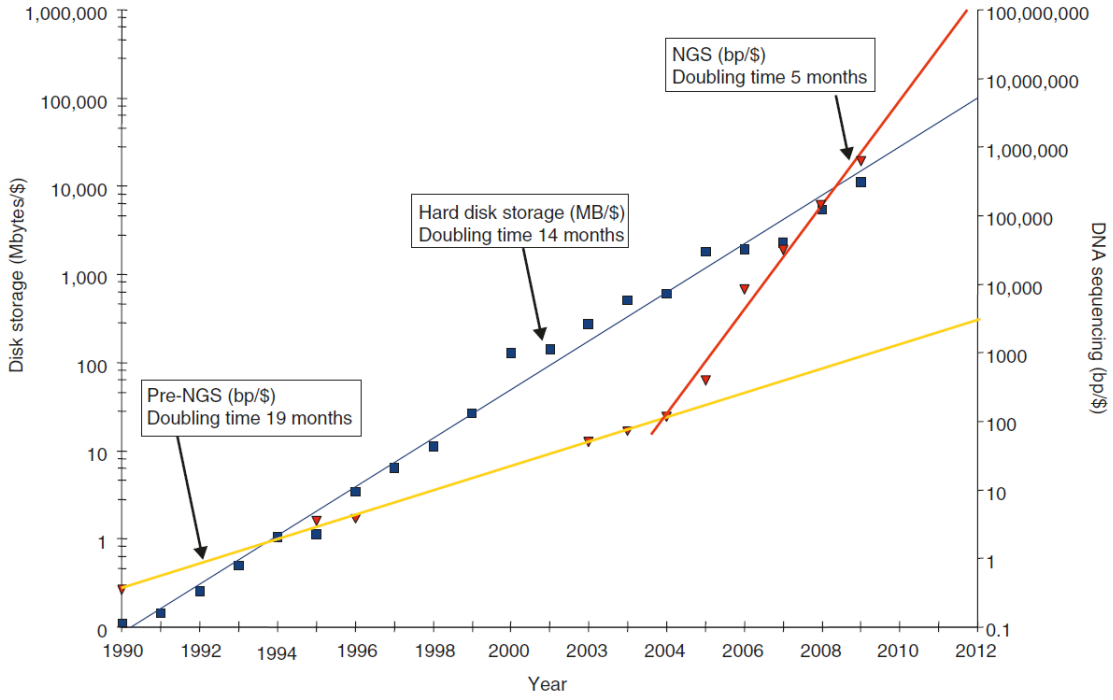


Figure 1.2: This plot is from Stein (2010) which provides the following description: "The cost of DNA sequencing, expressed in base pairs per dollar, is shown (Red triangles). The cost follows an exponential curve (Yellow line) with a doubling time slightly slower than disk storage until 2004, when next generation sequencing (NGS) causes an inflection in the curve to a doubling time of less than 6 months (Red line). Also shown is the historic cost of disk prices in megabytes per US dollar. The long-term trend (Blue line, which is a straight line here because the plot is logarithmic) shows exponential growth in storage per dollar with a doubling time of roughly 1.5 years. These curves are not corrected for inflation or for the fully loaded cost of sequencing and disk storage, which would include personnel costs, depreciation and overhead."

1.3 Thesis structure

The following is a summary of chapters that comprise this thesis and what they cover:

Chapter 2 - Review of distributed computing
In chapter 2 we review a selection of distributed computing technologies that are commonly applied to biological datasets. In particular we also discuss in detail the MapReduce programming paradigm that is employed extensively in this thesis for the analysis of biological data.

Chapter 3 - Molecular Biology of Sequencing and Bias
In chapter 3 we introduce the fundamentals of nucleic acid sequencing with respect to the biochemical, biotechnological and computational processes involved. We will review the technologies employed, and discuss in detail the types of bias that can be introduced in steps of a sequencing workflow. We conclude by discussing methods and models employed for mitigating bias in sequencing data.

Chapter 4 - Annotation of publicly deposited sequencing experiments
Chapter 4 investigates and discusses the level of annotation of 29,958 sequencing experiments deposited in a public repository, the SRA (Sequence Read Archive), which as of January 2017 contains in excess of 1 petabyte of sequencing data. We utilise data mining techniques to quantify the level of annotation in this important resource.

Chapter 5 - Testing MapReduce against the Protein Data Bank
In chapter 5 we discuss the use of MapReduce for applications in structural biology. We test MapReduce on the Hadoop platform for the processing of semi-structured molecular data deposited in the Protein Data Bank (PDB). In particular, we test and benchmark Hadoop in running external structural analysis software, and in executing molecular docking software for the docking of a small molecule against entries in the PDB.

Chapter 6 - Analysis of Transcriptomics datasets using MapReduce
In chapter 6, we make use of MapReduce on Spark to design and develop an analysis system that can quantify sequence-specific deviations in the distribution of mapped RNA sequence reads to a reference genome that result from bias inherent in typical next-generation sequencing workflows. We will specifically develop the system using MapReduce so as to be extremely scalable and suitable for systematic study of large,

high-throughput sequencing datasets. We apply the system to perform analyses of the transcriptomes of the fruit fly species *D. melanogaster* in which we compare the read distribution between *wild* and *mutant-type* Drosophila. We also perform similar analysis on RNA-Seq samples from *H. sapiens* which were produced in a controlled manner using *in-vitro* transcription (IVT).

Chapter 7 - Conclusion

Appendix

The Appendix for this thesis can be found online at the following DOI:
`https://doi.org/10.5281/zenodo.1213356`

1.4 Main contributions

The following are the main contributions of this thesis:

- An investigation that has quantified the level of annotation of key sequencing protocol steps (fragmentation, ligation and enrichment), which are documented to be potential sources of bias. This examined the metadata of sequencing experiment datasets deposited in the SRA, which is one of the largest public repositories for high-throughput sequencing data. The finding of this research was that overall there is poor annotation of the metadata for sequencing experiments; only 5.58% of all SRA records had annotation for all three steps and this inhibits systematic study of the underlying sequencing data.
- A scalable framework that uses MapReduce on Hadoop for the processing of semi-structured datasets, which was tested with the entirety of the Protein Data Bank using structural analysis and molecular docking applications. Benchmarking shows this appears to be faster than the same batch jobs submitted to the Openlava (LSF based) batch-scheduler cluster on the same apparatus.
- A scalable analysis system that employs MapReduce on Spark to quantify sequence-specific deviations in the distribution of mapped reads in RNA-Seq data by examining *intra-exon k-mers*. The system can be run using local distributed computing architecture, or can be deployed to be hosted by cloud service providers offering Spark, and is, therefore, suitable for systematic study of large, high-throughput sequencing datasets

1.5 Publications

Research covered in this thesis resulted in a number of publications. This is a list of the publications in chronological order.

1. Alnasir, J. and Shanahan H. P. (2015). Investigation into the annotation of protocol sequencing steps in the sequence read archive. *GigaScience*, 4:23.
2. Alnasir, J. and Shanahan, H. P. (Ed.) (2017). Transcriptomics: Quantifying non-uniform read distribution using MapReduce. *International Journal of Foundations of Computer Science. (Accepted 21/11/2017)*
3. Alnasir, J. and Shanahan H. P. (2017). A novel method to detect bias in Short Read NGS RNA-Seq data. *Journal of Integrative Bioinformatics*, 14:3.

1.6 Conferences

Research covered in this thesis has also been presented at a number of conferences. This is a list of the conferences in chronological order.

1. Alnasir, J. and Shanahan H. P. (2014). Annotation of next-generation sequencing protocol steps in the SRA (sequence read archive) and Big Data for Science. 27. Abstract from Developing data-driven Biology in Morocco, Tangier, Morocco

2. Alnasir, J. and Shanahan H. P. (2015). Applying Apache Hadoop, Hive and MapReduce to Legacy Systems and Applications. *Big Data Analytics Training Workshop, MSTI ENSIAS*, Rabat, Morocco.

3. Alnasir, J. and Shanahan, H. P. (2015). PDB-Hadoop: Parallelising user applications on the Protein Data Bank using Apache Hadoop. *Poster session presented at 3DSig Structural Bioinformatics and Computational Biophysics* 2015, Dublin, Ireland.

4. Alnasir, J. and Shanahan, H. P. (Ed.) (2016). Transcriptomics on Spark Workshop: Introducing Hercules, an Apache Spark MapReduce algorithm for quantifying non-uniform gene expression. *Abstract from CloudTech'16*, Marrakech, Morocco.

5. Alnasir, J. and Shanahan, H. P. (Ed.) (2016). Transcriptomics: Leveraging a MapReduce algorithm and Python for gene-expression analysis on Apache Spark. *Abstract from MatBio '16, King's College London, United Kingdom.*

Chapter 2

Distributed Computing on Biological Datasets

"Divide each difficulty into as many parts as is feasible and necessary to resolve it" – Rene Descartes

In this chapter we will briefly introduce notable distributed computing technologies, but will focus in particular on those that we have employed in this research thesis. Much of this thesis will focus on MapReduce technologies, specifically Hadoop in chapter 5 used in batch mode analysis of molecular structure data, and Spark in chapter 6 to analyse short read transcriptomics data. As a comparison, we have also explored other technologies, in particular a batch-scheduling cluster implementation called Openlava. This chapter also briefly introduces some relevant distributed computing technologies such as grids and High Performance Computing using Message Passing Interface. However, in order to cover the technologies used in this thesis, we focus most of this chapter on the discussion of batch cluster systems (section 2.2 onwards), and MapReduce on Hadoop and Spark (section 2.4 onwards) prior to their application to biological datasets, which is covered in chapters 5 and 6. Finally, we shall look at a selection of 'omics, bioinformatics and structural biology applications that utilise these distributed computing technologies.

2.1 Distributed computing

Distributed computing is a model in which data, software and computational resources are distributed amongst networked computers termed *compute nodes*. Compute nodes are grouped together and are collectively referred to as a *cluster*. The compute nodes in a distributed system communicate with each other by passing messages across the

network in order to coordinate their activity to achieve a common computational goal. This improves performance through concurrency (Coulouris et al., 2005). Distributed systems are also characterised by their lack of a global clock, and are independent of the failure of individual components. A global clock is a requirement in parallel systems which process multiple data items simultaneously for each clock cycle. Concurrency and independent component failures are important issues we will examine in more detail when we discuss Hadoop and MapReduce (section 2.4). The focus of this research thesis is the analysis of large biological datasets, such as sequencing data that often constitute Big Data, and which are becoming increasingly large and difficult to process (Stephens et al., 2015). That said, distributed computing systems encompass a variety of topologies and architectures which we will introduce in the sections that follow.

2.1.1 Clusters, grids and clouds

Distributed computing topologies, that is the way in which constituent computing component parts are interrelated and connected through networks, can be fundamentally categorised as clusters, grids and clouds (Hussain et al., 2013). The main architectures in distributed computing are: High Performance Computing (HPC), batch-scheduled cluster computing and Hadoop cluster computing which we discuss in the remaining sections of this chapter with more emphasis on the latter two architectures. The main topologies in distributed computing are briefly described below:

Cluster: A group of compute nodes that are integrated through network hardware and software to appear as a single computing resource. With the exception of High Performance Computing clusters (outlined in section 2.1.2) and NOWs (Networks of Workstations (Arpaci et al., 1995), not covered herein), clusters can comprise commodity hardware, for instance off-the-shelf equipment. An important point regarding describing clusters raised by Dongarra et al. (2005) is that because of the variety of ways in which clusters can be constructed, and taking into account their increasingly frequent assembly using commodity hardware, the term cluster is often used too broadly without delineating between the various sub-categories of clusters, a distinction that has important consequences for the way in which they function and are programmed for. As this research thesis is concerned with commodity clusters, we use their definition of them: "a parallel computer exclusively comprising commodity computing subsystems and commercial networks such that the computing nodes are developed and employed in stand-alone configurations for broad (even mass) commercial markets, and the networks are dedicated to the private use of the cluster (non-worldly).". Clusters comprise of compute nodes that are generally interconnected through high-speed, low-latency dedicated switches and, as they typically use the TCP/IP protocol,

are allocated to a subnet. Clusters may be composed of a collection of *homogenous* compute nodes, which is preferable for optimal performance in Hadoop (Rao et al., 2012), or may be a *heterogeneous* collection of compute nodes each having a specialist purpose, for example dedicated storage, increased CPU cores or GPU capabilities. This chapter will focus discussion on batch-scheduler and Hadoop clusters.

Grid: A grid uses the internet as a medium for pooling computing resources which are interconnected in a mesh network topology, whereby each node co-operates in distributing network data, and for which there is no centralised point of control. Foster et al. (2008) have produced a check-list of what constitutes a grid, and point out that in order to broker resources, grids use standard, open, discoverable protocols and interfaces to facilitate dynamic resource-sharing with interested parties. An interesting paradigm of grid computing known as volunteering distributed computing has evolved out of the growth of the internet during the 2000s onwards and involves allocating work to volunteering users on the internet (commonly referred to as the @home projects) with tasks typically executed while the user's machine is idle (Krieger and Vriend, 2002; Beberg et al., 2009). Grid computing, however, although used in scientific computing, is beyond the scope of this research thesis.

Cloud: A cloud provides access to computing resources via the internet through a service provider and with minimal human interaction between the user and service provider. Resources are accessed on-demand, generically as a service, without regard for physical location or specific low-level hardware and software configuration (Smith and Nair, 2005). This has been made possible by the developments in virtualisation technologies such as Xen, and Windows Azure Hypervisor (WAH) (Younge et al., 2011; Barham et al., 2003). Services are purchased on-demand in a metered fashion, often to augment local resources and aid in completion of large or time-critical computing tasks. Mell et al. (2011) at NIST (National Institute of Standards and Technology) have provided a detailed definition of a cloud, and as they outline there are various service models in cloud computing. In the Software as a Service (SaaS) model, the service provider manages (automatically) the hardware, network and low-level software configuration required to provide resources as services - for example, web servers, and databases. In the Platform as a Service (PaaS) model programming languages, libraries, services, and tools supported by the provider can be used to deploy user-created applications or acquired applications. The Infrastructure as a Service (IaaS) model allows provisioning of more fundamental resources such as processing, storage, networks and other services that allow the user, for example, to install operating systems and user-created software and libraries. In chapter 5 we will discuss deployment of a Hadoop distributed application for structural Biology on a well known cloud-computing service

provider Amazon Web Services (AWS) (Miller et al., 2010a).

2.1.2 High Performance Computing and MPI

High Performance Computing (HPC), also known as "supercomputing", is a distributed computing model in which computing resources are aggregated in a way to provide more processing power to solve large and computationally complex problems, such as those involved in simulation and mathematical modelling that are too large for a standard computer, or which would take too long to be feasible. To achieve this, HPC systems are often reliant on a high degree of inter-process communication, for instance Message Passing Interface (MPI), which we will discuss later in this subsection. Although HPC systems are typically clusters, their compute nodes are typically specialist, non-standard computers unlike those used in clusters consisting of standard "commodity" hardware. HPC compute nodes usually have especially designed architecture, i.e. specialist processors (Brochard, 2006), storage and high-speed networking interconnects that allow them to communicate at high speed and process and store high-throughput data (Hill et al., 2000). The type of interconnect employed in compute cluster differentiates between commodity clusters and HPC clusters. Commodity clusters predominantly use off-the-shelf, Ethernet networking interconnects, such as Gigabit, whereas HPC clusters make use of higher speed, albeit near-commodity, networking technologies such as Infiniband. Shipman et al review the scalability of Infiniband in two different MPI implementations that support it - MVAPICH which is the most widely used MPI distribution for Infiniband, and OpenMPI. They found that in using OpenMPI, latency in the communication of small messages was improved by up to 10% in medium/large jobs, and that as much as 300% decrease in memory usage can be achieved (Shipman et al., 2006). Specialisation in processing architecture is owed to the history and origins of supercomputing - as early as 1960s a taxonomy for the specific categories of processor architecture employed by HPC systems was established by Flynn (1966). As a result of the dedicated hardware used in HPC, specialist *low-level* distributed programming knowledge, for example Message Passing Interface (MPI) discussed below, that utilises this architecture has been necessary to make use of HPC systems.

Message Passing Interface (MPI) is a standard for the development of parallel programming software and libraries for parallel computing architectures that standardises syntax and which was initially defined in 1992 by the MPI Forum, a group of researchers from academia and industry (Message Passing Interface Forum, 1993). Hitherto the formulation of MPI, no standard specifications for parallel programming were defined and various systems existed albeit with many having similar semantics. MPI facilitates concurrent programming by specifically dictating the standard syntax to be used for

the messages passed between communicating processes. As MPI incorporates features that were developed for prior existing parallel systems, it supports a wide variety of architectures such as multiple computers with distributed memory, shared memory multiple processors and heterogeneous combinations of these.

MPI provides a variety of communication functions that facilitate inter-process communication - for instance, using user-defined datatypes it is possible to transfer a triangle of data within a matrix to another process by specifying the vector start positions and lengths. This allows for only the requisite parts of data in the matrix to be sent across the network thereby minimising network traffic and memory-to-memory copying (Dongarra et al., 1995). MPI, therefore, offers extremely fine granularity, i.e. high resolution, of control over the processes involved in the execution of parallel, concurrent programs running across networked computers in clusters. Although MPI is extensively used in High Performance Computing, it can be used on clusters of standard machines or workstations. MPI requires the programmer to explicitly handle parallel functionality at a lower level than for instance Hadoop, which automates parallelism of users programs through the MapReduce formalism. Whilst MapReduce has parallels to MPI programming, especially in relation to MPI functions *scatter* and *reduce*, it offers automatic parallelism, as well as **data-locality** and **fault-tolerance** (Jin and Sun, 2013) - important features we will discuss later in this chapter. The focus of this thesis is the application of distributed computing technologies to high-throughput biological datasets.

2.2 Batch-scheduled cluster computing

A batch-scheduler (also referred to as a job-scheduler or workload management software), is a central software component of a compute cluster that manages the workload of computational jobs on a cluster and allocates cluster resources to them (Kaplan and Nelson, 1993). We will refer to the technique as batch-scheduling and the central software component as the job-scheduler. Generally, a computational job on a batch-scheduled system is a normal user program that runs on a single compute node, but can also be a more specialist distributed program that comprises components written to run on multiple nodes which communicate by passing messages, for example using Message Passing Interface (MPI) (discussed in section 2.1.2). The generalised architecture of a typical batch-scheduled system is depicted in Figure 2.1.

Computational jobs are submitted in a batch to the job scheduler which adds them to a queue. The job scheduler decides on how to prioritise competing jobs and what resources to allocate to the jobs. The jobs are then submitted by the job scheduler to

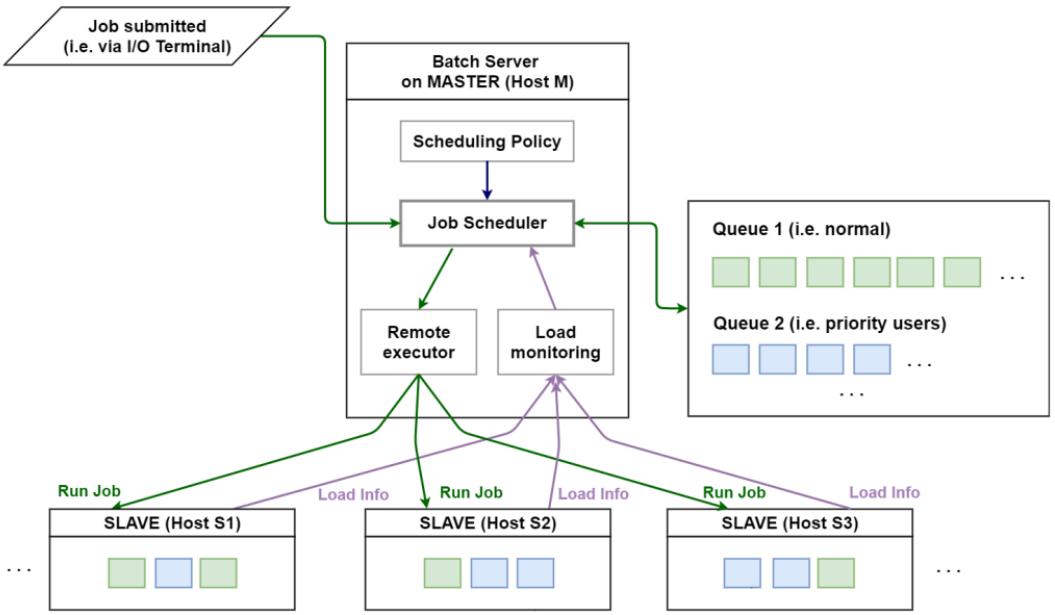


Figure 2.1: Generalised architecture of a batch cluster. Users submit jobs to a batch-server that typically has a job-scheduler as a main component. The job scheduler maintains a set of queues in which jobs are placed. Jobs are executed according to a scheduling policy and the load (CPU and RAM utilisation) on the target slave compute node. The jobs are "sent" to run by the remote executor, that is a process communicates with the slave node to ensure the job is executed on that machine. NB: It's also possible for the master compute node, on which the batch-server runs, to run jobs. Frequently though in practice the master node is set to provide less resources for running such jobs or is disabled from doing so in order to prevent the master node, which co-ordinates activity on the cluster, from being overloaded and to allow it to be more responsive.

compute nodes of the cluster using a scheduling algorithm (Chun and Culler, 2002).

Batch-scheduled cluster systems couple job flow control with the ability to reserve (and limit) the allocation of cluster resources such as, for example, CPU cores, physical RAM, virtual memory and execution time. This allows cluster resources in tenanted clusters, that is systems that are used concurrently by multiple users, to be shared amongst users in different ways, for example different measures of resources can be allocated to these users (typically in groups). The provision of multiple queues, which typically have different priorities or may be available only to certain users as well as assignment of job priorities, allows administrators to define priority users, thereby allowing more important jobs to be completed first (Zhao and Stankovic, 1989).

2.3 Platform's LSF/Openlava

Platform Computing's (now part of IBM) Load Sharing Facility (LSF) is a batch-scheduler comprising of a suite of UNIX command-line workload management software which has also been released opensource as the Openlava project (under the GNU license). In this section we focus on Openlava which is typical of the established batch-scheduler technology used in structural biology. We employ Openlava and discuss its performance in chapter 5 where we assign it as a control in a benchmarking comparison between batch-scheduled cluster computing and Hadoop cluster computing for a large-scale application on the Protein Data Bank (PDB).

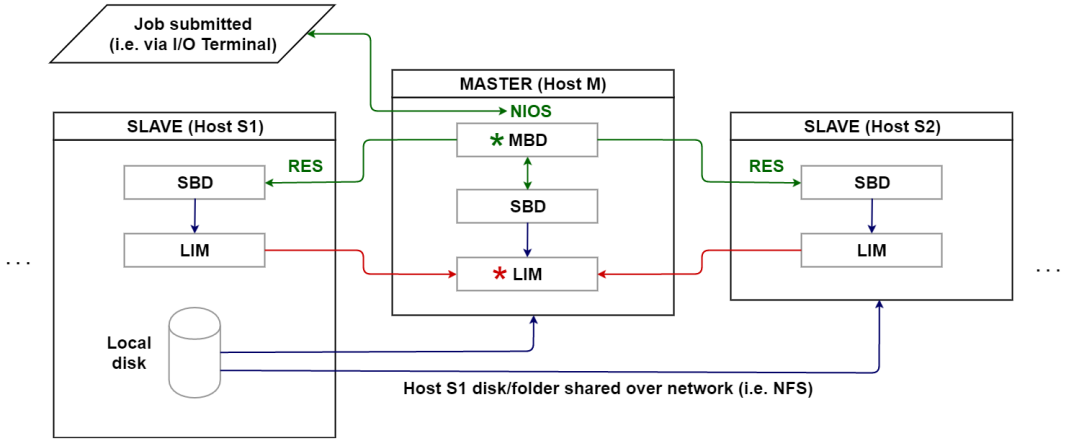


Figure 2.2: Architecture of an Openlava cluster comprising of a master node (host M) and two slave nodes (hosts S1 and S2).

Openlava batch-scheduler clusters, an example of which is depicted in Figure 2.2, have a tree hierarchy with two levels in which a master compute node is at the root and slave compute nodes comprise the child nodes of the master. An Openlava cluster consists of a number of core components - software daemons - that run on the compute nodes and communicate with each other to coordinate the activity of the cluster such as monitoring the load on each node, executing jobs on a remote host and handling I/O from the jobs. The ubiquitous components that run on all of the compute nodes are the Load Information Manager (LIM), Slave Batch Daemon (SBD), Remote Execution Service (RES) and Network I/O Service (NIOS). The Master Batch Daemon (MBD) runs only on the master compute nodes to accept jobs which are submitted to slave nodes. At the slave node level, LIM monitors the load (process utilisation and memory usage) on the node it runs on and relays this information periodically to a designated master LIM process. Usually this is LIM on the master node, but if a master LIM becomes unavailable, then the next LIM, as designated in the cluster configuration, is

used as the master. SBD receives batch jobs from the MBD that have been submitted to the cluster via the *bsub* command[1]. Openlava jobs can be submitted from any node, and this is made possible by a pair of (coupled) daemons which communicate to achieve this: NIOS which runs on a slave compute node (where the job was submitted, typically from a terminal) accepts the jobs and executes them on the remote host through RES (Remote Execution Service) on the remote host, whilst handling the standard I/O to and from the executing job.

Openlava provides a number of features, most notably: **Fair-share scheduling**, which allows resources to be allocated to users according to policies that can be configured by an administrator; **Job pre-emption** which ensures that critical users, groups and jobs can gain access to resource while other jobs are running; **Scalability** that allows addition/removal of compute nodes in an easy manner and which is cloud and VM friendly. However, batch-scheduled clusters offer only "course granularity" control of concurrency at the job-level (unlike MPI systems) and do not render the same level of fault-tolerance and data-locality through lack of a distributed file system such as HDFS that Hadoop provides (discussed in section 2.4.3). Batch-scheduled cluster systems use the job scheduler to deploy "whole" executable programs to compute nodes of the cluster which may or may not run in a parallel fashion - for instance a single program when submitted will run on only one node, while multiple submitted jobs may run either on a single node or be distributed across multiple nodes depending on the load on each compute node and the scheduling rules set. Hadoop, however, distributes function components of programs (such as *map* and *reduce* functions, discussed in the next section) to the nodes ensuring parallel execution. This is because the input data to these distributed function components of Hadoop MapReduce programs is first automatically partitioned into chunks of data known as *splits*. Each split is processed by an instances of a distributed function across one or more nodes. Even if a Hadoop MapReduce job is running on a single node, multiple copies of the function will be operating on the split data in a single *container* - a container is a bundle of CPU and memory resources which we will discuss further in section 2.4.4. Batch-scheduled cluster systems, therefore, are generally easy to construct and use, and require no specialist programming knowledge. They are advantageous in running batches of many conventional compute jobs that are distributed amongst hosts in the cluster, albeit with an associated computationally administrative overhead (for the execution of tasks by software daemon components of the cluster) for each job submitted.

[1] It is noteworthy that Openlava is command-line compatible with LSF.

2.4 Apache Hadoop and MapReduce

Apache Hadoop is a software framework for distributed processing and storage which is typically installed on a Linux compute cluster to facilitate large scale distributed data analysis, though it can be run on a stand-alone single computer node usually for the purposes of prototyping. Hadoop clusters may be built using commodity hardware, for instance off the shelf equipment such as used in computer farms, and key features are **fault-tolerance** and **data-locality**. For fault-tolerance, scaling up a cluster to add more machines and disks increases the probability of a failure occurring. For data-locality, this provides the ability of the framework to execute code on the same node, or at least the same rack of the cluster as where the input data resides – colloquially this is termed "bringing the compute to the data, as opposed to bringing the data to the compute". This reduces the amount of network traffic during processing, thereby avoiding network bottlenecks (Guo et al., 2012). The fault-tolerance and data-locality of Hadoop are made possible by its distributed file system (HDFS) which replicates blocks of filesystem data across different nodes in the cluster.

The Apache Hadoop project is a software "ecosystem", that is a collection of inter-related, interacting projects forming a common technological platform (Messerschmitt et al., 2005; Joshua et al., 2013). Table 2.1 lists the projects that are part of the Hadoop ecosystem.

Project	Description
Ambari	A web-based tool for provisioning, managing, and monitoring Apache Hadoop clusters which includes support for Hadoop HDFS, Hadoop MapReduce, Hive, HCatalog, HBase, ZooKeeper, Oozie, Pig and Sqoop.
Avro	A data serialisation system.
Cassandra	A scalable multi-master database with no single points of failure.
Chukwa	A data collection system for managing large distributed systems.
HBase	A scalable, distributed database that supports structured data storage for large tables.
Hive	A data warehouse infrastructure that provides data summarisation and ad hoc querying.
Mahout	A scalable machine learning and data mining library.
Pig	A high-level data-flow language and execution framework for parallel computation.
*Spark	A fast and general compute engine for Hadoop data. Spark provides a simple and expressive programming model that supports a wide range of applications, including ETL (Extract Transform and Load), machine learning, stream processing, and graph computation.
Tez	A generalised data-flow programming framework, built on Hadoop YARN, allows execution of an arbitrary DAG (Directed Acyclic Graph) of tasks to process data for both batch and interactive use-cases. Serves as a replacement to Hadoop MapReduce as the underlying execution engine.
ZooKeeper	A high-performance coordination service for distributed applications.

Table 2.1: Projects that are part of the Apache Hadoop ecosystem (Apache Software Foundation, 2016a). *We make extensive use of Apache Spark in chapter 6 for transcriptomics analysis.

In this research we have extensively utilised Apache MapReduce and Spark (chapters 2, 5 and 6). MapReduce, which we will discuss in the next subsection, is a programming paradigm employed by both Hadoop and Spark for the execution of distributed algorithms across a Hadoop or Spark cluster which is dependent on Hadoop's Distributed File System (HDFS). We have also used YARN (Yet Another Resource Negotiator) as the Hadoop and Spark cluster resource scheduler, which we will discuss later in section 2.5. In the following sections we discuss MapReduce, HDFS, YARN and Spark in further detail.

2.4.1 Notation used

Throughout this thesis we will adopt a notation for data structures and MapReduce operations. In particular, we make frequent use of the tuple data structure that comprises

a number of variables which are enclosed in angled parenthesis as follows:

$$\langle a, b, c \rangle \tag{2.1}$$

Other examples are:

$$\langle a \rangle \tag{2.2}$$
$$\langle a, b \rangle \tag{2.3}$$
$$\langle a, b, c, ... \rangle \tag{2.4}$$
$$\tag{2.5}$$

We can therefore also represent a set of N tuples which can be indexed by i and the first index starting at 1 as:

$$\langle a, b, c \rangle_{i=1}^{N} \tag{2.6}$$

The variables a, b, c can be strings, integers, floating-point numbers, a function, or more complex constructions - for instance b could represent another tuple that is nested in another $tuple2$, such as follows:

$$b = \langle 1, 2, 3 \rangle \tag{2.7}$$
$$tuple2 = \langle a, b, c \rangle \tag{2.8}$$
$$\therefore tuple2 = \langle a, \langle 1, 2, 3 \rangle, c \rangle \tag{2.9}$$

Much of this thesis discusses MapReduce, and we therefore follow the conventions used by Fish et al. (2015). An important structure in MapReduce is the key-value pair, which we represent as a tuple of the form $\langle k, v \rangle$ where k is the key and v is the value. It is important to note that in the key-value pair, the key k is a unique identifier and is typically a string, whereas the value v can be of any type - v might be a string (for instance DNA or Protein sequence), a real number (for instance an average), an integer, or a list of values in a nested tuple as described in the $tuple2$ example above - v may therefore be any data structure. As we shall discuss in the next section, MapReduce programs are implemented by map and reduce functions which we denote by the Greek letters μ and ρ, respectively.

2.4.2 MapReduce formalism

MapReduce is a programming model and implementation used by both Hadoop and Spark to enable parallel execution of algorithms which are distributed across a cluster. MapReduce is designed for processing and generating large data sets, and is based on the map and reduce functions commonly used in functional programming (Hughes, 1989) and which are conceptually similar to the scatter and reduce functions in the Message Passing Interface (MPI) standard (Snir, 1998). We will discuss map and reduce functions in the context of Hadoop and Spark shortly. Hadoop and Spark ensures that programs that implement MapReduce are automatically parallelised and executed on the cluster in a distributed fashion. The infrastructures of Hadoop and Spark consist of a cluster resource scheduler, distributed file system and associated runtime libraries and is fault-tolerant. This is an important requirement for scalability because the larger the number of machines and disks, the higher the probability of failure, and this is addressed by filesystem data replication which we will discuss in section 2.4.3. The infrastructures handle the detail of partitioning the input data, and scheduling the program's execution across the set of machines (referred to as compute nodes) as well as the requisite inter-machine communication (Dean and Ghemawat, 2008). With this in mind, MapReduce has been designed to facilitate programmers in utilising the distributed computing resources without prior specialised expertise in concurrent programming such as MPI and also provides fault-tolerance and ease of scalability (Reyes-Ortiz et al., 2015).

As mentioned above, key components of the MapReduce implementation are the map and reduce functions. The main data structure associated with these is the key-value pair $\langle k, v\rangle$ where k is the key and v is the value. An input to a Map Reduce function is a list of N key-value pair tuples $\langle k, v\rangle_{i=1}^{N}$ Fish et al. (2015) and definitions of map and reduce functions specifically their inputs and outputs are given below:

Mapper: A function, denoted by μ, that accepts a single key-value pair $\langle k,v\rangle$ and returns list of key-value pairs, as follows:

$$\langle k_i, v_1^{'}\rangle ... \langle k_i, v_N^{'}\rangle \tag{2.10}$$

Reducer: A function, denoted by ρ, that accepts a single key k and a list of values: $\langle k, v_1, ..., v_m\rangle$ associated with that key and returns the same key with a smaller list of values (often just a single value) for the same key, as follows:

$$\langle k, v_1^{'} ... v_M^{'}\rangle \tag{2.11}$$

A MapReduce program typically comprises a number of map and reduce steps, and there can be any number of these. The Hadoop and Spark platforms manage the execution of the map and reduce functions as well as the input, output and intermediate data generated. Input data is first partitioned into chunks known as *splits* which enable map and reduce functions to operate independently on partitions of the data on the same or different compute nodes of the cluster depending on where the data resides. This serves the purpose of data-locality and also, as the data is replicated across the distributed file system (HDFS), allows for fault-tolerance. The execution of the steps on a Hadoop cluster involves an important intermediate processes termed *shuffle* which is performed between map and reduce functions. The purpose of the shuffle is to group intermediate data by key to ensure the next map or reduce function receives only key-value pairs for a given key. Multiple map or reduce steps are then performed for all key-value pairs for all keys $\langle k, v \rangle_{i=1}^{N}$.

As we will make use of MapReduce in chapter 5 for application in computational biology and in chapter 6 for transcriptomics analysis it is useful for us to introduce the programming model by way of an example. A common introductory example is a MapReduce program to count word frequencies in input text and which comprises a map and reduce function, its working is shown in Figure 2.3. The map function μ splits the text into individual tokens where each token represents a word and serves as the key in the key-value pair, and assigns a 1 to the value of the token. The next function is a reduce function ρ which serves to aggregate values of the same key through summation, and in this example returns a single value. The result is that the frequency of words in the input text is computed in a distributed fashion across the cluster and without requiring the programmer to explicitly deal with the allocation of work to each node of the cluster.

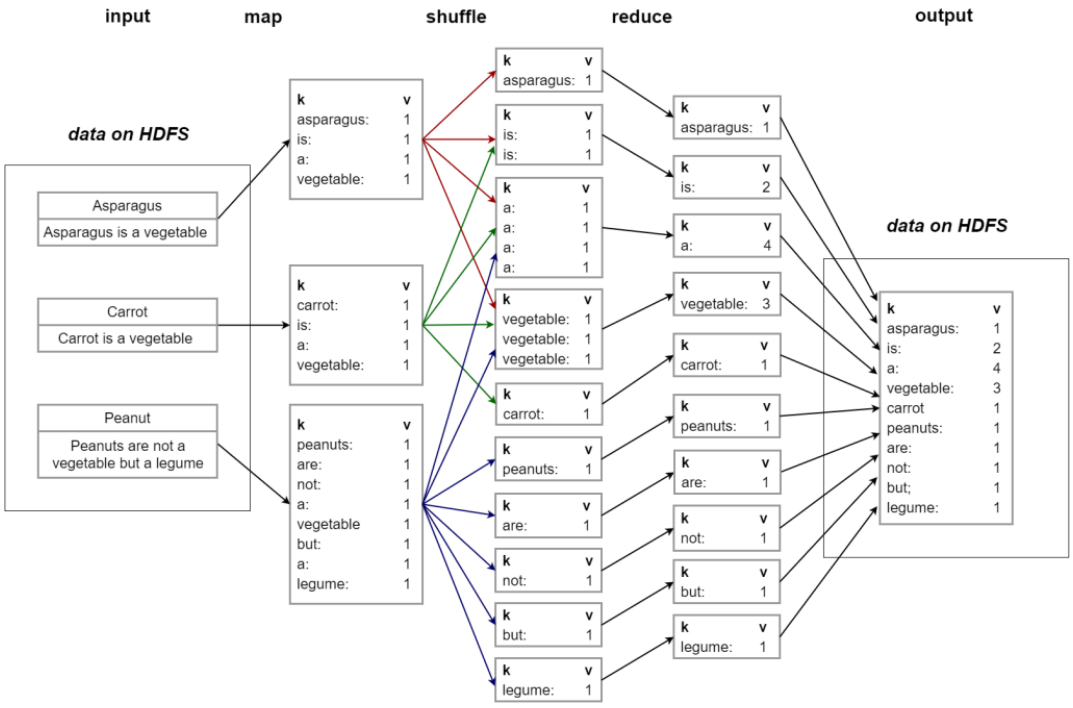


Figure 2.3: An example of the MapReduce processes in a program to count frequency of words in input key-value pairs.

Algorithm 1: Word count example map and reduce functions. Multiple map and reduce functions operate on portions of the input data known as *splits*. The reduce function takes a group of partitioned key-value pairs for the same key k and returns the aggregated sum of the values for those keys.

```
1  function map(line);
   Input  : Input string of text (line).
   Output: ⟨k, v⟩_{i=1}^{N}, key-value pairs, where k=word, v=1
2  lstWords ← call line.split(" ");          // tokenise the text into separate words
3  for each word in lstWords do
4  |  emit ⟨word, 1⟩;
5  end
6  function reduce(kvGroup);
   Input  : ⟨k, v⟩_{i=1}^{N}, group of key-value pairs for the same key (kvGroup)
   Output: ⟨k′, v′⟩, key-value pair, where k=word, v=frequency
7  s ← 0;
8  for each v in kvGroup do
9  |  s ← s + v;
10 end
11 emit ⟨k, s⟩;
```

Although it may be argued that implementing programs using the MapReduce model constrains application development, the *quid pro quo* is that MapReduce applications gain from a massive parallelisation on account of the scalability of the Hadoop or Spark platforms on which they run. Large datasets, for instance those rapidly generated by high-throughput technologies such as Next-generation sequencing (Stephens et al., 2015; Ward et al., 2013), have sufficient volume such that they cannot be processed by a typical standalone computer (Laney, 2001; Borgman, 2015). These datasets can be easily processed using MapReduce on Hadoop or Spark and gain from the inherent parallelism in the method. The power of this technology was demonstrated in 2009 in what could be considered a milestone event - using Apache Hadoop, Yahoo! were able to win the 2009 Gray sort competition by sorting 500 GB in under 1 minute (59 seconds), and 100 TB of data was sorted at a rate of 0.578 TB/minute (OMalley and Murthy, 2009). In chapter 5 we demonstrate that, owing to the architecture of the Hadoop platform, MapReduce methods are also more efficient than traditional batch-scheduling clusters in processing very large amounts of data, and in chapter 6 we describe a method that uses Spark for the analysis of transcriptomics data. In the next section we will discuss the architecture of important components of the Hadoop

platform such as its distributed filesystem and resource scheduler which facilitate parallelisation of MapReduce.

2.4.3 HDFS, a distributed filesystem

Hadoop utilises a distributed filesystem, that is a storage system whereby the blocks of each file are distributed (and often replicated) and stored across multiple nodes of the cluster. Files on a distributed filesystem can be accessed *as if they reside on a local file system, on a single node*. Hadoop's filesystem is called Hadoop Distributed File System (HDFS). It is designed for the storage and processing of large datasets (files of typically Gigabytes to Terabytes in size), and this is achieved by splitting larger files into smaller chunks [2] to distribute amongst individual compute nodes which store the underlying data on the local filesystem. HDFS is central to the Hadoop platform's main features - data-locality and fault-tolerance, because HDFS, in addition to partitioning and storing data across many nodes, also replicates filesystem data blocks. These characteristics allow Hadoop and MapReduce programs to be executed in parallel, for instance on different portions of input data at different locations in the cluster, and to provide data redundancy - a replication factor of 3, that is three copies of each filesystem block is set by default in Hadoop. Moreover the architecture of HDFS enables Hadoop clusters to be scaled massively, increasing compute, storage and I/O capacity by adding new compute nodes. Compute nodes are assigned a unique identifier which is independent of IP address, which is interrogated when a node joins the cluster. This serves to ensure all nodes are using the same version of the Hadoop/Spark software so as to preserve system integrity and prevent data loss or corruption (Shvachko et al., 2010).The process of making hardware modifications is managed by the cluster configuration which is obscured from the applications and so scaling up does not require users to modify their Hadoop applications. This allows Hadoop and MapReduce programs to leverage massive scalability and has enabled, for example, Yahoo! (in 2010) to build a Hadoop cluster of 25,000 nodes that stores 25 petabytes (10^{15}) of data (Shvachko et al., 2010).

Figure 2.4 depicts the architecture of the HDFS distributed filesystem in a Hadoop cluster. HDFS stores filesystem blocks and their metadata on separate servers and the infrastructure therefore comprises a NameNode to store metadata and DataNodes to store data blocks. DataNodes also provide fault-tolerance by storing replicated blocks on different DataNodes. Filesystem integrity is maintained by communication between the NameNode and DataNodes by information packets termed *heartbeats* which re-

[2] The typical block size is 64 MB but this can be adjusted in the HDFS configuration files. Unlike filesystems such as FAT or NFS, HDFS blocks can be half-filled and are not rounded up to the block size.

lay status information from the DataNodes back to the NameNode. The NameNode can issue commands to the DataNodes by replying to the heartbeats with a response that contains instructions to, for instance, replicate blocks, remove duplicated blocks, shutdown or request further status information (Apache Software Foundation, 2016c).

Hadoop applications and MapReduce programs access data in a parallel fashion through HDFS and are managed by Hadoop's resource scheduler YARN (Yet Another Resource Negotiator) which we shall discuss in more detail in section 2.4.4. It is noteworthy that the Apache Spark platform (which we have utilised extensively in chapter 6) offers the choice between its own built-in scheduler and running Spark MapReduce programs through YARN. We have elected to use the latter because the cluster (machine name: *Bigdata*) employed in our department (CSSB - Centre for Systems and Synthetic Biology, dept. of Computer Science, Royal Holloway) is used for both Hadoop and Spark, and given YARN is the main scheduler for Hadoop, it is logical to avoid the scenario in which two resource scheduling systems on the cluster share the same resources.

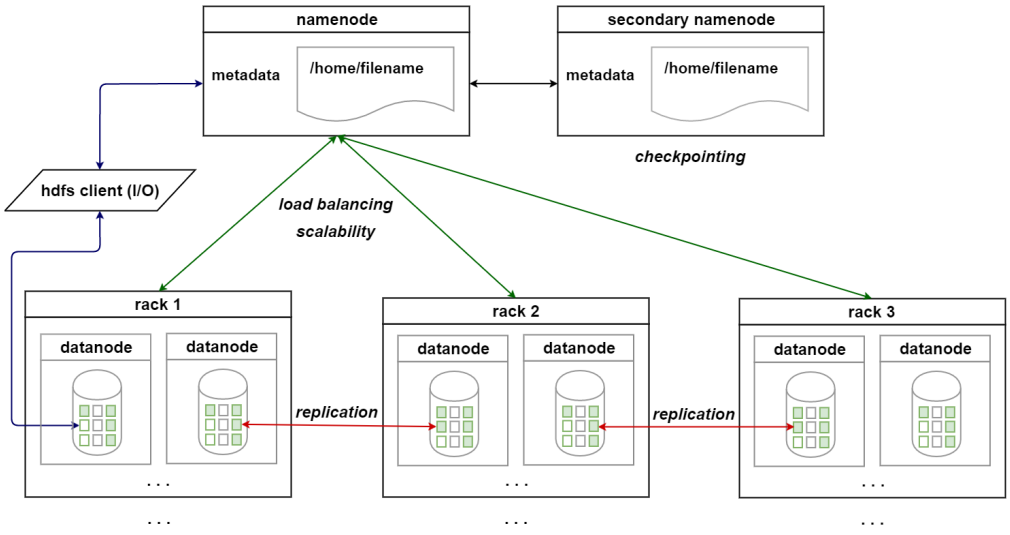


Figure 2.4: Architecture of the HDFS filesystem of a Hadoop cluster. The NameNode (and its secondary backup) contains metadata for filesystem blocks and periodically communicates with DataNodes within racks via heartbeats (Green arrows). Heartbeats contain status information from the DataNodes, and instructions from the NameNode to the DataNodes. HDFS blocks are replicated across DataNodes of the cluster (Red arrows). A HDFS client can read/write HDFS blocks by first contacting the NameNode to find the block location and then directly communicate with the DataNode where the block resides (Blue arrows). The HDFS client exports a filesystem interface to allow for file system access via a Hadoop application or Linux command line.

2.4.4 YARN, a job and resource scheduler

YARN (Yet Another Resource Negotiator) is a core component of the Apache Hadoop framework as of version 2.0 that is responsible for cluster management, in particular resource allocation and job scheduling. When MapReduce jobs are run on Hadoop (or Spark), YARN acts as a central resource manager to instantiate MapReduce processes which are run in virtual containers - a container is a logical group of resources such as ⟨4 GB RAM, 2x CPUs⟩. In YARN, container resources are stipulated in the clusters configuration file *yarn-site.xml* which, amongst other parameters, defines the amount of RAM available for map and reduce functions running in a container. The architecture of YARN is depicted in Figure 2.5 and is somewhat analogous to that of HDFS in having a central master component the ResourceManager RM, as well as compute node level client components known as NodeManagers NM which communicate using heartbeats in a similar fashion to HDFS. Additionally YARN has an important *per-job* component known as the ApplicationMaster AM which coordinates aspects of the job lifecycle such as execution flow control, handling faults, and dynamically increasing or decreasing resource consumption (Vavilapalli et al., 2013). For each application, a single instance of the AM runs on a compute-node level in a container and handles scheduling and creation of containers in which MapReduce processes run.

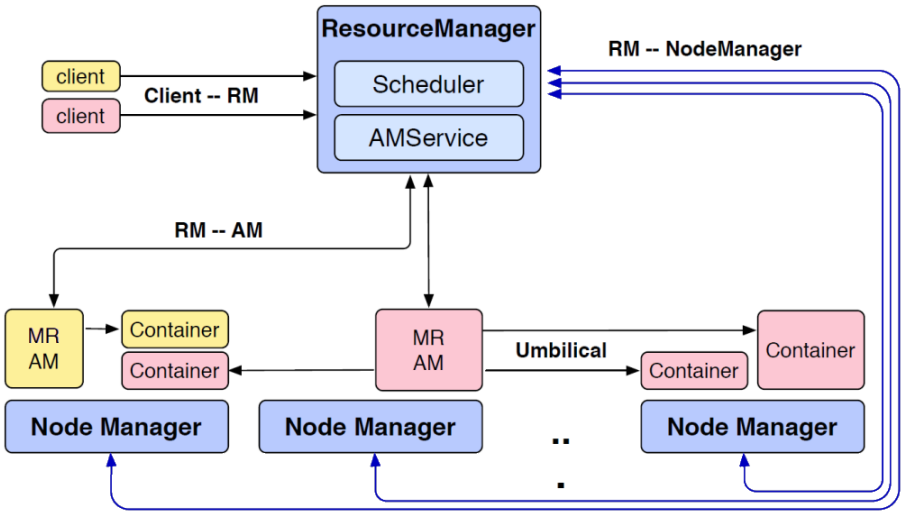


Figure 2.5: Architecture of YARN showing system components (Blue) and two MapReduce (MR) applications running (Pink, Yellow) each having their own Application Manager (AM) container. Image from Vavilapalli et al. (2013)

Let us consider a typical YARN configuration scenario, a cluster comprising of slave nodes each with 32 GB of physical RAM could be configured for a total of 28 GB

allocated for YARN and 4 GB remaining for the Operating System. In such a set-up, allocating YARN a container size of 4 GB would result in a maximum of 7 container processes (each executing a mapper or reducer process) per compute node at any one instance. The main cluster we have utilised (in chapters 5 and 6 of this thesis) has the same such YARN configuration and is a machine comprised of one master which has a E5-2620 hexa-core CPU @ 2.10 GHz and five slave nodes, each of which has 32 GB of RAM and an Intel Xeon E3-1220v3 4-core CPU @ 3.1GHz. The master node in this configuration is configured only to schedule jobs and not to partake in running them directly. Furthermore we have utilised Amazon web services (AWS) for testing of the application framework for computational biology on the Protein Data Bank described in chapter 5, this has allowed us to test and deploy our research on the cloud.

YARN offers three scheduling modes, namely *FIFO* (First in first out), *capacity* and *fair* which provide flexibility in how the submission of multiple jobs is managed during cluster operation. Our departmental cluster is multi-tenanted and is therefore configured to use *capacity* scheduling mode to maximise cluster utilisation in the case where multiple user jobs may be running. Discussion of these modes is beyond the scope of this thesis and the reader is directed to the reference book by Murthy et al. (2013) that covers YARN scheduling in detail.

2.4.5 Hadoop streaming

Hadoop streaming is a useful feature of Hadoop that allows *map* and *reduce* functions to be written in any programming or scripting language that supports the UNIX POSIX standard I/O streams *stdin* (standard input) and *stdout* (standard output) (Apache Software Foundation, 2016b). This facilitates the development of *map* and *reduce* functions as external executables (binary executables or scripts). These external map and reduce functions exchange data with the Hadoop framework, hence it is termed *Hadoop streaming*, by reading input (line by line) from *stdin* and writing output, a process also known as *emit*, to *stdout*. As depicted in Figure 2.4.5 they are spawned and monitored by the Hadoop system as a separate process for each operation (i.e. *map* or *reduce*), until the Hadoop job is completed. The key-value pairs $\langle k, v \rangle_{i=1}^{N}$ which serve as input and outputs of these functions are exchanged over the streams in tab-delimited format, although this is configurable.

As Hadoop is implemented in Java, a MapReduce job that employs Hadoop streaming is initiated by running the Hadoop streaming jar file from the command line and specifying the HDFS input and output folders, as well as the paths to the external *map* and *reduce* executable programs. Given that these external programs are spawned by Hadoop and important consideration is the communication of status by

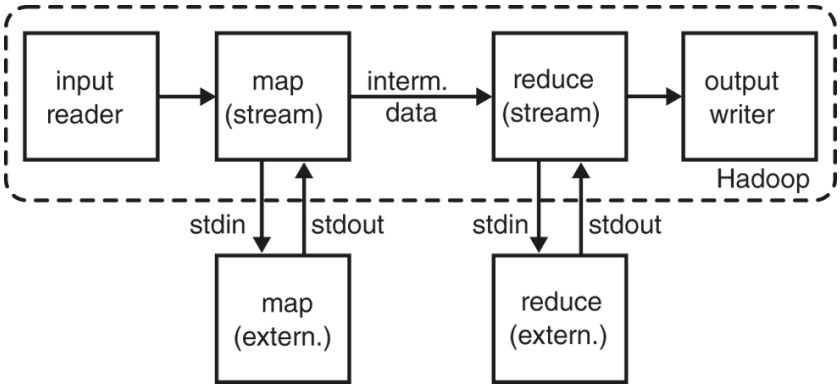


Figure 2.6: Exchange of data in Hadoop streaming which takes place between the Hadoop system and externally implemented *map* and *reduce* functions. Communication occurs over the standard input and output streams *stdin* and *stdout*, respectively. Image from (Big Data Technology blog, 2015)

the external executable to the Hadoop system. External applications must achieve this by returning a UNIX exit status of `zero` if the *map* or *reduce* step was successful, or `non-zero` if otherwise. For this reason, an optional Boolean parameter ($stream.non.zero.exit.is.failure = True|False$) may be specified on execution of a Hadoop streaming job to allow or prevent Hadoop from treating a failed *map* or *reduce* step as a whole MapReduce job failure.

We will make use of Hadoop Streaming MapReduce in chapter 5 in which we develop a system employing a *map* step implemented in BASH for the purpose of analysing semi-structured datasets in structural biology.

2.5 Apache Spark

Spark is a cluster computing framework that utilises a distributed file system (HDFS) and a resource scheduler (in this research thesis we use Apache YARN) and provides an application programming interface (API) for distributed computation. Spark is designed to overcome some of the constraints of Apache Hadoop offering significant performance improvements but keeping the fault-tolerance and scalability features of MapReduce by utilising HDFS (Shanahan and Dai, 2015). The main constraint Spark overcomes is Hadoop's acyclic data flow model by allowing the re-usability of intermediate data in the form of a data structure that is central to Spark, the Resilient Distributed Dataset (RDD). The RDD serves as an abstraction for distributed memory that allows in-memory computations on large clusters in a fault-tolerant manner, and because an RDD is partitioned across multiple compute nodes, can be rebuilt if

for some reason a partition is lost (Zaharia et al., 2012). The RDD is an immutable (read-only) data structure that can encapsulate objects from a number of different programming languages and typically on Spark this is Python, Java, or Scala. In contrast Hadoop development, notwithstanding that Hadoop MapReduce programs can be written in any language supporting the UNIX POSIX standard I/O streams (section 2.4.5), is predominantly Java based. This is significant from a development perspective because Java requires programmers to possess more specialist object-orientated programming (OOP) knowledge than for example Python, and Java programs tend to have more dependencies on runtime libraries. Furthermore, Python is now the most popular language in the bioinformatics field.

An RDD is usually created in two ways, by referencing an external data source, for instance a dataset or file on HDFS, or by parallelising an existing Spark datastructure. Parallelising a datastructure, for instance an array in Spark, allows it to be operated on in a parallel fashion whereby Spark handles the caching of the RDD in memory across nodes of the cluster. As a result of this, Spark MapReduce operations gain significant performance enhancements over their Hadoop counterparts. This can be achieved because Spark creates an execution plan, in the form of a directed acyclic graph (DAG), of Spark and MapReduce operations to be performed on RDDs. The execution plan models dependencies and allows Spark to optimise execution of a jobs' components in a way that is not constrained to linearity of execution like MapReduce on Hadoop. Spark categorises some functions/procedures in Spark programs as *action* events, for instance a *reduce* step, and others as *transformations*, for instance a *map* step. This allows Spark to process the execution plan DAG using a method known as *lazy evaluation*, that is only *action* events cause data to be loaded into memory, whereas *transformations* in the execution plan are only executed when an *action* with dependency on that transformation is executed (Apache Software Foundation, 2014). This method improves on cluster utilisation over Hadoop because it allows cluster resources to be sequestered and released on an *ad-hoc* basis throughout a complex job. This avoids the rather less desirable but typical scenario in Hadoop in which resources (for instance executor processes - which are finite and dictated by cluster configuration) are reserved at the outset then released back to the cluster (hence making them available to other jobs) only when the running job is finished.

Apache Spark supports Python (through the PySpark API) allowing rapid development with easily installable modules, and offers substantial performance benefits over Hadoop. It is noteworthy that other popular languages in the field of bioinformatics, such as R, also have interfaces to Spark - for example SparkR (Apache Software Foundation, 2015). For these reasons we have elected to use Spark in the analysis of large

transcriptomics datasets, which we discuss in detail in chapter 6.

2.6 Application of MapReduce to Bioinformatics

The emergence of MapReduce based platforms that we have discussed such as Hadoop and Spark have not been overlooked by researchers in bioinformatics. A number of projects within the Apache Hadoop ecosystem find useful application in bioinformatics. Taylor provide a good introduction to these in his review paper (Taylor, 2010). These include the data-warehousing framework Hive (Thusoo et al., 2009) which has an SQL type query language, the high level data-flow language Pig (Olston et al., 2008) which compiles scripts into sequences of MapReduce steps for execution on Hadoop, the machine-learning and clustering facilities offered by Mahout (Lyubimov and Palumbo, 2016), and HBase a distributed scalable database (George, 2011). All of these projects utilise Hadoop's cluster infrastructure and distributed file system (HDFS) and therefore gain from the scalability and fault-tolerance inherent in their design, as discussed earlier in section 2.4.

In terms of software applications MapReduce has been employed for a variety of problems in processing biological and sequencing datasets. Some notable projects in the area of sequence alignment are, Cloudburst (Schatz, 2009) and CloudAligner (Nguyen et al., 2011), which are both based on the RMAP alignment algorithm (Smith et al., 2008), and CloudBlast (Matsunaga et al., 2008) which is based on the BLAST algorithm (Altschul et al., 1990). For de novo genome assembly, that is assembly of sequence reads without a reference genome, there is a de Bruijn graph based implementation using MapReduce named Contrail (Schatz et al., 2010b).

There are also tools implemented in MapReduce for the analysis of assembled sequencing data, for instance Crossbow (Langmead et al., 2009) is designed for SNP (Single Nucleotide Polymorphism) detection. It uses the Bowtie (Trapnell et al., 2009) and the SNP caller SOAPsnp (Li et al., 2009b). Differential expression (using RNA-Seq) can be measured using the Myrna software pipeline (Langmead et al., 2010) - pipelines are data-flows comprising of sequential steps in which bioinformatics software are applied to the data Leipzig (2016).

Additionally, a number of programming libraries that facilitate the manipulation and processing of sequencing data file formats such as SAM Sequence Alignment Map and BAM (Binary Alignment Map) have arisen such as the Java based libraries Genome Analysis Toolkit (GATK) (McKenna et al., 2010) developed by the Broad Institute and Hadoop-BAM (Niemenmaa et al., 2012) as well as the Scala based SparkSeq (Wiewiórka et al., 2014). The GATK provides functions for data management in the form of

data access patterns, that is low level implementation is separated from higher level functions, and also provides functions for analysis calculations. The Broad Institute have also developed a Workflow Definition Language (WDL) for use in data analysis pipelines (discussed in the next section). It is a high-level language that is designed to be human readable and writable, it allows researchers to describe analysis tasks, daisy-chain tasks into workflows, and utilise advanced features such as parallelization (Broad Institute, 2016b). WDL was developed out of the necessity for standardisation amongst a number of different pipeline solutions, thereby providing a universal standard. In order to execute analysis pipelines written in WDL, an execution engine is necessary. Cromwell is such an engine, also designed by the developers of WDL, to run on any platform (Locally, HPC, Google - support for other platforms such as Microsoft Azure and AWS is forthcoming) and can scale elastically to workflow needs (Broad Institute, 2016a).

Provision of pipeline development specifically for the Hadoop platform is also available. For instance, SparkSeq is a MapReduce library for building genomic analysis pipelines using Scala on Apache Spark. Whilst Scala is supported on the Spark platform it lacks the same user base in bioinformatics as it enjoys amongst the data analytics and machine learning communities. Given the vast amounts of sequencing data being produced (Stephens et al., 2015; Ward et al., 2013), the purpose of these tools is to exploit the scalability that is characteristic of the MapReduce and which the Hadoop and Spark platforms offer, and this offsets any difficulty in developing or re-writing applications using the MapReduce paradigm. It is noteworthy that MapReduce can be especially suited for, for example the construction of a de Bruijn graph for de Novo genome assembly. For example, Contrail is able to build adjacency lists for all the *k-mers* in the genomic sequence reads and then uses distributed aggregation functions such as *reduce* to compress simple chains of length N in $O(log(N))$ rounds using a randomized parallel list ranking algorithm (Schatz et al., 2010b). However, the development of universal standards, such as WDL offers researchers a means of utilising tools developed for such platforms in a more user-friendly way.

This thesis also puts these technologies to use by employing MapReduce on Hadoop in chapter 5 for batch mode analysis of molecular structure data, and we will develop a system using MapReduce on Spark to quantify sequence specific deviation in read distribution in short read transcriptomics data in chapter 6.

2.7 Cloud-service providers and Bioinformatics

As discussed in section 2.1.1, cloud service providers offer "computation-as-a-service" through the provision of hardware and software resources, on-demand, as a service. A consequence of the advancements in high-throughput next-generation sequencing (Dai et al., 2012) and the data-driven and integrative nature of bioinformatics (Dudley et al., 2010), is that cloud-services such as for instance Amazon AWS, Microsoft Azure, and Google Cloud, are increasingly being used in research (Schatz et al., 2010a; Shanahan et al., 2014)

Stein outlined the problems facing the bioinformatics research community, in particular those research projects that routinely work with sequencing data, that would drive researchers to increasingly utilise cloud-services (Stein, 2010). The key issues cited were increasing data production, the falling cost of sequencing relative to storage costs, and the costs of running and maintaining in-house computational resources. It also pointed out that existing working practices for dealing with sequencing data were somewhat inefficient. For example, researchers have traditionally made use of the wealth of experimental data being deposited in public database repositories by downloading their datasets across the internet for local storage and processing. Often these datasets are mirrored locally and synchronised periodically - a process that is costly, inefficient and error-prone. In section 2.4.2, we discussed the concept of data-locality, that is bringing the compute to the data in the context of MapReduce. This useful methodology is not only applicable at the low level of software implementation and execution of code on a cluster, but also applies at higher levels, for example at the database access or the research project level. For this reason, cloud service providers are making large datasets accessible to cloud users. AWS for instance, offer a number of Biological datasets through their S3 storage system, for example 1000 Genomes, ENCODE (Encyclopaedia of DNA Elements), Ensembl, GenBank, Influenza Virus, NIH Database of Genotypes and Phenotypes (dbGaP), Model Organism and UniGene. This allows users to deploy their applications to the cloud to connect to and directly access these resources without the need to download and synchronise them locally.

The two main ways in which scientific and bioinformatics research projects utilise cloud services are through IaaS (Infrastructure as a Service) and PaaS (Platform as a Service). In the IaaS approach, processing, storage and networking resources are acquired and leased from the service provider and configured by the end user to be utilised through the use of virtual disk images. These virtual disks are provided in proprietary formats, for instance the AMI (Amazon Machine Image) on AWS or VHD (Virtual Hard Disk) on Azure, serve as a bit-for-bit copy of the state of a particular

VM (Shanahan et al., 2014). They are typically provisioned by the service provider with an installation of commonly used Operating Systems configured to run on the cloud service's Infrastructure, and service providers usually offer a selection of such images. This allows the end user to then install and precisely configure their own or third party software, save the state of the virtual machine, and deploy the images elsewhere. In contrast, in the PaaS approach, the end user is not tasked with low level configuration of software and libraries which are instead provided to the user readily configured to enable rapid development and deployment to the cloud. For example AWS provides a PaaS for MapReduce called Elastic MapReduce which it describes as a "Managed framework for Hadoop that makes it easy, fast, and cost-effective to process vast amounts of data across dynamically scalable Amazon EC2 instances" (Amazon, 2016). In fact MapReduce is offered as a PaaS by all of the major cloud-service providers (Amazon AWS, Google Cloud and Microsoft Azure) (Gunarathne et al., 2010).

Biological data is often analysed is through pipelines that are usually created by researchers to be specifically tailored to the particular analysis needs of their research. Typically they comprise of a series of sequential steps that apply existing bioinformatics software to the data at various stages in the data-flow and are often implemented using scripting tools (Leipzig, 2016). A number of platforms which manage the workflow in pipelines without the need to employ scripting or programming have emerged, such as for example Galaxy (Giardine et al., 2005), Taverna (Wolstencroft et al., 2013) and Pegasus (Deelman et al., 2015) which are also deployable to grids and clouds, typically as a single virtual-machine instance. They are operated through a GUI and allow the user design pipelines by specifying the data to be processed, the software processes to apply to the data, the order of the steps to fulfil the workflow, and provide tools to visualise the results. As well as uploading data from a local disk, data can be also integrated into these pipeline workflows from a variety of external resources such as online databases by specifying URLs to the remote data. This makes it possible to develop pipelines that support an integrative approach to analyses by combining data and tools from different fields. One of the aims of such tools is to facilitate researchers in reproducing experiments which is necessary to verify hypothesis testing (Goecks et al., 2010), although in this respect there is a trade-off between ease of developing frameworks and the ability to develop pipelines comprising of more complex workflows. Additionally, the interoperability of such systems, which each have gained a significant user base, is lacking and can result in duplication of research efforts. To this end a project called Tavaxy has been developed as a solution to integrating existing Taverna and Galaxy workflows in a single environment and is also deployable to cloud service infrastructure (Abouelhoda et al., 2012).

The emphasis of both the Galaxy and Taverna workflow management frameworks is to abstract the technical command-line scripting involved in developing bioinformatics pipelines and are deployed as a single virtual-machine instance, to a local server or to the cloud. Whilst they directly execute bioinformatics software tools within the same virtual machine instance, or call remote services, they have no inherent control over low level underlying processes such as virtualisation or the distribution of computational work across multiple nodes of a cluster. The Galaxy workflow documentation states that the framework is designed to run jobs locally on a single system but can be run on a cluster through a Distributed Resource Management Application API (DRMAA) (Troger et al., 2007), which allows Galaxy to submit jobs to a cluster (Galaxy project, 2017). For Taverna, there is a plug-in written in Java for the Taverna framework which allows it to run fully distributed jobs on a PBS (Portable Batch System) cluster (Taverna PBS, 2017). Whilst the plug-in is available for this project (Center for Public Health Genomics, University of Virginia,, 2017), it appears the project web page and supporting documentation are no longer available.

An important concern with utilising the cloud for bioinformatics, in particular for medical and clinical research applications, is that of data security, especially considering such raw data can be patient identifying (Kahn, 2011). Although not within the scope of this thesis, it is important to mention that there is ongoing research exploring cloud-security in the context of bioinformatics (Rocha and Correia, 2011; Subashini and Kavitha, 2011), and of particular relevance there are a number of interesting research papers on applying, for instance, Homomorphic encryption (Rivest et al., 1978) to sequencing data (Kantarcioglu et al., 2008; Lauter et al., 2014).

Research into the use of cloud-service providers for the development and hosting of applications in bioinformatics has been undertaken since as early as 2009, for example in publications by Qiu et al. (2009) and Wagener et al. (2009). Additionally there are extremely large amounts of high-throughput data that are being generated by technologies such as the Next-generation sequencing, which we discuss in the next chapter (Stephens et al., 2015; Ward et al., 2013). Considering such data require specialist methods for processing (Laney, 2001; Borgman, 2015), such as MapReduce, which we have discussed in detail in this chapter, there is need for specialist infrastructure to host and support this work. By offering state-of-the art technologies such as Hadoop and Spark, employing new programming paradigms such as MapReduce which offer amongst other things scalability and fault-tolerance, cloud-service providers are ideally located from a market perspective to host scientific research projects that make use of these resources. A key consideration in this regard, aside from the technical ones we have discussed in this chapter, is whether such services will be cost-effective (Kudtarkar

et al., 2010).

Recent developments have seen a change in the landscape of distributed computing, particularly the emergence of technologies for processing high-throughput data such as Hadoop and Spark, as well as some of the bioinformatics applications developed with MapReduce discussed in section 2.6 that can be hosted on cloud-services. With this in mind, in this thesis we will therefore test and develop distributed applications with MapReduce that are readily deployable to cloud-service infrastructures such as AWS, Azure and Google Cloud. In particular we develop, and discuss in detail, two applications for the analysis of high-throughput biological data. Firstly an application for the analysis of semi-structured molecular data in the Protein Data Bank in chapter 5, which serves as a test and use case, and for the analysis of short-read transcriptomics data in chapter 6, where specifically we develop a system for the quantification of sequence specific deviation in RNA-Seq reads.

Chapter 3

Sequencing and Next-generation Sequencing

"Nothing has such power to broaden the mind as the ability to investigate systematically and truly all that comes under thy observation in life" – Marcus Aurelius

In this chapter we will introduce the fundamentals of nucleic acid sequencing and different sequencing technologies. Since the focus of much of this thesis is on bias, by means of a review of the literature we shall look in some detail at the biases that can be introduced in the protocol steps of a typical sequencing workflow. Finally we will conclude by looking at models that have been devised to quantify and mitigate bias in sequencing data.

