Cloud based computing technologies for genomic medicine

Andrian Yang

A thesis in fulfillment of the requirements for the degree of
Doctor of Philosophy

Molecular, Structural and Computational Division
Victor Chang Cardiac Research Institute

and

St Vincent's Clinical School
Faculty of Medicine
The University of New South Wales

February 2019
Abstract

Recent advances in single-cell RNA-sequencing (scRNA-seq) methods have enabled the study of cellular heterogeneity at the single-cell resolution. However, current tools for processing and analysing RNA-seq data are not equipped to handle the large amount of data generated in single-cell studies. With the exponential growth in the number gene expression profiles generated by scRNA-seq methods, there is a need to develop scalable tools for large-scale data analysis and interpretation. In this thesis, I report several new scalable bioinformatics methods that I have developed for the analysis of scRNA-seq data:

1. Falco - a new cloud-based framework for processing of large-scale scRNA-seq data. Falco utilises standard Big Data frameworks such as Apache Hadoop and Apache Spark to enable scalable data analysis. The Falco framework is designed to perform read processing, alignment, gene expression quantification, and transcript reconstruction - all in a parallel and distributed manner. We demonstrated Falco's scalability using real data sets, with Falco achieving a speed up of 1.7x to 145x compared to single-node execution. Falco also allows for cost efficient analysis, providing savings of up to 65%.

2. Scavenger - a new pipeline to recover false negative, non-aligned reads in RNA-seq data. Scavenger utilises a novel mechanism for the recovery of such reads based on similarity with aligned reads. Using real data, we demonstrated how Scavenger is able to recover a good portion of non-aligned reads and how reads recovered have more variance compared to aligned reads. Genes with substantial increase in expression after recovery are typically lowly-expressed genes and are enriched for pseudogenes, suggesting that the expression of pseudogenes may be under-reported.

3. Starmap - a new tool for visualisation of scRNA-seq data to help with the exploration and interpretation of the large amount of data. Starmap combines two visual paradigms, the 3D scatter plot and the star plot, to allow visualisation of both the high level structure of the data and the cell-level features. Starmap is designed to be cross-platform and supports an immersive mode which allows for visualisation using low-cost VR headsets.

Declaration relating to disposition of project thesis/dissertation

I hereby grant to the University of New South Wales or its agents the right to archive and to make available my thesis or dissertation in whole or in part in the University libraries in all forms of media, now or hereafter known, subject to the provisions of the Copyright Act 1968. I retain all property rights, such as patent rights. I also retain the right to use in future works (such as articles or books) all or part of this thesis or dissertation.

I also authorise University Microfilms to use the 350 word abstract of my thesis in Dissertation Abstracts International (this is applicable to doctoral theses only).

Signature Witness Date
Originality Statement

‘I hereby declare that this submission is my own work and to the best of my knowledge it contains no materials previously published or written by another person, or substantial proportions of material which have been accepted for the award of any other degree or diploma at UNSW or any other educational institution, except where due acknowledgement is made in the thesis. Any contribution made to the research by others, with whom I have worked at UNSW or elsewhere, is explicitly acknowledged in the thesis. I also declare that the intellectual content of this thesis is the product of my own work, except to the extent that assistance from others in the project’s design and conception or in style, presentation and linguistic expression is acknowledged.’

Andrian Yang
November 20, 2018
Copyright Statement

‘I hereby grant the University of New South Wales or its agents the right to archive and to make available my thesis or dissertation in whole or part in the University libraries in all forms of media, now or here after known, subject to the provisions of the Copyright Act 1968. I retain all proprietary rights, such as patent rights. I also retain the right to use in future works (such as articles or books) all or part of this thesis or dissertation.

I also authorise University Microfilms to use the 350 word abstract of my thesis in Dissertation Abstract International (this is applicable to doctoral theses only).

I have either used no substantial portions of copyright material in my thesis or I have obtained permission to use copyright material; where permission has not been granted I have applied/will apply for a partial restriction of the digital copy of my thesis or dissertation.’

Andrian Yang  
November 20, 2018

Authenticity Statement

‘I certify that the Library deposit digital copy is a direct equivalent of the final officially approved version of my thesis. No emendation of content has occurred and if there are any minor variations in formatting, they are the result of the conversion to digital format.’

Andrian Yang  
November 20, 2018
UNSW is supportive of candidates publishing their research results during their candidature as detailed in the UNSW Thesis Examination Procedure.

Publications can be used in their thesis in lieu of a Chapter if:
- The student contributed greater than 50% of the content in the publication and is the “primary author”, i.e. the student was responsible primarily for the planning, execution and preparation of the work for publication
- The student has approval to include the publication in their thesis in lieu of a Chapter from their supervisor and Postgraduate Coordinator.
- The publication is not subject to any obligations or contractual agreements with a third party that would constrain its inclusion in the thesis

Please indicate whether this thesis contains published material or not.

☐ This thesis contains no publications, either published or submitted for publication

☒ Some of the work described in this thesis has been published and it has been documented in the relevant Chapters with acknowledgement

☐ This thesis has publications (either published or submitted for publication) incorporated into it in lieu of a chapter and the details are presented below

CANDIDATE’S DECLARATION
I declare that:
- I have complied with the Thesis Examination Procedure
- where I have used a publication in lieu of a Chapter, the listed publication(s) below meet(s) the requirements to be included in the thesis.

<table>
<thead>
<tr>
<th>Name</th>
<th>Signature</th>
<th>Date (dd/mm/yy)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Andrian Yang</td>
<td></td>
<td>20/11/18</td>
</tr>
</tbody>
</table>
Abstract

Recent advances in single-cell RNA-sequencing (scRNA-seq) methods have enabled the study of cellular heterogeneity at the single-cell resolution. However, current tools for processing and analysing RNA-seq data are not equipped to handle the large amount of data generated in single-cell studies. With the exponential growth in the number gene expression profiles generated by scRNA-seq methods, there is a need to develop scalable tools for large-scale data analysis and interpretation. In this thesis, I report several new scalable bioinformatics methods that I have developed for the analysis of scRNA-seq data:

1. Falco - a new cloud-based framework for processing of large-scale scRNA-seq data. Falco utilises standard Big Data frameworks such as Apache Hadoop and Apache Spark to enable scalable data analysis. The Falco framework is designed to perform read processing, alignment, gene expression quantification, and transcript reconstruction - all in a parallel and distributed manner. We demonstrated Falco’s scalability using real data sets, with Falco achieving a speed up of 1.7x to 145x compared to single-node execution. Falco also allows for cost efficient analysis, providing savings of up to 65%.

2. Scavenger - a new pipeline to recover false negative, non-aligned reads in RNA-seq data. Scavenger utilises a novel mechanism for the recovery of such reads based on similarity with aligned reads. Using real data, we demonstrated how Scavenger is able to recover a good portion of non-aligned reads and how reads recovered have more variance compared to aligned reads. Genes with substantial increase in expression after recovery are typically lowly-expressed genes and are enriched for pseudogenes, suggesting that the expression of pseudogenes may be under-reported.

3. Starmap - a new tool for visualisation of scRNA-seq data to help with the exploration and interpretation of the large amount of data. Starmap combines two visual paradigms, the 3D scatter plot and the star plot, to allow visualisation of both the high level structure of the data and the cell-level features. Starmap is designed to be cross-platform and supports an immersive mode which allows for visualisation using low-cost VR headsets.
Preface

All research conducted during my PhD candidature is multidisciplinary in nature and involved collaboration with my supervisors, researchers, and students. As a result, I have contributed to a number of work that have been published, or submitted for publication, in international peer-reviewed journals. This thesis is a summary of my own work and therefore only includes materials in which I have made the most significant contribution in.

List of publications


List of submitted publications


List of oral presentations at international conferences


2. Yang A, Troup M, Ho JWK. A quick and flexible transcriptomic feature quantification framework on the cloud. Presented at the European Student Council Symposium 2016, Netherlands
List of poster presentations at international conferences


Acknowledgements

I would first like to thank my supervisor, Dr Joshua Ho, for his help and guidance throughout the course of my PhD. He has been a great mentor, providing encouragement and support whenever needed. I have learnt so much from him over this past few years and am very grateful for being his student.

I would like to thank my co-supervisor, Dr Catherine Suter, for her advice and help throughout my PhD. I would also like to thank both Dr Eleni Giannoulatou and Dr Nicola Smith for their mentorship and support especially during the later stages of my PhD.

I would like to thank both current and past members of the Ho and Giannoulatou laboratories from the Victor Chang Cardiac Research Institute: David Humphreys, Michael Troup, Tomasz Szczesnik, Paul Lin, Xin Wang, Djordje Djordjevic, Joshua Tang, Eddie Ip, Ralph Patrick along with many others not listed here, in no particular order. Thank you all for the thought-provoking discussions, assistance and fun we have had over these past few years.

I would also like to thank the other members of the Victor Chang Cardiac Research Institute, from members of my review panel for their advice and encouragement, through to members of the IT team, particularly Adam Phipps and Andrew Hall, for the discussions we have had over the years and their help.

I would like to thank my friends for their support and encouragement. Last but not least, I would like to thank my parents, especially my mum, and my siblings for their unwavering support and belief in me. This thesis would not have been possible without them.
Abbreviations

**ABySS** Assembly By Short Sequences

**API** Application programming interface

**ARCHS4** All RNA-seq and ChIP-seq sample and signature search

**AWS** Amazon Web Services

**BAM** Binary Alignment Map

**BASIC** BCR assembly from single cells

**BEERS** Benchmarker for Evaluating the Effectiveness of RNA-Seq Software

**BLAST** Basic Local Alignment Search Tool

**BS-seq** Bisulfite sequencing

**BWA** Burrows-Wheeler Aligner

**ChIP** Chromatin immunoprecipitation

**ChIP-Seq** ChIP sequencing

**CloVR** Cloud Virtual Resource

**CPU** Central processing unit

**DaaS** Data as a service

**DNA** Deoxyribonucleic acid

**DNA-seq** DNA sequencing

**EC2** Elastic Compute Cloud

**EGI** European Grid Infrastructure
EMR Elastic MapReduce
ENA European Nucleotide Archive
ENCODE Encyclopedia of DNA Elements
ESC Embryonic stem cell
FDA Food and Drug Administration
FPGA Field-programmable gate array
FPS Frame per seconds
GATK Genome Analysis Toolkit
GB Gigabyte
GPGPU General-purpose computing on GPU
GPU Graphics processing unit
GSNAP Genomic Short-read Nucleotide Alignment Program
GTF Gene transfer format
HDFS Hadoop Distributed File System
HISAT Hierarchical indexing for spliced alignment of transcripts
IaaS Infrastructure as a service
IGV Integrative Genomics Viewer
IT Information technology
MACS Model-based Analysis of ChIP-Seq
MB Megabyte
mESC Mouse ESC
MGP Mouse Genome Project
MIT Massachusetts Institute of Technology
MPI Message Passing Interface
MR Metamorphic relation
MRI  Magnetic resonance imaging
mRNA  Messenger RNA
MT  Metamorphic testing
NCBI  National Center for Biotechnology Information
NCL  Non-collinear
NGS  Next generation sequencing
NIST  National Institute of Standards and Technology
OpenGL  Open Graphics Library
PaaS  Platform as a service
PBMC  Peripheral blood mononuclear cell
PCA  Principal component analysis
PET  Positron-emission tomography
PRINSEQ  Preprocessing and Information of Sequences
PVM  Parallel Virtual Machine
QA  Quality assurance
RAM  Random-access memory
RAxML  Randomized Axelerated Maximum Likelihood
RNA  Ribonucleic acid
RNA-seq  RNA sequencing
ROP  Read Origin Protocol
RPM  Reads per million
rRNA  Ribosomal RNA
SaaS  Software as a service
SAM  Sequence Alignment Map
scRNA-seq  Single-cell RNA sequencing
SIMD  Single instruction multiple data
SMP  Symmetric multiprocessing
SMRT  Single molecule real time
SNP  Single nucleotide polymorphisms
SOLiD  Sequencing by Oligonucleotide Ligation and Detection
SRA  Sequence Read Archive
SSD  Solid State Drives
STAR  Spliced Transcripts Alignment to a Reference
SW  Smith–Waterman
t-SNE  t-Distributed Stochastic Neighbor Embedding
TB  Terabyte
UCSC  University of California Santa Cruz
UMI  Unique molecular identifier
USD  United States Dollar
VCAT  Variant Calling Assessment Tool
VCF  Variant Call Format
VM  Virtual machine
VR  Virtual reality
WGS  Whole genome sequencing
YARN  Yet Another Resource Negotiator
Contents

1 Introduction 1

1.1 Evolution of sequencing technologies ................. 2

1.1.1 Early Sequencing Technologies ...................... 2

1.1.2 Next Generation Sequencing Technologies ............ 3

1.1.3 Beyond Next Generation Sequencing Technologies .... 5

1.2 Application of sequencing technologies .................. 6

1.2.1 Genome Analysis .................................... 6

1.2.2 Transcriptome Analysis .............................. 7

1.3 Bioinformatics analysis of sequencing data ............... 8

1.3.1 Scaling up bioinformatics analysis of sequencing data .. 10

1.4 Problem statement and thesis aims ....................... 12

2 Scalability of big data bioinformatics 14

2.1 Overview ............................................. 16
5.3.5 Parallelising Scavenger ............................................. 86
5.3.6 Datasets ............................................................... 89
5.4 Results ................................................................. 90
  5.4.1 Recovery of reads on simulated data ......................... 90
  5.4.2 Divergence of personal genome results in false-negative non-aligned reads 95
  5.4.3 Effect of Scavenger recovery pipeline on downstream analysis .... 101
  5.4.4 Applying Scavenger recovery on single-cell RNA-seq data ........ 105
5.5 Discussion ............................................................ 108

6 starmap: Immersive visualisation of single cell data using smartphone-enabled virtual reality 112
  6.1 Overview .......................................................... 114
  6.2 Motivation ......................................................... 114
  6.3 Introduction ..................................................... 115
  6.4 Implementation .................................................. 116
  6.5 Performance evaluation ........................................ 119
  6.6 Conclusion ........................................................ 120
  6.7 Materials and methods ........................................ 123
    6.7.1 Input data format for starmap ............................ 123
    6.7.2 Processing of sample datasets ............................ 123
# Discussion

7.1 Towards a scalable analysis tool for large scale sequencing data . . . . . . 126

7.2 Towards a more sensitive alignment tool . . . . . . . . . . . . . . . . . . . 128

7.3 Towards a scalable visualisation tool for large scale data . . . . . . . . . . 130

7.4 Concluding remarks . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 131
List of Figures

2.1 Examples of MapReduce-based (a) and Spark-based (b) big data bioinformatics analysis frameworks. .............................................................. 26

3.1 The Falco framework process pipeline. In the splitting step, reads from one or multiple FASTQ files are split into multiple chunks of size 256 Mb uncompressed. This step makes use of Apache Hadoop MapReduce. A pre-processing step is executed if the data require pre-processing using MapReduce. In the main analysis step, sequence alignment and gene expression quantification are carried out in a highly parallelised fashion using the Apache Spark framework. ................................................................. 33

3.2 Falco processing time split by steps for STAR+featureCounts (a) and HISAT2+HTSeq (b) pipelines for mouse embryonic stem cell and human brain data analysis. ................................................................. 37

3.3 Size of the FASTQ file (in terms of gigabytes) of individual cells in each single-cell RNA-seq data set. (a) Mouse embryonic stem cells (SRA accession: ERP005988). (b) Human brain cells (SRA accession: SRP057196). ................................................................. 38

4.1 The Falco 2.0 framework alignment-only pipeline. The pipeline is composed of the splitting and pre-processing steps from the original Falco framework and the new Spark-based alignment step from the Falco 2.0 framework. The alignment step is composed of two stages - an alignment stage, where read chunks are aligned and stored in a temporary location in HDFS, and a concatenation stage, where alignment chunks from the same sample are concatenated to obtain the full alignment result. ................................................................. 50
4.2 The Falco 2.0 framework transcript assembly pipeline. The pipeline is also composed of the splitting and pre-processing steps from the original Falco framework in addition to the new Spark-based transcript assembly step from the Falco 2.0 framework. The transcript assembly step is composed of a number of stages, including an alignment stage, which performs alignment of read chunks and binning of the alignment result; an assembly stage which performs transcript assembly in parallel, and a merging step, where assembled transcripts are merged with the reference annotation to produce an updated annotation.

4.3 Falco 2.0 alignment-only processing time split by steps and stages for STAR and HISAT2 pipelines in the analysis of mouse embryonic stem cell and human brain single cell data. The timings shown do not include cluster initialisation time, as it is constant across differing cluster sizes.

4.4 Example of novel transcripts found within the CREG2 gene in the human brain single cell data. The two novel transcripts (highlighted in yellow) are supported by the read coverage data, which is obtained from combining the alignment records of the 100 randomly selected samples from the human brain data.

4.5 Falco 2.0 transcript assembly processing time split by steps and stages for STAR + StringTie with reference, STAR + Scallop, and HISAT2 + StringTie pipelines in the analysis of human brain single cell data. The timings shown do not include cluster initialisation time, as it is constant across differing cluster sizes, or later stages of the transcript assembly step, as the total time taken for the remaining steps is < 30 seconds.

4.6 Distribution of read bin sizes for (A) human brain and (B) mouse embryonic stem cell single cell datasets.
5.1 Overview of the Scavenger pipeline. Scavenger first aligns sequencing reads against the reference genome using the STAR alignment tool in the source execution step. Scavenger then extracts both the aligned and unaligned reads from the source alignment result and creates a sequencing reads file based on the aligned reads and an artificial genome file containing chromosomes built using sequences of unaligned reads. The sequences of aligned reads are then aligned to the artificial genomes using the same alignment tool from the source execution (STAR) in the follow-up execution steps to find aligned reads which have similar sequences to unaligned reads. Next, consensus filtering is performed to select putative sites for re-alignment based on where the majority of aligned reads originate from in the reference genome. Finally, re-alignment is performed for unaligned reads which pass consensus filtering and the source alignment result is updated based on the result of re-alignment.

5.2 The number of reads containing SNPs found within reads aligned in source execution and reads recovered by Scavenger. A. The number of reads with ≥1 SNPs found within reads. B. The number of reads with high number of SNPs (>5) found within reads.

5.3 Distribution of number of SNPs found within reads recovered during source alignment and reads recovered by Scavenger. Scavenger is able to recover reads with a higher number of SNPs (>5) compared to source alignment.

5.4 Gene expression in source alignment and after Scavenger recovery for genes whose reads are recovered. Coloured points indicate genes with expression difference of greater than 1 fold change.

5.5 Effect of Scavenger read recovery on gene expression for normal (bulk) RNA-seq datasets. A. The number of genes whose reads are recovered by Scavenger, categorised based on the fold change in normalised expression (RPM) between source alignment and after Scavenger recovery. B. The number of genes with more than 1 fold change in normalised expression categorised based on their gene types.

5.6 Genome browser view of alignment in source execution and after recovery with Scavenger for FTH1P5 pseudogene from one human bulk RNA-seq dataset (SRR6337341). FTH1P5 gene was chosen as it is the only pseudogene consistently recovered in at least half of the human bulk RNA-seq dataset.
5.7 The number of reads recovered by Scavenger categorised by gene type for A. normal (bulk) RNA-seq datasets and B. single-cell RNA-seq datasets. In general, most reads are located in a region without a feature or within a protein coding gene. However, a high percentage of reads in human bulk RNA-seq datasets are located in other gene types, more specifically mitochondrial genes, due to the high source expression of these genes.

5.8 Effect of Scavenger read recovery on gene expression for single-cell RNA-seq datasets. A. The number of genes whose reads are recovered by Scavenger, categorised based on the fold change in normalised expression (RPM) between source alignment and after Scavenger recovery. B. The number of genes with more than 1 fold change in normalised expression categorised based on their gene types.

6.1 starmap visualisation of a single-cell RNA-seq data containing 68,000 peripheral blood mononuclear cells (a) Overview of the 3D scatter plot with visualised in non-VR mode (b) Star plot visualisation showing the relative expression of gene modules for each cell in VR mode.

6.2 Control mapping for remote control in starmap.

6.3 Loading and processing time of starmap with varying dataset sizes on mobile devices.

6.4 Loading and processing time of starmap with varying dataset sizes on desktop devices.
# List of Tables

3.1 scRNA-seq processing time with or without Falco ........................................ 35

3.2 Falco cost analysis: on-demand vs. spot instances for STAR + featureCounts .......... 36

3.3 Falco cost analysis: on-demand vs. spot instances for HISAT2 + HTSeq ................. 39

4.1 Comparison of runtime for alignment of single cell datasets with and without
the Falco 2.0 framework ................................................................................................. 59

4.2 Comparison of runtime for alignment of the human brain single cell dataset
using Rail-RNA and Falco 2.0 frameworks ..................................................................... 61

4.3 Accuracy of assembled transcripts for simulated data from single node runs ............. 64

4.4 Accuracy of assembled transcripts for simulated data from Falco 2.0-based runs ....... 64

4.5 Precision of assembled transcripts for human brain single cell dataset across
different transcript assembly approaches ........................................................................ 66

4.6 Precision of assembled transcripts for mouse embryonic stem cell single cell
dataset across different transcript assembly approaches .............................................. 66

4.7 Comparison of runtime for transcript assembly of single cell dataset with and
without the Falco 2.0 framework .................................................................................... 69
5.1 List of datasets used for Scavenger testing and evaluation. The datasets are
divided into three sections: 1. Datasets from selected non-reference mouse
strain, 2. Normal (bulk) RNA-seq dataset from either human or mouse, and
3. Single-cell RNA-seq dataset from mouse. 88

5.2 Alignment statistics for simulated datasets before and after recovery of reads
with Scavenger using default parameters for STAR. 92

5.3 Alignment statistics for simulated datasets before and after recovery of reads
with Scavenger using optimised parameters for STAR. 92

5.4 Unaligned reads identified by ROP in the simulated dataset. 94

5.5 Alignment statistic for unaligned reads recovered by ROP lost read identifi-
cation step. 94

5.6 Alignment statistics for all RNA-seq datasets in source alignment with STAR
and after recovery of reads with Scavenger. 99

5.7 Alignment statistics for simulated datasets before and after recovery of reads
with Scavenger using default parameters for Subread. 111

5.8 Alignment statistics for simulated datasets before and after recovery of reads
with Scavenger using optimised parameters for Subread. 111

6.1 Keyboard control scheme and voice control commands for starmap. 118

6.2 Rendering performance test on mobile devices with varying dataset sizes.
Checkmarks indicates that the mobile device is able to render the dataset
with a minimum of 30 frames per seconds. 122

6.3 Rendering performance test on mobile devices with varying dataset sizes.
Checkmarks indicates that the mobile device is able to render the dataset
with a minimum of 30 frames per seconds. 122
Chapter 1

Introduction
1. Introduction

Since the release of the human genome sequence in 2001 [1], there has been increasing use of sequencing to solve biological problems owing to major technological advancements which has made sequencing more accessible and affordable. With the realisation of the $1000 genome and recent technological advancements in single-cell capture technology, it is expected that the use of sequencing for biological analysis will become even more widespread.

However, the generation of sequencing data is only half of the story. In order to utilise sequencing technology to solve biological problem, we need to turn to computers to extract useful information from the data generated. Bioinformatics software designed for sequence analysis has evolved alongside sequencing technology in order to perform new and innovative analysis in more efficient and scalable ways. With the projected rapid increase in sequencing data generation, there is a pressing need to design bioinformatics software that can scale to handle the large volume of sequencing data in an efficient manner.

1.1 Evolution of sequencing technologies

1.1.1 Early Sequencing Technologies

The first generation of DNA sequencing technologies are represented by the Sanger sequencing method [2] and Maxam-Gilbert method developed in the 1970s [3]. The Sanger sequencing method involves the use of chain-terminating nucleotide analogs which produces DNA molecules of differing length during synthesis of the complementary strand. The Maxam-Gilbert sequencing method, on the other hand, involves cleaving of DNA molecules at differing nucleotides which yield fragments of differing sizes. After the generation of DNA library of varying lengths, the nucleotide sequence was resolved using gel electrophoresis to separate the DNA molecules based on their length.
1. Introduction

The Maxam-Gilbert method was originally preferred over the Sanger method as it allows for direct use of the purified DNA, unlike the Sanger method which require synthesis of DNA using DNA polymerase. However, the Sanger sequencing method eventually become the dominant method due to a number of innovative developments which enable the semi-automation of the sequencing process. One of the major developments was the replacement of the radio-labelled nucleotide analogs with fluorescent-based nucleotide analogs based on the work by Hood et al in 1986 [4]. The switch to fluorescent-based detection of nucleotide allows for continuous and automated reading of fluorescence signal by computer, as well as a reduction in the number of reactions from four in the original protocol (for each of the nucleotide bases) to just one.

The first automated sequencing machine based on the Sanger sequencing method was released in 1987 by Applied Biosystems, with Sanger-based sequencing machines being crucial for the completion of the Human Genome Project. However, with the development of the next generation sequencing technology, the use of Sanger sequencing has largely been reduced, though it is still used today due to the ability for Sanger sequencing method to sequence reads up to 1 kilobases in length.