3.1 An introduction to nucleic acid sequencing

This chapter examines processes applied to nucleic acids, specifically DNA and RNA, at the molecular level, that occur as a result of synthetic or natural biomolecular processes. DNA and RNA are polymeric, that is they are chains comprised of smaller sub-units called nucleotides (more generally residues or moieties). Nucleotides (depicted in Figure 3.1) are individual molecular structures comprising of three sub-units: a nitrogenous base (Adenine, Thymine, Guanine, Cytosine, Uracil), a five-Carbon sugar (ribose in RNA, or deoxyribose in DNA), and at least one phosphate group.

The synthesis of a nucleic acid chain of DNA or RNA, called a *strand*, typically requires a template strand from which a new complementary strand is constructed. This attribute of complementarity allows nucleic acids to be self-replicating. Fundamentally, A single-stranded molecular chain of DNA (ssDNA) can serve as a template from which

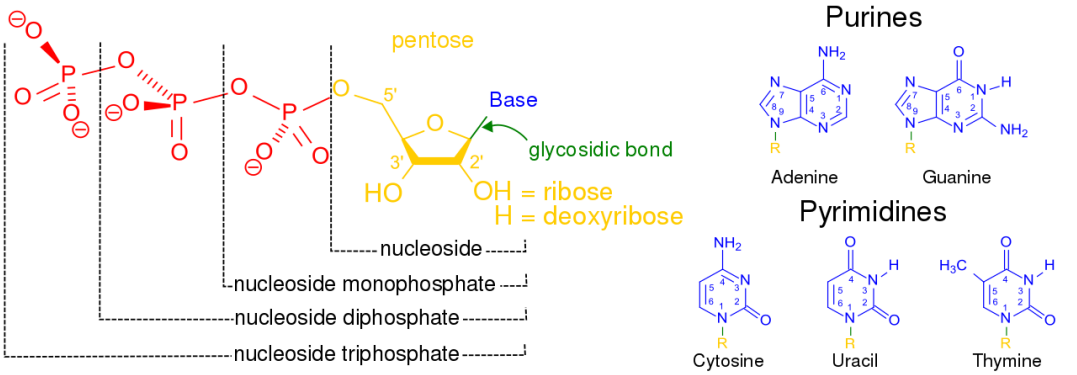


Figure 3.1: The structure of a nucleotide is shown on the left. It comprises a 5-Carbon sugar (a pentose, Yellow), to which a phosphate group is attached to carbon number 5 in the sugar ring - this is the 5' end, and a base (Blue) is attached to carbon number 3 - this is the 3' end. Bases, which can either be single-ring pyrimidines or dual-ring purines are shown on the right. Bases on separate strands bind complementary to each other (A with T, and G with C), and the 5' and 3' ends bind neighbouring nucleotides together. Image from (Wikimedia Commons, 2017).

a double-stranded chain of DNA (dsDNA) is formed by the addition of complementary nucleotides (typically A, T, G, C). A single-stranded ssDNA can, therefore, also serve as the template for the construction of a single-stranded molecular chain of RNA from nucleotides (typically A, U, G, C) - the process is termed transcription because the sequence information in the template strand is transcribed (copied) into a new strand of complementary sequence. The ends of nucleic acid polymer chains are designated 5' and 3' and denote the regions of the five-Carbon sugar (Figure 3.1) to which regions of the nucleotide partake in binding to a neighbouring nucleotide to form a polymeric nucleic acid chain.

The molecular processes mentioned above are facilitated by enzymes, which *catalyse* the reactions, that is they reduce the energy required to activate the reaction between interacting molecules, thereby allowing the reaction to proceed at a higher rate than if an enzyme catalyst was absent. Without catalysis by enzymes such reactions would proceed too slowly to be viable for life to proceed (Stryer, 1998). An important class of enzyme in DNA and RNA synthesis are Polymerases, of which there are many types. These catalyse the addition of a nucleotide into an elongating nucleic acid chain. Importantly, Polymerase activity only occurs in the 5' $\rightarrow$ 3' direction.

In order for the polymerase enzyme to commence catalysis of the reaction, the template strand must first be "primed" with a complementary sequence, called a primer - the process is depicted in Figure 3.2. The primer initiates the beginning of the nascent strand (the emerging strand that is built up from complementary nucleotides) and runs

anti-parallel to the template strand. Priming sequences to which the primer binds (with its complementary sequence) can occur at any position along the template strand and do not need to start at the first nucleotide of the template.

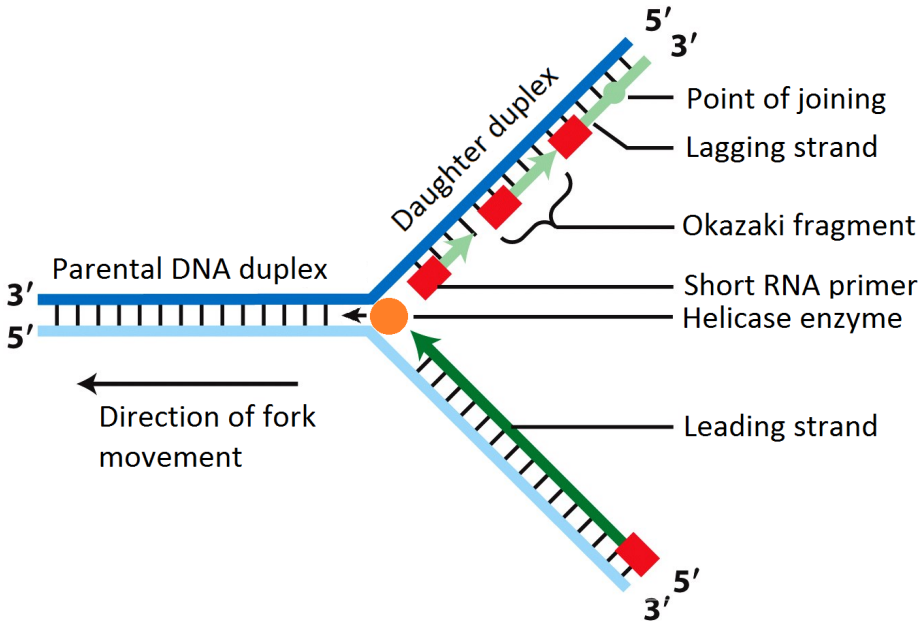


Figure 3.2: Transcription of dsDNA. The two strands are separated by the helicase enzyme (Orange), creating a replication-fork. The 3' template strand of the fork (bottom right) has a complementary 5' end which can be synthesised contiguously. However, the 5' template strand (top right) has a complementary 3' end which cannot be synthesised contiguously (because polymerase operates in the 5' $\rightarrow$ 3' direction). This is, therefore, synthesised in segments called Okazaki fragments, which are primed at multiple regions by a short RNA primer. Image from (Lodish et al., 1995).

These processes are central to the biochemistry of nucleic acids which is discussed in detail in this chapter with respect to sequencing and bias. In the following section we will discuss Sanger sequencing, a foundational method in DNA sequencing.

3.2 Sanger sequencing

In 1977, a method was outlined by which the sequence of nucleotides in DNA could be determined by primed synthesis with DNA polymerase (Sanger et al., 1977). Key processes of the method have seen refinements in chemistry, automation and miniaturisation which have allowed Sanger based methods to be employed in a range of different applications from small-scale (kilobase) to larger scale (megabase) projects. These have allowed fragment read lengths of up to 750bp in modern Sanger derived

technologies, a significant increase from the 80bp that was achievable with the introduction of dideoxynucleotide chain terminator chemistry outlined by Sanger and his colleagues (Tucker et al., 2009) - we outline this in the next section.

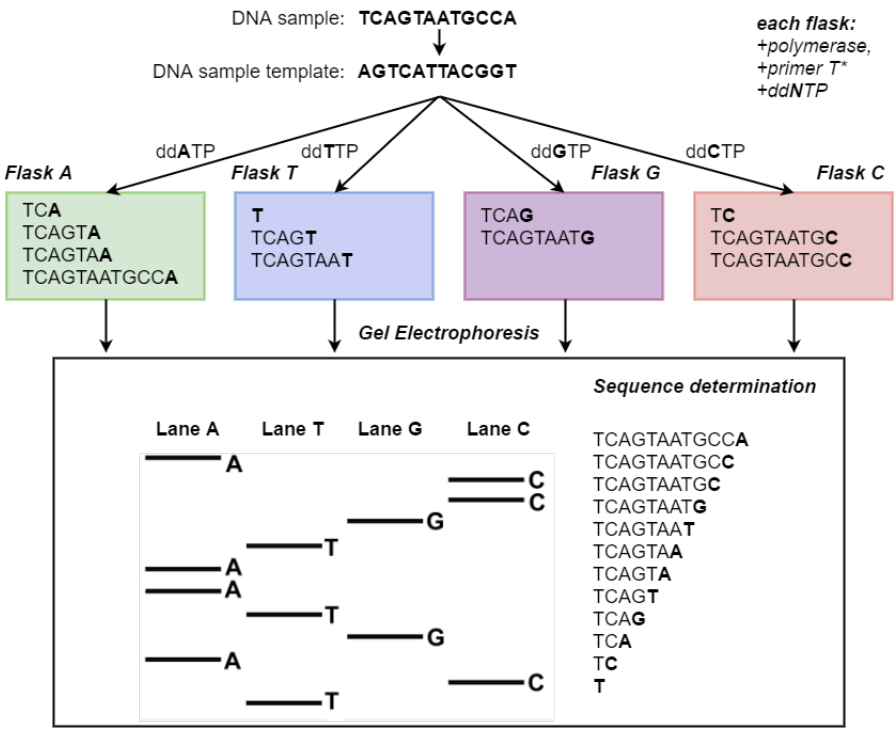


Figure 3.3: The Sanger sequencing method. First, a complementary template is made from the DNA sample. A primer is then added to initiate the reaction and Polymerase enzyme which catalyses the reaction for the addition of each nucleotide to the nascent sequence. Normal nucleotides (dNTPs) are added to each flask in excess, where N is the letter A,T,G,C, with the exception of the specific chain terminating nucleotide each flask represents - to this flask a modified nucleotide (ddNTP) is added. For example, in flask A the modified nucleotide ddATP is substituted dATP. These modified nucleotides i.e. ddATP will terminate the sequence at this nucleotide. Fragments are finally size separated in different lanes by electrophesis on a gel medium to elucidate the original sample sequence.

3.2.1 Sequence determination with DNA polymerase

DNA polymerase is the enzyme responsible for constructing the sequence of bases in the polymer of DNA by catalysing the reaction to join individual chemical units - nucleotides - by phosphodiester bonds. The Sanger method utilises DNA polymerase and involves four reaction flasks each representing a nucleotide (A, T, G, and C) - see Figure 3.3. In each of the flasks is placed DNA polymerase to catalyse the reaction, a primer to initiate the reaction and all four standard nucleotides in excess - the exception that the nucleotide the flask represents is substituted with a modified nucleotide, a di-deoxynucleotide (ddATP, ddTTP, ddGTP, or ddCTP) also known as a chain-elongating inhibitor of DNA polymerase. The di-deoxynucleotide, for example ddATP, differs from its normal equivalent nucleotide dATP by virtue of lacking the 3' OH group which is necessary for the formation of the phosphodiester bond that links the nucleotides to form the DNA sequence. A sequence fragment is then built up by DNA polymerase in sequence until the modified di-deoxynucleotide is encountered which prevents DNA polymerase adding the next nucleotide and therefore terminates the sequence. To each flask is added a template strand of DNA which is made from the original sample (its sequence compliment). When the reactions are complete each flask will contain various lengths of fragments each terminated by the substituted di-deoxynucleotide. The contents of the flasks then undergo gel-electrophoresis which allows separation of the sequence fragments by molecular weight. It is then possible to ascertain the terminal nucleotide of each fragment and therefore the sequence of the original sample from deposits of the fragments on the gel which are positioned according to their increasing molecular weight.

Sanger sequencing in combination with the PCR (Polymerase Chain Reaction) technique (Saiki et al., 1985) in which a single copy or a few copies a strand of DNA can be amplified by several orders of magnitude has pioneered nucleic acid sequencing and thereby transformed molecular biology. The application of Sanger sequencing involves cloning of DNA fragments in BACs (Bacterial Artificial Chromosome) or other suitably sized vector, amplification of templates, the sequencing process itself which involves electrophoresis and finally assembly of fragments into larger contigs. Bias is introduced in Sanger sequencing by molecular processes such as non-specific binding of the primer to the DNA fragments and formation of secondary structures which reduce the accuracy of the sequences determined (Tucker et al., 2009).

A technique known as shotgun sequencing overcomes the short read-length limitations of Sanger based chain termination methods. This is achieved by fragmenting the DNA sample into shorter reads (typically up to 20 kb in size) which are cloned in a vector and that are suitable for chain termination sequencing (Weber and Myers, 1997)

from both ends of the strand (Roach et al., 1995)[1]. The process is repeated and finally *in-silico* assembly (Staden, 1979) of overlapping short read fragments [2] is applied to elucidate the sequence of the original reads (Anderson, 1981; Weber and Myers, 1997).

Next-generation sequencing techniques were developed to overcome the limitations of Sanger sequencing which are mainly due to the lengthy procedures involved, high-cost, low throughput and biases affecting the fidelity of the sequence, which will be discussed in section 3.3.

3.3 Next-generation sequencing (NGS)

Next-generation sequencing methods have revolutionised nucleic acid sequencing largely as a result of the employment of fluorescence-based nucleotide chemistry to generate a light signal on nucleotide incorporation (Soper et al., 2003; Smith et al., 1985; Prober et al., 1987), miniaturisation and massively-parallel sequencing reactions (Mardis, 2006). Though these have, to a degree, simplified the core sequencing process allowing reactions to be performed in clusters to generate enough signal and in parallel to increase throughput, Next-generation technologies share the same complex preparatory procedures (Shendure and Ji, 2008) which will be discussed in detail in section 3.5. Such high-throughput sequencing technologies generate millions to billions of reads in a matter of days and generate large-scale data sets (Metzker, 2010). In this section we shall introduce the main Next-generation sequencing techniques.

3.3.1 DNA-Seq

High-throughput sequencing methods have developed over the last three decades from semi-automated methods to massively-parallel Next-generation sequencing (Shendure et al., 2004). In 1986 the first semi-automated DNA sequencer using fluorescence detection was developed at Leroy E. Hood's laboratory at the California Institute of Technology (Smith et al., 1985). Shortly after, the development of fully automated sequencing machines followed, with the first being the ABI 370 produced by Applied Biosystems in 1987, and later in the same year the Genesis 2000 produced by Dupont (Hall, 1993). During the 1990s two large parallel projects were launched aimed at sequencing the genome of *H. sapiens* - The Human Genome project by the National Institute of Health's International Human Genome Sequencing Consortium (IHGSC) and a simultaneous effort by Celera Genomics run by Craig Venter (Venter et al.,

[1] Each sequence is called an end-read. Two reads from the same clone are known as mate pairs.

[2] Reads are of sufficient length to be aligned with less computational effort than would be required for very short fragments

2001). In February 2001 the drafts of the human genome sequence were published simultaneously by both groups and these endeavours culminated in the emergence of new methods such as shotgun sequencing (Chial, 2008). As mentioned in section 3.1, shotgun sequencing overcomes the short read-length limitations of Sanger based chain termination methods.

In 2005 (Margulies et al., 2005) the 454 Life Sciences Corporation (later acquired by Roche) described a massively parallel sequencing technique that would reduce the cost and increase throughput as compared with state-of-the-art capillary electrophoresis instruments at the time. Instead of Sanger based chain termination methods their strategy utilised pyrosequencing - detection of Pyrophosphate release on nucleotide incorporation through the use of light producing luciferase enzyme (Ronaghi et al., 1996; Ronaghi, 2001) - and achieved an increase in throughput of two orders of magnitude over Sanger sequencing technologies of that period. The pyrosequencing method is also referred to as *sequencing-by-synthesis*. A number of other high-throughput DNA sequencing technologies based on *sequencing-by-synthesis* have emerged during the last two decades, each have specific flow-cell chemistry (Heather and Chain, 2016; Chial, 2008). These have enabled the sequencing of DNA to reach read lengths of hundreds of basepairs, and utilising massively parallelised strategies, production of gigabases of data in a single run.

In this thesis our focus is on RNA-Seq on the Illumina platform, a popular Next-generation sequencing technology that employs pyrosequencing. Figure 3.4 depicts the sequencing process on the Illumina platform. We will also discuss Nanopore sequencing (section 3.3.3) and Microarrays (section 3.4) because Microarrays have contributed extensively to the body of nucleic acid data in public repositories, and Nanopore devices are an emerging NGS technology that offers long read length at extremely low cost which is likely to contribute significant amounts of sequencing data in future.

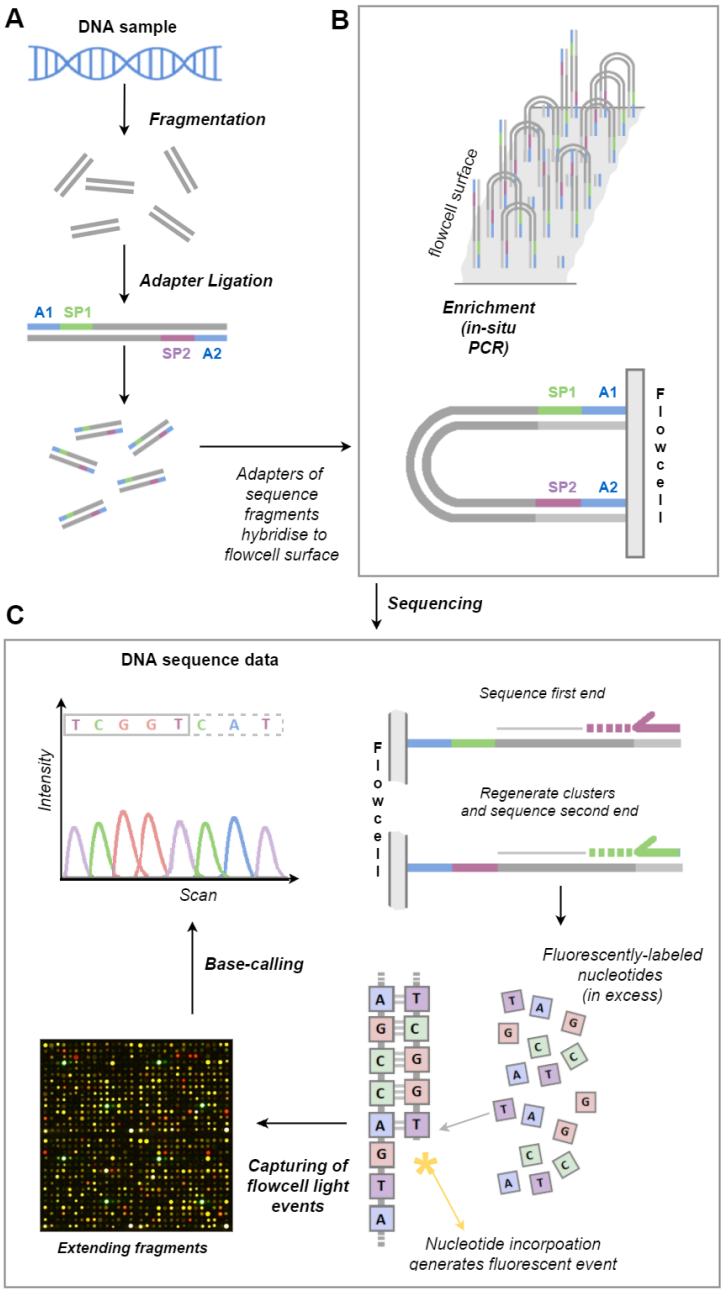


Figure 3.4: Next-generation sequencing on the Illumina platform **A)** Fragmentation and of DNA sample into sizes suitable for sequencing apparatus and attachment of synthetic adapters (A1 and A2). SP1 and SP2 refer to the sequence primers that are included in the adapter sequence at the 5' end. **B)** Enrichment of clusters of DNA fragments (*in-situ*) on the flowcell. **C)** (anti-clockwise from top-left) The first end of the fragments are sequenced, the clusters are re-generated and the second end is then sequenced. Incorporation of nucleotides into the extending sequencing fragment generates light events which are captured. Base-calling determines the sequence of the DNA of extending from the scan trace.

These Next-generation technologies have evolved to transform the field of DNA sequencing facilitating sequencing of the complete genomes of a number of species at lower cost and in shorter time than previous methods. The technological advances in DNA-Seq have been applied to a variety of fields such as Genomics, Evolutionary biology, Transcriptomics, Metagenomics, Medicine and Forensics, and given the unparalleled scale at which data is being produced (Stephens et al., 2015; Ward et al., 2013), pose significant data processing challenges - as noted by Nekrutenko and Taylor (2012) "our capacity to generate such sequencing data greatly outpaces our ability to analyse it". Furthermore whilst Next-generation sequencing technologies may employ different types of chemistry in the sequencing process, they typically share protocol steps for DNA sample preparation prior to sequencing (Shendure and Ji, 2008), and bias may be introduced in these steps which we will examine further later in this chapter.

3.3.2 RNA-Seq

RNA-Seq is a high-throughput Next-generation sequencing technique for estimating the concentration of all transcripts in a transcriptome, this is in contrast to microarrays (discussed in section 3.4), which are constrained to identification and quantification of pre-selected target sequences based on complimentary probes immobilised on the array (Russell et al., 2008). RNA-Seq provides wider coverage of the transcriptome as its methods involve the direct sequencing of transcripts of RNA found in the sample (Wang et al., 2009b; Kukurba and Montgomery, 2015). RNA-Seq can, therefore, be used to study various types of RNA present: total RNA, mRNA, pre-mRNA, and non-coding RNA (ncRNA), such as microRNA and long ncRNA enabling it to be used to study alternative splicing events (Park et al., 2013; Liu et al., 2014). Furthermore RNA-Seq achieves this at a higher resolution (Kukurba and Montgomery, 2015) than other technologies.

The transcriptome can then be constructed by mapping read data back to a reference genome (a process involving the *alignment* of sequences in the read data to the reference). In order to quantify gene expression, this mapping process should be combined with gene boundary information in order to count the number of transcripts that map to a given gene or exon region (Mortazavi et al., 2008; Wang et al., 2009b; Garber et al., 2011). The alignment of short read transcriptomic data to a reference genome is challenging in a number of ways. As Trapnell and Salzberg (2009) et al point out, firstly, the method needs to take into account variation between sequencing reads and their true source in the reference, which result from polymorphisms (insertion, deletion and mutation events), as well as sequencing errors. Secondly, how mapped reads are reported when they map to repeating, but different locations, of the genome. Thirdly,

whilst reads that map entirely within an exon are more straightforward to deal with, what about reads that align across exon boundaries (and therefore also span intron boundaries)? Additionally, there is the challenge of processing the vast amounts of data from sequencing runs that are required to be aligned for scientific investigation. With respect to reads that span intron boundries and align across different exons, sequence alignment tools (colloquially *mappers*), can be classified into non splice aware, and splice aware mappers. Figure 3.5 below, depicts this problem, and describes how each type report the mapped reads.

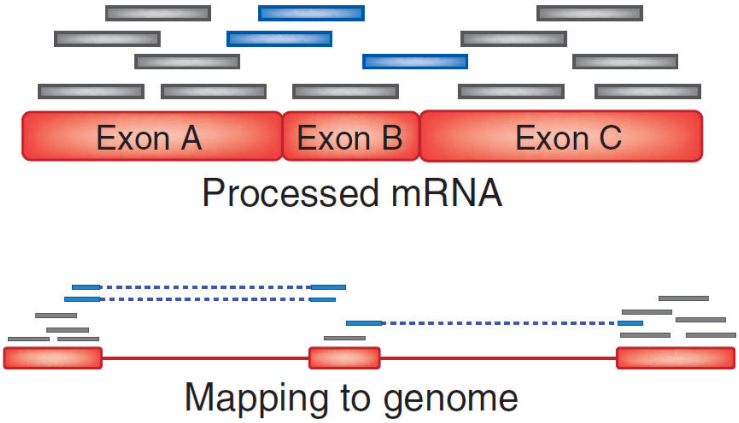


Figure 3.5: This figure is from Trapnell and Salzberg (2009) which provides the following description: RNA-Seq assays produce short reads sequenced from processed mRNAs. Aligning these reads to the genome with a non-splice junction aware aligner (such as Bowtie or Maq) will produce the alignments shown in black but will fail to align the blue reads. A splice junction aware aligner (such as TopHat or ERANGE) will also report the (blue) alignments spanning intron boundaries.

An approach to tackling this problem, and one that is employed by the very popular Tophat alignment tool (a splice junction aware mapper) is to employ Bowtie (non splice junction aware), to first map reads that align within exons and then map the remaining reads that span introns. By virtue of the method, this also yields information about novel splice junctions (Trapnell et al., 2009).

RNA-Seq read data could also be used to construct a transcriptome *de novo* without the need for a reference genome, for instance when a full genome of the species under study is not available. However, given the short reads and alternative splicing de novo assembly of the transcriptome is non-trivial (as noted by (Haas et al., 2013) 1 Gigabyte of RAM is required per million paired end reads, and a transcriptome typically comprises 100s of millions of reads) and has largely been impractical. Algorithms such as Cufflinks can be applied to RNA-Seq data to measure de novo

transcript isoform expression by assembly of transcripts, and estimation of abundances whilst an accompanying program (Cuffdiff) can be used to detect significant changes in transcript expression, splicing, and promoter use (Trapnell et al., 2010).

Unlike microarrays RNA-Seq is not reliant on prior knowledge of the gene structure and can, therefore, be used in applications that go beyond quantification the transcriptome. RNA-Seq can be used to assay junctions between exons, to investigate RNA editing events and allows study of allele-specific expression such as SNPs (Single Nucleotide Polymorphisms) (Malone and Oliver, 2011). RNA-Seq has also enabled the study and quantification of transcript isoforms (splice variants of a gene) and (Richard et al., 2010) describe methods by which alternative isoforms can be predicted and quantified solely from exon expression levels in RNA-Seq data.

RNA-Seq has revolutionised the field of transcriptomics and has transformed our view of the extent and complexity of the transcriptome through deep-sequencing (Wang et al., 2009b) and also as a result of the increased precision the technique offers over other methods. Recent developments in the RNA-Seq workflow, from sample preparation to sequencing have furthered our understanding of the transcriptome but have also required substantial effort for data analysis and computation, and given the complexity of RNA-Seq workflow necessitates study of the bias that can be introduced in the preparatory steps (Kukurba and Montgomery, 2015; Raz et al., 2011). Characterisation of bias in RNA-Seq is especially incumbent given that the method sequences and measures the transcriptome indirectly using reverse transcribed complementary DNA (cDNA) (Ozsolak et al., 2009). We shall discuss the introduction of bias in Next-generation sequencing workflows later in this chapter.

3.3.3 Nanopore sequencing

Nanopore sequencing, first conceived in 1995 (Deamer and Akeson, 2000), and under development since then utilises a nanopore, namely a small opening of approximately 1.4 - 2.6nm in diameter (Wang et al., 2013) in a biological membrane, small enough for a single nucleotide to pass through. Nucleotides that pass through the nanopore induce changes in the electric current across the membrane, the quantification of which can determine which nucleotide has passed through the pore (Branton et al., 2008; Feng et al., 2015b). In May 2015, a new, low cost and compact DNA sequencer was made commercially available by Oxford Nanopore Technologies (Urban et al., 2015; Check et al., 2015). Founded in 2005, Oxford Nanopore Technologies developed the MinION, a USB DNA sequencing device that uses nanopore sequencing. The employment of nanosequencing by devices such as the MinION potentially offers read lengths of tens of kilobases (kb) (Laver et al., 2015) nonetheless noisy. A variety of other nanopore

sequencing devices have also been developed by Oxford Nanopore such as the PromethION and SmidgION, and the VolTRAX which is currently under construction and automates sample preparation (Technologies, 2016).

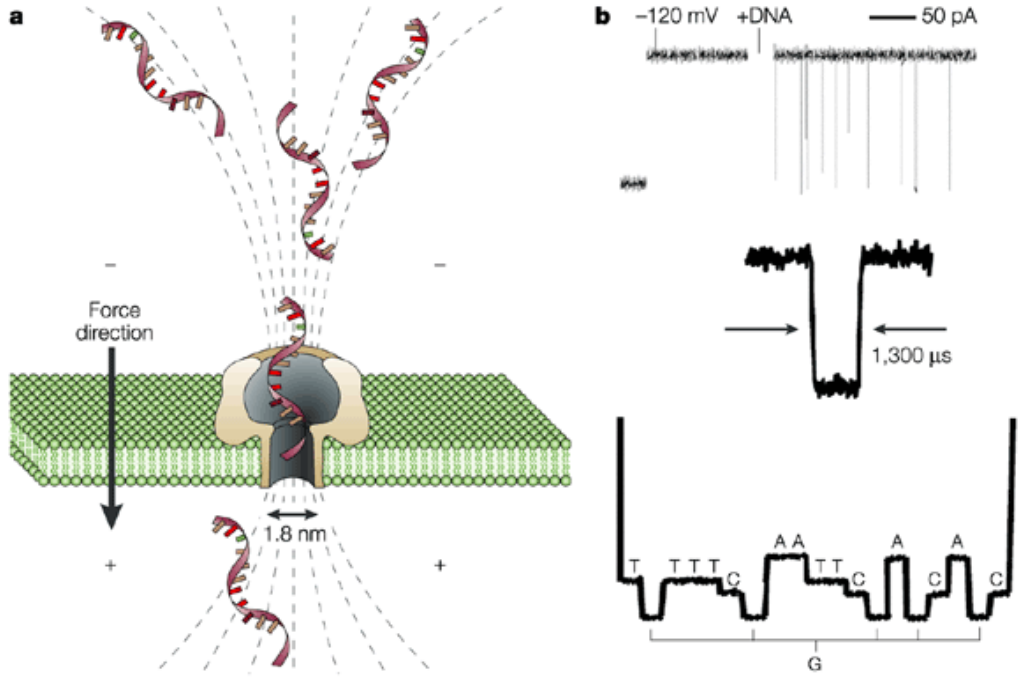


Figure 3.6: **a)** Schematic of the nanopore, lipid-bilayer and measurement technique. Single strands of DNA are drawn through the nanopore of approximately 1.4 - 2.6nm; Bases are identified from a recording of the current and time profile. **b)** Traces of electric current at different time scales. The top two traces represent real data. The bottom trace represents an optimal trace in pA, pico-amperes which has not yet been achieved by nanosequencing technologies. (Part a, original illustration by R. Meller; redrawn with permission from Amit Meller, Rowland Institute for Science; part b, redrawn with permission from D. Branton. Whole image reproduced from (LaVan et al., 2002))

Nanosequencing employed by the MinION, however, suffers from high error-rates (between 5% and 40% error according to (Goodwin et al., 2015), 38.2% according to (Laver et al., 2015)) and when (Mikheyev and Tin, 2014) re-sequenced a lambda phage genome, and amplicons from a snake venom gland transcriptome they observed that a single run generated 150 megabases of raw data with only a quarter of the reads mapping to the reference with lower than 10% average identity. Such error-rates are problematic for genomic assembly in a single run and necessitate additional runs to correct for error (Kilianski et al., 2015), i.e. by increasing the sequencing depth (the number of reads, on average, that are likely to be aligned at a given reference base position).

Despite this, nanopore sequencing offers significant advantages over existing se-

quencing - adapter-sequence-free, ultra-long reads, high throughput and low sample material requirement (Feng et al., 2015b). As noted previously, Next-generation techniques typically employ sequencing by synthesis and require complicated preparation protocols be applied to the sample, including fragmentation to desired fragment length, enrichment of sample by enzymatic amplification and ligation (attachment) of synthetic sequencing adapters. These workflow steps are shared by Next-generation sequencing methods (Shendure and Ji, 2008) but are not part of the nanopore sequencing workflow, and in bypassing these steps the MinION will eliminate bias introduced by these processes (Laver et al., 2015). Given, however, the relatively low cost and extremely long read length as compared with other sequencing technologies, high-throughput nanopore sequencing devices such as the MinION are likely, over the coming years, to be applied to a variety of applications.

3.4 DNA Microarrays and the Transcriptome

Prior to the introduction of RNA-Seq, Microarrays have been the predominant technology in quantifying the transcriptome and have, therefore, generated a significant amount of transcriptomic data to the field of molecular biology. In this section we will provide an introduction to Microarrays.

A microarray (also referred to as a DNA chip) exploits hybridization which occurs between two DNA strands for the purpose of quantifying the relative abundance of specific DNA sequences. It is comprised of microscopic spots of single-stranded DNA (ssDNA) of specific complementary sequence attached to a solid surface (usually by photolithography) known as probes or oligos which bind (hybridise) to fragments of DNA in the sample (termed the targets). Quantification of the sample is achieved by fluorophore- or chemiluminescence-labelling of the nucleic acid sample targets (Soper et al., 2003) which hybridise to probes of complementary sequence on the chip thereby producing a signal. Microarrays have been employed in a range of sequencing applications most notably gene expression profiling using mRNA, SNP detection, GeneID (identification of organisms from signature sequence), Chromatin immunoprecipitation or ChIP (elucidation of protein-DNA binding sites) and alternative splicing studies.

In addition to transcriptomics microarrays are also utilised in the clinical setting, where for instance, it is necessary to determine whether DNA from an individual patient suffering from a disease contains a mutation in genes such as BRCA1 or BRCA2. The arrangement of probes on a microarray allows for many sequence specific features of a disease or condition to be incorporated on a single chip and, therefore, makes them an extremely important diagnostic tool. According to (Trevino et al., 2007) in their 2007

paper, in the clinical setting gene-expression signatures - specific sequences that can be interrogated by microarrays - have been identified for a range of pathologies including, but no limited to: acute lymphoblast leukaemia, breast cancer, prostate cancer, lung cancer, colon cancer, multiple tumour types, apoptosis-induction, tumourigenesis, and drug response.

DNA microarrays have been applied widely in quantification of the transcriptome and have facilitated large scale studies of gene expression allowing the quantification of tens of thousands of transcripts (Sealfon and Chu, 2011). Microarrays are relatively inexpensive and are, therefore, routinely used in genome-wide transcript profiling and in experiments where known sequences are a to be quantified - for instance Clark *et al.* describe the design of microarray oligonucleotides that are specific to spliced RNA in an investigation into mRNA processing in the yeast *Saccharomyces cerevisiae* (Clark et al., 2002). As a result of their low cost and wide range of applications there has been a wealth of data generated from microarrays. As of October 2016, ArrayExpress has a total of 68,149 deposited microarray experiments, 2,092,646 assays totalling 44.35 TB of archived data (ArrayExpress, EMBL-EBI, 2017). GEO has 1,956,365 samples across 4,348 datasets and 21,000 project submissions. (GEO, NCBI, 2016) - this extensive body of work makes microarrays an important source of transcriptomic and SNP data.

Key considerations in the employment of microarrays are the selection of the microarray type and probe design. The selection of probes depends on the type of experiment the microarray will be used in and necessitates probes that are highly specific but also can operate uniformly at given thermodynamic conditions (Russell et al., 2008). In order to ensure target specificity it is important to ensure probes do not hybridise to related target sequences. Furthermore it is necessary to decide on what cross-section or region of the target (sample) is to be interrogated by the microarray experiment and what selection of microarrays (and their probes) are available to meet these criteria. Discussion of probe design is beyond the scope of this thesis but typically is based on genomic sequence or known or predicted open reading frames (ORFs, which are regions of the genome that are likely to be transcribed and translated into RNA and proteins), and each gene model requires multiple probes (Malone and Oliver, 2011).

According to (Russell et al., 2008), the sources of variation in microarray experiments for gene expression estimation fall into the following categories: variation within the biological sample, variation due to performance of the microarray and variation in spot signal measurement.

An advantage of microarrays is that some of the biases present have been characterised and well-researched strategies exist for mitigating such bias (Malone and Oliver, 2011) where the associated project metadata records sufficient experimental informa-

tion. The main disadvantages are that microarrays are constrained to known sequences, and because they are prone to the problem of cross-hybridisation can suffer from a decrease in sensitivity and specificity. For example, it has been demonstrated that widely used Affymetric GeneChip microarrays do not reliably measure gene expression where runs of Guanine are present within the probes (Memon et al., 2010). This effect occurs because Guanine can form alternative Hoogsteen hydrogen bonding allowing runs of four or more Guanine to form stable structures (termed G-Quadruplexes) resulting in cross-hybridisation and interference in measurement of the signal.

3.5 Biases in sequencing workflows

In a typical sequencing workflow a nucleic acid sample prior to being sequenced will undergo a number of steps (fig. 3.7): sample preparation, nucleic acid extraction, chemical modification (blunting, phosphorylation, ligation of instrument specific synthetic chemical sequence adapters) and sometimes chemical amplification (usually using PCR). There are also related sequencing workflows, such as targeted sequencing - these are discussed briefly in section 3.6.4, as well as workflows which do not involve PCR amplification. As we have discussed earlier Next-generation sequencing methods are massively parallel and the results of such high throughput workflows allow the characterisation of millions to billions of reads in a matter of days and generate large-scale data sets (Metzker, 2010). Bias introduced in the preparatory steps can have a profound effect on the raw data and typically manifest themselves as sequence specific or positional biases, whilst bias introduced by the sequencing process itself are often systematic in nature (Meacham et al., 2011) - we will investigate the steps which introduce these types of bias further in the remainder of this chapter.

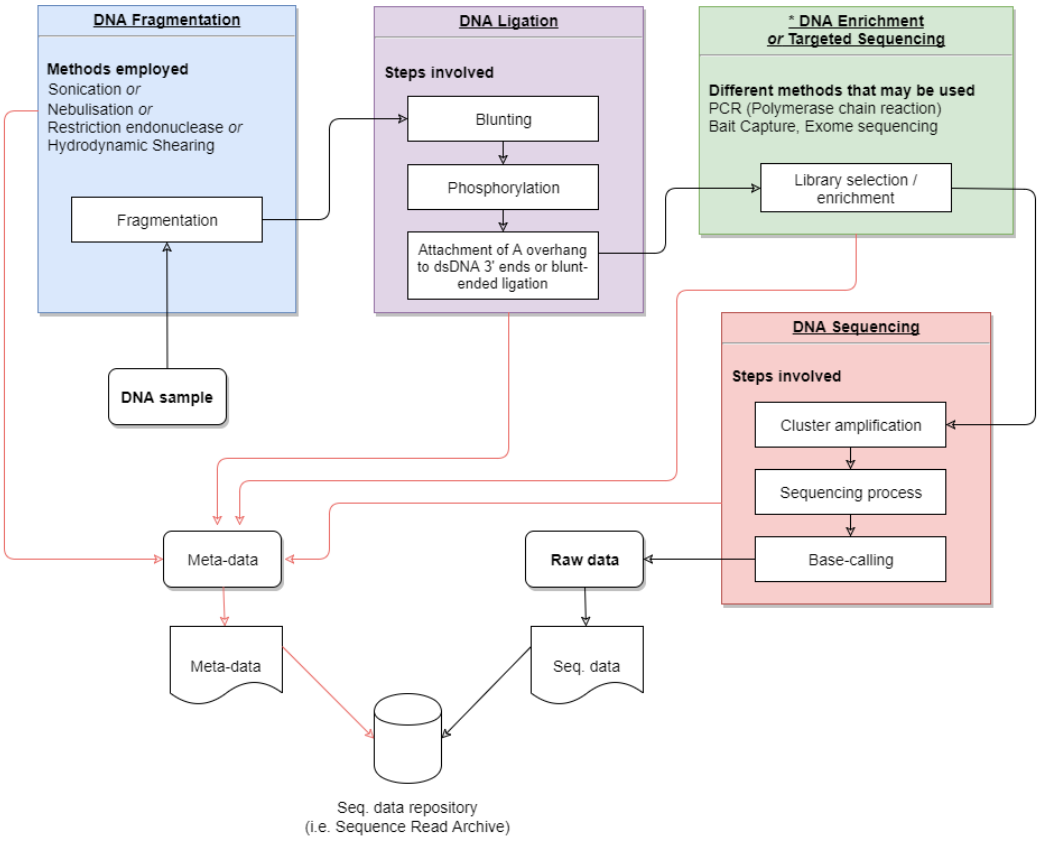


Figure 3.7: A typical Next-generation sequencing workflow. * NB: Some workflows are PCR enrichment free (for instance Whole Genome Sequencing), others may employ a library selection step, known as Targeted Sequencing (such as RNA-bait capture and Exome sequencing, section 3.6.4), where biochemical/biomechanical steps are applied to the sample to select and enrich it for specific fragments. The sequencing workflow is shown by the black arrows; red arrows depict the metadata that should be captured from these sequencing workflow steps.

There is an extensive body of literature within the Bioinformatics community on the workflows to analyse short-read data from next generation sequencers. For example, over 80 papers are listed in the review of Miller, Koren and Sutton in 2010 related to the assembly of sequence read data alone (Miller et al., 2010b). On the other hand, comparatively little work has been done on determining in a thorough fashion how the protocol steps prior to sequencing can affect the final results (Mardis, 2006). As we will demonstrate in section 3.6 some of these steps are known to be prone to introducing bias in the sequencing data derived from the sample. This bias manifests itself as a deviation from the ideal uniform distribution of reads (Ross et al., 2013) and is an important factor in both genome assembly (which requires sufficient reads to form overlaps of sequence to assemble contigs) and impacts on expression studies which

rely on the quantification of a sequence expressed (transcribed) in a sample (Meacham et al., 2011; Lahens et al., 2014). In section 3.5 we discuss the various protocol steps and possible sources of bias from them.

Bioinformatics studies seeking to characterise and model systematic errors are as a result important, and in the case of platform specific biases such methods can be applied in the interim before the technology is refined. In order for these methods to be applied to existing datasets, adequate metadata is required in the repository databases that source the sequencing datasets, this is discussed further in chapter 4.

An example of such systematic errors that have been characterised and modelled are base call errors. Such errors are a common feature of sequencing technologies and Next-generation sequencing (Meacham et al., 2011). Although these technologies have significantly reduced the costs and increased throughput, they have been shown to be more error prone than preceding technologies. Some of the systematic bias observed, as Roberts et al. (2011) point out, can be categorised as - (i) Positional biases - whereby fragments (reads) align preferentially to the start or end of the underlying transcript (ii) Sequence specific biases - where the likelihood of a read fragment being aligned to the underlying transcript depends on the sequence of nucleotides flanking the read. These can contribute to coverage bias, that is deviations in the expected distribution of read fragments to a reference genome. In RNA-Seq, Lahens et al. (2014) demonstrated that coverage bias has been found to be the result of the application of routinely used mRNA selection protocols, such as poly-A and ribosomal depletion, both of which resulted in significant fold changes in the coverage as compared to sequencing the same library but without mRNA selection. The data sets from this paper will be used extensively in chapter 6.

Similarly, Hansen et al investigated biases in RNA-Seq data resulting from random hexamer priming; a method used in library preparation of dsDNA samples from RNA to be sequenced on the Illumina Genome Analyser (Hansen et al., 2010). Their work demonstrated that random hexamer priming results in a non-uniform distribution of reads. This occurs because of the positional influence on nucleotide frequencies in nucleotides up to the thirteenth nucleotide from the 5' end of the reads. This positional influence was reproduced across experiments, indicating that there is a consistent bias occurring. An outcome of their work is a bias count reweighing scheme they have developed to mitigate the impact of the biases.

With a view on demonstrating potential impact on biological inferences, by focusing on the Illumina platform Meacham *et al.* characterised systematic errors (positional and sequence specific) that could be misinterpreted as heterozygous sites in individuals and SNPs in population analysis. They found that the majority of systematic errors were

sequences preceded by a G and the most common being GGT where a T is substituted for a G. It has also been demonstrated that wrong base calls are frequently preceded by base G indicating a difficulty in the Illumina base-caller software in differentiating between GGG and GGT (Dohm et al., 2008).

Evidently, systematic bias can be platform specific. A number of studies have compared different sequencing platforms in order to investigate bias originating in them. Harismendy et al. (2009) compared Roche 454, Illumina GA, and the ABI SOLiD platforms by sequencing the same 260 kb in four individuals using long-established capillary electrophoresis, employed by the ABI 3730xL platform, as a gold standard. They found that the coefficient of correlation (r) of per-base sequence coverage depth, between replicate samples sequenced on the same platform, were 0.62, 0.90, and 0.88 on Roche 454, Illumina GA, and ABI SOLiD, respectively. However, they found that the same samples sequenced across different platforms showed poor correlation (r <0.19). In another, very similar study, Roche Genome Sequencer FLX System, Illumina Genome Analyzer, and Applied Biosystems SOLiD system were compared (Suzuki et al., 2011). This used samples from *E. coli* DH1 strain, which were sequenced on each platform then aligned to the reference genome. The study found that, sequencing fidelity was lowest in Illumina GA, whilst the ABI SOLiD sequencing platform suffered from the largest number of un-mappable reads due to sequencing errors - about half of the reads could not be aligned to the reference genome. Table 3.1 summarises the bias and error profiles of some of the major sequencing platforms.

Table 3.1: Summary of bias specific to particular NGS platforms

Bias originating from different NGS platforms	
Platform	**Bias / Error profile**
454 / Pyrosequencing	As a result of the pyrosequencing chemistry and base-calling measurement applied on this platform, difficulties arise in correctly identifying the length of homopolymer runs, that is runs of the same nucleotide (Margulies et al., 2005; Quince et al., 2009; Balzer et al., 2011). Additionally, replication bias has been identified which results in multiple reads for a unique DNA fragment occurring, in a random manner, within a run (Mariette et al., 2011). Niu et al. (2010) found such duplicates can account for up to 44% of the total reads of a run, and point out that, as these include natural duplicates, the removal of those duplicates arising from the pyrosequencing process itself is challenging.

ABI SOLiD	Lower numbers of mappable reads (though Illumina GA was similar) as compared to and Roche platform, with only 35% of reads passing quality filters. However, after filtering although 96% of reads map to the reference, this means only 34% of the SOLiD reads (43% for Illumina) are usable, and this is also an inefficiency (Suzuki et al., 2011).
Illumina	**Genoma Analyser:** In their comparison study, Harismendy et al. (2009) found that regions of low coverage were AT-rich repetitive sequences, which is consistent with other studies (Dohm et al., 2008; Hillier et al., 2008). Overrepresentation of amplicon ends in the DNA samples, after fragmentation but prior to library generation, leading to extreme sequence coverage bias, has been noted by Harismendy et al. (2009). Futhermore, wrong base calls made by the Illumina base-caller software are frequently preceded by base G Dohm et al. (2008); Nakamura et al. (2011). **Genome Analyser IIx, HiSeq and MiSeq:** Additionally, amplicon sequencing using these instruments has been found result in erroneous reads. These can contribute to the overestimation of diversity in ecological studies (Bokulich et al., 2013).
MinION / Nanosequencing	This technique, which is applied on the Oxford Nanopore MinION device, suffers from high error-rates - between 5% and 40% error according to Goodwin et al. (2015), 38.2% according to Laver et al. (2015) who also found GC bias manifesting as underrepresentation of regions of high GC content (they analysed the device by sequencing three bacterial genomes of varying GC content).

Additionally, in order to study how quality and interpretability of results in RNA-Seq data is affected by technical variation, Raz *et al.* used multiple Human RNA samples to assess RNA fragmentation, RNA fractionation, cDNA synthesis, and single versus multiple tag counting. They used the Helicos^{TM} Single Molecule Sequencing (SMS) platform with 1st-strand cDNA-based methods (RNA-Seq) (Raz et al., 2011). In doing so their aim was to understand how the source of RNA affects transcript detection and how potential errors may be introduced in protocol steps and how these variations compound to impact conclusions drawn. They found that RNA fragmentation methods hampered detection of short RNAs, and that an incomplete view of the transcriptome results from PolyA RNA which excludes thousands of annotated and even more un-annotated transcripts. They claim a more complete view of the transcriptome is achieved at lower cost with Ribosomal-depleted RNA methods and that single tag counting was advantageous to expression measurements and detecting short RNAs relative to multi-read protocols. As they state in their paper, each of the preparatory methods has its own advantages and disadvantages and their work hopes to enable researchers to choose optimal preparatory protocols.

A relevant study on motifs that induce sequencing errors was undertaken by Allhoff et al (Allhoff et al., 2013) who described a statistical framework for identifying sequence specific errors caused as a result of preceding bases they term CSEs (context-specific errors). Their method involved pooling genomic positions and screening for strand biases a method they demonstrate to yield greater statistical power for identifying biases.

Finally Cheung et al (Cheung et al., 2011) studied ChIP data from Illumina's Genome Analyser and found three types of systematic sequencing errors caused by GC content, mappability of sequencing reads, and regional biases that might be generated by local structure and have devised a normalisation scheme that can be applied to downstream data analysis.

A thorough understanding of what protocols are applied prior to sequencing could provide much more subtle analyses to the ones applied above which vary according to the pipeline of protocol steps. Furthermore, it would enable best practice in terms of these protocols to come to the fore. In the following section we discuss the range of potential biases that occur at some of the steps that occur prior to sequencing and demonstrate that while a thorough study is absent, there is evidence of bias that comes from these steps.

3.6 NGS protocol steps prior to sequencing

The core work-flow processes that are shared by next generation sequencing technologies and involve protocol steps where biases may be introduced are shown in Figure 3.7. The protocol steps carried out before sequencing are classified into three classes, DNA/RNA fragmentation, DNA ligation and DNA enrichment that are described in detail, Furthermore we have summarised the sources, impact and possible means of ameliorating such biases in Table 3.2 at the end of this section. It should be noted that there are various protocols that do not require the fragmentation of DNA prior to ligation, such as PCR amplicon methods.

3.6.1 DNA/RNA fragmentation

Next-generation sequencing platforms require fragmentation of double-stranded DNA (dsDNA) into sequence fragments (fragment templates or mate-pair templates) of an appropriate size dictated by the read-length on the platform (Sambrook and Russell, 2006b). There are currently five methodologies in use for fractionating dsDNA: enzymatically (with restriction endonucleases), sonication (Sambrook and Russell, 2006b),

nebulization(Sambrook and Russell, 2006a), hydrodynamic shearing, and chemical shearing.

Bias in the fragmentation protocol step results in a large size distribution of fragment lengths. Enzymatic fragmentation employs type II restriction endonucleases to cleave dsDNA at (or at close proximity to) short (3-8bp) recognition sequence sites (Orlowski and Bujnicki, 2008) but is known to introduce bias due to factors that might impair the activity of sequence site recognition that the technique relies upon. Kamps-Hughes et al Kamps-Hughes et al. (2013) utilised Illumina high-throughput sequencing to assay the enzymatic activity of type II restriction endonucleases. They examined cognate site cleavage and non-specific, non-cognate site cleavage (referred to as star activity) of restriction endonucleases (EcoRI and MefI) by mapping millions of site-flanked reads back to the E. coli and D. melanogaster genomes. Their study demonstrated that despite the high sequence specificity they exhibit, this characteristic is dependent on a number of factors such as enzyme concentration, sequence context, buffer concentration and nucleotides flanking the cleavage site.

Fractionation by sonication is a method in which the dsDNA is subjected to short periods of sonication that generate fragmented DNA as a result of hydrodynamic shearing stresses (Sambrook and Russell, 2006b). Chromatin complexes of DNA and proteins have been shown to be refractory to shearing by sonication and this results in under-representation of the sequences affected (Keohavong and Thilly, 1989; Grokhovsky, 2006).

Fragmentation by nebulisation forces the DNA solution through a small hole producing a fine mist of aerosol droplets containing the fractionated dsDNA. The fragment size is determined by the viscosity of the DNA solution, speed at which the solution is ejected, pressure of the gas and temperature (Sambrook and Russell, 2006a).

Hydrodynamic shearing is a method of DNA fragmentation that involves injecting the sample solution through a narrow diameter orifice at high velocity. The resulting shearing stresses on the DNA strands cause them to break, resulting in an approximately normally distributed fragment size. Swartz and Farman investigated the effect of hydrodynamic shearing on the sequencing of telomere-associated sequences (Schwartz and Farman, 2010). They state that searches for telomeric sequences in fungal genomic databases typically do not yield many results and have found hydrodynamic shearing to be a cause of this. They found that sub-terminal regions of DNA are resistant to shearing, with breakages only occurring at the next cleavable location in relation to the terminal end of the fragments. This results in an overrepresentation of terminal fragments. Telomeric regions, however, are underrepresented because as all terminal fragments are cleaved at a similar location there exist no contigs that connect

the terminal and sub-terminal sequences.

Chemical shearing utilises heat and a divalent metal cation (usually magnesium or zinc), that is a metal ion with a valency of 2 i.e. a positive charge of 2+ to cleave the DNA / RNA. This is possible because the phosphate groups that form the backbone linking the nucleotides in RNA and DNA carry a negative charge. The length of the resulting sheared fragments can be controlled by the incubation time (Illumina, 2016).

3.6.2 DNA ligation

Blunting Unwanted 5' and 3' overhangs are removed from the double stranded ds-DNA to facilitate the ligation of platform specific synthetic DNA sequence adapters to the fragments - a process termed blunting. A number of enzymes can be utilised for this purpose such as (Klenow DNA Polymerase, T4 DNA polymerase and Mung Bean Nuclease). The enzyme is used to repair the ends of the dsDNA fragments by ensuring that the ends of the complementary strands are in line with each other. Such polymerase enzymes possess 5' $\rightarrow$ 3' polymerisation activity and 3' $\rightarrow$ 5' exonuclease activity but lack 5' $\rightarrow$ 3' exonuclease activity. The effect is that 3' overhangs on ssDNA fragments are removed by the 3' $\rightarrow$ 5' exonuclease activity. As 5' $\rightarrow$ 3' exonuclease activity is lacking in these enzymes 5' overhangs remain intact and any complementary 3' receding strand is extended and brought in-line with the 5' overhung strand by the enzymes 5' $\rightarrow$ 3' polymerase activity. This ensures both ends are blunted, namely there isn't a single stranded DNA overhang. The fidelity of polymerase enzymes used in this step is variable and Klenow polymerase has been shown to have an average error rate (mutations per base replicated) of 1.3 x10^{-4} (Keohavong and Thilly, 1989).

Phosphorylation It is necessary to phosphorylate the 5' ends of the blunted fragments as polymerase activity occurs in the 5' $\rightarrow$ 3' direction and can be carried out enzymatically using T4-PNK (Polynucleotide Kinase) which catalyses the transfer of the -phosphate of ATP to the 5' hydroxyl end. The efficiency of T4-PNK varies depending on the 5 nucleotide and this can manifest itself as bias if a proportion of fragment ends remain un-phosphorylated. Differences in the binding interactions between T4-PNK and the kinase substrate result in T4-PNK exhibiting bias in the preference of certain nucleotides at the first and second sequence positions of the substrate resulting in a greater activity in the phosphorylation of 5 G than for 5 C (Eastberg et al., 2004).

Attachment of A overhang to dsDNA 3' ends or blunt-ended ligation Synthetic sequencing adapters (such as those used on the Illumina platform) normally possess a 5' T-overhang to facilitate their ligation to the fragments to be sequenced. It follows that molecules in the sequencing fragment library must possess a complementary 3'A overhang; a genetically modified Klenow exo-minus is usually used to achieve this

(Sanger, 2013). The enzyme possesses no exonuclease activity, but retains 5' → 3' polymerase activity. This is used to catalyse the attachment of A overhang to the 3' end of the sequencing fragments so as to complement 5' T-overhang on the platform specific adapters. Alternatively blunt-ended ligation can also be used to ligate sequence adapter with sequencing fragments.

Library preparation methods that utilise DNA ligase enzyme ligate dsDNA fragments with 5' A' overhang to synthetic sequence adapters with 5' T' overhang have been shown to be biased against sequences starting with T as opposed to blunt-ended ligation (Seguin-Orlando et al., 2013).

In their study on the impacts of Illumina sequencing bias and its implications on characterising ancient DNA (Seguin-Orlando et al., 2013) sequenced modern DNA in parallel with different ligation strategies, and in order to eliminate shearing as a source of the bias sheared samples using different methods for the same ligation strategy. Their results show that the bias against sequence fragments with 5' T was not likely due to shearing method but is a result of the 3' A to 5' T overhang ligation method (which is a primary method on the Illumina platforms) and correlates inversely with the concentration of sequence adapters (which is normally kept low so as to prevent hybridisation of the adapters with each other). They explain how the post-mortem degradation of ancient DNA (resulting in C deamination to U) generates misincorporation patterns in the sequencing library that can be used to recognise and characterise true ancient DNA and that these patterns can be altered during certain library construction protocol methods and they cite the Taq and Phusion polymerase enzymes that are integral to Illumina sequencing protocols as a cause of this undesired modification during library preparation.

Adapter Ligation The ligation of synthetic dsDNA sequencing adapters (with 5' T overhang) to the fragment dsDNA (5' phosphorylated, with 3' A-overhang) again requires the use of DNA Ligase enzyme (potential biases discussed above) and is added in excess (concentration of 10:1) so as to ensure the attachment of as many adapters as possible in unit time. Housby et al (Housby, 1998) point out that most studies of the DNA replication process have centred around the fidelity of DNA polymerase and the importance of understanding all the mechanisms which ensure faithful copying of DNA sequence at replication.

3.6.3 DNA enrichment

In order to achieve quantities of the sequence samples sufficient for sequencing an enrichment process must be applied to the adapter-ligated fragment library. PCR (Polymerase Chain reaction) is a mainstay method in DNA enrichment (Kozarewa

et al., 2009) and is useful for enriching the fragment library as it replicates only those fragments to which an adapter is attached as the adapter encapsulates a PCR primer binding region. Those fragments not ligated to adapters will not be replicated by virtue of lacking the PCR primer site (which is located on the ligated adapter). PCR amplification, however, may introduce bias in the form of non-uniform distribution of reads and can introduce artefacts into the library prepared for sequencing (Acinas et al., 2005). The significant variation in fidelity of polymerase enzymes that PCR is dependent on has long been established Keohavong and Thilly (1989). There are a number of origins for such artefacts: re-arrangement of the DNA resulting in Chimera formation, formation of hetero-duplex molecules and DNA polymerase errors further discussion of them is beyond the scope of this article, the reader is directed a study by Acinas et al who looked at PCR-induced artefacts in sequencing library construction.

It is long-established that PCR is impaired by GC-bias in the fragments to be enriched (Kozarewa et al., 2009; Acinas et al., 2005; Chen et al., 2013a). Kozarewa et al demonstrated that using a PCR amplification free enrichment step (relying solely on cluster amplification on the sequencing platform for the enrichment of the library) resulted in more uniform distribution of reads. If PCR is employed, one approach to reducing amplification noise is the use of Unique Molecular Identifiers/Indexes (UMIs), that is molecular labels (random oligonucleotides) that distinctly identify each nucleic acid molecule in the sample. The method allows for the differentiation between those molecules that arise uniquely in the sample from those that are duplicated by the enzymatic PCR process. This information can then be used to reduce PCR amplification bias in the data (Islam et al., 2014).

Given the biases in polymerase activity (Spitaleri et al., 2004; Keohavong and Thilly, 1989) a number of commercially produced genetically-modified polymerase enzymes have been developed to confer greater fidelity. An investigation by Quail et al (Quail et al., 2012) compared the fidelity of two commercially available polymerase enzymes (Kappa HiFi and Phusion polymerase) against PCR-free sequencing of four genomes of varying GC-Content. They demonstrated variation between Kappa-Hifi and Phusion polymerases and found the profile of Kappa-Hifi (as depicted by plots of normalised sequencing depth vs %coverage) to be closer to the profile of no amplification as compared with Phusion polymerase.

DNA damage has also been shown to influence nucleotide incorporation and can introduce bias dependent on the preferences of the polymerase catalysing the reaction. Investigation into nucleotide incorporation preferences of different polymerases can be achieved using modified nucleotide 8-Oxo-7,8-dihydro-2-deoxyguanosine (8-oxodG) which can exhibit both an anti-conformation (allowing normal Watson-Crick base pair-

ing) or syn-conformation (which partakes in Hoogsteen bonding) (Sikorsky et al., 2007). Sikorsky et al investigated this and describe how the ratio of dCMP to dAMP insertion corresponding to 8-oxodG is dependent on the class of polymerase and found Taq DNA polymerase fidelity and amplification efficiency are susceptible to lesions on the fragment to be enriched.

Other enrichment methodologies can also be a source of bias; MDA (multiple strand displacement amplification) can result in preferential amplification of certain sequences (Jiao et al., 2011).

3.6.4 Targeted sequencing

A group of extremely useful transcriptomics methods have emerged - known as targeted sequencing - which allow researchers to focus high-throughput sequencing efforts on a specific subset of the transcriptome, by means of applying biotechnological selection methods (Bashiardes et al., 2005). These methods are generally cheaper than whole genome sequencing due to focused sequencing effort and they therefore also reduce the computational resources necessary to process the resulting data. Two techniques commonly employed for targeted sequencing are exome sequencing and amplicon sequencing. Exome sequencing has a number of advantages, importantly it facilitates targeted study of mutations and variants in protein-coding regions which is invaluable for disease studies and clinical diagnostic applications (Bamshad et al., 2011; ORoak et al., 2012; Rehm, 2013; Alazami et al., 2015).

In exome sequencing, there are two main target selection and enrichment strategies: (i) array-based capture (such as applied in Roche NimbleGen) and (ii) In-solution capture (such as applied in Agilent SureSelect). Array-based capture methods employ microarrays with ssDNA oligonucleotide probes to select regions of interest in the exome (Albert et al., 2007; Okou et al., 2007). The sample dsDNA molecules are first fragmented, to a size range suitable for the downstream sequencing platform used. The fragments are blunted and end-repaired with sequence adapters, and primed with universal priming sequences. A microarray is then used to capture the desired sequences which hybridise to it - those fragments which do not hybridise are washed away, whilst the remainder are eluted and enriched with PCR prior to sequencing. For in-Solution capture methods, the fragments are first fragmented, again to a size range suitable for the downstream sequencing platform used, blunted and end-repaired with sequence adapters. Capture of the desired sequences is achieved by adding biotinylated complementary sequences, known as "baits", to the solution (in SureSelect these are RNA) which hybridise to sample fragments containing the complentary sequence of interest (Gnirke et al., 2009; Mamanova et al., 2010). Next, streptavidin coated magnetic beads

- that bind to biotin in the baits which are hybridised to the complementary sequences - are subjected to strong magnets to filter the fragments of interest (that are now attached to the beads). The remaining un-hybridised fragments are washed away from the beads, and the baits digested. This yields the fragments containing the sequences of interest which are then sequenced. Amplicon sequencing utilises products of PCR, referred to as amplicons, encapsulating targeted regions, which are enriched by PCR and sequenced (Bybee et al., 2011). The customised amplicons are produced using two ssDNA oligonucleotide probes that hybridise to the genomic DNA, flanking either side of regions of interest. Extension of one of the oligonucleotide probes, by ligation, is performed such that it forms a continuous strand with the second to produce a strand that spans the region of interest. PCR is performed which, in addition to enriching the amplicons, adds unique molecular indexes and sequencing primers. Finally the amplicons are then sequenced.

With respect to bias in targeted sequencing methods, García-García et al. (2016) compared three DNA solution-capture enrichment methods for Next-generation sequencing, namely Illumina NRCCE and Agilent SureSelect against NimbleGen SeqCap EZ Choice (used as a reference). They examined the depth and uniformity of coverage, enrichment in targeted regions, performance in GC-rich regions, and the ability to generate consistent variant datasets. They found that NimbleGen offered the best depth of coverage and uniformity, while NRCCE showed the highest enrichment for target levels but also suffered from high duplicate rates. They observed that the SureSelectQXT showed an overall quality close to that of NimbleGen. The three methods compared were all found to produce suitable datasets for standard DNA variant discovery. Bias has also been identified in amplicon sequencing. Bokulich et al. (2013), investigated bias in amplicon sequencing using Illumina's GAIIx, HiSeq and MiSeq instruments, in particular for diversity estimation in microbial ecology. They found that erroneous reads contribute to the overestimation of microbial diversity, and that these are likely to be attributable to errors introduced in PCR during short-amplicon sample preparation, and the fact that these techniques are rapidly evolving, i.e. the employment of changing chemistries, for which the bias has not yet been characterised and the error rates remain unknown. They propose a *per-nucleotide read* filtering process, using the Illumina Phred score. Their method is specific to amplicon sequencing for diversity estimation, but can be applied consistently across datasets produced by all of the Illumina instruments they studied. Another study focused specifically on 16S rRNA amplicon sequencing demonstrated that library preparation protocols and the choice of primers are the most significant sources of bias and cause distinct error patterns (Schirmer et al., 2015). In testing different error correction strategies to mitigate these effects,

they also propose propose quality filtering of Illumina reads in amplicon sequencing. Specifically, they found that quality trimming of reads using Sickle (Nikhil Joshi, UC Davis, 2016), combined with error correction using BayesHammer (Nikolenko et al., 2013), followed by read overlapping using PANDAseq (Masella et al., 2012), was the most successful method which reduced substitution error rates on average by 93%.

Table 3.2: Summary of bias in NGS sample preparatory protocol steps. Protocol steps common to both DNA and RNA sequencing are shown in the first part of the table, whilst those particular to RNA sequencing are covered in the latter part of the table. The source(s), impact, and steps that may ameliorate them, and/or potential work that could improve the methods are listed. NB: Where other newer protocol methods are suggested in lieu, it is important to note that potential bias in these alternatives may yet be characterised.

Bias in the fragmentation step	
Protocol	**Enzymatic fragmentation with type II restriction endonucleases**
Source(s)	The protocol is dependent on factors such as enzyme concentration, sequence context, buffer concentration and nucleotides flanking the cleavage site. Different restriction endonucleases have varying preference for specific sequences flanking the cleavage site (paper cited studied 8 base flanking sequences). (Kamps-Hughes et al., 2013)
Impact	Sequence-specific deviations in the coverage of reads.
Amelioration	Use of a different fragmentation method; Quantitative correction of the representation of transcripts flanked by the affected sequences may theoretically be possible (i.e. using count-reweighing).
Protocol	**Shearing by sonication**
Source(s)	Occurs when shearing chromatin complexes of DNA and proteins. These have been shown to be refractory to shearing by sonication (Keohavong and Thilly, 1989; Grokhovsky, 2006).
Impact	Under-representation of the sequences affected.
Amelioration	Use of a different fragmentation method; Detailed data on the sequences affected (that flank cleavage sites) and the extent to which they cause bias in fragment distribution has been presented in Grokhovsky et al. (2011). This could potentially be used to model the effect and, if possible, apply quantitative correction to the sequenced data (Nechipurenko et al., 2014).
Protocol	**Hydrodynamic shearing**
Source(s)	Sub-terminal regions of DNA are resistant to shearing. This restricts breakages to only occurring at the next cleavable location in relation to the terminal end of the fragments.
Impact	Over-representation of terminal fragments. Under-representation of Telomeric regions as all terminal fragments are cleaved at a similar location there exist no contigs that connect the terminal and sub-terminal sequences. This is particularly a problem when applying the technique for sequencing species with shorter telomeric regions as important biology might be missed. Schwartz and Farman (2010).
Amelioration	Consideration of the technique's limitations (i.e in certain species, or studying telomeric regions); Use of a different fragmentation method;
Bias in fragment blunting, adapter ligation and phosphorylation	
Protocol	**Blunting with polymerases**
Source(s)	Typically the following enzymes: Klenow DNA Polymerase, T4 DNA polymerase and Mung Bean Nuclease. The fidelity of these enzymes is variable (Keohavong and Thilly, 1989).

Impact	Errors due to occurrences of mutations in replicated bases.
Amelioration	Selection of a suitable polymerase enzyme with a higher fidelity (where possible) - for instance, Kappa HiFi and Phusion polymerases which have higher fidelity than that of Taq polymerase (Quail et al., 2012), may improve results in blunting.
Protocol	**Phosphorylation with T4 PNK enzymes**
Source(s)	The efficiency of T4-PNK varies depending on the 5' nucleotide and bias occurs if a proportion of fragment ends remain un-phosphorylated.
Impact	Bias manifests in the preference of certain nucleotides at the first and second sequence positions of the substrate resulting in a greater activity in the phosphorylation of 5 G than for 5 C (Eastberg et al., 2004). NB: A different, structural effect is also seen in T4 RNA ligase (see RNA-Seq section below).
Amelioration	None discussed in the literature.
	Bias in enrichment
Protocol	**PCR**
Source(s)	PCR is impaired by GC-bias in the fragments to be enriched (Day et al., 1996; Kozarewa et al., 2009; Ross et al., 2013; Acinas et al., 2005; Chen et al., 2013a).
Impact	Genome coverage bias, manifesting as poor read coverage (underrepresentation) of sequences occurring in regions of GC extremes.
Amelioration	PCR-free amplification is the ideal - whilst WGS (Whole Genome Sequencing) is PCR-free this requires large amounts of input DNA which is a limiting factor for some sample types (Aird et al., 2011; Ross et al., 2013); Kozarewa et al demonstrated that using a PCR amplification free enrichment step (relying solely on cluster amplification on the sequencing platform for the enrichment of the library) resulted in more uniform distribution of reads. Another study by Aird et al investigated PCR reaction conditions and thermocyclers. They found the steepness (duration) of the thermocycling step, the length of the denaturation step and choice of polymerase enzyme along with the GC content of the sample species influences the amplification of sample sequences. They advocate optimising PCR protocol steps and selecting polymerase enzymes depending on the GC content of the species sequenced. They also found that their optimised protocol performs comparably to the PCR-free method outlined by Kozarewa et al (Aird et al., 2011; Kozarewa et al., 2009).
Protocol	**MDA**
Source(s)	MDA (multiple strand displacement amplification) can result in preferential amplification of certain sequences and structural artefacts (in particular chimera formation (Lasken and Stockwell, 2007)). MDA followed by mate-pair sequencing is typically used in cancer genomics applications for the analysis of mutation events (in particular translocations and inversions) across long sequence distances in tumor specimens with low tissue material. (Jiao et al., 2011).

Impact	As cancer genomics applications aim to study mutation, bias free methods are required. These are also highly dependent on the fidelity of the enzymes used in amplification. MDA has been shown to be sensitive to nucleotide runs (repeated bases) which results in nucleotide level mutations and is also known to introduces inversions when applied to prokaryotic genomes (Sjöblom et al., 2006; Lasken and Stockwell, 2007).
Amelioration	None discussed in the literature.
Bias occurring in Targeted sequencing	
Protocol	**Amplicon sequencing**
Source(s)	Bias can be introduced in PCR during short-amplicon sample preparation. Also, as targeted sequencing techniques are rapidly evolving, different chemistries may be applied for which the bias has not yet been characterised (Bokulich et al., 2013). Library preparation protocols, and the choice of primers used, were found to be the most significant sources of bias in 16S rRNA amplicon sequencing (Schirmer et al., 2015).
Impact	Nucleotide substitutions, often following certain motifs, that contribute to the error rate, which is particularly problematic in diversity studies - this has been associated with the Illumina platform. Coverage bias, in regions of GC extremes, as a result of PCR (discussed in above table entry (under bias in enrichment section)).
Amelioration	For 16S rRNA amplicon sequencing (Schirmer et al., 2015), on the Illumuna platform, Schirmer et al. (2015) suggest quality trimming of reads using Sickle (Nikhil Joshi, UC Davis, 2016), combined with error correction using BayesHammer (Nikolenko et al., 2013), followed by read overlapping using PANDAseq (Masella et al., 2012). For diversity estimation applications on Illumina's GAIIx, HiSeq and MiSeq instruments, Bokulich et al. (2013) propose an "informed" *per-nucleotide read* filtering process using the Illumina Phred score, which is detailed in their paper.
Bias occurring in RNA sequencing sample preparation	
Protocol	**RNA-ligase adapter ligation**
Source(s)	Structural bias in T4 RNA ligase-mediated 3-adapter ligation (Zhuang et al., 2012)
Impact	Over-representation of mRNAs with specific sequence features or secondary structures that are preferentially ligated by T4 RNA ligase.
Amelioration	In applying RNA-Seq to the study of miRNAs, Sorefen et al suggest the use of a refined in which modified Illumina adapters containing an additional degenerate, randonly generated sequence aer used over standard Illumina adapters. (Sorefan et al., 2012)
Protocol	**Poly-A selection / Oligo(dT) priming during cDNA synthesis from mRNA**

Source(s)	Oligo(dT) priming of mRNA is facilitated by the 3' poly-A sequence present on the mRNA molecule which is primed to initiate reverse transcription of the mRNA. When poly-A sequences occur internally (the study cited examined 8-14 base repeats) within an mRNA (i.e. in positions other than the 3' end) these can also be unintentionally primed in addition to that of the 3' end poly-A sequence. This situation results in two truncated cDNA fragments instead of a full length cDNA transcript: one will have been reverse-transcribed from the desired 3' poly-A sequence on the mRNA but terminates at the first internal occurrence of the poly-A sequence, and the other will be primed from the undesired poly-A sequence up until the end of the mRNA (Nam et al., 2002).
Impact	Not only is this protocol used RNA-Seq, it has been mainstay in methods that were precursors to RNA-Seq, such as Massively parallel signature sequencing (MPSS) and the older Expressed Sequence Tag (EST) approach to quantifying mRNA transcript expression (Reinartz et al., 2002; Saha et al., 2002; Adams et al., 1991). Internal poly(A) priming results in multiple ESTs from the same gene - The EST technique was used extensively to annotate the human genome.
Amelioration	Nam et al. (2002) propose the use of anchored Oligo-dT primers as they found that the proportion of reverse transcribed dDNAs correctly primed from the poly-A sequences at the 3' end of the mRNAs were further enriched. This method is now routinely applied; Zhang et al. (2012) describe a method for producing strand-specific libraries for RNA-Seq that have been prepared without poly(A) selection.
Protocol	**Random hexamer priming of RNA sample**
Source(s)	The procedure results in positional influence on nucleotide frequencies in nucleotides up to the thirteenth from the 5' end of the reads. (Hansen et al., 2010).
Impact	Non-uniform distribution of reads; predictable nucleotide frequencies.
Amelioration	Use of a different priming method; Hansen et al. (2010), who reported the effect, propose a read count re-weighing scheme. Weights are calculated from the hexamer occurring at the start of the reads, this weight is then used correct the biased distribution by re-weighing the counts of the read.

3.7 Methods and models to characterise and mitigate biases in RNA-Seq data

As discussed in section 3.3.2, RNA-Seq is the newest method utilising Next-generation sequencing, to quantify the abundance of RNA transcripts in a transcriptome, to study molecular events at a higher fidelity and with greater range than Microarrays (Mutz et al., 2013; Kukurba and Montgomery, 2015). Microarray technologies suffer from unwanted sources of bias, in particular the hybridisation of probes is known to lack specificity leading to increased variability. These sources are well characterised and estimates of gene expression have, therefore, been amenable to improvements by the application of statistical techniques. RNA-Seq technology, however, is comparatively younger and sources of bias and variability are still under investigation. The focus of this thesis is on the application of distributed computing to the analysis of biological datasets, and as we will apply MapReduce to sequence data generated by RNA-Seq in chapter 6, we will briefly introduce some methods and models that have been proposed to characterise and mitigate bias in RNA-Seq data.

The main obstacle to obtaining accurate estimates of transcript expression from RNA-Seq data is non-uniformity in the distribution of mapped reads to the reference genome, which reduces the certainty that the measured counts of mapped reads reflects the true expression of the transcript within the cell's transcriptome. As discussed in this chapter, these bias have numerous sources such as, for example *wet-lab* sample preparatory techniques, the sequencing process itself (Dohm et al., 2008) and the potential for errors in post-sequencing data processing. They perturb the uniformity of the distribution of mapped reads to a reference genome (Lahens et al., 2014) and such bias manifests itself as sequence-specific or positional (Roberts et al., 2011). For the *positional* biases due to random hexamer priming in sample preparation identified by (Hansen et al., 2010) and discussed in section 3.3.2, they postulate a model in which RNA-Seq data primed by random hexamers contains 13-mers which are under- and over-represented across the transcripts, and not just the 5' end of the reads. They propose a read count re-weighing scheme in which each transcript is assigned a weight based on the hexamer occurring at the first position of its reads. This weight is then used to re-weigh the counts of the read based on the observation that counts of the given hexamer have a biased distribution at the start of the reads, whilst the distribution at the end of the reads appears unbiased. They point out that trimming the leading 5' end of the reads of the hexamer does not improve the bias in nucleotide frequencies. Interestingly, they state that in trimming the 3' end, however, which by virtue of the transcript length has less reliable read quality, they observed an improvement in the

uniformity of mapped reads, confirming this is not an artefact of sequencing. We shall investigate this in chapter 6.

An extremely useful method for characterising the bias introduced in RNA-Seq library preparation is proposed by Lahens et al. (2014). They have utilised IVT (In Vitro Transcription) in *E. coli* to clone of a pool of approximately 1000 pre selected human plasmids from the Mammalian Gene Collection (MGC) (Temple et al., 2009). Because the sequences and expression levels of these plasmids is known, and they do not undergo splicing, this allowed them to generate a highly controlled samples, and therefore a controlled data set, in which the source of biological variation in the samples is minimised. They then subjected these samples to different RNA-Seq preparatory protocols, specifically varying the step in which mRNA is selected, enabling study and quantification of the effect of these steps on coverage levels of the MGC transcripts when they were aligned to the human reference genome (hg19/grch37). They found that the mRNA selection methods employed in RNA-Seq protocols, poly-A and ribosomal depletion, both resulted in significant fold changes in the coverage of the IVT MGC plasmids when compared to sequencing the IVT MGC plasmids directly (without mRNA selection). Because the bias introduced in this dataset is well characterised and attributed we will examine some samples from this dataset in chapter 6.

An important source of bias in both RNA-Seq and microarray data is that of extremes of GC-content (Risso et al., 2011; Benjamini and Speed, 2012), and this has been further characterised and incorporated into models proposed by various researchers in order to apply corrections to RNA-Seq expression estimates (Wu et al., 2011). For instance, Roberts et al. (2011) in their detailed study that compared normalisation methods in both Microarray and RNA-Seq data, found that after applying expression corrections, high log-fold change was found to correlate with high GC-content, but that this occurred in some species and not others (observed in the Human dataset they used but not in yeast). The outcome of their work, which seeks to improve estimates by their existing gene expression analysis tool Cufflinks (Trapnell et al., 2012), is a correction method which utilises weights computed from the ratio of FPKM normalised nucleotide frequencies to the background to re-weigh transcripts on a nucleotide-by-nucleotide basis. Additionally, in another research paper by the authors of the investigation into the hexamer priming effect, proposed a method to overcome GC-content effects involves using regression to remove systematic bias and quantile normalization to correct global distortions (Hansen et al., 2012). Reviews of commonly used normalisation methods applicable to RNA-Seq data can be found in research papers by (Li et al., 2015) and Dillies et al. (2013).

As GC-content effects are an important source of bias in RNA-Seq data, incorpora-

tion of terms for GC bias in bias models is therefore necessary (Benjamini and Speed, 2012), especially given the observations that GC-content bias can vary between samples from the same species, and has been demonstrated to be a batch effect (Pickrell et al., 2010; Hansen et al., 2012). To this end, Love et al. (2016) show that including fragment GC-content and GC-stretches as bias parameters achieved a four-fold decrease in false positives at a FDR (False Discovery Rate) threshold of $a = 0.01$ (FDR and statistical corrections for multiple comparisons are discussed in chapter 6, section 6.3.5).

It has also been reported that a type of *sequence-specific* bias has been observed in RNA-Seq data that involves dinucleotides (*2-mers*) that are related to GC-content (i.e. AT, TA, GG, GC, CG, CC), specifically that a strong linear relationship exists between expression level and these aforementioned dinucleotides (Zheng et al., 2011). The authors propose mitigation of these effects on expression estimates by incorporating dinucleotide frequencies as parameters in addition to those for GC content and transcript length.

As discussed earlier on in this chapter, in section 3.3.2, the estimation of transcript abundance is crucial in RNA-Seq expression studies (Soneson and Delorenzi, 2013), and this presents a number of challenges (Garber et al., 2011) - for example, multi-mapped reads, as outlined in Ji et al. (2011); Feng et al. (2015a), where reads map to many locations but are only recorded in the above count data in one location, represent a source of bias, as are sequence-specific, positional (Roberts et al., 2011) and GC content biases (Zheng et al., 2011; Risso et al., 2011). The problem is further compounded by the increasing rate at which data is being generated by high-throughput techniques employed in RNA-Seq technologies. As a result, a number of software applications have been developed to address these challenges. For example, for increasing the speed of transcript abundance, Sailfish (Patro et al., 2014) employs a k-mer approach, that can accurately estimate transcript coverage using the counts of the k-mers, and which bypasses the computationally complex process of sequence alignment. Sailfish also applies bias correction based on transcript length, GC content and dinucleotide frequencies - potential bias factors investigated and suggested by (Zheng et al., 2011). Kallisto (Bray et al., 2016) is another software tool for RNA-Seq estimation which is able to analyse 30 million unaligned paired-end RNA-Seq reads in less than 10 minutes on a standard laptop computer (as of 2016). Building on these approaches, Salmon (Patro et al., 2017), also developed by one of the authors of Sailfish, implements feature-rich bias models, based on the findings in the literature to correct estimates for certain common biases, such as sequence-specific bias at 5' and 3' ends of reads, position-specific bias in coverage of reads, and fragment GC-content. It has two modes of operation: *quasi-mapping* mode - raw, unaligned reads are provided (in FASTA/Q format), indexes are built for

the reference genome, and estimation performed without aligning to the reference, but instead using the index of *k-mers*, and *aligned* mode - estimation is performed using pre-aligned reads (in SAM/BAM format). By utilising the indexed k-mers, the *quasi-mapping* mode is able to make substantial savings on memory and hard disk space by circumventing the need to generate intermediate files and thereby avoid I/O associated with reading large alignment files. These tools provide an efficient means of quantifying RNA-Seq transcript abundance, and because they apply corrections to the estimates by using methods informed by the literature, it is likely that they will be updated, or superseded, by new tools so as to keep up with future research on RNA-Seq biases that have yet to be characterised, or to implement new, relevant statistical techniques.

In this chapter we have thoroughly investigated and discussed in detail the sources of bias in Next-generation sequencing data with a focus on those that result in deviations from uniform read distribution in RNA-Seq data. We have seen that the sequencing of nucleic acid samples is dependent on the application of a sequence of complex preparatory protocol steps on the sample. Detailed documentation of the protocols applied is therefore necessary to understand and characterise the different types of bias present in sequencing data and also to investigate others that may yet be presently unidentified. Such information, given the diverse array of applications to which RNA-Seq can be applied, is also necessary to apply remedial methods such as correction of expression estimates using the bias characterising models we have introduced (Christelle and Watson, 2015) and, though our focus is on RNA-Seq data, this is also true of microarray data (Trevino et al., 2007). In chapter 4 we will therefore investigate the level of detail in the annotation of sequencing experiments deposited in the SRA, a large public repository of sequencing data that contains more than 29,000 at the time of publishing our investigation (Alnasir and Shanahan, 2015a). In particular we will use data mining techniques applied to the metadata of the SRA to examine the annotation of three key preparatory protocol steps that are known to introduce bias, namely: nucleic acid *fragmentation* known to cause biased distribution of sequence fragment lengths (Keohavong and Thilly, 1989; Kamps-Hughes et al., 2013), *enrichment* in which PCR (Polymerase chain reaction) amplification techniques introduce GC-content bias (Kozarewa et al., 2009; Acinas et al., 2005; Chen et al., 2013a), and *adapter ligation* which can suffer from the preferential selection of certain nucleotides flanking the sequencing fragments (Eastberg et al., 2004; Seguin-Orlando et al., 2013).

The above-mentioned *positional* and *sequence-specific* biases and their impact on the uniformity of distribution of mapped reads to a reference genome confound expression estimates in RNA-Seq experiments. In Chapter 6 we therefore propose a simple method to quantify deviations in the distribution of reads by examining *intra-exon*

k-mer motif correlations (correlations of motifs within the same exon). Considering the volume of data currently being produced by High-throughput Next-generation sequencing technologies (Stephens et al., 2015; Ward et al., 2013), and research projects such as the comprehensive assessment of RNA-Seq accuracy and reproducibility by the Sequencing Quality Consortium (SEQC) which analysed over 100 billion reads (10Tb) (SEQC/MAQC-III Consortium and others, 2014), we implement our method using distributed technologies, in particular MapReduce which we have introduced in chapter 2. Furthermore in chapter 6 we will apply the technique to the analysis of the transcriptomes of two *D. melanogaster* (Fruit fly) datasets, as well as *H. sapiens* samples produced by IVT using different RNA-Seq protocols, using the Apache Spark cluster computing platform.

Chapter 4

Experiment annotation - Limitations of NGS metadata

"Data do not give up their secrets easily. They must be tortured to confess."
– Jeff Hopper, Bell Labs

The reproducibility of experiments and the means to quantify biases in sequencing data are key features that determine the quality of a next-generation sequencing study (Nekrutenko and Taylor, 2012). In order to ensure a study possesses these features and meets the requirements for reproducibility and quantification of bias, sufficient descriptive information must be stored about the experiment and the data generated - this is termed metadata. Metadata is defined as "data that describes and provides information about other data." (Borgman, 2015). As discussed in chapter 3, the production of sequencing data in a typical next-generation sequencing workflow from sample preparation to sequencing is complex. In addition to the preparatory protocol steps, there are also many other processes, procedures and methods for which metadata must be captured in order to adequately represent the study experiment, a process known as annotation. For example information about the experimental design, samples, variables such as dose (in a dose response experiment), equipment used, protocol methods employed and the raw data generated, without which it would be impossible to reproduce experiments or quantify bias introduced in these processes. Therefore, the lack of such information in scientific studies of this kind would mean that conclusions derived from their experiments are likely to be unreliable (Raz et al., 2011).

It is clear that archival of experimental data, in a well-organised fashion so that it may be conveniently retrieved, is paramount. In this chapter we will introduce guidelines for metadata annotation of next-generation sequencing and microarray experiments which define *what* should be captured in the metadata and *how*. We will

explore the level and quality of annotation of the preparatory protocols in publicly deposited data. In particular, we focus on the SRA, one of the main repositories of next-generation sequencing data (Alnasir and Shanahan, 2015a). We will describe in detail the SRA metadata schema and the relevant fields for this study. Having constructed a list of relevant keywords for the above preparatory steps described in chapter 3 (section 3.4 onwards), we will present their abundance in the SRA metadata, and highlight the non-standard methods used in describing the annotation. Finally we will discuss the implications of the low level of coverage of these keywords, the overall structure of the metadata in the SRA, and the generally poor state of annotation in this regard.

4.1 Sequencing metadata

As we have outlined in chapter 3 (c.f. Figure 3.4) the workflow for the production of high-throughput sequencing data from nucleic acid samples is complex. There are a series of protocol steps to be followed in the preparation of samples for next-generation sequencing (Shendure and Ji, 2008). As noted previously, possible sources of bias in the preparatory protocol steps are *DNA fractionation* (Orlowski and Bujnicki, 2008; Kamps-Hughes et al., 2013; Keohavong and Thilly, 1989; Schwartz and Farman, 2010), *blunting*, *phosphorylation* (Eastberg et al., 2004), *adapter ligation* (Seguin-Orlando et al., 2013; Housby, 1998), and *library enrichment* (Kozarewa et al., 2009; Acinas et al., 2005; Chen et al., 2013a; Spitaleri et al., 2004; Sikorsky et al., 2007; Jiao et al., 2011).

We have described the above-mentioned sequencing work-flow steps and these are described in detail in chapter 3 (section 3.4 onwards). Whilst we have focused on sequence read DNA data and the relevant metadata, as noted by (Shendure and Ji, 2008) Next-generation technologies share the same complex preparatory procedures. Additionally, RNA sequencing data has also been demonstrated to be prone to biases in RNA-specific protocol steps (Raz et al., 2011), for example random hexamer priming (Hansen et al., 2010), and therefore, the adequate and accurate capture of metadata for RNA-Seq studies is equally important. Sufficient metadata for Next-generation sequencing is especially important given the large amount of raw data that is being produced by these technologies (Stephens et al., 2015), and the number of studies relying on these methods - as of November 2016, PubMed listed 22,254 papers where the search terms *DNA-Seq, RNA-Seq*, and *next-generation sequencing* were found in the title or abstract of the paper (PubMed, 2016).

4.2 Guidelines for metadata annotation

As mentioned in the introduction of this thesis (chapter 1, section 1.2 onwards) the vast majority of the raw sequence read data are being deposited in public repositories such as the Sequence Read Archive (SRA), Gene Expression Omnibus (GEO), and ArrayExpress (Leinonen et al., 2011; Edgar, 2002; Brazma, 2003). There exist a number of different guidelines for annotating genomic and transcriptomic data sets. These include MIAME (Minimum Information for a Micro-array Experiment)(Brazma et al., 2001) and MINSEQE (Minimum Information for a Sequencing experiment)(Functional Genomics Data Society, 2012b) both devised by the Functional Genomics Data Society (FGED) which stipulate *what* information should be recorded. The MIAME specification describes the minimum information for a microarray experiment that is required to enable unambiguous interpretation of the experiment's results and which is sufficient to potentially reproduce the experiment. It consists of six core information requirements (Functional Genomics Data Society, 2012a) - see Table 4.1. The MINSEQE specification, however, was devised for high-throughput, next-generation sequencing and comprises five core information requirements. It has the same core objectives as the MIAME guidelines but in addition aims to maximise the value of high-throughput research by improving integration of multiple sequencing experiments (Functional Genomics Data Society, 2012b). Table 4.1 lists the core information requirements for both the MIAME and MINSEQE specifications.

Additionally, there are document markup formats such as MAGE-ML (Microarray Gene Expression Markup Language) and MAGE-TAB (a tab-delimited form of MAGE-ML) which stipulate *how* the information should be recorded in the metadata (Rayner et al., 2006). Ostensibly, the appropriate use of the above guidelines (notwithstanding the absence of an agreed vocabulary) should ensure that one can disentangle the effects of different biases.

4.3 The SRA (Sequence Read Archive)

The SRA is one of the primary repositories for high-throughput sequencing data (Leinonen et al., 2011). As of December 2013 there were 29,598 studies in the database (according to our querying of the SRA database), this has risen to 84,193 as of September 2016[1]. The archive is synchronised periodically as part of the International Nucleotide Sequence Database Collaboration (INSDC), involving partners in the U.S. (NCBI), Europe (ENA) and Japan (DDBJ), and this allows data deposited to any site that is

[1]The metadata extraction timestamps indicating the date on which the SRA metadata was extracted were 2013-12-03 and 2016-09-21 respectively (Alnasir, 2015; Bioconductor project v2.14, 2013)

Specification requirements	
MIAME	**MINSEQE**
1. Raw hybridisation data	1. Description of the biological system, samples, and experimental variables being studied
2. Final processed (normalised) data for gene expression data matrices	2. Sequence read data for each assay. FASTQ format is recommended, with a description of the scale used for quality scores.
3. Sample annotation to include experimental factors (e.g., compound and dose in a dose response experiment)	3. Final (processed or summary) data for the set of assays in the study
4. Experimental design and sample-data relationships	4. Experimental design and sample-data relationships
5. Sufficient annotation of the array (e.g., gene identifiers, genomic coordinates, probe oligonucleotide sequences or reference to a commercial array catalogue number)	5. Essential laboratory and data-processing protocols employed.
6. Essential laboratory and data-processing protocols employed.	

Table 4.1: MIAME and MINSEQE core information requirements (Functional Genomics Data Society, 2012a).

part of the collaboration to be accessed via any of the others (Cochrane et al., 2010). In addition to the raw sequence data which comprise the bulk of the total data in the archive, metadata describing experimental parameters is also stored and accessible. A number of such parameters are recorded with depositions - for example the design of the experiment, details of species, cell lines, samples and identifiers for sequencing platforms and protocols (Nakazato and et al, 2013). As outlined below, there is a detailed database schema for conveying the metadata associated with each deposition. Given the large number of depositions and facilities for depositing metadata, the SRA is an excellent database to examine the range of different protocols.

4.3.1 SRA database schema

The schema for the Bioconductor SRAdb SQLite relational database (which is derived from and reflects the underlying NCBI SRA XML data) was utilised for our investigations and is shown in Figure 4.1 which focuses in particular on the metadata for the sequence read data (EMBL-EBI, 2013a). The SRA metadata is organised and stored in a relational database format across a number of tables: *Submission, Study, Experiment, Run, Sample and Analysis*. In the SRA database schema, a *Study* is the

top-level entity and may comprise of a single or multiple *Experiments*. Each *Experiment* record is associated with a single *Run* record and a single *Sample* record. The *Submission* record exists as a front-end container record used by the repository, and as noted in the SRA handbook (SRA, 2010), the *Submission* metadata record pertains to the submission "package" or "envelope" conveying the data to the archive. Each *Study*, *Experiment*, *Sample* and *Run* has its own *Submission* record. The fields in bold are those relevant to our investigations because they record metadata about the *Study* (e.g. a description of the sequencing study and abstract), *Sample* (e.g. origin and method of isolation) and *Experiment* (e.g. its design, construction of DNA/RNA sequencing library, protocols applied etc...) (EMBL-EBI, 2013b). Each of the fields are free-text. We note that this is a key point in our later analysis.

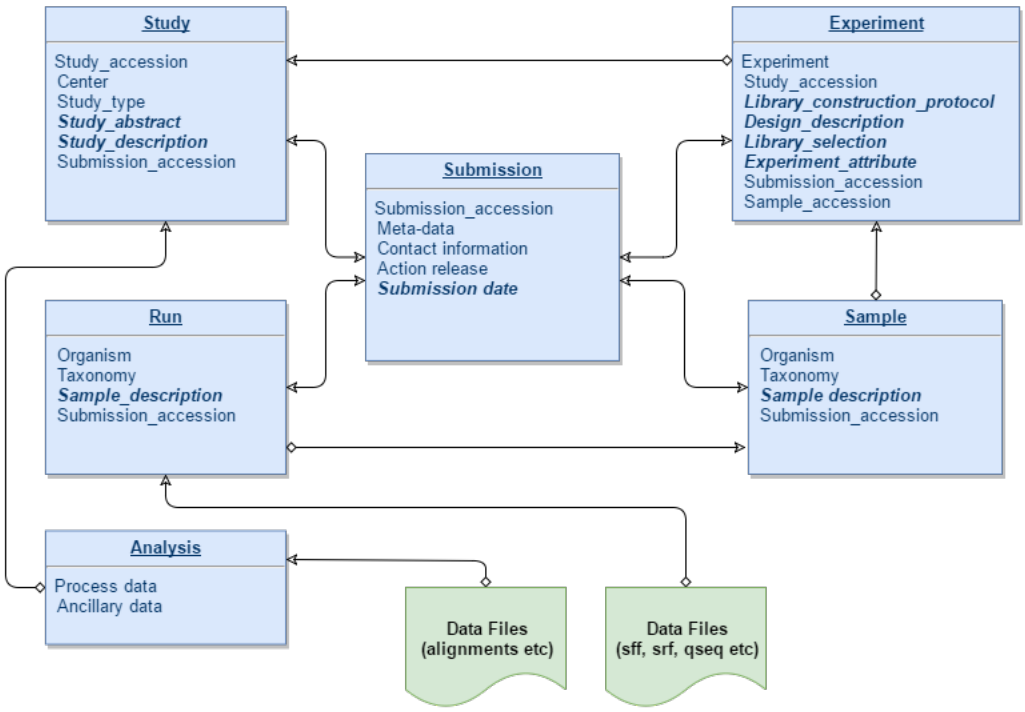


Figure 4.1: Schema diagram of the SRA relational SQLite database based on the SQL metadata. Field names in emphasis have been probed for protocol step annotation (Table 4.2) together with submission table date-stamp. Diamonds represent one-to-many relationships. Fields in bold emphasis are those with relevant experimental metadata.

The raw sequence data is stored under specific named directories described in the metadata. The *Study* table is the master table in this case. For each study entry, there are many *Experiment* entries and corresponding Sample and *Run* entries. Given that *Experiment, Sample and Run* tables have many-to-one relationships to each *Study* entry, we have aggregated them by *Study* entry.

Table	Field	Description
Study	*study_abstract*	Briefly describes the goals, purpose and scope of the study. This need not be listed if it can be inherited from a referenced publication.
Study	*study_description*	More extensive free-form description of the study.
Sample	*description*	Free-form text describing the sample, its origin and its method of isolation.
Experiment	*design_description*	More details about the set-up and goals of the experiment as supplied by the investigator.
Experiment	*library_selection*	Whether any method was used to select and/or enrich the material being sequenced.
Experiment	*library_construction_protocol*	Free-form text describing the protocol by which the sequencing library was constructed.
Experiment	*_attribute*	Properties and attributes of the experiment. These can be entered as free-form tag-value pairs.

Table 4.2: Relevant fields in metadata (obtained from col_desc_table found in SQLite DB) (Bioconductor project v2.14, 2013).

4.3.2 XML DTD and SRAdb SQLite differences

The metadata for the SRA public repository (Leinonen et al., 2011) for sequence read DNA-Seq data was acquired from the Bioconductor project in SQLite format (Bioconductor project v2.14, 2013)[2]. In order to ensure that the SRAdb is a good proxy for the underlying NCBI SRA XML data all fields from both the Bioconductor SRAdb SQL schema and NCBI SRA XML fields were extracted into separate text files for each SRA table. Each field of the NCBI XSD XML schema was then tested for its presence (or absence) in the corresponding Bioconductor SRAdb SQL table. Three fields were found in the XML data which are not mapped to SRAdb SQLite and may contain further protocol data, namely *Library_Descriptor, Sample_Attributes* and *Submission_Attribute*. However, on further inspection, these fields are either deprecated (*Library_Descriptor*), store only Biological sample data (*Sample_Attributes*) or carry data about the actual submission (*Submission_Attribute*).

4.3.3 A method for probing protocol annotation in the SRA

In the absence of greater structure in the fields, a structured word list relevant to the fragmentation, enrichment and adaptor-ligation protocol steps was constructed by examining, by hand, literature relating to protocol steps employed in Next-generation

[2]The metadata extraction timestamp indicating the date on which the SRA metadata was extracted was 2013-12-03 (Alnasir, 2015)

Protocol step structured word lists		
Fragmentation	**Adapter-ligation**	**Enrichment**
shear	adapter	clone%
restriction	ligat%	clonin%
digest	blunt%	vector%
fragment%	phosphoryl%	pcr
breaks	overhang	amplif%
acoustic	t4-pnk	polymerase
nebuli%	t4	taq
sonic%	pnk	phusion
	kinase	temperat%
	a-tail	thermal%
		anneal%
		denature%

Table 4.3: The % symbol denotes fuzzy-match logic, for instance `amplif%` will match "amplify" and "amplified".

sequencing workflows. This word list is shown in Table 4.3. The metadata table and column descriptions from the SRA developer documentation (EMBL-EBI, 2013a) were used as a guide to select appropriate fields (see Table 4.2) and were inspected using SQL queries to quantify the number of records that appear to be annotated for a given protocol step. Occurrences in the field under inspection of one or more of the words in the list for that given protocol step were recorded. Estimation of the false negative rate from a subset of instances was not performed.

A search for the keywords listed in Table 4.3 was carried over all the metadata fields (listed in Table 4.2) deposited in the SRA. The results are summarised in Table 4.4.

Given the complexity of the SRA database schema, here we will discuss some of the issues and outline notable findings.

4.3.4 Aggregating data over experiment records

Metadata from an *Experiment* record is directly associated with an individual set of sequence data deposited at the SRA. However, it is important to note that metadata deposited in one or some subset of *Experiment* records may in fact represent equivalent metadata for all the *Experiment* records of a given *Study*. In order to investigate this, the relevant fields of all the *Experiment* records for every given *Study* record were aggregated. Searches for the key words outlined above were repeated. Any hits in the above lists were treated as evidence of metadata for the protocol steps for the entire study.

4.4 Results

The incidence of keywords is shown in Table 4.4.

Table	Field	Total records (in table)	Annotation record counts			
			Fragmentation	Adapter ligation	Enrichment	All steps
Study	**study_abstract**	29,958	376 (1.27%)	138 (0.47%)	941 (3.18%)	12 (0.04%)
Study	**study_description**		292 (0.98%)	136 (0.51%)	488 (1.65%)	53 (0.18%)
Sample	**description**	480,222	1,632 (0.34%)	896 (0.19%)	2,159 (0.45%)	653 (0.14%)
Experiment	**design_description**	419,620	11,705 (2.79%)	6,382 (1.53%)	16,779 (4.00%)	2,691 (0.64%)
Experiment	**library_selection**		1,493 (0.36%)	0 (0%)	0 (0%)	0 (0%)
Experiment	**library_construction_protocol**		29,799 (7.10%)	24,486 (5.84%)	31,782 (7.57%)	17,021 (4.06%)
Experiment	**experiment_attribute**		422 (0.10%)	1,026 (0.24%)	2,814 (0.67%)	129 (0.03%)

Table 4.4: Each column (on the right-hand side of the table) represents a sequencing step for which a word list is used to filter records where this step is annotated. Counts are the number of table records (without aggregating) exhibiting this particular annotation. "All steps" indicates the number of fields containing all three types of protocol step annotation, i.e. they all have keywords from each of the keyword lists.

We investigated the abundance and quality of annotation of next-generation sequencing protocol steps in notable fields (listed in Table 4.2) of the SRA (schema shown in Figure 4.1). In order to determine the quality of protocol annotation in the SRA metadata we set a minimum standard of annotation - that is records having annotation for all three protocol steps as determined by the presence of words from the keyword lists listed in Table 4.3. The SRA metadata comprises of two main levels - the top-level *Study* (the *Study* master Table in the database), and at the bottom-level by *Experiment* (the *Experiment* and *Sample* child tables). This feature of the SRA schema is represented by a one-to-many relationship between *Study* and *Experiment* records, and in order to gain a full picture of the metadata in the SRA, we investigated annotation in the individual SRA tables as well as aggregated *Experiment* records by their top-level *Study* - annotation was found at both levels of the schema.

When we examined records in the bottom-level *Experiment* table without aggregating by top-level *Study*, the most populated field in terms of protocol annotation was the *library_construction _protocol* field of the experiment table (*Experiment:library_construction_protocol*). Despite being the most populated field with respect to protocol annotation, fragmentation, adapter ligation and enrichment were annotated at 7.10%, 5.84% and 7.57% of all records respectively with only 4.06% of entries annotating all three protocol steps. We also found that approximately half (212,070, 51.12%) of the total records have a null (empty) entry in the *library_construction_protocol* field. The next most annotated field in terms of next generation sequencing sample preparation

protocol steps was the experiment table (*Experiment: design_description*), with 2.79%, 1.53% and 4.00% for fragmentation, adapter ligation and enrichment annotation respectively with only 0.64% of the records covering all of the three main protocol steps. In the same manner, we also examined top-level *Study* records and as noted in Table 4.4, we found a small number of depositions have protocol information within their top-level *Study* abstracts and/or study descriptions. In general, proper annotation should occur in the *Experiment:library_construction_protocol* field. Understandably these fields may contain words for or describe a protocol step in the abstract if it constituted a notable aspect of the submitters experiment. The small number of entries in the *Study:study_abstract* field had corresponding entries in the correct *Experiment:library_construction_protocol* field - (99.2%, 100% and 100% for fragmentation, ligation and library enrichment annotations respectively). Likewise entries in the *Experiment:design_description* field also had corresponding *Experiment:library_construction_protocol* field entries (99.0%, 100%, 100% for fragmentation, ligation and library enrichment annotations respectively). This indicates duplication of annotation and redundant data in the database.

We aggregated bottom-level *Experiment* records by their top-level *Study* because as outlined in the reagent kit data sub-section (4.4.2) focusing on the *Experiment* records may not reflect the level of annotations over individual studies. Using structured queries that aggregate the records in this manner is appropriate given the one-to-many relationship between their two respective tables in the schema. We found significant variation in how metadata is stored for a given *Study* (see Figure 4.2). In the SRA the number of experiment records associated with a given top-level study can vary from a single study with 1 experiment record to a single study with 15,548 experiment records (there are many studies with few experiments, and few studies with many experiments). A study was considered as being annotated for all three protocol steps if at least one of its corresponding experiment records contained annotations for these three steps. The variation shows there are inconsistencies in how annotations are stored across multiple experiment records for a given study. Given the potential to store redundant metadata it is possible that in a study containing a small number of experiment records all may be annotated whilst in a larger study containing many experiment records only one or a select few of these experiment records could be annotated.

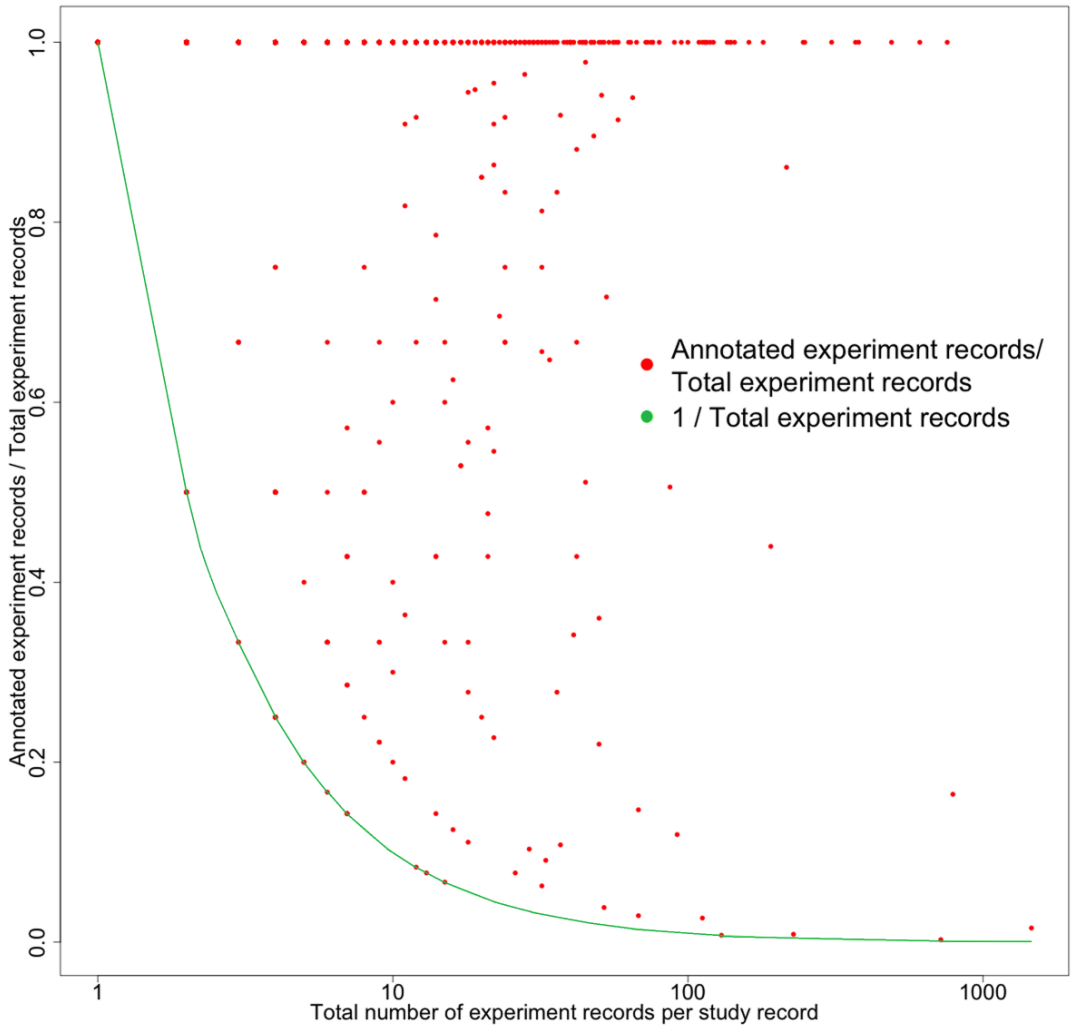


Figure 4.2: Ratio of annotated experiment records to total vs total number of experiment records per study. Only study records where at least one experiment record is fully annotated are included. Points where the ratio is 1 represent study records where all of the experiment records in a given study are fully annotated. The green line is a plot of 1/total number of experiment records in a given study. Points lying along this line are those studies where only one experiment record is fully annotated (presumably to represent the annotation of all the other experiment records). Points between these two curves represent studies where an intermediate number (neither 1 or all of the experiment records) are annotated.

In collating the annotation by the top-level *Study* as outlined above, we found the extent of annotation of the key next-generation sequencing workflow protocol steps (fragmentation, ligation and enrichment) is also low. Out of 29,598 study records, 21,799 (73.6%) of the studies where all their records are associated have no annotation whatsoever. Only 1,409 (4.7%) of the studies have full annotation. There is substantial overlap between the terms from the different lists that is shown in Figure 4.3. Our analysis of the SRA metadata found that only 84,911 out of a total of 414,788 experimental records (20.47%) exhibit annotation of any of the three protocol steps whilst only 16,930 (4.06%) of the total have had all three key annotation steps documented. These "fully annotated" records (those that have documented key protocol steps: fragmentation, ligation and library enrichment) comprise only 4.06% of all the records.

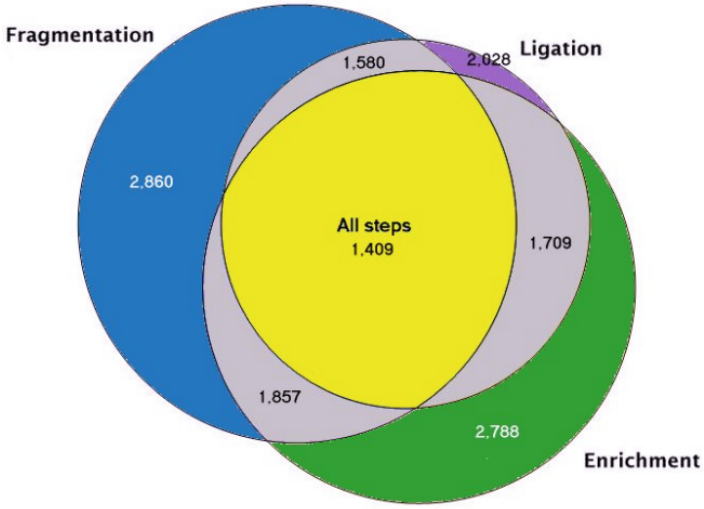


Figure 4.3: Venn diagram depicting the coverage of the annotations across the few SRA studies that possess at least one experiment record that has been annotated. Records sharing two annotation types are shown in grey: *fragmentation and ligation* (grey top), *ligation and enrichment* (grey right) and *fragmentation and enrichment* (grey bottom-left). Studies containing at least one experiment record with all three protocol step annotations are shown in yellow.

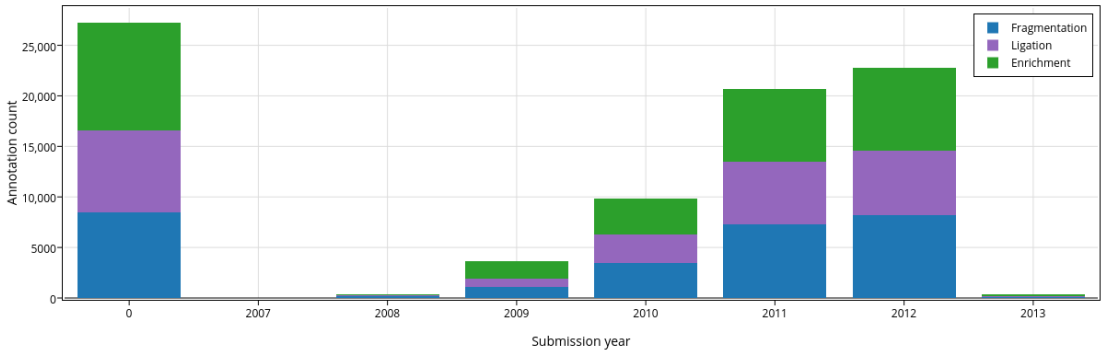


Figure 4.4: Annotations for protocol steps in *experiment:library_construction_protocol* field, by annotation type and year.

A significant number of *Experimental* level records were found to be undated - a breakdown of these records by annotation type and date is shown in Table 4.4;

4.4.1 URL as annotation

It is worth noting that 5,616 (1.33%) records of the field *Experiment: library_construction_protocol* contained a URL (uniform resource locator) - see Table 4.5 summarising URL counts for other fields. Whilst this is useful for external reference and in providing an audit trail in terms of protocol, the external resource referred to may be offline/unavailable at a time after original submission or the external resource may have changed (i.e. it may appear to reference the correct protocol, but in fact references a revised protocol). It is, therefore, necessary to ensure that a URL is only supplementary and does not constitute annotation of a protocol step in its entirety. Furthermore the use of URLs is not directly amenable to automated means of comparison as compared with metadata descriptors and values as outlined by the MIAME specification.(Brazma, 2003)

Table	Field	Total Records	URL Counts
study	study_abstract	29,598	444
study	study_description		817
study	study_attribute		422
sample	description	29,598	781
sample	sample_attribute		866
experiment	design_description	419,620	700
experiment	library_construction_protocol		5616
experiment	experiment_attribute		1,236

Table 4.5: Counts of fields containing a URL to an external site.

4.4.2 Reagent kit data

The use of reagent kits and corresponding annotation as such by some SRA users was also tested; additional SQL queries were written to search for the keywords "reagent" or "kit" in the experiment table's *library_construction_protocol* field. Counts of these records were compared against those probed for all annotation types. The results show that there is considerable overlap between annotated fields that discuss the protocol explicitly and those that mention a reagent kit, that is 23,288 of all experiment level records (5.55%) annotated for all protocol steps also contain keywords "reagent" or "kit" and this partitions across 2,055 (6.94%) top level SRA studies.

4.5 Discussion

From a variety of studies it is clear that potential biases may exist in next generation sequencing data due to the preparatory protocols carried out on the samples before submission. It is important to understand the size of these biases to determine the best practices and how they can affect issues such as the validity of comparative approaches in genomics using these data sets.

With this in mind, we have carried out a thorough study of the level of annotation of the initial protocol metadata deposited at the SRA. We have shown that the amount of annotation is very sparse with around 5% of the studies having keywords corresponding to all the steps that are relevant to the protocols. Around half of the experiment entries have a null (empty) entry in the fields where this data should be recorded.

In addition to the poor level of annotation as previously discussed, there are further issues associated with the SRA metadata. The annotation is inconsistent in that depositors are unclear about whether to provide annotation to all or some of the records at the experiment record level as shown by the large variance in experiment fields having annotation in any individual study. We have also found practices such as the use of URLs to provide a link to the appropriate annotation (see Supplementary Information). Accessing basic data such as date stamps is very difficult and appears to be stored inconsistently.

More seriously, the use of free text in these fields means that any large-scale computational survey of this metadata beyond the use of simple keyword searches would require advanced text mining techniques. (Nakazato and et al, 2013) undertook a study to constrain sequencing data by submission accession link cited in publications, the rationale being that this constraint would yield only higher quality submissions to the SRA. Their approach to the issue of the SRA meta-data, although very useful, is from a different perspective and stops short of examining the content of the metadata in the

context of the types of biases that can result due to the protocol steps in the next-generation sequencing workflow as we have done. Their work, however, corroborates our view: important fields of the metadata are free-text fields that are not amenable to efficient comparison.

The MINSEQE guidelines (Functional Genomics Data Society, 2012b) stipulates five essential components when submitting a High-throughput sequencing (HTS) experiment. In particular, the fifth component, essential experimental and data processing protocols concerns what we are discussing in this paper. Nonetheless, the SRA data on protocols is not amenable to automated methods due to the use of free-text fields and the absence of a more structured approach to recording important experiment protocol information. Depositors are likewise not obliged to complete this information.

It is unfortunate, given the size of the data sets being deposited in the SRA, more strictly enforced guidelines on the metadata have not been provided, along with the use of agreed vocabularies and ontologies. A more structured approach to the metadata would allow a deeper analysis and hence allow us to examine in much more detail the source of these biases along with the quality issue raised by Nakazato et al. above. Ideally this data would be represented via an ontology that is tailored towards protocols and submitters are obliged to fill out such data. Both approaches present their own challenges. For example, an ontology could be implemented in a similar fashion to the medical coding used in healthcare systems worldwide, and which is based on the International Classification of Diseases issued by the WHO (World Health Organisation) (World Health Organisation, 2017). This approach, however, would require an organisation to be set up to generate and maintain such a classification system for NGS experiments, human resources, and software to facilitate and manage the submissions process. If submitters are forcefully required to submit data for mandatory fields, this could deter depositions of experimental data, or result in submitters filling out inaccurate data to expedite and complete the submission process. A less elaborate, though still helpful, approach would be to oblige submitters to refer to the manufacturers reagent kit via a specific field with fixed values. Likewise, a clearer policy on the submission of protocol data at the study and experiment level would also avoid confusion.

The authors of the MIAME guidelines, published in 2001, have discussed the how their guidelines have been adopted in another paper (Brazma, 2009) - in particular with respect to its successes, challenges and failures, and have also commented on the MINSEQE specifications devised by FGED. They acknowledge that as MIAME were guidelines, this left a wide scope for methods to developed to implement MIAME, and have noted that some formats, such as MAGE-ML were not widely adopted - due to it's

complexity - whilst others, such as MAGE-TAB, which is tab-delimited and therefore, readable in a normal text-editor were more favourably adopted. They also point out that NGS experimental data, in contrast to that generated by Microarrays, is utilised in many different ways, for example in the re-squencing of genomes and metagenomes, and therefore, MINSEQE may not be relevant in these types of studies. Furthermore, in the case of NGS experimental data, depositions that involve sequencing human RNA or DNA, sufficiently long sequences may act as patient identifiable information. Therefore, only processed sequences, where such information has been anonymised, and not raw reads, should be deposited in public repositories. We agree with their view that, if the same successes achieved through the MIAME guidelines - which have enabled scientists to benefit from functional genomics - are not to be lost, then scientific journals should encourage the adoption of such practices, i.e. with respect to MINSEQE.

Potential implications The SRA is a huge resource of genomic and transcriptomic data with more than 60 trillion base pairs available for download as of September 2010 (Leinonen et al., 2011). In particular this resource should be invaluable for comparative genomics and meta-analyses. However, the poor level of annotation of protocol steps in preparing the data means that potential biases may exist in these data sets that will be unquantified.

Chapter 5

Testing Hadoop - The Protein Data Bank

"The true method of knowledge is experiment." – William Blake

The Hadoop platform, discussed in detail in chapter 2, is widely available, increasingly stable, and is becoming easier to install. In this chapter we will begin by discussing applications in structural Biology that have been implemented in Hadoop and discuss the adoption of the platform for this purpose. We will demonstrate that it can also be deployed for the analysis of large datasets made up of semi-structured data (Buneman, 1997; Abiteboul, 1997) such as the Protein Data Bank (PDB) (Abola et al., 1997; Berman et al., 2000).

Implementations of Hadoop are readily available on cloud computing platforms such as Amazon Web Services (Miller et al., 2010a). Nonetheless, the MapReduce formalism - traditionally implemented in Java - is not trivial to understand. Furthermore, in the field of structural Biology, the selection of software developed for the Hadoop platform is currently limited to applications which have been implemented specifically for a particular computational task, or is provided as a set of services to perform a specific set of tasks. Hence, there is a need to provide a framework that allows structural Biologists to run applications, using Hadoop without having to write bespoke code to make use of it.

In this chapter we will discuss the implementation of PDB-Hadoop, a framework that allows structural analysis tools to be executed on Hadoop clusters without needing to program or adapt existing tools to the MapReduce formalism, hence obviating the above-mentioned difficulties. In particular we focus on applying this technique using the PDB-Hadoop framework to the Protein Data Bank as both a use case and as a test case in this research thesis. We note it could also be applied to other semi-structured

datasets.

A comparison will be made between Hadoop cluster computing (PDB-Hadoop) and batch-scheduled cluster computing (Openlava), both discussed in technical detail in chapter 2, for executing three applications in structural Biology. Two of the applications compute dihedral (torsional) angles on peptide sequences in protein macromolecular structures over the entire Protein Data Bank. The third application performs the molecular docking of a ligand to 1000 structures in the Protein Data Bank.

5.1 Structural Biology on the Hadoop platform

The Protein Data Bank (PDB) is an archive of data describing the 3D shapes of proteins, nucleic acids, and molecular-complex assemblies derived from x-ray crystallographic, Nuclear Magnetic Resonance spectroscopy (NMR) and electron microscopy techniques (Abola et al., 1997; Berman et al., 2000). It also serves as a portal for structural genomics (Kouranov et al., 2006).

In chapter 2, sections 2.6 - 2.7, we discussed applications developed for the Hadoop platform for bioinformatics, processing of Next-generation sequencing data, and the deployment of such services to cloud service providers. There are also a number of applications for structural Biology implemented using MapReduce on Hadoop, which we will discuss in this section.

As discussed in Chapter 2, section 2.4.2, MapReduce on Hadoop enables high-throughput data processing which is scalable, fault tolerant, and which has enabled the record of sorting 100 TB of data at a rate of 0.578 TB/minute in the 2009 Gray sort competition (OMalley and Murthy, 2009). This benchmark was performed using Yahoo's *Hammer* cluster, which at the time comprised 3,452 nodes, each with 2x quad-core Xeon CPUs (2.5 GHz), 8 GB of RAM and 4x SATA hard disks. Each node was connected using 1 gigabit Ethernet. Hadoop can also be leveraged for applications in structural Biology. We shall discuss these applications in the following sub-sections. Whilst our focus will be on systems developed for the Hadoop platform, for purposes of comparison, we shall sometimes refer to similar systems implemented on other platforms.

In sections 5.1.1 - 5.1.2 we discuss two computationally intensive methods used in structural Biology that are amenable to Hadoop. The findings in these sections are listed in Table 5.2.

5.1.1 Molecular docking

Molecular docking typically involves simulating the electrostatic interactions between a ligand (often a potential drug molecule) and a target protein (Morris and Lim-Wilby, 2008; Meng et al., 2011). It is used to score ligands on their affinity to the target, usually for the purposes of drug development - a process that is complex, time-consuming and expensive (Moses et al., 2005; Rawlins, 2004). We will perform a docking experiment using our PDB-Hadoop framework, which we will discuss later in section 5.5.2. We shall first discuss existing work in this area.

A number of molecular docking applications have been implemented to exploit the Hadoop platform. For example, Ellingson and Baudry (2011) at the Oak Ridge National Laboratory in the US, have utilised AutoDock4 on a private 68 node Hadoop cluster to perform the docking of 2,637 compounds from the Directory of Useful Decoys (DUD) database (Huang et al., 2006), against the Human estrogen receptor alpha agonist protein (PDB entry 1L2I, (Shiau et al., 2002)). They used the DUD database because it contains ligands that bind to the target, as well as chemically similar ligands that do not, i.e. "decoys". This allowed them to test the reproducibility of the docking experiments - they found that the results of running AutoDock on Hadoop were consistent with the experiments of (Huang et al., 2006), specifically that the percentage of known binding ligands correlated with the percentage ranked in the DUD database. In their configuration they used 10 mappers/node on the 57 nodes of their cluster that were dedicated to run MapReduce Tasks giving 570 mappers running in parallel. This resulted in a 450x speed-up of AutoDock in performing the docking task on Hadoop as compared with utilising AutoDock itself to manage the parallelisation. Furthermore, they report that 95.59% of CPU time is used by AutoDock, and, therefore, there is less than a 5% overhead in running AutoDock in a Hadoop map process, and that, as the tails seen in the graphs of the CPU load were steep, this indicates that job initialisation and termination are not resource intensive.

As a comparison, Zhang et al. (2013) conducted the same experiment using the DUD database with MPI and a multi-threading parallel scheme at an extremely large scale (15,408 CPUs). They found that VinaLC scaled very well up to with an overhead of only 3.94%. 17 million flexible compound docking calculations were completed on 15,408 CPUs within 24 h. 70% of the targets in the DUD data set were recovered using VinaLC. These applications, and the others we will discuss are listed in Table 5.1 for comparison.

Another system for protein-ligand binding on Hadoop, developed by Hung and Hua (2013) is a scalable docking service called Cloud-PLBS (Cloud Protein Ligand Binding Service), which utilises the SMAP docking tool (Xie and Bourne, 2007). Their system

employs an additional virtualisation system, whereby the Hadoop slave nodes run on virtualised machines and are instantiated depending on the input job requirements. In terms of benchmarking performance, they compared stand-alone, sequential processing of protein-ligand pairs using SMAP, with parallel execution of SMAP within a Hadoop map function - specifically 2, 4, 6 and 8 mappers. They observed that in docking 40 protein-ligand pairs, reduction in execution time using Hadoop vs. stand-alone for 2, 4, 6 and 8 mappers was 33.92%, 56.97%, 70.21%, 77.65%, respectively.

They also tested the fault-tolerance of Hadoop in running their protein-ligand pair docking system by simulating task failures in 50% of the map steps by removing node(s) from service. They observed that the docking jobs still completed. This is because YARN's task-tracker service monitors the currently executing MapReduce tasks in the running containers through *heartbeats* (discussed in Chapter 2), and when a MapReduce task does not respond within the timeout period (specified in `mapred.task.timeout`, in *mapred-default.xml*), it re-schedules the *map* step job on a different node a up to a certain number of times (specified in `mapred.map.max.attempts`, in *mapred-default.xml*). As a consequence, the execution time for processing 40 protein-ligand pairs was increased by 391s (51.86%), 313s (63.75%), 263s (77.35%) and 165s (64.70%) for 2, 4, 6 and 8 mappers, respectively. As we have discussed in Chapter 2 (section 2.4.2), fault-tolerant distributed computation is a feature of Hadoop based applications, and this resilience in the execution of tasks is important, because the likelihood of a node failing increases with the scaling-up of a cluster. Fault-tolerance is also desirable in web-based services such as Cloud-PLBS, which serve to automate computational jobs and present the results to the user, without requiring third-party intervention to rectify failed jobs. It should be noted that no reference to source code for their system is provided in their paper, and the Cloud-PLBS service at *http://bioinfo.cs.pu.edu.tw/cloud-PLBS/index.html* is no longer available.

5.1.1.1 Clustering of protein-ligand complexes on Hadoop

One of the challenges in the field of molecular docking studies arises from the requirement to search the conformational space of protein-ligand complexes generated across docking experiments. This is necessary to select the most likely conformations of protein-ligand complexes, and, therefore, the putative ligands (potential drug molecules) which partake in these interactions. In such experiments large numbers of protein-ligand complexes are generated, docked, and scored (Estrada et al., 2012), and it is, therefore, necessary to select a subset of putative ligands based on significant protein-ligand interactions which may be developed as medicines.

In this paper, they observe that selecting the native conformation, based on the as-

sumption that the lowest energetically scored conformation (as computed by an energy function) represents the native binding of the ligand and protein, is not reliable, even in larger sets of conformations. This is often due ro non-native ligand-protein complexes generating falsely low energy scores. They point out that, whilst hierarchical clustering techniques are a logical way of addressing this problem - as the lowest scoring, most densely populated clusters overlap with native conformation - most clustering algorithms are computationally expensive, and scale poorly with large datasets. They have, therefore, implemented a system using MapReduce on Hadoop to address this issue. They used two datasets, of size 5 TB (3,872 million ligand conformations) and 1 TB (768 million ligand conformations), generated from the Docking at Home volunteer grid computing project (Docking@Home) (Estrada et al., 2010), which utilised the CHARMM (Brooks et al., 1983) algorithm. The previous examples we have discussed (Cloud-PLBS by Hung and Hua (2013), the structural alignment application by Liu et al. (2016) and the large scale docking experiment using the DUD dataset by Ellingson and Baudry (2011)) did not fully implement their solutions in MapReduce. This would have involved implementing (or re-implementing) algorithms using MapReduce, but instead exploited Hadoop's *map* step to encapsulate and execute external applications. Estrada et al. (2012)'s method, however, is fully implemented in MapReduce.

A map step is used which geometrically reduces the conformational space. This is stored in an Octree data structure (Samet, 1988), together with a unique identifier (an Octkey) used for traversal. This is achieved by projecting the x, y, z components of the conformations onto a 2D plane, and calculating their gradients (for x, y, and z) which are then encoded into a single point in the Octree. A reduce step is used to aggregate conformations in the Octree. Further MapReduce operations are then used to traverse the Octree using the Octkey identifier.

In order to compare the accuracy of their Hadoop-based Octree method (for selecting native conformations from the ensemble of complexes) against other approaches, namely Hierarchical clustering and Minimum Energy selection, they docked 100,000 protein-ligand complexes each for HIV, Trypsin, and P38alpha. They obtained 80%, 75% and 25% accuracy for Hadoop based Octree, Hierarchical Clustering and Minimum Energy methods, respectively.

They also examined the accuracy of selecting native conformations from the cross-docking data in the Docking@Home datasets, whereby each conformation of the ligand in the set of complexes is docked with each conformation of protein. In doing so, they compared their Octree method with the Energy Minimum approach, and observed 43.8% and 5.8% accuracy, respectively.

In testing the scalability in processing the 5 TB dataset which, as discussed con-

tains 3,872 million ligand conformations, they used a Hadoop cluster where each node possesses 32 cores (4x Octacore AMD Opteron 2.4 GHz), and up to a maximum of 32 nodes were available. The range of the scaling used was 1 node of 32 cores (to process 121 million conformations) to 32 nodes with a total of 1024 cores (to process the full dataset) and analysed 1, 2, 4, 8, 16, and 32 nodes - in all cases, the number of ligand conformations processed per core was 3.8 million.

It is not stated in their paper how many map processes were running per core, but it is assumed that it is 1 map step per core. Whilst they demonstrated their method was amenable to scaling, they observed an appreciable decrease in parallel efficiency with the increase in cores, from 99.1% down to 43.8% for 64 cores (2 nodes) and 1024 (32 nodes), respectively. This appears to be due to the increased overhead in communications between the processes as the number of processes increases (communications to computation ratio).

A similar application (Paschina et al., 2015) was developed on the Hadoop platform that partitions the results of molecular dynamics simulations. The trajectories of atom positions, velocities and energies as a function of time are clustered, as large datasets. This method yields important information about the most probable conformations of proteins in ensembles. Their system employs the GROMOS algorithm (Scott et al., 1999), which is not inherently parallel, by implementing it as a series of map and reduce functions so as to utilise Hadoop. They tested their parallelised MapReduce implementation of the GROMOS algorithm on a Hadoop cluster comprising of 1 master and 3 slave nodes, each comprising two hexa-core Xeon E5645 CPUs 32 GB of RAM and 2 TB of disk space. They observed up to 10 and 7 times speed-up (over using sequential GROMOS) of the first and second phases, and final two phases of their algorithm, respectively.

A docking application also relevant to our discussion, although not implemented in Hadoop, has been developed by Ocaña et al. (2014) using a scientific workflow management tool, SciCumulus deployed on AWS. Their system employed molecular docking, using AutoDock4 and Vina, on their platform to explore both drug discovery and scalability. Their drug discovery objective was the identification of putative drug ligands that bind to Cysteine Proteases of Protozoan genomes utilising 10,000 protein-ligand complexes. This aims to facilitate the development of drugs for the Neglected Tropical Diseases (NTDs). In investigating the scaling-up of the computational task, they utilised up to 32 heterogeneous nodes (containing varied numbers of cores) to include a total of 128 Amazon AWS EC2 cores. They observed an almost linear relationship between number of nodes and speed, but this plateaued as they approached the maximum of 32 nodes, suggesting less benefit in scaling beyond this. They point out that

this is likely due to more complicated load balancing in the set of heterogeneous nodes. The result of their docking experiments using the Cysteine Protease-ligand complexes identified 287 and 355 putative ligands for AutoDock4 and Vina, respectively. It is important to note, however, that these potential drug molecules have, on average, RMSDs greater than $2 - 3$ Å (Angstroms) which is the maximum accepted value for a useful result.

5.1.2 Structural Alignment

The alignment of proteins by structure, as opposed to by sequence, is a computational technique used to identify homologous polymer structures within proteins that may be conserved between proteins. The technique facilitates the study of the structural and evolutionary relationships of proteins with low sequence similarity (Gibrat et al., 1996; Orengo et al., 1997). A variety of of algorithms have been developed to perform structural alignment of proteins, such as, for example, STRUCTAL (Structural Analysis Algorithm), DALI (Distant Alignment) (Holm and Sander, 1998), CE (Combinatorial Extension) (Shindyalov and Bourne, 1998), VAST (Vector Alignment Search Tool) (Gibrat et al., 1996), and FATCAT (Flexible structure Alignment by Chaining Aligned fragment pairs allowing Twists) (Ye and Godzik, 2003). Also, SSAP (Sequential Structure Alignment Program) (Orengo and Taylor, 1996), and MUSTANG (Multiple Sequence Alignment Algorithm) (Konagurthu et al., 2006). The technique has been applied to the study of protein binding sites, and solvent exposed surfaces (these effect the energetics of protein-ligand conformations) (Ma et al., 2003; Konc and Janežič, 2010; Liu et al., 2016).

Structural alignment algorithms are usually computationally complex, (Kolodny and Linial, 2004) present a method which runs at best in approximately Polynomial time, but they also point out that approximations are often used. There is also a need to apply such techniques at scale. For example, Hadoop has been employed by Liu et al. (2016) who implemented structural alignment for binding site prediction. Using a test sets of 200 and 48 ligand-protein complexes, they were able to achieve 93% and 98% accuracy, respectively, and were able to improve the efficiency of the experiment by exploiting parallelisation. A service for structurally aligning pairs of proteins has also been implemented for the Hadoop platform by the developers of Cloud-PLBS (Hung and Lin, 2013) (discussed in the previous sub-section). It utilises the same distributed architecture as Cloud-PLBS, that is, individual Hadoop nodes running within their own VM, and each VM running a *map* and *reduce* process. As with Cloud-PBS, a web-interface is used to enable the user to provide input of two PDB files by their PDB-ID. To perform the structural alignment they state their system uses the DALI and VAST

algorithms. Whilst the authors do detail the basis of RMSD (Root Mean Square Deviation) in structural alignment algorithms, and discuss refinement methods in their paper, they claim to implement these algorithms in MapReduce. Also, as previously noted with Cloud-PLBS, there is no source code available, and the corresponding web service is unavailable. It is highly likely that, given the complexity of these algorithms and that there are already implementations available, the same method used in Cloud-PLBS - execution of an external program within a map step - is employed.

A similar bioinformatics SaaS (Software-as-a-Service) for structural alignment of proteins, has been developed for the Microsoft Azure platform - Cloud4Psi developed by Mrozek et al. (2014). Their service utilises three newer algorithms that are implemented in the BioJava project, and which are derived from CE (jCE), and FATCAT (jFATCAT-rigid and jFATCAT-flexible) (Prlić et al., 2012). They tested their system on a subset of 1,000 PDB structures for both scalability and reproducibility. For scalability, two different scaling methods were compared: *horizontal-scaling* (i.e. by adding more nodes to the system) and *vertical-scaling* (i.e. by using nodes with more CPU cores). They found that, whilst both scaling methods increased the n-fold speed-up for each of the three algorithms, both suffered a decrease in the performance gains - for horizontal scaling, this was found to be the result of increased disk I/O due to multiple nodes utilising a shared VHD (Virtual Hard-Disk), and for vertical scaling this was due to an increase in processes on the same node (due to higher specification of each node), resulting in increased CPU utilisation. As the horizontal scaling method suffered less from this effect, it was the method chosen. For reproducibility of results, they found that each of the three algorithms produced the same results independent of cluster configuration and scaling used.

Table 5.1: Summary of applications for structural Biology on Hadoop and other distributed computing platorms

Study	Type / Platform	Cluster	Dataset	Summary of findings
Ellingson and Baudry (2011)	Docking / Hadoop	68 nodes, each with 16 cores (570 mappers)	2,637 putative ligands from DUD database against a single receptor	**Reproducibility:** the percentage of known binding ligands correlated well with the percentage ranked in the DUD database. **Benchmarking:** Using 570 map processes, a speed-up 450x of AutoDock was achieved on Hadoop, compared with utilising AutoDock itself to manage the parallelisation. 95.59% of CPU time was used by AutoDock, and less than a 5% overhead in running AutoDock in map processes.
Zhang et al. (2013)	Docking / MPI	15K cores	Same dataset as above.	This research project seeks to reproduce that of Ellingson and Baudry (2011) using a mixed MPI and multi-threading parallel scheme instead of Hadoop by re-writing AutoDock Vina to support MPI (VinaLC). **Reproducibility:** 70% of the targets in the DUD data set were recovered using VinaLC. **Scalability:** VinaLC scales very well up to 15,408 CPUs with an overhead of only 3.94%. 17 million flexible compound docking calculations were completed on 15408 CPUs within 24 h.

Hung and Hua (2013)	Docking / Hadoop	4 nodes, each with 8 cores	40 protein-ligand pairs	**Benchmarking:** in docking 40 ligand pairs (Hadoop running SMAP vs. Stand-alone SMAP), they achieved a reduction in execution time using Hadoop vs. stand-alone was 33.92, 56.97, 70.21, 77.65 for 2, 4, 6 and 8 mappers, respectively. **Fault-tolerance:** When node failure was simulated in 50% of the nodes, execution time was increased by 391s, 313s, 263s and 165s for 2, 4, 6 and 8 mappers, respectively.