1.1.2 Next Generation Sequencing Technologies

Current advances in genomics have been largely driven by the development of the so-called next generation sequencing (NGS) technology, which is mainly characterised by the massively parallel sequencing capability. The high throughput of NGS was achieved through the creation of many millions of individual spatially-separated reaction centres, each containing a clonally amplified DNA template, and direct detection of sequencing output without the use of electrophoresis. The simultaneous detection of information from all reaction centres allow for the NGS platforms to sequence many millions of DNA molecules in parallel. There
1. Introduction

has been a number of sequencing methods developed for NGS which varies in the approach used for sequencing and detection. However, these approaches share similar limitations of shorter length of reads (< 1 kilobases) that can be sequenced and the need for amplification of template DNA.

An early realisation of NGS technology is the development of pyrosequencing. Pyrosequencing is a real-time based sequencing approach which uses the release of pyrophosphate as a result of nucleotide integration during DNA synthesis to trigger the emission of light [5]. To detect and determine the nucleotide that produces the light emission, pyrosequencing requires iterative addition of each different nucleotides in a cyclical manner. This sequencing method was utilised by the Roche/454 platform and was able to originally produce reads ranging from 400-500 base pairs [6].

The next NGS technology developed was the sequencing-by-synthesis approach which shares a similar approach to the Sanger sequencing method with the use of terminating fluorescent-labelled nucleotides. However, the termination of synthesis used in this approach is able to be reversed through cleavage of the fluorophores after the integration and imaging steps, therefore allowing for the continuation of the synthesis step [7]. This approach was first utilised by Solexa, which was incorporated into Illumina sequencing, in their Genome Analyser machine. This machine was originally only able to produce very short reads of 35 base pair lengths, though they have the advantage of being able to produce reads from both ends of the DNA template. Later evolution of the sequencing machine produced by Illumina has both increased the throughput and length of reads that can be sequenced, leading to the release of HiSeq 10X machine which fulfilled the $1000 genome promise.

Sequencing by Oligonucleotide Ligation and Detection (SOLiD) is an NGS technology which was developed by Applied Biosystems [8]. As the name implies, SOLiD sequencing method works by ligation of fluorescently labelled probes whose first two sequences are known. After imaging and detection, the probe is partially cleaved to allow for further ligation of another
1. Introduction

fluorescently labelled probe. The process is repeated multiple times, using a starting primer of differing offsets to obtain the full sequence from the DNA template. A similar ligation-based approach is also used by Complete Genomics’ combinatorial probe-anchor ligation method where fluorescently labelled probe with a known base in a constant position is ligated to an anchor [9]. The probe-anchor ligation step is repeated multiple times, with new probe that has a known base in a new position, in order to obtain the full sequence.

Another NGS technique is the Ion Torrent sequencing technology developed by Life Technologies [10]. The Ion Torrent sequencing method is similar in concept to pyrosequencing, in that it is a real-time based sequencing approach which utilises the incremental addition of nucleotides. However, the Ion Torrent technology is the first technology which does not utilise optical based detection and instead use changes in pH caused by the release of hydrogen ions during nucleotide integration for detection.

1.1.3 Beyond Next Generation Sequencing Technologies

As previously mentioned, one of the major limitations in the current NGS technology is the short length of reads produced, which reduce the ability to resolve complex regions in genome as a result of repetitive elements, copy number variations and structural variation. To help resolve these limitations, a number of sequencing methods have been developed which are able to produce long sequencing reads, up to a maximum read length of 100 kilobases [11]. There are currently two major types of methods for long-read sequencing - the first and most popular method is the single molecule real time (SMRT) sequencing method utilised by PacBio sequencing [12] and Oxford Nanopore sequencing [13]; and the second method is the ‘synthetic’ based approach, which reconstruct long read sequence in silico from barcoded short read sequences [14].

The PacBio sequencing method is based on the previous fluorescent labelled nucleotide se-
1. Introduction

Sequencing approach, with the main difference being the real-time detection of nucleotide incorporation instead of the cycle-based incorporation and detection. This method works by the continuous detection of light emission from reaction centres containing embedded DNA polymerase that will incorporate the fluorescent labelled nucleotide during DNA synthesis. The Oxford Nanopore sequencing method utilises a completely different approach where sequencing is performed by passing DNA molecules through a nanometer-sized hole of a transmembrane protein. The passage of DNA molecules causes a change in current as different nucleotides pass through the hole, allowing for the detection of the DNA sequence in real time.

1.2 Application of sequencing technologies

1.2.1 Genome Analysis

Early application of sequencing technology was primarily focused on decoding the genome of human and other model organisms, such as mouse [15], fly [16], rice [17] and fowl [18]. This was largely due to the cost to perform sequencing and the limited accessibility of sequencing technology for researchers outside of genome centres. The development of NGS technology has driven down the cost of sequencing thus making the technology much more accessible. This has resulted in an increase in the use of genome sequencing to study the underlying genetic factor behind diseases. The reduced cost of sequencing has also enabled the study of whole genome sequencing at population level, as evidenced by the completion of the 1000 genome project which is the first large scale study of genetic variation in human population [19]. Recently, there have been efforts undertaken to integrate whole genome sequencing for clinical diagnostics, such as the MedSeq project [20].

Sequencing technology has also been used to provide insight into genomic regulation. Chro-
1. Introduction

Chromatin immunoprecipitation (ChIP) sequencing was developed for the identification of in-vivo protein-DNA interaction [21] and has enabled the study of regulatory elements of the genome, including transcription factors binding and histone marks modification. On the other hand, chromosome conformation capture technologies, such as 3C [22], 4C [23], 5C [24] and Hi-C [25], are designed for detection of DNA-DNA interaction within and between chromosomes for the study of chromatin spatial structure and transcriptional regulation mechanism. Another type of genetic regulation which can be studied with sequencing technology is methylation of DNA sequence using bisulfite sequencing [26]. Methylation of the nucleotide Cytosine in CpG islands is a type of heritable genetic modification which does not involve a change in DNA sequence and has been associated with transcriptional silencing [27].

1.2.2 Transcriptome Analysis

Aside from genome analysis, another major application of sequencing technology is for transcriptome analysis. RNA sequencing (RNA-seq) is accomplished by sequencing complementary DNA libraries generated from captured RNA molecules. The use of RNA-seq for transcriptome analysis provides a number advantages compared to the previous microarray-based approaches, such as the higher accuracy of transcript abundance estimation, particularly for lowly expressed genes, and the ability to detect novel genes and transcripts isoforms, as RNA-seq does not require information regarding genome sequence of the organism. The primary usage of RNA-seq is for gene expression analysis, which is inferred through quantification of messenger RNA molecules. However, RNA-seq also provides functional genomic information, thereby allowing for detection of allele-specific gene expression, alternative splicing, gene-fusion events and RNA-editing events.

Normal RNA-seq methods provides averaged expression profiles from a population of cells,
therefore masking both the transcriptional heterogeneity of cell sub-populations and co-expression patterns between different cells. However, recent advancements in single cell capture technology has enabled the study of transcriptome at the cellular level with the development of single-cell RNA-sequencing (scRNA-seq). Since the inception of scRNA-seq method, there has been an exponential growth in the number of single cells studied in scRNA-seq, starting with a single mouse blastomere in 2009 \cite{28} up to 1.3 million mouse brain cells in 2017 \cite{29}. Many studies have utilised scRNA-seq to uncover transcriptional heterogeneity of individual cells, as shown by the discovery of novel cell-type subpopulations within stem cells \cite{30}, and to provide new insights into the mechanisms of cell-state transitions, such as the trajectory analysis of human pluripotent stem cell differentiation pathways through pseudo-time analysis \cite{31}. While there are still a number of challenges with using scRNA-seq for transcriptome analysis, there is no doubt that scRNA-seq will become more prevalent given the wealth of cellular diversity information provided. In fact, scRNA-seq is one the fundamental technologies underpinning the Human Cell Atlas project \cite{32}, which aims to comprehensively map all cell types in humans.

The main focus of this thesis is on transcriptome analysis, in particular single-cell transcriptome analysis. The rate of evolution in scRNA-seq methods and the large-scale adoption of the technology in current and upcoming studies will result in rapid increase in sequencing data generation. Current tools for the analysis of sequencing data will not be able to adequately process these large amounts of data in an efficient manner and thus there is a need to develop scalable tools to efficiently process this data.

1.3 Bioinformatics analysis of sequencing data

After sequencing data has been generated, the data needs to first be processed using bioinformatics software designed for sequence analysis in order to extract meaningful results from
1. Introduction

the data. There are different procedures for the processing of sequencing data depending on the type of analysis being performed. However, the first few steps in processing sequencing data are typically quite similar regardless of the application of sequencing technology.

The first step in the analysis of sequencing data is a quality control analysis of the raw sequencing data using tools such as FastQC [33], which provides a report of the quality of the sequencing reads. Depending on the result of quality control analysis, the sequencing data may need to go through pre-processing steps to perform tasks such as removing low quality reads, trimming of adaptor sequences and filtering of unwanted sequences. There are a number of tools developed for pre-processing sequencing data, including Trimmomatic [34], PRINSEQ [35] and cutadapt [36].

After the quality control steps, the sequencing reads are aligned against the reference genome or transcriptome of the organism studied in order to find the location from which the reads originate from. The type of alignment tool used will depend on the application of sequencing technology, with reads from RNA-seq requiring the use of a splice-aware alignment tool which can take into account large gaps in alignment due to intron splicing. Examples of alignment tools for DNA-seq include BWA [37], bowtie2 [38] and NovoAlign [39], while examples of alignment tools for RNA-seq include STAR [40], HISAT [41] and Subread [42]. An alternative step to read alignment is de-novo sequence assembly, which is required for studying organisms without reference genomes or transcriptomes. Tools for sequence assembly include velvet [43] for DNA-seq reads and trinity [44] for RNA-seq reads.

The steps following the alignment of reads varies depending on the type of analysis to be performed. In genome sequencing analysis, the aligned reads can be used for detecting genomic variation, such as single nucleotide polymorphism and insertion/deletion, with tools such as GATK [45]. For ChIP-seq analysis, the aligned reads are used for detecting ‘peaks’, which indicates the location in the genome where the protein of interest binds to, using tools such as MACS [46]. In the case of RNA-seq analysis, the aligned reads can be processed into
1. Introduction

gene counts by quantifying the number of reads that overlap known gene transcripts using tools such as HTSeq \cite{47} and featureCount \cite{48}. Alternatively, the aligned reads can be used to reconstruct transcripts to identify alternative splicing events and/or new transcript isoforms with tools like StringTie \cite{49} and Scallop \cite{50}. There are further downstream analysis that are typically performed following these steps, such as differential expression analysis, dimensionality reduction, and data visualisation.

1.3.1 Scaling up bioinformatics analysis of sequencing data

Analysis of sequencing data is a time consuming process due to the complexity of the task involved. As highlighted previously, the increasing availability of sequencing technology and the development of new methods such as scRNA-seq means that there is a need to design scalable analysis tools that can efficiently process the data. There have been a number of approaches used for speeding up the analysis of sequencing data and these approaches can largely be classified as either software-based optimisation or hardware-based optimisation.

Software-based optimisation typically involves development of efficient new algorithms to reduce the amount of computation required. One early example of software-based optimisation is the use of heuristic methods for sequence alignment, as implemented by FASTA \cite{51} and BLAST \cite{52}, to perform similarity search within a database of sequences. A more recent example of software-based optimisation is the development of a pseudoalignment-based approach for feature quantification of RNA-seq data, which bypasses the time consuming step of aligning reads to the genome \cite{53}.

Hardware-based optimisation of bioinformatics software is mainly done through parallelisation of computational task. Analysis of sequencing data is particularly suited to parallelisation as it involves execution of the same set of tasks which can be done independently of each other. This type of optimisation has been enabled by the evolution of new hardware
1. Introduction

technologies, such as multi-processor system, cluster computing, grid computing and cloud computing. However, developing software that runs on distributed computing infrastructure is non-trivial due to the unique issues and challenges that are present, including the heterogeneity of computing resources, communication between processes and the scheduling of tasks. An example of bioinformatics software designed for distributed computing infrastructure is mpiBLAST, a parallelised implementation of BLAST using the Message Passing Interface (MPI) communication protocol to orchestrate sequence similarity searches across nodes within a cluster [54].

Recently developed bioinformatics software has started to adopt cloud computing infrastructure for the parallelisation of analysis. Modern cloud computing is becoming more widely accessible due to the growth of cloud computing providers, such as Amazon Web Services, Google Cloud Platform and Microsoft Azure. Cloud computing allows for on-demand access to scalable computing resources, thus eliminating the barrier to accessing large computing infrastructure which was previously limited to large institutions due to the cost associated with set up and maintenance. The adoption of cloud computing technology is also driven by development of software frameworks for big data analysis, such as MapReduce [55] and Apache Spark [56], which are designed for parallelised and distributed computation across large computing clusters. The capability of these big data framework to process very large datasets was demonstrated by their achievement in winning the Daytona Gray contest [57], which is an industry standard benchmark to measure the speed of sorting 100 terabytes of data. The Hadoop framework, which is an open-source implementation of MapReduce, won the contest in 2013 for completing the benchmark in 72 minutes using a cluster containing 2100 nodes, while the Spark framework won the contest in 2014 by completing the benchmark in 23 minutes with a cluster of 206 nodes.
1. Introduction

1.4 Problem statement and thesis aims

My research project is motivated by the following technical problem: How can we build a new generation of highly scalable software to improve the speed, accuracy and interpretability of bioinformatics analysis of NGS data? My main approach is to make use of modern cloud computing technology, and demonstrate the applicability of my tools using RNA-seq data, especially single-cell RNA-seq data. More specifically, this thesis aims to:

1. Develop a scalable framework for the analysis of single-cell transcriptomics data using industry standard Big Data analysis frameworks, such as Apache Hadoop and Apache Spark
2. Develop a cloud-enabled tool for the recovery of unaligned reads using concepts borrowed from the state-of-the-art software testing method of metamorphic testing
3. Enable the visualisation of large biomedical data

The structure of the rest of the thesis is as follows: in chapter two, I perform an in-depth review on the development of scalable bioinformatics software, motivated by the evolution of computing technology. In this chapter, I also cover efforts to test the correctness of bioinformatics software.

Chapter three and four focuses on the development of a scalable framework for the analysis of single-cell transcriptomics data. In chapter three, I introduce the Falco framework – a cloud-based framework for the quantification of single-cell transcriptomic data. The following chapter, chapter four, detail further work I have undertaken to extend the capabilities of the Falco framework.

Chapter five describes the development of a pipeline for the recovery of unaligned reads util-
1. Introduction

ising their similarity to previously aligned reads. The Scavenger pipeline developed supports
the use of cloud-computing in order to parallelise the time-consuming recovery process.

In chapter six, I present a web-based visualisation tool designed for visualisation of millions
of data points. starmap incorporates two visual paradigm to allow the user to explore a high
level overview of the data, while also allowing for user to focus in on fine-grained features of
individual data points.

In the final chapter, I discuss the result of the work developed and examine future prospects
in the development of scalable bioinformatics tools for the analysis of sequencing data.
Chapter 2

Scalability of big data bioinformatics
2. Scalability of big data bioinformatics

The text and figures included in this chapter are adapted from the following publication:


Acknowledgement

The work in the publication was co-authored by research assistant Michael Troup and my supervisor. For the publication, I have contributed to the research and writing for the section on scalability in big data bioinformatics, while Mr Troup has contributed to the validation section. This chapter only contains the section on scalability, meaning that I am the sole author of all the text, figures and tables in this chapter.
2. Scalability of big data bioinformatics

2.1 Overview

This chapter examines one of the important aspects that are central to modern big data bioinformatics analysis – software scalability. We argue in this chapter that the issue of scalability is common to all big data bioinformatics analyses and that it can be tackled using the divide-and-conquer approach. Scalability is defined as the ability for a program to scale based on workload and it has always been an important consideration when developing bioinformatics algorithms and programs. Nonetheless the surge of volume and variety of biological and biomedical data has posed new challenges. We discuss how modern cloud computing and big data programming frameworks such as MapReduce and Spark are being used to effectively implement divide-and-conquer in a distributed computing environment. We hope this chapter will raise awareness of this critical issue in bioinformatics.

2.2 Introduction

The term big data is used to describe data which are large with respect to the following characteristics: volume (amount of data generated), variety (type of data generated), velocity (speed of data generation), variability (inconsistency of data) and veracity (quality of captured data) [58]. Sequencing data is the most obvious example of big data in the field of bioinformatics, especially with the advancement in next-generation sequencing (NGS) technology and single cell capture technology. Other examples of big data in bioinformatics include electronic health records, which contain a variety of information including phenotypic, diagnostic and treatment information; and medical imaging data, such as those produced by magnetic resonance imaging (MRI), positron emission tomography (PET) and ultrasound. Furthermore, emerging big data relevant to biomedical research also include data from social networks and wearable devices.
2. **Scalability of big data bioinformatics**

One particularly major advancement in experimental molecular biology within the last decade has been the significant increase in sequencing data available for analysis, at a cheaper cost [59]. The cost of sequencing per genome has reduced from $100,000,000 in 2001, to $10,000,000 in 2007, down to a figure close to $1000 today. The $1000 genome is already a reality [60]. Currently, the data that comes out of a NGS machine are in the order of several hundred gigabytes for a single human genome. With the rapid advancement in single-cell capture technology and the increasing interest in single-cell studies, it is expected that the amount of sequencing data generated will increase substantially as each single-cell run can generate profiles for hundreds to thousands of samples [61]. In this chapter, we will focus specifically on bioinformatics software that deals with NGS data as this is currently one of the most prominent and rapidly expanding source of big data in bioinformatics.

In this chapter, we argue that one of the main issues that are fundamental to designing and running big data bioinformatics analysis is the need for analysis tools which can scale to handle the large and unpredictable volume of data (Scalability) [62, 63, 64]. In general, there are many other issues associated with bioinformatics big data analysis, such as storage, security and integration [65]. However, these issues have existed even before the rise of big data in bioinformatics, and these issues are typically targeted to specific use cases, such as the storage of sensitive patient data and integration of several specific types of data. Solutions to these specific issues are available [66, 67], though there may be additional challenges associated in implementing the solution due to the increased volume and noise. Nonetheless, these issues are mostly specific to individual application areas. We believe that if we can effectively deal with the problem of scalability, it will go a long way in terms of making big data analysis more widespread in practice. This chapter aims to provide an overview of the technological development that deals with the scalability problems in big data bioinformatics for sequence-based analysis tools.
2. Scalability of big data bioinformatics

2.3 Scalability

Scalability is not a unique challenge in big data analysis. In fact, software scalability has always been an issue since the early days of bioinformatics because of the high algorithmic complexity of some of the algorithms such as those involving global multiple sequence alignment. The early focus on scalability is on parallelising the computation, while a lot less attention is paid on optimally distributing the data. Efforts to make bioinformatics software scalable have continuously been made with the evolution of new hardware technologies, such as cluster computing, grid computing, Graphical Processing Unit (GPU) technology, and cloud computing. Currently in the age of big data bioinformatics, the focus is not only on parallelising computational intensive algorithms, but also on highly distributed storage and efficient communication among various distributed storage or computational units. Furthermore, the volume and variety of data can change dynamically in response to potentially unpredictable user demand. For example, in a medium-sized local sequencing centre, the volume of data can grow rapidly during certain unexpected peak periods, but remain constant during other periods. This variability of demand on computational resources is also a critical feature of modern big data bioinformatics analysis. In this section, we will review the evolution of parallel distributed computing technologies and how they have contributed to solving the issue of scalability of bioinformatics software. In particular, we will discuss how modern cloud computing technology and big data analysis frameworks, such as MapReduce and Spark, can be effectively used to deal with the scalability problem in the big data era.

2.3.1 Cluster Computing

Early attempts at scaling bioinformatics software beyond massively parallel (super) computers involved networking individual computers into clusters to form a parallelised distributed-memory machine. In this configuration, computations are performed by splitting and dis-
2. Scalability of big data bioinformatics

Distributing tasks across Central Processing Units (CPUs) in a way that is similar to the symmetric multiprocessing (SMP) approach utilised in massively parallel computers. Unlike SMP, which relies on a shared main memory, clusters have distributed-memory, with each node having its own memory and hard drive, thus presenting a new challenge in developing software for cluster environments. To help with the development of cluster-based software, communications protocols and software tools, such as Message Passing Interface (MPI) [68] and Parallel Virtual Machine (PVM) [69], have been developed for orchestrating computations across nodes. An example of bioinformatics software designed for cluster computing is mpiBLAST, an MPI-based, parallelised implementation of the basic local alignment search tool (BLAST) algorithm which performs pairwise sequence similarity between a query sequence and a library or database of sequences [54]. The approach taken by mpiBLAST includes the use of a distributed database to reduce both the number of sequences searched and disk I/O in each node, thereby improving the performance of the BLAST algorithm. MASON is another example of MPI-based bioinformatics software for performing multiple sequence alignment algorithms using the ClustalW algorithm [70]. MASON speeds up the execution of ClustalW by parallelising the time and compute-intensive step of calculating a distance matrix of the input sequences, and the final progressive alignment stage.

2.3.2 Grid Computing

The next approach in scaling bioinformatics software comes with the introduction of grid computing, which represents an evolution in the distributed computing infrastructure. Grid computing allows for a collection of heterogeneous hardware, such as desktops, servers and clusters, which may be located in different geographical locations, to be connected through the Internet to form a massively distributed high performance environment [71]. Although conceptually similar to a cluster, grid computing presents a different set of challenges for developing software. The comparatively large latency between nodes in a grid environment
2. Scalability of big data bioinformatics

compared to a cluster environment means that software for grid needs to be designed with minimum communication between nodes. Furthermore, the heterogeneity of the grid environment means that software may need to take into account differences in the underlying operating system and the system architecture of the nodes. Development of bioinformatics software for a Grid typically uses a middleware layer which abstracts away the underlying grid architecture management. A widely-used middleware layer is the Globus Toolkit, a software toolkit for managing and developing in a grid environment [72]. An example of a bioinformatics software for the Grid environment is GridBLAST, an implementation of BLAST with Globus as the middleware layer for distributing BLAST queries across nodes in the grid [73]. Aside from Globus, there are also bioinformatics-specific grid middleware layers such as myGrid [74] and Squid [75].

2.3.3 GPGPU

The introduction of general-purpose computing on GPUs (GPGPUs) revived interest in the massively parallel approach initially used before the distributed computing approach became the mainstream. GPUs are specialised processing units designed for performing graphic rendering. Unlike a CPU, which has a limited number of multi-processing units, a GPU has a large number of processing units in the order of hundreds and thousands, thus allowing for the high computational throughput required for rendering 3D graphics. Though the GPU is not a new technology, early GPU architectures were hardwired for graphics rendering and thus it was not until the development of a more generalised architecture which supported general-purpose computing that GPU become more widely used for computation. As with other technologies, there are challenges associated with implementing bioinformatics software on GPUs due to the single instruction multiple data (SIMD) programming paradigm where data are processed in parallel using the same set of instructions. Due to its architecture, computation for GPU will need to be designed with minimum level of branching.
2. Scalability of big data bioinformatics

(homogenous execution) with high computational complexity in order to fully take advantage of the high multiprocessing capability of the GPU. One of the early bioinformatics software utilising GPGPUs is GPU-RAxML (Randomized Axelerated Maximum Likelihood), a GPU based implementation of RAxML program for the construction of phylogenetic trees using a Maximum Likelihood method [76]. GPU-RAxML utilises the BrookGPU programming environment [77], which supports both OpenGL and DirectX graphic libraries, to parallelise the longest loop in the RAxML program, which accounts for 50% of the execution time. Another example of GPU-accelerated bioinformatics software is CUDASW++, an implementation of the dynamic-programming based Smith-Waterman (SW) algorithm for local sequence alignment [78]. CUDASW++ utilises the CUDA (Compute Unified Device Architecture) programming environment [79], developed for NVIDIA GPU, to implement two parallelisation strategies of the SW algorithm based on the length of the subject sequence.

2.3.4 Cloud Computing

Cloud computing is defined by the United States’ National Institute of Standards and Technology as ‘...a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.’ [80]. Though similar to cluster and grid computing in that cloud computing is based on collections of commodity hardware, cloud computing utilises hypervisor technology to provide dynamic access to ‘virtualised’ computing resources. Virtualisation enables a single hardware resource to ‘host’ a number of independent virtual machines that can run on different operating systems, and which each share some of the underlying hardware resources. Cloud computing is very well suited for big data bioinformatics applications as it allows for on-demand provisioning of resources with a pay-as-you-go model, thus eliminating the need of purchasing and maintaining costly local computing infrastruc-
2. Scalability of big data bioinformatics

ture for performing analyses. Furthermore, the on-demand provisioning of cloud computing also enables the scaling of computational resources to match the workload being performed at any particular time.