Estrada et al. (2012)	Docking & Clustering / Hadoop	32 nodes each with 32 cores (4x Octa-core AMD Opteron 2.4 GHz).	5 TB dataset for scalability tests (3,872 M ligand conformations) and to test their Octree method 100,000 protein-ligand complexes each for HIV, Trypsin, and P38alpha.	**Scalability:** 1 node with 32 cores (to process 121 M of the conformations) to 32 nodes with a total of 1024 cores (to process the full dataset) and analysed 1, 2, 4, 8, 16, and 32 nodes - in all cases, the number of ligand conformations processed per core was 3.8 M. They observed an appreciable decrease in parallel efficiency with the increase in cores, from 99.1% down to 43.8% for 64 cores (2 nodes) and 1024 (32 nodes), respectively. Likely due to the increased overhead in communications between the processes as the number of processes increases (communications to computation ratio). **Accuracy:** Hadoop-based Octree method (for selecting native conformations from the ensemble of complexes) was tested against other approaches: Hierarchical clustering and Minimum Energy selection, they docked 100,000 protein-ligand complexes each for HIV, Trypsin, and P38alpha. They obtained 80%, 75% and 25% accuracy for Hadoop based Octree, Hierarchical Clustering and Minimum Energy methods, respectively. Also tested accuracy of selecting native conformations from the cross-docking data in the Docking@Home datasets. In doing so, they compared their Octree method with the Energy Minimum approach, and observed 43.8% and 5.8% accuracy, respectively.

Ocaña et al. (2014)	Docking / SciCumulus	up to 128 AWS EC2 cores in nodes containing different numbers of cores.	10,000 protein-ligand pairs. Docking results reported for subset of 1,000.	**Scalability:** up to 32 nodes, an almost linear relationship with speed was observed, but thereafter started to plateau, suggesting less benefit in scaling beyond this. Likely a result of more complex load balancing in the heterogeneous set of nodes. In docking for Cysteine Protease-ligand complexes, 287 and 355 putative ligands were found for AutoDock4 and Vina, respectively. NB: On average, their RMSDs were greater than $2 - 3$ Å (Angstroms), the maximum accepted value for a useful result.
Liu et al. (2016)	Structural Alignment / Hadoop	13 node cluster (12 slave nodes). CPU/RAM resources for each node not specified.	200 and 48 ligand-protein complexes	**Accuracy:** For binding site prediction using structural alignment, they achieved 93% and 98% accuracy for 200 and 48 ligand-protein complexes respectively. **Scaling:** Job runtime decreased with increasing number of concurrent mappers, with peak performance using 8 mappers. Increasing beyond 8 mappers to a total of 30 mappers resulted in a slight, steady degradation in performance.
Hung and Hua (2013)	Structural Alignment / Hadoop	4 nodes, each with 8 cores	40 protein-ligand pairs	**Benchmarking:** For each algorithm (DALI and VAST), sequential execution (single process) was compared with 2, 4 and 8 map processes. Execution time was improved by factors proportional to the number of mappers. Their structural refinement method, after alignment, improved RMSD scores from DALI and VAST by approximately 7% and 6%, respectively.

Mrozek et al. (2014)	Structural Alignment / MS Azure	1 node with up to 8 cores (Azure A4/ExtraLarge node)	1,000 PDB structures	**Scalability:** Two different scaling methods were compared, *horizontal-scaling* and *vertical-scaling*, for the three algorithms, jCE, jFATCAT-rigid and jFATCAT-flexible. Both scaling strategies increased the n-fold speed-up for all algorithms, but resulted in decreased acceleration with scale-up. In horizontal scaling, this resulted from increased disk I/O (multiple nodes utilising a shared VHD). In vertical scaling, this resulted from increased CPU utilisation (increase in number of processes on higher-spec. machines). **Reproducibility:** each of the three algorithms produced the same results independent of cluster configuration and scaling strategy.
Paschina et al. (2015)	Protein Clustering / Hadoop	1 master and 3 slave nodes, each comprising two hexa-core Xeon E5645 CPUs 32 GB of RAM	500 and 2000 molecular structures	**Scalability:** Speed-up of up to 10 and 7 times speed-up (over using sequential GROMOS) was achieved for the first and second phases, and final two phases of their algorithm, respectively. **Reproducibility:**They computed the most probable conformations from the molecular dynamics (MD) trajectory of the human Hsp70 chaperone protein in complex with ADP ligand. They found that their Hadoop implementation obtained the same results as sequential execution of the GROMOS algorithm in the GROMACS package.

5.1.3 Other Bioinformatics applications using Hadoop

Large-scale processing of molecular data is desirable in both applications. Such techniques facilitate the *in-silico* study of vast arrays of molecular compounds and macromolecular structures that are available from large molecular databases, which are also increasing in size and diversity (Degtyarenko et al., 2007; Pence and Williams, 2010; Allen, 2002; Wang et al., 2009a; Abola et al., 1997; Berman et al., 2000). A number of scalable structural Biology methods have been provided by the bioinformatics Group at UCL (University College London) as web-based services through their Protein Analysis Workbench (Buchan et al., 2013). These are accessible via SOAP (Simple Object Access Protocol), and XML-RPC (Extensible Markup Language-Remote Procedure Call) protocols. Importantly, the most commonly used methods have also been deployed as Java packages specifically for the Hadoop platform. This includes PSIPRED for protein structure prediction (McGuffin et al., 2000), GenTHREADER for protein fold recognition method using genomic sequences (Jones, 1999), BioSerf - a homology modelling protocol, MEMSAT for improving accuracy of transmembrane protein topology prediction (Jones, 2007), DomPred for protein domain boundary prediction (Bryson et al., 2007), MetSite for predicting clusters of metal-binding residues (Sodhi et al., 2004), and FFPred which uses a machine learning approach for predicting protein function (Lobley et al., 2008).

5.1.4 Scalability, performance, consistency and gains in using Hadoop

The publications we have reviewed in 5.1, with respect to structural Biology applications on Hadoop, indicate that some adjustment of parameters have been made, but these largely focused on how the applications scale with the number of nodes. They show that performance is linear though the gains in performance tend to reduce as the systems are scaled up. In molecular docking, Estrada et al. (2012) observed a fall in parallel efficiency of 55.3% (99.1% - 43.8%) when scaling from 2 nodes to 32 nodes. In structural alignment, Liu et al. (2016) observed that performance degraded slightly after 8 mappers was increased to 30. Furthermore, this trend has also been observed on the MS Azure platform we have discussed for comparison - in scaling Cloud4Psi, also a structural alignment application Mrozek et al. (2014), observed that horizontal scaling resulted in performance degradation as a result of an increase in nodes sharing a virtual disk, and that vertical scaling resulted in performance degradation as a result of increased CPU utilisation (due to more processes running per node).

The comparison of vertical and horizontal scaling in Cloud4Psi indicates significant change in performance, so configuration *is* important. As noted with the Cloud4Psi

example, there can be a significant variation depending on configuration. Performance gains across applications, therefore, are dependent on configuration.

The platform, and method of distribution, also dictate performance and scalability. In observing two identical docking operations on the DUD database, one using a 1,088 core Hadoop cluster (Ellingson and Baudry, 2011), and the other using 15,408 cores with a mixed parallel MPI implementation of AutoDock (Zhang et al., 2013), the Hadoop cluster took 69 hours, and the MPI implementation completed within 24 hours. Whilst this is certainly a result of the number of cores, the MPI system scaled better, with only very slight degradation in performance after 6,000 CPU cores. Although this comparison involved the same docking operation and dataset using different platforms, currently, there are no comparisons of performance between Hadoop and batch-schedulers on the same cluster apparatus in the literature. In section 5.5 of this chapter, we test a framework implemented using Hadoop with that of batch-scheduled processing on the same cluster apparatus, for applications in structural Biology, and provide an assessment of their performance.

As Hadoop platforms stabilise, the significant advantage of its employment is of using a platform where the computation is expressed explicitly in terms of an algebra. This makes building workflows easier by allowing the developer to concentrate on the calculation, rather than the process. Nonetheless, there is a significant gap in take up as such systems remain difficult to manage and deploy - hence there is a need for an easy to use system that we discuss in section 5.2.

5.1.5 Adoption of Hadoop for application to Bioinformatics and Structural Biology

We have discussed bioinformatics applications and software that has been developed the Hadoop platform in sections 2.6 and 5.1 - 5.1.3. There are, many applications for Hadoop in bioinformatics. On the other hand, there has not been widespread adoption of MapReduce for applications in structural Biology.

From the review in 5.1 we show Hadoop to be a stable and effective platform. Hence, the main reason is because the amount of time required to port software to Hadoop and maintain it does not offer sufficient benefit, and is non-trivial. We have observed and discussed positive results in structural Biology using Hadoop. These are listed in Table 5.1 alongside some similar research projects implemented on other distributed computing platforms for comparison. It is, likely that powerful, but highly specific methods such as MapReduce, will find application only in a subset of scientific computing tasks. One example is in the processing of high-throughput Next-generation sequencing data, which Hadoop is ideally suited for.

Aside from the key advantages of Hadoop which we have already covered in detail in Chapter 2 (section 2.4), namely high-throughput processing, scalability and fault-tolerance, the underlying software is also Open Source, well supported and documented, and can be applied to un-structured and semi-structured data. However, the Hadoop ecosystem is rapidly evolving, which requires regular release upgrades. These can be potentially be difficult to deploy, and often add new components to the ecosystem which increase the potential to introduce bugs that may affect different areas of the system.

To address this, organisations such as Cloudera and Hortonworks (Cloudera, 2016; Hortonworks, 2016) provide supported Hadoop-stack releases, and cloud-service providers such as Amazon offer managed-services, for example Elastic Map Reduce (EMR) (Amazon, 2016). Implementing systems on the Hadoop platform, as discussed, also requires specialist programming knowledge of MapReduce, and if the relevant Hadoop cluster is maintained in-house, specialist skills in maintaining an Hadoop cluster are also required. For this reason, managed services are often utilised by enterprise companies because such systems have been deployed and tested by technical experts and therefore mitigates risks, and dispenses the need to employ or train in-house skilled personnel to maintain a Hadoop cluster.

In the remainder of this chapter we will present PDB-Hadoop, a package that has been developed, for research and testing purposes, with the factors we have discussed in mind and is easy to deploy. It allows the user apply the high-throughput capabilities of Hadoop Streaming for processing macro-molecular structures in the Protein Data Bank (PDB). The framework consists of a simple set of bash scripts that can be used to run existing applications in structural Biology using Hadoop, without requiring extensive expertise in MapReduce programming, and can also be used to process other datasets consisting of semi-structured data.

5.2 PDB-Hadoop - Structural Biology framework

PDB-Hadoop is a framework that we have developed to facilitate the high-throughput execution of protein structure analysis tools to be carried out on the entire (or subsets of) the Protein Data Bank using the Apache Hadoop platform. A user can deploy PDB-Hadoop on their local or cloud-based Hadoop platform without having to explicitly write Hadoop code which commonly requires programming experience in Java and MapReduce.

In PDB-Hadoop the execution of the user's structural Biology tool is performed within a single *map* step, i.e. it has a *mapper* architecture. It can function on a stand-alone basis or may be extended to include further MapReduce operations. A useful

feature of PDB-Hadoop is that it can be applied to run on any data set that features a large number of small-sized flat files - conventionally Hadoop is most often applied only to large files due to the inefficiency of HDFS (Hadoop's Distributed File System) large block sizes for storing small files (Mackey et al., 2009; Dong et al., 2010). This issue is solved by concatenation of the dataset prior to transfer to HDFS, which we will discuss in the following section of this chapter. The PDB also provides an ideal test case for the application of MapReduce to other similar and large semi-structured datasets consisting of large numbers of typically small files.

PDB-Hadoop makes use of the scalability and fault-tolerance capabilities offered by Apache Hadoop (discussed in chapter 2 section 2.4). It also enables the user to conveniently utilise Hadoop, a powerful distributed computing platform that is supported by a large number of cloud service providers, without having to explicitly write their own MapReduce code. PDB-Hadoop can also be integrated within larger analysis pipelines that are commonplace in bioinformatics analyses.

5.3 The Protein Data Bank - a semi-structured dataset

As of March 2017 the PDB contains 128,330 database entries, in multiple file formats for each entry. These PDB entries are provided in compressed or uncompressed format through an FTP archive. The snapshot of the entire FTP archive as of March 2017 is $\sim$ 757 GB. This indicates the same data in different file formats. However, the entirety of the PDB entries when only using the Brookhaven PDB file format[1] (the most widely used format, used herein) is $\sim$ 100 GB. It is these 128,330 PDB file format entries that we have used and refer to in this chapter, and to which we will refer to as the entirety of the PDB. As previously mentioned, an important consideration is that Hadoop's distributed file system (HDFS) is not optimised for the large numbers of small files that comprise the Protein Data Bank (and other such similarly structured datasets). The solution used here is to concatenate the entirety (or a subset) of the PDB into a single file.

Datasets such as the PDB have a unique record level identifier referred to as an accession number – in the case of the PDB this is the PDB-ID that also serves as the filename of the macromolecular structure. The PDB-Hadoop framework is based on Hadoop streaming MapReduce (discussed in chapter 2, section 2.4.5) and employs a *map* step μ_P (defined later in section 5.4.1). As discussed in chapter 2 (section 2.4.1), the central structure of MapReduce is the key-value pair tuple $\langle k, v \rangle$, where k is the key and v is the value. In PDB-Hadoop we, therefore, represent a PDB database entry

[1] Brookhaven PDB format files usually have a *.ent* filename extension

as a key-value pair tuple, where we designate a MapReduce key k for the accession number (PDB-ID), and the value v holds the PDB data structure (representing the macromolecular structure). The PDB entry records for the structure are stored in v as a single string as we shall discuss shortly. The set of these key-value pair tuples, representing a set P of N PDB entries, is defined as follows:

$$P = \langle p_1, p_2, ..., p_N \rangle \quad \begin{array}{l} \text{where } p_i \text{ is a PDB file for a} \\ \text{single PDB database entry} \end{array} \tag{5.1}$$

The set of tuples P representing the PDB entries is iterated over using the Linux *find* command recursively on the folder containing the PDB files[2]. The concatenation process is achieved using a bash script provided in the framework that implements the procedure, which we define as *cnvpdb*, and which is detailed in the Appendix. The procedure is a one off-step - i.e. it is performed once on a local copy of the PDB and generates a data file represented by:

$$P' \quad = \quad \langle k, v \rangle_{i=1}^{N} \tag{5.2}$$

The concatenation procedure *cnvpdb* (Appendix 1.1) may also be carried out periodically in cases where a local copy of the PDB is to be synchronised with the online Protein Data Bank archive site.

5.4 PDB-Hadoop architecture

As noted previously, the architecture of PDB-Hadoop, depicted in Figure 5.1, is centred around a single *map* step μ_P (defined in 5.4.1 below) in which the users structural analysis software is run. This map step encapsulates the user's analysis program that is executed sequentially within a YARN container for each key-value pair in the *split* of P' which resides on Hadoop's HDFS filesystem. As discussed in chapter 2 section 2.4.4, a YARN container is a logical group of resources such as ⟨4 GB RAM, 2x CPUs⟩, and because the user program is executed within a map step its resources (memory and CPU utilisation) are allocated and controlled by virtue of instances of the map step running sequentially in a YARN container. The architecture of PDB-Hadoop is depicted in Figure 5.1. Across the whole Hadoop cluster there will be multiple container instances running on each compute node which achieves parallelisation of the user job and affords

[2]The PDB format files are distributed by the PDB database into approximately 1000 filesystem sub-directories, and so recursive iteration of these folders is necessary to access each PDB file.

scalability (and redundancy).

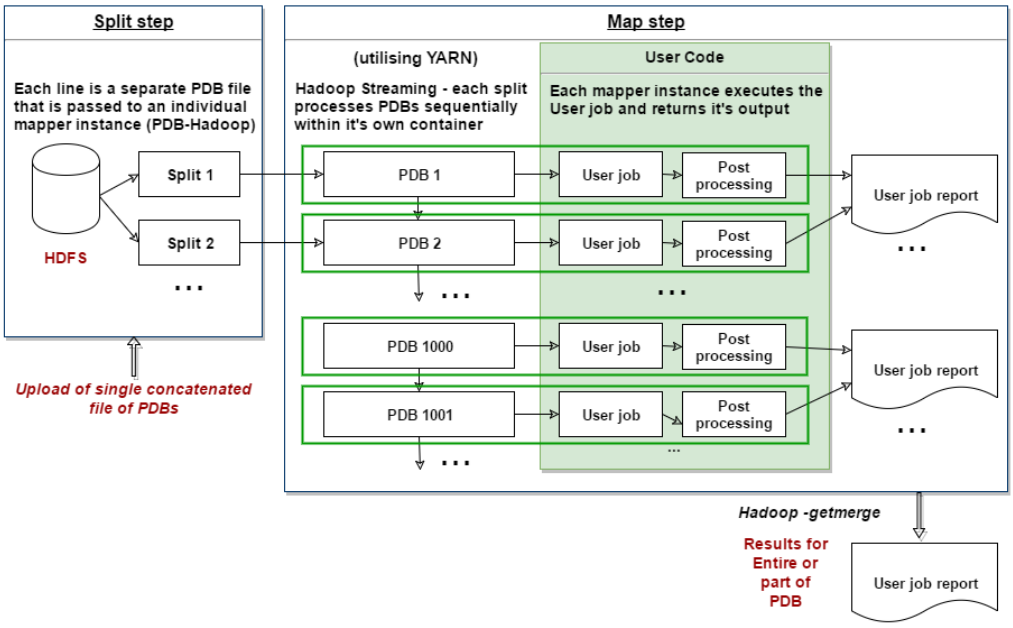


Figure 5.1: Architecture of PDB-Hadoop depicting HDFS *splits* and the processes employed within the μ_P *map* step. YARN containers are shown in green rectangles. The number of containers instantiated on a single node is dependent on the memory configuration of YARN.

5.4.1 Spawning the user analysis software - PDB map μ_P

This map step μ_P, described in algorithm 2, receives a list of key-value pair $\langle k,v \rangle$ comprising of the accession id (PDB-ID), k and record data for the molecular structure stored as a single string v. It is responsible for spawning an instance of the user's analysis software with the YARN container as the process owner according to a set of user defined parameters. Each instance of the map step also captures the output of the user's analysis job. For this reason, PDB-Hadoop provides the user the option to run a user operation on the job's output, such as text-parsing, typically extracting a specific set of outputs from a log file - we term this *post-processing*. Because PDB-Hadoop utilises Hadoop standard I/O streaming, this processing step, if enabled, occurs during the processing of each PDB-entry, that is after the job output has been produced by the user software and before the result is returned to the I/O stream for Hadoop to capture in the output logs.

Algorithm 2: MapReduce *map* step, implemented using Hadoop Streaming, to encapsulate the execution of user's structural analysis job and *post-processing* step. It recieves a partition of data (Hadoop *split*) of P' to operate on.

1 function $\mu_P(P')$;

Input : Set of tuples P' in the form $\langle k, v \rangle_{i=1}^{N}$ representing concatenated PDB entries, path to user *analysis* program, path to user *post-processing* program (*postprocProgramPath*).

Output: results of user *analysis* program and any *post-processing* echoed to Hadoop container's standard I/O stream.

2 $CRLF \leftarrow$ #13#10; *// ASCII codes for CRLF EOL (End of Line) delimiter*

3 $CD \leftarrow$ "^|"; *// Our custom delimiter*

4 $TmpDir \leftarrow$ "/user/tmp/folder/; *// user-defined temporary folder*

5 $maxPDBSize \leftarrow 0$

6 **for** *each* $\langle k, v \rangle \in P'$ **do**

7 $pdbID \leftarrow k$;

8 $pdbFileName \leftarrow PDBID +$ ".pdb";

9 **if** *Local filesystem FileExists(pdbFileName)* **then**

10 remove $TmpDir/pdbFileName$;

11 **end**

12 $pdbData \leftarrow v$;

13 $pdbData \leftarrow$ replace all instances of CD in $pdbData$ with $CRLF$;

14 write $pdbData$ to local file: $TmpDir/pdbFileName$;

15 **if** *is set* $maxPDBSize$ **then**

16 $pdbSize \leftarrow$ sizeOf($TmpDir/pdbFileName$);

17 **if** $pdbSize > maxPDBSize$ **then** { **continue** to next PDB file; } ;

18 **end**

19 $preOUT \leftarrow$ Run user job on $TmpDir/pdbFileName$;

20 **if** *is set postprocProgramPath* **then**

21 $preOUT \rightarrow TmpDir/\_pdbFileName$; *// temp. file for pre-processing output*

22 $preOUT \leftarrow$ Run user post-processing on $TmpDir/\_pdbFileName$;

23 **end**

24 $pdbOUT \leftarrow$ add line numbers to $preOUT$; *// useful for later sorting*

25 echo $pdbOUT$; *// echo pdbOUT to standard I/O stream*

26 **end**

5.4.2 Post-processing of the user analysis job

As discussed in 5.4.1, having generated a set of outputs, it may be necessary to execute a post-processing step, such as selecting specific lines from the outputs. With this in mind, an optional *post-processing* step has been incorporated into the map step μ_P of PDB-Hadoop that allows the user the opportunity to process the output of each job prior to saving to HDFS. This can be set to use a user defined script (or shell command such as *grep*), hence the user may obtain the desired output required for each PDB entry. For example, a user's molecular docking analysis software may produce a report comprising of various information in addition to the docking scores - in this case it would be desirable to extract only itemised scores, leaving out header and other information, and the *post-processing* option enables extraction of such scores prior to deposition of the final report on HDFS.

5.4.3 Running the user analysis job

Preparatory steps for running the user analysis job together with details on PDB-Hadoop parameters can be found in the Appendix.

5.5 Testing and benchmarking PDB-Hadoop

It is important to understand how MapReduce compares with more traditional methods - specifically batch-schedulers. In order to test the PDB-Hadoop framework, which as discussed is built on Hadoop streaming, against a batch-scheduler (Openlava) (Kaplan and Nelson, 1993; IBM / Openlava, 2017), we executed three structural Biology jobs on each platform using the same apparatus (physical cluster). The jobs were also executed on a different Hadoop cluster to explore how configuration affects performance. These jobs comprise two structural analysis jobs which were run on the entirety of the PDB and a molecular docking job on a subset of the PDB (1000 macromolecular structures - their accession identifiers are listed in Appendix (Alnasir, 2018)). We utilised three separate executables to be run in the PDB-Hadoop map step μ_P as well as submission to the batch-scheduler. The three types of jobs used were as follows:

1. A Python script (Artemis), developed for this thesis (Alnasir, 2017a), that produces a report of residues in a PDB file with bonds computed per residue, as well as torsional angles for residues in each chain.

2. An executable (Dihedrals-64), also developed for this thesis (Alnasir, 2017b), natively compiled for Linux which also computes dihedral angles in a PDB file by chain. Dihedrals-64 was developed using Object Orientated programming methodology using Delphi/Pascal.

3. A molecular docking job, utilising AutoDock Vina (Trott and Olson, 2010), to dock to a custom PDB file (specifically a two amino acid residue dipeptide ligand) with entries in the Protein Data Bank. The *post-processing* feature was used to extract docking scores from Vina's generated output.

Description of these computational tasks in structural Biology which have been used to test and benchmark PDB-Hadoop is given in the sub-sections below.

5.5.1 Dihedral (torsional) angles in peptide polymers

Dihedral angles (also referred to as torsional angles) in peptides are angles measured between select atoms along a torsional (rotatable) bond in neighbouring amino acid moieties (also termed residues). They yield important information about the structural conformation of the residues, that is the arrangement of their atoms in 3D space, in the peptide (secondary structure) (Morris et al., 1992). There are three such dihedral angles that can be measured: ϕ, ψ, and ω and are shown in Figure 5.2 below.

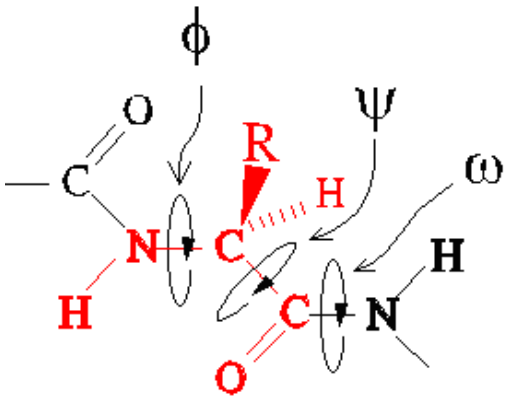


Figure 5.2: Depiction of the ϕ, ψ, and ω. angles in a peptide. The ϕ (Phi) angle is measured from the C of one residue to the C of the next residue. The ψ (Psi) angle is measured from the N of one residue to the N of the next. The ω (Omega) angle is measured from the Cα of one residue to the Cα of the next.

The plot of ϕ (Phi) angle vs. ψ (Psi) on the x and y axis respectively, known as a Ramachandran plot (Kleywegt and Jones, 1996; Zhou et al., 2011), provides useful information about the conformation of the underlying peptide. In this type of plot for all of the residues in a peptide it is typically observed that some residues are in α-Helix conformation others may be in β-strand conformation which is discernible by the clustering of vertices within defined regions. Using these measurements software programs such as ProCheck allow assessment of the stereochemical quality of protein structures (Laskowski et al., 1993).

5.5.2 Molecular docking *in-silico*

Molecular docking is the *in-silico* process of bringing 3D models of molecules together in proximity and computationally simulating their electro-static interactions to elucidate binding modes. It is a key tool in structural molecular Biology and computer-assisted drug design, and typically involves a pair of molecules in which one is designated the ligand and the other the receptor (Morris and Lim-Wilby, 2008; Meng et al., 2011). Commonly the ligand is a small *oligo*-peptide and the receptor is a protein, the models of which are derived from x-ray crystallographic, Nuclear Magnetic Resonance spectroscopy (NMR) and electron microscopy techniques. Figure 5.3 below depicts the oligo-peptide used in the docking tests in benchmarking PDB-Hadoop. The ligand is docked against 1000 molecular structures (receptors) obtained from the Protein Data Bank. It is important to note that the docking of the dipeptide detailed represents a small-scale, toy problem which has been performed on a limited set of structures.

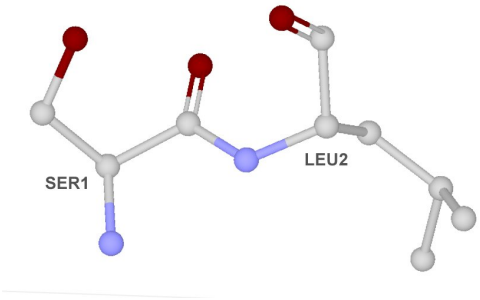


Figure 5.3: The ball and stick depiction of the *in-silico* Ser-Leu oligo-peptide molecule (in this case a dipeptide) used in the docking test and rendered in Zeus molecular viewer (Alnasir, 2010). It consists of, from the direction of the N-terminus to the C-terminus, an L-Serine amino acid moiety and a L-Leucine amino acid moiety, where L- refers to the *levorotatory* amino acids predominantly found in nature, and has a molecular formula of $H_2NCH(CH_2OH)CONHCH(CH_2C(CH_3)_2)COOH$. NB: Hydrogen atoms have been omitted for brevity.

5.5.3 Configuration of the *Bigdata* cluster

The three job types were run on the *Bigdata* cluster, which resides in the department of Computer Science at Royal Holloway university. It has both Hadoop (v2.7.1) and Openlava (v2.2) cluster software installed on the same apparatus. The Hadoop cluster utilises the YARN scheduler and consists of one master node which has a E5-2620 hexa-core CPU @ 2.10 GHz and six slave nodes, each of which has 32 GB of RAM and an Intel Xeon E3-1220v3 4-core CPU @ 3.1GHz. The master node in this setup is configured only to schedule jobs and not to partake in running them directly, YARN was allocated a total of 28 GB/node of RAM and 3 CPU cores/node thereby allowing up to 7 containers of size 4 GB on each node of the cluster, leaving 4 GB of RAM/node and 1 core/node reserved for the OS. The Openlava cluster software was installed on the same cluster nodes.

The Openlava metrics used to measure the load on each node (CPU and memory utilisation) were *r1m* - the 1 minute exponentially averaged CPU run queue length and *pg* - the memory paging rate exponentially averaged over the last minute, in pages per second (IBM / Openlava, 2017). The *run queue length* is defined as *the sum of the number of processes waiting in the run-queue plus the number currently executing.* Lower values indicate less CPU load (IBM, 2014a).

The *r1m run queue length* metric was chosen because *r1m* is averaged over 1 minute, whereas *r15m* is averaged over 15 minutes. *r1m* therefore offers a suitable time resolution, i.e. more instantaneous than *r15m*, which is important given that we will be providing the cluster batches of jobs with a large number of computational tasks (PDB

molecular structures). The types of jobs we will run will present the cluster with a sustained load and so the frequent monitoring of CPU load (using *r1m*) and memory usage (using *pg*) allows the scheduler to manage resources appropriately and be more responsive to changes in the load as the jobs are executed. Openlava was configured to enable the *round-robin* scheduling algorithm across nodes. The cut-off *r1m* load was set at 6 (run queue lengths). This is the load (concurrent processes) at which a compute node will no longer accept jobs causing the round-robin scheduling process to pick the next available compute node that has not reached the load limit. This is to allow the compute nodes to reach saturation (i.e. full usage of CPU and memory resources), and prevent a given compute node from exceeding this load. As each compute node has a Quad-core CPU, the *r1m* value of 6 allows for 2 processes to placed in the run queue per CPU core, where 3 cores are used for jobs and 1 core is reserved for the OS.

5.6 Results

5.6.1 Benchmarking - BatchScheduler and Hadoop clusters

Batch scheduler (Openlava)						
Job Type	**Scale**	**Run 1 (mins)**	**Run 2 (mins)**	**Run 3 (mins)**	**Average (mins)**	**S.D.**
Artemis	entire PDB	47.15	49.38	45.62	**47.38**	**1.89**
Dihedrals	entire PDB	494.89	500.32	499.77	**498.32**	**2.99**
Docking	x1000 PDBs	38.07	32.38	34.92	**35.12**	**2.85**
PDB-Hadoop						
Job Type	**Scale**	**Run 1 (mins)**	**Run 2 (mins)**	**Run 3 (mins)**	**Average (mins)**	**S.D.**
Artemis	entire PDB	37.27	36.45	37.80	**37.17**	**0.68**
Dihedrals	entire PDB	308.31	334.88	325.62	**322.93**	**23.72**
Docking	x1000 PDBs	31.70	31.80	31.45	**31.65**	**0.18**
					Artemis	**21.55%**
				%Speedup:	Dihedrals	**34.20%**
					Docking	**9.88%**

Table 5.2: Summary of jobs used and times taken (in minutes) for comparison. Benchmark times are also averaged over three runs per job type, and the standard deviation between these run times is given.

We note the difference between the performance times of these jobs using Openlava vs. Hadoop on the *Bigdata* cluster. Sample output for the computation of dihedral angles and molecular docking jobs is shown in the Appendix (Alnasir, 2018).

In our investigation, the variability between the job runs is of less significance than that between the different platforms used. Therefore, each job type was run 3 times to determine the average job run time.

There is little data on how many runs are performed in the literature for the type of distributed applications in Bioinformatics that we have reviewed. The existing research projects and structural Biology applications that we have discussed in 5.1, do not specify how many runs were performed when assessing the running time of their systems implemented for Hadoop, with the exception of the protein clustering study by Paschina et al. (2015), which repeated runs 10 times (the longest Hadoop job duration in that study was only 312.50 seconds).

As with some of the other studies we discussed in section 5.1, the job durations are significantly longer - our longest Hadoop job (Dihedrals) was 470.48 minutes (>7.5 hours), and the longest Openlava job was 498.32 minutes (>8 hours). As the runs we have performed are significantly longer, hence their measurement need only be correct to minutes.

5.6.2 Batch-scheduler submission and the *batch-effect*

Openlava Artemis Jobs submitted as a Batch				
# of PDBs per batch job	**Run 1 (secs)**	**Run 2 (secs)**	**Run 3 (secs)**	**Average (secs)**
100	24	24	24	**24.0**
200	49	48	48	**48.33**
400	99	99	99	**99.0**
800	198	198	198	**198.0**
Openlava Artemis Jobs submitted as a Job Array				
# of PDBs per batch job	**Run 1 (secs)**	**Run 2 (secs)**	**Run 3 (secs)**	**Average (secs)**
100	32	32	32	**32.0**
200	66	66	66	**66.0**
400	132	133	133	**132.66**
800	266	268	266	**266.66**
Openlava Artemis Jobs submitted as 1 job per file				
# of PDBs	**Run 1 (secs)**	**Run 2 (secs)**	**Run 3 (secs)**	**Average (secs)**
100	32	32	32	**32.0**
200	68	67	67	**67.33**
400	132	132	133	**132.33**
800	266	266	266	**266.0**

Table 5.3: Time taken (in seconds) to complete Openlava Artemis Jobs on PDB files when these are submitted as a batch, as a Job array, or as individual jobs for comparison.

In order to verify if there was a difference in job run-time (due to overheads in scheduling) between different methods of submitting jobs to Openlava, we submitted Artemis jobs on the same sets of PDB files using three methods: i) as a single batch, where the files in a single folder comprise a single Openlava job ii) as a Job array, where the input files are sequentially numbered and submitted as a single command, and iii) as individual jobs, where 1 job is created per file. The set of PDB files were the first 100, 200, 400 and 800 PDB macromolecular structures from the list of accession numbers given in the Appendix.

In the comparison between the Hadoop and Openlava batch-scheduler platforms (reported in chp5benchmarking) the Artemis and Dihedrals-64 runs, which encompass the entire PDB, were submitted as batches. As the Vina docking job operated on 1000 structures, it was convenient to submit these as individual jobs. This is discussed further, later in 5.7.1.

Table 5.3 shows the run times for submitting the files using the different submission methods described above. We observe that in submitting the same sets of PDB files as individual jobs the overall running time was, on average and in all cases, slower

than submitting the same set of PDB files as a single batch. Surprisingly, the timings for the Job array submission method are almost identical to those of individual job submissions (and not of batch submission, as we expected). This is shown in Figure 5.4, below:

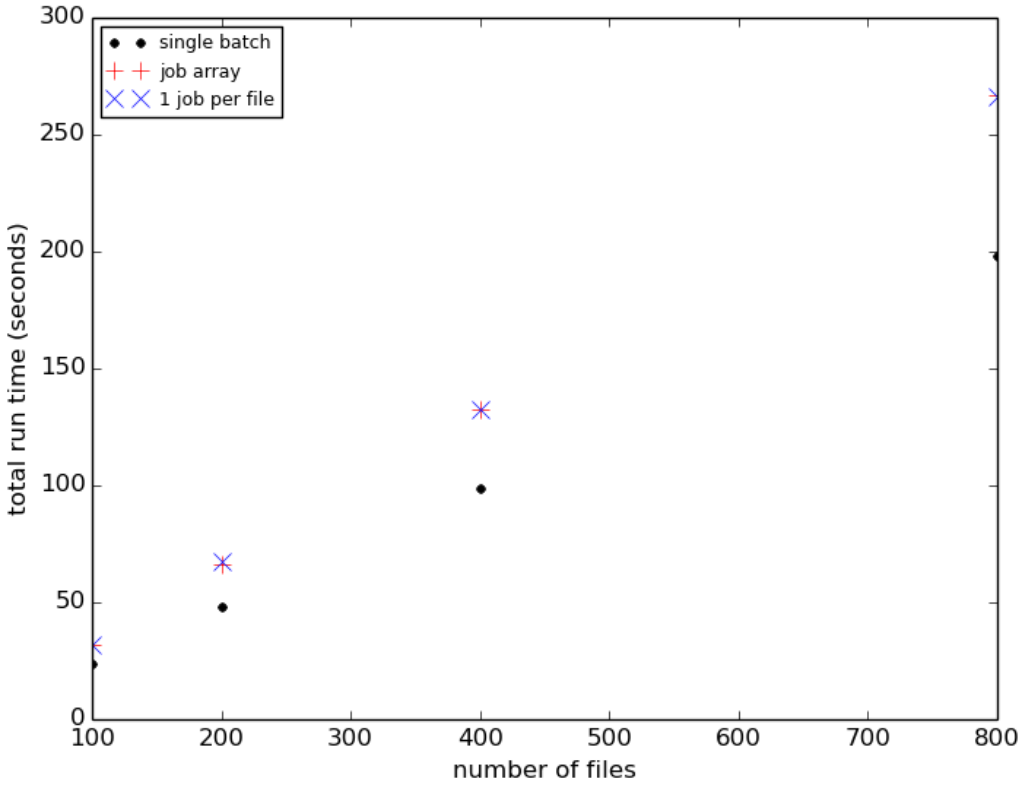


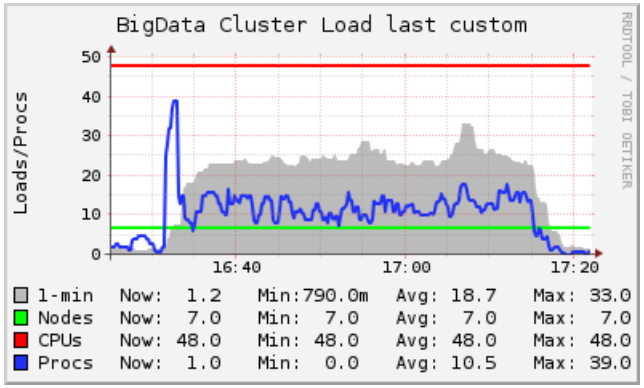
Figure 5.4: Time taken (in seconds) to complete Openlava Artemis Jobs on PDB files when these are submitted as a single batch, as a Job array, or as individual jobs for comparison.

For 100 files, submitted individually this took 32s vs. 24s (as a batch), for 200 this took 67.33s vs. 48.33 (as a batch), for 400 this took 132.33 vs. 99.0s (as a batch), and for 800 it was found to take 226s vs. 198s (as a batch). This corresponds to a difference of 8.0, 19.0, 33.3, 28s, for the different job submission methods for 100, 200, 400 and 800 PDB files, respectively. We will refer to this positive effect on performance of submitting jobs to Openlava, as a batch, as the *batch effect*.

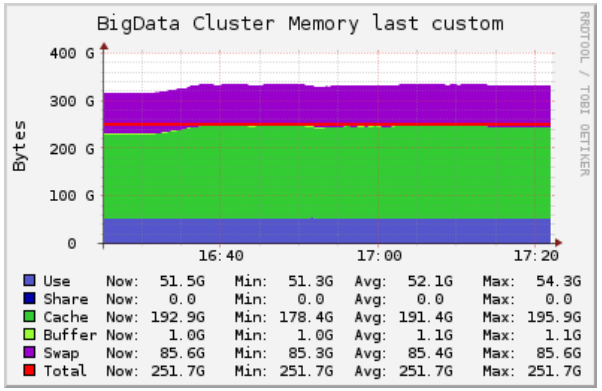
5.6.3 PDB-Hadoop vs. Batch-scheduler - Cluster load metrics

A visual representations of *Bigdata* cluster load metrics for the first run in each job type on both the Openlava batchscheduler and Hadoop platforms is given in Figures 5.4 - 5.9 for comparison of the two platforms (we expect similar load and memory usage on *New-Bigdata*). These have been obtained using the Ganglia cluster monitoring tool (Massie et al., 2004), which was also used by Ellingson and Baudry (2011) at Oak Ridge National Laboratory in Tennessee, USA, in their research on molecular docking with Hadoop. The results are organised in Figures a), b), and c) for each job type on each platform, and these depict cluster load, cluster memory usage and network traffic (for all nodes), respectively. Further details in these Figures are defined in the Figure legends.

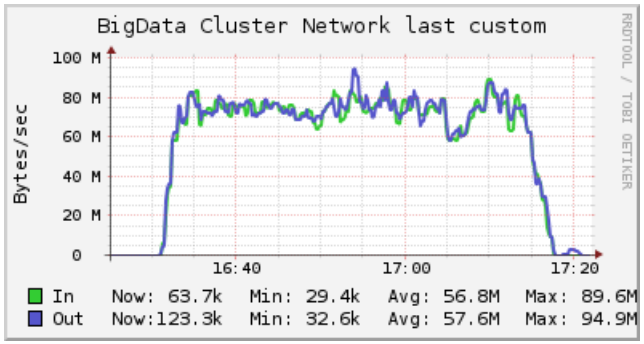
Batchscheduler (Openlava) running Artemis



(a) Cluster overall load



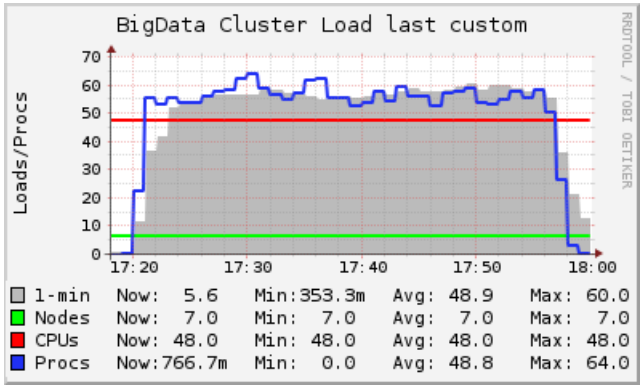
(b) Cluster memory usage



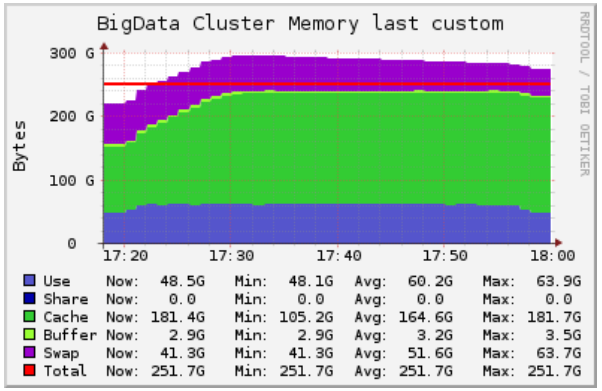
(c) Cluster network usage

Figure 5.5: Cluster load as measured by the Ganglia cluster monitoring tool for Artemis running using Openlava. (a) depicts the CPU load (as number of processes running), where the Blue trace line shows the processes running, the Red line denotes the number of CPU cores available, and the Green line denotes the number of nodes present. (b) depicts current, minimum, average and maximum memory usage in GB. (c) depicts the total cluster network traffic in Bytes/sec, where the Green trace line shows total of the network interfaces input Bytes/sec and Blue trace line shows total of the network interfaces output Bytes/sec.

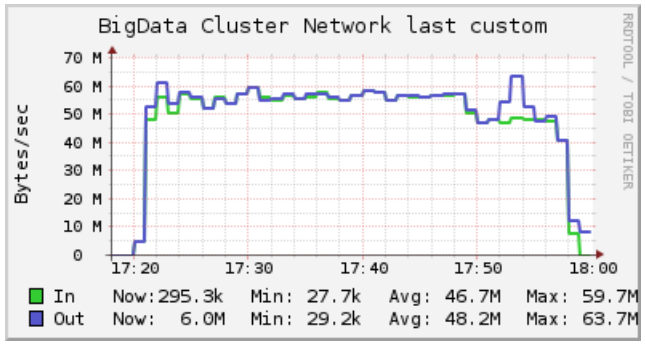
PDB-Hadoop running Artemis



(a) Cluster overall load



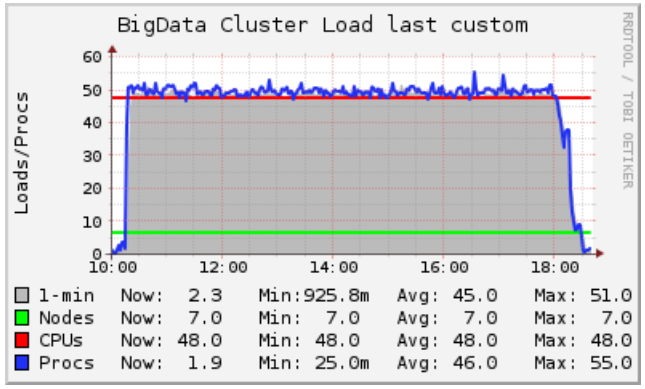
(b) Cluster memory usage



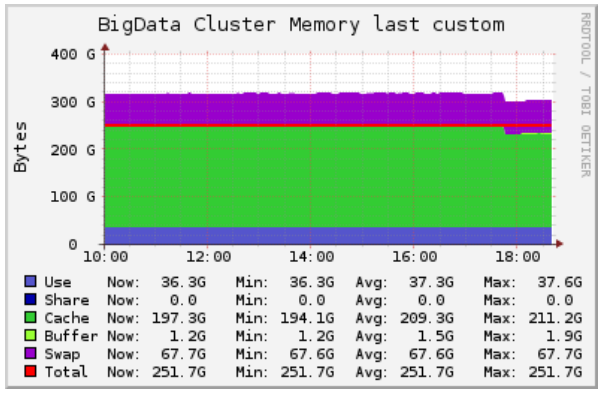
(c) Cluster network usage

Figure 5.6: Cluster load as measured by the Ganglia cluster monitoring tool for Artemis running using PDB-Hadoop. (a) depicts the CPU load (as number of processes running), where the Blue trace line shows the processes running, the Red line denotes the number of CPU cores available, and the Green line denotes the number of nodes present. (b) depicts current, minimum, average and maximum memory usage in GB. (c) depicts the total cluster network traffic in Bytes/sec, where the Green trace line shows total of the network interfaces input Bytes/sec and Blue trace line shows total of the network interfaces output Bytes/sec.

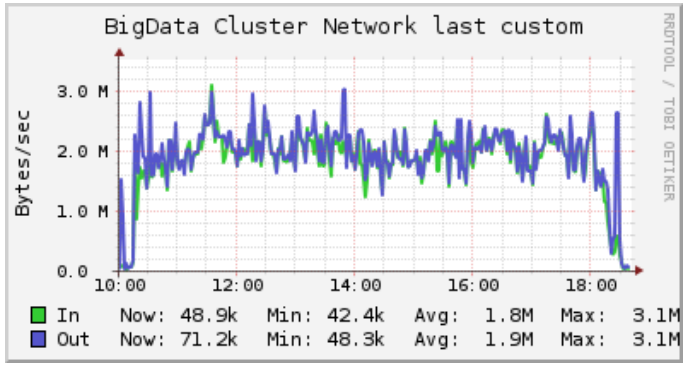
Batchscheduler (Openlava) running Dihedrals-64



(a) Cluster overall load



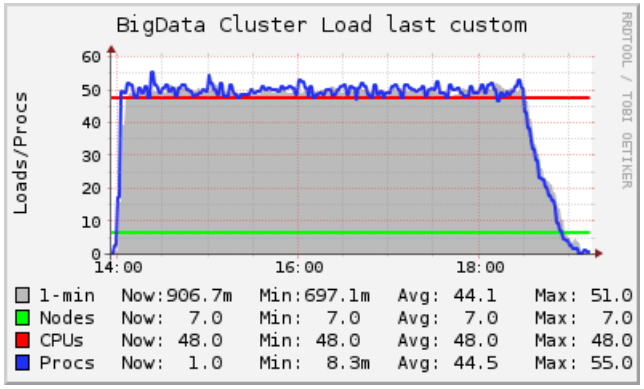
(b) Cluster memory usage



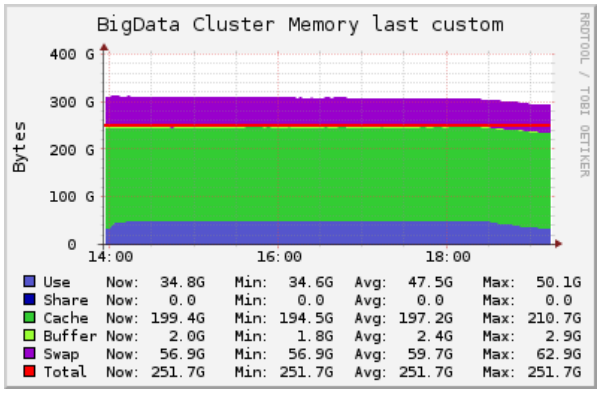
(c) Cluster network usage

Figure 5.7: Cluster load as measured by the Ganglia cluster monitoring tool for Dihedrals-64 running using Openlava. (a) depicts the CPU load (as number of processes running), where the Blue trace line shows the processes running, the Red line denotes the number of CPU cores available, and the Green line denotes the number of nodes present. (b) depicts current, minimum, average and maximum memory usage in GB. (c) depicts the total cluster network traffic in Bytes/sec, where the Green trace line shows total of the network interfaces input Bytes/sec and Blue trace line shows total of the network interfaces output Bytes/sec.

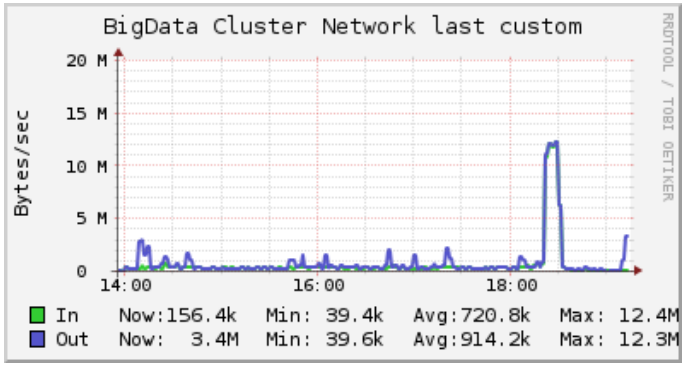
PDB-Hadoop running Dihedrals-64



(a) Cluster overall load



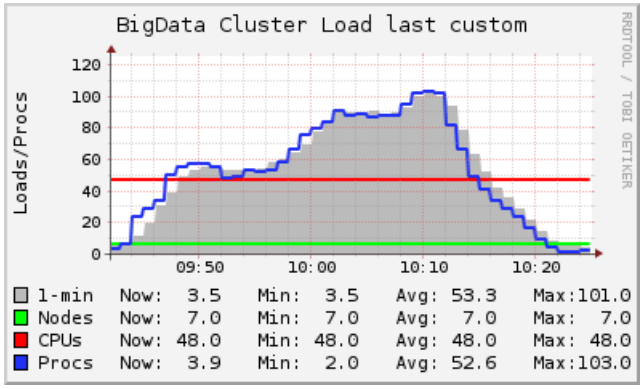
(b) Cluster memory usage



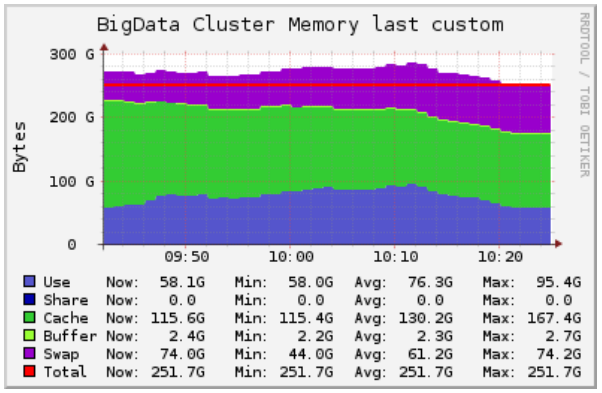
(c) Cluster network usage

Figure 5.8: Cluster load as measured by the Ganglia cluster monitoring tool for Dihedrals-64 running using PDB-Hadoop. (a) depicts the CPU load (as number of processes running), where the Blue trace line shows the processes running, the Red line denotes the number of CPU cores available, and the Green line denotes the number of nodes present. (b) depicts current, minimum, average and maximum memory usage in GB. (c) depicts the total cluster network traffic in Bytes/sec, where the Green trace line shows total of the network interfaces input Bytes/sec and Blue trace line shows total of the network interfaces output Bytes/sec.

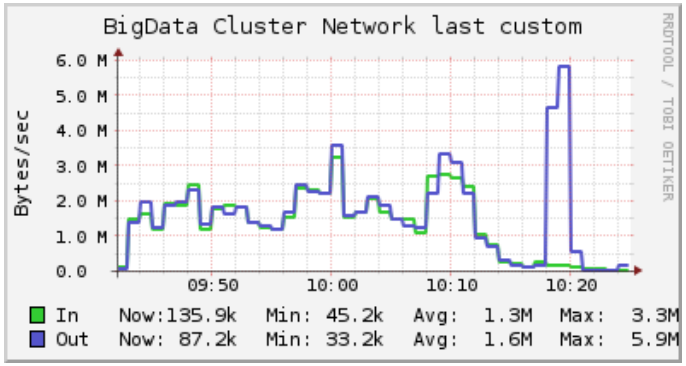
Batchscheduler (Openlava) running AutoDock Vina



(a) Cluster overall load



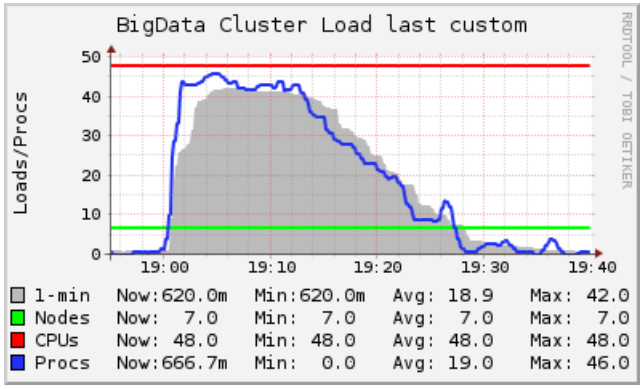
(b) Cluster memory usage



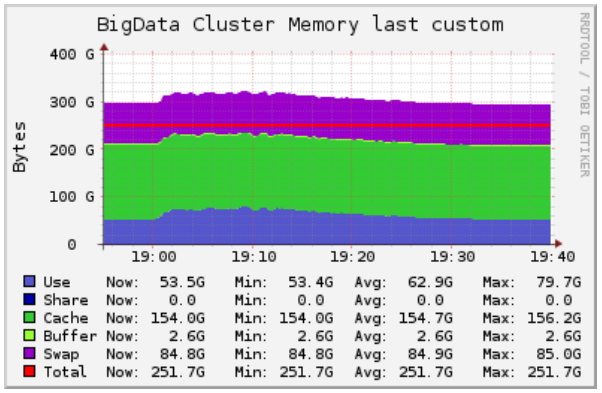
(c) Cluster network usage

Figure 5.9: Cluster load as measured by the Ganglia cluster monitoring tool for Autodock Vina running using Openlava. (a) depicts the CPU load (as number of processes running), where the Blue trace line shows the processes running, the Red line denotes the number of CPU cores available, and the Green line denotes the number of nodes present. (b) depicts current, minimum, average and maximum memory usage in GB. (c) depicts the total cluster network traffic in Bytes/sec, where the Green trace line shows total of the network interfaces input Bytes/sec and Blue trace line shows total of the network interfaces output Bytes/sec.

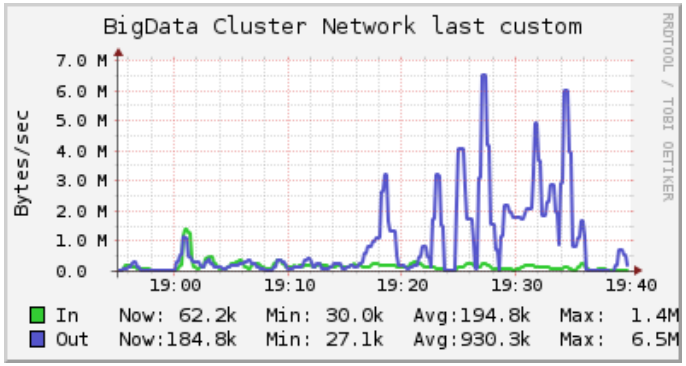
PDB-Hadoop running AutoDock Vina



(a) Cluster overall load



(b) Cluster memory usage



(c) Cluster network usage

Figure 5.10: Cluster load as measured by the Ganglia cluster monitoring tool for AutoDock Vina running using PDB-Hadoop. (a) depicts the CPU load (as number of processes running), where the Blue trace line shows the processes running, the Red line denotes the number of CPU cores available, and the Green line denotes the number of nodes present. (b) depicts current, minimum, average and maximum memory usage in GB. (c) depicts the total cluster network traffic in Bytes/sec, where the Green trace line shows total of the network interfaces input Bytes/sec and Blue trace line shows total of the network interfaces output Bytes/sec.

5.6.3.1 Artemis comparison

In running the two Artemis jobs, the Ganglia cluster monitoring tool shows that the overall cluster load is distributed evenly across the duration of the jobs in both Openlava (Figure 5.5a) and PDB-Hadoop (Figure 5.6a). However, there is a significant spike in the cluster load in the same job running on Openlava during the first 5 minutes of the job submission, which is not seen in PDB-Hadoop or the other Openlava jobs.

In terms of overall network bandwidth used, for the most part this remains relatively steady across the job durations, at approximately 60 Mb/sec in the Hadoop Artemis job, and slightly higher in the Openlava Artemis job at 65 Mb/sec, although there is a small drop followed by a spike at the end of the Hadoop job. The Openlava job had a lower average number of concurrent processes (10.5, Figure 5.5a) than the PDB-Hadoop job (48.8, Figure 5.6a). Peak swap memory usage, which occurs when memory consumption exceeds boundaries set by the OS on physical ram, as shown by Figures 5.5b and 5.6b, peaked at 85.6 GB (average 85.4 GB) for the Openlava Artemis job as compared to 63.7 GB (average 51.6 GB) in the PDB-Hadoop Artemis job, indicating Hadoop makes more efficient use of memory.

5.6.3.2 Dihedrals comparison

In observing the metrics in the two Dihedrals jobs (Figures 5.7a and 5.8b), we observe steady loads on the cluster across the duration of the two jobs. There are, however, a few spikes across the duration of the Openlava job, notably at the end of the job. The Openlava job had a higher average number of concurrent processes (46, Figure 5.7a) than the PDB-Hadoop job (44.5, Figure 5.8a). Peak swap memory usage, as shown by Figures 5.7b and 5.8b, peaked at 67.7 GB (average 67.6 GB) for the Openlava Dihedrals job as compared to 62.9 GB (average 59.7 GB) in the PDB-Hadoop Dihedrals job.

5.6.3.3 Vina comparison

In the two Autodock Vina docking jobs (Figures 5.9a and 5.10a), we observed an approximately peak-shaped distribution of load, with the apex of the peak occurring at the end of the docking job running with Openlava (at approx. 30 mins), but at the start of the job with PDB-Hadoop (at job start, and lasting 13 minutes into the job). Because the cluster CPU load of these two jobs was not distributed evenly across the job durations, we observed that in the Openlava job, the average number of concurrent processes was 52.6 but peaked at 103, and in the corresponding PDB-Hadoop job was, on average 19, but peaked at 46. Figures 5.9b and 5.10b show that swap memory usage peaked at 74.2 GB (average 61.2 GB) for the Openlava docking job as compared to

85.0 GB (average 84.9 GB) in the PDB-Hadoop docking job.

5.7 Discussion

As discussed in section 5.5, we ran and monitored three types of Structural Biology jobs on both the batch-scheduled (Openlava) and Hadoop platforms (using our PDB-Hadoop framework). The jobs consist of two structural analysis jobs, and a molecular docking job on a subset of the Protein Data Bank (1000 macromolecular structures). Table 5.2 lists the time taken for the jobs to complete (in minutes), which we will analyse and discuss in the next sub-sections, after first discussing the optimisation and configuration of the *Bigdata* cluster apparatus, which hosts both Openlava and Hadoop (for general comparison).

5.7.1 Optimisation of the Hadoop and Openlava platforms on *Bigdata* cluster

Attempts were made to optimise the performance of Openlava as well as Hadoop on the *Bigdata* cluster apparatus. So as to ensure the fairest comparison, Openlava was configured to add a specific *priority* job queue which utilises round-robin scheduling, with appropriate load indices, and both cluster platforms were configured to allocate the maximum resources to running the jobs, whilst also reserving sufficient resources for the operating system. Configuration of the *Bigdata* cluster apparatus used for this comparison is discussed in section 5.5.3.

Furthermore, the benchmarking runs were scheduled and carried out during times when the cluster was not heavily loaded. There is an overhead in submitting each job to a batch-scheduler due to the method of scheduling discussed above. Submitting a folder with a large number of files, where each file constitutes a separate job, would result in substantial overheads and increases in the overall job time. Submitting jobs to Openlava, where each file is an individual job, would result in an unfair comparison.

For this reason, the Artemis and Dihedrals jobs that analyse the entirety of the PDB and contain 128,330 files (as of March 2017), were submitted to Openlava in batches, and each batch comprised approximately 100 PDB entries. The rational for this was that when we extracted the entire PDB archive to local disk, the files were arranged into 1060 sub-folders, each containing approximately 100 PDB files. Therefore, when all of the files in the PDB are processed folder-by-folder, this results in 1060 jobs, where each job consists of a single script which processes a sub-folder as a single job. The vina docking job, however, contains less files to process, and was therefore submitted to the batch-scheduler as each file per job (for 1000 molecular structures).

In order for the Openlava batch-scheduler to be able to process these jobs consecutively without delay it was necessary to set related parameters in the *lsb.params*

configuration file. These were: the interval for dispatching jobs by master batch daemon (`MBD_SLEEP_TIME`, default 20 seconds) to 2 seconds, the interval for checking jobs by slave batch daemon (`SBD_SLEEP_TIME`, default 15 seconds) to 10 seconds, and the interval for a host to accept two batch jobs subsequently (`JOB_ACCEPT_INTERVAL`, default to equal `MBD_SLEEP_TIME` seconds) to 1 second.

5.7.2 Analysis of PDB-Hadoop and Openlava benchmarking results

We shall now discuss and present analysis of the job types. We will discuss the Artemis and Dihedrals job types together, because they involve the same type of computation but with different programs.

5.7.2.1 Artemis and Dihedrals jobs for computing torsional angles

We observed, on average, a speed-up of 21.55% in running the Artemis job, to compute dihedral angles, using PDB-Hadoop compared to using the Openlava batch-scheduler to run the same job (Table 5.2). In running the Dihedrals job to perform the same task, we observed, on average, a speed-up of 34.20% when using PDB-Hadoop compared to using the Openlava batch-scheduler to run the same job.

We note that the Artemis and Dihedrals jobs per PDB are typically very short - the analysis of a PDB (accession: `3HPV`, 841.2 Kb, *Crystal structure and functional analysis of the extradiol dioxygenase LapB from a long-chain alkylphenol degradation pathway in Pseudomonas*, Cho et al. (2009)) took 7.89s for Dihedrals-64, and only 1.21s for Artemis to complete.

In both job types these performance gains can be explained by the number of concurrent processes as well as the peak swap memory usage. Generally, the higher the number of concurrent process across a set of CPUs, the higher the throughput achievable. However, there are exceptions to this which result in lower throughput despite a higher number of concurrent processes: i.e. when each process consumes an excess of memory than is physically present, requiring swap memory operations[3](Red Hat, 2017), or when the number of processes is too high requiring frequent interleaving (a CPU having too many associated processes and having to switch between them too frequently).

The Ganglia cluster monitoring tool shows that both the Artemis and Dihedrals Openlava jobs (Figures 5.5a/b and 5.7a/b) had a lower average number of concurrent processes and higher peak swap memory usage than the corresponding jobs on PDB-Hadoop (Figures 5.6a/b and 5.8a/b). For the processes, this was measured to be 10.5

[3]Solid state disks can help mitigate this, by removing the need to access a spinning disk platter, though there are still disk I/O operations involved which take longer than accessing RAM directly.

vs. 48.8 for the Artemis job on Openlava and PDB-Hadoop, respectively, and 46.0 vs. 48.8 for the Dihedrals job on Openlava and PDB-Hadoop, respectively. For the average swap memory usage, this was 85.4 GB vs. 51.6 GB for the Artemis job on Openlava and PDB-Hadoop, respectively, and 67.6 GB vs. 59.7 GB for the Dihedrals job on Openlava and PDB-Hadoop, respectively. This indicates that, for the large number of computations that these jobs consist of, Hadoop is more efficient in managing load and associated memory usage than the batch-scheduler.

From the results we can see that the computation of dihedral angles using a natively compiled Linux program (Dihedrals-64), to perform the same calculation as Artemis python script, was much slower, on average, on both platforms. The job running time on Openlava was 494.88 min vs. 47.38 min for Dihedrals-64 and Artemis, respectively, and on PDB-Hadoop it was 322.93 min vs. 37.17 min for Dihedrals-64 and Artemis, respectively.

The difference between the average run times of the Artemis and Dihedrals jobs (on both platforms) is likely because Dihedrals-64 was part of a larger software package and hence not optimised, even though it was natively compiled into Linux from Object Pascal (using the Free Pascal compiler (FreePascal, 2017)). The source code-base for the Linux version of Dihedrals-64 was the much larger Zeus molecular visualisation software developed for Microsoft Windows with the visualisation code had been removed prior to compiling. Hence Dihedrals-64's object orientated code was not written specifically for computing torsional angles, but was a feature of the larger package which requires the whole molecular model first to be loaded into memory, and then individual residue objects instantiated. Artemis, however, was pre-compiled into bytecode (*.pyc* format) and written specifically for iterating PDB records to compute torsional angles. Furthermore, differences in the way reading from disk is handled by objects in the two languages (Delphi vs. Python) will also contribute to the differences in run times.

5.7.2.2 AutoDock Vina job for molecular docking

A speed-up of almost 10% (9.88%) was achieved in using PDB-Hadoop for *in-silico* molecular docking using AutoDock Vina compared to using Openlava. The two Autodock Vina docking jobs are more computationally demanding than the computation of dihedral angles because they require extensive exploration of conformational space (3D spatial arrangement of the molecules) when docking the ligand to the target protein in each task. The docking operation searches for the conformation with the lowest energy - as an indicator of the most energetically stable conformation - and therefore the time it takes to reach this state varies for each protein-ligand pair processed. As a result, we have observed that the cluster load reach high peaks in both jobs, and the load

is not distributed uniformly across the duration of each job (Figures 5.9a and 5.10a). There is an approximately peak-shaped distribution of load, with the apex of the peak occurring at the end of the docking job running with Openlava (Figure 5.9a, at approx. 30 mins), but at the start of the job with PDB-Hadoop (Figure 5.10a, at job start, and lasting 13 minutes into the job).

In the Openlava job, the average number of concurrent processes was 52.6 but peaked at 103 (Figure 5.9a), and in the corresponding PDB-Hadoop job was, on average 19, but peaked at 46 (Figure 5.10a). Peak swap memory usage, as shown by Figures 5.9b and 5.10b, peaked at 74.2 GB (average 61.2 GB) for the Openlava docking job as compared to 85.0 GB (average 84.9 GB) in the PDB-Hadoop docking job, respectively.

This difference can be explained by the order in which the protein-ligand pairs is processed being slightly different in the PDB-Hadoop and Openlava jobs. The PDB-Hadoop job iterates the list of 1000 macromolecular target proteins as splits which are a function of the HDFS block size, whereas the batch-scheduled job uses the linux *find* command to iterate over a folder containing the same macromolecular structures PDB files. These files are submitted to Openlava individually[4], and therefore each file is submitted to the least loaded host and waits in the queue prior to being despatched to a suitable slave host machine. As shown in 5.6.2 there is a significant performance hit from reading individual files. These differences mean that docking jobs that take longer than average peak at different times of the job run.

An interesting occurrence in regard to the utilisation of cluster resources in this job type using PDB-Hadoop was that a number of docking jobs utilising Autodock Vina failed during the testing process. This was found to be because Vina implements its own support for concurrency through the use of multi-threading (Trott and Olson, 2010), which was conflicting with Hadoop. The situation in which each Vina thread reserved its own memory, unaware of the multiple instances spawned by PDB-Hadoop, led to memory depletion and job failure. Disabling Vina's multi-threading in the PDB-Hadoop job resolved this issue by means of delegating the concurrency to MapReduce and YARN. A more extensive test of docking using Autodock Vina was performed with PDB-Hadoop. This involved docking the above-mentioned putative oligo-peptide ligand against the entire Protein Data Bank (excluding pdb files > 2 Mb). The entire job was completed in less than 31 hours (single run) which indicates its robustness. This robustness is also supported by the replication factor of data on HDFS.

[4]NB: Only the Artemis and Dihedrals jobs were submitted in batches of approximately 100 PDB files

5.7.2.3 Resource management and scheduling in Hadoop vs. Batch-scheduled cluster computing

The performance increases for the Artemis, Dihedrals and Docking jobs, observed when using PDB-Hadoop over Openlava, are primarily the result of the *batch-effect* on Openlava (discussed in 5.6.2), which is dependent on how the two platforms manage concurrency when scheduling the jobs. Differences in data locality may also be contributing to the job run-times, but these are not reflected in the ganglia network bandwidth usage for the two platforms. Concurrency and data access are managed differently on the two platforms, as we will now discuss in this section.

On both platforms, concurrency is dependent on cluster configuration and load. The difference, however, is in how the two platforms process and schedule the jobs on submission. On Openlava the different ways in which jobs that comprise large numbers of files are submitted results in a *batch-effect*. On YARN, however, this is does not occur because the job input data is partitioned into splits, the size of which is optimised based on the cluster configuration (discussed in chapter 2 section 2.4.4). On Hadoop, at job submission, the user has little control on how many splits are generated for map steps, as this is decided by YARN.

YARN computes the optimum number of containers, to spawn across nodes of the cluster simultaneously, using a number of variables. The key variables in this calculation are the input data size (containing the PDB entries P'), the number of splits the input data will be partitioned into (determined by the the HDFS block size, `dfs.block.size` in *hdfs-site.xml*), and the number of containers that YARN can run on each node (a function of `yarn.nodemanager.resource.memory-mb`, and `mapreduce.map.memory.mb` in *yarn-site.xml* and *mapred-site.xml*, respectively).

In Hadoop, the maximum number of simultaneously spawned containers (running mappers in PDB-Hadoop), therefore, is a product of the maximum memory allocated to YARN for containers on each node and the number of nodes (because each node in the *Bigdata* Hadoop cluster configuration is homogeneous and has the same configuration). When the number of splits exceeds the limit of maximum concurrent containers, the maximum number of concurrent containers will run concurrently, and when one of these containers has completed processing a waiting container starts, keeping CPU and memory resources fully saturated. When the number of splits is less than the limit of maximum concurrent containers, the number of concurrent containers will match the split count.

Openlava, however, uses job queues along with used-defined load metrics and scheduling policies (in *lsb.queues* and *lsb.hosts*) to determine which host to schedule jobs to from the waiting queues (IBM / Openlava, 2017). This means when jobs are submitted

they are held in a job queue with a status of `PENDING` until a suitable slave host machine is available, the job then runs and its status becomes `RUNNING`. This delay on Openlava is also influenced by the way in which jobs are submitted - as we observe in Table 5.3 there is a batch-effect which increases this delay when PDB files are submitted ss individual jobs. This effect occurs, even when the jobs are submitted as a Job array, that is as a single job comprising of individual files specified by an index range (see LSF documentation: IBM (2014b)).

The greater utilisation of CPU and memory utilisation of YARN is also clearly illustrated in the ganglia generated graphs of cluster load for the Artemis job running on the two platforms - the PDB-Hadoop load (Figure 5.6a) is greater, and of shorter duration than that the Openlava load (Figure 5.5a). The *Bigdata* cluster, in running the Openlava Artemis job, experiences an average load average of 10.5 over approximately 47 minutes, with a peak of 39 at the start of the job. The Hadoop Artemis job, running on the same apparatus experiences a load average of 48.8 with a peak of 64 over approximately 37 minutes.

This batch-effect is smaller in the Vina docking job simply because there are less files in the analysis and hence we still see a smaller difference (9.88%) between the run times. Ganglia shows a substantial difference in terms of how the runs are deployed on Openlava and Hadoop (probably because of a different order in the way the PDB files are analysed). On the other hand, there is a difference in the network communications (i.e. Hadoop has a smaller level of overall network communication during the job) which can explain the remaining difference.

A significant difference between the two platforms, is that Openlava does not possess the data-locality inherent in MapReduce on the distributed HDFS filesystem (discussed in chapter 2 section 2.4.3). This reduces job execution time in PDB-Hadoop by bringing the compute to the data (Hadoop mappers are executed on the node in which the split can be found on HDFS), as opposed to bringing the data to the compute (Openlava job processes will access the source data through a network NFS share, which involves streaming the data over the network connection).

It is clear that quite substantial differences in performance occur between these systems - even though they are running on the same hardware. This is in spite of the optimisations that we have made to the configuration of the batch-scheduler. These involved modifying the load indices to support high load, as well as minimising the job accept intervals. Contributing to this variance are relatively subtle issues in terms of batch submissions of numbers of files, hence performance may be quite variable depending on the type of application.

5.7.3 PDB-Hadoop's niche

In this chapter we have discussed, in detail, existing systems for structural Biology on Hadoop, a powerful distributed computing platform that is supported by a large number of cloud service providers. With the exception of the Protein Analysis Workbench developed by UCL (Buchan et al., 2013), all of these systems have been developed for the applications of molecular docking, structural alignment and protein clustering (Ellingson and Baudry, 2011; Hung and Hua, 2013; Estrada et al., 2012; Liu et al., 2016; Hung and Hua, 2013). Whilst the Protein Analysis Workbench provides a set of services to perform a range of specific set of tasks, the other applications have been implemented specifically for a particular computational task. PDB-Hadoop has been conceived, implemented and tested to be more flexible. Specifically, the framework allows *any* structural analysis tool to be run using Hadoop so as to exploit the high-throughput, scalable and fault-tolerant capabilities inherent in the Apache Hadoop platform (discussed in chapter 2 section 2.4). PDB-Hadoop facilitates structural biologists to perform analyses on semi-structured data without having to explicitly write their own MapReduce code. It can also be used to run applications on any data set based on a large number of small-sized flat files, for example Fasta files that are commonplace in sequencing applications.

5.8 Conclusion

In this chapter we have demonstrated good performance gains in using PDB-Hadoop to run the three structural Biology job types we have discussed, over batch-scheduled cluster computing - speed-ups of 21.55%, 34.20%, 9.88% were obtained for jobs involving python script (Artemis), a native linux program (Dihedrals) and a natively compiled molecular docking application (AutoDock Vina). These are consistent with the performance gains demonstrated in the applications implemented for Hadoop that we have discussed in section 5.1, and summarised in Table 5.1. This shows that Hadoop is competitive with the traditional batch-scheduled approach to processing structural Biology data.

The improvements are most notable for the cases where the runs are dominated by file access. The Artemis and Dihedrals-64 runs are short (less than a second) per PDB entry and run over the entirety of the PDB. This improvement is due to the batch effect where YARN optimises the number of contains and ensures the applications are run on the nodes where the data resides.

These findings, together with the scalability, fault tolerance, and importantly the increasing availability and documentation of the Hadoop platform make it an attrac-

tive technology. Although some of the applications we have discussed confirmed good reproducibility when existing non-Hadoop implementations were compared with their Hadoop implementations, the adoption of Hadoop for applications in structural Biology, however, requires non-trivial development using MapReduce. We observe these complex and intricate implementations of MapReduce in projects such as the hierarchical protein clustering method employed by Estrada et al. (2012) for clustering conformers in protein-ligand ensembles, and the clustering method developed by Paschina et al. (2015) to search for conformations from trajectories produced from molecular dynamics simulations. One method of overcoming this complexity in developing applications using MapReduce has emerged to exploit the power of Hadoop - employing a simple map step which is trivial in that it encapsulates simple execution of a computational task. This method has been used in the Cloud-PLBS implemented by Hung and Hua (2013), the structural alignment application implemented by Liu et al. (2016) using existing structural alignment tools, and the large scale docking experiment using the DUD dataset by Ellingson and Baudry (2011).

These implementations, however, are limited to performing a specific computational task. PDB-Hadoop utilises a map step in the same way, but overcomes limitations of the aforementioned applications by allowing the user to specify *any* structural Biology software that they require to run. It also allows the *post-processing* of results generated by the users to tool, and can be daisy-chained to other map or reduce steps in the same job. The PDB-Hadoop software is also compact, comprising of a handful of bash scripts, and is easy to deploy to Hadoop supported cloud services, such as as AWS or Elastic MapReduce. It can therefore can be integrated within larger analysis pipelines, a common practice in bioinformatics.

In the next chapter we will make extensive use of explicit MapReduce on Spark for the analysis of short read Next-generation sequencing transcriptomics data, specifically for the quantification of bias in such datasets.

Chapter 6

Bias detection in NGS data using sequence motifs in exons

"Not everything that can be counted counts, and not everything that counts can be counted." – Albert Einstein

As discussed in chapter 3 (latter part of section 3.3, as well as sections 3.4 and 3.6) biases due to sequence-specific motifs are an issue in microarray data (Memon et al., 2010; Upton et al., 2008), have also been shown to affect RNA-Seq data (Zheng et al., 2011; Risso et al., 2011) and RNA primers (Hansen et al., 2010). This manifests itself as sequence-specific deviations in the distribution of mapped reads to a reference genome (Roberts et al., 2011; Li et al., 2010) and predictable dinucleotide frequencies (Zheng et al., 2011). Furthermore GC content effects have been demonstrated in both Microarray, Illumina Genome Analyser and RNA-Seq data (Benjamini and Speed, 2012; Risso et al., 2011). Therefore a method that can quantify these effects by way of deep, transcript analysis is necessary. In this chapter we will describe a novel analysis method, based on analysing sequence motif correlations, that employs MapReduce to quantify bias in Next-generation sequencing (NGS) data at the exon level. We will look at the input data formats used and explain the implementation of the method, which comprises of two phases, described below:

i a distributed phase capable of handling high-throughput transcriptomics datasets that yields motif count data for reads of all exons in the dataset.

ii a non-distributed motif counts analysis phase that quantifies sequence-specific deviations in the distribution of mapped reads by computing correlations of the motif counts computed by the distributed first phase.

Firstly, we will test and validate our analysis method for quantifying sequence-specific deviations in the read distribution of mapped reads, by using an artificial dataset. This dataset consists of an artificial GTF annotation file of artificial chromosomes and a SAM reads file of artificial reads. We shall then present the results of our analysis of three samples from *H. sapiens* that were produced in a controlled way using IVT (in vitro transcription), and by applying different library preparation protocols to each sample during RNA-Seq. These samples from Lahens et al. (2014) are known to demonstrate *intra-exon* coverage bias (we have used their *H. sapiens* data). Although the raw data deposited in GEO for this dataset has not been aligned, the library preparation protocol for each sample, and alignment and post-processing strategy have been clearly documented. We shall also analyse the complete transcriptomes of two species of *D. melanogaster* (wild and mutant types, each having two replicates). Samples from these data sets are from the Drosophila species and were chosen because its genome and transcriptome are extremely well annotated.

Likewise, the Drosophila samples are transcriptomic data sets where the sequence reads data have already been aligned *in-silico* to the reference genome by established research labs working in this field. The appropriate alignment parameters the researchers have chosen and the protocol steps applied are also documented in section 6.1.3. Finally we discuss the results in terms of the parameters that we have examined such as the mean read GC content for the exon, Motif GC content, motif-spacing and motif sequence.

The source code for both phases of the analysis described in this chapter, together with results for the 4-mer analysis of the two aforementioned Drosophila datasets, has been archived on Zenodo with a DOI (Digital Object Identifier) (Alnasir, 2017c).

Existing research has mainly focused on the GC content of exons. We will use the average GC content of the reads belonging to a given exon as a proxy for the GC content of that exon $\overline{ge}$. In this research we have additionally investigated the effect of the GC content of *sequence-specific* motifs *gm* on the distribution of reads. This allows for a more thorough investigation at the exon level because we quantify sequence-specific deviations in the distribution of mapped reads as both a function of exon GC content and as a function of GC content of *sequence-specific* motifs. Whilst Zheng et al. (2011) studied dinucleotide frequencies (*2-mer* motifs) in NGS data, we examine correlations of pairs of *intra-exon 4-mer* motifs at various spacings. Having explored a data set chosen specifically to explore *intra-exon* coverage bias, we examine a data set that was gathered to examine a specific Biology. The method employed in this chapter requires the alignment of the short reads.

6.1 Quantifying *sequence-specific* deviations in the read distribution using MapReduce

Our method to quantify sequence-specific deviations in the distribution of reads mapped to a reference genome uses counts of reads overlapping motifs and works at the deep, read level. In order to provide the capacity to process the amounts of data typical in transcriptomic datasets (Stephens et al., 2015) our analysis employs parallel distributed computing algorithms and infrastructure using the Apache Spark platform, discussed in chapter 2, section 2.5.

In order to do this it is necessary to explain gene structure. The exon (depicted in Figure 6.1) is the atomic (i.e. non-reducible) unit of function in the transcriptome (Gilbert, 1978). Exons are expressed biologically in mRNA (messenger RNA) which are translated to peptide sequences or proteins. Exons spliced at different locations, a process termed alternate-splicing, can combine with each other to form different gene isoforms, the transcripts of which are then translated into proteins. Our analysis method applies to exon read sequences, specifically coding sequences (CDS) prior to splicing.

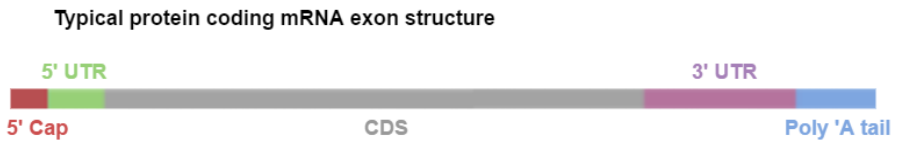


Figure 6.1: Typical mRNA protein coding fragment (Exon), which consists of a 5' cap, 5'UTR (untranslated region), Coding sequence (CDS), 3'UTR (untranslated region) and a Poly-Adenylated tail.

In an ideal transcriptomic data set mapped reads would be of sufficient length to span entire exons, and would therefore be uniformly distributed across an exon (Figure 6.2, part A). However, RNA-Seq often generates short reads, the length of which is dependent on the sequencing platform, and as a result mapped reads are typically not distributed uniformly across exons (Figure 6.2, part B) (Roberts et al., 2011). Furthermore the number of mapped reads is a function of the number of fragments sequenced and the feature length (i.e. length of the exon), for this reason a number of normalisation methods are used to quantify the number of mapped reads to a feature such as Reads per Kilobase Million (RPKM) (Mortazavi et al., 2008), and Transcripts per million (TPM) (Wagner et al., 2012) - review of normalisation methods for RNA-Seq can be found in research publications by Li et al. (2015) and Dillies et al. (2013). Our method allows us to investigate the distribution of mapped reads in large datasets.

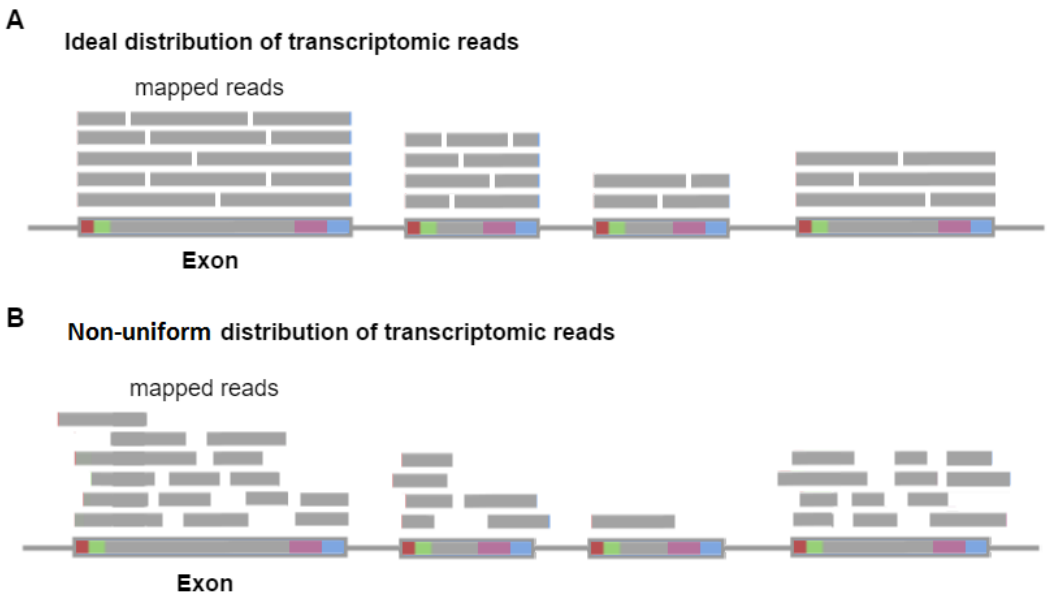


Figure 6.2: Schematic representation of RNA-Seq read distribution across exons. **A)** Ideal distribution of RNA-Seq reads mapped to an exon if the reads were contiguous and of sufficient length. **B)** Non-uniform distribution of RNA-Seq reads mapped to an exon.

As discussed, the coverage of reads across an exon is important in RNA-Seq, particularly in application where quantitative measurement of expression is performed (Ross et al., 2013; Hansen et al., 2010). It has been pointed out that bias in the *intra-exonal* distribution of reads is due to both naturally occurring splicing patterns (involving reads that map to multiple locations), and technical error in library selection, and is common in RNA-Seq (Lahens et al., 2014). In this paper, using an in-vitro transcription method, they have demonstrated and quantified coverage bias that exists between samples containing the same set of mRNA transcripts, but that were prepared using different RNA-Seq library selection protocols. Because the exon is the non-reducible transcriptional unit of the gene, and are the components of mRNA transcripts by virtue of splicing, *intra-exon* deviations in the uniformity of distribution of reads across a given exon are of technical origin (i.e. due to the read length of the sequencing platform and library selection methods). These are therefore more problematic than *inter-exon* differences in read coverage (i.e. between different exons), which are likely to be of biological origin.

With this in mind, we have developed a method for quantifying sequence-specific deviations in the distribution of mapped reads across an exon. This is achieved by picking a short sequence motif (typically *4-mers*) which can occur at various positions

within the sequence of the exon. Next, *intra-exon* pairs of these motif occurrences were picked based on their distance apart from each other within the exon and the number of overlapping reads covering each motif position in the pair was counted (Figure 6.3). We chose to examine *intra-exon* motif-pairs because in an ideal transcriptomic data set the counts for each motif in the pair would be identical as reads mapped to the exon under inspection would be uniformly distributed. We term the distance between the motif-pairs *motif-spacing* and we have chosen to examine motif pairs that are spaced at 10, 50, 100 and 200 bp apart.

The motif-pair spacings were selected for a number of reasons. The *H. sapiens* and *D. melanogaster* RNA-Seq datasets have read lengths of 100 and 36 bp, respectively. This range of spacings between occurrences of the motif allows us to explore motif-pairs that are very likely to be on the same short read (10 bp for the D. melanogaster datasets; 50 bp for the human datasets). Therefore, these are likely to show a high correlation compared to motif pairs that ill not be on the same read (200 bp), and hence susceptible to bias.

Uniformity of read distribution was quantified by computing the correlation of the counts for the given motif pair in all exons within the dataset by aggregating the motif pair counts at a given distance apart (motif-spacing) regardless of position within the exon. An ideal dataset would have perfect correlations for motif pairs (for instance +1.0 for the Pearson correlation coefficient method we discuss later in section 6.3.3). In order to thoroughly examine the affect of sequence-specific motifs on uniformity of read distribution we analysed the correlation for all *4-mer* motifs ranging from `AAAA` to `GGGG` (i.e. 4^4 combinations). We will discuss the details on how this method is implemented in MapReduce in subsection 6.1.4 after we first introduce the input data file formats for transcriptomic data in the next subsection.

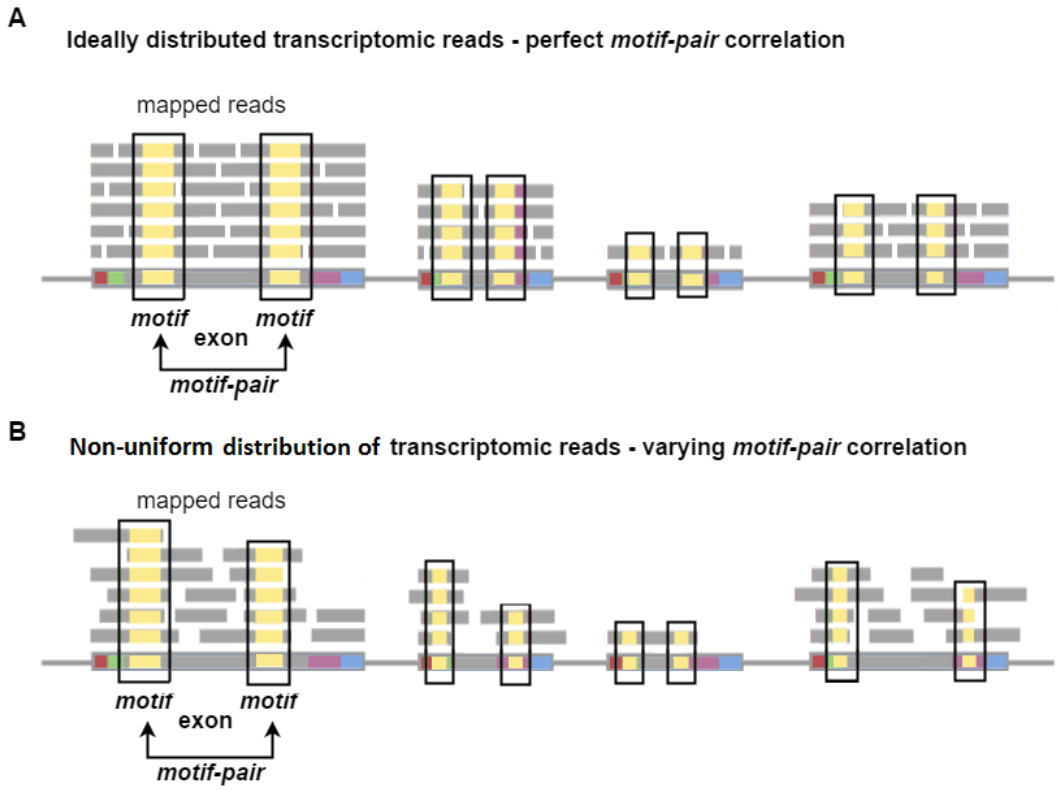


Figure 6.3: Schematic depicting method for quantification of read coverage using short pairs of sequence motifs (typically *4-mers*) within the reads shown in yellow. **A)** Ideal distribution of RNA-Seq reads mapped to an exon if the reads were contiguous and of sufficient length - motif-pairs show perfect correlation. **B)** Non-uniform distribution of RNA-Seq reads mapped to an exon - motif-pairs show variable correlation.

6.1.1 Transcriptomics input data

In transcriptomics analysis, and for our task of analysing the distribution of reads aligned to a reference genome, two input data files (sample records shown in Figure 6.4) are required: i) a reference genome annotation file defining exon boundaries for the species under investigation and ii) an aligned (mapped) reads data file produced by sequence alignment software. Reference genome annotation data have a widely adopted standard for storage of gene structure information - the GTF (Gene Transfer Format) file format which comprises of tab-delimited fields (Sanger, 2012). From the input GTF data file the fields we are interested in are the chromosome name $c_{i,j}$, the feature type $t_{i,j}$, feature start position $a_{i,j}$ and feature end position $b_{i,j}$, where i, j are the indices of the chromosome and exon respectively. As we only examine GTF features of type "CDS" (coding sequence) we can dispense with referring to $t_{i,j}$ from herein. The reason for this is that, as shown in Figure 6.1, the CDS region of the exon excludes the 5' cap, 5'UTR and 3'UTR and Poly-Adenylated tail - regions that contain regulatory sequences that would bias our motif analysis. We represent genome annotation data contained in a GTF annotation file of length IJ lines by the tuple defined below:

$$G_{i,j} = \langle c_{i,j}, a_{i,j}, b_{i,j} \rangle_{i,j=1}^{IJ} \tag{6.1}$$

A

$c_{i,j}$		$t_{i,j}$	$a_{i,j}$	$b_{i,j}$					
chr3RHet	dm3_flyBaseGene	start_codon	2479058	2479060	0.000000	+	.	gene_id "CG12449"	transcript_id "CG12449-RB"
chr3RHet	dm3_flyBaseGene	CDS	2479058	2479147	0.000000	+	0	gene_id "CG12449"	transcript_id "CG12449-RB"
chr3RHet	dm3_flyBaseGene	exon	2479026	2479147	0.000000	+	.	gene_id "CG12449"	transcript_id "CG12449-RB"
chr3RHet	dm3_flyBaseGene	CDS	2480187	2481026	0.000000	+	0	gene_id "CG12449"	transcript_id "CG12449-RB"
chr3RHet	dm3_flyBaseGene	exon	2480187	2481026	0.000000	+	.	gene_id "CG12449"	transcript_id "CG12449-RB"
chr3RHet	dm3_flyBaseGene	CDS	2481086	2481457	0.000000	+	0	gene_id "CG12449"	transcript_id "CG12449-RB"
chr3RHet	dm3_flyBaseGene	stop_codon	2481458	2481460	0.000000	+	.	gene_id "CG12449"	transcript_id "CG12449-RB"
chr3RHet	dm3_flyBaseGene	stop_codon	2481458	2481460	0.000000	+	.	gene_id "CG12449"	transcript_id "CG12449-RB"

B

		d_k	r_k		g_k			s_k
HWI-ST571_69:4:2102:1601:181957:ACAGTG	0	chr3RHet	1681646	3	36M	*	0	0	GTTTGATCAAAGGGCCGAATAGACCGGGTTTCAAGT

Figure 6.4: Fields utilised from the GTF and SAM input data. **A)** A small selection of typical chromosome records (where index $j = n \ldots j = n + 7$ for given chromosome i) from a GTF genome annotation file. **B)** A typical single mapped read from a SAM reads alignment file (remainder part of record omitted for brevity)

Additionally, SAM (Sequence Alignment/Map) is a widely adopted tab-delimited file format for storing biological sequences aligned to a reference sequence, and is a common output file of read alignment software (Li et al., 2009a). The SAM file has a line for each read which we will index by k, and various fields for each read, of which we

are interested in the read chromosome name d_k, read position r_k, sequence alignment information which is encoded in a string termed a CIGAR string (Concise Idiosyncratic Gapped Alignment Report) g_k (also discussed in further detail by Li et al.), and the sequence itself s_k, for each read S_k. To represent all the reads in the input SAM file of length N lines we define the following tuple:

$$S_k = \langle d_k, r_k, g_k, s_k \rangle_{k=1}^{N} \tag{6.2}$$

Table 6.1 lists the fields in both input data files (SAM and GTF) which we will utilise for our analyses.

GTF (Genome Annotation) tuple G		
Field	**Variable**	**Description**
chromosome	$c_{i,j}$	Name of the chromosome sequence in which the feature resides, i.e. "chr1". There are multiple features per chromosome.
feature type	$t_{i,j}$	Name of the feature type, i.e. "CDS", "start_codon", "stop_codon", and "exon"
feature start	$a_{i,j}$	Start position of the feature relative to the chromosome sequence named in $c_{i,j}$, the sequence numbering for the first base starts at 1
feature end	$b_{i,j}$	End position of the feature relative to the chromosome sequence named in $c_{i,j}$, the sequence numbering for the first base starts at 1
SAM (Sequence Aligned Mapped reads) tuple S		
Field	**Variable**	**Description**
chromosome	d_k	Name of the chromosome in which the read occurs (referred to as RNAME in SAM specification)
read position	r_k	Leftmost mapping position of the read relative to the chromosome sequence named in d_k, the sequence numbering for the first base starts at 1
read CIGAR	g_k	CIGAR string containing read mapping information
read sequence	s_k	Sequence string of bases in the read

Table 6.1: GTF and SAM input data fields used in our analyses and their assigned variables in the input data tuples (G and S) we have defined. Both file formats are tab-delimited, with each GTF feature or SAM read occurring on its own line. The GTF tuple G is indexed by i and j which refer to the chromosome and feature indices respectively and the SAM tuple S is indexed by k (Sanger, 2012; Li et al., 2009a).

6.1.2 *H. sapiens* IVT (*In-vitro* Transcription) RNA-Seq dataset

As discussed in chapter 3 (section 3.7), and in section 6.1 in order to explore bias in RNA-Seq, we have analysed three Human (*H. sapiens*) transcriptome samples from an RNA-Seq dataset prepared, in a highly controlled way, by IVT (*In-vitro* Transcription) (Lahens et al., 2014). The RNA in these samples has been transcribed from cDNA clones in *E. coli* DH5α cells. The dataset comprises of a pool of 1,062 RNAs from a full-length human cDNA library sequenced using RNA-Seq. The first sample, *IVT-Only*, had its IVT RNA subjected to ribosomal RNA depletion prior to sequencing, whilst the second sample, *IVT-PolyAsel*, had polyadenylated selection applied instead of ribosomal depletion - these are two different, routinely used protocols for selecting specifically mature (mRNA) from RNA samples. The third sample, *IVT-Plasmids*, represents direct sequencing of the Human IVT plasmids without RNA-selection (i.e. niether ribosomal depletion nor polyA selection were applied). The datasets were produced by the Smilow Center for Translational Research, Philidelphia, USA and featured in a publication by Lahens et al. (2014), and the third sample was one of the controls used in the original research paper. This data is deposited at the GEO database with the ID GSE50445 (Lahens NF, 2012). The information regarding the source of the biological samples, the sample preparatory protocols and post-sequencing processing that were applied have been documented in Table 6.2.

6.1.3 *D. melanogaster* transcriptomics datasets

In order investigate *intra-exon* motif pair correlation within the reads from a "real world" example, we have used two Drosophila (species *D. melanogaster*) transcriptomics datasets which differ by mutation *gl[60j]* in the eye-antennal disc. These are the full transcriptomes of the wild and mutant-type glass eye mutations, acquired from Stein Aerts Laboratory of Computational Biology at University of Leuven, Belgium. This data is deposited at the GEO database with the ID GSE39781 (Aerts, 2012). The *D. melanogaster* datasets featured in a research publication by Naval-Sánchez et al. (2013) and the information regarding the source of the biological samples, the sample preparatory protocols and post-sequencing processing that were applied have been documented in Table 6.2 below.

***H. sapiens* (*IVT-Plasmids*, *IVT-Only* and *IVTpolyAsel*)**	***D. Melanogaster* (wild and mutant)**
Replicates: 1x .sam file for each IVT sample (no replicates)	**Replicates:** 4x .sam files (two replicates for each species).
Sample preparation	**Sample Preparation**
Glycerol stocks containing individual cDNAs (cloned into pCMV-Sport 6 plasmid) from the MGC (Mammalian Gene Collection) (Temple et al., 2009) were produced. Plasmid DNA was extracted from these glycerol stocks and plated at 50 ng per well in 384-well plates. The contents of three 384-well plates (total of 1,062 human transcripts) were collected. The plasmid library was then amplified by transferring 10 ng into *E. coli* DH5α cells (Invitrogen, Life Technologies, Carlsbad, CA, USA, catalog no. 18258012). The heat shock method was used to transform E. coli (See Lahens et al. (2014) for more details). The Plasmids were then purified using Qiagen (Hilden, Germany) maxiprep kit (catalog no. 12163), according to the manufacturer's protocol. Samples were sequenced on the Illumina HiSeq 2000 platform.	For the wild type *D. melanogaster*, fly stocks (Canton-S and strain RAL-208) were obtained from the inbred collection of T. Mackay. For the mutant-r2 type *D. melanogaster*, fly stocks (stock 507) were obtained from the Bloomington Stock Center. Eye-antennal and wing imaginal discs were dissected. RNA was extracted, yielding $\sim$ 3 mg of total RNA per sample. The samples were processed into libraries according to the Illumina TruSeq protocol with appropriate indices, pooled. Sequencing of the transcriptomes was performed on the Illumina HISeq 2000 platform.
RNA-Seq	**RNA-Seq**
After sequencing, raw reads from the samples were aligned to the human genome (GRCh37/hg19) using the RNA-Seq Unified Mapper (v2.0.4) with default parameters. Only reads that mapped to a single location were used (selected from the *RUM_Unique* aligned reads file).	After sequencing, Fastx-clipper (Gordon and Hannon, 2010) was used to discard reads containing residuals of adapter sequences were discarded (FastX clipper version 0.0.13 with option -M15). Quality control was applied to the raw sequence reads and performed using the FastQC software (Andrews et al., 2010) (version 0.9), checking for PHRED quality >20 and different primer contaminations. The reads were then aligned using TopHat v2.0 (Trapnell et al., 2009) with default parameters, to the Flybase Drosophila Melanogaster genome version r5.45 (released March 2012) (Gramates et al., 2016).

Table 6.2: The sample and library preparation protocols, together with the data-processing steps, applied to the RNA-Seq datasets used in this analysis. (Left) *H. sapiens* IVT RNA-Seq. (Right) *D. melanogaster*.

6.1.4 Overview of the implementation

The implementation of our analysis method comprises of two main phases (depicted in Figure 6.5). In the first phase, motif count and position information is distilled for all exons in the SAM input tuple S. All reads in S must first be partitioned by exon, and the occurrences of the motif in the read sequence s_k and their positions counted. MapReduce on the Apache Spark platform is employed for this first phase, and involves 3 map steps and a reduce step.

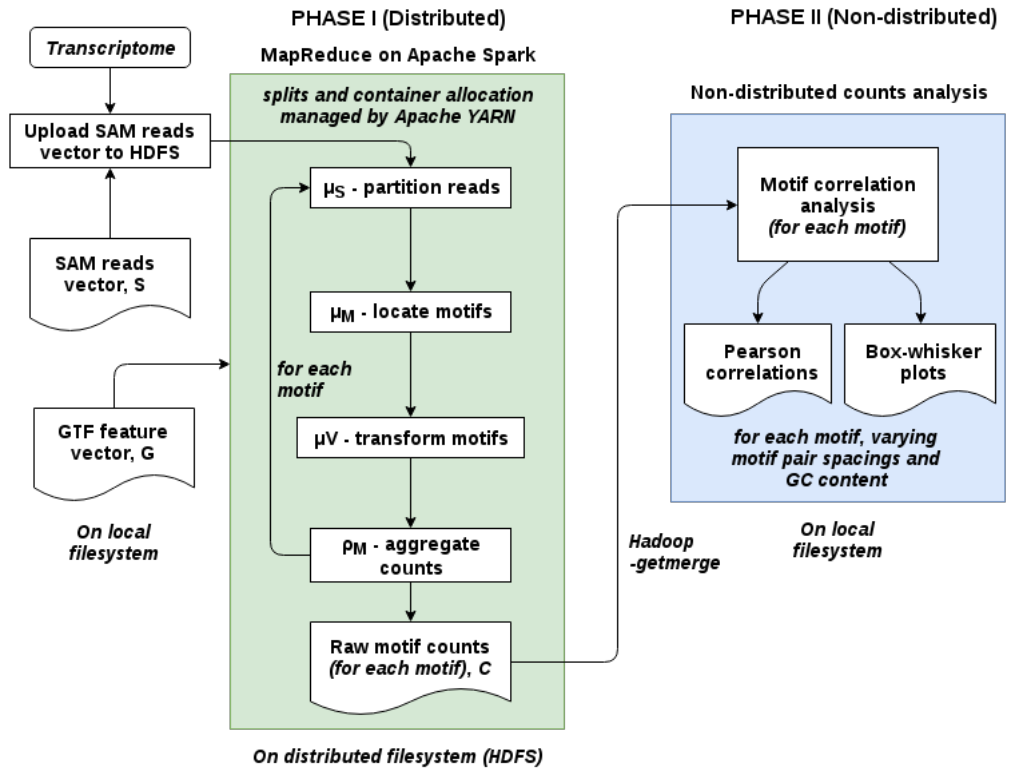


Figure 6.5: Overview of method for quantifying sequence-specific deviations in read distribution. Phase I, the distributed phase, comprises 3 map steps and a reduce step on Apache Spark, with intermediate data being stored on HDFS. Phase II, the non-distributed phase, counts analysis phase utilises raw motif count and position data generated by phase I, which has been stored on the local file system.

The MapReduce steps are daisy-chained such that the output of one MapReduce step is the input of the next step until the final reduce step. The order and function of these steps is outlined in Table 6.3.

Name of step	Designation	Purpose
1. GTF-SAM map	μ_S	Partitions reads in S by exons in G.
2. MOTIF map	μ_M	Returns a set of positions in which the motif occurs in the read.
3. VECTOR map	μ_V	Maps each occurrence of the motif with a value of 1.
4. MOTIF reduce	ρ_M	Aggregates counts for the motif at given positions in the exon.

Table 6.3: MapReduce steps employed in the distributed phase I of our analyses method. Phase I utilises as input widely-used transcriptomic data file formats: SAM aligned reads and GTF genome annotation (which we represent at tuple sets S and G respectively) and yields distilled motif counts data $C_{n,m}$ comprising of exon, position of motif and count for each exon and each motif. Data in $C_{n,m}$ is then processed by a non-distributed phase II to compute correlations which can quantify sequence-specific deviations in the distribution of mapped reads across exons.

The first map step is μ_S map which partitions reads by exon via an exon lookup function el. el utilises the GTF annotation tuple $G_{i,j}$ returns a key for the exon the read is mapped to (mapping occurs prior to our analysis by the read-alignment software). After applying the lookup function el, the output of the μ_S map step are the partitioned reads E. E is a list of key-value pair tuples comprising of the exon-key (which we discuss later in this section) and the raw read data as the value, and is defined below:

$$E = \langle e, S(e) \rangle_{i=1}^{N} \tag{6.3}$$

The second map step μ_M takes a read and returns an exon key e and vector P of the positions in which the motif occurs for that exon. The third map step μ_V and final reduce step ρ_M transform and aggregate the data by exon $e_{n,m}$, position $q_{n,m}$ and count $z_{n,m}$ - This transformation is depicted in the next section in Figure 6.8. This processing yields tuple $C_{n,m}$, which is the final output from the distributed phase, where q is the motif position on the exon e and z is the count of motifs overlapping position q, and n, m are the indices of the count tuple and *4-mer* motif respectively. $C_{n,m}$ is defined below:

$$C_{n,m} = \langle e_{n,m}, q_{n,m}, z_{n,m} \rangle_{n,m=1}^{NM} \tag{6.4}$$

In the next section we will detail phase I of the analyses method which utilises MapReduce.

6.2 Phase I - Counting sequence motifs with MapReduce

As we have introduced in chapter 2, MapReduce is a paradigm for processing large datasets in a parallel fashion on distributed infrastructure, and is implemented by way of distributed functions such as map μ and reduce ρ. The key-value pair $\langle e, v \rangle$ is the main structure in MapReduce, where e is the key and v is the value. MapReduce functions act on key-value pairs and a typical input to a function is a list of key-value pairs $\langle e, v \rangle_{i=1}^{N}$ within a *split* to a total size of N. In this chapter, so as not to interfere with k which we define as the index of the read in the SAM reads tuple S, we will designate e as the MapReduce key and its value as v. Furthermore we do not depict the splitting of input and output data and their distribution to MapReduce functions running on nodes of the cluster, which is covered in detail in chapter 2. In applying MapReduce to our method to quantify sequence-specific deviations in the distribution of mapped reads, we composed key e from the GTF tuple G that allows us to assign RNA-Seq reads to a given unique exon. This MapReduce key e is composed of feature start $a_{i,j}$ and feature end $b_{i,j}$ for each exon boundary. It is constructed by concatenating the 10 digit left-padded string representation of $a_{i,j}$ with the 10 digit left-padded string representation of $b_{i,j}$, producing a final, unique 20 digit key for each exon e, and an example is depicted in Figure 6.6 below:

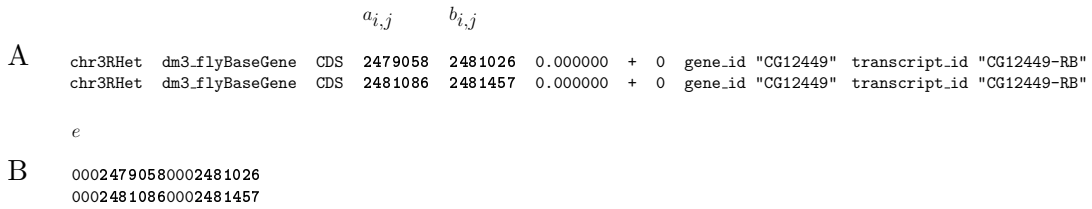


Figure 6.6: Composition of exon key e from $a_{i,j}, b_{i,j}$ (feature start position and feature end position, respectively) of GTF data G **A)** A small selection of typical chromosome records from a GTF genome annotation file. **B)** corresponding exon key e for each GTF record in A.

6.2.1 Preparation for MapReduce on Apache Spark

Preparatory steps for running the Phase I MapReduce job together with details on suitable Apache Spark parameters can be found in the Appendix.

6.2.2 Partitioning reads by exon - GTF-SAM map μ_S

In order to partition transcriptomics reads in the SAM reads vector S by exon e a MapReduce map operation μ_S is performed. The output of this map step ensures that a set of reads are assigned to the exon which they are aligned to (using read alignment information from the upstream read alignment software) and for each exon e comprises of a list of key-value pairs. The associated values $\langle v_1 ... v_M \rangle$ for the exon key e are the short reads S_k obtained from the SAM reads vector S. The output of the μ_S step is therefore in the form of $\langle e, S(e) \rangle$. Each sequence read has a read start position r_k (generated by the alignment software) which is relative to the start position of the chromosome named in d_k and which matches the corresponding chromosome named in $c_{i,j}$. It is important to note that in order to partition reads by the exon they are aligned to, the μ_S step employs on a lookup function el (its algorithm is defined in 3). el utilises the GTF annotation tuple G to return the exon key e for the exon that the read is aligned to. This is achieved using the exon's feature start position $a_{i,j}$ and feature end position $b_{i,j}$, where i, j are the indices of the chromosome and exon respectively. The positions of the reads relative to the range of the exon are were also evaluated as depicted below:

Figure 6.7: Selection of typically distributed RNA-Seq reads mapped to an exon, where $a_{i,j}, b_{i,j}$ are the feature start position and feature end position, respectively. The reads shown in yellow straddle the left of the exon ($Sl = True$) are selected, as are those in green contained within the exon ($Rw = True$) and those straddling the right ($Sl = True$). Reads shown in grey are not selected for motif count analysis.

Reads were selected by implementing a Boolean function $Rs_k(r_k)$ (see function 6.5 below) which selects reads depending on whether their read start or read end positions r_k occur in a GTF feature range $a_{i,j} - b_{i,j}$. As the SAM input data S does not possess a field for read end position, the read end position is computed from $x_{i,j}$ and the CIGAR string g_k. We designate $x_{i,j}$ and $y_{i,j}$ to be the read start and end positions respectively.

We only select reads that are a contiguous alignment match (as determined from g_k) and ignore subsequent alignments. We are interested in selecting reads ($Rs = True$) which have start positions that straddle the left of the exon Sl, reads that have end positions that straddle the right Sr of exons (i.e. straddling either side, Se) or that have both ends contained within the exon Rw as follows:

$$Rs_k \quad = \quad (Rw_k \vee Sl_k \vee Sr_k) \tag{6.5}$$

The $Rs_k(r_k)$ function defined above is utilised by the exon lookup function el which we define below (algorithm 3). el is a recursive function that employs a binary search method and takes as arguments a subset of the GTF vector G, the read's chromosome name d_k, read start $x_{i,j}$ and read end $y_{i,j}$ positions. It returns the exon key e or -1 if the read was not found within any exon of G. It is important to note that in order to apply this binary search technique, G must be numerically sorted by $a_{i,j}$ in ascending order. The subset G' comprises of the records of G that correspond to the chromosome name in the read d_k (which is equal to $c_{i,j}$) and is defined below:

$$G' = \langle G(c_{i,j}) \rangle \tag{6.6}$$

In applying the binary search method el uses a pivot m, which is the middle of the GTF features subset G', rounded to the lowest integer as follows:

$$m = \frac{|G'|}{2} \tag{6.7}$$

Algorithm 3: Exon lookup function to find the exon e to which a read S_k is sequence-aligned to. This function is recursive and operates on a subset of GTF annotation records G' for a given chromosome name c_k, where $G' \in G$. G' must be numerically sorted by $a_{i,j}$ in ascending order.

```
1  function el (G', x_k, y_k);
     Input  : Subset of GTF annotation G', read start position x_k and read end
              position y_k
     Output: Exon key e constructed from feature start a and feature end b
2  Compute m;
3  g_m ← mth tuple of G';
4  a_m ← a entry of g_m;
5  b_m ← b entry of g_m;
6  if m = 0 ∨ m = 1 then
7  |  return -1 ;                                          // read not found
8  end
9
10 if Rs_k(x_k, y_k, a_m, b_m) then
11 |  return a_m, b_m as key e ;                           // read found, return e
12 end
13
14 if x_k < a_m then
15 |  call el(G'[1..m], x_k, y_k) ;                       // x_k left of pivot m, recurse left-hand side
16 |
17 end
18 if x_k > a_m then
19 |  call el(G'[m..n], x_k, y_k) ;                       // x_k right of pivot m, recurse right-hand side
20 |
21 end
```

Initially G' contains exons for the given chromosome name $c_{i,j}$ which is recorded in the SAM read chromosome name d_k. *el* is recursive, if the exon is not found in the first call G' is subsequently further divided on each recursion of the function and re-assigned the left or right side subset of G', where the pivot is m, in a binary search fashion.

The result of *el* is returned by the μ_S map function in the form of $\langle e, S(e)\rangle$. μ_S is defined below:

$$\mu_S(S) = \langle e, S(e)\rangle \quad \begin{array}{l}\text{where key } e \text{ is constructed} \\ \text{from } a, b \text{ found by } el \text{ function}\end{array} \tag{6.8}$$

6.2.3 Optimisation of lookup function *el* using binary search

The GTF lookup function *el* described in algorithm 3 is extensively used in the distributed MapReduce phase I of our analysis. It is called repeatedly, in parallel across the cluster by the μ_S map step to partition the reads by exon and receives a large amount of data. To illustrate this point, the SAM reads vector S and GTF feature vector G for the *D. melanogaster* dataset contain 12,960,778 reads, and 252,879 GTF features (99,867 of type "CDS") respectively. If the *el* function were to employ a simple linear search for each read S_k the time complexity would be $O(nm)$ which would be grossly inefficient - it was therefore necessary to optimise this function using a binary search method.

6.2.4 Counting motifs within reads - MOTIF map μ_M

The preceding μ_S map function was used to partition reads by exon returning $\langle e, S(e)\rangle$ which is then passed into the next operation - the μ_M map step. μ_M uses the nucleic acid sequence s_k in the partitioned read and a *4-mer* search motif which is a string we designate f. It returns a tuple $\langle e, P\rangle$ consisting of an exon-key e and a vector P of the start positions at which the search motif occurs for each key and read of the input $\langle e, S(e)\rangle$. To achieve this μ_M map step employs a motif search function sf, which is defined in algorithm 4. It is important to note that the values contained in P returned by μ_M are chromosome sequence relative positions (relative to start position of chromosome sequence named in $c_{i,j}$) and need to be converted into exon-relative positions (relative to $a_{i,j}$). This is achieved by computing offset o_P for the motif start position using the read start position x_k and the exon's GTF feature start position $a_{i,j}$ and applying it to the result P of the motif search function sf as follows:

$$o_P = x_k - a_{i,j} \tag{6.9}$$

$$\forall p \in P \quad : \quad p = p + o_P \tag{6.10}$$

Furthermore, in applying the motif search function (algorithm 4 below) we ignore overlapping sequences in the string but allow for immediately consecutive occurrences of the motif f. For example when searching for the *4-mer* motif `GGGG` in a sequence read s_k containing run of six Gs (`GGGGGG`) the search function sf would return a single position ($P = \langle 1 \rangle$), whereas for a sequence containing a run of eight Gs (`GGGGGGGG`) sf would return two positions ($P = \langle 1, 5 \rangle$).

Algorithm 4: Motif search function sf to locate all non-overlapping occurrences of search motif f in read sequence s_k. This function returns an exon key e and a set of start positions P for each occurrence of the *4-mer* search motif.

```
1  function sf(s_k, f);
   Input  : Sequence read s_k, and search motif f
   Output: P, a set of positions in which the search motif f occurs in sequence
           read s_k
2  x ← 0;
3  P ← ⟨⟩;
4  l ← string length of f ;                          // 4 for 4-mer
5  do
6  |  x ← position of substring f in s_k, starting from position x;
7  |  append x to P;
8  |  x ← x + l;
9  while x ≠ −1;
10 Return P
```

The μ_M map step returns the exon-key e and above re-computed motif positions P in the tuple form $\langle e, P \rangle$, and is defined below:

$$\mu_M(\langle e, S(e) \rangle) = \langle e, P \rangle \quad \text{where } P \text{ is computed from } sf \text{ function} \tag{6.11}$$

6.2.5 Data transformation - VECTOR map μ_V and MOTIF reduce ρ_M

The final MapReduce steps of the distributed phase are the μ_V and ρ_M steps which together transform intermediate data from the μ_M step in the tuple form $\langle e, P\rangle$ into motif count and position data in the tuple form $\langle e_{n,m}, q_{n,m}, z_{n,m}\rangle_{n,m=1}^{NM}$ which is the final output of the distributed phase I of our analysis method, this process is depicted in Figure 6.8 below:

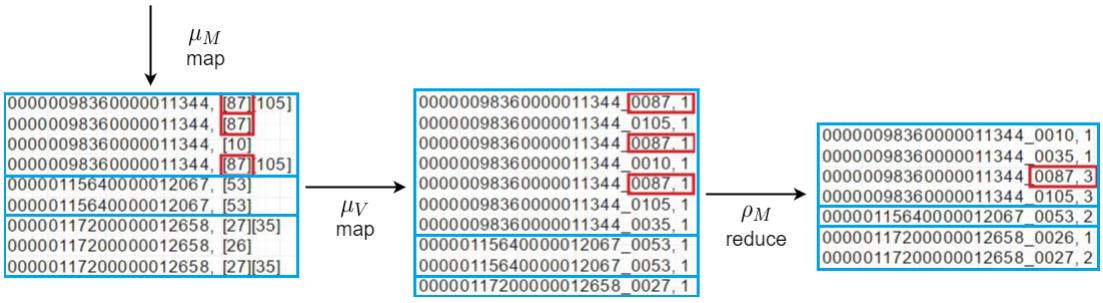


Figure 6.8: Transformation of intermediate data by MapReduce steps. In this example, we illustrate that these steps have computed that three short reads have overlap of a specific motif which starts at position 87 of a specific exon

The penultimate operation in phase I of our analysis method is a μ_V step which operates on tuple $\langle e, P\rangle$ and maps a 1 for each position q in set of motif positions P for each key e. The output yields multiple records in the tuple form $\langle e, q_n, 1\rangle_{n=1}^{N}$ to a total size of N positions for each exon e, as follows:

$$\mu_V(\langle e, P\rangle) = \langle e, q_n, 1\rangle \quad \text{for all } p \text{ in } P \tag{6.12}$$

The ρ_M is the final operation in phase I and takes for input the set of tuples $\langle e, q_n, 1\rangle_{n=1}^{N}$ produced by the preceding μ_V step described above. ρ_M aggregates multiple records for a given exon and position into single record of motif counts for a given exon and position in the form $\langle e_{n,m}, q_{n,m}, z_{n,m}\rangle_{n,m=1}^{NM}$ as follows:

$$\rho_M(\langle e, q_n, 1\rangle_{n=1}^{N}) = e, q_n, |q| \quad \text{for all } q \text{ in } e \tag{6.13}$$

6.2.6 Mean Exon GC content - EXON-GC map μ_G and EXON-GC reduce ρ_G

The subsequent motif analysis phase II requires that the mean GC content for the reads in each exon be pre-computed. It is convenient to use MapReduce for this computation and we use two daisy-chained steps map μ_G and reduce ρ_G. They are computed once per exon, that is they are not repeated for each motif. μ_G takes output from the μ_S step, computes the GC content of each read for each exon in the key-value pair $\langle e, S(e)\rangle$ and ρ_G aggregates average GC content per read gr (as a percentage) by exon key e. The output from ρ_G is the tuple GE, which is defined below:

$$GE \quad = \quad \langle e, \overline{ge}\rangle \tag{6.14}$$

A function gc is used by μ_G to compute the GC content of a given read sequence s_k and is described in algorithm 5 below:

Algorithm 5: Function to compute GC content of a read sequence

```
1 function gc(s_k);
  Input  : Read sequence s_k
  Output: GC content of read sequence s_k as a percentage
2 a ← count of bases ∈{A,T,G,C} contained in read sequence s_k;
3 gc ← count of bases ∈{G,C} contained in read sequence s_k;
4 gcp ← (gc/a) * 100
5 Return gcp
```

The μ_G map step is applied to partitioned reads from the μ_S map step and is defined below:

$$\mu_G(\langle e, S(e)\rangle) = \langle e, gr\rangle \quad \begin{array}{l}\text{where } gr \text{ is computed from } gc \\ \text{function}\end{array} \tag{6.15}$$

The ρ_G reduce step receives output from μ_G above and produces the output GE by aggregating all the average $\overline{gr}$ for exons in e into an individual tuple for $\overline{ge}$, and is defined below:

$$\rho_G(\langle e, gr\rangle) = \langle e, \overline{gr}\rangle \quad \text{for all } gr \text{ in } e \tag{6.16}$$

6.3 Phase II - Motif correlations analysis

In this section we will discuss phase II of our analysis method which quantifies sequence-specific deviations in distribution of aligned reads to an exon using data produced by phase I. Specifically we use motif position and count data $C_{n,m}$, in the tuple form $\langle e_{n,m}, q_{n,m}, z_{n,m}\rangle_{n,m=1}^{NM}$ where n, m are the tuple index and motif index respectively, and the mean exon GC content of reads in each given exon, in the tuple form $\langle e, \overline{ge}\rangle$. We will continue to use e as the exon key, the motif position and count tuples C, and the exon mean GC tuples GE which is summarised in Table 6.4, any other variables we will re-use and re-define accordingly. Furthermore we use f_m to denote the four character string representation of the *4-mer* motif indexed by m.

Motif counts tuple $C_{n,m}$	
Variable	**Description**
$e_{n,m}$	Exon key e, a unique identifier for the exon computed by μ_S
$q_{n,m}$	Position in which motif m occurs
$z_{n,m}$	Count of overlapping motif m at position $q_{n,m}$
Mean exon GC tuple GE	
e	Exon key e
$\overline{ge}$	Mean GC content of all reads in exon e

Table 6.4: Data produced by MapReduce steps employed in the distributed Phase I of our analyses method.

For our analyses we are interested in a number of properties of the data, at the transcript level, and in particular how these may cause deviation in the distribution of mapped reads. As discussed in section 6.1, in order to investigate how sequence-specific motifs affect read distribution we examined all the possible *4-mer* motifs ranging from `AAAA` to `GGGG`, indexed by m. Furthermore, as depicted in Figure 6.3, we also examined the counts of overlapping motifs at particular base-pair (bp) spacings for all exons. We use d to refer to the spacing distance of motif-pairs in all exons in which motif m occurs, where d is defined below:

$$d \in \{10, 50, 100, 200\} \text{ bp.} \tag{6.17}$$

The effect of extremes of GC content in sequencing data (as well as microarray data) has been discussed in numerous studies (Chen et al., 2013b; Memon et al., 2010), and we therefore also investigate the effect of the mean GC content of reads within the exon $\overline{ge}$ and the GC content of the *4-mer* motif itself gm. In order to partition reads by mean GC content (which we will discuss later) we also define binned GC content

ranges (30-40%, 40-50%, 50-60% and 60-70%) for $\overline{ge}$ as follows:

$$\overline{ge} \in \{30-40\%, 40-50\%, 50-60\%, 60-70\%\} \tag{6.18}$$

The GC content of a given *4-mer* motif gm is defined as being a value from the set defined below:

$$gm \in \{0\%, 25\%, 50\%, 75\%, 100\%\} \tag{6.19}$$

6.3.1 Preparation for motif correlations analysis on local file system

As outlined in Figure 6.5, phase I is distributed on the Apache Spark platform, and the results are therefore stored on HDFS (Hadoop distributed file system), where each motif m in $C_{n,m}$ is stored in a separate file. In order for these results to be processed by phase II of our analysis method, which is non-distributed, the results are first downloaded from HDFS on the cluster to the local file system.

6.3.2 Noise removal

Transcriptomics data is noisy. This is a result of both biological noise (Struhl, 2007; Lykke-Andersen and Jensen, 2006), and importantly the uncertainty in the quantification of transcripts - referred to as shot noise (Anders and Huber, 2010). For this reason, to tackle the shot noise, we apply a noise removal process which discards low counts of overlapping motifs. In order to decide on an appropriate cut-off, we first examined the motif data files containing the positions and counts of the motifs. We found records which had counts of 1 or 2 for alongside more substantial counts in the same exon - we expect, theoretically, similar counts across the same exon, which would represent uniform coverage of mapped reads. We therefore decided to deem low counts as those that are within the first quartile Q_1 (25th percentile) of the read counts for all exons in the motif data file for motif m. Inspection of the resulting filtered data demonstrated that only these low counts were removed.

6.3.3 Pearson correlation

The Pearson correlation co-efficient quantifies the similarity between two sets of measurements, specifically linear correlation. The test produces a result where a score of +1 indicates positive correlation, -1 indicates inverse correlation and 0 indicates no correlation between the two sets of measurements (Mendenhall and Sincich, 1992). To

investigate the effect of the sequence-specificity of the *4-mer* motif m and their spacing distance d on the distribution of mapped reads, we compute the Pearson correlation coefficients of counts of each motif in the motif-pair (the two sets of measurements) at varying distances for all exons. We designate tuple D (defined below in equation 6.20) to store the Pearson correlation for each motif spacing $\rho(d)$ where f_m is the string sequence of the *4-mer* motif:

$$D \quad = \quad \langle f_m, r(10)_m, r(50)_m, r(100)_m, r(200)_m \rangle_m^M \tag{6.20}$$

We compute the Pearson correlation between the two sets of counts $v_{n,m}, w_{n,m}$ in the motif-pair for motif m on each exon e at a fixed separation of d bp. Each set of counts is of the overlapping motif m at positions separated at a spacing of d bp on the exon and we allow tolerances t of ± 2 bp for 10 and 50 spaced motifs, and ± 4 bp for 100 and 200 bp spaced motifs. In order to compute the Pearson correlation coefficients to store for each spacing d in the correlations tuple D, we utilise a function mp (defined in algorithm 6) which uses the position and counts tuple $\langle e_{n,m}, q_{n,m}, z_{n,m} \rangle_{n,m=1}^{NM}$, motif-pair spacing distance d, spacing tolerance t, and returns motif-pair counts tuple $W_{n,m}$ which takes the form below:

$$W_{n,m} = \langle e, q_{n,m}, v_{n,m}, q_{o,m}, w_{n,m} \rangle_{n,m}^{N_{match} M} \quad \text{where } q_{o,m} = q_{n,m} + d \pm t \tag{6.21}$$

The motif-pair counts tuple $W_{n,m}$ includes positional and counts information for the two motifs in the motif-pair, this is depicted in Figure 6.9 below:

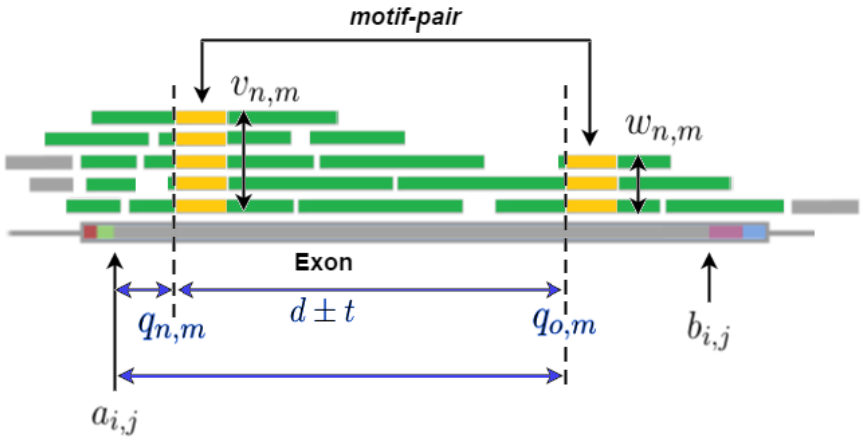


Figure 6.9: Motif-pair information held in tuple $W_{n,m}$ for an exon containing mapped reads. Mapped reads are shown in green and the motif-pair in yellow. The positions of each motif $q_{n,m}, q_{o,m}$ relative to the exon feature start $a_{i,j}$ are shown in blue and by dashed lines, as are the distance between them $d \pm t$. Their respective motif counts $v_{n,m}, w_{n,m}$ are also shown by double-headed vertical arrows to represent quantification of overlapping reads containing the motif at that position.

Algorithm 6: Return set of tuples of motif-pairs at distances of d bp with tolerance of t bp for a specific motif m. We use i, j to iterate through index n of $W_{n,m}$.

```
1  function mp(C_{n,m}, d, t, m);
   Input  : Set of tuples C_{n,m} for all motifs in the form ⟨e_{n,m}, q_{n,m}, z_{n,m}⟩_{n,m=1}^{NM}
   Output: Set of tuples W_{n,m} of motif-pair counts in the form
                ⟨e, q_{n,m}, v_{n,m}, q_{o,m}, w_{n,m}⟩_{n,m}^{N_match M} where q_{o,m} = q_{n,m} + d ± t
2  A ∈ C_{[1..N]} for motif m; // allows us to collapse index n, m to n
3  B ← A; // copy of A
4  W_{n,m} = ⟨⟩;
5  for i = 1 to |A| do
6      for j = i + 1 to |B| do
7          q_A ← q_{n,m} entry of A_i ;          // position of 1st motif in pair
8          q_B ← q_{n,m} entry of B_j ;          // position of 2nd motif in pair
9          if |q_B − q_A| ≥ d − t ∧ |q_B − q_A| ≥ d + t then
               // a motif-pair was found
10             e_{n,m} entry of W_{n,m} ← e_{n,m} entry of A_i;
               // store motif position for each item in the pair
11             q_{n,m} entry of W_{n,m} ← q_{n,m} entry of A_i;
12             q_{o,m} entry of W_{n,m} ← q_{n,m} entry of B_j;
               // store motif count for each item in the pair
13             v_{n,m} entry of W_{n,m} ← z_{n,m} entry of A_i;
14             w_{n,m} entry of W_{n,m} ← z_{n,m} entry of B_j;
15             append W_{n,m} to W;
16         end
17
18     end
19 end
20 Return W
```

The two sets of measurements from the motif-pair from tuple $W_{n,m}$ are therefore $v_{n,m}, w_{n,m}$, respectively, for a specific motif m and distance d, which are used to compute the Pearson correlation coefficient, is defined below. N_{match} is the number of motifs in the set of matching motif-pairs, and $\bar{v}_n$ and $\bar{w}_n$, are the mean of the counts of overlapping reads containing the motif at positions $q_{n,m}$ and $q_{o,m}$, respectively.

$$\rho(d) = \frac{\sum_{n=1}^{N_{match}}(v_n - \bar{v}_n)(w_n - \bar{w}_n)}{\sqrt{\sum_{n=1}^{|v|}(v_n - \bar{v}_n)^2 \sum_{n=1}^{|w|}(w_n - \bar{w}_n)^2}} \tag{6.22}$$

6.3.4 Examining mean exon GC content, Motif GC content and motif-spacing

Extremes in GC content in next-generation sequencing data, that is sequences that have a high or low ratio of GC base composition to that of all bases in the sequence, are known to result in low coverage of reads (fewer reads that align to a sequence at a given position in the reference genome) (Chen et al., 2013b). Runs of four or more guanine residues have been demonstrated to reduce the reliability of gene expression measurement using microarrays (Memon et al., 2010). We therefore describe a method to investigate the effect of different GC parameters on the distribution of mapped reads for pairs of motifs spaced at varying distances.

In section 6.2.6 we defined $\overline{ge}$ and gm to represent mean exon GC content and motif GC content respectively. With these parameters in mind we take motif-pairs produced by algorithm 6 and partition them according by $\overline{ge}$ and gm for each spacing d, regardless of the *4-mer* sequence. We then chose a control that was used to compare Pearson correlation coefficients of motif-pairs. This control was motif-pairs having a GC content of 50% which was then compared to motif-pairs having different GC contents (i.e. comparing 50% vs. 0%, 25%, 75% and 100%).

For statistical purposes we utilise Welch's t-test and Wilcoxon's test to test the Null Hypothesis that the correlations for motif-pairs where the GC content of the motif is not equal to 50% is drawn from the same distribution as that for 50% GC content. Welch's t-test is a variant of the student's t-test which is applied to two samples with unequal variances and is used to test the null hypothesis that two populations have equal mean. Wilcoxon's test is used to test the null hypothesis that two related paired samples come from the same distributions and is a non-parametric test. That is it does not assume that the data comes from a known distribution (Mendenhall and Sincich, 1992).

6.3.5 p-values and the multiple-comparisons problem

As we are performing a considerable number of tests on the input transcriptome there is a possibility that we may reject a null hypothesis when it is true (i.e., a "Type I" error). For instance we might test the null hypothesis that there is *no difference in motif-pair correlation between motif-pairs with a GC content of 50% and those with 0%*,

then mistakenly reject this null hypothesis and assume that we *have* found a difference when in fact the result was due to chance. This problem arises because the more tests we perform the greater the chance that a rare result occurs. This is known as the multiple-comparisons problem and there have been a number of proposed methods to minimise "Type I" errors.

The Bonferroni correction is a multiple-comparison correction used when multiple independent or dependant test are being performed simultaneously on the same data (Dunn, 1961). However, the Bonferroni method is considered by some researchers to be too conservative (Perneger, 1998), and whilst reducing the chance of a "Type I" error does so at the expense of a making a "Type II error" (Armstrong, 2014), i.e. we might fail to reject a false null hypothesis and so miss a significant finding. An alternative method to the Bonferroni correction is to adjust the p-value to account for the increase in chance that we may mistakingly reject the null hypothesis, this is known as the false discovery rate (FDR).

The significance cut-off for an experiment is termed α and a $p-value$ that is below the α value for a test causes us to reject the null hypothesis and take the result as significant. We will use a False positive rate (α) value of 0.05 (5%) which is a widely accepted value for experiments in science. This means that we require 95% confidence that the result is not due to chance.

In order for the multiple-comparisons problem not to compound the effect of repeated measurements on our p-values, we utilise FDR corrected p-values which we provide in parenthesis in our results. FDR corrected results are produced by pooling all the p-values for all the comparison tests on a given dataset and using a density distribution along with our α (0.05) to re-compute each p-value (Benjamini and Hochberg, 1995).

6.3.6 Testing - Synthetic transcriptomics read data

To validate our method for quantifying sequence-specific deviations in the read distribution of mapped reads we created a synthetic transcriptome. This consists of an artificial GTF genome annotation of artificial chromosomes and their features together with a SAM reads file of artificial reads. The SAM reads file was constructed by randomly generating a 50,000 bp sequence template seed in the form of a string of 50,000 characters. The base composition of this seed is shown in Table 6.5 below:

Synthetic seed	
Base	**Count**
Adenine (A)	12,558
Thymine (T)	12,662
Guanine (G)	12,297
Cytosine (C)	12,483
Total bases	50000
GC content	49.6%

Table 6.5: Nucleotide composition of synthetic sequence seed fragment used to generate synthetic transcriptomic reads for calibration of the algorithm.

Artificial "synthetic" reads were then created by generating random start and random end positions from which reads were constructed by copying the sequence from the seed. We deemed it unnecessary to derive complementary reads from the template as the reference genome is not necessary, and therefore we chose to generate corresponding CIGAR strings for each read which were contiguous alignment matches of the same length as the artificial read sequences. Each sequence read has the same length of its exon feature, but has varying number of copies - this represents uniform distribution of reads but at different levels of expression.

In applying our analysis method to this synthetic dataset we expect to observe only perfect Pearson correlations for any *intra-exon* sequence motif-pairs found within the inspected reads.

As discussed in section 6.1 in real transcriptomics data the distribution of reads means that motif-pairs spaced farther apart will show poorer correlation. In the synthetic reads this should not be the case because the reads are generated as contiguous reads which are of the same length as the exon. In order to verify this we have applied phase II of our analysis method to this synthetic dataset and plotted the results as a scattermatrix plot of correlation ρ (r^2), partitioned by spacing d, motif GC gm and mean GC content of reads in the exon $\overline{ge}$ (Figure 6.10). We can see in the scattermatrix plot that all of the *intra-exon* motif pairs are correlated at +1.0 (postive, perfect correlation) for all parameters. It also shows that there is no dependence of correla-

tion on GC content. This is as we would expect from a synthetic randomly generated transcriptome in which there have been no molecular or sequencing processes applied.