Modern cloud computing is widely accessible, and is not limited to researchers in a large university or institution that have a large computer cluster. There are currently three major cloud computing providers which offer pay-as-you-go access to computing resources – Amazon Web Services (AWS), Google Cloud Platform and Microsoft Azure. There are also a number of cloud computing platforms which are freely available to researchers, such as Atmosphere cloud from CyVerse [81], EGI cloud compute [82] and Nectar Research Cloud [83]. Cloud providers can offer access to “instances”, which are the virtual machines for which a user can select from various configurations, including number of CPU, amount of RAM and operating system. The configuration can range from that of a notebook (1 CPU, 2 GB RAM), to a desktop workstation (8 CPU, 16 GB RAM) or even a supercomputer (128 CPU, 2 TB RAM). Furthermore, some cloud providers also offer specialised instances, such as those with a field-programmable gate array (FPGA) or GPUs. An instance typically starts within minutes, and the user is charged for the duration of the instance’s lifetime. As well as traditional instances (servers or workstations in non-cloud terminology), cloud providers offer access to a range of other software and hardware offerings. Other offerings include compute facilities, storage and content delivery, database, networking, analytics, enterprise applications, mobile services, developer tools, management and security tools, and application services.

The services offered by cloud computing providers can be roughly categorised into Data as a Service (DaaS), Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS). Data as a Service is a service where data is provided on-demand for users through the Internet. This type of service is particularly relevant for bioinformatics with the increasing production of biological data as a way to store and share data for analysis. There are currently two DaaS providers for biological data – AWS Public Dataset [84]
2. Scalability of big data bioinformatics

and Google Genomics Public Data [85] — which provide free access to public data sources such as various reference genomes, the 1000 genomes project and The Cancer Genome Atlas. Software as a Service, on the other hand, provides on-demand access to software without the need to manage the underlying hardware and software resources. There have been many bioinformatics SaaS solutions to perform tasks ranging from short-read alignment (CloudAligner [86] and SparkBWA [87]), variant calling (Halvade [88] and Churchill [89]) and RNA-seq analysis (Oqtans [90] and Falco [91]). In Platform as a Service, users are provided with a platform for developing their own software. Some PaaS providers also provide support for automatically scaling the computing resource used based on the workload being run. Unlike the SaaS model where the user is only able to access the provided software, PaaS allows bioinformaticians to develop their own custom bioinformatics pipeline. Examples of PaaS platforms in bioinformatics include Galaxy Cloud [92], a cloud-based scientific workflow system, and DNANexus [93], an AWS based analysis and management platform for next-generation sequencing data. Finally, Infrastructure as a Service is the most basic type of service where users are given access to the virtualised ‘instance’. In bioinformatics, IaaS solutions are typically virtual machine (VM) containers in which a user can deploy to their own customised instance on the cloud. Both CloudBioLinux [94] and CloVR [95] are examples of IaaS solutions for performing bioinformatics analysis. Some cloud providers, especially those targeted for researchers, also provide some pre-built VM snapshots or images with pre-configured software tools which can be deployed to the instance for ease of use.

Containerisation is another virtualisation approach which is becoming increasingly popular in bioinformatics – driven by the introduction of Docker, a simplified, cross-platform tool for deploying application software to containers. Unlike virtual machines, container-based virtualisation – also known as operating-system level of virtualisation – only creates lightweight, isolated virtual environments (called containers) which utilise the system kernel without virtualising the underlying hardware. Containers have high degree of portability as they provide a consistent environment for software regardless of where it is executed. This is particularly
useful in the bioinformatics field to help researchers in reproducing their studies by setting up ‘analysis’ containers which can be reused and shared. Due to its popularity in the software industry, there are a growing number of cloud computing providers which support the deployment of containers, such as AWS EC2 container service, Google Container Engine and Azure Container service.

2.3.5 Programming Frameworks for Big Data Analysis on the Cloud

One important factor that has contributed to the widespread adoption of cloud computing is the development of software frameworks for big data analysis. The nature of big data means that it is difficult to analyse them efficiently using existing computational and statistical methods since they often do not deal with distributed storage and often do not cope well with large data size. MapReduce was introduced by Google in 2004 as both a programming model and an implementation for performing parallelised and distributed big data analyses on large clusters of commodity hardware. In the MapReduce programming model, computation is expressed as a series of Map and Reduce steps, which consumes and produces a list of key-value pairs. Apache Hadoop is an open-source implementation of the MapReduce programming model. The Hadoop framework is composed of several modules, including the Hadoop Distributed File System (HDFS), which is a distributed and fault-tolerant storage system, Hadoop YARN for resource management and job scheduling/monitoring, and the Hadoop MapReduce engine for analysis of data. Halvade is an example of a Hadoop-based bioinformatics tool for performing read alignment and variant calling for genomic data (Figure 2.1a). The Halvade framework is composed of a Map step, which performs alignment of sequencing reads to the reference genome using Burrows-Wheeler Aligner (BWA), and a Reduce step, which performs variant calling in a chromosomal region using Genome Analysis Toolkit (GATK). Other examples of MapReduce-based bioinformatics analysis tools include
2. Scalability of big data bioinformatics

Myrna, which performs RNA-sequencing gene expression analysis [97] and CloudAligner, a short-read sequence aligner.

One disadvantage of the MapReduce programming model is the need to decompose tasks into a series of map and reduce steps. Iterative tasks, such as clustering, do not suit the MapReduce model and thus perform poorly when implemented as MapReduce tasks. Apache Spark is a general purpose engine for big data analysis designed to support tasks incompatible with the MapReduce model, such as iterative tasks and streaming analytics, through the use of in-memory computation [56]. Unlike the MapReduce model, Spark provides a variety of operations, namely transformations and actions, which can be chained to form a complex workflow. Furthermore, Spark supports multiple deployment modes – local mode, standalone cluster mode and using cluster managers, such as YARN — thus allowing for integration with the Hadoop framework. An example of a Spark-based tool is Falco, a single-cell RNA-sequencing processing framework for feature quantification [91]. The Falco pipeline utilises Spark in the main analysis step for performing alignment and quantification of reads to produce gene counts from a portion of the sample, which are then combined to obtain the total gene counts per sample (Figure 2.1b). VariantSpark [98] and SparkBWA [87] are other examples of Spark-based tools for performing population-scale clustering based on genomic variation, and read alignment of genomic data, respectively.
Figure 2.1: Examples of MapReduce-based (a) and Spark-based (b) big data bioinformatics analysis frameworks.
2. Scalability of big data bioinformatics

2.4 Discussion

This chapter has surveyed some methods for implementing scalability for big data bioinformatics software, more specifically for sequence-based analysis tools. With the increasing push to generate sequencing data sets at the single-cell level of resolution [99, 100], it is clear that analysis of these large amount of NGS data will need to be performed on large-scale cloud and/or grid infrastructure, using software which can efficiently scale to handle the large amount of data. However, the techniques and concepts introduced in the chapter are not limited to sequence-based analysis tools and can be adopted to bioinformatics software in other fields. There are already a number of tools which have been developed using big data frameworks in other bioinformatics fields, such as AutoDockCloud, a tool for drug discovery through virtual molecular docking which utilises the Hadoop MapReduce framework [101], and PeakRanger, a tool for calling peaks from chromatin immunoprecipitation sequencing (ChIP-Seq) data also on the MapReduce framework [102].

The central methodological concept for dealing with scalability is divide-and-conquer. The goal is to divide a large task into many small portions and process them in parallel. The main requirement for the success of such an approach is to have an efficient means to manage the division and merging of data and computation, and the availability of a variable amount of IT resources on demand. Modern cloud-based computing technology and big data programming frameworks provide a systematic means to tackle these issues.
Chapter 3

Falco: A quick and flexible single-cell RNA-seq processing framework on the cloud
3. Falco: A quick and flexible single-cell RNA-seq processing framework on the cloud

The text and figures included in this chapter are adapted from the following publication:


Acknowledgement

The work in this chapter was co-authored by research assistant Michael Troup, postdoctoral fellow Dr Paul Lin and my supervisor. I conceived and developed the framework, along with performing the majority of the experiments. Mr Troup contributed in the development of some parts of the framework, while Dr Lin contributed to running some of the experiments. I am the sole author of all the text, figures and tables in this chapter.
3. Falco: A quick and flexible single-cell RNA-seq processing framework on the cloud

3.1 Overview

Single-cell RNA-seq (scRNA-seq) is increasingly used in a range of biomedical studies. Nonetheless, current RNA-seq analysis tools are not specifically designed to efficiently process scRNA-seq data due to their limited scalability. Here we introduce Falco, a cloud-based framework to enable parallelisation of existing RNA-seq processing pipelines using big data technologies of Apache Hadoop and Apache Spark for performing massively parallel analysis of large scale transcriptomic data. Using two public scRNA-seq data sets and two popular RNA-seq alignment/feature quantification pipelines, we show that the same processing pipeline runs 2.6 – 145.4 times faster using Falco than running on a highly optimised single node analysis. Falco also allows users to utilise low-cost spot instances of Amazon Web Services (AWS), providing a 65% reduction in cost of analysis.

3.2 Introduction

Major advancements in single-cell technology have resulted in an increasing interest in single-cell level studies, particularly in the field of transcriptomics [103]. Single-cell RNA sequencing (scRNA-seq) offers the promise of understanding transcriptional heterogeneity of individual cells, allowing for a clearer understanding of biological process [104, 105, 106, 107].

Each scRNA-seq experiment typically generates profiles of hundreds of cells, which is a magnitude larger than the typical amount of data generated by standard bulk RNA-seq experiments. Current RNA-seq processing pipelines are not specifically designed to handle such a large number of profiles. To fully realise the potential of scRNA-seq, we need a scalable and efficient computational solution. The premise of our solution is that state-of-the-art cloud computing technology, which is known for its scalability, elasticity and pay-as-you-go payment model, can allow for a highly efficient and cost-effective scRNA-seq analysis.
3. Falco: A quick and flexible single-cell RNA-seq processing framework on the cloud

There are a number of existing cloud-based next-generation sequencing bioinformatics tools based on the Hadoop framework, an open source implementation of MapReduce [55], or the Spark framework [56]. Halvade, written in Hadoop MapReduce, is designed to perform variant calling of genomic data from FASTQ files, though it also offers support for transcriptomic analysis [88]. SparkSeq [108] and SparkBWA [87], both written in Spark, offers interactive sequencing analysis of BAM files and alignment of FASTQ files respectively. These tools have limitations in the context of scRNA-seq analysis. Of the three tools, only SparkSeq allows for multi-sample analysis, although SparkSeq itself is also limited as it does not perform alignment, which is the main bottleneck in sequence analysis.

Here we use a different approach to utilising cloud-based big data technology. Our framework – Falco – is a framework that allows users to ‘plug-in’ their chosen RNA-seq alignment, quality control, preprocessing and feature quantification tools, and enable the resulting pipeline to run multi-sample analysis of large-scale transcriptomic data on the cloud. Falco utilises Amazon Elastic MapReduce (EMR), a big data processing service for deploying managed Hadoop and Spark clusters on the Amazon Web Services (AWS) cloud.

3.3 Framework

The Falco framework consists of a splitting step, an optional pre-processing step and the main analysis step (Figure 3.1). The first step, the splitting step, is a MapReduce job which splits FASTQ input files stored in the Amazon S3 storage service into multiple smaller FASTQ files. In the case of paired-end reads, the two reads are combined into a single record to ensure that paired-end reads are processed together. The splitting process is performed in order to increase the level of parallelism in analysis and normalise the performance of tools as each chunk will have the same maximum uncompressed size of 256 MB. The next step in the pipeline is an optional step for performing pre-processing of reads, such as adapter
3. Falco: A quick and flexible single-cell RNA-seq processing framework on the cloud

trimming and filtering reads based on quality. The pre-processing step is another MapReduce job which performs pre-processing of the split FASTQ files using any pre-processing tools chosen by the user. The user is asked to supply a shell script with commands to run their selected pre-processing tools, that is then called by the MapReduce job.

The final step of the pipeline is the main analysis step. It performs alignment and quantification of reads using the Spark framework. It was designed such that any RNA-seq alignment and quantification tools can be used within the Falco framework. In the current implementation, each split FASTQ file can be aligned using either STAR [40] or HISAT2 [41] and quantified using either featureCounts [48] or HTSeq [47]. By default, STAR and featureCount will be used for alignment and quantification, however the framework accepts any combination of the tools. The returned gene counts per split are then reduced (i.e., merged) to obtain the total read counts per gene in each sample. The gene count matrix is produced and stored into Amazon S3 storage. Aside from the gene counts, the analysis step also returns selected mapping and quantification reports generated by the selected alignment and quantification tools as well as optional RNA-seq alignment metrics from Picard tools [109].

As part of the pipeline, a script is provided to simplify the creation of the EMR cluster and configure the required software and references on the cluster. Similarly, each of the steps also has a corresponding submission script which will upload the files required for the step and submit the step to the EMR cluster for execution.
Figure 3.1: The Falco framework process pipeline. In the splitting step, reads from one or multiple FASTQ files are split into multiple chunks of size 256 Mb uncompressed. This step makes use of Apache Hadoop MapReduce. A pre-processing step is executed if the data require pre-processing using MapReduce. In the main analysis step, sequence alignment and gene expression quantification are carried out in a highly parallelised fashion using the Apache Spark framework.
3. Falco: A quick and flexible single-cell RNA-seq processing framework on the cloud

3.3.1 Customising Falco framework

The Falco framework allows the user to add custom alignment and/or quantification tools beyond what is provided by default. Instructions are provided in the github wiki which will take the user through the steps required to add their selected tool(s) to the framework. It is expected that the user has moderate to advanced Python proficiency in order to perform customisation of the framework. To ensure that the output of Falco matches that of non-Falco execution, the tools must be compatible with divide-and-conquer approach. Examples of tools which are not compatible with Falco approach include TopHat2 [110] and StringTie [49] as those tools use information from the entire read for performing calling and quantification, respectively. The divide-and-conquer approach used by Falco means that the tools only have partial information from the entire read and thus the output will not necessarily be the same.

3.4 Evaluation

To evaluate the performance of Falco, the runtime of two popular RNA-seq pipelines, STAR followed by featureCounts (S+F), and HISAT2 followed by HTSeq (H+H), is evaluated using two scRNA-seq data sets with and without using the Falco framework. A number of realistic scenarios for analysis in a single computing node were devised – from the naïve single processing approach to a highly parallelised approach. Furthermore, to demonstrate the scalability of Falco, EMR clusters with increasing numbers of core nodes (from 10 to 40) were used to show the effect of adding more computational resources on the runtime of Falco.

In all the comparison, the AWS EC2 instance type used for computation (core node for EMR) is r3.8xlarge (32 cores, 244GB of RAM and two 320GB SSDs). For Falco’s EMR cluster, a
3. Falco: A quick and flexible single-cell RNA-seq processing framework on the cloud

Table 3.1: scRNA-seq processing time with or without Falco

<table>
<thead>
<tr>
<th>System</th>
<th>Nodes</th>
<th>Mouse - embryonic stem cell (hours)</th>
<th>Human - brain (hours)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>S+F*</td>
<td>H+H*</td>
</tr>
<tr>
<td></td>
<td></td>
<td>S+F*</td>
<td>H+H*</td>
</tr>
<tr>
<td>Standalone</td>
<td>1 (1 process)</td>
<td>93.7</td>
<td>233.6</td>
</tr>
<tr>
<td></td>
<td>1 (12 processes)</td>
<td>21.1</td>
<td>32.6</td>
</tr>
<tr>
<td></td>
<td>1 (16 processes)</td>
<td>18.5</td>
<td>28.4</td>
</tr>
<tr>
<td>Falco</td>
<td>10</td>
<td>7.0</td>
<td>11.2</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>4.1</td>
<td>6.4</td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>3.3</td>
<td>4.8</td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>2.8</td>
<td>4.0</td>
</tr>
</tbody>
</table>

*S+F = STAR for aligner and featureCounts for quantification; H+H = HISAT2 for aligner and HTSeq for quantification. Standalone number of processes indicates the number of FASTQ file pairs that are processed in parallel. Timing for Falco includes initialisation and configuration time which are approximately 16 minutes.

A single r3.4xlarge (16 cores, 122GB RAM) was used as the master node for scheduling jobs and managing the cluster. The EMR cluster uses Amazon EMR release 4.6, which contains Apache Hadoop 2.7.2 and Apache Spark 1.6.1, and takes 16 minutes for initialisation and configuration in all cluster configurations used.

Two recently published scRNA-seq datasets were used for evaluation. The first dataset (SRA accession: ERP005988), is a mouse embryonic stem cell (mESC) single cell data containing 869 samples of 200 bp paired-end reads, totalling to $1.28 \times 10^{12}$ sequenced bases, stored in 1.02 Tb of gzipped FASTQ files [107]. The second dataset (SRA accession: SRP057196), is a smaller human brain single cell data containing 466 samples of 100 bp paired-end reads, totalling to $2.95 \times 10^{11}$ sequenced bases and 213.66 Gb of gzipped FASTQ files [106].

Comparing the performance of a single node, with different parallelisation approaches, against Falco shows that running the S+F pipeline on Falco results in a speedup of 2.6x (10 nodes vs 16 processes) to 33.4x (40 nodes vs 1 process) for the mouse dataset and 5.1x (10 nodes vs 16 processed) to 145.4x (40 node vs 1 process) for the human dataset. For the
3. Falco: A quick and flexible single-cell RNA-seq processing framework on the cloud

Table 3.2: Falco cost analysis: on-demand vs. spot instances for STAR + featureCounts

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Cluster size</th>
<th>Time (hours)</th>
<th>On-demand cost (USD)</th>
<th>Spot cost (USD)</th>
<th>% Savings</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mouse - embryonic</td>
<td>10 node</td>
<td>8</td>
<td>247.20</td>
<td>85.67</td>
<td>65.34</td>
</tr>
<tr>
<td></td>
<td>20 node</td>
<td>5</td>
<td>301.00</td>
<td>99.09</td>
<td>67.08</td>
</tr>
<tr>
<td></td>
<td>30 node</td>
<td>4</td>
<td>258.00</td>
<td>115.71</td>
<td>55.15</td>
</tr>
<tr>
<td></td>
<td>40 node</td>
<td>3</td>
<td>356.40</td>
<td>114.11</td>
<td>67.98</td>
</tr>
<tr>
<td>Human - brain</td>
<td>10 node</td>
<td>3</td>
<td>92.70</td>
<td>32.13</td>
<td>65.34</td>
</tr>
<tr>
<td></td>
<td>20 node</td>
<td>2</td>
<td>120.40</td>
<td>39.64</td>
<td>67.08</td>
</tr>
<tr>
<td></td>
<td>30 node</td>
<td>2</td>
<td>179.00</td>
<td>57.86</td>
<td>67.68</td>
</tr>
<tr>
<td></td>
<td>40 node</td>
<td>2</td>
<td>237.60</td>
<td>76.08</td>
<td>67.98</td>
</tr>
</tbody>
</table>

Time rounded up to whole hour including cluster startup. Price used for r3.8xlarge instance is USD$2.660/hr (on-demand price) and USD$0.64/hr (average spot price for June 2016).

The H+H pipeline, Falco gives a speedup of 2.5x (10 nodes vs 16 processes) to 58.4x (40 nodes vs 1 process) and 4.0x (10 nodes vs 16 processes) to 132.5x (40 nodes vs 1 process) for the mouse and brain datasets respectively (Table 3.1). The disparity in the speed-up between the two datasets is due to different pre-processing tools being employed, with the human dataset utilising more pre-processing steps in the original publication [106]. We also note that the gene expression quantification produced by a given pipeline is the same regardless of whether the Falco framework was used.

For the scalability comparison, it can be seen that the runtime of the pipeline decreases with increasing cluster size (Table 3.1), though the trend is gradual rather than linear. Analysis of the runtime for each step in the framework shows a similar gradual decrease in runtime for pre-processing and analysis steps (Figure 3.2). For the splitting step, a different trend is seen where there is little to no decrease in runtime for cluster size ≥ 20 nodes. The lack of speed up for splitting is due to the number of executors exceeding the number of files to be split and the limitation of time taken to split large files as the distribution of file size in both test datasets is uneven (Figure 3.3).

To save cost, EMR allows for the usage of reduced price spot computing resources. The
Figure 3.2: Falco processing time split by steps for STAR+featureCounts (a) and HISAT2+HTSeq (b) pipelines for mouse embryonic stem cell and human brain data analysis.
Figure 3.3: Size of the FASTQ file (in terms of gigabytes) of individual cells in each single-cell RNA-seq data set. (a) Mouse embryonic stem cells (SRA accession: ERP005988). (b) Human brain cells (SRA accession: SRP057196).
3. Falco: A quick and flexible single-cell RNA-seq processing framework on the cloud

Table 3.3: Falco cost analysis: on-demand vs. spot instances for HISAT2 + HTSeq

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Cluster size</th>
<th>Time (hours)</th>
<th>On-demand cost (USD)</th>
<th>Spot cost (USD)</th>
<th>% Savings</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mouse -</td>
<td>10</td>
<td>13</td>
<td>401.70</td>
<td>139.10</td>
<td>65.37</td>
</tr>
<tr>
<td>embryonic</td>
<td>20</td>
<td>7</td>
<td>421.40</td>
<td>138.60</td>
<td>67.11</td>
</tr>
<tr>
<td>stem cell</td>
<td>30</td>
<td>5</td>
<td>447.50</td>
<td>144.50</td>
<td>67.71</td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>4</td>
<td>475.20</td>
<td>152.00</td>
<td>68.01</td>
</tr>
<tr>
<td>Human -</td>
<td>10</td>
<td>5</td>
<td>154.50</td>
<td>53.50</td>
<td>65.37</td>
</tr>
<tr>
<td>brain</td>
<td>20</td>
<td>3</td>
<td>180.60</td>
<td>59.40</td>
<td>67.11</td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>2</td>
<td>179.00</td>
<td>57.80</td>
<td>67.71</td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>2</td>
<td>237.60</td>
<td>76.00</td>
<td>68.01</td>
</tr>
</tbody>
</table>

Time rounded up to whole hour including cluster startup. Price used for r3.8xlarge instance is USD$2.660/hr (on-demand price) and USD$0.64/hr (average spot price for June 2016). Spot prices fluctuate depending on the availability of the unused computing resource and the spot instance is obtained by supplying a bid for the resource. The use of spot instances for analysis provide a substantial saving of around 65% compared to using on-demand instances (Table 3.2 and 3.3). The trade-off with using spot instances is that the computing resource could be terminated should the market price for that resource exceed the user’s bid price.

3.5 Summary

Falco is a cloud-based framework for analysis of large-scale single-cell transcriptomic data. The framework is developed using the Big Data framework of Apache Hadoop and Spark in order to enable scalable analysis of large amounts of data. Falco implements an end-to-end pipeline for processing of scRNA-seq data - allowing for pre-processing of sequencing reads, followed by alignment and quantification of reads to produce a gene-expression matrix that can be used in further downstream analyses. One of the features of the Falco framework is the ability to customise the tools used for alignment and read quantification, with the Falco framework currently supporting STAR and HISAT2 for alignment, and HTSeq and
featureCount for quantification. Based on benchmarks using two scRNA-seq datasets, Falco is able to speed up the analysis by 2.6 to 145.4 times compared to performing analysis on a highly-optimised single computer execution. Falco utilises the AWS cloud computing environment due to the features offered by AWS, including EMR - a service that allows users to easily deploy managed Hadoop and Spark clusters - and spot instance - allowing for a reduction in analysis costs by up to 65% compared to on-demand instances.

The current version of Falco does not support sequencing reads produced by droplet-based single-cell capture methods, such as 10x chromium or Drop-seq, as a single FASTQ file produced from these methods contains reads from multiple samples (cells). Furthermore, the demultiplexing technique utilised in 10x Genomics’ Cell Ranger pipeline is not compatible with the analysis step of Falco as it requires sequencing reads to be aligned before the reads can be demultiplexed and quantified as the demultiplexing technique utilised is based on the amount of reads aligned per cell barcodes. In comparison, Falco performs quantification of reads immediately after alignment of reads, as Falco expects each input FASTQ file to contain only reads from a single sample.
Chapter 4

Falco 2.0: An extension of the Falco framework for alignment and transcript assembly of single-cell RNA-seq data
4. Falco 2.0: An extension of the Falco framework for alignment and transcript assembly of single-cell RNA-seq data

The text and figures included in this chapter are adapted from a manuscript that will soon be submitted.

Acknowledgement

The work in this chapter was co-authored by Honours student Abhinav Kishore and Benjamin Phipps, and my supervisor. Mr Kishore and Mr Phipps have each developed the prototype pipeline for one of the new modes and performed testing of the prototype pipeline under my guidance. I have developed the final version of the pipeline and have performed all experiments included in this chapter. I am also the sole author of all the text, figures and tables in this chapter.
4. Falco 2.0: An extension of the Falco framework for alignment and transcript assembly of single-cell RNA-seq data

4.1 Introduction

In the previous chapter, we detailed the development of the Falco framework for scalable analysis of single-cell RNA-seq (scRNA-seq) data on the cloud. The initial version of the Falco framework was designed for the quantification of scRNA-seq datasets as most downstream analysis for scRNA-seq, such as differential expression analysis, cell population clustering and cell markers identification, are based on gene expression. However, there are other types of downstream analysis which can be performed on scRNA-seq data that do not require gene expression, including novel isoform identification and immune cell receptor reconstruction. In this chapter, we extend the Falco framework to enable two additional modes of analysis: (1) alignment-only mode, where the output is an alignment file for each sample, and (2) transcript assembly mode, where the output is a reconstructed transcript isoform annotation based on the data. Collectively, these new modes will enable Falco to be a comprehensive, scalable bioinformatics platform for processing full-length single-cell RNA-seq data.