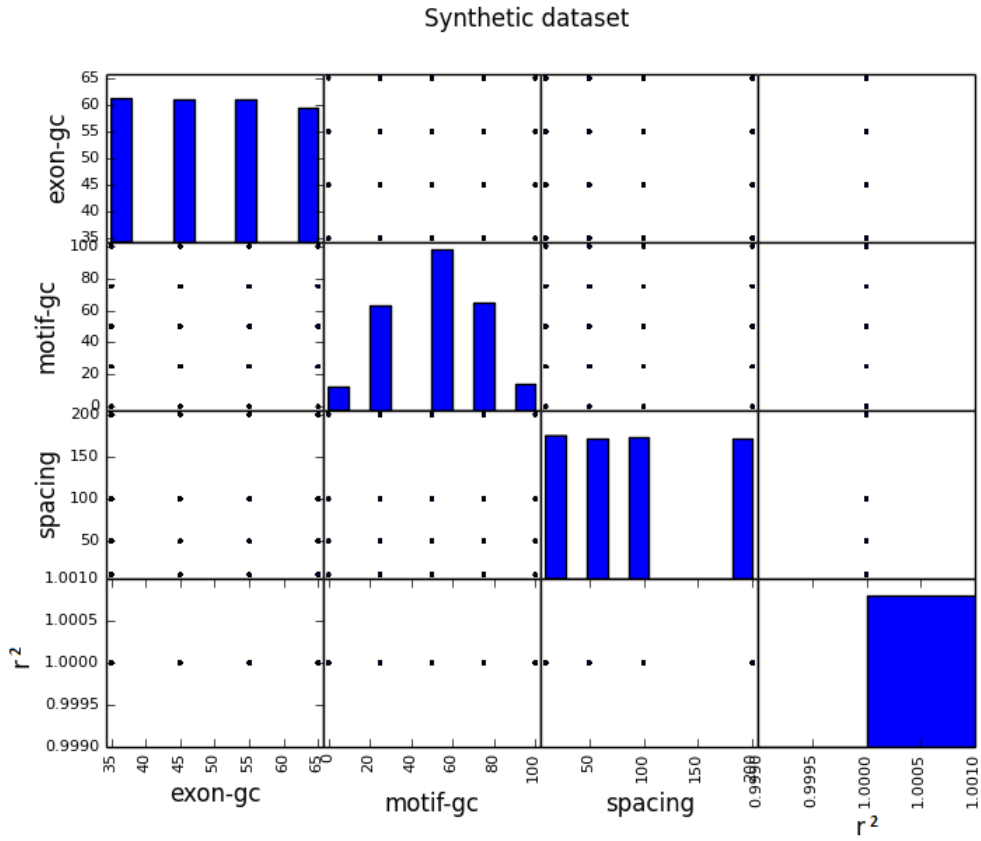


Figure 6.10: Comparison of correlations as a function of Motif and Exon GC content in the synthetic transcriptome. The distribution of correlations (lower right-most sub-plot) shows that the *intra-exon k-mer* pairs are perfectly correlated (score of +1). Distributions are shown horizontally by bar charts (in blue).

6.4 Results

6.4.1 Analysis of IVT (*In-Vitro* Transcribed) RNA in *H. sapiens*

As discussed in section 6.1.2, using the method designed and described in this chapter, we have analysed three *H. sapiens* samples which were prepared by IVT RNA-Seq. Lahens et al. (2014) in their study used RNA that has been in-vitro transcribed (IVT) from cDNA clones in *E. coli*. Their rationale was that the "nucleotide sequence at every base was known, the splicing pattern established, and the expression the level coverage is uniform across the transcript.". This means that any bias occurring in the coverage of reads in these three samples must be as a result of technical rather than biological origin.

We carried out our analysis on these three samples. We note that, as shown in Figure 6.11, although the median exon length in *H. sapiens* is 121 bp (shorter than Drosophila) and approximately 80% of the the exons are less than 200 bp (Sakharkar et al., 2004), the remaining 20% of exons will contribute to motif pair correlations at 200 bp apart. The distribution of protein coding exon lengths in *H. sapiens* is shown in Figure 6.11 below.

The first IVT sample we analysed was the *IVT-Plasmids* sample, as this was produced from sequencing the Human IVT plasmids directly without applying Ribosomal depletion or PolyA selection methods, and therefore represents a control. Importantly, the *IVT-Plasmids* sample, by virtue of not having RNA selection protocol steps applied, also reduces the technical sources of variation in read distribution. Table 6.6 shows a number of *4-mer* motif-pairs that have very high correlations (Pearson correlations very close to +1), and these high correlations are observed across all spacings. There are also some extremely low correlations due to a lack of 4-mer data (as indicated by the sample sizes in parenthesis). In order to visualise correlations across the *IVT-Plasmids* sample, we partitioned the results as a function of GC content of the motif and GC content of the exon and produced box and whisker plots (Figure 6.12). We observe reasonably good correlation, with a median correlation of approximately 0.35, across all of the *4-mer* pairs as a function of the GC content of the exons (right side of the figure), and reasonably good correlation across different Motif GC concentrations except that of 100% motif GC content. Table 6.7 shows that there is no dependence of correlation on Motif GC content in the *IVT-Plasmids* sample, i.e. there are no asterisk results, which means the null hypothesis, that *there is no statistically significant difference between a motif GC content of 50% with any other motif GC content*, holds true for all mean exon GC partitioned correlations.

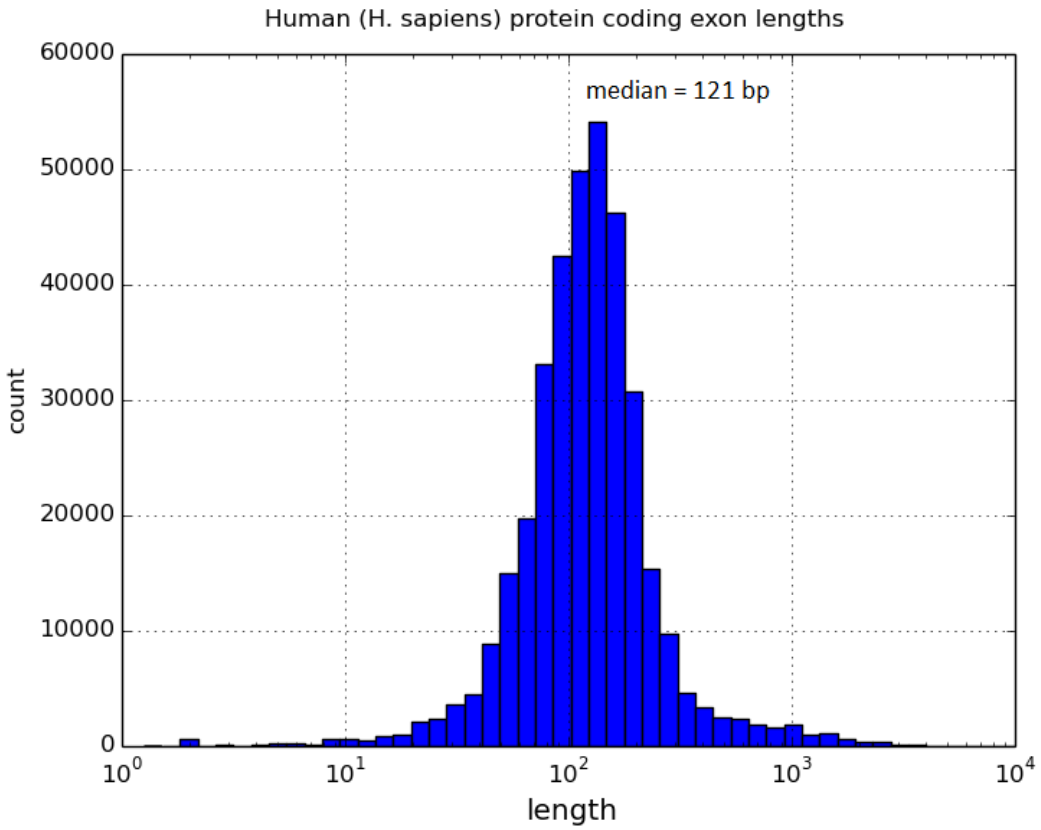


Figure 6.11: Histogram of *H. sapiens* Exon lengths . The data is computed from the hg19/GRCh37 genome annotation GTF which was used in the analysis of the IVT samples. The median exon length is 121 bp, mean 320 bp, Q1=84 bp, Q3=168 bp, the longest exon is 12218 bp.

A

IVTplasmids *H. sapiens*			
Lowest 10 Pearson-correlation outliers and their motifs			
R(10 bp)	R(50 bp)	R(100 bp)	R(200 bp)
ATCG=-0.0543 (23)	ACGA=-0.2715 (15)	TCGA=-0.2600 (16)	ACGA=-0.3050 (10)
CGTC=-0.0209 (44)	GACG=-0.0101 (39)	CGAA=-0.1953 (17)	TCGC=-0.1834 (13)
ACCG=-0.0129 (49)	AACG=0.0000 (9)	CGTT=-0.1510 (10)	ACCG=-0.1608 (18)
CGTT=-0.0052 (21)	TTCG=0.0000 (9)	GTCG=-0.0786 (19)	GTAC=-0.1465 (21)
CGTA=0.0000 (7)	TCGA=0.0000 (8)	CGGT=-0.0304 (45)	TCGA=-0.1274 (11)
CGAC=0.0479 (43)	CGTT=0.0000 (8)	CGTA=0.0000 (5)	CGAG=-0.1055 (41)
ACGC=0.1639 (40)	TCGT=0.0000 (8)	ACGT=0.0000 (9)	TGCG=-0.0935 (40)
GCGA=0.1665 (62)	TACG=0.0000 (7)	TACG=0.0000 (4)	GTCT=-0.0547 (54)
ACCC=0.1899 (260)	CGAT=0.0000 (9)	CCGT=0.0312 (37)	TAGG=-0.0541 (24)
GGGG=0.1991 (649)	TCGC=0.0884 (28)	CGAT=0.0407 (18)	GCGA=-0.0482 (18)
Highest 10 Pearson-correlation outliers and their motifs			
R(10 bp)	R(50 bp)	R(100 bp)	R(200 bp)
TAGA=0.9985 (48)	TAGG=0.9943 (19)	ATCG=0.9822 (16)	TCAA=0.8609 (57)
GTCG=0.9984 (19)	ACTA=0.9703 (21)	GTAT=0.9581 (33)	ATCG=0.8324 (10)
TTAG=0.9983 (39)	CTAG=0.9692 (19)	CGTC=0.9429 (31)	AGGC=0.7683 (137)
ACGT=0.9978 (19)	CCTA=0.9672 (26)	GTTA=0.9233 (23)	ATCC=0.7553 (49)
ATAG=0.9976 (37)	CATT=0.9644 (103)	AGTC=0.9189 (59)	GGAT=0.7430 (40)
ATTC=0.9974 (85)	ACTT=0.9594 (84)	GTCA=0.8984 (67)	TGAC=0.6765 (52)
GCAC=0.9971 (110)	GATA=0.9590 (27)	TCGT=0.8971 (12)	TACC=0.6553 (22)
TAGG=0.9970 (31)	CTAA=0.9572 (31)	GCAA=0.8938 (72)	TGGT=0.6435 (84)
TCGT=0.9968 (18)	TATC=0.9495 (27)	CATC=0.8817 (117)	GGTG=0.6410 (103)
ACAT=0.9968 (114)	TTAC=0.9389 (33)	ATTC=0.8620 (78)	ACGG=0.6184 (21)

Table 6.6: *H. sapiens* Pearson correlation co-efficient outliers (top ten and lowest ten) for different *intra-exon 4-mer* motif sequence pairs at 10, 50, 100 and 200 bp spacings.

B

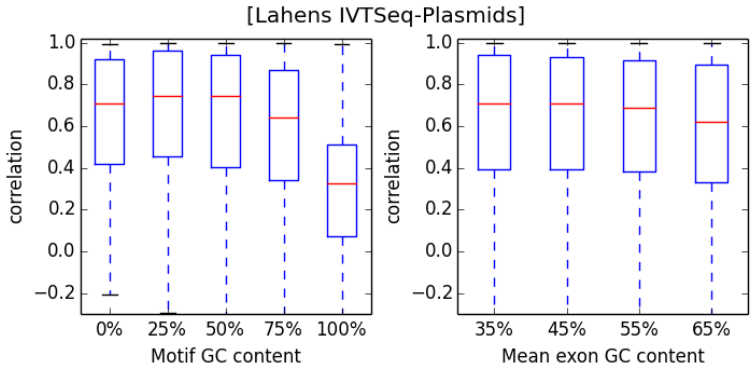


Figure 6.12: Box-whisker plot of the IVT plasmid sample correlations as a function of Motif GC and mean exon GC content.

Motif spacing: **200bp**								
Exon GC%	**30-40%**		**40-50%**		**50-60%**		**60-70%**	
Motif GC%	50	0	50	0	50	0	50	0
#Correlations	16	71	15	69	11	75	0	46
p(t-test)	$5.71x10^{-2}(1.85x10^{-1})$		$6.06x10^{-1}(9.35x10^{-1})$		$9.20x10^{-1}(9.66x10^{-1})$		insufficient	
p(Wilcoxon)	$9.18x10^{-1}(9.18x10^{-1})$		$5.70x10^{-1}(8.04x10^{-1})$		$8.93x10^{-1}(9.18x10^{-1})$		data	
Exon GC%	**30-40%**		**40-50%**		**50-60%**		**60-70%**	
Motif GC%	50	25	50	25	50	25	50	25
#Correlations	63	71	46	69	37	75	2	46
p(t-test)	$7.80x10^{-1}(9.64x10^{-1})$		$6.47x10^{-1}(9.35x10^{-1})$		$7.10x10^{-1}(9.42x10^{-1})$		insufficient	
p(Wilcoxon)	$4.31x10^{-1}(7.63x10^{-1})$		$3.97x10^{-1}(7.30x10^{-1})$		$7.34x10^{-1}(8.49x10^{-1})$		data	
Exon GC%	**30-40%**		**40-50%**		**50-60%**		**60-70%**	
Motif GC%	50	75	50	75	50	75	50	75
#Correlations	21	71	38	69	44	75	51	46
p(t-test)	$2.83x10^{-1}(5.36x10^{-1})$		$6.21x10^{-1}(9.35x10^{-1})$		$3.83x10^{-1}(6.80x10^{-1})$		$9.49x10^{-1}(9.66x10^{-1})$	
p(Wilcoxon)	$7.15x10^{-1}(8.49x10^{-1})$		$3.88x10^{-1}(7.30x10^{-1})$		$2.34x10^{-1}(5.85x10^{-1})$		$8.83x10^{-1}(9.18x10^{-1})$	
Exon GC%	**30-40%**		**40-50%**		**50-60%**		**60-70%**	
Motif GC%	50	100	50	100	50	100	50	100
#Correlations	1	71	10	69	15	75	16	46
p(t-test)	insufficient		$7.53x10^{-1}(9.64x10^{-1})$		$8.18x10^{-2}(2.25x10^{-1})$		$6.16x10^{-1}(9.35x10^{-1})$	
p(Wilcoxon)	data		$3.98x10^{-1}(7.30x10^{-1})$		$4.09x10^{-2}(2.04x10^{-1})$		$6.79x10^{-1}(8.49x10^{-1})$	

Table 6.7: T-test and Wilcoxon-test comparisons of Pearson correlations for motif-pairs at 200 bp spacing for varying motif GC and mean exon GC content in the *IVT-Plasmids* sample. FDR corrected p-values in parenthesis.

In order to compare the effect of applying different RNA selection protocol methods to the IVT samples, specifically ribosomal depletion vs. polyA selection, we compared the *IVT-Only* and *IVT-PolyA* samples respectively. Table 6.8 below shows that the highest outliers for both these IVT-Seq samples show a number of *4-mer* motif-pairs that have very high correlations. High correlation outliers are observed across all spacings. We produced box-whisker plots of correlation as a function of GC content of the motif and GC content of the exon (Figure 6.13). The trend of the data for *IVT-Only* and *IVT-PolyA*, as a function of Motif GC content and Mean GC content, is similar to that for the *IVT-Plasmids* data although the median correlation is somewhat less.

A

IVT-Only H. sapiens			
Lowest 10 Pearson-correlation outliers and their motifs			
R(10 bp)	R(50 bp)	R(100 bp)	R(200 bp)
CGTT=-0.1726 (18)	CGAC=-0.2017 (22)	CGAC=-0.1421 (28)	CTAC=-0.2360 (30)
CGTA=-0.1470 (13)	GCGT=-0.1374 (25)	GCGA=-0.1255 (33)	ACGC=-0.2332 (14)
ACGC=-0.1337 (35)	CGAT=-0.1171 (11)	ACCG=-0.1224 (25)	TCGG=-0.2111 (21)
CCGT=-0.0979 (37)	CCTA=-0.1010 (34)	GACG=-0.0923 (29)	GTAT=-0.1806 (13)
GCGT=-0.0746 (44)	TAAC=-0.0980 (42)	CGAA=-0.0903 (18)	CCGA=-0.1790 (17)
GAGT=-0.0628 (101)	CGAA=-0.0884 (18)	GTTA=-0.0891 (32)	CGGT=-0.1735 (16)
CGTC=-0.0298 (44)	TAGG=-0.0698 (32)	CGTT=-0.0832 (11)	ACGG=-0.1720 (16)
CGGA=-0.0217 (75)	AACC=-0.0657 (81)	CGGT=-0.0666 (34)	ACTA=-0.1571 (23)
GCGA=-0.0156 (52)	CGCA=-0.0563 (40)	GTTC=-0.0627 (48)	GACG=-0.1540 (26)
TACG=0.0000 (6)	GCTA=-0.0562 (26)	GTAG=-0.0583 (40)	TGCG=-0.1532 (34)
Highest 10 Pearson-correlation outliers and their motifs			
R(10 bp)	R(50 bp)	R(100 bp)	R(200 bp)
TACA=0.9989 (98)	TCGA=0.9805 (15)	ATCC=0.9609 (41)	TAGT=0.9913 (20)
CTCA=0.9983 (168)	GACT=0.9713 (56)	AGTA=0.9403 (31)	ATCC=0.9860 (35)
AGTA=0.9978 (50)	TGCA=0.9674 (114)	TATA=0.9261 (50)	GCAT=0.9734 (40)
TACT=0.9977 (47)	AGGT=0.9667 (70)	CGTG=0.9146 (39)	CTCG=0.9417 (19)
TGAA=0.9975 (229)	ATGC=0.9626 (44)	TCAC=0.8733 (77)	AGTC=0.8869 (34)
GTCA=0.9970 (81)	CCTT=0.9551 (132)	CACC=0.8694 (164)	TCAC=0.8492 (47)
GCAT=0.9969 (92)	TTGC=0.9544 (48)	CACT=0.8642 (109)	TGTG=0.8017 (111)
AAGA=0.9969 (257)	ATGG=0.9543 (93)	CCCT=0.8530 (275)	ATTC=0.7894 (52)
AACT=0.9965 (104)	CTCG=0.9507 (30)	TGAT=0.8525 (64)	CTGT=0.7682 (114)
GTAT=0.9963 (24)	CCGT=0.9480 (17)	GTAC=0.8448 (18)	TACC=0.7581 (22)

B

IVT-PolASel *H. sapiens*			
Lowest 10 Pearson-correlation outliers and their motifs			
R(10 bp)	R(50 bp)	R(100 bp)	R(200 bp)
CGTA=-0.1953 (11)	ATCG=-0.2190 (16)	CGAA=-0.2046 (13)	TGCG=-0.2025 (42)
CGTT=-0.1012 (20)	CGTT=-0.2000 (10)	TACC=-0.1573 (28)	ATCG=-0.1998 (12)
GCGA=-0.0976 (67)	AACG=-0.1797 (14)	TCGT=-0.1174 (25)	CGTT=-0.1923 (13)
ATCG=-0.0694 (31)	TCTA=-0.1429 (27)	GCGA=-0.1062 (34)	CGAC=-0.1810 (17)
AACG=-0.0510 (20)	ACTA=-0.1184 (23)	CGCA=-0.0956 (84)	ATAG=-0.1664 (21)
GCGT=-0.0388 (46)	CGCA=-0.1003 (59)	ATAG=-0.0786 (49)	CGTG=-0.1445 (49)
CTTA=-0.0373 (44)	GTTA=-0.0876 (31)	GTTG=-0.0708 (115)	GTTA=-0.1404 (40)
TAGC=-0.0171 (44)	CGAC=-0.0747 (40)	TCGC=-0.0676 (46)	AACG=-0.1316 (13)
GGTA=-0.0107 (66)	TACC=-0.0682 (13)	GACG=-0.0637 (41)	CGGT=-0.1314 (32)
GAGT=-0.0079 (110)	GCTA=-0.0583 (35)	TCGG=-0.0576 (63)	GTAG=-0.1307 (41)
Highest 10 Pearson-correlation outliers and their motifs			
R(10 bp)	R(50 bp)	R(100 bp)	R(200 bp)
TACT=0.9998 (45)	ACTT=0.9887 (68)	AGTA=0.9703 (36)	ATCC=0.9890 (41)
TACC=0.9997 (39)	AGAG=0.9886 (129)	CGAT=0.9035 (18)	ACGA=0.9751 (11)
CTAC=0.9992 (60)	CTAA=0.9726 (25)	TATA=0.8982 (88)	TAGT=0.9358 (33)
TAAC=0.9991 (27)	CACG=0.9685 (37)	CGTG=0.8641 (65)	GCAT=0.8991 (45)
AGTA=0.9986 (57)	TGGA=0.9652 (196)	ACAC=0.8495 (75)	CGAG=0.7917 (30)
AGAC=0.9979 (81)	AAGC=0.9606 (85)	TCAA=0.8321 (93)	TGTG=0.7810 (119)
TGAC=0.9976 (84)	TTGA=0.9571 (82)	CATA=0.8264 (39)	GACG=0.7799 (29)
CCTA=0.9976 (47)	AACT=0.9559 (87)	ACCT=0.8171 (93)	TGAA=0.7644 (123)
TGAA=0.9958 (234)	ATAA=0.9541 (77)	CACT=0.8108 (114)	AATC=0.7620 (35)
ATCC=0.9956 (71)	ATAG=0.9531 (42)	CCAA=0.7891 (124)	CATC=0.7204 (72)

Table 6.8: *H. sapiens* IVT-Seq Pearson correlation co-efficient outliers (top ten and lowest ten) for different *intra-exon 4-mer* motif sequence pairs at 10, 50, 100 and 200 bp spacings. A) IVT only library preparation B) IVT with PolyA library selection. Sample sizes are given in parenthesis.

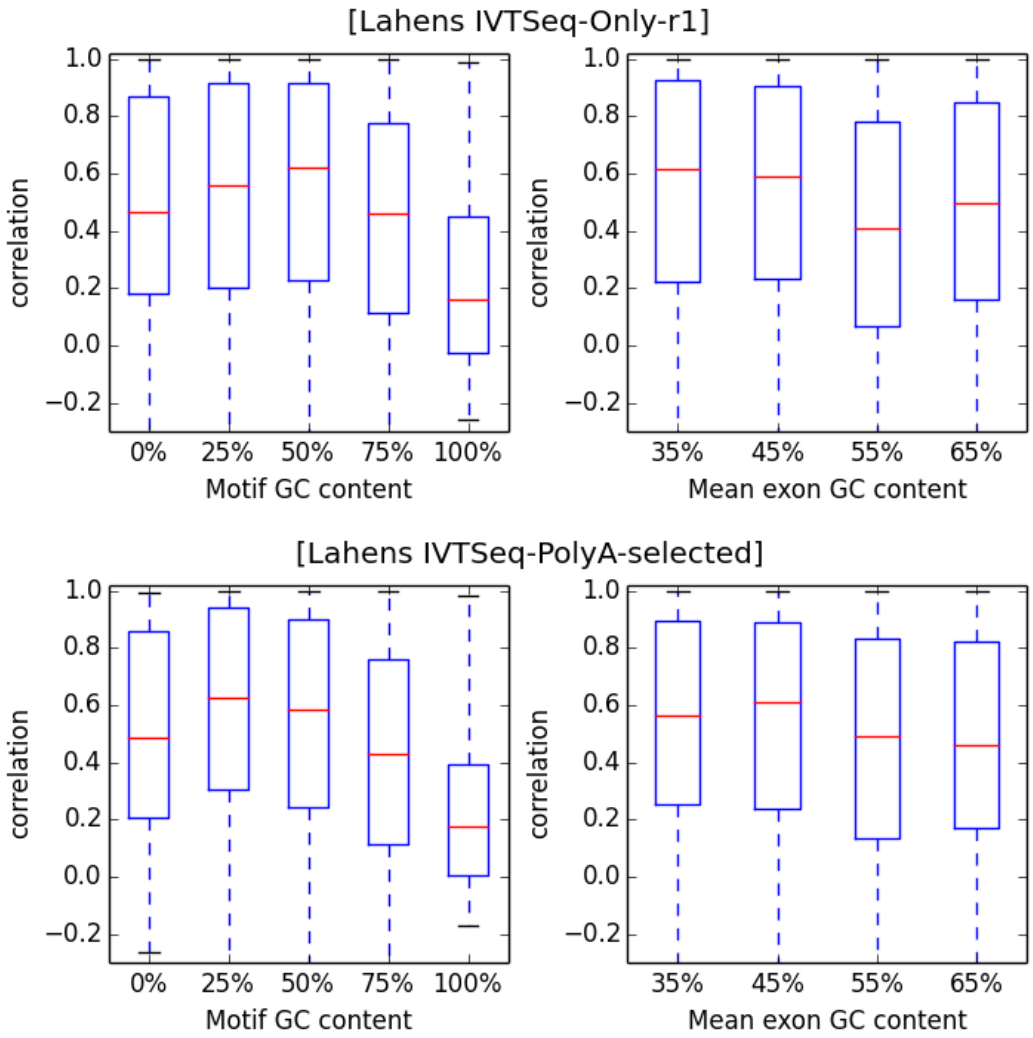


Figure 6.13: Correlation (Pearson's) as a function of *4-mer* motif and exon GC content for *H. sapiens* for the two IVT samples that had different RNA selection protocols applied: Top) *IVT-Only* sample, which underwent ribosomal depletion, and B) *IVT-PolyA* which underwent PolyA selection.

6.4.2 Analysis of Wild and Mutant-r2 type *D. melanogaster*

As discussed in section 6.1.3, using the method designed and described in this chapter, we have also analysed the whole transcriptomes of two Drosophila Fruitfly (species *D. melanogaster*) which only differ by mutation *gl[60j]* in the eye-antennal disc (Naval-Sánchez et al., 2013). We analysed two replicates for each of the wild and mutant Fruitfly specimens. Figure 6.14 shows the distribution of exon lengths in the Drosophila genome - the median exon length is 298 bp. This is important because it shows that most of the exons are longer than 200 bp and therefore can have data to compute correlations.

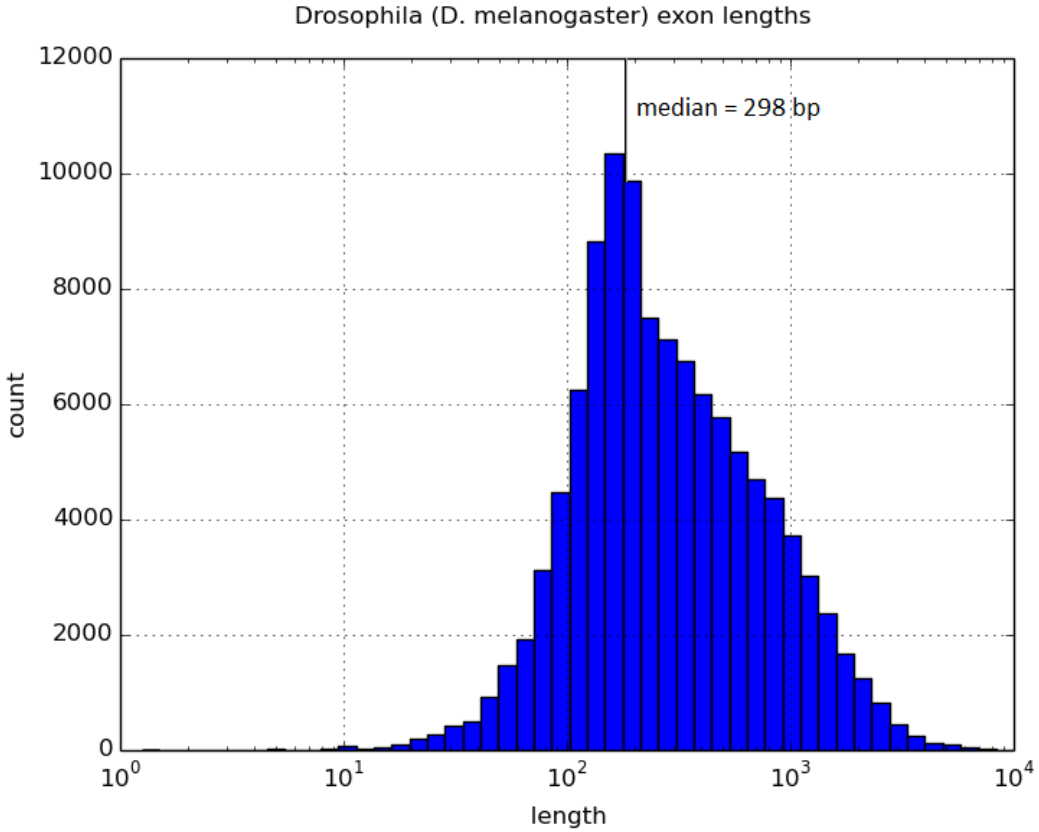


Figure 6.14: Histogram of Drosophila Exon lengths for both wild and mutant-r2 type *D. melanogaster*. The data is computed from the FlyBase genome annotation GTF (Version 6.15) which was used throughout the analysis. The median exon length is 298 bp, mean 540.1 bp, Q1=155 bp, Q3=637 bp, the longest exon is 28070 bp.

6.4.2.1 The Random Hexamer priming effect

As discussed in chapter 3 (sections 3.5 and 3.7), *positional biases* have also been described, whereby positional influence on nucleotide frequencies in nucleotides up to the thirteenth nucleotide from the 5' end of the reads has been observed. This has been demonstrated to be reproducible between experiments and the result of random hexamer priming in sample preparation (Hansen et al., 2010). Hansen et al. make the assertion that the hexamer priming effect is not a result of the actual sequencing process itself, and that excluding the first 10 bp from the 5' end of the reads would not remedy the positional influence of the nucleotide on its frequency. We therefore wanted test whether or not the hexamer priming effect impacts on the correlations of *4-mer* pairs. We have therefore also re-computed correlations with the first 10 bp omitted (from the 5' end) - these are listed alongside the aforementioned tables and are labelled *"(Excluding hexamer primers)"*.

A

Wild type *D. melanogaster*			
Lowest 10 Pearson-correlation outliers and their motifs			
R(10 bp)	R(50 bp)	R(100 bp)	R(200 bp)
TAGG=-0.0151 (539)	CCTA=-0.0074 (403)	GGGG=0.0024 (1991)	GGGG=-0.0187 (1525)
CATA=0.0108 (1150)	GGTT=-0.0051 (1243)	CCTA=0.0066 (627)	CCCC=-0.0043 (1721)
CCGA=0.0250 (1343)	CTAG=-0.0001 (331)	TACC=0.0203 (951)	ACCC=0.0030 (1346)
TAAC=0.0268 (790)	CCGG=0.0225 (2795)	GGTT=0.0259 (2008)	AACC=0.0063 (1559)
CCCC=0.0461 (2356)	CATA=0.0248 (636)	AACC=0.0460 (2028)	CCGG=0.0182 (3351)
CGTA=0.0624 (620)	GATA=0.0479 (664)	CGCG=0.0535 (2417)	GGGT=0.0292 (1310)
CGGG=0.0627 (2534)	TAAC=0.0485 (464)	GGTA=0.0577 (868)	GGTA=0.0350 (680)
GGTA=0.0639 (779)	AGGG=0.0522 (1313)	GGGA=0.0651 (2585)	GTCT=0.0367 (691)
GGGA=0.0710 (2197)	CGCG=0.0635 (1551)	TCCC=0.0728 (2637)	GGTT=0.0384 (1674)
AGGG=0.0832 (1702)	AGCC=0.0744 (1243)	TGGG=0.0800 (3963)	CCCA=0.0392 (3348)
Highest 10 Pearson-correlation outliers and their motifs			
R(10 bp)	R(50 bp)	R(100 bp)	R(200 bp)
TAGA=0.9860 (602)	TAGA=0.8947 (445)	AGTG=0.9289 (1402)	GAGT=0.8516 (975)
ATAG=0.9672 (894)	GTAC=0.8623 (529)	TATG=0.8375 (978)	ACGG=0.7927 (837)
TAGC=0.9325 (729)	TTAG=0.8317 (551)	CTAA=0.8110 (753)	ACTA=0.7202 (596)
AGTG=0.9002 (1368)	AGTG=0.8158 (920)	GCTA=0.8054 (750)	TACG=0.7173 (517)
AACG=0.8898 (1330)	CTAT=0.8059 (533)	CCAT=0.7493 (2080)	TCGT=0.7171 (1541)
ATCC=0.8336 (2306)	ACTA=0.8042 (515)	GCTG=0.7303 (9467)	CTAC=0.7036 (749)
CTGA=0.8192 (1577)	AGCA=0.7986 (4465)	GACT=0.7296 (1149)	ACAG=0.6966 (1360)
CACT=0.8175 (1706)	GTGT=0.7916 (1063)	TAGT=0.7249 (802)	ACCA=0.6960 (2168)
CGAG=0.8028 (2433)	CAGT=0.7776 (1182)	GAGT=0.7149 (1153)	CGTA=0.6860 (560)
AATT=0.8018 (3447)	AGTT=0.7770 (1506)	CGAA=0.7145 (2171)	TGTG=0.6842 (1867)

B

Wild-type *D. melanogaster* (Excluding hexamer primers)			
Lowest 10 Pearson-correlation outliers and their motifs			
R(10 bp)	R(50 bp)	R(100 bp)	R(200 bp)
CCCC=-0.0101 (715)	CCTA=-0.0193 (275)	GGGG=0.0047 (1263)	GGGG=-0.0177 (977)
TAGG=-0.0016 (380)	TCCC=0.0072 (1080)	GGTT=0.0226 (1381)	AACC=-0.0047 (1066)
CCGA=-0.0014 (1385)	GGTT=0.0323 (874)	GGGA=0.0408 (1819)	CCCC=-0.0015 (619)
GGCA=0.0416 (1775)	GGAA=0.0487 (1678)	AACC=0.0476 (1368)	GTTA=0.0201 (425)
TAAC=0.0505 (507)	CTAG=0.0510 (225)	CCTA=0.0708 (435)	ACCC=0.0238 (840)
GGTA=0.0518 (537)	CCGG=0.0516 (1840)	ATAC=0.0789 (733)	GGGT=0.0245 (913)
CCCG=0.0561 (1494)	CATA=0.0518 (438)	CCGC=0.0918 (3589)	GGTA=0.0352 (507)
AGAC=0.0571 (557)	GATA=0.0551 (483)	CATA=0.0927 (744)	GGTT=0.0365 (1169)
ACCG=0.0593 (1067)	TAAG=0.0575 (427)	TGGG=0.0951 (2817)	CCGG=0.0374 (2179)
AGTA=0.0598 (581)	GGGA=0.0656 (1164)	CGGG=0.1009 (1906)	CAGG=0.0533 (1572)
Highest 10 Pearson-correlation outliers and their motifs			
R(10 bp)	R(50 bp)	R(100 bp)	R(200 bp)
TAGA=0.9953 (406)	TTAG=0.9111 (412)	GCTA=0.8767 (557)	GTAG=0.8904 (604)
ATAG=0.9894 (596)	ACTA=0.8665 (382)	TATG=0.8419 (722)	GAGT=0.8409 (744)
GAGT=0.9799 (786)	ATGG=0.8578 (947)	ATAG=0.8360 (557)	ACGG=0.8378 (643)
GTGT=0.9712 (1252)	AGTT=0.8336 (1067)	TACT=0.8348 (625)	TACG=0.8099 (405)
TAGT=0.9540 (507)	GTAC=0.8307 (434)	AGTG=0.8281 (1211)	CAAT=0.7660 (1382)
CAGT=0.9485 (1130)	CTGT=0.7995 (910)	TAGT=0.8259 (576)	ACTA=0.7635 (436)
TGAA=0.9365 (1341)	CCAT=0.7990 (1001)	TCAC=0.8107 (872)	TCAA=0.7628 (1326)
AGCT=0.9224 (1861)	TGCT=0.7990 (2307)	CATT=0.8062 (1748)	GTAC=0.7502 (517)
CACT=0.9084 (1104)	TATG=0.7983 (477)	TGCG=0.7908 (1746)	GATA=0.7292 (587)
CGTA=0.9030 (392)	TAGT=0.7971 (380)	CTAA=0.7866 (531)	AGAC=0.7269 (567)

Table 6.9: *D. melanogaster* wild-type Pearson correlation co-efficient outliers (top ten and lowest ten) for different *intra-exon 4-mer* motif sequence pairs at 10, 50, 100 and 200 bp spacings. A) Includes random hexamer primers. B) Excludes random hexamer primer regions. Sample sizes are given in parenthesis.

A

Mutant-r2 type *D. melanogaster*			
Lowest 10 Pearson-correlation outliers and their motifs			
R(10 bp)	R(50 bp)	R(100 bp)	R(200 bp)
CCTA=-0.0163 (932)	GACT=-0.0208 (1403)	CCTA=-0.0116 (1083)	TACT=-0.0230 (1301)
TCTA=-0.0137 (1346)	GCTA=-0.0146 (1090)	TCCC=-0.0072 (4951)	GTAA=-0.0215 (1321)
TAGG=-0.0112 (944)	TAAC=-0.0124 (1099)	TCTA=-0.0071 (1505)	GGTA=-0.0203 (1135)
CCGA=-0.0083 (2821)	CATA=-0.0120 (1278)	ACTA=-0.0039 (1535)	AACC=-0.0166 (2747)
TAAC=-0.0040 (1623)	CTAC=-0.0114 (1345)	GGGG=0.0025 (4280)	TTAC=-0.0150 (1397)
GTAG=-0.0039 (1736)	GTCT=-0.0108 (1466)	GGGA=0.0091 (4967)	GTGT=-0.0113 (2178)
CATA=0.0017 (2064)	ATAC=-0.0108 (1180)	GATT=0.0103 (4102)	GTTA=-0.0107 (1368)
CTAG=0.0018 (805)	GAGT=-0.0081 (1634)	GTTT=0.0108 (6502)	TCAC=-0.0091 (2027)
GACC=0.0025 (2731)	TAAG=-0.0080 (1201)	ACGT=0.0115 (1450)	TAAC=-0.0081 (1267)
GATA=0.0032 (1784)	CCTA=-0.0073 (701)	CATA=0.0115 (2078)	GTGA=-0.0073 (1933)
Highest 10 Pearson-correlation outliers and their motifs			
R(10 bp)	R(50 bp)	R(100 bp)	R(200 bp)
TAGT=0.7594 (1485)	TACC=0.5456 (1016)	CAAG=0.3938 (3971)	CTTG=0.5247 (3120)
CACT=0.6408 (1754)	ATGT=0.5207 (1390)	CTTG=0.3705 (4073)	GCAG=0.4106 (6614)
GATG=0.5231 (3671)	CTTG=0.4582 (2911)	CTGC=0.3625 (8133)	TGAT=0.3769 (1844)
TGCA=0.4906 (3256)	TTAG=0.4365 (1190)	TGAT=0.3560 (2331)	CAAG=0.3720 (3118)
CAAG=0.4242 (3575)	CAAG=0.3459 (2740)	CATC=0.3467 (3747)	ACAT=0.3496 (1661)
GCAG=0.3975 (8213)	AGTT=0.3399 (1555)	GATG=0.3396 (3686)	CGAA=0.3235 (3096)
AAGG=0.3903 (4231)	TATG=0.3257 (1270)	TATC=0.3316 (2020)	GATC=0.3184 (2394)
TCGA=0.3841 (2228)	TTGT=0.3235 (2925)	ATCA=0.3291 (2324)	CATT=0.3037 (2055)
ACTC=0.3804 (2157)	AACT=0.3221 (1459)	ATCG=0.3053 (2304)	TGTG=0.3024 (1977)
AATG=0.3743 (2211)	GATG=0.3199 (2601)	CTCG=0.2964 (2798)	ACGG=0.3001 (2036)

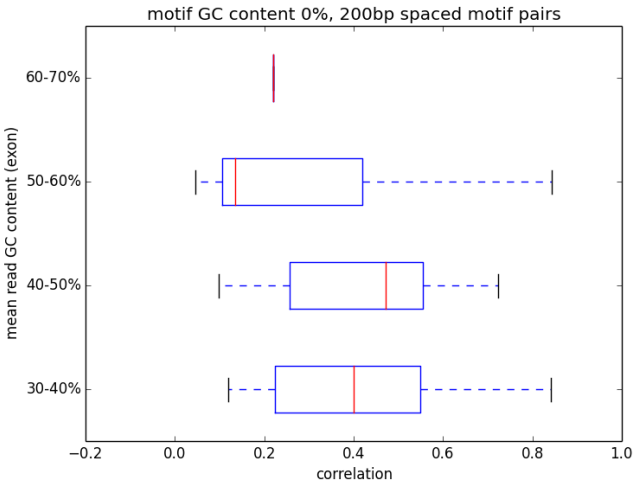
B

Mutant-r2 type *D. melanogaster* (Excluding hexamer primers)			
Lowest 10 Pearson-correlation outliers and their motifs			
R(10 bp)	R(50 bp)	R(100 bp)	R(200 bp)
CCTA=-0.0162 (572)	GACT=-0.0287 (1036)	CCTA=-0.0157 (730)	GTAA=-0.0249 (907)
TCTA=-0.0142 (865)	TAAC=-0.0143 (760)	ACTA=-0.0095 (1104)	AACC=-0.0182 (1761)
TAGG=-0.0099 (567)	GCTA=-0.0128 (794)	TCCC=-0.0042 (3001)	TACT=-0.0156 (913)
CTAC=-0.0095 (1136)	CATA=-0.0127 (917)	GGGA=-0.0037 (3418)	GTGT=-0.0104 (1415)
GACC=-0.0092 (1693)	CTAG=-0.0121 (405)	ATAC=0.0041 (1334)	TAAC=-0.0100 (919)
ACGT=-0.0079 (758)	CCTA=-0.0088 (450)	GGGG=0.0065 (2660)	TCAC=-0.0091 (1397)
CCGA=-0.0071 (1899)	ACTC=-0.0068 (1248)	GATT=0.0075 (2913)	TTAC=-0.0086 (952)
TAAC=-0.0066 (1055)	TAAG=-0.0061 (825)	CATA=0.0133 (1467)	TAGT=-0.0079 (859)
CCCC=-0.0012 (1996)	GTTA=-0.0036 (778)	TAAG=0.0161 (1181)	GGTA=-0.0058 (815)
GGTA=-0.0010 (800)	CTAC=-0.0030 (946)	CGTA=0.0176 (935)	ACCC=-0.0046 (1650)
Highest 10 Pearson-correlation outliers and their motifs			
R(10 bp)	R(50 bp)	R(100 bp)	R(200 bp)
CACT=0.5970 (1203)	TTAG=0.6797 (860)	CTGC=0.5069 (5354)	CTTG=0.5398 (2510)
TAGT=0.5784 (958)	AGTG=0.6621 (865)	TGAT=0.4771 (1766)	ATCT=0.5105 (1268)
ATCC=0.5545 (2744)	CGAT=0.5523 (1170)	CATC=0.4407 (2818)	GATC=0.4877 (1859)
CGAG=0.4880 (1706)	ATCT=0.5508 (1037)	AAGT=0.4335 (1556)	ACAT=0.4623 (1279)
AGTG=0.4433 (1173)	AACT=0.5247 (1103)	CATT=0.4153 (2020)	GCAG=0.4235 (4324)
TACT=0.4261 (1001)	ATGT=0.5102 (1076)	ACAT=0.4138 (1600)	ATCG=0.4184 (1374)
GATG=0.4180 (2289)	TATG=0.5052 (907)	CAAG=0.4104 (3139)	ACTG=0.3934 (1424)
TGTT=0.4163 (2695)	CTTG=0.4830 (2256)	CTCG=0.4018 (2164)	CGTG=0.3799 (930)
TCCA=0.4153 (2723)	AGTT=0.4709 (1174)	ATCA=0.3960 (1734)	CATT=0.3781 (1603)
CACA=0.4016 (2056)	ACAT=0.4561 (999)	TCGA=0.3794 (1918)	TGAT=0.3775 (1426)

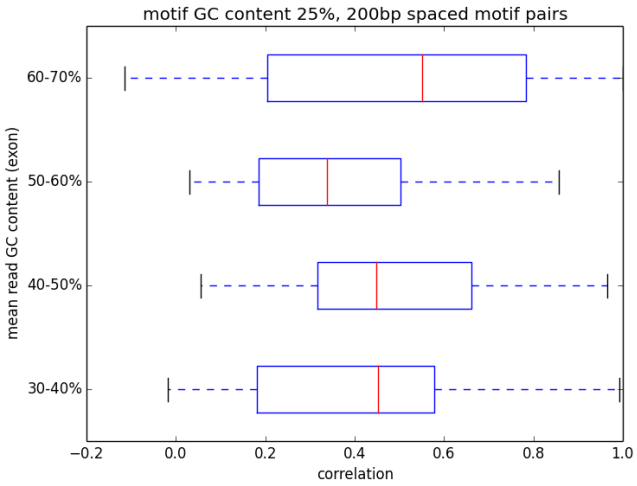
Table 6.10: *D. melanogaster* Mutant-r2-type Pearson correlation co-efficient outliers (top ten and lowest ten) for different *4-mer* motif sequence pairs at 10, 50, 100 and 200 bp spacings. A) Includes random hexamer primers. B) Excludes random hexamer primer regions. Sample sizes are given in parenthesis.

We can see that when the regions on the 5' end of the reads affected by random hexamer priming have been excluded from the correlations (right-hand side of Table 6.9 and Table 6.10 as compared with their corresponding left-hand sides) there is is very little effect on the motif-pair correlations in both the wild and mutant datasets. Furthermore, for each spacing, many of the same *4-mer* motif sequences that are observed in the lowest 10 outlier correlations are also observed as lowest outliers in the data where the hexamer priming regions have been excluded. We have also produced box and whisker plots which partition the correlations as a function of GC content for both the normal correlation data and the data excluding the hexamer priming regions. This is in order to explore the effect of these factors (GC content of the motif, GC content of the exon, and hexamer primer region) on the uniformity of mapped reads as designated by the correlation score between *intra-exon* motif pairs.

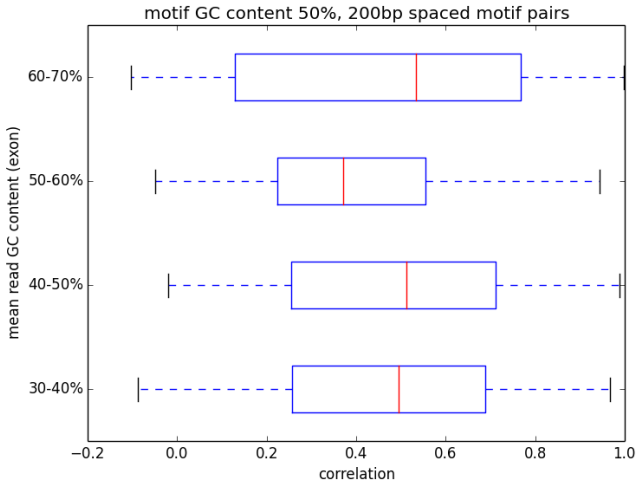
We observe that there is no difference in correlations between the normal data and the data in which the hexamer priming regions have been excluded - this is depicted in box-whisker plots for the wild-type data set at 200 bp (Figure 6.15 vs. Figure 6.16). These shows the correlations remain more or less the same for both groups, and across all motif GC and mean exon GC concentrations. The same lack of difference between the two aforementioned groups is observed for all motif-pair spacings in both replicates of the mutant-r2 and wild type dataset (plots are in the Appendix (Alnasir, 2018)). This suggests, as Hansen et al. (2010) postulate, that the random hexamer priming effect which results in predictable nucleotide frequencies, is not a result of sequencing but rather the preparatory protocols, and that this has no bearing on the *intra-exon* correlation of *4-mer* motif pairs in the datasets we examined. We will therefore focus on correlation as a function of GC content parameters in the remainder of the analysis.



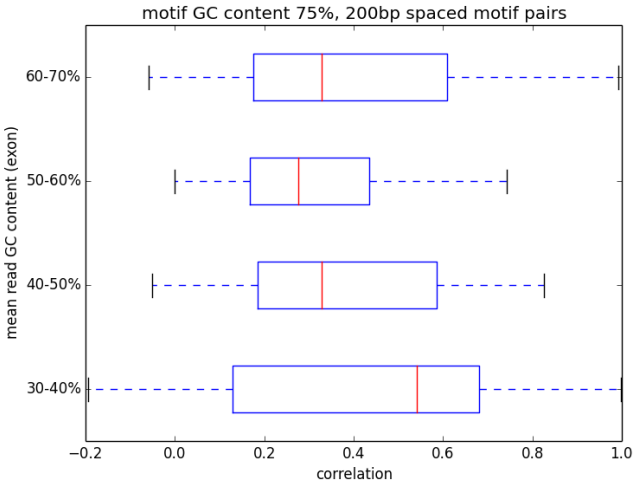
(a) Motif GC content of 0%



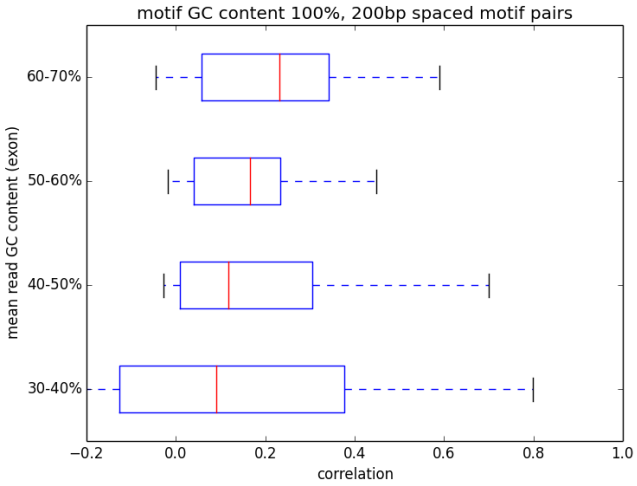
(b) Motif GC content of 25%



(c) Motif GC content of 50%

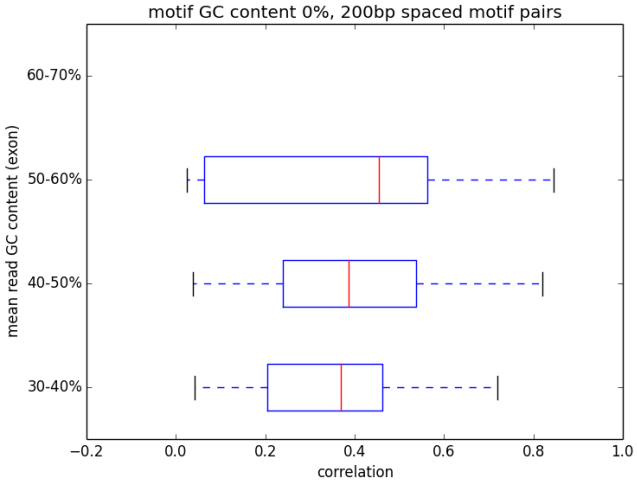


(d) Motif GC content of 75%

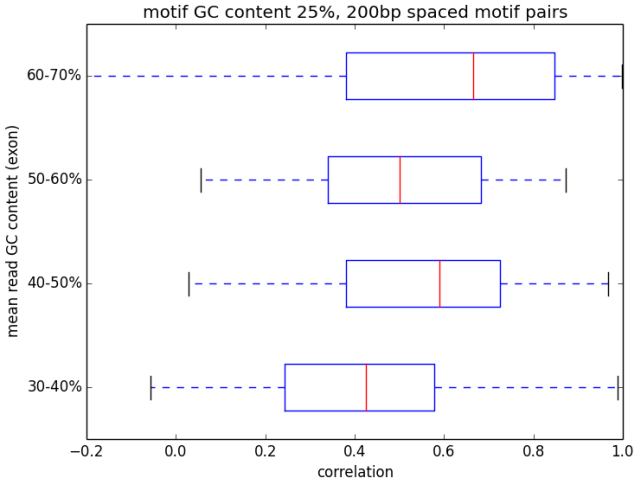


(e) Motif GC content of 100%

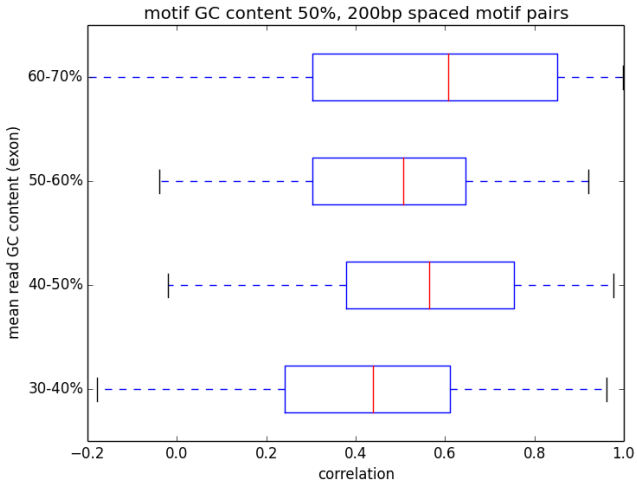
Figure 6.15: Box and whisker plots of motif-pair correlations at a distance of 200 bp for Wild-type *D. melanogaster*.



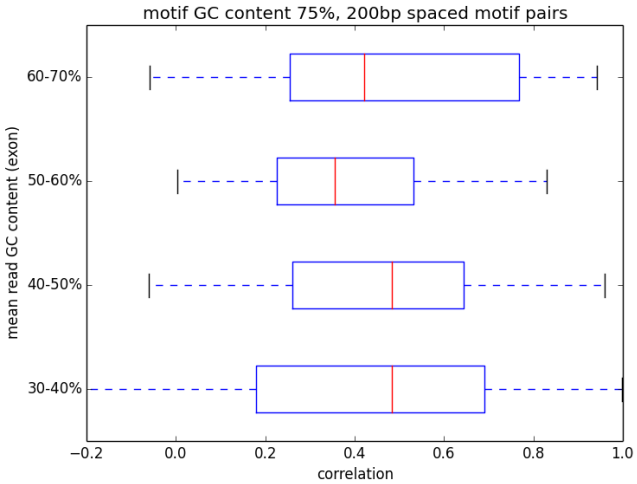
(a) Motif GC content of 0%



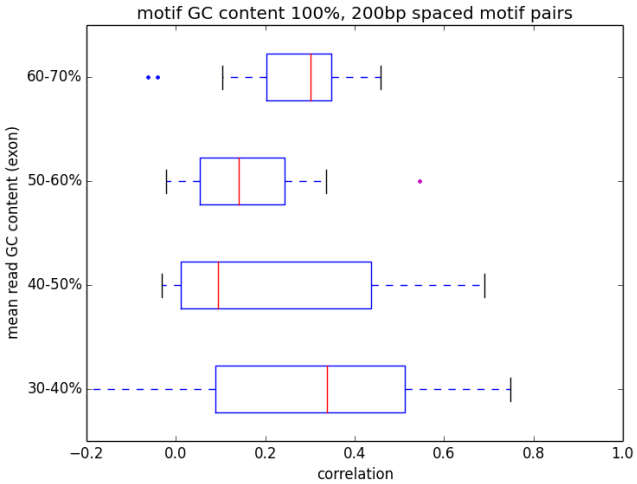
(b) Motif GC content of 25%



(c) Motif GC content of 50%



(d) Motif GC content of 75%



(e) Motif GC content of 100%

Figure 6.16: Box and whisker plots of motif pair correlations at a distance of 200 bp for Wild-type *D. melanogaster* (hexamer region excluded).

6.4.2.2 Spearman's rank - assessing non-linear relationships

In addition to the Pearson correlation calculation, which as discussed measures linear correlation, we have also computed the Spearman's rank correlation for outliers in the RNA-Seq datasets. This is because Spearman's rank correlation measures monotonic correlation - a relationship between variables in which as the value of one variable increases the other also increases, or as the value of one variable increases the value of the other decreases, but importantly in both cases this follows a less linear trend (than measured by Pearson's). This allows us to assess non-linear relationships. Nonetheless, Spearman's rank correlation for outliers in the wild and mutant-r2 datasets were computed and these obtained very similar results, these are listed in Table 6.11. We will therefore focus on Pearson correlations in the remainder of the data.

A

Wild type *D. melanogaster*			
Lowest 10 Spearman's rank outliers and their motifs			
R(10 bp)	R(50 bp)	R(100 bp)	R(200 bp)
GGGG=0.1154 (2001)	CCCT=0.0809 (1429)	GGGG=0.0383 (1991)	GGGG=0.0465 (1525)
CCCC=0.1297 (2356)	GGGG=0.0810 (1320)	CCCC=0.0395 (2338)	CCCC=0.0757 (1721)
CCCG=0.1493 (2697)	CGGG=0.0834 (2102)	CCCT=0.0624 (2096)	CCCG=0.0773 (2532)
ACCC=0.1808 (1592)	AGGG=0.0854 (1313)	ACCC=0.0790 (1688)	GCCC=0.0826 (3571)
CGGG=0.1881 (2534)	ACCC=0.0862 (1171)	CCCG=0.0833 (3244)	ACCC=0.0876 (1346)
CCCT=0.1922 (1879)	CCCG=0.0987 (2117)	AGGG=0.0900 (2026)	CCCT=0.0899 (1676)
AGGG=0.2117 (1702)	TCCC=0.1168 (1768)	CGCG=0.1004 (2417)	CGGG=0.1080 (2462)
GCCC=0.2279 (3931)	GGGA=0.1254 (1681)	TCCC=0.1032 (2637)	GGGC=0.1093 (3433)
TCCC=0.2338 (2394)	GCCC=0.1311 (2935)	CGGG=0.1054 (3343)	GGGA=0.1213 (2006)
GGGC=0.2340 (3853)	CCCC=0.1522 (1516)	GGGA=0.1088 (2585)	CGCG=0.1247 (1800)
Highest 10 Spearman's rank outliers and their motifs			
R(10 bp)	R(50 bp)	R(100 bp)	R(200 bp)
AGTG=0.8538 (1368)	CGTA=0.6616 (482)	AGTC=0.6273 (1121)	CGTG=0.6155 (1080)
ACAG=0.8500 (1627)	GACT=0.6567 (771)	CTTG=0.6269 (3793)	CAAG=0.6113 (3038)
TTAA=0.8492 (2906)	CTAA=0.6467 (505)	AGCT=0.6138 (3200)	CTTA=0.6071 (611)
CTAA=0.8453 (733)	CAAG=0.6465 (2594)	GCTA=0.6112 (750)	CTTG=0.6064 (3014)
TCTA=0.8412 (648)	CTTG=0.6387 (2778)	ACTC=0.6104 (1230)	TACG=0.6064 (517)
AGCT=0.8394 (2852)	ATAG=0.6370 (482)	CAAG=0.6055 (3774)	CACT=0.6021 (1408)
GTAA=0.8373 (771)	TAAG=0.6246 (560)	GATA=0.6040 (959)	GTAC=0.5998 (652)
ATAG=0.8372 (894)	TAGA=0.6197 (445)	GAGT=0.6005 (1153)	ACGT=0.5986 (492)
GCTA=0.8352 (653)	TAGT=0.6184 (542)	CGAG=0.5997 (2580)	AGTC=0.5967 (899)
TAAA=0.8352 (3691)	TACT=0.6141 (565)	CTAC=0.5940 (1002)	CTAC=0.5950 (749)

B

Mutant-r2 type *D. melanogaster*			
Lowest 10 Spearman's rank outliers and their motifs			
R(10 bp)	R(50 bp)	R(100 bp)	R(200 bp)
GGGG=0.0854 (4124)	GCCC=0.0241 (5162)	GGGG=0.0212 (4280)	GGGT=0.0088 (2729)
GCCC=0.0896 (6654)	TCCC=0.0374 (3261)	AGGG=0.0349 (3883)	AGGG=0.0160 (2882)
GGGT=0.0931 (3210)	CCCG=0.0442 (3315)	GGGC=0.0365 (7890)	ACCC=0.0300 (2759)
AGGG=0.1035 (3319)	GGGA=0.0478 (3247)	TCCC=0.0396 (4951)	GCCC=0.0399 (6101)
GGGC=0.1144 (6493)	ACCC=0.0503 (2330)	GGGA=0.0410 (4967)	CCCT=0.0416 (3041)
CCCT=0.1205 (3473)	GGGC=0.0508 (5046)	GGGT=0.0505 (3426)	GGGC=0.0480 (5943)
TCCC=0.1216 (4190)	CCCT=0.0573 (2590)	GCCC=0.0521 (8147)	CGGG=0.0535 (3827)
GGCC=0.1283 (6933)	AGGG=0.0643 (2534)	TGGG=0.0593 (6421)	CCCA=0.0535 (5479)
GGGA=0.1308 (4012)	GGGT=0.0650 (2223)	ACCC=0.0597 (3512)	TGGG=0.0574 (5180)
CGGG=0.1343 (3886)	CGGG=0.0656 (3287)	CCCT=0.0605 (3899)	TCCC=0.0630 (3814)
Highest 10 Spearman's rank outliers and their motifs			
R(10 bp)	R(50 bp)	R(100 bp)	R(200 bp)
ACAT=0.8316 (2095)	ACTG=0.6437 (1418)	CAGT=0.6202 (2216)	CTTG=0.6362 (3120)
ATTG=0.8217 (2392)	AGTG=0.6104 (1139)	CTTG=0.5943 (4073)	CACT=0.6182 (1483)
CACT=0.8212 (1754)	CACT=0.6099 (1138)	CACT=0.5938 (1828)	CAGT=0.6094 (1756)
AATG=0.8181 (2211)	CAAG=0.6086 (2740)	TCGT=0.5919 (2185)	ACTG=0.6050 (1807)
CATG=0.8162 (1780)	CAAT=0.6025 (1781)	AGTG=0.5907 (1757)	CTCG=0.5884 (2019)
AGTG=0.8160 (1678)	CTTG=0.5986 (2911)	AGCT=0.5899 (3344)	AGTT=0.5842 (1835)
ATGT=0.8147 (2011)	ACAT=0.5979 (1350)	ACTG=0.5860 (2198)	CAAG=0.5816 (3118)
ACTG=0.8124 (1939)	CATG=0.5959 (1247)	CTCG=0.5799 (2798)	CGAT=0.5814 (1796)
AGTT=0.8048 (2091)	CATT=0.5946 (1705)	CAAT=0.5787 (2594)	TCGT=0.5725 (1604)
CAGT=0.8033 (1930)	CAGT=0.5905 (1478)	CAAG=0.5762 (3971)	ATTG=0.5656 (2146)

Table 6.11: Spearman's rank outliers in *D. melanogaster* (top ten and lowest ten) for different *4-mer* motif sequence pairs at 10, 50, 100 and 200 bp spacings. A) Correlations for Wild-type dataset B) Correlations for Mutant-r2 type dataset. Sample sizes are given in parenthesis.

The box-plots of Spearman's rank show that the medians are approximately the same as those corresponding to the Pearson correlation coefficients, but that the Spearman's rank correlation for the wild-type dataset showed narrower spread than the Pearson correlation for wild-type (Figure 6.18).

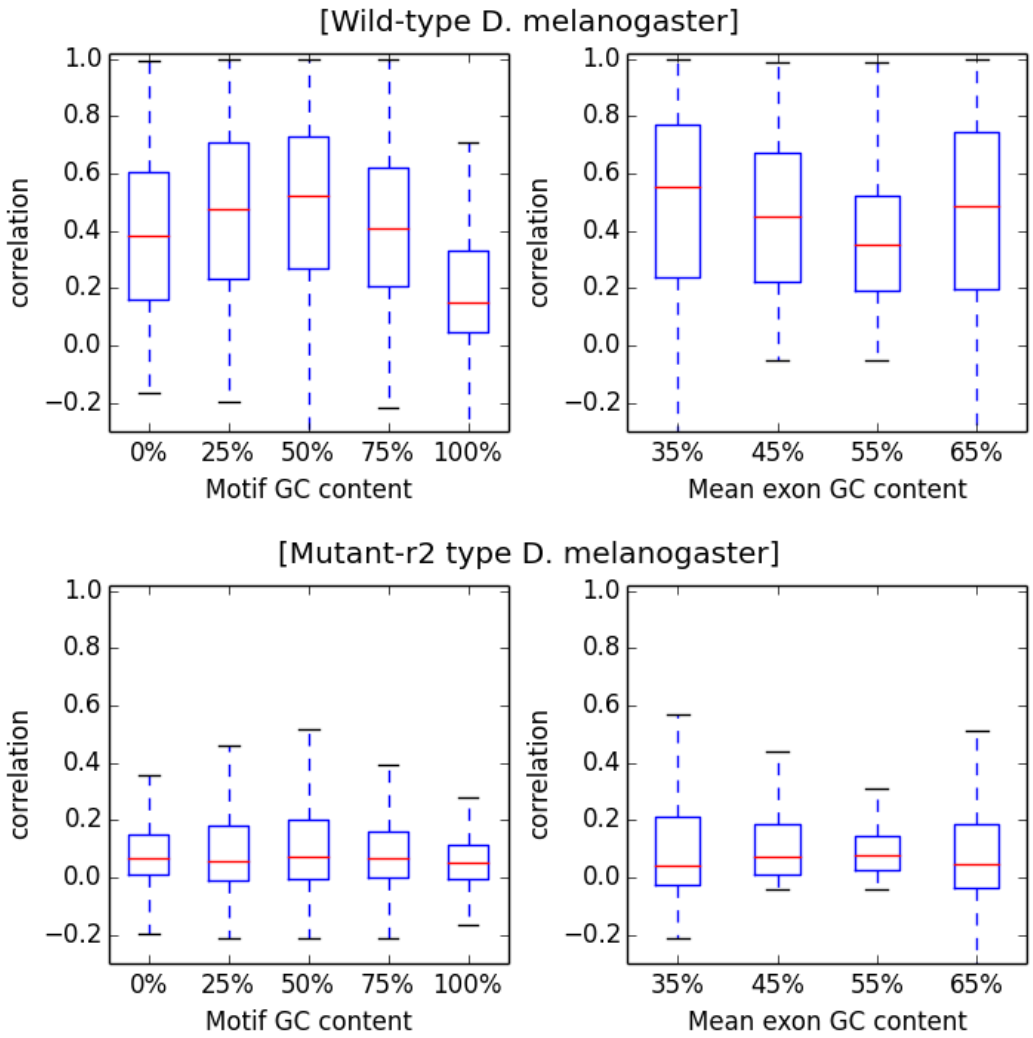


Figure 6.17: Correlation (Pearson's) as a function of *4-mer* motif and exon GC content in both wild and mutant-r2 *D. melanogaster* transcriptomes.

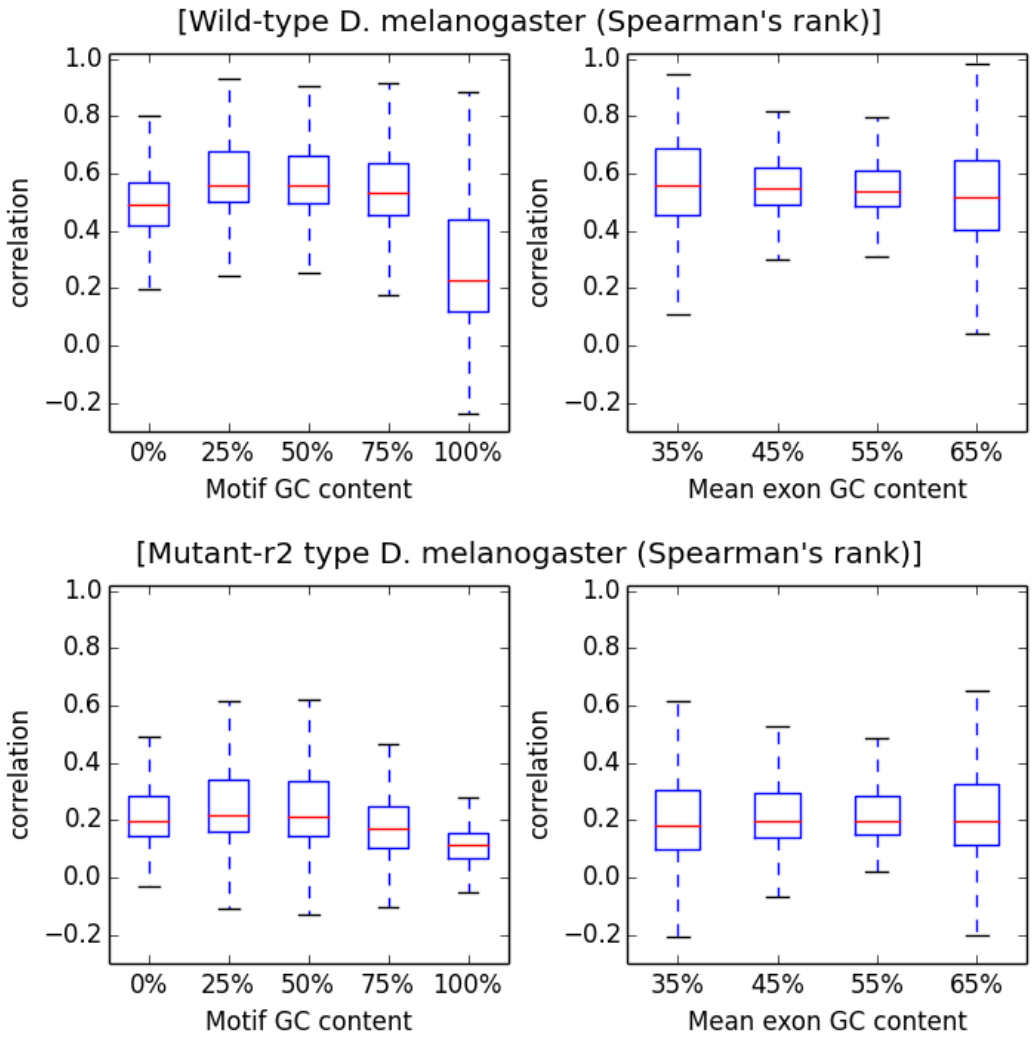


Figure 6.18: Correlation (Spearman's rank) as a function of *4-mer* motif and exon GC content in both wild and mutant-r2 *D. melanogaster* transcriptomes.