4.1.1 Scalable tools for read alignment

The main step in most RNA-seq analyses is the alignment of sequencing reads produced by the sequencing machine against the reference genome or transcriptome to find the location from which the reads originate. The positional information of the reads, together with the sequences of the reads themselves, forms the basis from which many different downstream analyses can be performed, such as gene expression analysis, variant calling, and novel isoform identification. The read alignment step is typically one of the most time consuming steps during RNA-seq analysis due to the complex algorithm utilised during the read alignment process. There have been a number of recently published tools which are designed to skip this expensive step through the use of pseudo-alignment methods, such as kallisto [53] and Salmon [111]. However, these tools are designed specifically for read quantification and
4. Falco 2.0: An extension of the Falco framework for alignment and transcript assembly of single-cell RNA-seq data

dependent, are not applicable to other types downstream analyses.

As previously mentioned in the introduction chapter, there are a number of tools which have been published for alignment of RNA-seq reads, including STAR [40], HISAT2 [41] and Subread [42]. While these tools have been optimised to perform read alignment in a time-efficient manner through the use of parallelisation, the level of parallelisation is typically limited to a single machine only. With the rapidly increasing number of profiles which can be generated by scRNA-seq techniques, there is a need to develop tools which can scale across many machines to perform read alignment of large datasets in a scalable manner. We have previously developed a scalable cloud-based framework for the analysis of scRNA-seq data, though the framework was designed for read quantification. In order to enable the Falco framework to perform other types of downstream analyses beyond what is currently supported, we will need to extend the framework to perform alignment-only mode, whereby the output of the analysis are alignment information for the individual scRNA-seq samples.

The idea of parallelising read alignment across distributed computing infrastructure is not novel - there are already existing tools which have been developed to perform read alignment on cluster computing, grid computing and cloud computing infrastructures. Within the context of tools developed using Big Data frameworks, there are Hadoop-based tools, such as Halvade-RNA [112] and HSRA [113], and Spark-based Rail-RNA [114] for alignment of spliced reads. Halvade-RNA is mainly designed for variant calling of RNA-seq data using STAR as the aligner and GATK [45] for variant calling, though it is able to optionally produce alignment information as the output. HSRA, on the other hand, is designed specifically for alignment of RNA-seq data using HISAT2 as the aligner. These two tools are not suitable for alignment of scRNA-seq datasets as they are mainly designed for processing individual samples and thus are unable to properly analyse the large number of samples present in scRNA-seq data. In contrast, Rail-RNA is able to perform multi-sample alignment of RNA-seq data using a modified Bowtie algorithm [115] which is able to handle spliced reads. It should be noted, however, that the alignment tool used by Rail-RNA is non-configurable,
4. Falco 2.0: An extension of the Falco framework for alignment and transcript assembly of single-cell RNA-seq data

unlike the Falco framework, which allows the user to select and customise the alignment tool used. Furthermore, Rail-RNA does not provide a way to pre-process the input sequencing reads, meaning that it is up to the user to manually pre-process the sequencing reads by themselves.

4.1.2 Scalable tools for transcript assembly

Unlike the prokaryote genome, genes within the eukaryotic genome are composed of both coding and non-coding components, called exons and introns, respectively. As such, there is an additional processing step called splicing that is performed between transcription of messenger RNA (mRNA) and translation of the ‘mature’ mRNA to remove the intronic regions - leaving only the coding exonic regions in the mature mRNA. However, the splicing process also enables the generation of multiple isoforms of proteins from the same precursor mRNA, through exclusion or inclusion of different subsets of exonic regions - a process known as alternative splicing. Alternative splicing is a commonly occurring process within the human genome, with >95% of the multi-exonic genes having 2 or more isoforms [116]. The different isoforms of proteins produced by alternative splicing have different functionality, with some isoforms being expressed in specific cell types [117] and novel isoforms arising from mutations resulting in diseases such as cancer [118].

Current methods of isoform analysis are largely dependent on existing transcript isoform information available from reference annotation, such as those published by ENCODE and UCSC. However, there are limitations with using annotation for studying isoforms as we are restricted to known transcripts only. While this is less of an issue in human and well-annotated model organisms, isoform analysis will not be as accurate for non-model organisms or organism with limited/partial annotation information. Furthermore, novel isoform which may arise due to mutation will not be detectable when using existing annotation. In order
to alleviate the problem of detecting new isoform for isoform analysis, transcript assembly can be utilised to detect and update existing annotations with novel isoforms.

As the name implies, transcript assembly is the process of recovering transcript sequence through assembly of reads. Transcript assembly is essentially a variation of the graph reduction problem [119], which is difficult to solve and therefore requires heuristic or approximation-based approaches. There are two types of approaches for performing transcript assembly, depending on the information used for the assembly process. In genome-guided transcriptome assembly, alignment information obtained by aligning reads against the genome are used to create graphs for representing overlapping reads, which are then used to compute gene isoforms and transcripts through graph traversal. This approach relies on the quality of the alignment information and is therefore more suitable for studying isoforms of genes from organisms with high quality reference genomes. Cufflinks [120], StringTie [49] and Scallop [50] are some examples of tools which utilise the genome-guided transcriptome assembly approach.

Another approach for transcriptome assembly is the de novo transcriptome assembly approach where transcripts are assembled from sequencing reads. In the de novo approach, De Bruijn graphs are constructed from the sequences of the reads followed by traversal of the graph in order to compute gene isoforms and transcripts. This approach is more suited for studying isoforms of genes from organisms where the reference genome is not available or is of poor quality, but also for studying isoforms of genes which have a high degree of editing and/or splicing, such as the immune genes. One disadvantage of the de novo approach is the higher computational cost compared to the genome-guided approach. However, it does outperform the previous approach especially in samples with high read coverage, though it is less sensitive in detecting transcripts with low abundance [119]. Tools designed to perform de novo transcriptome assembly include Trinity [44], Trans-ABySS [121] and Oases [122].

Current tools for transcriptome assembly are mainly designed for bulk RNA-seq datasets
4. Falco 2.0: An extension of the Falco framework for alignment and transcript assembly of single-cell RNA-seq data

and will not scale for analysing scRNA-seq datasets. There are a small number of tools which are designed specifically for scRNA-seq such as BASIC [123] and V(D)J Puzzle [124], although their functionality is limited to reconstructing immune cell (B- and T-cell) receptors for study of immune-repertoire diversity. These tools use a hybrid transcriptome assembly approach to reduce the computational cost of performing de novo assembly by applying a filtering step based on alignment of sequencing reads. BASIC implements this filtering step by aligning reads against a database of known immune sequence, followed by assembling reads that align to the database using Trinity. V(D)J Puzzle, on the other hand, aligns the sequencing reads against the genome, followed by filtering of reads that maps to certain immune region within the genome before performing de novo assembly using Trinity.

Due to the lack of scalable transcriptome assembly tools which can handle scRNA-seq datasets, we plan to incorporate a transcriptome assembly analysis feature into the Falco framework to enable assembly of transcriptomes for large datasets in a scalable manner. One of the motivations behind incorporating transcript assembly analysis is the ability to create a more accurate gene annotation - including novel gene isoforms - that can then be used for quantifying gene and/or isoform expression.

4.2 Methods

4.2.1 Alignment-only mode

As described in the previous chapter, the initial version of the Falco framework is composed of three steps - a splitting step for splitting and interleaving of input fastq files into read chunks, an optional pre-processing step for performing pre-processing of the read chunks and an analysis step for alignment and quantification of the read chunks. To implement the alignment-only mode within the Falco 2.0 framework, we have designed a new alignment
4. Falco 2.0: An extension of the Falco framework for alignment and transcript assembly of single-cell RNA-seq data

analysis step to replace the read quantification analysis step in the Falco framework (Figure 4.1). The alignment analysis step takes in the same read chunks input as the previous read quantification analysis step and will output a single alignment file for each sample into either S3 or HDFS, depending on the output location specified by user. Similar to the read quantification analysis step, the alignment analysis step is configurable by the user and currently supports both STAR or HISAT2 as the aligner. A new submission script has also been created to allow users to easily submit the alignment analysis step to the EMR cluster.

Structure

The alignment analysis step is a Spark job which consists of two stages - alignment of read chunks, followed by concatenation of the aligned chunks. In the alignment stage, the interleaved reads within the read chunks are first converted to FASTQ file format so that it can be read by the alignment tool. The alignment tool - STAR or HISAT2 - is then executed using Python’s built-in subprocess library in order to perform alignment of reads against the reference genome. The output of the alignment tool is a BAM alignment file in the case of STAR and a SAM alignment file in the case of HISAT2. As such, an extra processing step of converting SAM to BAM using Samtools is required when HISAT2 is used as the alignment tool. The binary-based BAM file format is chosen over the text-based SAM file format due to the space efficiency of the BAM format, which is achieved through compression of alignment records. The alignment chunk is then uploaded to a temporary location within HDFS or S3 and the location of the alignment chunk is output, together with the sample name from which the read chunks originate.

A shuffling process is then performed to group together the locations of the alignment chunks per sample. This is followed by a concatenation stage that combines the alignment chunks into a single alignment file for each sample. During the concatenation stage, the alignment chunks are iteratively copied from the temporary location into the local disk and
concatenated to a previously concatenated file using Samtools. The iterative concatenation of alignment chunks is chosen over batch concatenation of the alignment chunks due to the constraint of disk space available in the worker since there can be an arbitrary number of chunks for a single sample. Once all the chunks are concatenated into a single alignment file, it is then uploaded to the output location specified by the user, which can either be in S3 or HDFS. Finally, the alignment chunks stored in the temporary location are deleted to free up space for the next analysis.

Considerations

The current approach used in the alignment step is not the most efficient approach, particularly due to the writing and reading cost incurred from storing the alignment chunks in a temporary location on HDFS or S3 in between the two stages of the alignment step. One of the more efficient approaches which was explored during early development of the alignment step is to output the alignment records during the alignment stage, followed by the shuffling process to combine all alignment records for each sample, before writing the combined aligned records for each file into a single alignment file and uploading it to the output location. However, this approach suffers from memory issues due to the requirement of producing a single alignment file per sample, as a sample can have an arbitrary amount of read chunks, and the strict memory management of the Spark executor by the YARN resource manager, which will kill the executor if it utilises more memory than is allocated.
Figure 4.1: The Falco 2.0 framework alignment-only pipeline. The pipeline is composed of the splitting and pre-processing steps from the original Falco framework and the new Spark-based alignment step from the Falco 2.0 framework. The alignment step is composed of two stages - an alignment stage, where read chunks are aligned and stored in a temporary location in HDFS, and a concatenation stage, where alignment chunks from the same sample are concatenated to obtain the full alignment result.
4. Falco 2.0: An extension of the Falco framework for alignment and transcript assembly of single-cell RNA-seq data

One possible way to alleviate the memory issue is to produce multiple alignment files per sample (i.e. grouped by chunks or by chromosome), though this approach will require users to manually merge the alignment files by themselves which is not ideal, especially for large number of samples. Another potential solution which was explored is to increase the amount of memory allocated to the executor so that the executor can handle a larger amount of input. This approach unfortunately did not alleviate the issue of the executor being killed and is also not practical as the memory issue may occur again with even larger file sizes. Furthermore, by increasing the memory allocated per executor, it will reduce the number of executors available therefore reducing the processing capability of the cluster and increasing the analysis time. Given the limitations of the more efficient approach, we elected to use the current approach of storing alignment chunks in a temporary location in HDFS or S3 as this approach will not result in memory issues due to the small amount of output produced (i.e. location of alignment chunks rather than entire alignment records) and does not require the input FASTQ file to have a certain file size limit.

4.2.2 Transcript assembly mode

Transcript assembly was implemented within the Falco 2.0 framework using a similar approach as incorporating the alignment analysis - through the creation of a new transcript assembly step which performs alignment of sequencing reads followed by assembly of transcripts (Figure 4.2). The genome-guided transcript assembly approach was chosen over the de novo transcript assembly approach due to the high computational cost of de novo assembly and the complexity of adapting existing de novo transcript assembly tools to work with the parallelisation approach utilised by Falco. The input of the transcript assembly step is the read chunks input used by both the read quantification and alignment analysis steps, with the output of the step being an annotation file containing the assembled transcript. As with the other analysis steps, the transcript assembly step allows user to use either STAR
4. Falco 2.0: An extension of the Falco framework for alignment and transcript assembly of single-cell RNA-seq data

or HISAT2 as the aligner and either StringTie or Scallop as the transcript assembly tool.

Parallelising transcript assembly

Transcript assembly requires a different parallelisation approach compared to the alignment and read quantification analysis. In transcript assembly, reads from the same region – such as genes – will need to be processed together to allow for sharing of information from different reads in order to construct the transcript itself. This is in contrast to the alignment and read quantification analysis where reads can be processed independently of each other during both the alignment and quantification process. Given this constraint, the aligned reads will need to first be grouped together prior to the transcript assembly process. Here, we consider a number of approaches that can be utilised to parallelise transcript assembly of multiple samples while still allowing for sharing of information between reads.

A simple and naive approach for parallelising transcript assembly is to first group the reads by sample, followed by performing transcript assembly across samples in parallel. This approach can easily be implemented by extending the previous alignment step to perform transcript assembly after concatenation of alignment chunks. However, this approach has limitations in terms of the parallelisation capability as the amount of parallelisation that can be achieved during the transcript assembly processing is dependent on the number of samples in the dataset. This approach is also not suitable for the type of data to be analysed - primarily scRNA-seq datasets - due to the low amount of reads sequenced per cell (sample), which may limit the ability of the transcript assembly tool to assemble transcripts as there may not be enough reads to support the transcript, and the possibility of drop-out, where lowly expressed genes/transcripts are not captured and sequenced in a cell. In order to alleviate the issues inherent to scRNA-seq datasets, we will need to perform transcript assembly using the combined reads from all samples.
4. Falco 2.0: An extension of the Falco framework for alignment and transcript assembly of single-cell RNA-seq data

The need to combine reads across samples means that the approaches for parallelising transcript assembly will need to be based on segmenting the genome into sections which can be processed in parallel. A natural approach to divide the genome is to use chromosomes for segmentation. This approach is particularly advantageous as transcript assembly between different chromosomes can be done independently of each other. However, there are typically only a small number of chromosomes in the genome of an organism, particularly in genomes of reference organisms, therefore the amount of parallelisation that can be achieved is somewhat limited. The approach used in the implementation of the transcript assembly step is to divide the genome into smaller regions – or bins – to maximise the parallelisation of the transcript assembly process. Ideally, the bins should be created based on regions where reads are clustered, which indicates the presence of transcripts. Nonetheless, performing clustering of reads to create bins is a computationally expensive process and therefore the binning strategy used in the transcript analysis step is to create identically sized bins across the entire genome. This binning strategy does have a disadvantage of potentially separating transcripts across bins and thus producing partial non-overlapping transcripts. To solve this issue, the bins are made to overlap each other by at least the length of the longest gene in human – Titin – to ensure that at least one bin can produce the full length transcript. Moreover, a merging step is performed after transcript assembly of all bins with the reference annotation using StringTie in order to remove redundant transcripts and to produce an updated annotation file containing both reference transcript and novel transcript.
4. Falco 2.0: An extension of the Falco framework for alignment and transcript assembly of single-cell RNA-seq data

Figure 4.2: The Falco 2.0 framework transcript assembly pipeline. The pipeline is also composed of the splitting and pre-processing steps from the original Falco framework in addition to the new Spark-based transcript assembly step from the Falco 2.0 framework. The transcript assembly step is composed of a number of stages, including an alignment stage, which performs alignment of read chunks and binning of the alignment result; an assembly stage which perform transcript assembly in parallel, and a merging step, where assembled transcripts are merged with the reference annotation to produce an updated annotation.
4. Falco 2.0: An extension of the Falco framework for alignment and transcript assembly of single-cell RNA-seq data

Structure

The transcript assembly step is implemented as a Spark job consisting of four stages - alignment of read chunks, assembly of reads per bin, merging of assembled transcripts against the reference annotation and, optionally, comparison of the updated annotation against the reference annotation. The first stage – alignment of read chunks – is implemented in a similar manner to the alignment stage in the alignment analysis step, where read chunks are aligned against the reference genome using either STAR or HISAT2. However, unlike the alignment analysis step, the aligned reads are not stored in a temporary location, but rather each alignment record is output together with the names of the bins that overlap that particular read. The bin names are calculated based on the locations where the reads align to in the genome and each read may be output multiple times depending on the number of bins that it overlaps. In order to reduce the amount of data that needs to be shuffled, the read sequence and the sequence quality was removed from the alignment record as this information is not utilised in the transcript assembly process.

The alignment records are then shuffled in order to group records from the same bins together. This is followed by an assembly stage where the alignment records are written to an alignment file and sorted by co-ordinate using Samtools [125]. The transcript assembly tool – StringTie or Scallop – is then executed using Python’s subprocess library to perform genome-guided transcript assembly with the sorted alignment file as input. Depending on the transcript assembly tool chosen, users can also choose to utilise the reference annotation when performing transcript assembly. In this case, a partial annotation file, created by filtering the reference annotation to select only transcripts located in the chromosome of the bin being processed, is included as an input when executing StringTie. The annotation filtering step is performed to reduce both the execution time and the amount of output produced by StringTie, as it only needs to consider a smaller subset of reference transcripts during transcript assembly. After execution of the transcript assembly tool, the assembled
transcripts are then outputted together with the name of the bin.

The transcripts then undergo another shuffling process in order to sort the transcripts by the bin names and to group the transcripts across all bins. The aggregated transcripts are collected into the main 'driver' executor where it is passed into the merging stage. In the merging stage, the transcripts are first written into an annotation file, followed by execution of StringTie in GTF merge mode using both the assembled annotation file and reference annotation file as input. The resulting merged (updated) annotation file, containing both the reference transcripts and newly assembled transcripts, is then uploaded to the location specified by user in either S3 or HDFS.

The transcript assembly step also has an optional fourth stage that performs comparison of the merged annotation against the reference annotation using the GffCompare tool [49]. GffCompare will calculate the sensitivity and precision metrics of the updated annotation as compared to the reference annotation at base, exon, intron, intron chain, loci and transcript levels. The comparison statistics produced by the comparison tool will also be uploaded to the location specified by the user.

4.3 Results

4.3.1 Correctness and scalability of Falco’s alignment-only mode

Correctness

One of the features of the read-quantification mode in the initial version of the Falco framework is the production of the gene expression matrix that is identical to that produced in a sequential analysis, where reads are not split into smaller chunks. This was achieved through
4. Falco 2.0: An extension of the Falco framework for alignment and transcript assembly of single-cell RNA-seq data

careful selection of tools that are known to be deterministic (STAR, HTSeq \cite{47} and featureCounts \cite{48}) or by adjusting the parameters of the tool to ensure the output produced is deterministic (HISAT2). As such, it will be ideal for the alignment-only mode to also produce alignment outputs that are identical to those produced in a sequential analysis. In order to test this hypothesis, we randomly selected 100 files from both the mouse embryonic stem cell (ESC) single cell dataset and the human brain single cell dataset and performed alignment on these datasets using both sequencing alignment on a single node and Falco 2.0. We then compared the alignment file produced by the two different approaches to see if the outputs produced are identical. The comparison was performed by first sorting the alignment files by their read name using Samtools, followed by running the diff command with the two alignment files as input.

The result of the comparison shows that the alignment files produced with STAR as the alignment tool contain identical alignment records when run through either Falco 2.0 or sequentially, with some minor difference in the header of the alignment file due to the inclusion of the command used for running STAR in the program (PG) and text command (CO) records. In contrast, the alignment records produced by HISAT2 with default parameters shows some differences between Falco 2.0-based and sequential runs due to HISAT2 being non-deterministic. Therefore, the \texttt{--tmo} parameter was again used when running HISAT2 in order to make HISAT2 produce deterministic output by performing alignment within known transcripts only. The result of the comparison when running HISAT2 with the \texttt{--tmo} parameter shows that the alignment files produced contain identical alignment records, with a minor difference in the value of the PG record in the header of the alignment.

** Scalability analysis

In order to evaluate the performance of the Falco 2.0 alignment-only analysis, we performed a comparison of runtime for STAR and HISAT2 using two single-cell RNA-seq data sets
4. Falco 2.0: An extension of the Falco framework for alignment and transcript assembly of single-cell RNA-seq data

with and without using the Falco 2.0 framework, similar to the evaluation done for the initial version of the Falco framework. As with the evaluation of the Falco framework, the single-cell RNA-seq datasets used are a mouse embryonic stem cell (ESC) single cell dataset, containing 869 samples of 200 bp paired-end reads, stored in 1.02 Tb of gzipped FASTQ files [107]; and a human brain single cell data containing 466 samples of 100 bp paired-end reads stored in 213.66 Gb of gzipped FASTQ files [106]. We utilised the same configuration for analysis in a single computing node - ranging from the naive single processing approach to a highly parallelised approach - and for the size of the EMR clusters - ranging from 10 to 40 nodes, together with the same AWS EC2 instance type for single node (r3.8xlarge) and Falco 2.0 cluster (master - r3.4xlarge, core - r3.8xlarge). For a fair comparison between the single-node based runs and the Falco 2.0 runs, the timing for alignment on the Falco 2.0 framework includes the timing for both the cluster set-up and FASTQ splitting step as these pre-processing steps are only necessary when performing alignment using the Falco 2.0 framework.

Performing alignment using STAR on a single node with differing parallelisation approaches results in runtimes ranging from 35 hours down to 20 hours for the mouse dataset and 11 hours to 5 hours for the human dataset. In contrast, the runtime for alignment using STAR on the Falco 2.0 framework ranges from 8 hours down to just 3.5 hours for the mouse dataset and 1.7 hours down to less than an hour for the human dataset, representing a minimum speed-up of 2.5x (10 nodes vs 12 processes for the mouse dataset) up to 15.8x (40 nodes vs 1 process for the human dataset) (Table 4.1). Similarly, performing alignment using HISAT2 on a single node with differing parallelisation approach results in a minimum runtime of 15 hours and 3 hours for the mouse and human datasets, respectively, with the mouse dataset taking close to 2 days to run on 1 process. Falco 2.0, on the other hand, was able to complete the alignment for the mouse dataset in less than 6 hours and the human dataset in less than 1.2 hours, representing a speed-up ranging from 2.5x (10 nodes vs 16 processes for the human dataset) up to 16.4x (40 nodes vs 1 processes for the mouse dataset) (Table 4.1).
Table 4.1: Comparison of runtime for alignment of single cell datasets with and without the Falco 2.0 framework

<table>
<thead>
<tr>
<th>System</th>
<th>Nodes</th>
<th>Mouse - embryonic stem cell (hours)</th>
<th>Human - brain (hours)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>STAR</td>
<td>HISAT2</td>
</tr>
<tr>
<td>Standalone</td>
<td>1 (1 process)</td>
<td>34.9</td>
<td>42.7</td>
</tr>
<tr>
<td></td>
<td>1 (12 processes)</td>
<td>20.2</td>
<td>14.9</td>
</tr>
<tr>
<td></td>
<td>1 (16 processes)</td>
<td>N/A</td>
<td>14.9</td>
</tr>
<tr>
<td>Falco 2.0</td>
<td>10</td>
<td>8.0</td>
<td>5.9</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>4.7</td>
<td>3.6</td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>3.8</td>
<td>2.9</td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>3.5</td>
<td>2.6</td>
</tr>
</tbody>
</table>

Standalone number of processes indicates the number of FASTQ file pairs that are processed in parallel. Timing for Falco 2.0 includes initialisation and configuration time which are approximately 10 minutes. Runtime for STAR with 16 processes is not available as some STAR processes are killed by the operating system, resulting in failure of the job.

Comparison of runtimes across cluster sizes for alignment with Falco 2.0 framework shows a decrease in runtime with increasing cluster size (Table 4.1), indicating the scalability of the alignment-only analysis on the Falco 2.0 framework. However, the runtime does not linearly decrease with increasing cluster size, with the maximum speedup of 2x achieved by increasing the cluster size from 10 nodes to 20 nodes. The minimal difference in analysis time for cluster ≥ 20 nodes can partially be attributed to the constant initialisation time and the lack of speed-up in the splitting step, as previously highlighted in the scalability analysis for the initial Falco framework. Another reason for the lack of speedup is due to second stage in the alignment-only step that performs concatenation of the alignment chunks for each samples, meaning that the speedup for this stage is limited by the size of the input files and the subsequent number of read chunks that need to be concatenated (Figure 4.3). Therefore, the minimal reduction in runtime of the second stage for the mouse and human datasets can be explained by the uneven distribution in the size of the FASTQ files of both the mouse and human datasets, with some samples having input size that is 9x larger compared to the median input size.
4. Falco 2.0: An extension of the Falco framework for alignment and transcript assembly of single-cell RNA-seq data

Figure 4.3: Falco 2.0 alignment-only processing time split by steps and stages for STAR and HISAT2 pipelines in the analysis of mouse embryonic stem cell and human brain single cell data. The timings shown do not include cluster initialisation time, as it is constant across differing cluster sizes.
4. Falco 2.0: An extension of the Falco framework for alignment and transcript assembly of single-cell RNA-seq data

Table 4.2: Comparison of runtime for alignment of the human brain single cell dataset using Rail-RNA and Falco 2.0 frameworks

<table>
<thead>
<tr>
<th>Nodes</th>
<th>Rail-RNA</th>
<th>Falco 2.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>15.9</td>
<td>5.9</td>
</tr>
<tr>
<td>40</td>
<td>5.7</td>
<td>2.6</td>
</tr>
</tbody>
</table>

4.3.2 Comparison of Falco’s alignment-only mode with Rail-RNA

As part of the evaluation of the alignment-only analysis using the Falco 2.0 framework, we also compared the performance of Falco 2.0 against Rail-RNA, a previously published tool designed for scalable alignment of RNA-seq data developed using the MapReduce programming paradigm. For the comparison, Rail-RNA was configured to output only BAM files in order to reduce the extra processing steps required for producing the default outputs of sample statistics, coverage vectors and junction information. It should be noted that the cluster used for running Rail-RNA utilises a different instance type compared to the cluster used for running Falco 2.0 (c3.8xlarge for Rail-RNA vs r3.8xlarge for Falco 2.0) as Rail-RNA only provides support for a limited number of instance types. To ensure a fair comparison, the instances used for Rail-RNA cluster have the same configuration for CPU, storage and network performance as the instance used for Falco 2.0 cluster, with the only difference being the memory configuration.

Rail-RNA was able to perform alignment of the human brain dataset in about 6 hours using a 40 node cluster, increasing to 16 hours using a 10 node cluster. In contrast, Falco 2.0 was able to perform alignment of the human brain dataset in less than 1 hour using a 40 node cluster and in about 2 hours using a 10 node cluster, representing a speed-up of around 10x compared to Rail-RNA (Table 4.2). The type of alignment file produced by Rail-RNA differs to that produced by the Falco 2.0 framework as Rail-RNA by default produces a single alignment file for each chromosome per sample, meaning that users will have to manually
4. Falco 2.0: An extension of the Falco framework for alignment and transcript assembly of single-cell RNA-seq data

combine the alignment files in order to get a single alignment file per sample. While RailRNA does provide an option to produce a single alignment file per sample, toggling this option resulted in Rail-RNA failing to complete during BAM writing step. The use of the MapReduce paradigm also means that Rail-RNA produces a lot more intermediate files compared to the Falco 2.0 framework, with Rail-RNA producing 2.4 TB of intermediate files for alignment of the 220 GB human brain dataset. In comparison, Falco 2.0 framework only produced a maximum of 200 GB of intermediate files (alignment chunks) for the alignment of the same dataset.

4.3.3 Evaluation and application of Falco’s transcript assembly mode

As with the alignment-only mode, we first checked if the output produced by Falco 2.0 alignment-only mode matches the output produced from single-node analysis. For this test, we evaluated three different pipeline configurations – STAR + StringTie with reference, STAR + Scallop and HISAT + StringTie without reference – using both simulated data and samples from human and mouse single-cell RNA-seq datasets. The simulated data is used to evaluate the performance of the pipelines tested in recovering transcripts from reference annotations, while the 100 randomly selected human and mouse single-cell RNA-seq datasets are used to evaluate the concordance between the assembled transcripts. Concordance evaluation between the output produced by Falco 2.0 and single-node analysis is performed by comparing the accuracy of the assembled transcript against the reference annotation as reported by the GffCompare tool. GffCompare measures accuracy of the assembled transcripts using two metrics - sensitivity, which is defined as the ratio between the number of correctly assembled transcripts and the total number of transcripts in the reference annotation; and precision, which is defined as the ratio between the number of correctly assembled transcripts and the total number of assembled transcripts. A transcript is determined by GffCompare as
4. Falco 2.0: An extension of the Falco framework for alignment and transcript assembly of single-cell RNA-seq data

correct if there is an 80% overlap for a single-exon transcript or if there is a transcript with a matching intron chain sequence in the reference annotation for a multi-exon transcript.

For the simulated dataset, Polyester \[126\] was used to generate a 100-bp paired-end human synthetic RNA-seq dataset, with 1000 reads samples for each gene with zero-error rate. In order to evaluate the ability of the pipelines to recover transcripts from the reference annotation, assembled transcripts prior to merging with reference annotation were used for comparison to the reference annotation with GffCompare. From the statistics of the transcript assembled from single node run (Table 4.3), it can be seen that reference-guided transcript assembly (STAR + StringTie with reference) has a high sensitivity and precision across all features. This is unlike the de-novo transcript assembly approaches (STAR + Scallop and HISAT + StringTie) which have high sensitivity and precision for base, exon, intron and locus, but very low precision on intron chain and transcript level. The low accuracy rate of intron chain and transcript features for the de-novo approaches can be explained by the limitations of the Polyester tool, which is unable to generate reads with the correct intron chain when using the reference annotation GTF file as input.
Table 4.3: Accuracy of assembled transcripts for simulated data from single node runs.

<table>
<thead>
<tr>
<th>Feature</th>
<th>STAR + StringTie (with reference)</th>
<th>STAR + Scallop</th>
<th>HISAT + StringTie</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Sensitivity (%)</td>
<td>Precision (%)</td>
<td>Sensitivity (%)</td>
</tr>
<tr>
<td>Base</td>
<td>97.3</td>
<td>99.8</td>
<td>87.7</td>
</tr>
<tr>
<td>Exon</td>
<td>97.3</td>
<td>98.4</td>
<td>57.0</td>
</tr>
<tr>
<td>Intron</td>
<td>96.8</td>
<td>99.3</td>
<td>70.7</td>
</tr>
<tr>
<td>Intron Chain</td>
<td>93.6</td>
<td>84</td>
<td>31.9</td>
</tr>
<tr>
<td>Transcript</td>
<td>94.1</td>
<td>85.7</td>
<td>33.9</td>
</tr>
<tr>
<td>Locus</td>
<td>98.3</td>
<td>99.4</td>
<td>71.9</td>
</tr>
</tbody>
</table>

Table 4.4: Accuracy of assembled transcripts for simulated data from Falco 2.0-based runs.

<table>
<thead>
<tr>
<th>Feature</th>
<th>STAR + StringTie (with reference)</th>
<th>STAR + Scallop</th>
<th>HISAT + StringTie</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Sensitivity (%)</td>
<td>Precision (%)</td>
<td>Sensitivity (%)</td>
</tr>
<tr>
<td>Base</td>
<td>96.2</td>
<td>99.9</td>
<td>88.6</td>
</tr>
<tr>
<td>Exon</td>
<td>96.4</td>
<td>97.9</td>
<td>59.6</td>
</tr>
<tr>
<td>Intron</td>
<td>95.5</td>
<td>99.3</td>
<td>74.5</td>
</tr>
<tr>
<td>Intron Chain</td>
<td>92.8</td>
<td>83.1</td>
<td>33.4</td>
</tr>
<tr>
<td>Transcript</td>
<td>93.3</td>
<td>84.9</td>
<td>35.2</td>
</tr>
<tr>
<td>Locus</td>
<td>98.1</td>
<td>99.3</td>
<td>72.5</td>
</tr>
</tbody>
</table>
4. Falco 2.0: An extension of the Falco framework for alignment and transcript assembly of single-cell RNA-seq data

Comparison of the statistics for transcripts produced by Falco 2.0 transcript assembly mode (Table 4.4) against single-node runs shows differences between the result of the transcript assembly processes, though the results do share a high degree of concordance. For the reference-guided transcript assembly pipeline, the transcripts assembled by the Falco 2.0 framework have lower sensitivity and precision compared to the single node runs due to the higher number of missed features. In contrast, the transcripts assembled using de novo transcript assembly pipelines on Falco 2.0 have a slightly higher sensitivity and precision for exon, intron and locus features, as there are fewer features missed and less novel features introduced. However, the result of de novo transcript assembly approaches also have a lower sensitivity and precision for intron chain and transcript features due to the presence of more assembled transcripts. The difference between the statistics for transcripts assembled using Falco 2.0 and single-node runs can likely be attributed to the binning approaches utilised by the transcript assembly step in Falco 2.0, which may result in partially assembled transcripts in cases where the transcripts spans multiple bins. As seen from the result of transcript assembly with Falco 2.0, this issue is more prevalent in the de novo transcript assembly approaches as there is no reference annotation present to repress the creation of partial transcripts.
Table 4.5: Precision of assembled transcripts for human brain single cell dataset across different transcript assembly approaches.

<table>
<thead>
<tr>
<th>Feature</th>
<th>STAR + StringTie (with reference)</th>
<th>STAR + Scallop</th>
<th>HISAT + StringTie</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Individual (%)</td>
<td>Pooled (%)</td>
<td>Falco 2.0 (%)</td>
</tr>
<tr>
<td>Base</td>
<td>42.7</td>
<td>63.0</td>
<td>41.8</td>
</tr>
<tr>
<td>Exon</td>
<td>76.5</td>
<td>92.3</td>
<td>79.0</td>
</tr>
<tr>
<td>Intron</td>
<td>88.0</td>
<td>96.9</td>
<td>92.3</td>
</tr>
<tr>
<td>Intron Chain</td>
<td>79.1</td>
<td>94.5</td>
<td>85.9</td>
</tr>
<tr>
<td>Transcript</td>
<td>57.5</td>
<td>83.0</td>
<td>60.2</td>
</tr>
<tr>
<td>Locus</td>
<td>32.2</td>
<td>61.5</td>
<td>32.4</td>
</tr>
</tbody>
</table>

Table 4.6: Precision of assembled transcripts for mouse embryonic stem cell single cell dataset across different transcript assembly approaches.

<table>
<thead>
<tr>
<th>Feature</th>
<th>STAR + StringTie (with reference)</th>
<th>STAR + Scallop</th>
<th>HISAT + StringTie</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Individual (%)</td>
<td>Pooled (%)</td>
<td>Falco 2.0 (%)</td>
</tr>
<tr>
<td>Base</td>
<td>52.8</td>
<td>90.3</td>
<td>55.0</td>
</tr>
<tr>
<td>Exon</td>
<td>60.8</td>
<td>95.9</td>
<td>78.3</td>
</tr>
<tr>
<td>Intron</td>
<td>77.0</td>
<td>96.9</td>
<td>88.0</td>
</tr>
<tr>
<td>Intron Chain</td>
<td>57.0</td>
<td>93.4</td>
<td>76.1</td>
</tr>
<tr>
<td>Transcript</td>
<td>48.1</td>
<td>90.9</td>
<td>58.8</td>
</tr>
<tr>
<td>Locus</td>
<td>40.4</td>
<td>86.1</td>
<td>41.7</td>
</tr>
</tbody>
</table>
4. Falco 2.0: An extension of the Falco framework for alignment and transcript assembly of single-cell RNA-seq data

To evaluate the performance of the transcript assembly mode on real scRNA-seq datasets, 100 samples were randomly selected from each of the human brain and mouse embryonic stem cell datasets, similar to the test performed during evaluation of the alignment-only mode. Since the datasets are composed of multiple samples, two different iterative transcript assembly approaches are utilised - transcript assembly on individual samples, followed by merging of all assembled transcripts (individual approach); and transcript assembly on pooled reads from all samples (pooled approach). While previous studies have shown that the pooled iterative transcript assembly approach will result in less discovery of novel transcripts from the samples [127], the issues associated with the nature of scRNA-seq necessitate the need to pool reads across samples in order to ensure full recovery of transcript. However, the individual iterative transcripts assembly approach may still be applicable for scRNA-seq data as the assembled transcripts from each sample will be merged together using StringTie merge mode, which is able to collapse transcripts within the same region into a representative transcript across the entire dataset. For this evaluation, GffCompare is run on transcript assembly after merging with the reference annotation, meaning that the sensitivity of the assembled transcripts will be identical across both iterative and Falco-based approaches as the reference transcripts will be present within the assembled transcript. As such, only the precision metric is usable for accuracy comparison as this metric is calculated based on the total number of transcripts assembled, which will differ across the different approaches.

Comparison of the precision metrics of the two iterative approaches shows differences in accuracy depending on the tool used for transcript assembly. StringTie is shown to perform better with the pooled iterative approach due to the lower number of transcripts assembled with the pooled approach, which is consistent with previous work. In contrast, Scallop performs better with the individual transcript assembly approach as it assembled more transcript in the pooled iterative approach (Table 4.5 and Table 4.6). This difference in the performance of the two transcript assembly tools can be attributed to the different transcript algorithms used by the tools and the different threshold used for transcript reconstruction.
4. Falco 2.0: An extension of the Falco framework for alignment and transcript assembly of single-cell RNA-seq data

When comparing the iterative approaches against Falco 2.0, it can be seen that the performance of Falco 2.0 transcript assembly mode falls in between the two iterative approaches, with StringTie based pipelines having more similarity to the individual iterative approach and the Scallop based pipeline having more similarity to the pooled iterative approach (Table 4.5 and (Table 4.6)). These results indicate that the transcripts assembled using Falco 2.0 are similar to those produced with the pooled approach from which the transcript assembly step is modelled after, though the binning approach utilised for parallelisation of the transcript assembly process does result in more transcripts being assembled.

As the final step in correctness evaluation, we performed validation of the transcripts assembled by Falco by visualising the assembled transcript and the read coverage from the 100 samples on the IGV genome browser [128]. Falco 2.0 transcript assembly mode is able to assemble a new transcript which is not present in the reference annotation, as shown in Figure 4.4 where a new transcript (MSTRG.66885.1) is found within the intron of the CREG2 gene in the human dataset. Furthermore, the presence of a new transcript (MSTRG.66887.1) next to a processed pseudogene (AC092570.1) may indicate that the reference annotation for the pseudogene may be inaccurate, especially as there are no reads that align to the reference transcript location. These results show the ability of the Falco 2.0 transcript assembly mode

Figure 4.4: Example of novel transcripts found within the CREG2 gene in the human brain single cell data. The two novel transcripts (highlighted in yellow) are supported by the read coverage data, which is obtained from combining the alignment records of the 100 randomly selected samples from the human brain data.
4. Falco 2.0: An extension of the Falco framework for alignment and transcript assembly of single-cell RNA-seq data

Table 4.7: Comparison of runtime for transcript assembly of single cell dataset with and without the Falco 2.0 framework

<table>
<thead>
<tr>
<th>System</th>
<th>Nodes</th>
<th>STAR + StringTie (with reference)</th>
<th>STAR + Scallop</th>
<th>HISAT + StringTie</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standalone</td>
<td>1 (1 process)</td>
<td>17.2</td>
<td>16.2</td>
<td>16.3</td>
</tr>
<tr>
<td></td>
<td>1 (12 processes)</td>
<td>4.2</td>
<td>5.5</td>
<td>5.7</td>
</tr>
<tr>
<td></td>
<td>1 (16 processes)</td>
<td>N/A</td>
<td>N/A</td>
<td>4.1</td>
</tr>
<tr>
<td>Falco 2.0</td>
<td>10 node</td>
<td>2.9</td>
<td>2.8</td>
<td>2.3</td>
</tr>
<tr>
<td></td>
<td>20 node</td>
<td>1.7</td>
<td>1.7</td>
<td>1.4</td>
</tr>
<tr>
<td></td>
<td>30 node</td>
<td>1.3</td>
<td>1.3</td>
<td>1.1</td>
</tr>
<tr>
<td></td>
<td>40 node</td>
<td>1.1</td>
<td>1.1</td>
<td>0.9</td>
</tr>
</tbody>
</table>

Standalone number of processes indicates the number of FASTQ file pairs that are processed in parallel. Timing for Falco includes initialisation and configuration time which are approximately 10 minutes. Runtime for STAR with 16 processes is not available as some STAR processes are killed by the operating system, resulting in failure of the job.

in recovering new transcripts from the underlying dataset.

4.3.4 Scalability of Falco’s transcript assembly mode

The performance evaluation of the transcript assembly mode is done in a similar manner to the performance evaluation of alignment-only mode, with runtime comparison of transcript assembly on a single node with varying degrees of parallelisation against the runtime of transcript assembly on the Falco 2.0 framework with differing cluster sizes. For the single node runs, the individual iterative transcript assembly approach was used for performing transcript assembly of the scRNA-seq datasets, though only the timings for the alignment and transcript assembly of individual samples are used for calculating the runtime for single node runs. As before, the runtime for the Falco 2.0 transcript assembly process includes both cluster initialisation time and FASTQ splitting steps.
4. Falco 2.0: An extension of the Falco framework for alignment and transcript assembly of single-cell RNA-seq data

Figure 4.5: Falco 2.0 transcript assembly processing time split by steps and stages for STAR + StringTie with reference, STAR + Scallop, and HISAT2 + StringTie pipelines in the analysis of human brain single cell data. The timings shown do not include cluster initialisation time, as it is constant across differing cluster sizes, or later stages of the transcript assembly step, as the total time taken for the remaining steps is < 30 seconds.
4. Falco 2.0: An extension of the Falco framework for alignment and transcript assembly of single-cell RNA-seq data

The runtime of transcript assembly for the human brain dataset on a single node ranges from 4.1 hours (HISAT + StringTie, 16 processes) up to 17.2 hours (STAR + StringTie with reference, 1 process). In comparison, the time taken for transcript assembly of the human dataset using the Falco 2.0 framework is at most 2.9 hours (STAR + StringTie with reference, 10 nodes), with a minimum time of 0.9 hours (HISAT + StringTie, 40 nodes) (Table 4.7). Comparison of runtimes between the same pipeline across the single node and Falco runs shows that Falco is able to achieve an average speed-up ranging from 1.7x (12/16 processes vs 10 nodes) up to 16.5x (1 process vs 40 nodes). Furthermore, comparison of runtime for Falco 2.0 across increasing cluster size also shows the scalability of the transcript assembly mode in Falco 2.0, with a maximum speedup of 1.7x when increasing the cluster size from 10 nodes to 20 nodes (Table 4.7). The reduction in speedup for clusters with > 20 nodes is again due to a number of factors, including the constant cluster start-up time and lack of speed-up in the FASTQ splitting step due to the uneven distribution of file sizes (Figure 4.5). There is also a limiting factor specific to the transcript assembly step as a result of an uneven distribution of bin sizes, with the largest bin in the human dataset having close to 20 million reads (Figure 4.6).

The performance analysis of Falco 2.0 transcript assembly mode does not include the timings for the mouse ESC dataset as this dataset fails to complete when run using the Falco 2.0 framework. There are two different types of issues that result in the failure of the transcript assembly step - memory (RAM) and serialisation. The memory issue was encountered during execution of transcript assembly using Scallop on the third largest bin in the mouse dataset, which contains around 11 million reads (Figure 4.6). Due to a large number of alignment records that need to be processed, Scallop uses up more memory beyond the assigned 30 GB of memory, resulting in the executor being killed as there is a strict limit imposed by the YARN resource manager on the memory consumption of executors. The serialisation issue, on the other hand, was encountered when performing transcript assembly using StringTie on the two largest bins in the mouse dataset, containing around 30 million reads each (4.6).
4. Falco 2.0: An extension of the Falco framework for alignment and transcript assembly of single-cell RNA-seq data

Figure 4.6: Distribution of read bin sizes for (A) human brain and (B) mouse embryonic stem cell single cell datasets.
4. Falco 2.0: An extension of the Falco framework for alignment and transcript assembly of single-cell RNA-seq data

and is a result of the limitation of the PySpark framework used for developing the Falco 2.0 framework. In both PySpark and Spark frameworks, the input and output from executors undergo a serialisation process to allow for data to be transferred to different executors and nodes. For the PySpark framework, the serialisation process is handled by the built-in Pickle library and there is a limitation in the size of data that can be serialised. The size of the data for the two largest bins exceeds the maximum limit of the data that can be serialised by the Pickle library, therefore resulting in an error during reading of the alignment records into the executor.

The two issues encountered during processing of the mouse dataset are primarily a result of the large input size of the dataset and the need to pool together reads for performing transcript assembly. One approach that can be used to process such a large dataset is to divide the samples into smaller batches and perform transcript assembly on these smaller batches, followed by merging of the assembled transcripts from each batch together to obtain a consensus set of transcripts across the full dataset. This approach is not ideal as the user needs to manually perform batching and merging of the assembled transcripts. Furthermore, the batching process may result in some transcripts being missed as there will only be partial information available during the transcript assembly process.

4.4 Discussion

The Falco 2.0 framework extends the capability of the existing Falco framework through the addition of two processing modes - alignment-only and transcript assembly modes. The alignment-only mode is designed to enable other types of RNA-seq analysis beyond gene expression analysis by producing read alignment output (in BAM format), which can then be used as input for further downstream analysis tools. As with the initial framework, the alignment-only mode on Falco 2.0 is configurable by the user and currently supports
4. Falco 2.0: An extension of the Falco framework for alignment and transcript assembly of single-cell RNA-seq data

STAR and HISAT2 as the alignment tool. From benchmarks using two single-cell RNA-seq datasets, we show that Falco 2.0 is able to speed-up the alignment process by 2.5 to 16.4 times compared to alignment using a single node with highly-optimised execution while still producing identical alignment output. Comparison of Falco 2.0 alignment only mode against a similarly designed tool, Rail-RNA, shows the capability of Falco 2.0 in performing alignment in a time-efficient manner, with Falco 2.0 achieving an average of 10 times speed-up compared to Rail-RNA. There are, however, some limitations with the alignment-only mode in terms of the scalability and storage due to limitations with the Spark framework used and the need to produce a single alignment file per sample so users will not need to merge the result of the alignment manually.

The addition of transcript assembly mode in the Falco 2.0 framework is designed to allow for the identification of novel transcripts by creating a more accurate gene annotation from the scRNA-seq dataset being analysed. Unlike in the alignment-only mode of Falco 2.0 and the gene expression analysis mode of Falco frameworks, the transcript assembly mode requires a different approach for parallelising transcript assembly as this process requires reads from the same region to be processed together. Using both simulated and real scRNA-seq datasets, we show that the binning-based approach used by Falco 2.0 transcript assembly mode is able to successfully perform transcript assembly, with a high concordance between the transcripts assembled with Falco 2.0 and transcripts assembled using two iterative transcript assembly approaches. Of the two single-cell RNA-seq datasets tested, Falco 2.0 transcript assembly was able to speed-up the analysis of the human brain dataset by a minimum of 1.7 times, up to a maximum of 16.5 times. Unfortunately, Falco 2.0 transcript assembly mode failed to perform transcript assembly for the larger mouse ESC dataset due to a number of factors, including high memory usage of the Scallop transcript assembly tool and a limitation with the amount of data that can be handled by the PySpark framework. One approach which can be used for processing such large datasets is to divide and process the data in smaller batches, though this will require the user to manually divide the data into batches and to
4. Falco 2.0: An extension of the Falco framework for alignment and transcript assembly of single-cell RNA-seq data

merge the transcript assembled from each batch. Regardless, we show that the Falco 2.0 transcript assembly mode is able to perform transcript assembly in a scalable manner and is also able to identify novel transcripts present within scRNA-seq datasets.
Chapter 5

Scavenger: A pipeline for recovery of unaligned reads utilising similarity with aligned reads
5. Scavenger: A pipeline for recovery of unaligned reads utilising similarity with aligned reads

The text and figures included in this chapter are adapted from the following publication:


Acknowledgement

The work in this chapter was co-authored by research student Joshua Tang, research assistant Michael Troup and my supervisor. Mr Tang and I jointly designed, developed and tested the pipeline. Mr Troup contributed in the development of the cloud-based framework for parallelising the pipeline. I am the sole author of all the text, figures and tables in this chapter.
5. Scavenger: A pipeline for recovery of unaligned reads utilising similarity with aligned reads

5.1 Overview

Read alignment is an important step in RNA-seq analysis as the result of alignment forms the basis for further downstream analyses. However, recent studies have shown that published alignment tools have variable mapping sensitivity and do not necessarily align reads which should have been aligned, a problem we termed as the false-negative non-alignment problem. We have developed Scavenger, a pipeline for recovering unaligned reads using a novel mechanism which utilises information from aligned reads. Scavenger performs recovery of unaligned reads by re-aligning unaligned reads against a putative location derived from aligned reads with sequence similarity against unaligned reads. We show that Scavenger can successfully recover unaligned reads in both simulated and real RNA-seq datasets, including single-cell RNA-seq data. The reads recovered contain more genetic variants compared to previously aligned reads, indicating that divergence between personal and reference genomes plays a role in the false-negative non-alignment problem. We also explored the impact of read recovery on downstream analyses, in particular gene expression analysis, and showed that Scavenger is able to both recover genes which were previously non-expressed and also increase gene expression, with lowly expressed genes having the most impact from the addition of recovered reads. We also found that the majority of genes with >1 fold change in expression after recovery are categorised as pseudogenes, indicating that pseudogene expression can be affected by the false-negative non-alignment problem. Scavenger helps to solve the false-negative non-alignment problem through recovery of unaligned reads using information from previously aligned reads.
5. Scavenger: A pipeline for recovery of unaligned reads utilising similarity with aligned reads

5.2 Introduction

Read alignment is the process of mapping high-throughput sequencing reads against a reference genome or transcriptome to identify the locations from which the reads originate. This step is typically one of the first steps in the analysis of RNA sequencing (RNA-seq) data prior to downstream analyses such as variant calling and gene expression analysis. There have been a number of published tools which have been developed to perform RNA-seq alignment, such as HISAT2 [41], STAR [40], Subread [42], CRAC [129], MapSplice2 [130] and GSNAP [131]. More recently, new alignment-free tools have been developed specifically for gene expression analysis which skips the alignment of reads to the reference and instead performs pseudoalignment. However, these alignment-free tools are only applicable to specific types of analyses and have limitations compared to traditional alignment methods [132]. The correctness of alignment programs are crucial to the accuracy of the downstream analyses. Unfortunately, previous studies have shown that while these tools have low false positive rates, they do not necessarily have low false negative rates [133, 134]. This means that while many of the reads were likely to be correctly aligned, there are still many incorrectly unaligned reads which should have been aligned. These incorrectly unaligned reads, or false negative non-alignments, adversely affect the accuracy of the alignment produced and can also affect the result of downstream analyses, such as variant calling, indel (insertion-deletion) detection and gene fusion detection [134].