6.4.2.3 GC content of the replicates

In order to examine the effect of GC content on the distribution of mapped reads, we have plotted intra-exon *4-mer* motif-pair correlations as a function of both motif and exon GC content, for both *D. melanogaster* datasets (Figure 6.17). In the wild-type dataset we observe notable variation in the correlation as a function of GC content of the motif, whereas no variation in the correlation is observed as a function of the mean exon GC content. This indicates that the GC content of the motif has an effect (causing a deviation in the distribution of mapped reads to an exon) rather than the overall GC content. In the mutant-r2 type dataset, no variation is observed.

A

Wild-r2 type *D. melanogaster*			
Lowest 10 Pearson-correlation outliers and their motifs			
R(10 bp)	R(50 bp)	R(100 bp)	R(200 bp)
TAGG=-0.0154 (534)	CCTA=-0.0078 (392)	GGGG=0.0023 (1977)	GGGG=-0.0188 (1507)
CATA=0.0100 (1141)	CTAG=0.0005 (320)	CCTA=0.0065 (611)	CCCC=-0.0045 (1699)
CCGA=0.0248 (1337)	GGTT=0.0173 (757)	CGCG=0.0533 (2398)	ACCC=0.0030 (1332)
TAAC=0.0267 (781)	CATA=0.0204 (630)	GGGA=0.0648 (2538)	CCGG=0.0185 (3330)
CCCC=0.0460 (2346)	CCGG=0.0224 (2776)	AACC=0.0713 (1206)	GGGT=0.0288 (1286)
CGTA=0.0622 (613)	GATA=0.0479 (662)	TCCC=0.0736 (2599)	GTCT=0.0370 (684)
CGGG=0.0625 (2519)	TAAC=0.0484 (456)	TGGG=0.0797 (3909)	CCCA=0.0391 (3305)
GGGA=0.0713 (2185)	AGGG=0.0520 (1302)	CGGG=0.0904 (3318)	AACC=0.0425 (957)
AGGG=0.0831 (1690)	CGCG=0.0629 (1524)	TCTC=0.0930 (1669)	AGGG=0.0479 (1509)
CGTT=0.0844 (1242)	AGCC=0.0747 (1237)	CATA=0.1012 (1018)	ATAG=0.0550 (532)
Highest 10 Pearson-correlation outliers and their motifs			
R(10 bp)	R(50 bp)	R(100 bp)	R(200 bp)
TAGA=0.9860 (600)	TAGA=0.9033 (440)	TATG=0.8378 (964)	GAGT=0.8517 (961)
ATAG=0.9672 (892)	TACC=0.8850 (336)	GCTA=0.8293 (743)	ACGG=0.7927 (836)
TAGC=0.9325 (727)	GTAC=0.8623 (523)	CTAA=0.8113 (738)	ACTA=0.7205 (588)
AACG=0.8898 (1322)	GGTA=0.8370 (357)	TAGT=0.7988 (787)	TCGT=0.7189 (1534)
AGTG=0.8618 (1575)	TTAG=0.8317 (544)	CCAT=0.7495 (2063)	TACG=0.7173 (514)
ATCC=0.8336 (2296)	AGTG=0.8219 (1072)	GCTG=0.7304 (9423)	CTAC=0.7047 (740)
CTGA=0.8193 (1555)	CTAT=0.8069 (518)	AGTG=0.7304 (1610)	ACCA=0.6961 (2155)
CACT=0.8175 (1694)	ACTA=0.8049 (509)	GACT=0.7297 (1132)	CGTA=0.6874 (552)
CGAG=0.8028 (2426)	AGCA=0.7988 (4445)	CGAA=0.7159 (2143)	TGTG=0.6844 (1849)
AATT=0.8019 (3416)	TCTA=0.7966 (342)	GAGT=0.7149 (1141)	GAAA=0.6757 (2726)

B

Mutant-r3 type *D. melanogaster*			
Lowest 10 Pearson-correlation outliers and their motifs			
R(10 bp)	R(50 bp)	R(100 bp)	R(200 bp)
GTAG=-0.0110 (4451)	GACT=-0.0155 (3709)	CCTA=-0.0079 (3680)	TACT=-0.0167 (3730)
TCTA=-0.0109 (4014)	ATAC=-0.0121 (3061)	ACTA=-0.0067 (3991)	TTAC=-0.0142 (3633)
CCTA=-0.0099 (3058)	CATA=-0.0108 (3062)	ACGT=-0.0028 (4259)	GGTA=-0.0133 (3456)
TAGG=-0.0090 (3009)	TAAC=-0.0100 (2946)	TCCC=-0.0013 (14111)	AGTG=-0.0114 (5061)
CCGA=-0.0085 (7526)	TAAG=-0.0086 (3286)	TCTA=-0.0010 (5028)	TCAC=-0.0113 (5532)
TAAC=-0.0067 (3901)	TCAT=-0.0073 (5180)	TAGA=-0.0007 (4971)	GTTA=-0.0104 (3718)
CGTA=-0.0057 (2955)	GCTA=-0.0069 (3236)	CGTA=0.0014 (3747)	CTCA=-0.0095 (5707)
GATA=-0.0045 (4236)	GATA=-0.0059 (3368)	ACGC=0.0019 (7405)	GTAA=-0.0093 (3437)
CTAC=-0.0034 (4526)	ATTG=-0.0058 (5479)	GATT=0.0027 (8208)	GTGT=-0.0092 (5366)
CTAG=-0.0025 (3236)	CCTA=-0.0058 (2559)	TAGT=0.0027 (3768)	AACC=-0.0084 (7436)
Highest 10 Pearson-correlation outliers and their motifs			
R(10 bp)	R(50 bp)	R(100 bp)	R(200 bp)
TAGT=0.5918 (3456)	TACC=0.2893 (2881)	TAAC=0.1998 (4402)	CGAA=0.1921 (7254)
AAGG=0.2979 (9354)	TAGC=0.2644 (3270)	CAAG=0.1738 (12860)	ACGG=0.1903 (5536)
CACT=0.2686 (5633)	CTTG=0.1926 (8830)	GCGT=0.1731 (7475)	CAAG=0.1680 (10024)
TATC=0.2670 (4379)	GTGT=0.1847 (4722)	TATC=0.1621 (5203)	TCAA=0.1667 (8631)
TGGA=0.2447 (13432)	TATG=0.1786 (3069)	AACA=0.1608 (14808)	CCCT=0.1633 (9198)
CAAG=0.2332 (10781)	CGGA=0.1735 (7791)	CCCG=0.1565 (14386)	AGGA=0.1460 (11653)
ATCC=0.2281 (8329)	CAAG=0.1729 (8732)	ATCG=0.1526 (7561)	CTTA=0.1453 (3943)
CCCT=0.2256 (9530)	GTTT=0.1697 (8514)	ACCA=0.1421 (13261)	CTTG=0.1422 (10212)
AATT=0.2210 (10274)	AATT=0.1612 (7287)	CTAG=0.1413 (4125)	GTAC=0.1415 (3391)
TACG=0.2165 (2974)	GGTA=0.1547 (2923)	CCAA=0.1409 (14643)	TACC=0.1394 (3586)

Table 6.12: *D. melanogaster* Pearson correlation co-efficient outliers (top ten and lowest ten) for different *intra-exon 4-mer* motif sequence pairs at 10, 50, 100 and 200 bp spacings. A) 2nd Replicate from the wild-type dataset B) 2nd Replicate from the mutant-type dataset. Sample sizes are given in parenthesis.

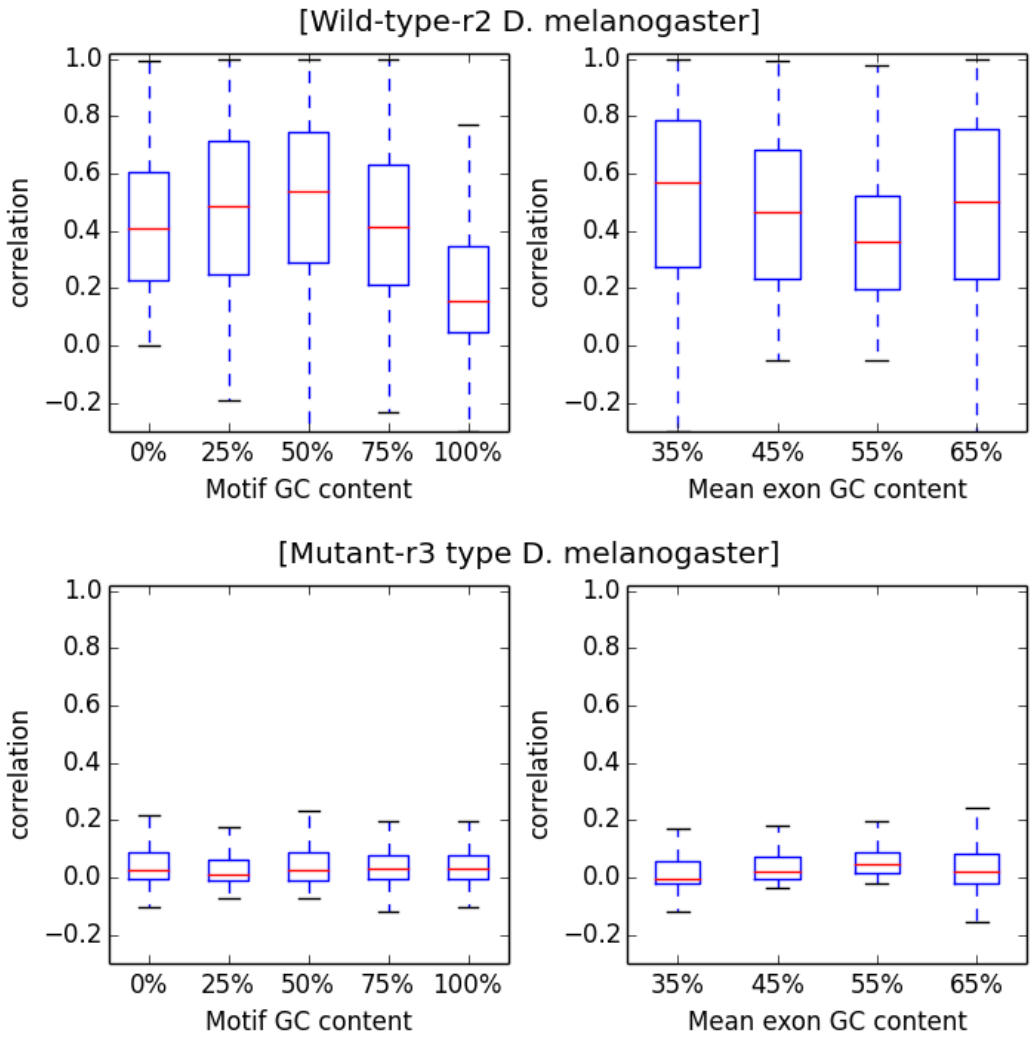


Figure 6.19: Correlation (Pearson's) as a function of *4-mer* motif and exon GC content for 2nd replicates of both wild and mutant-r2 *D. melanogaster* transcriptomes.

6.4.2.4 T-test and Wilcoxon tests on wild and mutant type D. Melanogaster datasets

Investigating correlation as a function of GC content of the motif *gm* further, we computed Welch's t-test and Wilcoxon's tests to statistically compare the partitioned correlations, and to test significance of the variation for each spacing of *intra-exon 4-mer* motif pairs. We define a null hypothesis, H_0 ,that there is no difference in correlation between for Motif GC *gm* of 50% vs. the Motif GC being tested. Table 6.13, below, shows the results of these statistical tests for motif pairs spaced at 200 bp. We can see significant variation in the FDR corrected p-values (discussed in section 6.3.5) between the Motif GC *gm* of 50% and Motif GC concentrations of 75% and 100% across all Exon GC concentrations $\overline{ge}$, which is indicated by asterisks that mean we can reject the null hypothesis. We also observed similar results for those same comparisons for motif pairs spaced at 10, 50 and 100bp (Appendix (Alnasir, 2018)). We note that we see a significant pattern of difference for motif GC of 100% in the wild-type data (Table 6.13) which is not seen in the mutant-r2 type dataset (Table 6.14).

Motif spacing: **200 bp**								
Exon GC%	**30-40%**		**40-50%**		**50-60%**		**60-70%**	
Motif GC%	50	0	50	0	50	0	50	0
#Correlations	16	96	16	96	16	96	1	96
p(t-test)	$3.32\text{x}10^{-1}(4.65\text{x}10^{-1})$		$3.40\text{x}10^{-1}(4.66\text{x}10^{-1})$		$1.30\text{x}10^{-1}(2.23\text{x}10^{-1})$		insufficient	
p(Wilcoxon)	$1.09\text{x}10^{-1}(1.91\text{x}10^{-1})$		$3.79\text{x}10^{-1}(5.09\text{x}10^{-1})$		$1.34\text{x}10^{-1}(2.28\text{x}10^{-1})$		data	
Exon GC%	**30-40%**		**40-50%**		**50-60%**		**60-70%**	
Motif GC%	50	25	50	25	50	25	50	25
#Correlations	64	96	64	96	64	96	47	96
p(t-test)	$2.03\text{x}10^{-1}(3.12\text{x}10^{-1})$		$9.22\text{x}10^{-1}(9.68\text{x}10^{-1})$		$1.50\text{x}10^{-1}(2.43\text{x}10^{-1})$		$9.68\text{x}10^{-1}(9.91\text{x}10^{-1})$	
p(Wilcoxon)	$3.53\text{x}10^{-1}(4.83\text{x}10^{-1})$		$7.58\text{x}10^{-1}(8.69\text{x}10^{-1})$		$2.09\text{x}10^{-1}(3.06\text{x}10^{-1})$		$8.24\text{x}10^{-1}(8.95\text{x}10^{-1})$	
Exon GC%	**30-40%**		**40-50%**		**50-60%**		**60-70%**	
Motif GC%	50	75	50	75	50	75	50	75
#Correlations	60	96	64	96	64	96	64	96
p(t-test)	$6.78\text{x}10^{-1}(7.63\text{x}10^{-1})$		$3.97\text{x}10^{-3}(1.32\text{x}10^{-2})$*		$7.37\text{x}10^{-3}(2.01\text{x}10^{-2})$*		$7.15\text{x}10^{-2}(1.37\text{x}10^{-1})$	
p(Wilcoxon)	$9.18\text{x}10^{-1}(9.64\text{x}10^{-1})$		$1.00\text{x}10^{-2}(3.95\text{x}10^{-2})$*		$2.78\text{x}10^{-2}(7.00\text{x}10^{-2})$		$2.97\text{x}10^{-2}(7.21\text{x}10^{-2})$	
Exon GC%	**30-40%**		**40-50%**		**50-60%**		**60-70%**	
Motif GC%	50	100	50	100	50	100	50	100
#Correlations	16	96	16	96	16	96	16	96
p(t-test)	$2.75\text{x}10^{-3}(9.63\text{x}10^{-3})$*		$2.84\text{x}10^{-4}(1.49\text{x}10^{-3})$*		$9.10\text{x}10^{-7}(1.43\text{x}10^{-5})$*		$8.96\text{x}10^{-5}(6.08\text{x}10^{-4})$*	
p(Wilcoxon)	$5.23\text{x}10^{-3}(2.84\text{x}10^{-2})$*		$7.03\text{x}10^{-2}(1.53\text{x}10^{-1})$		$3.78\text{x}10^{-3}(2.84\text{x}10^{-2})$*		$2.00\text{x}10^{-2}(5.47\text{x}10^{-2})$	

Table 6.13: T-test and Wilcoxon-test comparisons of Pearson correlations for motif-pairs at 200 bp spacing for varying motif GC and mean exon GC content in Wild-type *D. melanogaster*. FDR corrected p-values in parenthesis, using a False positive rate of 5% ($\alpha = 0.05$). * suggests rejection of the null hypothesis (i.e. that there is no difference in correlation between for Motif GC of 50% vs. the Motif GC being tested.).

Motif spacing: **200 bp**								
Exon GC%	**30-40%**		**40-50%**		**50-60%**		**60-70%**	
Motif GC%	50	0	50	0	50	0	50	0
#Correlations	16	96	16	96	16	96	13	96
p(t-test)	$6.07\text{x}10^{-1}(7.94\text{x}10^{-1})$		$3.26\text{x}10^{-1}(5.97\text{x}10^{-1})$		$3.60\text{x}10^{-1}(6.33\text{x}10^{-1})$		$6.23\text{x}10^{-1}(7.98\text{x}10^{-1})$	
p(Wilcoxon)	$9.59\text{x}10^{-1}(9.59\text{x}10^{-1})$		$6.79\text{x}10^{-1}(8.05\text{x}10^{-1})$		$2.55\text{x}10^{-1}(5.63\text{x}10^{-1})$		$6.00\text{x}10^{-1}(7.90\text{x}10^{-1})$	
Exon GC%	**30-40%**		**40-50%**		**50-60%**		**60-70%**	
Motif GC%	50	25	50	25	50	25	50	25
#Correlations	64	96	64	96	64	96	63	96
p(t-test)	$6.08\text{x}10^{-1}(7.94\text{x}10^{-1})$		$8.93\text{x}10^{-1}(9.68\text{x}10^{-1})$		$8.18\text{x}10^{-2}(3.27\text{x}10^{-1})$		$9.68\text{x}10^{-1}(9.68\text{x}10^{-1})$	
p(Wilcoxon)	$5.70\text{x}10^{-1}(7.90\text{x}10^{-1})$		$5.12\text{x}10^{-1}(7.90\text{x}10^{-1})$		$5.67\text{x}10^{-2}(3.34\text{x}10^{-1})$		$7.17\text{x}10^{-1}(8.20\text{x}10^{-1})$	
Exon GC%	**30-40%**		**40-50%**		**50-60%**		**60-70%**	
Motif GC%	50	75	50	75	50	75	50	75
#Correlations	64	96	64	96	64	96	64	96
p(t-test)	$2.54\text{x}10^{-1}(5.60\text{x}10^{-1})$		$1.52\text{x}10^{-2}(9.73\text{x}10^{-2})$		$6.01\text{x}10^{-1}(7.94\text{x}10^{-1})$		$3.66\text{x}10^{-1}(6.33\text{x}10^{-1})$	
p(Wilcoxon)	$6.40\text{x}10^{-1}(7.90\text{x}10^{-1})$		$9.32\text{x}10^{-2}(3.73\text{x}10^{-1})$		$6.25\text{x}10^{-1}(7.90\text{x}10^{-1})$		$5.88\text{x}10^{-1}(7.90\text{x}10^{-1})$	
Exon GC%	**30-40%**		**40-50%**		**50-60%**		**60-70%**	
Motif GC%	50	100	50	100	50	100	50	100
#Correlations	16	96	16	96	16	96	16	96
p(t-test)	$5.47\text{x}10^{-1}(7.78\text{x}10^{-1})$		$1.06\text{x}10^{-1}(3.90\text{x}10^{-1})$		$5.52\text{x}10^{-2}(2.52\text{x}10^{-1})$		$4.57\text{x}10^{-1}(7.23\text{x}10^{-1})$	
p(Wilcoxon)	$7.17\text{x}10^{-1}(8.20\text{x}10^{-1})$		$3.52\text{x}10^{-1}(6.63\text{x}10^{-1})$		$6.42\text{x}10^{-1}(7.90\text{x}10^{-1})$		$9.18\text{x}10^{-1}(9.47\text{x}10^{-1})$	

Table 6.14: T-test and Wilcoxon-test comparisons of Pearson correlations for motif-pairs at 200 bp spacing for varying motif GC and mean exon GC content in Mutant-r2 type *D. melanogaster*. FDR corrected p-values in parenthesis, using a False positive rate of 5% ($\alpha = 0.05$).

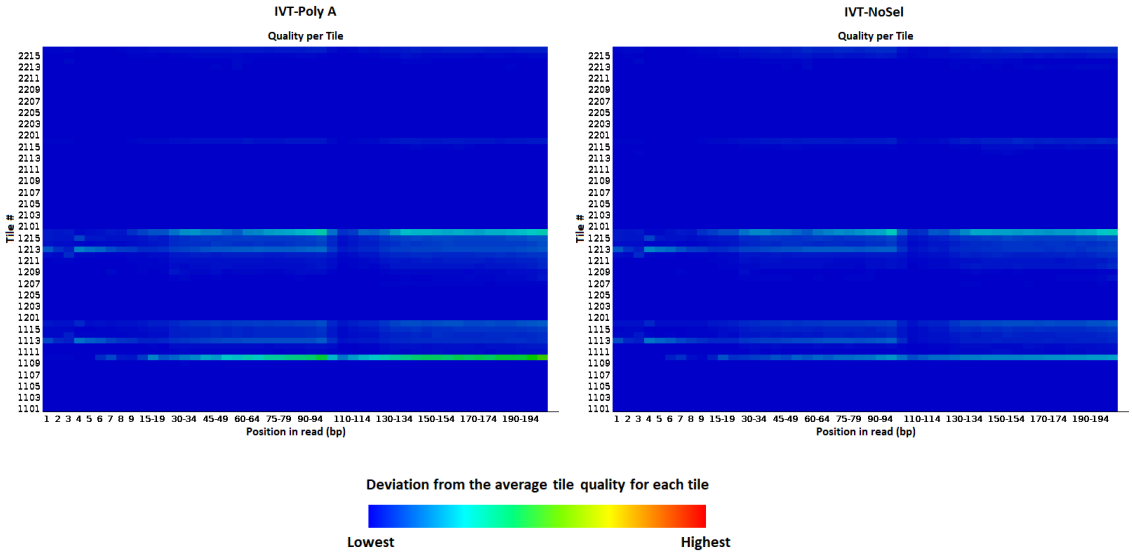


Figure 6.20: Heatmaps of the Sequencing *per-tile* Means, obtained from FastQC analysis for the two RNA-selection *H. sapiens* IVT RNA-Seq samples analysed in this thesis - sequencing tile information was not present in the *IVT-Plasmids* sample. Left) *IVT-PolyA* Right) IVT-NoSel.

6.4.3 FastQC and BamQC - Quality Control analysis of datasets

Quality Control (QC) checks were performed to investigate sources of variation in the data sets we have analysed. Firstly, Qualimap (García-Alcalde et al., 2012) was run on all of the samples and replicates. Secondly, FastQC (Andrews et al., 2010) was run on the source Fasta files. The results are presented below.

A

Wild-type *D. melanogaster*	
Parameter	Result
number of reads	12,960,778
number of mapped reads	12,960,778 (100%)
number of mapped bases	554,135,221 bp
number of sequenced bases	442,559,999 bp
mean mapping quality	135.086
mean coverage data	3.284X
std coverage data	80.2288X
Base composition / GC content	
Parameter	Result
number of A's	114,904,069 bp (25.96%)
number of C's	107,246,603 bp (24.23%)
number of T's	115,649,248 bp (26.13%)
number of G's	104,760,079 bp (23.67%)
number of N's	111,575,222 bp (25.21%)
GC percentage	47.9%

B

Mutant-r2 *D. melanogaster*	
Parameter	Result
number of reads	15,099,081
number of mapped reads	15,099,081 (100%)
number of mapped bases	656,286,624 bp
number of sequenced bases	530,325,061 bp
mean mapping quality	139.9086
mean coverage data	3.8894X
std coverage data	93.0446X
Base composition / GC content	
Parameter	Result
number of A's	136,501,323 bp (25.74%)
number of C's	129,898,946 bp (24.49%)
number of T's	137,081,079 bp (25.85%)
number of G's	126,843,713 bp (23.92%)
number of N's	125,961,563 bp (23.75%)
GC percentage	48.41%

Table 6.15: BamQC analysis summary for (A) wild-type and (B) mutant-r2 Drosophila.

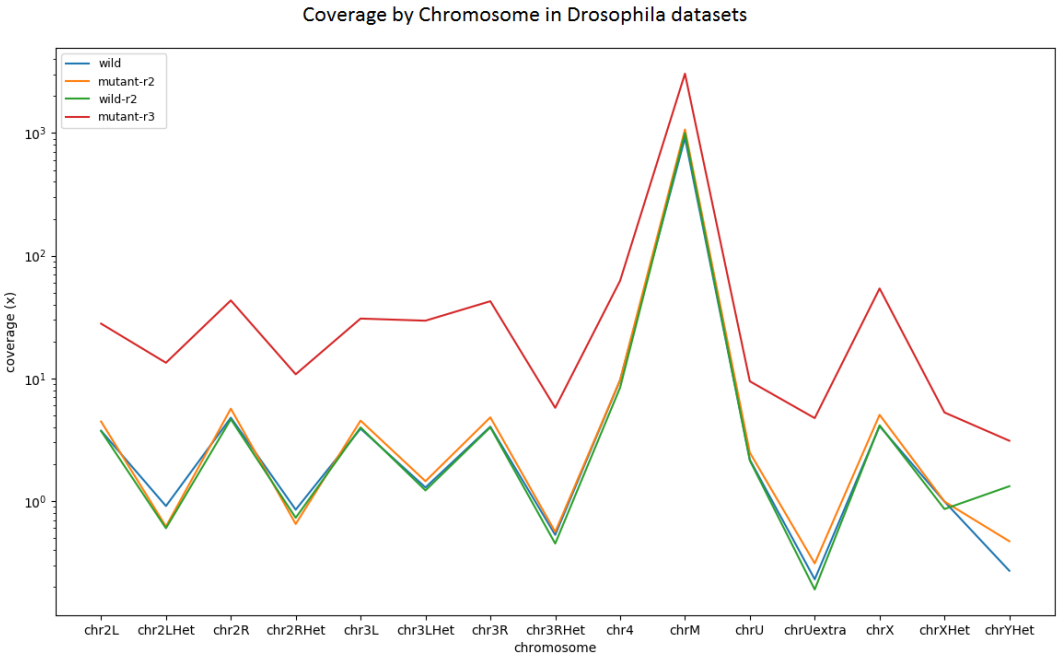


Figure 6.21: Plot of chromosome coverage, obtained from BamQC analysis results for wild-type and mutant-r2 Drosophila.

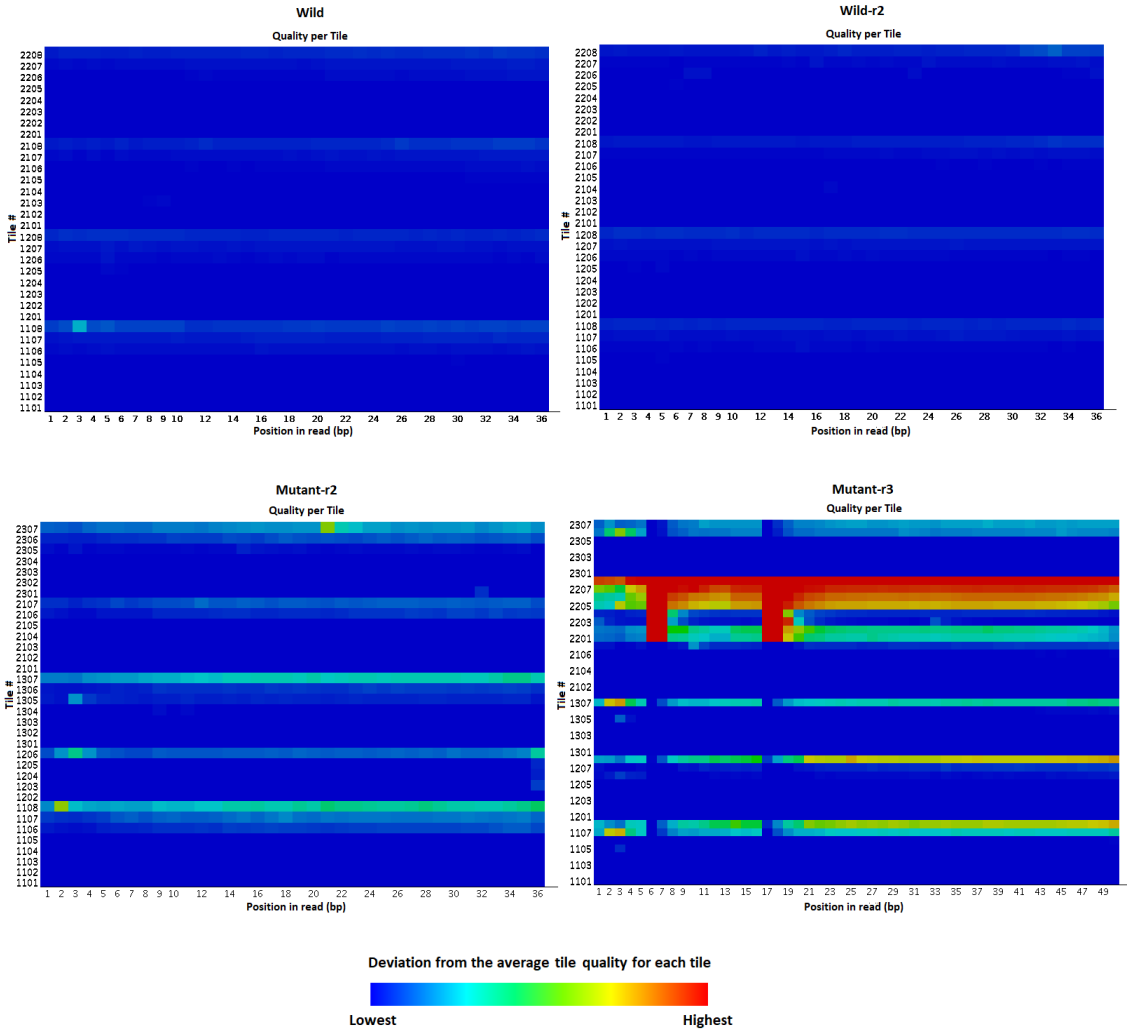


Figure 6.22: Heatmaps of the Sequencing *per-tile* Means, obtained from FastQC analysis for all *D. melanogaster* RNA-Seq samples and their replicates analysed in this thesis Top) Wild-type replicates Bottom) Mutant-type replicates.

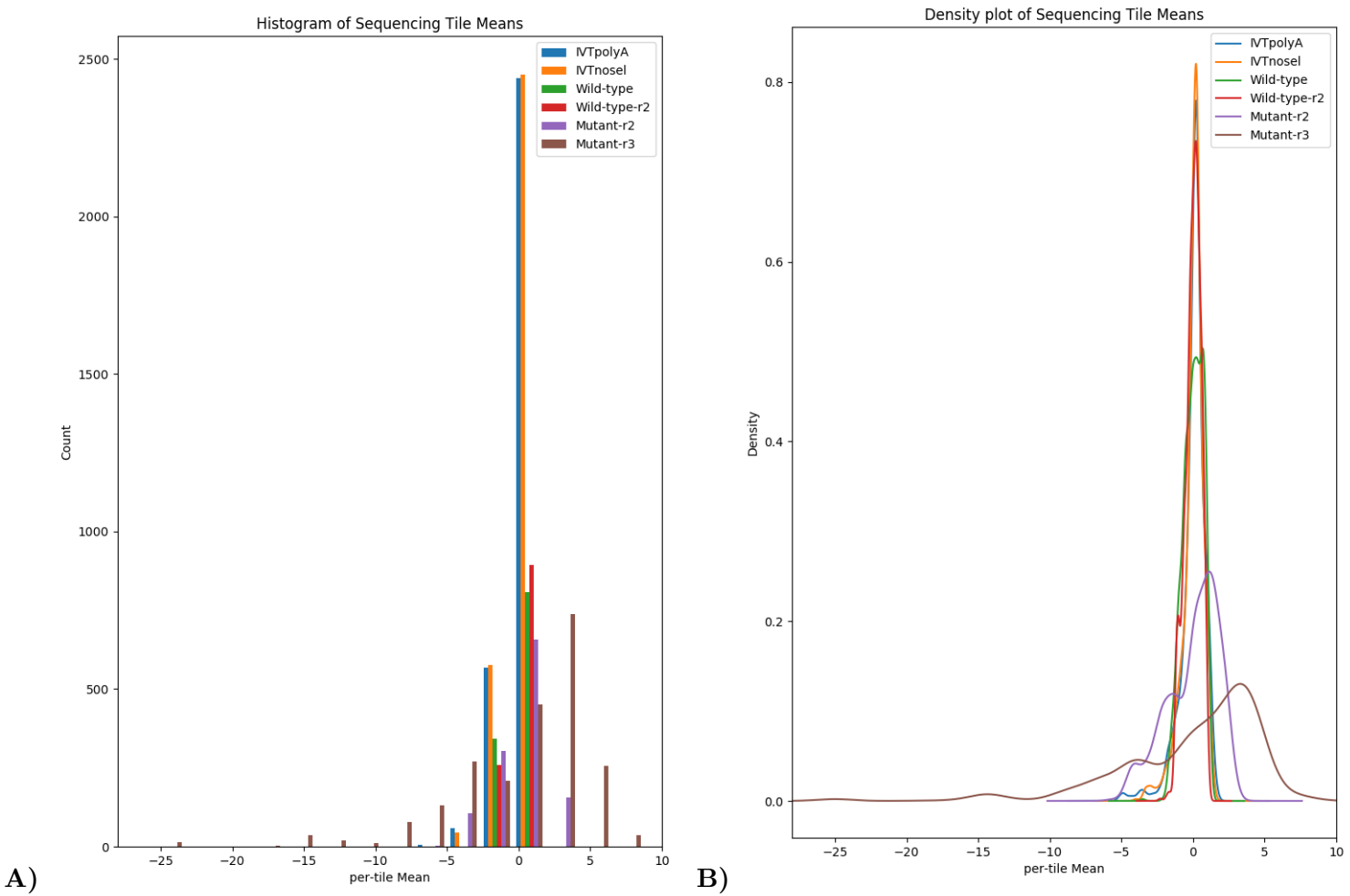


Figure 6.23: Visualisation of the distributions of the Sequencing *per-tile* Means, obtained from FastQC analysis for all RNA-Seq samples analysed in this thesis (species *D. melanogaster* and *H. sapiens*). A) Histogram B) Density plot. The distributions in the IVT datasets, and both wild type replicates, show that less of the tiles in the flowcells of the sequencing apparatus deviate from the *per-tile* Means. This is indicated by the tallest, narrowest peaks, centered around 0, for the IVTPolyA dataset (Blue) followed by IVTnosel (Orange), and then the first wild-type replicate (Red). The second wild-type replicate (Green) was the third tallest peak, and showed reasonably good narrow distribution of the data about the *per-tile* Means, albeit lower than the IVT datasets and first wild type replicate. The two mutant replicates, however, show very wide distributions about the *per-tile* Means in the flowcells, as depicted by the shorter, wider curves (Purple and Brown) respectively. NB: The *IVT-Plasmids* sample did not carry sequencing tile information.

Given there is no biological precedent for the differences observed between the wild and mutant species, and that the low correlations are seen across the mutant replicates, independent of both spacing and GC content, the source of variance is likely to be technical in origin. With this in mind, in order to see if the source of variance between the *D. melanogaster* data sets could be identified by means of Quality Control (QC) checks, we ran Qualimap (García-Alcalde et al., 2012) on all of the *D. melanogaster* samples and replicates, and then ran FastQC (Andrews et al., 2010) on all of the datasets we analysed. In this way, we worked our way backwards through the analysis (i.e. working upstream from our analysis method), starting with the *D. melanogaster* datasets. Firstly, in order to see if there were any alignment or coverage issues in the mutant dataset, we ran Qualimap to analyse the source BAM (Binary Alignment files) that the SAM files were directly derived from. Next, in order to analyse the raw unaligned reads files (Fasta files) for any sequencing issues, we ran FastQC on all of the RNA-Seq replicates and samples, for all datasets that we have analysed in this thesis.

When we compare the BamQC analysis for the wild and mutant types (a summary of the results is given in Table 6.15, there is not much difference between them in terms of the mapping quality, Nucleotide frequencies and GC content. However, there is an increase in variance in the coverage of the mutant-type *D. melanogaster* dataset as indicated by the slightly higher standard deviation (93.045 vs. 80.229). To look into this further, we looked into how coverage across Chromosomes in the two samples was reported in the BamQC report, this is plotted in Figure 6.21. This shows substantial variation in coverage between different chromosomes that reads were mapped to in the organism. However a similar pattern is seen in all samples.

In comparing the FastQC analysis results for the data sets, we found that the *per tile sequencing quality* heatmaps were radically different for the Mutant and wild type data sets (Figure 6.22). These depict deviations from the average tile quality within areas of flowcells on the sequencing apparatus (the explicit values underlying the heatmaps are referred to as the per-tile Means) (Babraham Institute, 2015). In the wild type replicates, these heatmaps were plain and uniform, whereas both of the mutant replicates showed considerable variation from the Means. There were no differences in any of the other FastQC analysis parameters measured between the wild and mutant datasets. In order to investigate this effect further, and to contrast it with the IVT dataset samples, we plotted the distribution of per-tile Means as both a histogram and a density plot (Figure 6.23) for all the samples in all of the datasets we analysed in this thesis. In Figure 6.23, we observe that the distributions in the IVT and wild type datasets show the least deviation from the Mean tile qualities. The mutant type *D. melanogaster* datasets show considerable deviation from the Mean Tile

qualities - this accounts for the low correlations seen across the mutant replicates. By computing *intra-exon* motif-pair correlations we have observed that *sequencing errors occurring in flowcell tiles result in widespread deviations from the uniformity of mapped reads across exons, and are independent of sequence or GC content.*

6.5 Discussion

In this chapter we have extensively detailed our novel *k-mer* based analysis method that allows deep investigation into RNA-Seq read data at the exon level, and allows the quantification of sequence-specific deviations in the uniformity of the distribution of mapped reads to a reference genome. Our approach is based on the assumption that short reads from one region on an exon will be correlated with short reads from another region of the same exon (discussed in section 6.1). This is important work because gene expression studies rely on abundance estimates of RNA transcripts that can be hampered by deviations in the uniformity of read distribution. Another k-mer approach has been followed by Audoux et al. (2017), who have developed DE-kupl, a software tool to analyse k-mer content and detect k-mers with differential abundance directly from the sequencing files, prior to assembly or mapping. Their approach is different in that they aim to capture variation that is not represented in catalogues which comprise of mapped reads. In doing so, they therefore filter out mapped reads and study a much smaller subset of the data – the various filtering steps they apply, for instance, discarded 703.4 M reads and retained 3.6 M reads (from a total of 770 M). Our method, however, focuses on mapped reads that result from the typical read mapping and alignment protocols that are employed in RNA-Seq analysis.

In this thesis, in addition to the IVT dataset produced by Lahens et al. (2014), we have analysed RNA-Seq datasets produced from typical biological specimens using conventional RNA-Seq protocols. These were two small (relative to larger species), but whole transcriptomes - those of the fruit fly species *D. melanogaster* wild and mutant-r2 types, comprising of approximately 12.9 M and 15.0 M reads respectively. Our system can be applied to much larger datasets. The *D. melanogaster* datasets were chosen because of the species and its reference genome are extremely well studied and annotated, and the data has excellent provenance (detailed in section 6.1.3). The analysis of large numbers of reads is made possible because we employ state-of-the-art distributed techniques, specifically MapReduce on Apache Spark, and have optimised our implementation. The job runtime on the cluster (the specifications are listed in paragraph 2 of section 6.2.1) for the each Drosophila dataset was approximately 12 hours when we analysed *4-mers*. Given that GC effects have been identified in runs of four or more `G`s, we chose a value of 4 for k in our analysis - this represents an acceptable trade-off between complexity and job run time (the number of *k-mers* to process is exponential to the *k-mer* length). A computationally expensive step in Phase I of our analysis method is the partitioning of RNA-Seq reads into their corresponding exons in order to count *intra-exon k-mer* abundance, and the optimisation of this step

is discussed in sub-section 6.2.3.

The IVT dataset was produced in a highly controlled manner - we have analysed three *H.sapiens* samples from this, two were subjected to mRNA selection prior to RNA-Seq, and the other was not. These contain fewer reads than the *D. melanogaster*, with the *IVT-Only* sample having 0.406 M reads, the *IVT-PolyA* sample having 0.397 M reads and the *IVT-Plasmids* 0.181 M reads. Although these are comparatively small numbers of reads, the IVT samples have the *intra-exon* bias within them, as indicated by coverage of reads mapping to the source MGC plasmids used, well quantified. From the box and whisker plots of the two IVT samples that underwent mRNA selection RNA-Seq protocols (Ribosomal depletion and Poly-A selection), we observe a dependence of *intra-exon* motif pair correlation on motif GC content, discussed in the next section. This effect is also seen in the Wild-type replicates of the *D. melanogaster* samples, but not in the *IVT-Plasmids* sample which is not subjected to mRNA selection. The mutant D. melanogaster data has very low overall correlations and hence this pattern is not seen here.

6.5.1 Variation in *H. sapiens* IVT samples due to mRNA selection

In section 6.4.1 we have presented the results for the analysis of three RNA-Seq samples from *H. sapiens*. These were produced by selecting human plasmids from the MGC (Mammalian Gene Collection), using in-vitro transcription in *E. coli* (Lahens et al., 2014). We observe that the two IVT samples that were subjected to mRNA selection, *IVT-PolyA* and *IVT-Only*, which were subjected to PolyA selection and Ribosomal depletion respectively, have a similar profile in terms of the high correlations of outliers and intra-exon motif pair correlation as a function of GC content parameters. For both of these samples, a number of *4-mer* motif-pairs that have very high correlations (Pearson correlations very close to +1), and these high correlations are observed across all spacings (A and B of Table 6.8). The same table also shows some extremely low correlations due to a lack of 4-mer data (as indicated by the sample sizes in parenthesis). This is likely due to the fact that the IVT sample datasets were created in a highly controlled way, from a pool of only 1,062 human RNA transcripts and therefore have fewer mappable reads the the Drosophila datasets. The box and whisker plots of correlation as a function of motif GC content and mean exon GC content for these two samples are almost identical (Figure 6.13) - interestingly this shows the same dependence of intra-exon motif correlation on motif GC content as is observed in the wild-type replicates of the *D. melanogaster* datasets.

When we compare these two samples that underwent mRNA selection to that of the control sample (*IVT-Plasmids*), which did not, we see differences. In particular, the *IVT-Plasmids* control sample had no dependence of intra-exon motif pair correlation on GC content - neither motif GC nor exon GC. Although Figure 6.12 shows a decrease in correlation for motif GC content of 100%. In particular, as discussed in section 6.5.2, we have demonstrated that the mutant data has a noticeable variation across flow cells and this is a hypothesis for why the correlations are so low, introducing much more noise into the analysis.

6.5.2 Variation in *D. melanogaster* transcriptomics samples

In section 6.4.2 we have presented the results of our analysis of two transcriptomes drawn from *D. melanogaster* (Naval-Sánchez et al., 2013), which differ only by mutation *gl[60j]* in the eye-antennal disc. We have shown evidence that the wild-type data set exhibits bias that is specific to the sequence of the motif rather than the overall GC content of the exon. The effect appears not to be specific to particular sequences - for example the motifs GGGG and CCCC (200 bp spacing) have the lowest correlations (Spearman's) in the wild-type data (A of Table 6.11) but do not appear in the ten lowest correlations for the mutant data (B of Table 6.11). Conversely, there is no noticeable effect observed in the mutant data set and more specifically, the correlations are significantly smaller for the mutant data set. These effects are largely independent of changing the spacing between occurrences of the *4-mer* motifs.

It is important to note the consequences of the results we have obtained by applying our analysis method. These are two RNA-Seq data sets that have been generated in the same lab on the same species. As discussed in section 6.1.3, the protocols applied in preparing the sample and performing the sequencing are the same. Furthermore the data sets are gathered from the same type of tissue. There will be differences in the transcriptome because of the genetic perturbation; however we expect only a fraction of changes in expression and splicing. In the same respect, multi-mapped reads, as outlined in Ji et al. (2011); Feng et al. (2015a), may represent a source of bias if reads map to many locations but are only recorded in the above count data in one location. However, as we expect only a fraction of the transcriptome to be perturbed the difference between them, the changes in correlations should remain overall relatively small. What is observed are changes in correlations that represent a much more significant change in the distribution of the short reads between these two data sets.

6.6 Dependence of *intra-exon* correlation on GC content appears to be due to mRNA selection

The *intra-exon* motif correlations as a function of both GC content parameters are much higher in the *IVT-Plasmids* mRNA *selection free* RNA-Seq sample than in the other RNA-Seq samples that we analysed that did undergo mRNA selection: both ribosomal depletion (*IVT-Only*) and PolyA selection (*IVT-polyA* and Wild-type). Furthermore, both of the *H. sapiens* and wild-type *D. melanogaster* samples that underwent mRNA selection in the RNA-Seq process had slightly lower correlations than the H. sapiens *IVT-Plasmids* sample which did not, suggesting this is likely of technical origin. Im-

portantly, all the samples from all of the datasets we analysed were sequenced on the same platform - Illumina HiSeq 2000 (as detailed in Table 6.2).

As the dependence on overall GC concentration is not observed in the *IVT-Plasmids* control sample, but is observed in RNA-Seq samples that underwent mRNA selection, we can exclude platform specific sequencing bias as a source of this effect. This suggests that not only do mRNA selection protocols result in bias in the distribution of mapped RNA-Seq reads as Lahens et al. (2014) demonstrated, but that *mRNA selection is also responsible for the dependence of correlation on GC content* - we have observed this in all of the RNA-Seq samples that underwent mRNA selection across both *H. sapiens* and *D. melanogaster* species. Risso et al. (2011) also noted motif-specific GC effects, manifesting as deviations from uniform read distribution, which they attribute these problematic motifs being underrepresented. However, the GC effect we observe occurs in both the wild type replicates, which have large sample sizes (A of Table 6.9), as well as in the IVTpolyA and IVTnosel mRNA selection datasets, which have low sample sizes (due to the controlled way in which the samples were prepared)(Table 6.8). Furthermore, the numbers of counts for high GC content motifs, as indicated by the highest outliers (Table 6.9), is of the same order as other motifs. The dependence of *intra-exon* correlations on the GC content of the motifs appears to be due to mRNA selection methods, which are routinely employed in RNA-Seq experiments, and are known to introduce bias (Nam et al., 2002; Zhang et al., 2012; Lahens et al., 2014) (also described earlier in Chapter 3, Table 3.2).

.

Chapter 7

Conclusion

The recent increase in the generation of biological data due to the use of high-throughput technologies, decreased cost of sequencing, and large-scale multi-national collaborative projects have together resulted in large, complex datasets. These datasets are routinely deposited in public archives that now store data at an unprecedented scale together with their experimental metadata. The aforementioned developments necessitate specialist methods for processing such datasets, most often through the use of distributed computing. Furthermore, integrative approaches that combine experimental data, and biomedical research applications aimed at extracting information from the data for hypothesis testing, require techniques that are not only capable of handling high-throughput data, but that can also be integrated into pipelines. This approach ideally avoids the need to download large datasets by bringing computation to the data, hence research projects that utilise cloud services are increasing, and such methods can also be used to augment in-house computation.

In chapter 2 of this thesis we have therefore examined distributed technologies of relevance to high-throughput Biological datasets in considerable detail. The focus of the research has been on the MapReduce programming paradigm which is now an emerging method in distributed computing cluster systems for processing high-throughput datasets. We have contrasted MapReduce with batch-scheduled cluster computing, which has long been a traditional method in computational biology - both methods are deployable to the platforms that cloud services providers host. In comparing the two approaches and their technologies, we have covered their architectures in considerable technical detail, and have described how MapReduce inherently provides scalability and fault-tolerance. We have demonstrated MapReduce's utility for processing high-throughput sequencing data (chapter 6) and, as a test case, have shown it can also be applied to conventional, semi-structured data sets, such as the molecular data con-

tained in the Protein Data Bank, on which we performed molecular docking (chapter 5) (Alnasir and Shanahan, 2015b). Following on from this experimental research, we have also devised a novel method for the quantification of the deviation in read distribution caused by bias in RNA-Seq data (chapter 6) which we will discuss in more detail later in this section. This method was applied to the analysis of two whole transcriptomes of *D. melanogaster*, as well as three samples from *H. Sapiens*, prepared in a controlled way, using *in-vitro* transcription (IVT) with different RNA-Seq preparatory protocols applied. We have, therefore, demonstrated that MapReduce can also be leveraged for the analysis of short read transcriptomics sequencing data (Alnasir and Shanahan, 2017, 2016b,a).

In order to facilitate reproducibility of experiments and to understand the types of bias that may be introduced, a necessary accompaniment to high-throughput biological datasets are associated metadata that describe the experimental parameters used, and should contain information on the protocol steps and techniques used to produce the data. We will discuss research in this thesis (chapter 4) that contributes to this area, specifically with respect to the annotation of experimental metadata in the SRA, shortly after first discussing research we have conducted into bias introduced the preparatory steps of high-throughput sequencing projects.

As a significant part of this thesis concerns sequencing data produced by Next-generation high-throughput technologies, with this in mind, in chapter 3 we have discussed and elucidated in detail, at the molecular level, the *wet-lab* techniques that are employed in the preparatory steps applied to nucleic acid samples prior to sequencing. Moreover, we have conducted extensive research into the molecular mechanisms documented in the literature that are known to introduce bias into the sequencing process (Alnasir and Shanahan, 2015a, 2014). We have also discussed the evolution of different sequencing techniques and the way in which they work as well as the types of bias that they are prone to, though our focus is on RNA-Seq. To this end we have investigated and reviewed the types of bias that is present in RNA-Seq data, for instance sequence specific and positional bias as well as those caused by extremes of GC-content. In particular, we have concentrated on those bias that affect the measurement of RNA-expression by contributing to the non-uniformity of the distribution of mapped reads to a reference genome, an effect which is partly due to the short read length in RNA-Seq but also results from the *wet-lab* chemical preparatory steps applied to nucleic acid samples prior to sequencing. Measurement of RNA transcript expression is key to a number of important scientific and biomedical applications, and errors in expression estimates can affect downstream analysis reliant on the measurements and jeopardise scientific conclusions derived from RNA-Seq data. We therefore concluded chapter 3

by introducing and reviewing some of the prominent statistical models to characterise and mitigate bias, that manifests as deviations in the distribution of mapped transcriptomics reads to a reference genome in RNA-Seq data that are applied in *dry-lab*.

Considering the complexity of the *wet-lab* steps typically involved in preparing a nucleic acid sample for sequencing on a Next-generation sequencing platform, we conducted a detailed investigation into the annotation of sequencing data deposited in the SRA, one of the main repositories of raw NGS data. This was accomplished using data mining techniques that employed SQL (Structured Query Language) to query the Bioconductor SRA meta DB, a proxy for the SRA metadata which provides the SRA metadata as an SQLite database that is routinely synchronised with the SRA. Our method involved constructing a list of relevant keywords for each of the preparatory steps in a sequencing workflow, described in chapter 3 (section 3.4 onwards). This was used to quantify their abundance in the SRA metadata which we presented in chapter 4, and has been published together with the research work on bias in the GigaScience journal paper (Alnasir and Shanahan, 2015a). We have observed and demonstrated that the annotation of the SRA is very sparse - less than 6% (5.58%) of top-level studies possess keywords corresponding to all the steps that are relevant to the protocols. Furthermore, approximately half of the experiment entries have a null (empty) entry in the fields where this data should be recorded. We also highlighted the non-standard methods used in describing the annotation of metadata for deposited raw experimental data in the SRA, for instance in web URLs in place of textual description, and the observed variation in how many of the experiment records for a given study are annotated. The SRA is an extremely large repository for genomic and transcriptomic data (over 9 peta bases as of January 2017), it should, therefore, be invaluable for comparative genomics and meta-analyses. However, as this thesis has demonstrated in chapter 4, the poor level of annotation of protocol steps used to prepare sequencing data means that potential biases may exist in these datasets that will be unquantified.

In chapter 6 we have applied our transcriptomics analysis method, implemented in MapReduce, to three RNA-Seq samples from *H. sapiens* which were produced in a highly controlled manner, by selecting human plasmids from the MGC (Mammalian Gene Collection), using in-vitro transcription in *E. coli* (Lahens et al., 2014), and two transcriptomes drawn from *D. melanogaster* (Naval-Sánchez et al., 2013), which differ only by mutation *gl[60j]* in the eye-antennal disc. Our method works at a deep exon level to quantify deviations in the uniformity of distribution of mapped reads across exons.

Of the three IVT-Seq samples from we analysed, two were subjected to mRNA selection prior to RNA-Seq, and the other was not. We have shown that the two IVT

samples that were subjected to mRNA selection, *IVT-PolyA* and *IVT-Only* - subjected to PolyA selection and Ribosomal depletion respectively - have a similar profile in terms of the high correlations of outliers and intra-exon motif pair correlation as a function of GC content parameters. The *intra-exon* motif correlations as a function of both GC content parameters are much higher in the *IVT-Plasmids* mRNA *selection free* RNA-Seq sample than in the other RNA-Seq samples which we analysed that did undergo mRNA selection: both ribosomal depletion (*IVT-Only*) and PolyA selection (*IVT-polyA* and Wild-type). We have therefore demonstrated that the dependence of *intra-exon* correlations on the GC content appears to be due to mRNA selection methods. These techniques are routinely employed in RNA-Seq experiments, and have been implicated in introducing bias (Nam et al., 2002; Zhang et al., 2012; Lahens et al., 2014). This effect has been observed in all of the RNA-Seq samples that underwent mRNA selection across both *H. sapiens* and *D. melanogaster* species.

We have shown evidence that the wild-type data set exhibits bias that is specific to the sequence of the motif rather than the overall GC content of the exon. These effects are largely independent of changing the spacing between occurrences of the *4-mer* motifs. Conversely, there is no noticeable effect observed in the mutant data set and more specifically, the correlations are significantly smaller for the mutant data set.

The results we have obtained by applying our analysis method are consequential. The two *D. melanogaster* RNA-Seq data sets have been generated in the same lab on the same species. As discussed in section 6.1.3, the protocols applied in preparing the sample and performing the sequencing are the same - the datasets have good provenance. Furthermore the data sets are gathered from the same type of tissue. There will be differences in the transcriptome because of the genetic perturbation; however, we expect only a fraction of changes in expression and splicing (multi-mapped reads are discussed in 6.5.2). As we expect only a fraction of the transcriptome to be perturbed the difference between them, the changes in correlations should remain overall relatively small. What is observed are changes in correlations that represent a much more significant change in the distribution of the short reads between these two data sets. Running quality analysis software (FastQC) on the datasets reveals considerable deviation from the Mean Tile qualities in the mutant data set, which result from sequencing errors occurring in flowcell tiles. Using our method to compute the *intra-exon* motif-pair correlations, we have observed that sequencing errors result in widespread deviations from the uniformity of mapped reads across exons and accounts for the low correlations seen in the mutant replicates, and that these effects are independent of sequence or GC content.

In this thesis we have developed a platform to detect bias in NGS data that uses

industry standard data formats. In utilising MapReduce, our platform is highly scalable to allow for the processing of large high-throughput datasets, and is also deployable to cloud service infrastructures. The results we have obtained indicate that there is much to explore utilising these methods.

Bibliography

1000 Genomes Project Consortium, T. . G. P., Abecasis, G. R., Auton, A., Brooks, L. D., DePristo, M. A., Durbin, R. M., Handsaker, R. E., Kang, H. M., Marth, G. T., and McVean, G. A. (2012). An integrated map of genetic variation from 1,092 human genomes. *Nature*, 491(7422):56–65. 18, 19

1000 Genomes Project Consortium and others (2015). A global reference for human genetic variation. *Nature*, 526(7571):68–74. 18, 21

Abiteboul, S. (1997). Querying semi-structured data. *Database Theory ICDT'97*, pages 1–18. 110

Abola, E. E., Sussman, J. L., Prilusky, J., and Manning, N. O. (1997). Protein Data Bank archives of three-dimensional macromolecular structures. *Methods in enzymology*, 277:556–71. 20, 110, 111, 123

Abouelhoda, M., Issa, S. A., and Ghanem, M. (2012). Tavaxy: Integrating taverna and galaxy workflows with cloud computing support. *BMC bioinformatics*, 13(1):77. 53

Acinas, S. G., Sarma-Rupavtarm, R., Klepac-Ceraj, V., and Polz, M. F. (2005). PCR-induced sequence artifacts and bias: insights from comparison of two 16S rRNA clone libraries constructed from the same sample. *Applied and environmental microbiology*, 71(12):8966–9. 81, 86, 92, 95

Adams, M. D., Kelley, J. M., Gocayne, J. D., Dubnick, M., Polymeropoulos, M. H., Xiao, H., Merril, C. R., Wu, A., Olde, B., Moreno, R. F., et al. (1991). Complementary dna sequencing: expressed sequence tags and human genome project. *Science*, 252(5013):1651–1656. 88

Aerts, S. (2012). GEO-GSE39781: RNA-seq in wild-type and glass mutant eye-antennal discs in Drosophila melanogaster. https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE39781. [Online; accessed 27-March-2017]. 164

Aird, D., Ross, M. G., Chen, W.-S., Danielsson, M., Fennell, T., Russ, C., Jaffe, D. B., Nusbaum, C., and Gnirke, A. (2011). Analyzing and minimizing PCR amplification bias in Illumina sequencing libraries. *Genome biology*, 12(2):R18. 86

Alazami, A. M., Patel, N., Shamseldin, H. E., Anazi, S., Al-Dosari, M. S., Alzahrani, F., Hijazi, H., Alshammari, M., Aldahmesh, M. A., Salih, M. A., et al. (2015). Accelerating novel candidate gene discovery in neurogenetic disorders via whole-exome sequencing of prescreened multiplex consanguineous families. *Cell reports*, 10(2):148–161. 82

Albert, T. J., Molla, M. N., Muzny, D. M., Nazareth, L., Wheeler, D., Song, X., Richmond, T. A., Middle, C. M., Rodesch, M. J., Packard, C. J., et al. (2007). Direct selection of human genomic loci by microarray hybridization. *Nature methods*, 4(11):903–906. 82

Allen, F. H. (2002). The cambridge structural database: a quarter of a million crystal structures and rising. *Acta Crystallographica Section B: Structural Science*, 58(3):380–388. 123

Allhoff, M., Schönhuth, A., Martin, M., Costa, I. G., Rahmann, S., and Marschall, T. (2013). Discovering motifs that induce sequencing errors. *BMC bioinformatics*, 14 Suppl 5(Suppl 5):S1. 77

Alnasir, J; Shanahan, H. (2015). Supporting material for "Investigation into the annotation of protocol sequencing steps in the Sequence Read Archive". http://doi.org/10.5524/100142. [Online; accessed 15-May-2015]. 96, 99

Alnasir, J. (2010). Zeus desktop molecular visualisation software: Molecular viewer for windows. 133

Alnasir, J. (2017a). Artemis: Python library for reading/writing PDB files and computing dihedral angles. https://pure.royalholloway.ac.uk/portal/en/publications/artemis--python-library-for-readingwriting-pdb-files-and-computing-dihedral-angles(a.html. [Online; accessed 01-January-2017]. 131

Alnasir, J. (2017b). Dihedrals-64: PDB-Cli - Protein Data Bank Command Line Tools for Windows and Linux. http://personal.rhul.ac.uk/mxba/001/PDB/. [Online; accessed 01-January-2017]. 131

Alnasir, J. (2017c). Source code and results data for Transcriptomics Analysis System (Hercules). https://doi.org/10.5281/zenodo.801378. [Online; accessed 23-June-2017]. 156

Alnasir, J. (2018). Appendix for "The Analysis of High-Throughput Biological Datasets Utilising Distributed Computing". https://doi.org/10.5281/zenodo.1213356. [Appendix to this thesis; Online; accessed 05-April-2018]. 131, 135, 195, 206

Alnasir, J. and Shanahan, H. (2016a). Transcriptomics on spark workshop introducing hercules an apache spark mapreduce algorithm for quantifying non-uniform gene expression. 220

Alnasir, J. and Shanahan, H. P. (2015a). Investigation into the annotation of protocol sequencing steps in the sequence read archive. *GigaScience*, 4:23. 92, 95, 220, 221

Alnasir, J. and Shanahan, H. P. (2017). A novel method to detect bias in short read ngs data. *Journal of integrative bioinformatics*, 14(3). 220

Alnasir, J. J. and Shanahan, H. P. (2014). Annotation of next-generation sequencing protocol steps in the sra (sequence read archive) and big data for science. page 27. 220

Alnasir, J. J. and Shanahan, H. P. (2015b). Applying apache hadoop, hive and map reduce to legacy systems and applications. 220

Alnasir, J. J. and Shanahan, H. P. (2016b). Transcriptomics: Leveraging a mapreduce algorithm and python for gene-expression analysis on apache spark. 220

Altschul, S. F., Gish, W., Miller, W., Myers, E. W., and Lipman, D. J. (1990). Basic local alignment search tool. *Journal of molecular biology*, 215(3):403–410. 50

Amazon (2016). Amazon EMR (Elastic MapReduce). https://aws.amazon.com/emr/. [Online; accessed 14-April-2017]. 53, 125

Anders, S. and Huber, W. (2010). Differential expression analysis for sequence count data. *Genome biology*, 11(10):R106. 177

Anderson, S. (1981). Shotgun dna sequencing using cloned dnase i-generated fragments. *Nucleic Acids Research*, 9(13):3015–3027. 62

Andrews, S. et al. (2010). Fastqc: a quality control tool for high throughput sequence data. 165, 208, 212

Apache Software Foundation (2014). Spark 2.6 documentation. http://spark.apache.org/docs/latest/programming-guide.html. [Online; accessed 26-Jan-2017]. 49

Apache Software Foundation (2015). SparkR documentation. https://spark.apache.org/docs/latest/sparkr.html. [Online; accessed 21-October-2017]. 49

Apache Software Foundation (2016a). Hadoop 2.7 documentation. http://hadoop.apache.org/docs/r2.7.2/. [Online; accessed 06-Jan-2017]. 38

Apache Software Foundation (2016b). Hadoop streaming documentation. http://hadoop.apache.org/docs/current/hadoop-streaming/HadoopStreaming.html. [Online; accessed 14-April-2017]. 47

Apache Software Foundation (2016c). HDFS architecture documentation. http://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-hdfs/HdfsDesign.html. [Online; accessed 10-Jan-2017]. 45

Armstrong, R. A. (2014). When to use the bonferroni correction. *Ophthalmic and Physiological Optics*, 34(5):502–508. 182

Arpaci, R. H., Dusseau, A. C., Vahdat, A. M., Liu, L. T., Anderson, T. E., and Patterson, D. A. (1995). The interaction of parallel and sequential workloads on a network of workstations. In *ACM SIGMETRICS Performance Evaluation Review*, volume 23, pages 267–278. ACM. 30

ArrayExpress, EMBL-EBI (2017). ArrayExpress functional genomics data. https://www.ebi.ac.uk/arrayexpress/. [Online; accessed 3-January-2017]. 17, 71

Audoux, J., Philippe, N., Chikhi, R., Salson, M., Gabriel, M., Commes, T., and Gautheret, D. (2017). Exhaustive capture of biological variation in rna-seq data through k-mer decomposition. *bioRxiv*, page 122937. 214

Babraham Institute (2015). FastQC documentation - per-tile sequencing quality. https://www.bioinformatics.babraham.ac.uk/projects/fastqc/Help/3AnalysisModules/12PerTileSequenceQuality.html. [Online; accessed 03-Dec-2017]. 212

Balzer, S., Malde, K., and Jonassen, I. (2011). Systematic exploration of error sources in pyrosequencing flowgram data. *Bioinformatics*, 27(13):i304–i309. 75

Bamshad, M. J., Ng, S. B., Bigham, A. W., Tabor, H. K., Emond, M. J., Nickerson, D. A., and Shendure, J. (2011). Exome sequencing as a tool for mendelian disease gene discovery. *Nature Reviews Genetics*, 12(11):745–755. 82

Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Neugebauer, R., Pratt, I., and Warfield, A. (2003). Xen and the art of virtualization. In *ACM SIGOPS operating systems review*, volume 37, pages 164–177. ACM. 31

Bashiardes, S., Veile, R., Helms, C., Mardis, E. R., Bowcock, A. M., and Lovett, M. (2005). Direct genomic selection. *Nature methods*, 2(1):63–69. 82

Beberg, A. L., Ensign, D. L., Jayachandran, G., Khaliq, S., and Pande, V. S. (2009). Folding@ home: Lessons from eight years of volunteer distributed computing. In *Parallel & Distributed Processing, 2009. IPDPS 2009. IEEE International Symposium on*, pages 1–8. IEEE. 31

Benjamini, Y. and Hochberg, Y. (1995). Controlling the false discovery rate: a practical and powerful approach to multiple testing. *Journal of the royal statistical society. Series B (Methodological)*, pages 289–300. 182

Benjamini, Y. and Speed, T. P. (2012). Summarizing and correcting the gc content bias in high-throughput sequencing. *Nucleic acids research*, page 001. 90, 91, 155

Bennett, S. (2004). Solexa ltd. *Pharmacogenomics*, 5(4):433–438. 19

Bentley, D. R., Balasubramanian, S., Swerdlow, H. P., Smith, G. P., Milton, J., Brown, C. G., Hall, K. P., Evers, D. J., Barnes, C. L., Bignell, H. R., et al. (2008). Accurate whole human genome sequencing using reversible terminator chemistry. *nature*, 456(7218):53–59. 19

Berman, H. M., Westbrook, J., Feng, Z., Gilliland, G., Bhat, T. N., Weissig, H., Shindyalov, I. N., and Bourne, P. E. (2000). The Protein Data Bank. *Nucleic acids research*, 28(1):235–42. 20, 110, 111, 123

Bernal, A., Ear, U., and Kyrpides, N. (2001). Genomes online database (gold): a monitor of genome projects world-wide. *Nucleic Acids Research*, 29(1):126–127. 16

Big Data Technology blog (2015). Hadoop streaming. `http://www.big4future.com/2015/08/how-to-run-php-map-reduce-job-using.html`. [Online; accessed 15-April-2017]. 48

Bioconductor project v2.14 (2013). Bioconductor: A compilation of metadata from NCBI SRA and tools. http://www.bioconductor.org/packages/2.14/bioc/html/SRAdb.html. [Online; accessed 2-December-2013]. 96, 99

Bokulich, N. A., Subramanian, S., Faith, J. J., Gevers, D., Gordon, J. I., Knight, R., Mills, D. A., and Caporaso, J. G. (2013). Quality-filtering vastly improves diversity estimates from illumina amplicon sequencing. *Nature methods*, 10(1):57–59. 76, 83, 87

Borgman, C. L. (2015). *Big Data, little data, no data: Scholarship in the networked world*. Mit Press. 17, 43, 54, 94

Branton, D., Deamer, D. W., Marziali, A., Bayley, H., Benner, S. A., Butler, T., Di Ventra, M., Garaj, S., Hibbs, A., Huang, X., et al. (2008). The potential and challenges of nanopore sequencing. *Nature biotechnology*, 26(10):1146–1153. 67

Bray, N. L., Pimentel, H., Melsted, P., and Pachter, L. (2016). Near-optimal probabilistic rna-seq quantification. *Nature biotechnology*, 34(5):525–527. 91

Brazma, A. (2003). ArrayExpress–a public repository for microarray gene expression data at the EBI. *Nucleic Acids Research*, 31(1):68–71. 17, 96, 106