There are a number of factors which contribute to the false negative non-alignment problem. One such factor is the type of algorithm utilised in the alignment tool. In order to efficiently perform alignment against a typically large reference genome in an acceptable amount of time, and to account for splicing events inherent in RNA-sequencing data, many alignment tools use heuristic-based matching of seed sequences generated from read sequences. Due to the typically short length of a seed sequence and the existence of repetitive regions within the genome, there may be multiple locations assigned to a given read which results in the
5. Scavenger: A pipeline for recovery of unaligned reads utilising similarity with aligned reads

alignment tool excluding the read due to ambiguity – a problem known as multi-mapping reads. Another factor which causes false negative non-alignment problems is the divergence between the reference genome and the personal genome of the organism being sequenced. The reference genome is typically constructed from a small number of samples and thus will only represent a limited degree of the organism’s diversity. Alignment of reads to the reference genome will thus be imperfect due to natural variation present in an individual organism. While alignment tools do take into account the variability between the reference genome and an individual’s genome by allowing for mismatches, insertions and deletions during alignment, they are unable to handle a substantial degree of genetic variation, such as hyper-edited sites, gene fusion and trans-splicing.

Correcting for a false negative non-alignment problem is much more difficult compared to correcting false positive reads. For false positive reads, there are a number of strategies which can be employed to help filter these type of reads, such as by removing lower quality alignments, removing reads with multiple alignment locations and re-aligning reads with a more specific alignment tool. Recovering false negative reads, on the other hand, is not as straightforward as it is not possible to identify their putative alignment region in the genome. One possible strategy for solving the false negative non-alignment problem is to tune the parameters used for alignment in order to maximise the number of reads aligned, such as by increasing the threshold for multi-mapping reads and/or increasing the number of mismatches allowed. However, this approach is limited as there is no ground truth in real data to help with optimisation, and increasing the number of reads aligned will also result in an increase in the number of false positive reads. Another strategy for solving the false-negative non-alignment problem is by incorporating variation information during alignment, in the form of utilising alternate loci sequences within the reference genome [135] or integration of a single nucleotide polymorphism database to the reference [11], to help minimise the effect of divergence of the personal genome compared to the reference genome. This approach is also limited as it requires existing variation information, which may not be
5. Scavenger: A pipeline for recovery of unaligned reads utilising similarity with aligned reads

available in non-model organisms.

We have recently applied the idea of Metamorphic Testing – a software testing technique designed for the situation where there is an absence of an oracle (a method to verify the correctness of any input) – for performing software testing on the STAR sequence aligner \[136\]. Metamorphic testing involves multiple executions of the program to be tested with differing inputs, constructed based on a set of relationships (Metamorphic relations - MR), and checking that the outputs produced satisfy the relationships \[137, 138\]. In our previous study \[136\], we developed an MR to test the realignability of previously aligned reads in the presence of irrelevant ‘control’ chromosomes constructed from previously unaligned reads. We discovered that a non-trivial amount of reads that were previously aligned to the reference genome were now aligned to the control chromosomes consisting of reads which were unable to be aligned to the reference. Further investigation indicated that some of the unaligned reads have high similarity to the aligned reads, indicating the possibility of these reads being false negative non-alignments.

In this chapter, we aim to tackle the problem of false-negative non-alignments by taking inspiration from our previous work on metamorphic testing. We have developed Scavenger, a pipeline designed to recover incorrectly unaligned reads by exploiting information from reads which are successfully aligned. We applied the Scavenger pipeline on a number of simulated and actual RNA-seq datasets, including both bulk (normal) and single-cell RNA-seq datasets, and demonstrated the ability of Scavenger in recovering unaligned reads from these datasets. We then analysed the impact of adding these recovered reads on downstream analyses, in particular gene expression analysis, and discovered that lowly expressed genes, in particular genes of the pseudogenes category, are more affected by the false-negative non-alignment problem. We also verified that the divergence between the personal genome and the reference genome is a contributing factor to the false-negative non-alignment problem and showed that Scavenger is able to recover reads which are unaligned due to higher degree of variability within the reads sequence.
5. Scavenger: A pipeline for recovery of unaligned reads utilising similarity with aligned reads

5.3 Methods

Scavenger is a python-based pipeline designed to recover unaligned reads by utilising information from aligned reads. The pipeline takes in sequencing reads in FASTQ format as the input, along with a reference genome sequence in FASTA format and a corresponding index for the alignment tool built using the reference genome. There are 4 main steps in the Scavenger pipeline - source execution of alignment tool, follow-up execution using aligned reads as input and unaligned reads as index, consensus filtering of follow-up execution result to obtain putative alignment location, and re-alignment of unaligned reads to the reference genome (Figure 5.1). The unaligned reads which are able to be successfully re-aligned back to the genome are then re-written back to the alignment result from the source execution.

5.3.1 Source execution

The first step of the Scavenger pipeline is the source execution where sequencing reads are aligned to the reference genome using a sequence alignment program. The alignment program used must satisfy the three properties which are required to validate the metamorphic relation underlying the read recovery pipeline - deterministic alignment, realignability of mapped reads, and non-realignability of unmapped reads. Currently, STAR is utilised for aligning RNA sequencing reads in the Scavenger pipeline as it has been previously evaluated as being a reliable general-purpose RNA-seq aligner, with good default performance, [133] as well as satisfying the three properties above [136]. The source execution step can be skipped if the user has previously performed alignment of sequencing reads by passing in the alignment file produced in either SAM or BAM format as input to the Scavenger pipeline.
5. Scavenger: A pipeline for recovery of unaligned reads utilizing similarity with aligned reads

Figure 5.1: Overview of the Scavenger pipeline. Scavenger first aligns sequencing reads against the reference genome using the STAR alignment tool in the source execution step. Scavenger then extracts both the aligned and unaligned reads from the source alignment result and creates a sequencing reads file based on the aligned reads and an artificial genome file containing chromosomes built using sequences of unaligned reads. The sequences of aligned reads are then aligned to the artificial genomes using the same alignment tool from the source execution (STAR) in the follow-up execution steps to find aligned reads which have similar sequences to unaligned reads. Next, consensus filtering is performed to select putative sites for re-alignment based on where the majority of aligned reads originate from in the reference genome. Finally, re-alignment is performed for unaligned reads which pass consensus filtering and the source alignment result is updated based on the result of re-alignment.
5. Scavenger: A pipeline for recovery of unaligned reads utilising similarity with aligned reads

5.3.2 Follow-up execution

In the follow-up execution step, both aligned and unaligned reads are first extracted from the alignment file produced during source execution. For reads which have been successfully and uniquely aligned, a sequencing reads file (in FASTQ format) is created using the reads’ sequence and qualities retrieved from the alignment records. In the case of reads which did not align to the reference genome, reads with identical sequences are first grouped together in order to minimise computational complexity and to reduce the potential location for alignment. The unique unaligned sequences are then extended with spacer sequences (sequence of N nucleotides) in order to form sequence bins of equal length and to ensure that aligned reads do not align between two unaligned sequences. These sequence bins are concatenated into artificial chromosomes and stored into a new temporary genome file. Depending on the alignment program utilised, a new index will then need to be created based on the temporary genome containing the artificial chromosomes prior to alignment. Finally, sequencing reads of previously aligned reads are aligned to the temporary genome containing unaligned read sequences using the alignment tool used in source execution. In the current Scavenger pipeline, STAR is again utilised in the follow-up execution with a number of extra parameters in order to disable spliced alignment to ensure that input reads only align to one unaligned read sequence and to remove the restriction of the number of locations (i.e. unaligned read sequence) that the input reads can align to in the temporary genome.

5.3.3 Consensus Filtering

The next step of the Scavenger pipeline is consensus filtering. Reads which align during the follow-up execution step are extracted from the alignment file produced from the previous step to obtain information regarding similarity between reads aligned during source execution and reads which did not align during source execution. Each unaligned sequence may have
5. Scavenger: A pipeline for recovery of unaligned reads utilising similarity with aligned reads

alignments to multiple aligned reads from the source execution. As these aligned reads may be aligned to different regions in the reference genome, consensus filtering is performed to select putative sites for re-alignment. For each unaligned sequence, intervals are created based on the reference genome location of previously aligned reads that align to the unaligned sequence. Overlapping intervals are then merged to form longer intervals to both reduce the number of putative sites and to increase the support for the interval to be selected as a putative site. An interval is considered as being a putative site if there is more than one read within the interval and the level of support for the interval (i.e. the number of aligned reads that fall within the interval) is greater than the consensus threshold, which is set to 60% of the number of previously aligned reads that align to the unaligned sequence by default. During this step, there is also an optional filtering criteria that can be utilised to remove unaligned sequences which likely originate from a low complexity region or tandem repeat region. The filtering method is based on the tandem repeat detection step used in the ROP tool [139], which uses MegaBLAST [140] to align reads against a repeat sequence database, such as RepBase [141].

5.3.4 Re-alignment

The final step is the re-alignment step where unaligned sequences which pass the filtering steps are re-aligned to the reference genome using the putative location obtained from reads aligned during source execution as a guide. For each unaligned sequence, the reference genome sequence around the putative location (extended 100 base pairs at both the start and the end of the putative location) is extracted and stored as the new genome for aligning the unaligned sequence. Alignment of the unaligned sequence is then performed against the new genome using either MegaBLAST or STAR depending on the putative location of the unaligned sequence originate from the unspliced alignment or from the spliced alignment during the source execution, respectively. MegaBLAST is utilised for unspliced alignment.
5. Scavenger: A pipeline for recovery of unaligned reads utilising similarity with aligned reads

due to its high sensitivity, though a strict parameter of 64% overlap and 85% query identity (which replicates the result of STAR alignment) is also utilised to reduce the false positive recovery of sequences. Unaligned sequences which are successfully and uniquely aligned back to the reference genome are then added back to the alignment file of the source execution by modifying the alignment records of previously unaligned reads whose sequence matches the recovered unaligned sequence.

5.3.5 Parallelising Scavenger

Both the consensus and re-alignment steps of the Scavenger pipeline are computationally expensive due to the potentially large number of unaligned reads to be processed. However, the processing of the inputs are independent to each other thus allowing for parallelisation of processing unaligned reads in order to reduce the overall runtime of the pipeline. Scavenger takes advantage of Python’s built-in multiprocessing library in order to parallelise the consensus and re-alignment steps across the available CPU cores of the machine.

To enhance the scalability of Scavenger, a framework has been provided to enable parallel computation of a read recovery session on cloud computing resources. Cloud computing enables convenient, on-demand network access to a shared pool of configurable computing resources [80]. Central to the model of cloud computing is the virtualisation of computing resources to enable sharing of pooled resources. These resources can be commissioned and decommissioned as the user requires. Scavenger has a framework that employs the resources offered by the cloud provider Amazon Web Services (AWS). The cloud provider enables the user, using their own account credentials, to create a number of computing "instances", which are the virtual machines upon which the user can perform their computational workload. In the case of AWS, such resources are termed "EC2 instances". An instance typically can be provisioned within minutes of the user request, and the user is charged by the hour. Some
5. Scavenger: A pipeline for recovery of unaligned reads utilising similarity with aligned reads

cloud providers, such as AWS, offer reduced price "spot" instances at a greatly reduced price, such that the user places a "bid" for a spot instance on the proviso that the instance will be terminated should the current market price for the instance exceed the initial bid price. To minimise the cost for users, Scavenger utilises AWS spot instances.

The cloud computing feature of Scavenger, after initial configuration on the user’s controlling computing resource, uses the AWS EC2 cloud instances to perform the various steps of read recovery, and also uses AWS cloud storage (S3) to store test data and results. The Scavenger cloud processing feature co-ordinates all interactions with the cloud resources, with logging information stored both locally and on the cloud. The user can elect to have a large job to be spread among a number of cloud instances, with Scavenger creating the instances and distributing the work load evenly amongst the instances. The cloud computing feature of Scavenger is optional, and the user can elect to use their own computing resources if desired.
Table 5.1: List of datasets used for Scavenger testing and evaluation. The datasets are divided into three sections: 1. Datasets from selected non-reference mouse strain, 2. Normal (bulk) RNA-seq dataset from either human or mouse, and 3. Single-cell RNA-seq dataset from mouse.

<table>
<thead>
<tr>
<th>Accession ID</th>
<th>Samples ID</th>
<th>Organism</th>
<th>Tissue/Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>SRP039411</td>
<td>SRR1182782 - SRR1182783</td>
<td>Mus Musculus</td>
<td>Liver</td>
</tr>
<tr>
<td>ERP000614</td>
<td>ERR032989; ERR032991; ERR032997; ERR032998; ERR033006; ERR033017 - ERR033019</td>
<td>Mus Musculus</td>
<td>Brain</td>
</tr>
<tr>
<td>SRP020636</td>
<td>SRR826292; SRR826299; SRR826308; SRR826315; SRR826340; SRR826356 - SRR826363</td>
<td>Mus Musculus</td>
<td>Liver</td>
</tr>
<tr>
<td>SRP068123</td>
<td>SRR3087147; SRR3087158; SRR3087171; SRR3087176</td>
<td>Mus Musculus</td>
<td>Hippocampus</td>
</tr>
<tr>
<td>SRP013610</td>
<td>SRR504764; SRR504766</td>
<td>Mus Musculus</td>
<td>Eye</td>
</tr>
<tr>
<td>SRP076218</td>
<td>SRR3641982; SRR3641983; SRR3641990; SRR3642003; SRR3642005; SRR3642012 - SRR3642014</td>
<td>Mus Musculus</td>
<td>Heart</td>
</tr>
<tr>
<td>SRP045630</td>
<td>SRR1554415; SRR1554417</td>
<td>Mus Musculus</td>
<td>Retina</td>
</tr>
<tr>
<td>SRP016501</td>
<td>SRR594393; SRR594401</td>
<td>Mus Musculus</td>
<td>Brain; Colon; Heart; Kidney; Liver; Lung; Skeletal Muscle; Spleen; Testes</td>
</tr>
<tr>
<td>SRP075605</td>
<td>SRR3578721; SRR3578725</td>
<td>Homo sapiens</td>
<td>Fibroblasts</td>
</tr>
<tr>
<td>SRP122535</td>
<td>SRR633739; SRR633734</td>
<td>Homo sapiens</td>
<td>ESC</td>
</tr>
<tr>
<td>SRP013027</td>
<td>SRR4422503; SRR4422506; SRR4422535; SRR4422538; SRR4422538; SRR4422626 - SRR4422629</td>
<td>Mus Musculus</td>
<td>Hindbrain; Limb; Heart</td>
</tr>
<tr>
<td>SRP045452</td>
<td>80 randomly selected samples</td>
<td>Mus Musculus</td>
<td>Hippocampus</td>
</tr>
</tbody>
</table>
5. Scavenger: A pipeline for recovery of unaligned reads utilising similarity with aligned reads

5.3.6 Datasets

Three different types of RNA-seq datasets – simulated, normal (bulk) and single-cell – were utilised to evaluate the Scavenger pipeline. The simulated datasets were obtained from a previous study [133] which generated 3 sets of simulated RNA-seq datasets from the hg19 reference genome using BEERS simulator [142] with varying parameters to emulate different level of dataset complexity. As the simulated datasets were formatted in FASTA format, high quality scores were added to each of the simulated reads to produce corresponding FASTQ files. These files were then input into Scavenger for both source alignment and read recovery with either STAR v2.5.3a or Subread v1.6.0 as the alignment tool. The GRCh37.p13 reference genome was obtained from GENCODE [143] and modified to contain reference chromosomes only, and used to create the indexes for each alignment tool. For STAR specifically, the annotation file was extracted from a previous study [133] and utilised in index creation to help increase the accuracy for alignment across splice junctions. In the evaluation of alignment results for simulated datasets, we used the analysis script that was used in the previous study [133] to analyse the correctness of the alignment results.

The normal and single-cell RNA-seq datasets were obtained from publicly available human and mouse datasets which were deposited to the NCBI Sequence Read Archive [144] (Table 5.1). Pre-processing of the datasets was performed using Trimmomatic v0.36 to remove low quality sequence and short reads. The pre-processed datasets were then analysed by Scavenger using STAR v2.5.3a as the alignment tool in the source execution and for re-alignment of spliced reads, together with BLAST v2.6.0 for re-alignment of unspliced reads. Indexes used for aligning of both human and mouse datasets were generated from GRCh38 and GRCm38 reference genomes respectively, which were obtained from GENCODE together with the corresponding annotation files (version 27 for human and version 15 for mouse). As before, annotation was used to augment the index to increase accuracy for alignment. The Repbase database [141] was also utilised to remove low complexity reads and reads
5. Scavenger: A pipeline for recovery of unaligned reads utilising similarity with aligned reads

from repetitive regions. For human datasets, the simple, humrep and humsub sequence files from Repbase were concatenated and used to create a BLAST database. Reads that passed consensus were aligned to this database and the aligned reads that have a minimum of 90% sequence identity and 80% sequence coverage were removed for further processing in Scavenger. A similar approach was used for the mouse datasets, but the simple and mousub sequence files were used instead. For mouse strain analysis, strain-specific VCF files for non-reference mouse strains containing SNPs derived against the reference C57BL/6J mouse genome were downloaded from the Mouse Genome Project (MGP) [145].

For running the alignment using STAR, the following command is used: `STAR --runThreadN <threads> <aligner_extra_args> --genomeDir <genome_index> --readFilesIn <read_files> --outFileNamePrefix <output_prefix>`. As for running the alignment using subread, the following command is used: `subread-align -T <threads> -t 0 <aligner_extra_args> -i <genome_index> <read_files> -o <output_file> <bam_option>`. And lastly, for running the alignment using BLAST, the follow command is used: `blastn -query unmapped_read -subject target_genome -task megablast -perc_identity <identity> -qcov_hsp_perc <coverage> -outfmt "17 SQ SR" -out <sam_output> -parse_deflines`. During follow-up alignment using STAR, the following parameters are additionally used: `--outFilterMultimapNmax <num_reads> --alignIntronMax 1 --seedSearchStartLmax 30`.

5.4 Results

5.4.1 Recovery of reads on simulated data

To evaluate the ability of the Scavenger pipeline to recover false-negative non-aligned reads, we first tested Scavenger using previously published human simulated data. The varying level of complexity of the simulated datasets represents the degree of divergence between the se-
quencing reads generated compared to the reference genome, ranging from low polymorphism and error rate (T1), moderate polymorphism and error rate (T2) and high polymorphism and error rate (T3). The results of the source execution of STAR with default parameters are consistent with the previously published result, with >99% of reads being aligned in both T1 and T2 and >90% of reads being aligned in T3 (Table 5.2). After running the Scavenger pipeline, we were able to recover between 4-30% of the previously unaligned reads in the three datasets, resulting in an increase of aligned reads ranging from ~1,500 to ~160,000. The majority of reads recovered by Scavenger are aligned in the correct position, with 79.4% of reads being correctly recovered in T1 and >98% of reads being correctly recovered in T2 and T3.

The difference in the number of aligned reads between the three datasets can be explained by the degree of divergence between the sequencing reads and the reference genome; and the limitation of the alignment tool in aligning reads which display a high degree of polymorphism. The simulated sequencing reads in both T1 and T2 have high homology to the reference genome due to the lower degree of polymorphism and error rate introduced meaning that the majority of these reads will be accurately mapped to the reference genome with a very small number of mismatches during alignment. In contrast, the sequencing reads in T3 – with the higher polymorphism and error rate – have a much higher degree of divergence compared to the reference genome thus resulting in more mismatches during alignment and therefore causing it to fail to be aligned. The Scavenger pipeline is able to recover more reads in T2 and T3 compared to T1 due to the greater number of aligned reads that contain mutations within the sequence. During follow up execution, Scavenger exploits the fact that these aligned reads will have closer similarity to the unaligned reads, which will also contain mutations, therefore resulting in the alignment of the aligned reads to the unaligned reads to obtain the putative location for the unaligned reads for recovery.
Table 5.2: Alignment statistics for simulated datasets before and after recovery of reads with Scavenger using default parameters for STAR.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Source execution</th>
<th>Scavenger pipeline</th>
<th>Unaligned reads recovered</th>
<th>% recovered reads correctly</th>
<th>% recovered reads incorrectly</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Aligned correctly</td>
<td>Aligned incorrectly</td>
<td>Unaligned correctly</td>
<td>Aligned correctly</td>
<td>Aligned incorrectly</td>
</tr>
<tr>
<td>T1</td>
<td>9,671,586</td>
<td>8,022</td>
<td>33,486</td>
<td>9,672,770</td>
<td>8,330</td>
</tr>
<tr>
<td>T2</td>
<td>9,617,585</td>
<td>17,163</td>
<td>56,827</td>
<td>9,634,469</td>
<td>17,496</td>
</tr>
<tr>
<td>T3</td>
<td>8,595,549</td>
<td>67,559</td>
<td>933,274</td>
<td>8,753,899</td>
<td>67,995</td>
</tr>
</tbody>
</table>

The result shown is an average from 3 samples.

Table 5.3: Alignment statistics for simulated datasets before and after recovery of reads with Scavenger using optimised parameters for STAR.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Source execution</th>
<th>Scavenger pipeline</th>
<th>Unaligned reads recovered</th>
<th>% recovered reads correctly</th>
<th>% recovered reads incorrectly</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Aligned correctly</td>
<td>Aligned incorrectly</td>
<td>Unaligned correctly</td>
<td>Aligned correctly</td>
<td>Aligned incorrectly</td>
</tr>
<tr>
<td>T1</td>
<td>9,673,309</td>
<td>6,861</td>
<td>33,680</td>
<td>9,673,362</td>
<td>6,948</td>
</tr>
<tr>
<td>T2</td>
<td>9,643,573</td>
<td>14,570</td>
<td>56,827</td>
<td>9,643,715</td>
<td>14,675</td>
</tr>
<tr>
<td>T3</td>
<td>9,437,748</td>
<td>75,395</td>
<td>933,274</td>
<td>9,445,855</td>
<td>75,448</td>
</tr>
</tbody>
</table>

The result shown is an average from 3 samples.
5. Scavenger: A pipeline for recovery of unaligned reads utilising similarity with aligned reads

Another method to solve the false-negative non-alignment problem is to adjust the parameters of the alignment tool utilised in order to allow alignment of reads with a higher degree of polymorphism. As has been shown previously, alignment of the simulated datasets using STAR with optimised parameters results in >99.2% of the reads being aligned, with T1 and T2 reaching nearly 99.9% of reads being aligned (Table 5.3). The Scavenger pipeline is unable to obtain the high degree of alignment achieved with parameter optimisation due to limitations in Scavenger’s approach to recover reads. Since Scavenger utilises information from aligned reads to find the putative location of unaligned reads for recovery, it is not possible to recover any unaligned reads from regions which have no read alignments. As such, the reads that the Scavenger pipeline is able to recover are reads from regions which already have alignment. This is unlike parameter optimisation, which allows for alignment with a higher threshold of mismatches in any region irrespective of whether there was alignment in the region. This observation can be seen in the high degree of overlap (>96.5%) of the reads recovered by the Scavenger pipeline compared to the reads recovered by optimised parameters. The Scavenger pipeline is still able to recover some reads which are unaligned with optimised parameters, particularly in T3 where Scavenger recovered ∼9.75% of previously unaligned reads. Unlike Scavenger recovery with default parameters, the majority of recovered reads after alignment with optimised parameters are incorrectly aligned in both the T1 and T2 datasets. Given the very high degree of alignment in these lower complexity datasets, it is likely that the unaligned reads are reads which can align to many locations in the genome and thus correctly recovering these reads is very difficult and error prone. These results indicate that parameter optimisation provides a solution to the false-negative non-alignment problem, performing better than Scavenger. However, given that performing parameter optimisation is not trivial due to lack of ground truth in real datasets, these results also show that Scavenger can be utilised as an alternative to help recover false-negative non-aligned reads.
Table 5.4: Unaligned reads identified by ROP in the simulated dataset.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Unaligned reads identified</th>
<th>Low Quality Reads</th>
<th>Low Complexity Reads</th>
<th>rRNA Reads</th>
<th>Lost Reads</th>
<th>Repeat reads</th>
<th>NCL Reads</th>
<th>Immune Reads</th>
<th>Microbial Reads</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>31,469</td>
<td>1</td>
<td>188</td>
<td>251</td>
<td>30,398</td>
<td>502</td>
<td>9</td>
<td>120</td>
<td>1</td>
</tr>
<tr>
<td>T2</td>
<td>27,328</td>
<td>0</td>
<td>306</td>
<td>148</td>
<td>23,508</td>
<td>1,690</td>
<td>28</td>
<td>1,639</td>
<td>9</td>
</tr>
<tr>
<td>T3</td>
<td>58,544</td>
<td>3</td>
<td>2,469</td>
<td>13</td>
<td>3,085</td>
<td>7,123</td>
<td>132</td>
<td>45,696</td>
<td>24</td>
</tr>
</tbody>
</table>

The result shown is an average from 3 samples.

Table 5.5: Alignment statistic for unaligned reads recovered by ROP lost read identification step.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Unaligned reads recovered</th>
<th>% recovered correct</th>
<th>% recovered read incorrect</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>29,614</td>
<td>5.1%</td>
<td>94.9%</td>
</tr>
<tr>
<td>T2</td>
<td>22,032</td>
<td>6.6%</td>
<td>93.4%</td>
</tr>
<tr>
<td>T3</td>
<td>2,986</td>
<td>7.4%</td>
<td>92.6%</td>
</tr>
</tbody>
</table>

The result shown is an average from 3 samples.
5. *Scavenger*: A pipeline for recovery of unaligned reads utilising similarity with aligned reads

We also performed a comparison of the Scavenger pipeline against a recently published tool, Read Origin Protocol (ROP), which is primarily designed to identify the origin of unaligned reads \[139\]. The ROP tool consists of 6 steps, with each step designed to identify different causes for unaligned reads: reads with low quality, lost human reads, reads from repeat sequences, non-collinear RNA reads, reads from V(D)J recombination and reads belonging to microbial communities. The result of running ROP on the simulated dataset shows that ROP is able to identify an average of $\sim29,000$ reads in the T1 and T2 datasets, and $\sim58,500$ reads in T3 dataset (Table 5.4). In particular, the majority of reads in the T1 and T2 dataset are correctly identified as lost human reads, while the majority of reads in T3 dataset are incorrectly identified as immune reads. Checking the correctness of ROP identified reads is not straightforward given that most steps within ROP does not produce alignment information. Thus, correctness testing was performed only on the genome-based alignment information produced during the lost reads steps. The result of the correctness testing shows that $>92.6\%$ of the reads identified by ROP are incorrectly aligned (Table 5.5).

5.4.2 Divergence of personal genome results in false-negative non-aligned reads

One factor which may affect the false-negative non-alignment problem is the divergence of sequences between the reference genome and personal genome which results in alignment tools being unable to properly align the reads due to the higher number of mismatches. To evaluate the ability of Scavenger in recovering these false-negative non-aligned reads which arise due to divergence of the personal genome, an experiment was devised where reads from non-reference inbred laboratory mouse strains were aligned to the reference C57BL/6J mouse genome to imitate alignment of reads from the personal genome against the reference genome. Multiple non-reference mouse strains – 129S1/SvImJ, A/J, CAST/EiJ, DBA/2J
5. Scavenger: A pipeline for recovery of unaligned reads utilising similarity with aligned reads

![Graph A](image)

**Figure 5.2**: The number of reads containing SNPs found within reads aligned in source execution and reads recovered by Scavenger. A. The number of reads with ≥ 1 SNPs found within reads. B. The number of reads with high number of SNPs (> 5) found within reads.
5. Scavenger: A pipeline for recovery of unaligned reads utilising similarity with aligned reads

and NOD/ShiLtJ – were utilised as the genomes of these strains have previously been characterised by the Mouse Genome Project (MGP), with variations from each strain identified relative to the reference mouse genome. We collected 80 publicly available RNA-seq samples from the selected mouse strains, with each strain having a minimum of 13 samples from at least 3 different projects with varying characteristics, and performed alignment of these samples against the reference genome using STAR with default parameters. The result of the source alignments shows that there is generally a high degree of mappability of the reads, ranging from 82.2% up to 98.1%. After recovery with Scavenger, we were able to re-align \(~4.75\%\) of unaligned reads in the source execution, corresponding to an increase in the number of aligned reads ranging from 17,000 to 396,000 reads (Table 5.6).

Further analysis was performed to evaluate the hypothesis that reads recovered by Scavenger have a higher degree of polymorphism due to the divergence between the 'personal' non-reference mouse strain genome against the reference genome. We randomly selected 1,000 unspliced reads which are aligned in the source execution and 1,000 unspliced reads recovered by Scavenger from each sample, and then calculated the number of single nucleotide polymorphisms (SNP) found within the location of the aligned reads from the list of strain-specific SNPs published by MGP against the reference mouse genome. The same analysis was then repeated a further 9 times, for a total of 10 iterations, to allow for significance testing. The majority of the reads which are either successfully aligned or recovered did not contain any of known SNPs. However, the number of reads which contain SNPs is significantly higher (p-value \(< 10^{-27}\)) in the reads recovered by Scavenger compared to the reads aligned in the source execution for 4 of the 5 strains analysed (Figure 5.2A). Furthermore, the number of reads with a high number of SNPs (\(> 5\)) are also significantly higher (p-value \(< 10^{-21}\)) in the reads recovered by Scavenger for all of the strains analysed indicating that Scavenger is able to recover reads which are more polymorphic compared to the reads aligned during the source execution (Figure 5.2B). These results validate the hypothesis that reads recovered by Scavenger have a higher degree of polymorphism as a result of the divergence
5. Scavenger: A pipeline for recovery of unaligned reads utilising similarity with aligned reads

Figure 5.3: Distribution of number of SNPs found within reads recovered during source alignment and reads recovered by Scavenger. Scavenger is able to recover reads with a higher number of SNPs (>5) compared to source alignment.

between the personal genome and the reference genome and further demonstrates the ability of Scavenger in dealing with the false-negative non-alignment problem.
Table 5.6: Alignment statistics for all RNA-seq datasets in source alignment with STAR and after recovery of reads with Scavenger.

<table>
<thead>
<tr>
<th>Accession ID</th>
<th>Read length</th>
<th>Total reads</th>
<th>Source aligned reads</th>
<th>Source unaligned reads</th>
<th>Source mapability</th>
<th>Scavenger aligned reads</th>
<th>Scavenger unaligned reads</th>
<th>Scavenger mapability</th>
<th>Scavenger recovered reads</th>
<th>Unaligned reads recovered (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SRP039411</td>
<td>97</td>
<td>47,077,051</td>
<td>44,052,9943,024,056</td>
<td>93.6%</td>
<td>44,162,052,2915,000</td>
<td>93.8%</td>
<td>109,057</td>
<td>3.61%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ERP000614</td>
<td>73</td>
<td>30,406,321</td>
<td>29,529,186877,136</td>
<td>97.1%</td>
<td>29,571,416834,905</td>
<td>97.3%</td>
<td>42,230</td>
<td>4.72%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SRP020636</td>
<td>93</td>
<td>10,695,056</td>
<td>10,023,96671,110</td>
<td>93.8%</td>
<td>10,053,119641,937</td>
<td>94%</td>
<td>29,173</td>
<td>4.43%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SRP068123</td>
<td>89</td>
<td>36,237,495</td>
<td>29,132,806714,689</td>
<td>82.2%</td>
<td>29,342,1656895,330</td>
<td>82.7%</td>
<td>209,360</td>
<td>2.9%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SRP013610</td>
<td>54</td>
<td>21,039,752</td>
<td>20,514,308525,444</td>
<td>97.5%</td>
<td>20,531,454508,298</td>
<td>97.6%</td>
<td>17,146</td>
<td>3.19%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SRP076218</td>
<td>86</td>
<td>20,183,248</td>
<td>19,802,286380,962</td>
<td>98.1%</td>
<td>19,822,443360,805</td>
<td>98.2%</td>
<td>20,157</td>
<td>5.49%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SRP045630</td>
<td>99</td>
<td>15,931,928</td>
<td>15,550,706381,221</td>
<td>97.6%</td>
<td>15,578,309353,618</td>
<td>97.8%</td>
<td>27,603</td>
<td>7.24%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SRP016501</td>
<td>48</td>
<td>85,677,826</td>
<td>82,218,772,459,055</td>
<td>96.2%</td>
<td>82,614,9843,062,842</td>
<td>96.6%</td>
<td>396,213</td>
<td>8.86%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SRP075605</td>
<td>51</td>
<td>30,851,404</td>
<td>29,278,7931,572,611</td>
<td>95%</td>
<td>29,356,2201,495,184</td>
<td>95.2%</td>
<td>77,427</td>
<td>5.26%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SRP125253</td>
<td>50</td>
<td>15,658,933</td>
<td>15,121,371537,562</td>
<td>96.6%</td>
<td>15,133,718525,214</td>
<td>96.7%</td>
<td>12,347</td>
<td>2.58%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SRP013027</td>
<td>100</td>
<td>28,031,517</td>
<td>26,043,7311,987,786</td>
<td>92.9%</td>
<td>26,092,5261,938,992</td>
<td>93.1%</td>
<td>48,794</td>
<td>2.49%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SRP045452</td>
<td>51</td>
<td>2,286,199</td>
<td>1,307,716 978,483</td>
<td>57.3%</td>
<td>1,313,084 973,116</td>
<td>57.5%</td>
<td>5,368</td>
<td>0.621%</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The result shown is an average of all samples per accession ID.
Figure 5.4: Gene expression in source alignment and after Scavenger recovery for genes whose reads are recovered. Coloured points indicates genes with expression difference of greater than 1 fold change.
5. Scavenger: A pipeline for recovery of unaligned reads utilising similarity with aligned reads

5.4.3 Effect of Scavenger recovery pipeline on downstream analysis

While alignment of reads is an important step in RNA-seq analysis, further downstream analyses are required in order to interpret the data into meaningful results. As one of the most common applications of RNA-seq analysis is gene expression analysis, we focused on identifying the effect of adding reads recovered by Scavenger on the expression of genes. The dataset utilised for testing consisted of 23 publicly available RNA-seq samples selected from 3 separate projects of varying characteristics, with 11 samples originating from two human projects and 12 samples originating from a single mouse project. The result of source execution using STAR with default parameters shows a high degree of mappability in all datasets, ranging from $\sim$95.9% in human datasets and $\sim$92.9% in the mouse dataset (Table 5.6). After recovery of reads with Scavenger, we were able to recover $\sim$3.1% of unaligned reads on average across the three datasets, corresponding to an increase ranging from 7,000 reads up to 102,000 reads. While the number of reads recovered are quite low relative to the number of previously aligned reads, the addition of tens and hundreds of thousands of reads is still likely to affect the expression of the genes.

Gene quantification of aligned reads is performed using featureCounts [48] to produce read counts per gene, which is then normalised to reads per million (RPM). In the source alignment, the number of genes expressed, defined as having non-zero read counts, in the human datasets average to 26,000 genes, while the number of genes expressed in the mouse dataset is 25,800 genes. In Scavenger recovered alignment, we see an increase of up to 3 expressed genes per sample, indicating the ability of Scavenger to recover genes which are falsely considered as non-expressed in the source alignment (Figure 5.5A). The recovery of reads in previously non-expressed genes is likely due to the extension of putative alignment locations, which may introduce regions which have no alignment in the source execution. Further investigation into the reads recovered by Scavenger shows that the reads are not distributed evenly across
5. Scavenger: A pipeline for recovery of unaligned reads utilising similarity with aligned reads

all the expressed genes – only ∼2150 and ∼5900 genes receiving an increase in read counts in human and mouse datasets, respectively. The majority of genes with increased read counts do not see much change in gene expression, with only ∼14 genes having more than 1 fold-change difference between source expression and recovered expression. Interestingly, genes which have substantial difference after recovery are generally genes with low expression in the source execution (log2(RPM) < 5), potentially indicating that some lowly expressed genes may actually have higher true expression than what is reported due to the alignment tool being unable to pick up these reads (Figure 5.4). This also has implications in further downstream analyses as lowly expressed genes are typically excluded from analysis, when instead it should not have been excluded as their true expression is actually higher.

We then performed further investigations into the genes with more than 1 fold-change difference after recovery to study the types of genes affected by the false-negative non-alignment problem. The majority of genes with recovered expression in the human and mouse dataset are classified as pseudogenes (>60%), with the second most frequent type being protein coding genes (22% and 9% for human and mouse dataset, respectively) (Figure 5.5B). Moreover, most genes with very low expression in the source alignment (log2(RPM+1) < 5) are in the pseudogenes category implying that many pseudogenes expression are likely to be under-reported due to reads originating from pseudogenes not being picked up by the alignment tool (Figure 5.4). Frequency analysis of the recovered genes also shows that some genes are consistently recovered across at least half of the samples in human and mouse datasets respectively, potentially indicating that these genes are harder to be picked up by the alignment tool due to its sequence being highly polymorphic. The finding that expression of pseudogenes are particularly affected by the false-negative non-alignment problem is significant as recent studies have shown that pseudogenes are incorrectly assumed to be non-functioning and actually have a role in regulating biological processes, particularly in diseases such as cancer [146, 147]. The reason that pseudogenes are more affected by Scavenger recovery is likely due to a number of factors, including the large number of mutations accumulated
5. Scavenger: A pipeline for recovery of unaligned reads utilising similarity with aligned reads

Figure 5.5: Effect of Scavenger read recovery on gene expression for normal (bulk) RNA-seq datasets. A. The number of genes whose reads are recovered by Scavenger, categorised based on the fold change in normalised expression (RPM) between source alignment and after Scavenger recovery. B. The number of genes with more than 1 fold change in normalised expression categorised based on their gene types.

which results in divergence between pseudogene sequences and personal genomes; and the typically low expression of pseudogenes and correspondingly, the number of reads from pseudogenes, which therefore are more affected by increase an in reads as a result of recovery by Scavenger (Figure 5.7).
Figure 5.6: Genome browser view of alignment in source execution and after recovery with Scavenger for FTH1P5 pseudogene from one human bulk RNA-seq dataset (SRR6337341). FTH1P5 gene was chosen as it is the only pseudogene consistently recovered in at least half of the human bulk RNA-seq dataset.
5. Scavenger: A pipeline for recovery of unaligned reads utilising similarity with aligned reads

![Graph showing read counts by gene type for human and mouse datasets.](image)

Figure 5.7: The number of reads recovered by Scavenger categorised by gene type for A. normal (bulk) RNA-seq datasets and B. single-cell RNA-seq datasets. In general, most reads are located in a region without a feature or within a protein coding gene. However, a high percentage of reads in human bulk RNA-seq datasets are located in other gene types, more specifically mitochondrial genes, due to the high source expression of these genes.

5.4.4 Applying Scavenger recovery on single-cell RNA-seq data

Single cell RNA-sequencing (scRNA-seq) is fast becoming a mainstream method for transcriptomics analysis due its ability to elucidate transcriptional heterogeneity of individual cells. However, there are a number of challenges when dealing with scRNA-seq datasets
5. Scavenger: A pipeline for recovery of unaligned reads utilising similarity with aligned reads

due to systematically low read counts, as a result of the small amount of transcripts which are captured during library preparation, and a high degree of technical noise [103]. Given Scavenger’s ability in recovering false-negative non-recovered reads in normal bulk RNA-seq datasets and the effect it has on downstream analyses, we hypothesise that recovery of unaligned reads in scRNA-seq datasets with Scavenger will likely have a greater impact on downstream analysis due to limited amount of reads available, while also helping with reducing technical noise. To test this hypothesis, 80 randomly selected samples were collected from a mouse brain scRNA-seq dataset and which are then aligned with STAR, followed by recovery of reads with Scavenger. The scRNA-seq samples have an average read depth of ∼2.3 million reads (after pre-processing), with ∼57.3% of the reads able to be aligned in the source execution (Table 5.6). Scavenger was only able to recover 0.6% of the unaligned reads, corresponding to an increase of ∼5,400 reads. The low number of reads which are able to be successfully recovered by the Scavenger pipeline is likely due to the low number of aligned in reads in source alignment, which provides less information that Scavenger can utilise during the follow-up execution.

As per the norm for scRNA-seq datasets, the number of genes with non-zero read counts is much lower compared to the number of non-expressed genes in bulk RNA-seq datasets, averaging 5,800. Of these expressed genes, only 12% of the genes (∼700) have an increase in read counts, with the majority of these genes having little difference in expression and ∼12 genes having a fold-change difference greater than 1 (Figure 5.8A). Unlike in bulk RNA-seq datasets, genes with substantial difference after recovery range from lowly expressed genes up to highly expressed genes, though genes with the greatest difference in expression are still those with low expression in the source alignment (Figure 5.4). Furthermore, a different pattern was also observed in the types of genes which have substantial difference in scRNA-seq datasets, with the protein coding category being the majority, followed by the pseudogene category (Figure 5.8B). The difference in pattern is likely due to comparatively higher abundance of protein coding genes and the low capture efficiency of scRNA-seq methods, meaning
5. Scavenger: A pipeline for recovery of unaligned reads utilising similarity with aligned reads

Figure 5.8: Effect of Scavenger read recovery on gene expression for single-cell RNA-seq datasets. A. The number of genes whose reads are recovered by Scavenger, categorised based on the fold change in normalised expression (RPM) between source alignment and after Scavenger recovery. B. The number of genes with more than 1 fold change in normalised expression categorised based on their gene types.
5. Scavenger: A pipeline for recovery of unaligned reads utilising similarity with aligned reads

that reads from pseudogenes are less likely to be captured and therefore rescued. This can be seen from the much lower number of pseudogenes expressed in scRNA-seq dataset (≈150) compared to bulk RNA-seq datasets (≈3,500).

5.5 Discussion

The false-negative non-alignment problem is a prevalent problem in many of the published RNA-seq alignment tools, resulting in loss of information from incorrectly unaligned reads. To help solve the false-negative non-alignment problem, we have developed Scavenger – a pipeline for recovery of unaligned reads using a novel mechanism based on sequence similarity between unaligned and aligned reads. Scavenger utilises the follow-up execution concept adapted from our previous work on metamorphic testing to find aligned reads from the source execution which have similar sequences to the unaligned reads by aligning the aligned reads against unaligned reads. The location of the aligned reads is then used as a guide to re-align the unaligned reads back to the reference genome using either BLAST or the original alignment tool depending on if the putative location originates from unspliced or spliced alignment, respectively, to ensure that splicing information is retained in recovered reads.

We have applied Scavenger on simulated datasets with varying degrees of complexity and showed that Scavenger is able to recover unaligned reads across all complexity levels with a reasonably high degree of accuracy. In particular, Scavenger is able to recover the most amount of reads in datasets that exhibit a high degree of complexity where read sequence is more divergent compared to the reference genome. We further show that although alignment of reads with optimised parameters are able produce a higher number of aligned reads compared to after recovery with Scavenger, the reads recovered by Scavenger have high degree of overlap to reads recovered with parameter optimisation. The lower number of reads
5. Scavenger: A pipeline for recovery of unaligned reads utilising similarity with aligned reads

recovered by after Scavenger is a result of Scavenger using information from aligned reads to find putative locations for unaligned reads, meaning that Scavenger is unable to recover reads from region with no alignment – unlike parameter optimisation which does not have the same limitation. Given the non-trivial difficulty of performing parameter optimisation on real datasets, we recommend the use of Scavenger as an alternative to help with recovering incorrectly unaligned reads.

There are a number of possible factors which may contribute to the false-negative non-alignment problem. One such factor is the divergence between the reference genome and the personal genome, leading to higher mismatches during alignment of sequenced reads against the reference genome. In order to validate that divergence of genomic sequences result in incorrectly unaligned reads, we devised an experiment whereby RNA-seq datasets from non-reference mouse strains were aligned against the reference mouse strain. We then analysed the reads which were aligned in the source execution against those recovered by Scavenger and showed that Scavenger is able to significantly recover more reads which have a higher number of reported strain-specific SNPs. This result both confirms that divergence of sequences between the reference genome and the personal genome does affect the false-negative non-alignment problems and that Scavenger is able to recover reads which are incorrectly unaligned due to a higher degree of sequence divergence.

As alignment of reads is only the first step in an RNA-seq data analysis, we also investigated the effect of the false-negative non-alignment problem on downstream analyses, in particular on gene expression analysis. After recovery of reads with Scavenger, we show that ~14 genes have more than 1 fold change in expression compared to the source alignment and that these genes are typically genes with low expression. Interestingly, the majority of genes with >1 expression difference belongs to the pseudogenes category, indicating that the expression of pseudogenes are likely to be under-reported due to reads from pseudogenes being incorrectly unaligned by the alignment tool.
5. Scavenger: A pipeline for recovery of unaligned reads utilising similarity with aligned reads

Given the ability of Scavenger to recover gene expression in normal (bulk) RNA-seq datasets, we then investigated the ability of Scavenger in recovering reads from scRNA-seq dataset as scRNA-seq datasets have the characteristics of having low reads counts and high degree of technical noise. Scavenger recovery affected the expression of 12% of the expressed genes, with ~12 genes having more than 1 fold change in expression. Unlike the bulk RNA-seq dataset, the genes with >1 change in expression range from lowly expressed genes up to highly expressed genes, with the genes belonging primarily to the protein coding category.

The current version of Scavenger supports STAR as the alignment tool for source execution and re-alignment of spliced reads. However, the user can choose to modify the alignment tool utilised by Scavenger with the alignment tool of their choice. Ideally the tool should satisfy the three properties underlying the read recovery pipeline – deterministic alignment, realignability of mapped reads, and non-realignability of unmapped reads – to ensure that the recovered reads are deterministic. To show the extensibility of Scavenger, we have tested Subread, another RNA-seq alignment tool, as a replacement for STAR within the Scavenger pipeline and demonstrated that Scavenger is still able to recover incorrectly unaligned reads with similar performance to STAR (Table 5.7 and 5.8). It should be noted that the recovery performance of Subread is different compared to STAR due to the different algorithm employed by Subread for alignment and, potentially, due to Subread violating the deterministic alignment property.
Table 5.7: Alignment statistics for simulated datasets before and after recovery of reads with Scavenger using default parameters for Subread.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Source execution</th>
<th>Scavenger pipeline</th>
<th>Unaligned reads recovered</th>
<th>% recovered correctly</th>
<th>% recovered incorrectly</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Aligned correctly</td>
<td>Aligned incorrectly</td>
<td>Unaligned correctly</td>
<td>Aligned correctly</td>
<td>Aligned incorrectly</td>
</tr>
<tr>
<td>T1</td>
<td>9,305,067</td>
<td>74,497</td>
<td>620,436</td>
<td>9,332,335</td>
<td>79,653</td>
</tr>
<tr>
<td>T2</td>
<td>8,985,799</td>
<td>87,576</td>
<td>926,625</td>
<td>9,107,130</td>
<td>126,051</td>
</tr>
<tr>
<td>T3</td>
<td>4,802,130</td>
<td>106,487</td>
<td>5,091,384</td>
<td>4,984,817</td>
<td>185,148</td>
</tr>
</tbody>
</table>

The result shown is an average from 3 samples.

Table 5.8: Alignment statistics for simulated datasets before and after recovery of reads with Scavenger using optimised parameters for Subread.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Source execution</th>
<th>Scavenger pipeline</th>
<th>Unaligned reads recovered</th>
<th>% recovered correctly</th>
<th>% recovered incorrectly</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Aligned correctly</td>
<td>Aligned incorrectly</td>
<td>Unaligned correctly</td>
<td>Aligned correctly</td>
<td>Aligned incorrectly</td>
</tr>
<tr>
<td>T1</td>
<td>9,416,480</td>
<td>262,926</td>
<td>320,594</td>
<td>9,419,057</td>
<td>264,906</td>
</tr>
<tr>
<td>T2</td>
<td>9,283,792</td>
<td>397,323</td>
<td>318,885</td>
<td>9,287,022</td>
<td>398,775</td>
</tr>
<tr>
<td>T3</td>
<td>7,111,603</td>
<td>2,251,068</td>
<td>637,330</td>
<td>7,122,864</td>
<td>2,251,625</td>
</tr>
</tbody>
</table>

The result shown is an average from 3 samples.
Chapter 6

starmap: Immersive visualisation of single cell data using smartphone-enabled virtual reality
6. starmap: Immersive visualisation of single cell data using smartphone-enabled virtual reality

The text and figures included in this chapter are adapted from the following publication:


Acknowledgement

The work in this chapter was co-authored by Honours student Yu Yao and Jianfu Li, and my supervisor. My supervisor and I conceived the application and have participated in its design and coordination. Mr Yao and Mr Li developed the application under our guidance and performed performance analysis. I am the sole author of all the text and figures in this chapter, with the exception of a figure describing the control scheme of the application.
6. starmap: Immersive visualisation of single cell data using smartphone-enabled virtual reality

6.1 Overview

Advances in single-cell RNA-seq technology, flow cytometry and mass cytometry have enabled the expression profiling of a large number of genes and proteins for hundreds of thousands of individual cells. However, current visualisation techniques do not allow for the effective display and understanding of the data due to the large number of points and use of non-immersive flat-screen visualisation. With the widespread availability of low-cost virtual reality (VR) devices, such as Google Cardboard, we propose the use of such devices as an immersive environment for visualising single-cell data in order to improve the navigation and exploration of the large numbers of cells. We have developed starmap, a VR program for visualising single-cell data designed to work with low-cost VR headsets. starmap offers a number of methods for interaction, such as wireless controller and voice control, and has a built-in star plot visualisation to allow users to explore features of the cells. starmap is freely accessible at https://vccri.github.io/starmap, with the corresponding source code available at https://github.com/VCCRI/starmap under the open source MIT license.