Brazma, A. (2009). Minimum information about a microarray experiment (miame)–successes, failures, challenges. *The Scientific World Journal*, 9:420–423. 108

Brazma, A., Hingamp, P., Quackenbush, J., Sherlock, G., Spellman, P., Stoeckert, C., Aach, J., Ansorge, W., Ball, C. A., Causton, H. C., et al. (2001). Minimum information about a microarray experiment (miame)toward standards for microarray data. *Nature genetics*, 29(4):365–371. 96

Broad Institute (2016a). Cromwell, execution engine for WDL - Documentation via Forum. https://gatkforums.broadinstitute.org/gatk/discussion/7349/the-art-of-the-pipeline-introducing-cromwell-wdl. [Online; accessed 21-Nov-2017]. 51

Broad Institute (2016b). WDL (Workflow Definition Language) specification and documentation. https://software.broadinstitute.org/wdl/documentation/spec. [Online; accessed 21-Nov-2017]. 51

Brochard, L. (2006). High performance computing technology, applications and business. *Informatik-Spektrum*, 29(3):191–200. 32

Brooks, B. R., Bruccoleri, R. E., Olafson, B. D., States, D. J., Swaminathan, S. a., and Karplus, M. (1983). Charmm: a program for macromolecular energy, minimization, and dynamics calculations. *Journal of computational chemistry*, 4(2):187–217. 114

Bryson, K., Cozzetto, D., and Jones, D. T. (2007). Computer-assisted protein domain boundary prediction using the dom-pred server. *Current Protein and Peptide Science*, 8(2):181–188. 123

Buchan, D. W., Minneci, F., Nugent, T. C., Bryson, K., and Jones, D. T. (2013). Scalable web services for the psipred protein analysis workbench. *Nucleic acids research*, 41(W1):W349–W357. 123, 153

Buneman, P. (1997). Semistructured data. In *Proceedings of the sixteenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*, pages 117–121. ACM. 110

Bybee, S. M., Bracken-Grissom, H., Haynes, B. D., Hermansen, R. A., Byers, R. L., Clement, M. J., Udall, J. A., Wilcox, E. R., and Crandall, K. A. (2011). Targeted amplicon sequencing (tas): a scalable next-gen approach to multilocus, multitaxa phylogenetics. *Genome biology and evolution*, 3:1312–1323. 83

Bywaters, K., McKay, C., Davila, A., and Quinn, R. (2016). In situ life and biosignature detection at mars analog sites using the oxford nanopore minion sequencer. *LPI Contributions*, 1912. 20

Canard, B. and Sarfati, R. S. (1994). Dna polymerase fluorescent substrates with reversible 3-tags. *Gene*, 148(1):1–6. 19

Center for Public Health Genomics, University of Virginia, (2017). SourceForge: TavernaPBS plug-in for PBS cluster operation. https://sourceforge.net/projects/tavernapbs/. [Online; accessed 26-June-2017]. 54

Check, H. E. et al. (2015). Pint-sized dna sequencer impresses first users (vol 521, pg 15, 2015). *Nature*, 521(7551):139–139. 67

Chen, Y.-C., Liu, T., Yu, C.-H., Chiang, T.-Y., and Hwang, C.-C. (2013a). Effects of GC bias in next-generation-sequencing data on de novo genome assembly. *PloS one*, 8(4):e62856. 81, 86, 92, 95

Chen, Y.-C., Liu, T., Yu, C.-H., Chiang, T.-Y., and Hwang, C.-C. (2013b). Effects of gc bias in next-generation-sequencing data on de novo genome assembly. *PloS one*, 8(4):e62856. 176, 181

Cheung, M.-S., Down, T. A., Latorre, I., and Ahringer, J. (2011). Systematic bias in high-throughput sequencing data and its correction by BEADS. *Nucleic acids research*, 39(15):e103. 77

Chial, H. (2008). Dna sequencing technologies key to the human genome project. *Nature Education*, 1(1):219. 63

Cho, J.-H., Jung, D.-K., Lee, K., and Rhee, S. (2009). Crystal structure and functional analysis of the extradiol dioxygenase lapb from a long-chain alkylphenol degradation pathway in pseudomonas. *The Journal of Biological Chemistry*, 284(49):34321. 148

Christelle, R. and Watson, M. (2015). Errors in rna-seq quantification affect genes of relevance to human disease. *Genome Biology*, 16. 92

Chun, B. N. and Culler, D. E. (2002). User-centric performance analysis of market-based cluster batch schedulers. In *Cluster Computing and the Grid, 2002. 2nd IEEE/ACM International Symposium on*, pages 30–30. IEEE. 34

Clark, T. A., Sugnet, C. W., and Ares, M. (2002). Genomewide analysis of mrna processing in yeast using splicing-specific microarrays. *Science*, 296(5569):907–910. 71

Clarke, T. and Begley S., Reuters (2015a). 1000 Genomes Project Releases Data from Pilot Projects on Path to Providing Database for 2,500 Human Genomes - Freely available data supporting next generation of human genetic research. http://www.reuters.com/article/us-usa-obama-precisionmedicine-idUSKBN0L313R20150130. [Online; accessed 02-February-2017]. 18

Clarke, T. and Begley S., Reuters (2015b). U.S. to Develop DNA Study of One Million People. https://www.technologyreview.com/s/534591/us-to-develop-dna-study-of-one-million-people/. [Online; accessed 02-February-2017]. 18

Cloudera (2016). About Cloudera. https://www.cloudera.com/more/about.html. [Online; accessed 01-February-2018]. 125

Cochrane, G., Karsch-Mizrachi, I., and Nakamura, Y. (2010). The international nucleotide sequence database collaboration. *Nucleic acids research*, page gkq1150. 97

Coulouris, G. F., Dollimore, J., and Kindberg, T. (2005). *Distributed systems: concepts and design*. pearson education. 30

Cyranoski, D. (2016). China's bid to be a dna superpower. *Nature*, 534(7608):462–463. 18

Dai, L., Gao, X., Guo, Y., Xiao, J., and Zhang, Z. (2012). Bioinformatics clouds for big data manipulation. *Biology direct*, 7(1):43. 52

Day, D. J., Speiser, P. W., Schulze, E., Bettendorf, M., Fitness, J., Barany, F., and White, P. C. (1996). Identification of non-amplifying cyp21 genes when using pcr-based diagnosis of 21-hydroxylase deficiency in congenital adrenal hyperplasia (cah) affected pedigrees. *Human Molecular Genetics*, 5(12):2039–2048. 86

Deamer, D. W. and Akeson, M. (2000). Nanopores and nucleic acids: prospects for ultrarapid sequencing. *Trends in biotechnology*, 18(4):147–151. 67

Dean, J. and Ghemawat, S. (2008). Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113. 40

Deelman, E., Vahi, K., Juve, G., Rynge, M., Callaghan, S., Maechling, P. J., Mayani, R., Chen, W., da Silva, R. F., Livny, M., et al. (2015). Pegasus, a workflow management system for science automation. *Future Generation Computer Systems*, 46:17–35. 53

Degtyarenko, K., De Matos, P., Ennis, M., Hastings, J., Zbinden, M., McNaught, A., Alcántara, R., Darsow, M., Guedj, M., and Ashburner, M. (2007). Chebi: a database and ontology for chemical entities of biological interest. *Nucleic acids research*, 36(suppl_1):D344–D350. 123

Dillies, M.-A., Rau, A., Aubert, J., Hennequet-Antier, C., Jeanmougin, M., Servant, N., Keime, C., Marot, G., Castel, D., Estelle, J., et al. (2013). A comprehensive evaluation of normalization methods for illumina high-throughput rna sequencing data analysis. *Briefings in bioinformatics*, 14(6):671–683. 90, 157

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):e105. 75, 76, 89

Dong, B., Qiu, J., Zheng, Q., Zhong, X., Li, J., and Li, Y. (2010). A novel approach to improving the efficiency of storing and accessing small files on hadoop: a case study by powerpoint files. In *Services Computing (SCC), 2010 IEEE International Conference on*, pages 65–72. IEEE. 126

Dongarra, J., Sterling, T., Simon, H., and Strohmaier, E. (2005). High-performance computing: clusters, constellations, mpps, and future directions. *Computing in Science & Engineering*, 7(2):51–59. 30

Dongarra, J. J., Otto, S. W., Snir, M., and Walker, D. (1995). An introduction to the mpi standard. *Communications of the ACM*, page 18. 33

Dudley, J. T., Pouliot, Y., Chen, R., Morgan, A. A., and Butte, A. J. (2010). Translational bioinformatics in the cloud: an affordable alternative. *Genome medicine*, 2(8):51. 21, 52

Dunn, O. J. (1961). Multiple comparisons among means. *Journal of the American Statistical Association*, 56(293):52–64. 182

Eastberg, J. H., Pelletier, J., and Stoddard, B. L. (2004). Recognition of DNA substrates by T4 bacteriophage polynucleotide kinase. *Nucleic acids research*, 32(2):653–60. 79, 86, 92, 95

Edgar, R. (2002). Gene Expression Omnibus: NCBI gene expression and hybridization array data repository. *Nucleic Acids Research*, 30(1):207–210. 17, 96

Ellingson, S. R. and Baudry, J. (2011). High-throughput virtual molecular docking: Hadoop implementation of autodock4 on a private cloud. In *Proceedings of the second international workshop on Emerging computational methods for the life sciences*, pages 33–38. ACM. 112, 114, 118, 124, 138, 153, 154

EMBL-EBI (2013a). Accessing ENA data programmatically: Retrieve SRA metadata in XML format. http://www.ebi.ac.uk/training/online/course/nucleotide-sequence-data-resources-ebi/accessing-ena-data-programmatically/. [Online; accessed 2-December-2013]. 97, 100

EMBL-EBI (2013b). Accessing ENA data programmatically: Retrieve SRA metadata in XML format. http://www.ebi.ac.uk/ena/submit/read-xml-format-1-4. [Online; accessed 2-December-2013]. 98

Estrada, T., Armen, R., and Taufer, M. (2010). Automatic selection of near-native protein-ligand conformations using a hierarchical clustering and volunteer computing. In *Proceedings of the First ACM International Conference on Bioinformatics and Computational Biology*, pages 204–213. ACM. 114

Estrada, T., Zhang, B., Cicotti, P., Armen, R. S., and Taufer, M. (2012). A scalable and accurate method for classifying protein–ligand binding geometries using a mapreduce approach. *Computers in biology and medicine*, 42(7):758–771. 113, 114, 120, 123, 153, 154

Feng, W., Sang, P., Lian, D., Dong, Y., Song, F., Li, M., He, B., Cao, F., and Liu, Y. (2015a). ResSeq: Enhancing Short-Read Sequencing Alignment By Rescuing Error-Containing Reads. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 12(4):795–798. 91, 217

Feng, Y., Zhang, Y., Ying, C., Wang, D., and Du, C. (2015b). Nanopore-based fourth-generation dna sequencing technology. *Genomics, proteomics & bioinformatics*, 13(1):4–16. 20, 67, 70

Fish, B., Kun, J., Lelkes, A. D., Reyzin, L., and Turán, G. (2015). On the computational complexity of mapreduce. In *International Symposium on Distributed Computing*, pages 1–15. Springer. 39, 40

Flynn, M. J. (1966). Very high-speed computing systems. *Proceedings of the IEEE*, 54(12):1901–1909. 32

Foster, I., Zhao, Y., Raicu, I., and Lu, S. (2008). Cloud computing and grid computing 360-degree compared. In *Grid Computing Environments Workshop, 2008. GCE'08*, pages 1–10. Ieee. 31

FreePascal (2017). Open source compiler for Pascal and Object Pascal. https://www.freepascal.org/. [Online; accessed 15-November-2016]. 149

Functional Genomics Data Society, F. (2012a). MIAME: Minimum Information About a Microarray Experiment. http://fged.org/projects/miame/. [Online; accessed 10-October-2016]. 96, 97

Functional Genomics Data Society, F. (2012b). MINSEQE: Minimum Information about a high throughput Nucleotide SeQuencing Experiment - a proposal for standards in functional genomic data reporting. http://www.mged.org/minseqe/MINSEQE.pdf. [Online; accessed 10-October-2016]. 96, 108

Galaxy project (2017). Galaxy - Running tools on a cluster. https://galaxyproject.org/admin/config/performance/cluster/. [Online; accessed 26-June-2017]. 54

Garber, M., Grabherr, M. G., Guttman, M., and Trapnell, C. (2011). Computational methods for transcriptome annotation and quantification using rna-seq. *Nature methods*, 8(6):469–477. 65, 91

García-Alcalde, F., Okonechnikov, K., Carbonell, J., Cruz, L. M., Götz, S., Tarazona, S., Dopazo, J., Meyer, T. F., and Conesa, A. (2012). Qualimap: evaluating next-generation sequencing alignment data. *Bioinformatics*, 28(20):2678–2679. 208, 212

García-García, G., Baux, D., Faugère, V., Moclyn, M., Koenig, M., Claustres, M., and Roux, A.-F. (2016). Assessment of the latest ngs enrichment capture methods in clinical context. *Scientific reports*, 6:20948. 83

Genomics England (2014). 100,000 Genomes project by numbers. https://www.genomicsengland.co.uk/the-100000-genomes-project-by-numbers/. [Online; accessed 24-November-2017]. 21

GEO, NCBI (2016). Gene Expression Omnibus. https://www.ncbi.nlm.nih.gov/geo/. [Online; accessed 12-October-2016]. 71

George, L. (2011). *HBase: The Definitive Guide: Random Access to Your Planet-Size Data*. " O'Reilly Media, Inc.". 50

Giardine, B., Riemer, C., Hardison, R. C., Burhans, R., Elnitski, L., Shah, P., Zhang, Y., Blankenberg, D., Albert, I., Taylor, J., et al. (2005). Galaxy: a platform for interactive large-scale genome analysis. *Genome research*, 15(10):1451–1455. 53

Gibrat, J.-F., Madej, T., and Bryant, S. H. (1996). Surprising similarities in structure comparison. *Current opinion in structural biology*, 6(3):377–385. 116

Gilbert, W. (1978). Why genes in pieces? *Nature*, 271(5645):501–501. 157

Gilbert, W. (1986). Origin of life: The rna world. *nature*, 319(6055). 16

Gnirke, A., Melnikov, A., Maguire, J., Rogov, P., LeProust, E. M., Brockman, W., Fennell, T., Giannoukos, G., Fisher, S., Russ, C., et al. (2009). Solution hybrid selection with ultra-long oligonucleotides for massively parallel targeted sequencing. *Nature biotechnology*, 27(2):182–189. 82

Goecks, J., Nekrutenko, A., and Taylor, J. (2010). Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences. *Genome biology*, 11(8):R86. 53

Goodwin, S., Gurtowski, J., Ethe-Sayers, S., Deshpande, P., Schatz, M. C., and McCombie, W. R. (2015). Oxford nanopore sequencing, hybrid error correction, and de novo assembly of a eukaryotic genome. *Genome research*, 25(11):1750–1756. 69, 76

Gordon, A. and Hannon, G. (2010). Fastx-toolkit. *FASTQ/A short-reads preprocessing tools (unpublished) http://hannonlab. cshl. edu/fastx_toolkit*. 165

Gramates, L. S., Marygold, S. J., dos Santos, G., Urbano, J.-M., Antonazzo, G., Matthews, B. B., Rey, A. J., Tabone, C. J., Crosby, M. A., Emmert, D. B., et al. (2016). Flybase at 25: looking to the future. *Nucleic Acids Research*, page gkw1016. 165

Grokhovsky, S. (2006). Specificity of dna cleavage by ultrasound. *Molecular Biology*, 40(2):276–283. 78, 85

Grokhovsky, S. L., Il'Icheva, I. A., Nechipurenko, D. Y., Golovkin, M. V., Panchenko, L. A., Polozov, R. V., and Nechipurenko, Y. D. (2011). Sequence-specific ultrasonic cleavage of dna. *Biophysical journal*, 100(1):117–125. 85

Gunarathne, T., Wu, T.-L., Qiu, J., and Fox, G. (2010). Mapreduce in the clouds for science. In *Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on*, pages 565–572. IEEE. 53

Guo, Z., Fox, G., and Zhou, M. (2012). Investigation of data locality in mapreduce. In *Proceedings of the 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (ccgrid 2012)*, pages 419–426. IEEE Computer Society. 37

Haas, B. J., Papanicolaou, A., Yassour, M., Grabherr, M., Blood, P. D., Bowden, J., Couger, M. B., Eccles, D., Li, B., Lieber, M., et al. (2013). De novo transcript sequence reconstruction from rna-seq using the trinity platform for reference generation and analysis. *Nature protocols*, 8(8):1494–1512. 66

Hall, L. (1993). *Sequencing Using the Du Pont Genesis^{TM} 2000 DNA Analysis System*, pages 357–372. Humana Press, Totowa, NJ. 62

Hansen, K. D., Brenner, S. E., and Dudoit, S. (2010). Biases in Illumina transcriptome sequencing caused by random hexamer priming. *Nucleic acids research*, 38(12):e131. 74, 88, 89, 95, 155, 158, 192, 195

Hansen, K. D., Irizarry, R. A., and Zhijin, W. (2012). Removing technical variability in rna-seq data using conditional quantile normalization. *Biostatistics*, 13(2):204–216. 90, 91

Harismendy, O., Ng, P. C., Strausberg, R. L., Wang, X., Stockwell, T. B., Beeson, K. Y., Schork, N. J., Murray, S. S., Topol, E. J., Levy, S., et al. (2009). Evaluation

of next generation sequencing platforms for population targeted sequencing studies. *Genome biology*, 10(3):R32. 75, 76

Heather, J. M. and Chain, B. (2016). The sequence of sequencers: The history of sequencing dna. *Genomics*, 107(1):1–8. 17, 63

Hill, M. D., Jouppi, N. P., and Sohi, G. (2000). *Readings in computer architecture*. Gulf Professional Publishing. 32

Hillier, L. W., Marth, G. T., Quinlan, A. R., Dooling, D., Fewell, G., Barnett, D., Fox, P., Glasscock, J. I., Hickenbotham, M., Huang, W., et al. (2008). Whole-genome sequencing and variant discovery in c. elegans. *Nature methods*, 5(2):183–188. 76

Holm, L. and Sander, C. (1998). Touring protein fold space with dali/fssp. *Nucleic acids research*, 26(1):316–319. 116

Hortonworks (2016). About Hortonworks. https://hortonworks.com/about-us/. [Online; accessed 01-February-2018]. 125

Housby, J. (1998). Fidelity of DNA ligation: a novel experimental approach based on the polymerisation of libraries of oligonucleotides. *Nucleic Acids Research*, 26(18):4259–4266. 80, 95

Huang, N., Shoichet, B. K., and Irwin, J. J. (2006). Benchmarking sets for molecular docking. *Journal of medicinal chemistry*, 49(23):6789–6801. 112

Hughes, J. (1989). Why functional programming matters. *The computer journal*, 32(2):98–107. 40

Hung, C.-L. and Hua, G.-J. (2013). Cloud computing for protein-ligand binding site comparison. *BioMed research international*, 2013. 112, 114, 119, 121, 153, 154

Hung, C.-L. and Lin, Y.-L. (2013). Implementation of a parallel protein structure alignment service on cloud. *International journal of genomics*, 2013. 116

Hussain, H., Malik, S. U. R., Hameed, A., Khan, S. U., Bickler, G., Min-Allah, N., Qureshi, M. B., Zhang, L., Yongji, W., Ghani, N., et al. (2013). A survey on resource allocation in high performance distributed computing systems. *Parallel Computing*, 39(11):709–736. 30

IBM (2014a). IBM Support: Technote. http://www-01.ibm.com/support/docview.wss?uid=isg3T1021046. [Online; accessed 11-March-2017]. 133

IBM (2014b). Job arrays: IBM Platform LSF v9.1.3 documentation. http://capsella.ccs.yorku.ca/homepage/lsfhpc/admin/jobarrays.html. [Online; accessed 11-March-2017]. 152

IBM / Openlava (2017). Openlava implementation of LSF documentation. http://www.openlava.org/documentation/man1/lsfintro.1.html. [Online; accessed 14-April-2017]. 131, 133, 151

ICGC (2017). ICGC Cancer Genome Projects by Cancer type and lead jurisdiction. http://icgc.org/icgc/cgp. [Online; accessed 24-November-2017]. 21

Illumina (2016). TruSeq RNA sample preparation guide v2. https://support.illumina.com/content/dam/illumina-support/documents/documentation/chemistry_documentation/samplepreps_truseq/truseqrna/truseq-rna-sample-prep-v2-guide-15026495-f.pdf. [Online; accessed 24-Oct-2017]. 79

International Human Genome Sequencing Consortium and others (2004). Finishing the euchromatic sequence of the human genome. *Nature*, 431(7011):931–945. 17

Islam, S., Zeisel, A., Joost, S., La Manno, G., Zajac, P., Kasper, M., Lönnerberg, P., and Linnarsson, S. (2014). Quantitative single-cell rna-seq with unique molecular identifiers. *Nature methods*, 11(2):163–166. 81

James Gallagher, BBC (2014). DNA project 'to make UK world genetic research leader'. http://www.bbc.co.uk/news/health-28488313. [Online; accessed 21-January-2017]. 18

Ji, Y., Xu, Y., Zhang, Q., Tsui, K.-W., Yuan, Y., Norris Jr., C., Liang, S., and Liang, H. (2011). BM-Map: Bayesian Mapping of Multireads for Next-Generation Sequencing Data. *Biometrics*, 67(4):1215–1224. 91, 217

Jiao, X., Rosenlund, M., Hooper, S. D., Tellgren-Roth, C., He, L., Fu, Y., Mangion, J., and Sjöblom, T. (2011). Structural alterations from multiple displacement amplification of a human genome revealed by mate-pair sequencing. *PloS one*, 6(7):e22250. 82, 86, 95

Jin, H. and Sun, X.-H. (2013). Performance comparison under failures of mpi and mapreduce: An analytical approach. *Future Generation Computer Systems*, 29(7):1808–1815. 33

John, K., Botkin, D., Burton, A., Castro-Wallace, S., Chaput, J., Dworkin, J., Lehman, N., Lupisella, M., Mason, C., Smith, D., et al. (2016). The biomolecule sequencer project: Nanopore sequencing as a dual-use tool for crew health and astrobiology investigations. 20

Jones, D. T. (1999). Genthreader: an efficient and reliable protein fold recognition method for genomic sequences. *Journal of molecular biology*, 287(4):797–815. 123

Jones, D. T. (2007). Improving the accuracy of transmembrane protein topology prediction using evolutionary information. *Bioinformatics*, 23(5):538–544. 123

Joshua, J., Alao, D., Okolie, S., and Awodele, O. (2013). Software ecosystem: Features, benefits and challenges. 37

Kahn, S. D. (2011). On the future of genomic data. *Science*, 331(6018):728–729. 18, 54

Kamps-Hughes, N., Quimby, A., Zhu, Z., and Johnson, E. A. (2013). Massively parallel characterization of restriction endonucleases. *Nucleic acids research*, 41(11):e119. 78, 85, 92, 95

Kantarcioglu, M., Jiang, W., Liu, Y., and Malin, B. (2008). A cryptographic approach to securely share and query genomic sequences. *IEEE Transactions on information technology in biomedicine*, 12(5):606–617. 54

Kaplan, J. A. and Nelson, M. L. (1993). A comparison of queueing, cluster and distributed computing systems. 33, 131

Keohavong, P. and Thilly, W. G. (1989). Fidelity of DNA polymerases in DNA amplification. *Proceedings of the National Academy of Sciences of the United States of America*, 86(23):9253–7. 78, 79, 81, 85, 92, 95

Kilianski, A., Haas, J. L., Corriveau, E. J., Liem, A. T., Willis, K. L., Kadavy, D. R., Rosenzweig, C. N., and Minot, S. S. (2015). Bacterial and viral identification and differentiation by amplicon sequencing on the minion nanopore sequencer. *Gigascience*, 4(1):1. 69

Kleywegt, G. J. and Jones, T. A. (1996). Phi/psi-chology: Ramachandran revisited. *Structure*, 4(12):1395–1400. 132

Kolodny, R. and Linial, N. (2004). Approximate protein structural alignment in polynomial time. *Proceedings of the National Academy of Sciences of the United States of America*, 101(33):12201–12206. 116

Konagurthu, A. S., Whisstock, J. C., Stuckey, P. J., and Lesk, A. M. (2006). Mustang: a multiple structural alignment algorithm. *Proteins: Structure, Function, and Bioinformatics*, 64(3):559–574. 116

Konc, J. and Janežič, D. (2010). Probis algorithm for detection of structurally similar protein binding sites by local structural alignment. *Bioinformatics*, 26(9):1160–1168. 116

Kouranov, A., Xie, L., de la Cruz, J., Chen, L., Westbrook, J., Bourne, P. E., and Berman, H. M. (2006). The rcsb pdb information portal for structural genomics. *Nucleic acids research*, 34(suppl 1):D302–D305. 111

Kozarewa, I., Ning, Z., Quail, M. A., Sanders, M. J., Berriman, M., and Turner, D. J. (2009). Amplification-free Illumina sequencing-library preparation facilitates improved mapping and assembly of (G+C)-biased genomes. *Nature methods*, 6(4):291–5. 80, 81, 86, 92, 95

Krieger, E. and Vriend, G. (2002). Models@ home: distributed computing in bioinformatics using a screensaver based approach. *Bioinformatics*, 18(2):315–318. 31

Kudtarkar, P., DeLuca, T. F., Fusaro, V. A., Tonellato, P. J., and Wall, D. P. (2010). Cost-effective cloud computing: a case study using the comparative genomics tool, roundup. *Evolutionary Bioinformatics*, 6:197. 54

Kukurba, K. R. and Montgomery, S. B. (2015). Rna sequencing and analysis. *Cold Spring Harbor Protocols*, 2015(11):pdb–top084970. 65, 67, 89

Lahens, N. F., Kavakli, I. H., Zhang, R., Hayer, K., Black, M. B., Dueck, H., Pizarro, A., Kim, J., Irizarry, R., Thomas, R. S., et al. (2014). Ivt-seq reveals extreme bias in rna sequencing. *Genome biology*, 15(6):R86. 74, 89, 90, 156, 158, 164, 165, 185, 214, 216, 218, 221, 222

Lahens NF, Kavakli IH, Z. R. H. K. B. M. D. H. P. A. K. J. I. R. T. R. G. G. H. J. (2012). GSE50445: IVT-seq reveals extreme bias in RNA-sequencing. https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE50445. [Online; accessed 01-November-2017]. 164

Laney, D. (2001). 3D data management: Controlling data volume, velocity, and variety. Technical report, META Group. 17, 43, 54

Langmead, B., Hansen, K. D., and Leek, J. T. (2010). Cloud-scale rna-sequencing differential expression analysis with myrna. *Genome biology*, 11(8):R83. 50

Langmead, B., Schatz, M. C., Lin, J., Pop, M., and Salzberg, S. L. (2009). Searching for snps with cloud computing. *Genome biology*, 10(11):R134. 50

Lasken, R. S. and Stockwell, T. B. (2007). Mechanism of chimera formation during the multiple displacement amplification reaction. *BMC biotechnology*, 7(1):19. 86, 87

Laskowski, R. A., MacArthur, M. W., Moss, D. S., and Thornton, J. M. (1993). Procheck: a program to check the stereochemical quality of protein structures. *Journal of applied crystallography*, 26(2):283–291. 132

Lauter, K., López-Alt, A., and Naehrig, M. (2014). Private computation on encrypted genomic data. In *International Conference on Cryptology and Information Security in Latin America*, pages 3–27. Springer. 54

LaVan, D. A., Lynn, D. M., and Langer, R. (2002). Moving smaller in drug discovery and delivery. *Nature Reviews Drug Discovery*, 1(1):77–84. 69

Laver, T., Harrison, J., ONeill, P., Moore, K., Farbos, A., Paszkiewicz, K., and Studholme, D. J. (2015). Assessing the performance of the oxford nanopore technologies minion. *Biomolecular detection and quantification*, 3:1–8. 20, 67, 69, 70, 76

Ledford, H. (2016). Astrazeneca launches project to sequence 2 million genomes. *Nature*, 532(7600):427. 19

Leinonen, R., Sugawara, H., and Shumway, M. (2011). The sequence read archive. *Nucleic acids research*, 39(Database issue):D19–21. 17, 96, 99, 109

Leipzig, J. (2016). A review of bioinformatic pipeline frameworks. *Briefings in bioinformatics*, page bbw020. 50, 53

Li, H., Handsaker, B., Wysoker, A., Fennell, T., Ruan, J., Homer, N., Marth, G., Abecasis, G., Durbin, R., et al. (2009a). The sequence alignment/map format and samtools. *Bioinformatics*, 25(16):2078–2079. 161, 163

Li, J., Jiang, H., and Wong, W. H. (2010). Modeling non-uniformity in short-read rates in rna-seq data. *Genome biology*, 11(5):1. 155

Li, P., Piao, Y., Shon, H. S., and Ryu, K. H. (2015). Comparing the normalization methods for the differential analysis of illumina high-throughput rna-seq data. *BMC bioinformatics*, 16(1):347. 90, 157

Li, R., Li, Y., Fang, X., Yang, H., Wang, J., Kristiansen, K., and Wang, J. (2009b). Snp detection for massively parallel whole-genome resequencing. *Genome research*, 19(6):1124–1132. 50

Liu, G., Liu, M., Chen, D., Chen, L., Zhu, J., Zhou, B., and Gao, J. (2016). Predicting protein ligand binding sites with structure alignment method on hadoop. *Current Proteomics*, 13(2):113–121. 114, 116, 121, 123, 153, 154

Liu, R., Loraine, A. E., and Dickerson, J. A. (2014). Comparisons of computational methods for differential alternative splicing detection using rna-seq in plant systems. *BMC bioinformatics*, 15(1):1. 65

Lobley, A. E., Nugent, T., Orengo, C. A., and Jones, D. T. (2008). Ffpred: an integrated feature-based function prediction server for vertebrate proteomes. *Nucleic acids research*, 36(suppl_2):W297–W302. 123

Lodish, H., Baltimore, D., Berk, A., Zipursky, S. L., Matsudaira, P., and Darnell, J. (1995). *Molecular cell biology*, volume 3. Scientific American Books New York. 58

Loman, N. J., Quick, J., and Simpson, J. T. (2015). A complete bacterial genome assembled de novo using only nanopore sequencing data. *Nature methods*, 12(8):733–735. 20

Love, M. I., Hogenesch, J. B., and Irizarry, R. A. (2016). Modeling of rna-seq fragment sequence bias reduces systematic errors in transcript abundance estimation. *Nature Biotechnology*, 34(12):1287–1291. 91

Lykke-Andersen, S. and Jensen, T. H. (2006). Cut it out: silencing of noise in the transcriptome. *Nature Structural and Molecular Biology*, 13(10):860. 177

Lyubimov, D. and Palumbo, A. (2016). *Apache Mahout: Beyond MapReduce*. CreateSpace Independent Publishing Platform. 50

Ma, B., Elkayam, T., Wolfson, H., and Nussinov, R. (2003). Protein–protein interactions: structurally conserved residues distinguish between binding sites and exposed protein surfaces. *Proceedings of the National Academy of Sciences*, 100(10):5772–5777. 116

Mackey, G., Sehrish, S., and Wang, J. (2009). Improving metadata management for small files in hdfs. In *Cluster Computing and Workshops, 2009. CLUSTER'09. IEEE International Conference on*, pages 1–4. IEEE. 126

Malone, J. H. and Oliver, B. (2011). Microarrays, deep sequencing and the true measure of the transcriptome. *BMC Biology*, 9(1):34. 67, 71

Mamanova, L., Coffey, A. J., Scott, C. E., Kozarewa, I., Turner, E. H., Kumar, A., Howard, E., Shendure, J., and Turner, D. J. (2010). Target-enrichment strategies for next-generation sequencing. *Nature methods*, 7(2):111–118. 82

Mardis, E. R. (2006). Anticipating the 1,000 dollar genome. *Genome biology*, 7(7):112. 62, 73

Mardis, E. R. (2011). A decade's perspective on DNA sequencing technology. *Nature*, 470(7333):198–203. 17

Margulies, M., Egholm, M., Altman, W. E., Attiya, S., Bader, J. S., Bemben, L. A., Berka, J., Braverman, M. S., Chen, Y.-J., Chen, Z., et al. (2005). Genome sequencing in microfabricated high-density picolitre reactors. *Nature*, 437(7057):376–380. 63, 75

Mariette, J., Noirot, C., and Klopp, C. (2011). Assessment of replicate bias in 454 pyrosequencing and a multi-purpose read-filtering tool. *BMC research notes*, 4(1):149. 75

Marx, V. (2015). The dna of a nation. *Nature*, 524(7566):503–505. 19, 21

Masella, A. P., Bartram, A. K., Truszkowski, J. M., Brown, D. G., and Neufeld, J. D. (2012). Pandaseq: paired-end assembler for illumina sequences. *BMC bioinformatics*, 13(1):31. 84, 87

Massie, M. L., Chun, B. N., and Culler, D. E. (2004). The ganglia distributed monitoring system: design, implementation, and experience. *Parallel Computing*, 30(7):817–840. 138

Matsunaga, A., Tsugawa, M., and Fortes, J. (2008). Cloudblast: Combining mapreduce and virtualization on distributed resources for bioinformatics applications. In *eScience, 2008. eScience'08. IEEE Fourth International Conference on*, pages 222–229. IEEE. 50

McGuffin, L. J., Bryson, K., and Jones, D. T. (2000). The psipred protein structure prediction server. *Bioinformatics*, 16(4):404–405. 123

McIntyre, A. B., Rizzardi, L., Angela, M. Y., Rosen, G. L., Alexander, N., Botkin, D. J., John, K. K., Castro-Wallace, S. L., Burton, A. S., Feinberg, A., et al. (2015). Nanopore sequencing in microgravity. *BioRxiv*, page 032342. 20

McKenna, A., Hanna, M., Banks, E., Sivachenko, A., Cibulskis, K., Kernytsky, A., Garimella, K., Altshuler, D., Gabriel, S., Daly, M., et al. (2010). The genome analysis toolkit: a mapreduce framework for analyzing next-generation dna sequencing data. *Genome research*, 20(9):1297–1303. 50

Meacham, F., Boffelli, D., Dhahbi, J., Martin, D. I. K., Singer, M., and Pachter, L. (2011). Identification and correction of systematic error in high-throughput sequence data. *BMC bioinformatics*, 12(1):451. 72, 74

Mell, P., Grance, T., et al. (2011). The nist definition of cloud computing. 31

Memon, F. N., Owen, A. M., Sanchez-Graillet, O., Upton, G. J., and Harrison, A. P. (2010). Identifying the impact of g-quadruplexes on affymetrix 3'arrays using cloud computing. *Journal of integrative bioinformatics*, 7(111). 72, 155, 176, 181

Mendenhall, W. and Sincich, T. (1992). *Statistics for engineering and the sciences.* Dellen Publishing Company. San Francisco, CA, USA. 177, 181

Meng, X.-Y., Zhang, H.-X., Mezei, M., and Cui, M. (2011). Molecular docking: a powerful approach for structure-based drug discovery. *Current computer-aided drug design*, 7(2):146–157. 112, 132

Message Passing Interface Forum (1993). MPI: a message passing interface standard. 32

Messerschmitt, D. G., Szyperski, C., et al. (2005). Software ecosystem: understanding an indispensable technology and industry. *MIT Press Books*, 1. 37

Metzker, M. L. (2010). Sequencing technologies - the next generation. *Nature reviews. Genetics*, 11(1):31–46. 17, 62, 72

Mikheyev, A. S. and Tin, M. M. (2014). A first look at the oxford nanopore minion sequencer. *Molecular ecology resources*, 14(6):1097–1102. 23, 69

Miller, F. P., Vandome, A. F., and McBrewster, J. (2010a). Amazon web services. 32, 110

Miller, J. R., Koren, S., and Sutton, G. (2010b). Assembly algorithms for next-generation sequencing data. *Genomics*, 95(6):315–27. 73

Morris, A. L., MacArthur, M. W., Hutchinson, E. G., and Thornton, J. M. (1992). Stereochemical quality of protein structure coordinates. *Proteins: Structure, Function, and Bioinformatics*, 12(4):345–364. 131

Morris, G. M. and Lim-Wilby, M. (2008). Molecular docking. *Molecular modeling of proteins*, pages 365–382. 112, 132

Mortazavi, A., Williams, B. A., McCue, K., Schaeffer, L., and Wold, B. (2008). Mapping and quantifying mammalian transcriptomes by rna-seq. *Nature methods*, 5(7):621–628. 65, 157

Moses, H., Dorsey, E. R., Matheson, D. H., and Thier, S. O. (2005). Financial anatomy of biomedical research. *Jama*, 294(11):1333–1342. 112

Mrozek, D., Małysiak-Mrozek, B., and Kłapciński, A. (2014). Cloud4psi: cloud computing for 3d protein structure similarity searching. *Bioinformatics*, 30(19):2822–2825. 117, 122, 123

Murthy, A. C., Vavilapalli, V. K., Eadline, D., Niemiec, J., and Markham, J. (2013). *Apache Hadoop YARN: moving beyond MapReduce and batch processing with Apache Hadoop 2*. Pearson Education. 47

Mutz, K.-O., Heilkenbrinker, A., Lönne, M., Walter, J.-G., and Stahl, F. (2013). Transcriptome analysis using next-generation sequencing. *Current opinion in biotechnology*, 24(1):22–30. 89

Nakamura, K., Oshima, T., Morimoto, T., Ikeda, S., Yoshikawa, H., Shiwa, Y., Ishikawa, S., Linak, M. C., Hirai, A., Takahashi, H., et al. (2011). Sequence-specific error profile of illumina sequencers. *Nucleic acids research*, 39(13):e90–e90. 76

Nakazato, T. and et al, O. (2013). Experimental Design-Based Functional Mining and Characterization of High-Throughput Sequencing Data in the Sequence Read Archive. *PLoS ONE*, 8(10):e77910. 97, 107

Nam, D. K., Lee, S., Zhou, G., Cao, X., Wang, C., Clark, T., Chen, J., Rowley, J. D., and Wang, S. M. (2002). Oligo (dt) primer generates a high frequency of truncated cdnas through internal poly (a) priming during reverse transcription. *Proceedings of the National Academy of Sciences*, 99(9):6152–6156. 88, 218, 222

Naval-Sánchez, M., Potier, D., Haagen, L., Sánchez, M., Munck, S., Van de Sande, B., Casares, F., Christiaens, V., and Aerts, S. (2013). Comparative motif discovery combined with comparative transcriptomics yields accurate targetome and enhancer predictions. *Genome research*, 23(1):74–88. 164, 191, 217, 221

Nechipurenko, D. Y., Ilicheva, I., Khodikov, M., Poptsova, M., Nechipurenko, Y. D., and Grokhovsky, S. (2014). Modeling of mechanochemical dna cleavage by the action of ultrasound. *Biophysics*, 59(6):861–868. 85

Nekrutenko, A. and Taylor, J. (2012). Next-geoneration sequencing data interpretation: enhancing reproducibility and accessibility. *Nature Reviews Genetics*, 13(9):667–672. 65, 94

Nguyen, T., Shi, W., and Ruden, D. (2011). Cloudaligner: A fast and full-featured mapreduce based tool for sequence mapping. *BMC research notes*, 4(1):171. 50

Niemenmaa, M., Kallio, A., Schumacher, A., Klemelä, P., Korpelainen, E., and Heljanko, K. (2012). Hadoop-bam: directly manipulating next generation sequencing data in the cloud. *Bioinformatics*, 28(6):876–877. 50

NIH (2008). Scientists Form International Cancer Genome Consortium. https://www.nih.gov/news-events/news-releases/scientists-form-international-cancer-genome-consortium. [Online; accessed 24-November-2017]. 19

Nikhil Joshi, UC Davis (2016). A windowed adaptive trimming tool for FASTQ files using quality. http://hadoop.apache.org/docs/r2.7.2/. [Online; accessed 30-Nov-2017]. 84, 87

Nikolenko, S. I., Korobeynikov, A. I., and Alekseyev, M. A. (2013). Bayeshammer: Bayesian clustering for error correction in single-cell sequencing. *BMC genomics*, 14(1):S7. 84, 87

Niu, B., Fu, L., Sun, S., and Li, W. (2010). Artificial and natural duplicates in pyrosequencing reads of metagenomic data. *BMC bioinformatics*, 11(1):187. 75

Ocaña, K., Benza, S., de Oliveira, D., Dias, J., and Mattoso, M. (2014). Exploring large scale receptor-ligand pairs in molecular docking workflows in hpc clouds. In *Parallel & Distributed Processing Symposium Workshops (IPDPSW), 2014 IEEE International*, pages 536–545. IEEE. 115, 121

Okou, D. T., Steinberg, K. M., Middle, C., Cutler, D. J., Albert, T. J., and Zwick, M. E. (2007). Microarray-based genomic selection for high-throughput resequencing. *Nature methods*, 4(11). 82

Olston, C., Reed, B., Srivastava, U., Kumar, R., and Tomkins, A. (2008). Pig latin: a not-so-foreign language for data processing. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1099–1110. ACM. 50

Orengo, C. A., Michie, A., Jones, S., Jones, D. T., Swindells, M., and Thornton, J. M. (1997). Cath–a hierarchic classification of protein domain structures. *Structure*, 5(8):1093–1109. 116

Orengo, C. A. and Taylor, W. R. (1996). [36] ssap: sequential structure alignment program for protein structure comparison. *Methods in enzymology*, 266:617–635. 116

Orlowski, J. and Bujnicki, J. M. (2008). Structural and evolutionary classification of Type II restriction enzymes based on theoretical and experimental analyses. *Nucleic acids research*, 36(11):3552–69. 78, 95

Ozsolak, F., Platt, A. R., Jones, D. R., Reifenberger, J. G., Sass, L. E., McInerney, P., Thompson, J. F., Bowers, J., Jarosz, M., and Milos, P. M. (2009). Direct rna sequencing. *Nature*, 461(7265):814–818. 67

OMalley, O. and Murthy, A. C. (2009). Winning a 60 second dash with a yellow elephant. 43, 111

ORoak, B. J., Vives, L., Fu, W., Egertson, J. D., Stanaway, I. B., Phelps, I. G., Carvill, G., Kumar, A., Lee, C., Ankenman, K., et al. (2012). Multiplex targeted sequencing identifies recurrently mutated genes in autism spectrum disorders. *Science*, 338(6114):1619–1622. 82

Pace, N. R. and Marsh, T. L. (1985). Rna catalysis and the origin of life. *Origins of Life and Evolution of Biospheres*, 16(2):97–116. 16

Park, J. W., Tokheim, C., Shen, S., and Xing, Y. (2013). Identifying differential alternative splicing events from rna sequencing data using rnaseq-mats. *Deep Sequencing Data Analysis*, pages 171–179. 65

Paschina, G., Roverelli, L., DAgostino, D., Chiappori, F., and Merelli, I. (2015). Clustering protein structures with hadoop. In *International Meeting on Computational Intelligence Methods for Bioinformatics and Biostatistics*, pages 141–153. Springer. 115, 122, 135, 154

Patro, R., Duggal, G., Love, M. I., Irizarry, R. A., and Kingsford, C. (2017). Salmon provides fast and bias-aware quantification of transcript expression. *Nature Methods*. 91

Patro, R., Mount, S. M., and Kingsford, C. (2014). Sailfish enables alignment-free isoform quantification from rna-seq reads using lightweight algorithms. *Nature biotechnology*, 32(5):462–464. 91

Pence, H. E. and Williams, A. (2010). Chemspider: an online chemical information resource. 123

Perneger, T. V. (1998). What's wrong with bonferroni adjustments. *Bmj*, 316(7139):1236–1238. 182

Pickrell, J. K., Marioni, J. C., Pai, A. A., Degner, J. F., Engelhardt, B. E., Nkadori, E., Veyrieras, J.-B., Stephens, M., Gilad, Y., and Pritchard, J. K. (2010). Understanding mechanisms underlying human gene expression variation with rna sequencing. *Nature*, 464(7289):768–772. 91

Prlić, A., Yates, A., Bliven, S. E., Rose, P. W., Jacobsen, J., Troshin, P. V., Chapman, M., Gao, J., Koh, C. H., Foisy, S., et al. (2012). Biojava: an open-source framework for bioinformatics in 2012. *Bioinformatics*, 28(20):2693–2695. 117

Prober, J. M., Trainor, G. L., Dam, R. J., Hobbs, F. W., Robertson, C. W., Zagursky, R. J., Cocuzza, A. J., Jensen, M. A., and Baumeister, K. (1987). A system for rapid dna sequencing with fluorescent chain-terminating dideoxynucleotides. *Science*, 238(4825):336–341. 62

PubMed, N. (2016). PubMed - NCBI. https://www.ncbi.nlm.nih.gov/pubmed/. [Online; accessed 06-November-2016]. 95

Qiu, X., Ekanayake, J., Beason, S., Gunarathne, T., Fox, G., Barga, R., and Gannon, D. (2009). Cloud technologies for bioinformatics applications. In *Proceedings of the 2nd Workshop on Many-task Computing on Grids and Supercomputers*, page 6. ACM. 54

Quail, M. A., Otto, T. D., Gu, Y., Harris, S. R., Skelly, T. F., McQuillan, J. A., Swerdlow, H. P., and Oyola, S. O. (2012). Optimal enzymes for amplifying sequencing libraries. *Nature methods*, 9(1):10–1. 81, 86

Quince, C., Lanzén, A., Curtis, T. P., Davenport, R. J., Hall, N., Head, I. M., Read, L. F., and Sloan, W. T. (2009). Accurate determination of microbial diversity from 454 pyrosequencing data. *Nature methods*, 6(9):639–641. 75

Rao, B. T., Sridevi, N., Reddy, V. K., and Reddy, L. (2012). Performance issues of heterogeneous hadoop clusters in cloud computing. *arXiv preprint arXiv:1207.0894*. 31

Rawlins, M. D. (2004). Cutting the cost of drug development? *Nature reviews Drug discovery*, 3(4):360–364. 112

Rayner, T. F., Rocca-Serra, P., Spellman, P. T., Causton, H. C., Farne, A., Holloway, E., Irizarry, R. A., Liu, J., Maier, D. S., Miller, M., et al. (2006). A simple

spreadsheet-based, miame-supportive format for microarray data: Mage-tab. *Bmc Bioinformatics*, 7(1):1. 96

Raz, T., Kapranov, P., Lipson, D., Letovsky, S., Milos, P. M., and Thompson, J. F. (2011). Protocol dependence of sequencing-based gene expression measurements. *PloS one*, 6(5):e19287. 67, 76, 94, 95

Red Hat (2017). Red Hat Enterprise Linux documentation: Swap space. https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/5/html/deployment_guide/ch-swapspace. [Online; accessed 01-April-2018]. 148

Regalado, A. (2015). The Search for Exceptional Genomes. https://www.technologyreview.com/s/534591/us-to-develop-dna-study-of-one-million-people/. [Online; accessed 02-February-2017]. 16

Rehm, H. L. (2013). Disease-targeted sequencing: a cornerstone in the clinic. *Nature Reviews Genetics*, 14(4):295–300. 82

Reinartz, J., Bruyns, E., Lin, J.-Z., Burcham, T., Brenner, S., Bowen, B., Kramer, M., and Woychik, R. (2002). Massively parallel signature sequencing (mpss) as a tool for in-depth quantitative gene expression profiling in all organisms. *Briefings in Functional Genomics*, 1(1):95–104. 88

Reyes-Ortiz, J. L., Oneto, L., and Anguita, D. (2015). Big data analytics in the cloud: Spark on hadoop vs mpi/openmp on beowulf. *Procedia Computer Science*, 53:121–130. 40

Rhee, M. and Burns, M. A. (2006). Nanopore sequencing technology: research trends and applications. *Trends in biotechnology*, 24(12):580–586. 20

Richard, H., Schulz, M. H., Sultan, M., Nürnberger, A., Schrinner, S., Balzereit, D., Dagand, E., Rasche, A., Lehrach, H., Vingron, M., et al. (2010). Prediction of alternative isoforms from exon expression levels in rna-seq experiments. *Nucleic acids research*, 38(10):e112–e112. 67

Risso, D., Schwartz, K., Sherlock, G., and Dudoit, S. (2011). Gc-content normalization for rna-seq data. *BMC bioinformatics*, 12(1):480. 90, 91, 155, 218

Rivest, R. L., Adleman, L., and Dertouzos, M. L. (1978). On data banks and privacy homomorphisms. 54

Roach, J. C., Boysen, C., Wang, K., and Hood, L. (1995). Pairwise end sequencing: a unified approach to genomic mapping and sequencing. *Genomics*, 26(2):345–353. 62

Roberts, A., Trapnell, C., Donaghey, J., Rinn, J. L., and Pachter, L. (2011). Improving RNA-Seq expression estimates by correcting for fragment bias. *Genome biology*, 12(3):R22. 74, 89, 90, 91, 155, 157

Rocha, F. and Correia, M. (2011). Lucy in the sky without diamonds: Stealing confidential data in the cloud. In *Dependable Systems and Networks Workshops (DSN-W), 2011 IEEE/IFIP 41st International Conference on*, pages 129–134. IEEE. 54

Ronaghi, M. (2001). Pyrosequencing sheds light on dna sequencing. *Genome research*, 11(1):3–11. 63

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):84–89. 63

Ross, M. G., Russ, C., Costello, M., Hollinger, A., Lennon, N. J., Hegarty, R., Nusbaum, C., and Jaffe, D. B. (2013). Characterizing and measuring bias in sequence data. *Genome biology*, 14(5):R51. 73, 86, 158

Russell, S., Meadows, L. A., and Russell, R. R. (2008). *Microarray technology in practice*. Academic Press. 65, 71

Saha, S., Sparks, A. B., Rago, C., Akmaev, V., Wang, C. J., Vogelstein, B., Kinzler, K. W., and Velculescu, V. E. (2002). Using the transcriptome to annotate the genome. *Nature biotechnology*, 20(5):508–512. 88

Saiki, R. K., Scharf, S., Faloona, F., Mullis, K. B., Horn, G. T., Erlich, H. A., and Arnheim, N. (1985). Enzymatic amplification of b-globin genomic sequences and restriction site analysis for diagnosis of sickle cell anemia. *Science*, 230(4732):1350–1354. 61

Sakharkar, M. K., Chow, V. T., and Kangueane, P. (2004). Distributions of exons and introns in the human genome. *In silico biology*, 4(4):387–393. 185

Sambrook, J. and Russell, D. W. (2006a). Fragmentation of DNA by nebulization. *CSH protocols*, 2006(4):pdb.prot4539–. 78

Sambrook, J. and Russell, D. W. (2006b). Fragmentation of DNA by sonication. *CSH protocols*, 2006(4):pdb.prot4538–. 77, 78

Samet, H. (1988). An overview of quadtrees, octrees, and related hierarchical data structures. *NATO ASI Series*, 40:51–68. 114

Sanger (2013). Illumina library preparation for long PCR products. ftp://ftp.ncbi.nlm.nih.gov/pub/factsheets/Factsheet_SRA.pdf. [Online; accessed 10-October-2016]. 80

Sanger, F., Nicklen, S., and Coulson, A. R. (1977). Dna sequencing with chain-terminating inhibitors. *Proceedings of the National Academy of Sciences*, 74(12):5463–5467. 58

Sanger, N. (2012). Ensembl, GTFGFF file format specification. http://www.ensembl.org/info/website/upload/gff.html. [Online; accessed 15-November-2016]. 161, 163

Schatz, M. C. (2009). Cloudburst: highly sensitive read mapping with mapreduce. *Bioinformatics*, 25(11):1363–1369. 50

Schatz, M. C., Langmead, B., and Salzberg, S. L. (2010a). Cloud computing and the dna data race. *Nature biotechnology*, 28(7):691. 52

Schatz, M. C., Sommer, D., Kelley, D., and Pop, M. (2010b). De novo assembly of large genomes using cloud computing. In *Proceedings of the Cold Spring Harbor Biology of Genomes Conference*. 50, 51

Schirmer, M., Ijaz, U. Z., D'Amore, R., Hall, N., Sloan, W. T., and Quince, C. (2015). Insight into biases and sequencing errors for amplicon sequencing with the illumina miseq platform. *Nucleic acids research*, 43(6):e37–e37. 83, 87

Schwartz, S. L. and Farman, M. L. (2010). Systematic overrepresentation of DNA termini and underrepresentation of subterminal regions among sequencing templates prepared from hydrodynamically sheared linear DNA molecules. *BMC genomics*, 11(1):87. 78, 85, 95

Scott, W. R., Hünenberger, P. H., Tironi, I. G., Mark, A. E., Billeter, S. R., Fennen, J., Torda, A. E., Huber, T., Krüger, P., and van Gunsteren, W. F. (1999). The gromos biomolecular simulation program package. *The Journal of Physical Chemistry A*, 103(19):3596–3607. 115

Sealfon, S. C. and Chu, T. T. (2011). Rna and dna microarrays. *Biological Microarrays: Methods and Protocols*, pages 3–34. 17, 71

Seguin-Orlando, A., Schubert, M., Clary, J., Stagegaard, J., Alberdi, M. T., Prado, J. L., Prieto, A., Willerslev, E., and Orlando, L. (2013). Ligation bias in illumina next-generation DNA libraries: implications for sequencing ancient genomes. *PloS one*, 8(10):e78575. 80, 92, 95

SEQC/MAQC-III Consortium and others (2014). A comprehensive assessment of rna-seq accuracy, reproducibility and information content by the sequencing quality control consortium. *Nature biotechnology*, 32(9):903–914. 93

Sequence Read Archive (2017). Overview of the Sequence Read Archive (SRA). https://trace.ncbi.nlm.nih.gov/Traces/sra/sra.cgi/. [Online; accessed 3-January-2017]. 17

Shanahan, H. P., Owen, A. M., and Harrison, A. P. (2014). Bioinformatics on the cloud computing platform azure. *PloS one*, 9(7):e102642. 52, 53

Shanahan, J. G. and Dai, L. (2015). Large scale distributed data science using apache spark. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 2323–2324. ACM. 48

Shendure, J. and Ji, H. (2008). Next-generation dna sequencing. *Nature biotechnology*, 26(10):1135–1145. 62, 65, 70, 95

Shendure, J., Mitra, R. D., Varma, C., and Church, G. M. (2004). Advanced sequencing technologies: methods and goals. *Nature Reviews Genetics*, 5(5):335–344. 62

Shiau, A. K., Katzenellenbogen, B. S., Barstad, D., Agard, D. A., Greene, G. L., Radek, J. T., Katzenellenbogen, J. A., Nettles, K. W., and Meyers, M. J. (2002). Structural characterization of a subtype-selective ligand reveals a novel mode of estrogen receptor antagonism. *Nature Structural and Molecular Biology*, 9(5):359. 112

Shindyalov, I. N. and Bourne, P. E. (1998). Protein structure alignment by incremental combinatorial extension (ce) of the optimal path. *Protein engineering*, 11(9):739–747. 116

Shipman, G. M., Woodall, T. S., Graham, R. L., Maccabe, A. B., and Bridges, P. G. (2006). Infiniband scalability in open mpi. In *Parallel and Distributed Processing Symposium, 2006. IPDPS 2006. 20th International*, pages 10–pp. IEEE. 32

Shvachko, K., Kuang, H., Radia, S., and Chansler, R. (2010). The hadoop distributed file system. In *Mass storage systems and technologies (MSST), 2010 IEEE 26th symposium on*, pages 1–10. IEEE. 44

Sikorsky, J. A., Primerano, D. A., Fenger, T. W., and Denvir, J. (2007). DNA damage reduces Taq DNA polymerase fidelity and PCR amplification efficiency. *Biochemical and biophysical research communications*, 355(2):431–7. 82, 95

Siva, N. (2015). Uk gears up to decode 100 000 genomes from nhs patients. *The Lancet*, 385(9963):103–104. 18

Sjöblom, T., Jones, S., Wood, L. D., Parsons, D. W., Lin, J., Barber, T. D., Mandelker, D., Leary, R. J., Ptak, J., Silliman, N., et al. (2006). The consensus coding sequences of human breast and colorectal cancers. *science*, 314(5797):268–274. 87

Smith, A. D., Xuan, Z., and Zhang, M. Q. (2008). Using quality scores and longer reads improves accuracy of solexa read mapping. *BMC bioinformatics*, 9(1):128. 50

Smith, D. J. and Burton, A. S. (2015). Sequencing to station in 12 months (targeting orbital 5 launch, march 30th). 20

Smith, J. E. and Nair, R. (2005). The architecture of virtual machines. *Computer*, 38(5):32–38. 31

Smith, L. M., Sanders, J. Z., Kaiser, R. J., Hughes, P., Dodd, C., Connell, C. R., Heiner, C., Kent, S., and Hood, L. E. (1985). Fluorescence detection in automated dna sequence analysis. *Nature*, 321(6071):674–679. 62

Snir, M. (1998). *MPI–the Complete Reference: The MPI core*, volume 1. MIT press. 40

Sodhi, J. S., Bryson, K., McGuffin, L. J., Ward, J. J., Wernisch, L., and Jones, D. T. (2004). Predicting metal-binding site residues in low-resolution structural models. *Journal of molecular biology*, 342(1):307–320. 123

Soneson, C. and Delorenzi, M. (2013). A comparison of methods for differential expression analysis of rna-seq data. *BMC bioinformatics*, 14(1):91. 91

Soper, S. A., Owens, C., Lassiter, S., Xu, Y., and Waddell, E. (2003). Dna sequencing using fluorescence detection. In *Topics in Fluorescence Spectroscopy*, pages 1–68. Springer. 62, 70

Sorefan, K., Pais, H., Hall, A. E., Kozomara, A., Griffiths-Jones, S., Moulton, V., and Dalmay, T. (2012). Reducing ligation bias of small rnas in libraries for next generation sequencing. *Silence*, 3(1):4. 87

Spitaleri, S., Piscitello, D., Di Martino, D., and Saravo, L. (2004). Experimental procedures comparing the activity of different Taq polymerases. *Forensic science international*, 146 Suppl:S167–9. 81, 95

SRA, N. (2010). The Sequence Read Arachive (SRA) Handbook (2010). https://www.ncbi.nlm.nih.gov/books/NBK47528/. [Online; accessed 10-October-2016]. 98

Staden, R. (1979). A strategy of dna sequencing employing computer programs. *Nucleic acids research*, 6(7):2601–2610. 62

Stein, L. D. (2010). The case for cloud computing in genome informatics. *Genome biology*, 11(5):207. 18, 24, 52

Stephens, Z. D., Lee, S. Y., Faghri, F., Campbell, R. H., Zhai, C., Efron, M. J., Iyer, R., Schatz, M. C., Sinha, S., and Robinson, G. E. (2015). Big data: Astronomical or genomical? *PLoS Biol*, 13(7):1–11. 18, 30, 43, 51, 54, 65, 93, 95, 157

Struhl, K. (2007). Transcriptional noise and the fidelity of initiation by rna polymerase ii. *Nature structural & molecular biology*, 14(2):103–105. 177

Stryer, L. (1998). *Biochemistry. 4th.* W.H. Freeman and Company. 16, 57

Subashini, S. and Kavitha, V. (2011). A survey on security issues in service delivery models of cloud computing. *Journal of network and computer applications*, 34(1):1–11. 54

Suzuki, S., Ono, N., Furusawa, C., Ying, B.-W., and Yomo, T. (2011). Comparison of sequence reads obtained from three next-generation sequencing platforms. *PloS one*, 6(5):e19534. 75, 76

Taverna PBS (2017). Taverna - Running on a PBS cluster. http://www.taverna.org.uk/2011/04/11/pbs-plugin-for-taverna-available/. [Online; accessed 26-June-2017]. 54

Taylor, R. C. (2010). An overview of the hadoop/mapreduce/hbase framework and its current applications in bioinformatics. *BMC bioinformatics*, 11(Suppl 12):S1. 50

Technologies, O. N. (2016). Oxford Nanopore Technologies. https://nanoporetech.com/products. [Online; accessed 27-October-2016]. 68

Temple, G., Gerhard, D. S., Rasooly, R., Feingold, E. A., Good, P. J., Robinson, C., Mandich, A., Derge, J. G., Lewis, J., Shoaf, D., et al. (2009). The completion of the mammalian gene collection (mgc). *Genome research*, 19(12):2324–2333. 90, 165

Thusoo, A., Sarma, J. S., Jain, N., Shao, Z., Chakka, P., Anthony, S., Liu, H., Wyckoff, P., and Murthy, R. (2009). Hive: a warehousing solution over a map-reduce framework. *Proceedings of the VLDB Endowment*, 2(2):1626–1629. 50

Trapnell, C., Pachter, L., and Salzberg, S. L. (2009). Tophat: discovering splice junctions with rna-seq. *Bioinformatics*, 25(9):1105–1111. 50, 66, 165

Trapnell, C., Roberts, A., Goff, L., Pertea, G., Kim, D., Kelley, D. R., Pimentel, H., Salzberg, S. L., Rinn, J. L., and Pachter, L. (2012). Differential gene and transcript expression analysis of rna-seq experiments with tophat and cufflinks. *Nature protocols*, 7(3):562–578. 90

Trapnell, C. and Salzberg, S. L. (2009). How to map billions of short reads onto genomes. *Nature biotechnology*, 27(5):455–457. 65, 66

Trapnell, C., Williams, B. A., Pertea, G., Mortazavi, A., Kwan, G., Van Baren, M. J., Salzberg, S. L., Wold, B. J., and Pachter, L. (2010). Transcript assembly and quantification by rna-seq reveals unannotated transcripts and isoform switching during cell differentiation. *Nature biotechnology*, 28(5):511–515. 67

Trevino, V., Falciani, F., and Barrera-Saldaña, H. A. (2007). Dna microarrays: a powerful genomic tool for biomedical and clinical research. *Molecular Medicine-Cambridge MA then New York-*, 13(9/10):527. 70, 92

Troger, P., Rajic, H., Haas, A., and Domagalski, P. (2007). Standardization of an api for distributed resource management systems. In *Cluster Computing and the Grid, 2007. CCGRID 2007. Seventh IEEE International Symposium on*, pages 619–626. IEEE. 54

Trott, O. and Olson, A. J. (2010). AutoDock Vina: improving the speed and accuracy of docking with a new scoring function, efficient optimization, and multithreading. *Journal of computational chemistry*, 31(2):455–61. 131, 150

Tucker, T., Marra, M., and Friedman, J. M. (2009). Massively Parallel Sequencing: The Next Big Thing in Genetic Medicine. *The American Journal of Human Genetics*, 85(2):142–154. 59, 61

Upton, G. J., Langdon, W. B., and Harrison, A. P. (2008). G-spots cause incorrect expression measurement in affymetrix microarrays. *BMC genomics*, 9(1):1. 155

Urban, J. M., Bliss, J., Lawrence, C. E., and Gerbi, S. A. (2015). Sequencing ultra-long dna molecules with the oxford nanopore minion. *bioRxiv*, page 019281. 67

Vavilapalli, V. K., Murthy, A. C., Douglas, C., Agarwal, S., Konar, M., Evans, R., Graves, T., Lowe, J., Shah, H., Seth, S., et al. (2013). Apache hadoop yarn: Yet another resource negotiator. In *Proceedings of the 4th annual Symposium on Cloud Computing*, page 5. ACM. 46

Velculescu, V. E., Madden, S. L., Zhang, L., Lash, A. E., Yu, J., Rago, C., Lal, A., Wang, C. J., Beaudry, G. A., Ciriello, K. M., et al. (1999). Analysis of human transcriptomes. *Nature genetics*, 23(4):387–388. 17

Venter, J. C., Adams, M. D., Myers, E. W., Li, P. W., Mural, R. J., Sutton, G. G., Smith, H. O., Yandell, M., Evans, C. A., Holt, R. A., et al. (2001). The sequence of the human genome. *science*, 291(5507):1304–1351. 17, 62

Wagener, J., Spjuth, O., Willighagen, E. L., and Wikberg, J. E. (2009). Xmpp for cloud computing in bioinformatics supporting discovery and invocation of asynchronous web services. *BMC bioinformatics*, 10(1):279. 54

Wagner, G. P., Kin, K., and Lynch, V. J. (2012). Measurement of mrna abundance using rna-seq data: Rpkm measure is inconsistent among samples. *Theory in Biosciences*, 131(4):281–285. 157

Wang, Y., Xiao, J., Suzek, T. O., Zhang, J., Wang, J., and Bryant, S. H. (2009a). Pubchem: a public information system for analyzing bioactivities of small molecules. *Nucleic acids research*, 37(suppl_2):W623–W633. 123

Wang, Y., Yang, Q., and Wang, Z. (2013). The evolution of nanopore sequencing. *Frontiers in genetics*, 5:449–449. 67

Wang, Z., Gerstein, M., and Snyder, M. (2009b). Rna-seq: a revolutionary tool for transcriptomics. *Nature reviews genetics*, 10(1):57–63. 65, 67

Ward, R. M., Schmieder, R., Highnam, G., and Mittelman, D. (2013). Big data challenges and opportunities in high-throughput sequencing. *Systems Biomedicine*, 1(1):29–34. 18, 43, 51, 54, 65, 93

Weber, J. L. and Myers, E. W. (1997). Human whole-genome shotgun sequencing. *Genome Research*, 7(5):401–409. 61, 62

Wiewiórka, M. S., Messina, A., Pacholewska, A., Maffioletti, S., Gawrysiak, P., and Okoniewski, M. J. (2014). Sparkseq: fast, scalable, cloud-ready tool for the interactive genomic data analysis with nucleotide precision. *Bioinformatics*, page btu343. 50

Wikimedia Commons (2017). Structure of a nucleotide. https://commons.wikimedia.org/wiki/File:Nucleotides_1.svg. [Online; accessed 23-May-2017]. 57

Wolstencroft, K., Haines, R., Fellows, D., Williams, A., Withers, D., Owen, S., Soiland-Reyes, S., Dunlop, I., Nenadic, A., Fisher, P., et al. (2013). The taverna workflow suite: designing and executing workflows of web services on the desktop, web or in the cloud. *Nucleic acids research*, page gkt328. 53

World Health Organisation (2017). WHO, International Classification of Disease- ICD-11 Revision. http://hadoop.apache.org/docs/r2.7.2/. [Online; accessed 21-Dec-2017]. 108

Wu, Z., Wang, X., and Zhang, X. (2011). Using non-uniform read distribution models to improve isoform expression inference in rna-seq. *Bioinformatics*, 27(4):502–508. 90

Xie, L. and Bourne, P. E. (2007). A robust and efficient algorithm for the shape description of protein structures and its application in predicting ligand binding sites. *BMC bioinformatics*, 8(4):S9. 112

Ye, Y. and Godzik, A. (2003). Flexible structure alignment by chaining aligned fragment pairs allowing twists. *Bioinformatics*, 19(suppl_2):ii246–ii255. 116

Younge, A. J., Henschel, R., Brown, J. T., Von Laszewski, G., Qiu, J., and Fox, G. C. (2011). Analysis of virtualization technologies for high performance computing environments. In *Cloud Computing (CLOUD), 2011 IEEE International Conference on*, pages 9–16. IEEE. 31

Zaharia, M., Chowdhury, M., Das, T., Dave, A., Ma, J., McCauley, M., Franklin, M. J., Shenker, S., and Stoica, I. (2012). Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation*, pages 2–2. USENIX Association. 49

Zhang, X., Wong, S. E., and Lightstone, F. C. (2013). Message passing interface and multithreading hybrid for parallel molecular docking of large databases on petascale high performance computing machines. *Journal of computational chemistry*, 34(11):915–927. 112, 118, 124

Zhang, Z., Theurkauf, W. E., Weng, Z., and Zamore, P. D. (2012). Strand-specific libraries for high throughput rna sequencing (rna-seq) prepared without poly (a) selection. *Silence*, 3(1):9. 88, 218, 222

Zhao, W. and Stankovic, J. A. (1989). Performance analysis of fcfs and improved fcfs scheduling algorithms for dynamic real-time computer systems. In *Real Time Systems Symposium, 1989., Proceedings.*, pages 156–165. IEEE. 34

Zheng, W., Chung, L. M., and Zhao, H. (2011). Bias detection and correction in rna-sequencing data. *BMC bioinformatics*, 12(1):1. 91, 155, 156

Zhou, A. Q., O'hern, C. S., and Regan, L. (2011). Revisiting the ramachandran plot from a new angle. *Protein Science*, 20(7):1166–1171. 132

Zhuang, F., Fuchs, R. T., Sun, Z., Zheng, Y., and Robb, G. B. (2012). Structural bias in t4 rna ligase-mediated 3-adapter ligation. *Nucleic acids research*, 40(7):e54–e54. 87