6.2 Motivation

A common method for visualisation of RNA-seq data, including single-cell RNA-seq data, is a scatter plot which shows the high level structure of the data. Since a scatter plot can only show up to three dimensions of data, dimensionality reduction techniques, such as Principal Component Analysis and t-SNE, are typically used to reduce the high dimensional gene expression data so that it can be visualised in the scatter plot. A scatter plot can also be used to show further information, such as cluster membership or expression of genes, through the use of colours. Another common method for visualisation of RNA-seq data is heatmap visualisation, which is typically used for visualising gene expression across samples.
6. starmap: Immersive visualisation of single cell data using smartphone-enabled virtual reality

The genes selected for heatmap visualisation are typically genes which are differentially expressed or genes which are known to be markers for certain cell type and/or conditions. Unlike scatter plot visualisation, heatmap visualisation is designed to provide a more granular view of the data - typically focusing on a smaller subset of genes and/or samples.

Current visualisation tools do not allow the viewer to easily see both the high level structure of the data and the more granular cell-level information. Furthermore, these visualisation methods do not necessarily scale for large single-cell data, especially in the case of a heatmap, and is limited in the amount of data that can be shown. In this chapter, we plan to design a tool for visualising large-scale single-cell data which can show the high level structure of the data while also allowing for the user to visualise cell-level information. We plan to also incorporate an immersive visual environment using low-cost VR headsets to allow the user to better navigate and explore the large amount of data points visualised.

6.3 Introduction

Advances in single-cell RNA-seq technology \cite{148, 149}, flow cytometry \cite{150} and mass cytometry \cite{151} have enabled the expression profiling of a large number of genes and proteins for hundreds of thousands of individual cells. This has opened up opportunities to explore cellular heterogeneity and has applications in almost all disciplines of biology and medicine. Dimensionality reduction methods, such as Principal Component Analysis (PCA) or other non-linear methods \cite{152}, can reduce the large number of features into lower dimensions to allow for efficient analysis of data. The data can then be visualised in two or three dimensions on a computer screen or on a printed page. However, the display and understanding of thousands of points are often challenging enough, let alone hundreds of thousands of points in the case of single cell data.
6. starmap: Immersive visualisation of single cell data using smartphone-enabled virtual reality

An effective visualisation should allow a viewer to not only assess the high-level clustering structure of cells, but also allow viewers to seamlessly ‘zoom’ into the data to discover fine local grouping of single cells, and to explore their gene/protein expression profiles. This type of visualisation is not easy to achieve using standard computer flat-screen visualisation.

Here, we develop a new smartphone-enabled virtual reality (VR) program, starmap, that enables immersive visualisation of single-cell data for hundreds of thousands of cells using a mobile-enabled web browser and low-cost VR head mount device. starmap breaks new ground on large scale data visualisation in two ways. First, it introduces a scalable visual design that combines the benefit of a three-dimensional (3D) scatter plot (Figure 6.1a) for exploring clustering structure and the benefit of star plots [153] (also known as radar chart) for multivariate visualisation of an individual cell (Figure 6.1b). Second, starmap is designed to utilise low-cost virtual reality (VR) headsets, such as Google Cardboard and Daydream, to enable widespread adoption of VR data visualisation. We reason that an immersive visual experience will likely improve the navigation and exploration of hundreds of thousands of cells.

6.4 Implementation

starmap, is designed to visualise single cell data primarily through a web browser on either a computer or a smartphone. starmap is built using web frameworks designed for creation of 3D and VR experiences, such as A-Frame and Three.js, which are cross-platform and can be adapted to computer screens and VR devices. Input data for starmap, in the form the 3D coordinates, cluster assignment, and value of up to 12 features per cell in a comma-separated values format, can be uploaded from a local file on your device or from third-party cloud storage, such as Google Drive, Microsoft OneDrive or iCloud Drive, depending on the support of the mobile smartphone’s operating system. starmap offers a number of options
6. starmap: Immersive visualisation of single cell data using smartphone-enabled virtual reality

Figure 6.1: starmap visualisation of a single-cell RNA-seq data containing 68,000 peripheral blood mononuclear cells (a) Overview of the 3D scatter plot with visualised in non-VR mode (b) Star plot visualisation showing the relative expression of gene modules for each cell in VR mode.
6. starmap: Immersive visualisation of single cell data using smartphone-enabled virtual reality

Table 6.1: Keyboard control scheme and voice control commands for starmap.

<table>
<thead>
<tr>
<th>Command</th>
<th>Keyboard Control</th>
<th>Voice Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>forward</td>
<td>w</td>
<td>forward</td>
</tr>
<tr>
<td>backward</td>
<td>s</td>
<td>backward</td>
</tr>
<tr>
<td>left</td>
<td>a</td>
<td>left</td>
</tr>
<tr>
<td>right</td>
<td>d</td>
<td>right</td>
</tr>
<tr>
<td>zoom in</td>
<td>q</td>
<td>in</td>
</tr>
<tr>
<td>zoom out</td>
<td>e</td>
<td>out</td>
</tr>
<tr>
<td>rotate Y-axis clockwise</td>
<td>left arrow</td>
<td>N\A</td>
</tr>
<tr>
<td>rotate Y-axis anti-clockwise</td>
<td>right arrow</td>
<td>rotate</td>
</tr>
<tr>
<td>rotate X-axis clockwise</td>
<td>up arrow</td>
<td>N\A</td>
</tr>
<tr>
<td>rotate X-axis anti-clockwise</td>
<td>down arrow</td>
<td>N\A</td>
</tr>
<tr>
<td>click on toolbox (VR mode)</td>
<td>N\A</td>
<td>select</td>
</tr>
<tr>
<td>reset toolbox (VR mode)</td>
<td>N\A</td>
<td>reset</td>
</tr>
<tr>
<td>reset view (VR mode)</td>
<td>N\A</td>
<td>init</td>
</tr>
</tbody>
</table>

for navigation through the VR space. Users are able to use a wireless keyboard or wireless hand-held remote controller in order to scale, rotate and navigate within the VR space (Table 6.1 and Figure 6.2). starmap also offers a voice control feature that allows user to navigate the VR space (Table 6.1), though this feature is available only in Google Chrome (desktop and mobile) as voice control utilises the SpeechRecognition API which is currently only supported by Google Chrome web browser.

For the star plot visualisation, starmap supports up to 12 radial coordinates which emanates from the centre to the circumference of the circle, with the coordinate indicating the expression level of a particular feature. The feature typically represents a biological feature such as a gene, in the case of RNA-seq data, or a protein, in the case of flow cytometry data. However, the feature can also represent a loading value for a principal component, the mean expression of a group of genes or proteins, or some other aggregate representation of related features.
6. starmap: Immersive visualisation of single cell data using smartphone-enabled virtual reality

![Remote control diagram]

Figure 6.2: Control mapping for remote control in starmap.

6.5 Performance evaluation

We evaluated the performance of starmap on two key metrics – data loading time and rendering performance. The dataset utilised for the performance testing is a synthetic dataset with varying number of points – ranging from 200,000 up to 1,500,000 – generated using Python programming language. The web browser utilised for testing in both mobile and desktop devices is Google Chrome.

In the loading time performance test, we evaluated the time taken by starmap to load a file from the local device, process the data and render the visualisation. We selected this metric as the size of a typical single-cell dataset is in the order of hundreds of thousands and thus it is important that starmap is able to quickly load and visualise the data. The result of the loading time performance test in both mobile and desktop devices shows that newer devices are able to load data faster than older devices, due to the increase in processing power (both
6. starmap: Immersive visualisation of single cell data using smartphone-enabled virtual reality

CPU and GPU) on newer devices (Figure 6.3 and 6.4). The result of the performance test also shows that desktop devices are able to load the data faster than mobile devices, with all desktop devices able to load the largest dataset containing 1,500,000 points in \( \leq 6 \) seconds.

For the rendering performance test, we evaluated the ability of the device to maintain a stable visualisation at 30 frame per seconds (FPS) using the statistics component of A-Frame.js to monitor the framerate rendered on screen. The result of the rendering performance test shows a similar trend as the previous test, with newer devices able to successfully achieve a framerate of 30 FPS due to better processing power on newer devices (Supplementary Tables 2 and 3). Furthermore, the result of the test shows that desktop devices are able to render larger datasets at a stable 30 FPS compared to mobile devices, due to the greater processing capability of desktop GPUs compared to mobile GPUs as starmap utilises GPU acceleration.

6.6 Conclusion

We have developed starmap, the first tool of its kind to enable visualisation of hundreds of thousands of single cells in a highly immersive virtual environment. starmap combines two visual paradigms - a 3D scatter plot with star plots - to enable users to both visualise the high level structure of their data and explore the cell-level information so that they can more effectively understand the large amount of information contained within their data. We show that starmap is able to handle large dataset sizes - ranging up to 1.5 million - making it a suitable tool for visualising large scale single-cell datasets. Two example datasets are provided, based on previously published scRNA-seq and flow cytometry data, to help users get started with using starmap. However, starmap is not limited to visualising only these types of data as the format of the input data to starmap is quite flexible and therefore can be adapted to display many different types of data.
6. *starmap*: Immersive visualisation of single cell data using smartphone-enabled virtual reality

![Figure 6.3: Loading and processing time of starmap with varying dataset sizes on mobile devices.](image)

![Figure 6.4: Loading and processing time of starmap with varying dataset sizes on desktop devices.](image)

121
6. starmap: Immersive visualisation of single cell data using smartphone-enabled virtual reality

Table 6.2: Rendering performance test on mobile devices with varying dataset sizes. Checkmarks indicates that the mobile device is able to render the dataset with a minimum of 30 frames per seconds.

<table>
<thead>
<tr>
<th>Mobile device (name, year)</th>
<th>Dataset Size</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>200,000</td>
</tr>
<tr>
<td>iPhone 6 (2014)</td>
<td>✓</td>
</tr>
<tr>
<td>iPhone 7 (2016)</td>
<td>✓</td>
</tr>
<tr>
<td>iPhone X (2017)</td>
<td>✓</td>
</tr>
<tr>
<td>Samsung S5 (2014)</td>
<td>✓</td>
</tr>
<tr>
<td>Samsung S7 (2016)</td>
<td>✓</td>
</tr>
<tr>
<td>Samsung S9 (2018)</td>
<td>✓</td>
</tr>
</tbody>
</table>

Table 6.3: Rendering performance test on mobile devices with varying dataset sizes. Checkmarks indicates that the mobile device is able to render the dataset with a minimum of 30 frames per seconds.

<table>
<thead>
<tr>
<th>Desktop device (name, GPU, year)</th>
<th>Dataset Size</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>300,000</td>
</tr>
<tr>
<td>Apple MacBook Pro</td>
<td>✓</td>
</tr>
<tr>
<td>(Intel Iris, 2014)</td>
<td></td>
</tr>
<tr>
<td>Apple MacBook Pro</td>
<td>✓</td>
</tr>
<tr>
<td>(Intel Iris Pro, 2015)</td>
<td></td>
</tr>
<tr>
<td>Apple MacBook Pro</td>
<td>✓</td>
</tr>
<tr>
<td>(Intel Iris Plus 640, 2017)</td>
<td></td>
</tr>
<tr>
<td>Dell Alienware 15 R2</td>
<td>✓</td>
</tr>
<tr>
<td>(Nvidia GTX 970M, 2015)</td>
<td></td>
</tr>
<tr>
<td>Dell Alienware 15 R3</td>
<td>✓</td>
</tr>
<tr>
<td>(Nvidia GTX 1070, 2016)</td>
<td></td>
</tr>
<tr>
<td>Dell Alienware 15 R4</td>
<td>✓</td>
</tr>
<tr>
<td>(Nvidia GTX 1070OC, 2017)</td>
<td></td>
</tr>
</tbody>
</table>

We have designed starmap using a highly performant web-based visualisation framework to enable cross-platform support across a wide range of devices, including smartphones, tablets and desktop computers. starmap is also designed to work with low cost VR devices, such as Google Cardboard and Daydream, to lower the barrier of entry for users to access the immersive visualisation program. While these low cost VR have less features and capabilities
6. starmap: Immersive visualisation of single cell data using smartphone-enabled virtual reality

compared to commercial VR devices, such as lack of positional tracking or specialised controllers, we have alleviated the problem of limited interaction capability by providing support for multiple input methods, including keyboard (wired and wireless), wireless remote control and voice control.

6.7 Materials and methods

6.7.1 Input data format for starmap

starmap accepts as input a csv file or a zip-compressed csv file. The csv file needs to contain a header row with the following column names - x, y, z and cluster - corresponding to the 3D coordinates of points and the cluster label assigned for each point (with outliers assigned the value of -1). In addition to the required columns, starmap also accepts extra columns (up to 12) corresponding to features which will be visualised in the star plot. The values for all columns must be of numeric types. Examples of the input data based on the single-cell RNA-seq data and flow cytometry data described below are available on starmap’s GitHub repository.

6.7.2 Processing of sample datasets

Single-cell RNA-seq data processing

We utilised the 68,000 peripheral blood mononuclear (PBMC) single-cell RNA-seq data from Zheng et al. [154] to showcase the ability of starmap to visualise the entire data set using a 3D t-SNE projection, grouped into 10 cluster, along with the expression of gene modules in
6. starmap: Immersive visualisation of single cell data using smartphone-enabled virtual reality

each single cell as a star plot. We obtained the gene expression profile for the PBMCs and the source code utilised previously for analysis of the PBMC data from 10XGenomics GitHub repository. In short, gene profiles were normalised according to the barcode (UMI) counts, followed by Principal Component Analysis on the top 1000 most variable genes. The PCA values were then used to perform t-SNE to produce 3D coordinates and k-means clustering to obtain 10 clusters. For gene module expression of individual clusters, we calculated the log2-transformed mean normalised expression of the genes specific to each cluster for all 10 clusters.

Flow cytometry data processing

We used a flow cytometry data set from cell populations of HIV-exposed uninfected and un-exposed infants generated by Aghaeepour et al. We combined the measurements from 10 selected samples to obtain a combined 4,000,000 cells measurements of 8 biomarkers. The data were then adjusted to truncate negative values to zeroes and hyperlog-transformed to obtain a normalised measurement. Clustering was then performed to group the cells into 3 clusters and Principal Component Analysis was used to reduce the dimension of the data for visualisation. Due to the large number of cells within the dataset, stratified sampling was used to downsample the data to 500,000 cells.
Chapter 7

Discussion
7. Discussion

Sequencing is now becoming an important tool in the analysis of biological systems. This is brought about by the increasing accessibility of sequencing technology and continually decreasing cost for sequencing as a result of technological advancement, therefore enabling the widespread use of sequencing for the study of genome and transcriptome. Recent advancement in single-cell capture technology has enabled the study of transcriptome at an unprecedented single-cell level, thus enabling better understanding of the transcriptional heterogeneity of cells and giving greater insight into the biological system.

The increasing adoption of sequencing technology has resulted in the generation of large amounts of sequencing data, meaning that there is an urgent need for the development of tools that can handle the analysis of these large volume of data. This thesis has contributed new computational methods and software for scalable analysis of next generation sequencing data, in particular single-cell RNA-seq data. This is achieved through the use of key innovative techniques, such as cloud-computing, big data frameworks, modern software testing techniques, and cutting-edge visualisation tool.

7.1 Towards a scalable analysis tool for large scale sequencing data

Analysis of sequencing data typically starts with alignment of reads against the genome or transcriptome to determine the location where the reads originate from. This process of read alignment is very complex and is typically one of the most time consuming tasks in analysing sequencing data. With the large amount of sequencing data generated, particularly single-cell RNA-sequencing data, there is a need for tools that can utilise distributed computing architectures, such as grid computing, cluster computing and cloud computing, in order to perform the processing of sequencing data in a distributed and scalable manner.
In this thesis, we have developed a scalable analysis framework for the processing of large scale single-cell RNA-seq datasets called Falco. The Falco framework utilises cloud computing technology to allow for easy access to scalable computing infrastructure, and Big Data frameworks of Apache Hadoop and Spark, which are designed for scalable analysis of large volumes of data. The Falco framework provides support for the most common type of scRNA-seq analysis of read quantification to obtain gene expression matrix for further downstream analysis. The subsequent Falco 2.0 framework adds further support for alignment-only analysis, to allow for other types of downstream analysis, and transcript reconstruction, to allow for identification and quantification of de-novo transcript isoform. Both the Falco and Falco 2.0 frameworks are able to achieve a speed up ranging from 1.7x to 145.4x when compared to a single-node processing with differing parallelisation strategies and shows scalability with increasing cluster size.

There is a limitation with the current Falco framework in that it only provides partial support for the analysis of sequencing reads produced by droplet-based single-cell capture technology. While Falco is able to perform alignment-only analysis for droplet-based single-cell data, it does not support read quantification or transcript reconstruction due to the barcode demultiplexing processing required for droplet-based data prior to read quantification and the limited transcript information captured by library preparation used in droplet-based techniques, respectively. Thus, further work is required in order to design a scalable analysis framework for processing droplet-based sequencing data, especially as this is becoming the most popular method for performing single-cell transcriptomics analysis.

While the Falco and Falco 2.0 frameworks are primarily designed for analysis of single-cell RNA-seq data, it also applicable for performing large-scale analysis of bulk RNA-seq data. Sequence database, such as Sequence Read Archive (SRA) and European Nucleotide Archive (ENA), contains many publicly available RNA-seq datasets representing a wealth of information, which can unlock greater insight when combined. While some databases do provide processed output, such as gene expression matrix, it can difficult to combine
7. Discussion

the results together due to differing processing methods which may produce incompatible outputs. There have been previous efforts to reprocess data from these sequence database in a uniform manner, such as Recount2 [156] and ARCHS4 [157], though the output is limited to gene expression data. Falco and Falco 2.0 frameworks will allow for reprocessing of large bulk RNA-seq datasets using user-configurable pipelines with support for alignment and transcript assembly analyses in addition to gene expression analysis.

7.2 Towards a more sensitive alignment tool

As previously mentioned, read alignment is one of the main steps in the analysis of sequencing data. Therefore, the accuracy of further downstream analysis is highly dependent on the correctness of the read alignment process. Recent studies have indicated that many published read alignment tool for RNA-seq have high accuracy, in that aligned reads are generally aligned in the correct location [133]. However, these alignment tools do not necessarily have high sensitivity as reads which should be aligned are incorrectly missed by the alignment tools. This problem of false-negative non-alignment can be attributed to a number of factors, such as the heuristic-based algorithm in the alignment process and the divergence between the personal and reference genome.

As part of this thesis, we have developed Scavenger, a pipeline designed for recovery of unaligned reads by utilising similarity information from aligned reads. Scavenger utilises the follow-up execution concept from our previous work on metamorphic testing of RNA-seq alignment tools in order to find aligned reads that have similar sequence to unaligned reads through re-alignment of aligned reads against unaligned reads. Testing on simulated dataset with varying degrees of complexity shows the ability of Scavenger to recover previously unaligned reads with a reasonably high degree of accuracy, especially on dataset with higher complexity. We also validated how divergence of personal and reference genome resulted in
7. Discussion

incorrectly unaligned reads through analysis of SNP found in reads recovered by Scavenger. Finally, we show that Scavenger is able to increase the expression of some lowly expressed genes in source execution through the recovery of unaligned reads.

Scavenger currently utilises a stringent criteria for recovery of reads to ensure a high false-positive rate for recovered reads, though this does result in a lower number of recovered reads. One such criteria is the removal of reads which have multiple alignment location in original execution due to the complexity of determining the true location of these multi-mapping reads. It will be useful to incorporate multi-mapping reads as part of the read recovery process as these reads have been shown to align to the true location [133] and can therefore help with recovering more unaligned reads. Another approach which may help with increasing the number of reads recovered is to integrate a more sophisticated method for determining putative location for read recovery that takes into account the alignment statistic and sequence similarity between the previously aligned and unaligned reads. Improving the recovery rate of Scavenger will help with further downstream analysis, particularly in pseudogene analysis as Scavenger has been shown to help with recovering the expression of lowly expressed pseudogenes for bulk RNA-seq dataset.

Solving the false-negative non-alignment problem is not trivial. As previous study has shown, it possible to reduce the number of incorrectly unaligned reads by carefully optimising the parameters of the alignment tool. This optimisation approach is somewhat limited as it requires ground truth information, which is not available in real datasets. Another approach, which is now being adopted by a number of alignment tools, is to incorporate variation information during the alignment process to minimise the effect of divergence of the personal genome compared to the reference genome. This approach also suffers from the limitation of requiring prior knowledge - in this case variation information - which may not necessarily be available in non-model organisms. Scavenger does not require prior knowledge for recovery of unaligned reads and thus it is well suited for recovering reads in cases where no prior knowledge is available and/or in addition to previous approaches.
7. Discussion

7.3 Towards a scalable visualisation tool for large scale data

Data visualisation is one of the key steps in downstream analysis of sequencing data and is typically done for two major reasons – exploration of data to gain a better understanding of the data, and presentation of data to convey information. There are a number of data visualisation techniques which are utilised for visualising the result of sequencing data analysis, such as a scatter plot to show the clustering of samples. However, not all techniques are suitable for visualisation of large-scale single-cell data due to the limited amount of information which can be conveyed, such as in the case of heatmap visualisation. Moreover, the majority of visualisations are produced as a static display for a flat-screen visualisation, which also introduces limitations in displaying the large amounts of information contained in single-cell data.

To tackle the problem of visualising large-scale data, we have developed starmap - an immersive visualisation tool designed for single-cell data. starmap allows for interactive visualisation of both the high-level structure of the data with 3D scatter plot visualisation and cell-level information using star plot (radar chart) visualisation to allow for better comprehension of the data. starmap is designed to work on a variety of devices, such as computer, tablet and mobile devices, and features an immersive virtual reality mode that works with low cost VR devices, such as Google Cardboard and Daydream, to allow the user to better explore and navigate through the large amount of data. While starmap is primarily designed for visualisation of single-cell RNA-seq data, it is also able to visualise other types of data, such as flow cytometry data.

The current version of starmap is more suited for the individual exploration of data. However, data analysis is typically a collaborative effort and therefore it will advantageous to be able to perform exploration the data together in an immersive manner. This will require
7. Discussion

the addition of multi-user sessions which allow multiple users to simultaneously view and interact with the data. For data presentation, starmap currently does not support native sharing of visualisation, though users can share the data file with each other to generate the same visualisation. The addition of file storage and sharing into starmap will allow for ease in sharing of visualisation for both data exploration and presentation of information.

7.4 Concluding remarks

The use of sequencing for studying biological problem is fast becoming ubiquitous due to the wealth of information that can be gleaned from sequencing data. Along with new technological advancements, such as single-cell capture technology, sequencing will enable the study of biological system at an unprecedented level. With this, however, comes the challenge of processing the large amount of data generated as current tools are not designed to handle such large datasets in an efficient manner. This thesis has contributed to solving this challenge through development of tools for the analysis of large amount of sequencing data, in particular single-cell RNA-seq data. We hope this will encourage development of more tools for analysing large sequencing data.
Bibliography


Discussion


Discussion


Discussion


Discussion


Discussion


Discussion


[67] Health Information Privacy. URL: https://www.hhs.gov/hipaa/
Discussion


[82] *EGI — Cloud Compute*. URL: [https://www.egi.eu/services/cloud-compute/](https://www.egi.eu/services/cloud-compute/)


Discussion


[93] DNAnexus. URL: [https://www.dnanexus.com/](https://www.dnanexus.com/)


Discussion

[97] Ben Langmead, KD Hansen, and JT Leek. “Cloud-scale RNA-sequencing differential expression analysis with Myrna”. In: *Genome Biology* Figure 1 (2010), pp. 1–11.


Discussion


[118] Fran Supek, Belén Miñana, Juan Valcárcel, Toni Gabaldón, and Ben Lehner. “Synonymous Mutations Frequently Act as Driver Mutations in Human Cancers”. In: Cell 156.6 (2014), pp. 1324–1335.

Discussion


143
Discussion


Discussion


Discussion


[149] Xiaoping Han, Renying Wang, Yincong Zhou, Lijiang Fei, Huiyu Sun, Shujing Lai, Assieh Saadatpour, Ziming Zhou, Haide Chen, Fang Ye, Daosheng Huang, Yang Xu, Wentao Huang, Mengmeng Jiang, Xinyi Jiang, Jie Mao, Yao Chen, Chenyu Lu, Jin Xie, Qun Fang, Yibin Wang, Rui Yue, Tiefeng Li, He Huang, Stuart H. Orkin, Guo-Cheng Yuan, Ming Chen, and Guoji Guo. “Mapping the Mouse Cell Atlas by Microwell-Seq”. In: *Cell* 172.5 (2018), 1091–1107.e17.


Discussion

